

SISSA

Scuola
Internazionale
Superiore di
Studi Avanzati

Mathematics Area - PhD course in
Mathematical Analysis, Modelling, and Applications

Data-Driven Reduced Order Modelling and Transfer Learning for the Structural Optimization of Cruise Ships

Advisor:

Prof. Gianluigi Rozza

Candidate:

Lorenzo Fabris

Co-advisor:

Asst. Prof. Marco Tezzele

Industrial advisors:

Ing. Ciro Busiello

Ing. Mauro Sicchiero

Academic Year 2025-2026



FUNDING

This thesis has been developed in the context of the "Development of advanced reduced methods of parametric optimization for the structural analysis of cruise vessels" project, co-financed by Fincantieri S.p.A. and the NextGenerationEU. The project is part of the Italian Piano Nazionale di Ripresa e Resilienza (PNRR), Missione 4, Componente 2 "Dalla Ricerca all'Impresa" - Investimento 3.3 "Introduzione di dottorati innovativi che rispondono ai fabbisogni di innovazione delle imprese e promuovono l'assunzione dei ricercatori dalle imprese", CUP G41J22000270008.

ABSTRACT

In this work we propose an optimization pipeline for the reduction of the structural mass of passenger cruise ships in the preliminary design phase. In simulation-driven design, the large cost of high-fidelity evaluations can be mitigated by leveraging data-driven reduced order models, exploring the design space via multi- and single-objective optimization of the surrogates and only validating the most promising candidates.

To tackle the complexity in the formulation of effective optimization problems, we propose a reparameterization procedure that adapts the decision variables to the emerging structural behavior of the ship evaluated by the surrogates. Starting with a coarse problem and iterating optimization and refinement, we show that the final configuration outperforms parameterizations of similar complexity produced by expert designers.

We then use a transfer-learning approach to design a one-shot reparameterization procedure, which leverages the results from past optimizations to refine the problem without the need for surrogates. To enhance the optimization via metaheuristic algorithms, we propose a population transfer procedure that accelerates the exploration of promising areas of the design space, leveraging the data from past optimizations.

The resulting pipeline is extensively evaluated on multiple test cases of industrial interest, both on simplified midship sections and full-scale ships.

Finally, to facilitate the introduction of life-cycle quantities in the design phase, we develop an open-source library for the implementation and evaluation of digital twins based on probabilistic graphical models, and showcase its workflow on a simplified structural health monitoring application.

This work has been conducted within the framework of the SHOPMEPA project, an industrial Ph.D. grant co-financed by Fincantieri S.p.A. and the NextGenerationEU initiative.

CONTENTS

Abstract	v
Contents	vii
List of Figures	xi
List of Tables	xix
List of Algorithms	xxi
Acronyms	xxiii
1 Introduction	1
2 Problem definition	5
2.1 Structural problem formulation	6
2.2 Notation	12
2.3 Numerical test cases	13
2.3.1 Midship section variants	14
2.3.2 Full ship fleet	16
2.4 Conclusions	19
3 Surrogates construction	21
3.1 Introduction	22
3.2 Proper orthogonal decomposition	23
3.3 Gaussian process regression	24
3.4 Multi-fidelity regression	27
3.5 Full surrogate	28
3.6 Numerical results	29
3.6.1 Midship section MS00	29
3.6.2 Full ship FS6	34
3.7 Conclusions	38
4 Surrogate-assisted optimization	39
4.1 Introduction	40

4.2	Multi-objective optimization	42
4.2.1	Non-dominated sorting genetic algorithm-III	43
4.2.2	Infill	45
4.3	Single-objective optimization	47
4.3.1	Principal directions search	49
4.3.2	Bayesian optimization	50
4.3.3	Particle swarm optimization	52
4.4	Numerical results	55
4.4.1	Midship section MS00	55
4.4.2	Full ship FS6	56
4.5	Conclusions	61
5	Response-based reparameterization	63
5.1	Introduction	64
5.2	Reparameterization	66
5.3	Numerical results	69
5.3.1	Midship section MS00	70
5.3.2	Full ship FS6	75
5.4	Conclusions	79
6	Transfer learning-based reparameterization	81
6.1	Introduction	82
6.2	Reparameterization	83
6.3	Numerical results	86
6.3.1	Midship section variants	86
6.3.2	Full ship fleet	89
6.4	Conclusions	94
7	Population transfer	95
7.1	Introduction	96
7.2	Transfer procedure	97
7.3	Numerical results	104
7.3.1	Midship section variants	104
7.3.2	Full ship fleet	110
7.4	Conclusions	113
8	Structural digital twins	115
8.1	Introduction	116
8.2	Probabilistic graphical models	118
8.3	pgmtwin	120
8.4	Numerical results	122
8.4.1	Structural health monitoring of a cantilever beam	122

8.5	Conclusions	129
9	Conclusions	131
9.1	Future perspectives	132
	Bibliography	135
	Appendices	
A	Database of optimized ships	157
A.1	Midship section variants	158
A.2	Full ship fleet	159

LIST OF FIGURES

1.1	Pipeline for the structural optimization of cruise ships. The response-based reparameterization refines the problem leveraging the surrogates, while transfer learning extracts knowledge from past optimizations to enable a one-shot reparameterization before the initial sampling and the transfer of populations to accelerate the exploration of the domain.	3
2.1	Simulations of a ship being deformed under a hogging load condition on the left, and sagging on the right. Displacements are magnified, with colors representing the von Mises stress from blue (zero) to the maximum value. .	6
2.2	Comparison of the shear Q and bending moment M , for the hogging and sagging load conditions. The mandated values are represented with dashed lines, while the actual loads for the FEA use solid lines. For confidentiality reasons, numerical values on the vertical axis are omitted. Negative abscissae correspond to coordinates behind the stern. Shear and bending moment values are scaled independently to enable qualitative comparison.	7
2.3	The shear and bending moment for the additional loads due to factoring correction are represented in the top plot. For confidentiality reasons, numerical values on the vertical axis are omitted. Shear and bending moment values are scaled independently to enable qualitative comparison. Negative abscissae correspond to coordinates behind the stern. The bottom plot shows the resulting coefficients for the post-processing of the hogging and sagging FEA results.	8
2.4	Yielding and buckling under hogging and sagging load conditions. Elements colored in red fail the structural stability tests mandated by the regulatory entities.	9
2.5	Trade-offs between different QoIs. Axes are muted due to confidentiality of data.	11
2.6	Computational mesh of a full ship on the left, the patches subdivision at the center, and a coarse parameterization on the right. Dark blue patches and parameterized elements correspond to elements with fixed thickness. . . .	11

2.7	Full view of the midship section on the left, the division into patches in the middle, and the initial parameterized sections on the right.	14
2.8	Von Mises criterion for the midship section under hogging load condition, on the left, and sagging on the right.	14
2.9	Three midship section variants models, depicted in light blue, and the default model represented with transparency. On the left, MS01 with the top deck removed. At the center, MS02 with one extra deck. On the right, MS15 with a shorter length, wider bottom section and two decks removed.	15
2.10	The full ship models, with their initial parameterizations. Each color corresponds to a different parameterized section, except for dark blue which represents elements with fixed thickness. The models are not to scale. . . .	17
3.1	Representation of a GPR prediction before and after its update. The training samples are shown as black dots, with the sample in red absent from the training in the top figure, and included in the bottom one. The true function is represented with a dashed black line, while the GPR predictions are in red. The confidence level corresponding to σ , 2σ , 3σ are represented in increasingly lighter shades of blue.	26
3.2	Representation of NARGPAS on an automobile simulation. Image courtesy of the authors of [159].	28
3.3	Side view of the midship section MS00, depicting the most energetic POD modes from top to bottom, for σ_x (left) and τ_{xz} (right) under hogging. The color scale is symmetric, with blue for negative values and red for positive values.	30
3.4	Evolution of the range of normalized singular values decay of the stress tensor components. The values come from the high-fidelity database before and after optimization, for the midship section MS00 with 5 parameterized sections.	30
3.5	Comparison of the 5-CV errors on n_y and n_b , for different database sizes and ranks. The predictions are relative to MS00 with 5 parameterized sections. The plots highlight the median error and the gray box identifies the chosen truncation rank.	31
3.6	Comparison of the 5-CV errors on the stress tensor components and their post-processed results, with medians highlighted, for different predictors of reduced coefficients. The values are relative to the hogging condition, simulated on MS00 with 5 parameterized sections.	32
3.7	Comparison of the 5-CV errors on the stress tensor components and their post-process results, with medians highlighted, for different predictors of reduced coefficients. The values are relative to the hogging condition, simulated on MS00 with 17 parameterized sections.	33
3.8	Comparison of the 5-CV errors on n_y and n_b , with medians highlighted, for different prediction approaches. The values are relative to MS00, with 5 and 17 parameterized sections.	33

3.9	Side view of the full ship FS6, depicting the most energetic POD modes from top to bottom, for σ_x (left) and τ_{xz} (right) under hogging. The color scale is symmetric, with blue for negative values and red for positive values.	34
3.10	Evolution of the range of normalized singular values decay of the stress tensor components. The values come from the high-fidelity database before and after optimization, for the full ship FS6 with 10 parameterized sections.	35
3.11	Comparison of the 5-CV errors on n_y and n_b , for different database sizes and ranks. The predictions are relative to FS6 with 10 parameterized sections. The plots highlight the median error and the gray box identifies the chosen truncation rank.	35
3.12	Comparison of the 5-CV errors on the stress tensor components and their post-processed results, with medians highlighted, for different predictors of reduced coefficients. The values are relative to the hogging condition, simulated on the full ship FS6 with 10 parameterized sections.	36
3.13	Comparison of the 5-CV errors on the stress tensor components and their post-processed results, with medians highlighted, for different predictors of reduced coefficients. The values are relative to the hogging condition, simulated on the full ship FS6 with 40 parameterized sections.	37
3.14	Comparison of the 5-CV errors on n_y and n_b , with medians highlighted, for different prediction approaches. The values are relative to FS6, with 10 and 40 parameterized sections.	37
4.1	On the left, position i^* in the low-fidelity PF is selected as maximizer of the infill criterion Δ evaluated on matrices $\mathbf{C}^{LL}$ and $\mathbf{C}^{LH}$ using Equation (4.5). The addition of the i^* -th sample to the high-fidelity set is simulated with the removal of row (and column) i^* from $\mathbf{C}^{LL}$, its addition to $\mathbf{C}^{LH}$, followed by the removal of row i^* . The next sample is selected using the updated matrices.	47
4.2	On the left, the GPR prediction with its uncertainty on the top and the acquisition functions at the bottom. On the right, the updated GPR after the addition of the previously selected sample, for each acquisition function.	51
4.3	Representation of the velocity update calculation in particle swarm optimization. The grid cells correspond to the areas of the continuous domain that evaluate with the same $\mathbf{x}^{eval}$, the black lines represent the active linear constraints and domain bounds, and the cells in red are associated to infeasible $\mathbf{x}^{eval}$. The particles are represented with circles trailing a path of past positions, and the stars of the corresponding color show the associated $\mathbf{x}^{pbest}$. In this case, the global best $\mathbf{x}^*$ is in orange. The arrows show the components of the velocity update: the orange arrows are given by $\mathbf{x}^* - \mathbf{x}$ and the colored arrows come from $\mathbf{x}^{pbest} - \mathbf{x}$	55

4.4	Comparison of multi-objective infill criterion values and structural QoIs errors corrected by the high-fidelity simulation, during repeated multi-objective optimization of MS00. Different colors of the criterion values denote the rank of the selected sample in the infill. The error plots highlight the median of the distribution.	56
4.5	Evolution of the surrogate PF in subsequent runs of the genetic algorithm, for MS00. Each plot depicts the 2-dimensional PF computed from the projection of the full 5-dimensional PF.	57
4.6	Comparison of multi-objective infill criterion values and structural QoIs errors corrected by the high-fidelity simulation, during repeated multi-objective optimization of FS6. Different colors of the criterion values denote the rank of the selected sample in the infill. The error plots highlight the median of the distribution.	58
4.7	Comparison of single-objective optimization algorithms, starting from a single high-fidelity sample. The test case is FS6 with 40 parameterized sections. . .	59
4.8	Comparison of single-objective optimization algorithms, starting from 30 high-fidelity samples. The test case is FS6 with 40 parameterized sections. . . .	60
4.9	Comparison of single-objective optimization algorithms, starting from 84 high-fidelity samples. The test case is FS6 with 40 parameterized sections. . . .	60
5.1	Three parameterizations of FS0, produced by the designers. On the left, 16 decision variables are assigned to the higher accommodation decks and the bottom. At the center and the right, the parameterizations extend to the shell and the bulkheads, and use 40 and 76 parameters respectively. Different colors denote different parameterized sections, and the unparameterized parts of the ship are transparent.	64
5.2	Buckling behavior for x_4 in MS00, obtained by modifying the value of the parameter while keeping the remaining thicknesses fixed at their optimal values. Red indicates buckled elements.	66
5.3	The reparameterization procedure applied to a simplified deck composed of 6 patches. The patch colors in steps 1, 2, and 4 represent different thicknesses. In step 1, all patches have the same thickness as selected by the single-objective optimization. In step 2, different thicknesses are evaluated to obtain the structural QoIs of each patch. In step 3, the structural responses determine the reparameterization problem, and its optimal solution clusters the patches in two groups. In step 4, the optimized thicknesses are assigned to the patches and the new mapping to the decision variables is thus obtained.	68
5.4	The reparameterization of x_4 in MS00 selects two thickness values, 6 and 9 mm. The buckling behavior for a single parameter is represented by the figures on top, with red indicating buckled elements. Figure 5.4c shows the solution to Problem 5.1, and Figure 5.4d shows the corresponding grouping of patches.	71

5.5	Evolution of the minimum and maximum for the normalized singular values decay of the stress tensor components, for the midship section with 10 parameters. Values are reported for the high-fidelity database before and after optimization.	71
5.6	Evolution of the 5-CV error distribution for the midship section with 10 parameters. Values are reported for the high-fidelity database before and after optimization. The gray box identifies the chosen truncation rank.	72
5.7	Evolution of the minimum and maximum for the normalized singular values decay of the stress tensor components, for MS00 with 17 parameters. Values are reported for the high-fidelity database before and after optimization.	72
5.8	Evolution of the 5-CV error distribution for MS00 with 17 parameters. Values are reported for the high-fidelity database before and after optimization. The gray box identifies the chosen truncation rank.	73
5.9	Evolution of the surrogate PF for MS00, for different parameterizations.	73
5.10	Evolution of the optimal high-fidelity m_{gap} of MS00, for the iterative reparameterization approach.	74
5.11	Evolution of the optimal high-fidelity m_{gap} of MS00, for different reparameterization approaches.	74
5.12	Details of the reparameterization of deck 14. Figure 5.12a shows the absolute value of the σ_x from the stress tensor under sagging condition, with red representing the largest values and blue 0. Figures 5.12b, 5.12c and 5.12d show the assignments to different decision variable for the 16, 27 and 40 parameters models, respectively.	75
5.13	Evolution of the minimum and maximum for the normalized singular values decay of the stress tensor components, for FS6 with 16 parameters. Values are reported for the high-fidelity database before and after optimization.	76
5.14	Evolution of the 5-CV error distribution for FS6 with 16 parameters. Values are reported for the high-fidelity database before and after optimization. The gray box identifies the chosen truncation rank.	76
5.15	Evolution of the surrogate PFs for FS6, for different parameterizations of the iterative approach. The threshold for the VCG constraint is not shown, since the projected PF lies well below its value.	77
5.16	Evolution of the optimal high-fidelity m_{gap} of FS6, for the iterative reparameterization approach.	77
5.17	Evolution of the optimal high-fidelity m_{gap} of FS6, for different reparameterization approaches.	78
6.1	Schematic representation of a prediction using tree boosting. Each tree has different splits, which determine different paths to the leaf value, highlighted in red. The final output is the summation of all predictions, with the trees beyond the first acting as correctors to the first element of the ensemble.	85

6.2	Comparison of the optima reached for FS0 with different parameterizations.	90
6.3	Comparison of the optima reached for FS3 with different parameterizations.	91
6.4	Comparison of the optima reached for FS4 with different parameterizations.	92
6.5	Comparison of the optima reached for FS5 with different parameterizations.	93
7.1	Chamfer distance matrix, with the columns corresponding to sections of the source ship (in blue), and the rows corresponding to sections of the target ship (in red). Lower values for the matrix entries correspond to more similar sections, as can be seen between the entry corresponding to the sets of accommodation decks, while the highest distance is observed between external bulkheads and the bottom parts of the shell.	102
7.2	Comparison between the Chamfer distance computed on the scaled coordinates, the distance between the embedding coordinates computed via MDS, and the corresponding absolute error.	102
7.3	Covariance matrix between the source and target sections. The highest values are found for the entries corresponding to the pairs with lower Chamfer distance.	102
7.4	Transfer matrix between the source and target sections. Most rows have a single entry with a value close to 1.	103
7.5	Transfer matrix between the source and target sections. The source shell is divided in two sections, which have very similar contribution to the output for the single shell section of the target ship.	103
7.6	PF of gaps loss and average number of unique transferred configurations for population transfer, evaluated on the training set of midship section variants.	105
7.7	Evolution of IGD during NSGA-III for the midship section variants MS00 through MS06. The baseline random initialization is in blue, while the evolution of the seeded population is in orange. The shaded area shows the minimum and maximum values measured on 10 repeated runs.	107
7.8	Evolution of m_{gap} during PSO, for the midship section variants MS00 through MS06. The baseline random initialization is in blue, while the evolution of the seeded population is in orange. The shaded area shows the minimum and maximum values measured on 10 repeated runs. The dashed line represents the reference optima from the database.	109
7.9	PF of gaps loss and average number of unique transferred configurations for population transfer, evaluated on the training dataset of full ships.	110
7.10	Evolution of IGD during NSGA-III for FS7. The baseline random initialization is in blue, while the evolution of the seeded population is in orange. The shaded area shows the minimum and maximum values measured on 10 repeated runs.	111

7.11	Evolution of IGD during NSGA-III for FS9. The baseline random initialization is in blue, while the evolution of the seeded population is in orange. The shaded area shows the minimum and maximum values measured on 10 repeated runs.	111
7.12	Evolution of m_{gap} during PSO, for FS7. The baseline random initialization is in blue, while the evolution of the seeded population is in orange. The shaded area shows the minimum and maximum values measured on 10 repeated runs. The dashed line represents the reference optimum from the database. . . .	112
7.13	Evolution of m_{gap} during PSO, for FS9. The baseline random initialization is in blue, while the evolution of the seeded population is in orange. The shaded area shows the minimum and maximum values measured on 10 repeated runs. The dashed line represents the reference optimum from the database. . . .	112
8.1	Representation of a DT, combining a physical asset, a digital asset, and a decision-making component. The exchange of information is obtained via the generation and communication of different data from one component to the next.	117
8.2	The first two slices of a DBN for a generic DT system. Circles denote random variables, squares denote actions taken, and diamonds identify the reward.	119
8.3	Implementation diagram of a new DT system using pgmtwin.	121
8.4	Strain of the deformed cantilever beam under three different levels of damage. No damage on the left, 58% damage in the section closest to the clamped end at the center, and 71% of damage in the section further from the clamped end on the right. The colors represent the magnitude of the strain, with blue being 0 and red indicating the largest values.	123
8.5	Transition matrices of the SHM maintenance actions. The "do nothing" and "minor repair" maintenance actions risk degrading the structural integrity of the beam, while "perfect repair" guarantees full restoration. Darker shades of blue represent higher probability of transition.	124
8.6	Placement of the displacement sensors on the cantilever's mesh.	124
8.7	Confusion matrix for the SHM inverse problem. Darker shades of blue indicate a higher score associated to the truth and prediction combination. Low damage on the section farther from the clamped end results in lower displacement values and a more difficult discernment of the actual D	125
8.8	Tabular policy for the SHM, obtained via Q-learning with the stochastic ActionUpdateComponent. Darker shades of blue represent higher probability of choice.	125

8.9	Simulation of the baseline SHM system, using sensors corrupted by Gaussian noise, stochastic physical asset update, and tabular policy. The top plot represents the estimation of the digital state, the plot in the middle compares the action selected and the one that would be chosen with perfect information, while the bottom plot shows the evolution of the cumulative reward. Darker shades of blue represent higher confidence associated to the assimilated state.	126
8.10	DBN for the assimilation. The evidence nodes $D_{IP}^{(1)}$, $D^{(0)}$, and $U_{PREV}^{(1)}$ render the rest of the full DBN superfluous for the inference of $D^{(1)}$.	127
8.11	Simulation of the SHM PGM-based system, using 2 samples for PGM inference, Gaussian sensor noise, stochastic physical asset update, and tabular policy. The top plot represents the estimation of the digital state, the plot in the middle compares the action selected and the one that would be chosen with perfect information, while the bottom plot shows the evolution of the cumulative reward. Darker shades of blue represent higher confidence associated to the assimilated state.	128

LIST OF TABLES

2.1	Parameters description of the midship section test cases. Thickness values are in mm.	15
2.2	Midship section variants features.	16
2.3	Details of the full ship models, for their basic parameterization.	18
5.1	Optimization results for different parameterizations of MS00.	75
5.2	Optimization results for different parameterizations of FS6.	78
6.1	Features of a single patch.	83
6.2	Midship section variants optima - default configuration in training data. . .	87
6.3	Midship section variants optima - default configuration missing from training data.	88
6.4	FS0 - comparison of optima.	90
6.5	FS3 - comparison of optima.	91
6.6	FS4 - comparison of optima.	92
6.7	FS5 - comparison of optima.	93
7.1	Comparison of gaps loss for different combinations of transfer hyper-parameters, for the validation set of midship section variants.	106
7.2	Comparison of gaps loss for different combinations of transfer hyper-parameters, for the validation set of full ships.	110
8.1	Comparison of the baseline DT and the PGM-based version, for different numbers of total time slices.	128
A.1	Midship section variants optimization constraints.	158
A.2	Midship section variants optima - test ships.	159
A.3	Midship section variants optima - train ships.	160
A.4	Full ships optima.	161

LIST OF ALGORITHMS

1	Template of a genetic algorithm.	44
2	Non-dominated sorting.	44
3	NSGA-III selection strategy.	45
4	Down-sampling procedure for single-objective populations.	49
5	principal directions search.	50
6	Specialized BO implementation.	53
7	Particle swarm optimization.	54
8	Population transfer.	104

ACRONYMS

10-CV	10-fold cross-validation
5-CV	5-fold cross-validation
ACO	ant colony optimization
ALE	arbitrary Lagrangian Eulerian
ARD	automatic relevance determination
AS	active subspace
BN	Bayesian network
BO	Bayesian optimization
CFD	computational fluid dynamics
COG	center of gravity
CPD	conditional probability distribution
DAG	directed acyclic graph
DBN	dynamic Bayesian network
DE	differential evolution
DIRECT	Dlviding RECTangles
DNV	Det Norske Veritas
DT	digital twin
EI	expected improvement
ES	evolutionary strategy
FEA	finite element analysis
GA	genetic algorithm
GNN	graph neural network

GP	Gaussian process
GPR	Gaussian process regression
IGD	inverse generational distance
ILP	integer linear program
IMO	International Maritime Organization
KLE	Karhunen-Loeve expansion
L-BFGS	low-memory Broyden–Fletcher–Goldfarb–Shanno
LCG	longitudinal center of gravity
MAPE	mean absolute percentage error
MDS	multi-dimensional scaling
ML	machine learning
MOR	model order reduction
MSE	mean squared error
NARGP	auto-regressive non-linear Gaussian process
NARGPAS	auto-regressive non-linear Gaussian process with active subspaces
NLCB	negative lower confidence bound
NSGA	non-dominated sorting genetic algorithm
NSGA-III	non-dominated sorting genetic algorithm III
PDE	partial differential equation
PDS	principal directions search
PF	Pareto front
PGM	probabilistic graphical model
PI	probability of improvement
PME	parametric model embedding
POD	proper orthogonal decomposition
PS	Pareto set
PSO	particle swarm optimization
QoI	quantity of interest
RB	reduced basis
RBF	radial basis function
RL	reinforcement learning
ROM	reduced order model
RSM	response surface method

SHM	structural health monitoring
SVD	singular value decomposition
TCG	transversal center of gravity
TL	transfer learning
TPE	tree-structured Parzen estimator
TR	trust region
UQ	uncertainty quantification
VCG	vertical center of gravity

INTRODUCTION

Modern engineering requires efficient optimization pipelines to tackle the ever increasing complexity of market demands, reduce manufacturing and operational costs, comply with international safety and environmental regulations, and satisfy production constraints.

In the shipbuilding industry, the emission reduction targets set by the [International Maritime Organization \(IMO\)](#) [101, 97] and the digitalization of the ship [123, 81] pose new challenges to the designers, who are tasked with the integration of innovative propulsion systems into new and existing vessels, as well as managing the many interacting subsystems that will enable the [digital twin \(DT\)](#) of the ship. All of this while ensuring structural resilience [57] and reducing manufacturing costs.

To address these requirements, the design process is evolving towards a holistic approach [139, 127] in which the simulation of the structural responses, the fluid dynamics properties of hull and propeller, and the life-cycle of the vessel all concur to the final design [98, 185].

Despite the advances in numerical modeling and computational power, which continue to improve the accuracy of [finite element analysis \(FEA\)](#) and [computational fluid dynamics \(CFD\)](#) simulations while reducing their computational cost, the large time-to-solution of high-fidelity analyses prevents the effective optimization of modern engineering systems. On the one hand, even expert designers can be overwhelmed by the number of decision variables and constraints due to regulatory and industrial requirements, as well as the number of competing [quantities of interest \(QoIs\)](#) from different stakeholders. On the other hand, automated exploration of the design space requires a large number of simulations to validate a new design, often with minimal changes to the design variables.

The work presented in this thesis, carried out in collaboration with Fincantieri S.p.A., addresses these challenges. We focus on the structural optimization of cruise ships in the preliminary design phase, where the designers aim to minimize the total mass of the ship, subject to regulatory and safety constraints. To address this demand, we propose a surrogate-based optimization pipeline [73]. By leveraging data-driven [model order reduction \(MOR\)](#) [30, 184, 39], we reduce the computational cost of [FEA](#) simulations, trading off accuracy for a reduction of the time-to-solution. The exploration of the design space is carried out using the surrogates, with multi- and single-objective optimization,

and only the best candidates are validated with high-fidelity simulations.

Our contributions span multiple stages of the optimization pipeline. We first assess the choice of surrogate models, proposing a hybrid approach that combines the reconstruction of the fields of stress tensors via [proper orthogonal decomposition \(POD\)](#) and [Gaussian process regression \(GPR\)](#), followed by non-linear post-processing. For multi-objective optimization, which approximates the [Pareto front \(PF\)](#) and enables designers to analyze the trade-offs between competing objectives, we propose a novel infill strategy for the adaptive selection of new high-fidelity simulations. For single-objective optimization, we extend [particle swarm optimization \(PSO\)](#) [105, 74] with a repair procedure based on an [integer linear program \(ILP\)](#) [166] to handle discrete decision variables and linear constraints, and a local search refinement phase. The resulting algorithm is compared with alternative black-box optimization approaches, including [Bayesian optimization \(BO\)](#) [174], as well as established optimization frameworks such as Optuna [2] and Nevergrad [153].

To address the high dimensionality of the design space, we propose a data-driven parameterization refinement procedure [65]. Rather than relying on parameter reduction techniques such as [active subspace \(AS\)](#) [45, 44] or [parametric model embedding \(PME\)](#) [171, 170], the optimization pipeline starts from a coarse parameterization and progressively introduces new decision variables through the analysis of structural responses obtained from the surrogates. As a result, the formulation of the optimization problem is adapted to the structural behavior emerging during the optimization process, avoiding the need for an extensive initial sampling, and reducing the dependence on manually defined parameterizations.

We then leverage [transfer learning \(TL\)](#) [137, 87] to further enhance the optimization pipeline. First, we reformulate the reparameterization procedure to enable a one-shot refinement of the optimization problem prior to the construction of surrogate models, thus eliminating the need for an extensive initial database of high-fidelity simulations. We also develop a population transfer methodology based on [GPR](#) [85], initializing the populations of the metaheuristic optimizations with configurations from past optimizations to accelerate the exploration of promising areas of the current design spaces.

The proposed pipeline is evaluated on representative test cases, using a family of midship section variants and a group of full-scale commercial designs.

Finally, we address the adoption of [DTs](#) in the industrial practice. To facilitate the development and assessment of [DTs](#) for complex engineering systems, we introduce `pgmtwin`, an open-source Python library for the implementation and analysis of [DT](#) architectures based on [probabilistic graphical models \(PGMs\)](#) [114, 103]. `pgmtwin` provides the user with easily extensible components, standardized procedures, and ready-to-use [DT](#) systems. We demonstrate the streamlined workflow of `pgmtwin` on the implementation of a [DT](#) for the [structural health monitoring \(SHM\)](#) of a simplified cantilever beam.

The rest of the thesis is organized following the structure of the pipeline, represented in Figure 1.1.

Chapter 2 presents the industrial setting in which the optimization pipeline is used,

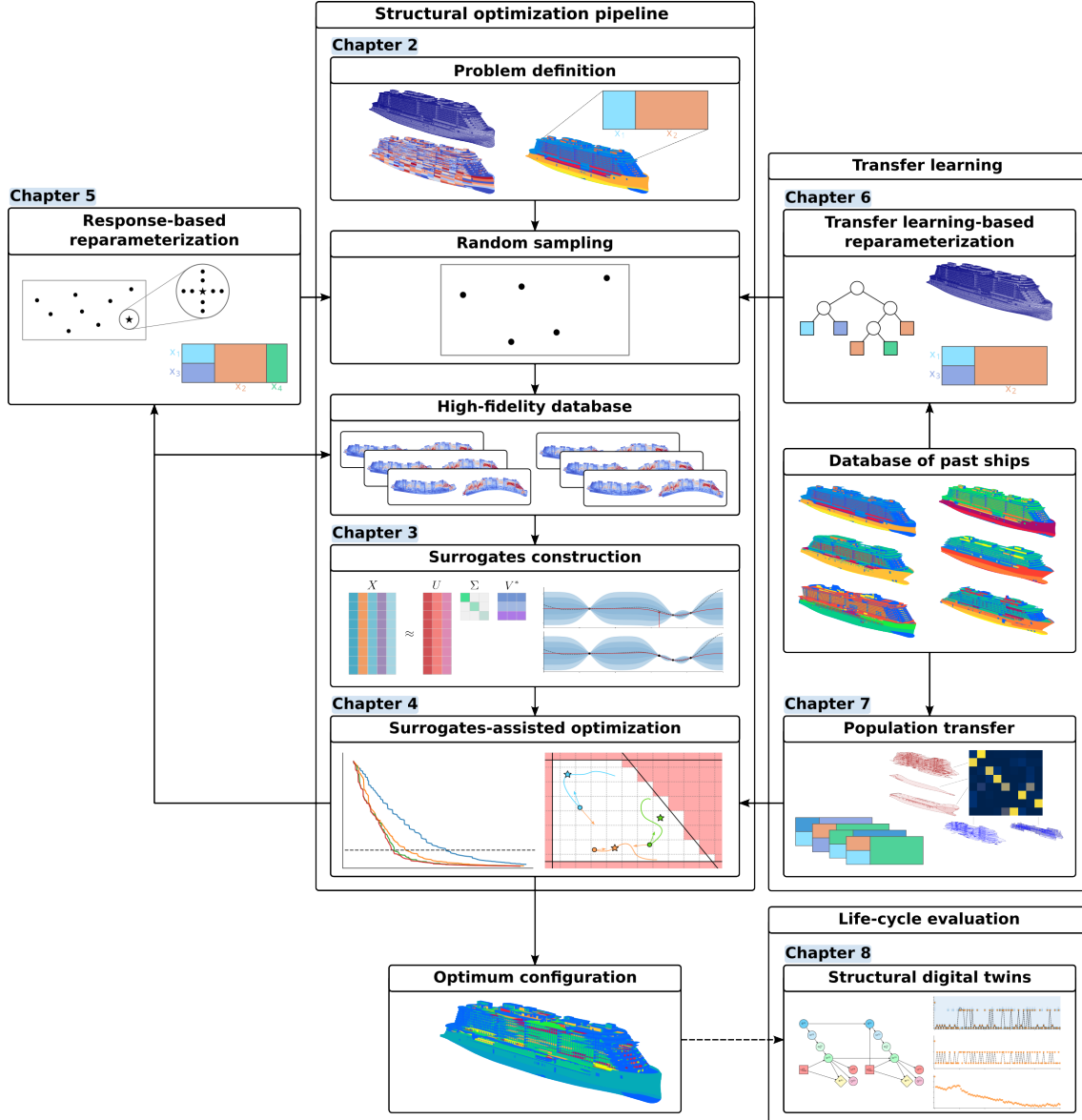


Figure 1.1: Pipeline for the structural optimization of cruise ships. The response-based reparameterization refines the problem leveraging the surrogates, while transfer learning extracts knowledge from past optimizations to enable a one-shot reparameterization before the initial sampling and the transfer of populations to accelerate the exploration of the domain.

detailing the structural problem, its quantities of interest, the feasibility constraints and the parametric domain. The chapter ends with the description of the models used in the numerical evaluations, a dataset of simplified midship section variants and a group of commercial, full-scale vessels.

Chapter 3 discusses the approaches used for the model order reduction of the structural problem, focusing on proper orthogonal decomposition with [GPR](#). The numerical results evaluate the chosen approach on a simplified midship section and a full ship model, showing its effectiveness.

Chapter 4 presents the approaches to multi- and single-objective optimization of the surrogates. The chapter discusses the implementation of a novel infill criterion for the approximation of the [PF](#), and the integration of discrete, constrained versions of [BO](#) and [PSO](#). The proposed methods are then evaluated on a midship section and a full ship.

Chapter 5 proposes a novel method for the refinement of the optimization problem, in which the set of decision variables is adapted to the behavior of the model through the analysis of the surrogate structural response. The integration of the procedure in the optimization pipeline leads to higher quality configurations on both a midship section and a full-scale model.

Chapter 6 adapts the refinement procedure from the previous chapter to a [TL](#) approach, relaxing the requirement of high-fidelity evaluations and surrogates construction. The resulting optimization pipeline is able to achieve promising trade-offs between the final optimum and the total number of simulations, during evaluation on multiple midship sections and full-scale ships.

Chapter 7 applies [TL](#) to the multi- and single-objective procedures, proposing a novel method for the transfer of elite configurations from past optimizations to the current one. The construction of initial populations for the metaheuristics accelerates the exploration of the domain and achieves better optima on a group of midship sections and full-scale ships.

Chapter 8 discusses the use of [DTs](#) for the integration of the life-cycle of the ship in the design phase. The novel `pgmtwin` library is proposed for the study of general [DTs](#), and demonstrated on the [SHM](#) of a cantilever beam.

Chapter 9 presents the closing remarks on the contributions of the thesis and draws possible directions for future work.

PROBLEM DEFINITION

This chapter presents the high-fidelity [finite element analysis \(FEA\)](#) simulation around which the optimization pipeline from Chapter 1 has been designed.

We consider the global structural response of the ship against extreme waves, with the resulting stress configurations being post-processed to obtain the yielding and buckling state of the plates that compose the structures. The definitions of the simulation loads, the criteria for structural failure identification, and the validation of the [FEA](#) setup come from regulatory entities [57, 156].

Other than yielding and buckling of the structural elements, the designers are also interested in the vertical deflection incurred by the ship, which constitutes another index of structural resilience. Although there is no regulatory limit for the vertical deflection, in naval engineering practice it constitutes a useful benchmark to compare ships of similar length and provides an indicator for the structural continuity, reflecting whether the arrangement of load-bearing structures is effective. Then, the position of the [center of gravity \(COG\)](#) of the model is subjected to constraints imposed by the static stability requirements of the vessel. All of these [quantities of interest \(QoIs\)](#) are balanced against the total mass of the ship, which constitutes a competing objective of the design phase. Indeed, the designers aim to identify the best trade-off between structural resilience and safety on one side, and total weight and performance of the ship on the other.

In this thesis, the design variables are limited to the thickness of primary structural elements, while the secondary stiffeners have immutable sizing and spacing.

We remark that the use of a global model in structural optimization of large ships is still uncommon [190, 9, 5], with previous works often adopting simplified models [125]. A minority of works define a single ring of the main hull, and extrude to a prismatic beam for [FEA](#) [14, 33, 128]. A more sophisticated approach is the complete modeling of the midship section, which corresponds to the central body of the ship with the bow and stern removed [180, 32, 47, 134, 151]. Some of the previous works also use design variables for the spacing of stiffeners, their thickness and their profile, but this approach requires either re-meshing or a more involved modeling of the [FEA](#) [162].

This chapter is structured as follows. In Section 2.1, the [FEA](#) setup and the post-processing for the computation of the [QoIs](#) are presented and discussed. Section 2.2

introduces the notation that will be used throughout the thesis. The numerical test cases are presented in Section 2.3. We use a simplified midship section and its variants as an inexpensive illustrative problem, while a group of 10 full-scale, commercial ship models constitute the large-scale applications.

2.1 Structural problem formulation

In our setting, the geometry of a ship is given as a computational mesh. The primary structural elements, which are manufactured by welding large steel plates, are modeled with groups of triangular and quadrilateral thin plate elements with edges of about 700 mm. The secondary structural elements form the webs of stiffeners, which are modeled with both thin plate and beam elements of varying sizes.

In a [FEA](#) simulation, we consider a linear elasticity problem where the load conditions are designed to reproduce the effects of the most extreme waves that the ship can encounter during her lifetime [125]. The hogging wave is characterized by a crest at mid-length, while the sagging wave presents a trough at mid-length. Figure 2.1 represents the results of a simulation that employs the two load conditions.

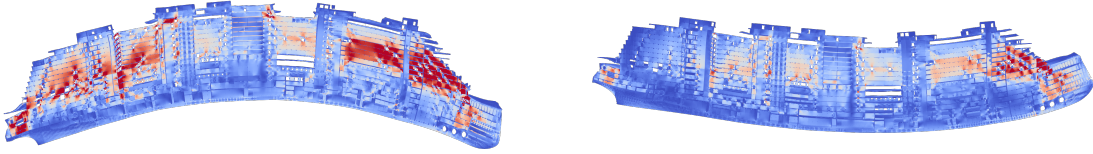


Figure 2.1: Simulations of a ship being deformed under a hogging load condition on the left, and sagging on the right. Displacements are magnified, with colors representing the von Mises stress from blue (zero) to the maximum value.

The detailed bending moments and shear forces associated to each load condition are prescribed by the classification societies. We consider the set of rules by [Det Norske Veritas \(DNV\)](#) [57] and RINA [156].

The simulations are carried out with MSC NASTRAN's [93] SOL 101, on a computational mesh that uses CTRIA3 and CQUAD4 shell elements. The results are post-processed to obtain the 3D stress tensors of each shell element under each load condition. The translation from 2D to 3D of the stress tensor is required by the certified tooling employed in other phases of post-process and design, and is carried out via the computation of a rotation matrix from the centroid's geometry. We denote the 3D tensors as

$$S_e^l = \begin{bmatrix} \sigma_x & \tau_{xy} & \tau_{xz} \\ \tau_{xy} & \sigma_y & \tau_{yz} \\ \tau_{xz} & \tau_{yz} & \sigma_z \end{bmatrix}, \quad (2.1)$$

where l denotes the load condition, e the shell element index, σ the direct stresses, and τ the shear components. Subscripts to the stress components identify the axis, or plane, of action.

However, the hogging and sagging load conditions mandated by the classification rules can not be directly interpreted as physical operating scenarios. Indeed, the corresponding forces and loads are artificially designed to elicit extreme responses and the FEA simulations can only approximate the mandated values through boundary conditions [125]. Figure 2.2 shows the discrepancy between the shear force Q and bending moment M mandated by the regulatory entities, and the actual loads used in FEA.

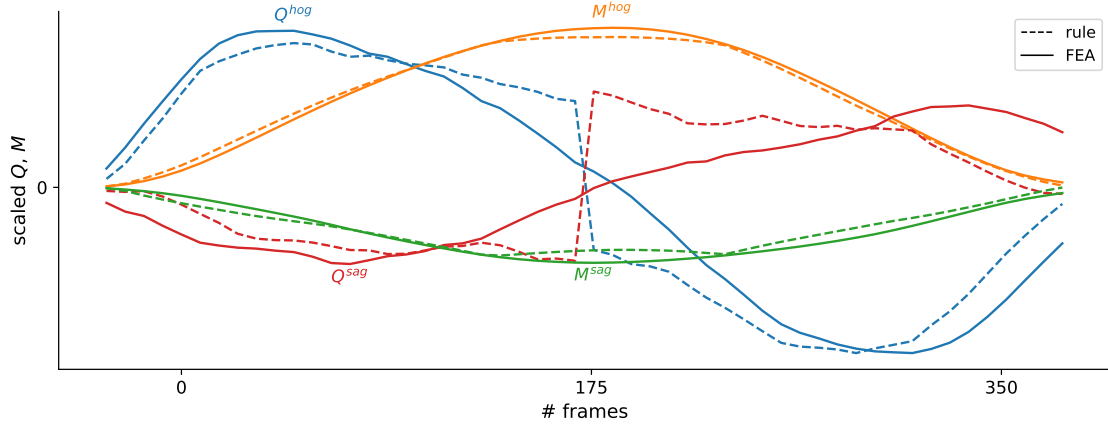


Figure 2.2: Comparison of the shear Q and bending moment M , for the hogging and sagging load conditions. The mandated values are represented with dashed lines, while the actual loads for the FEA use solid lines. For confidentiality reasons, numerical values on the vertical axis are omitted. Negative abscissae correspond to coordinates behind the stern. Shear and bending moment values are scaled independently to enable qualitative comparison.

To account for this discrepancy, the designers apply a correction method called wave-factoring approach, where the stresses computed under approximated hogging and sagging are corrected by a linear interpolation of two additional load conditions, called wave-A and wave-B. The first simulation of a new ship is used to compute a set of correction coefficients for the approximated results, which from that point on will be applied to all simulations as post-processing step. Let Q denote the vector of shear force and M be the corresponding bending moment associated to each longitudinal section of the ship, called frame. Then, from the simulation of the factoring waves we obtain the vectors Q^A , Q^B , M^A , and M^B . For $l \in \{\text{hog}, \text{sag}\}$ and each frame f , the vectors of coefficients to be used in the interpolation are derived from the following system

$$\begin{bmatrix} Q_{\text{rule},f}^l - Q_{\text{FEA},f}^l \\ M_{\text{rule},f}^l - M_{\text{FEA},f}^l \end{bmatrix} = \begin{bmatrix} Q_f^A & Q_f^B \\ M_f^A & M_f^B \end{bmatrix} \begin{bmatrix} A_f^l \\ B_f^l \end{bmatrix}, \quad (2.2)$$

where $Q_{\text{rule},f}^l - Q_{\text{FEA},f}^l$ denotes the difference between the regulatory shear and the actual simulation at frame f , and $M_{\text{rule},f}^l - M_{\text{FEA},f}^l$ is the analogous for the bending moment. The coefficients computed for the frames are then linearly interpolated along the longitudinal dimension to obtain A_e^l and B_e^l associated to each element e . The corrected field of stress

tensors $\bar{S}_e^l$ is then obtained as

$$\bar{S}_e^l = S_e^l + A_e^l S_e^A + B_e^l S_e^B, \quad (2.3)$$

where S_e^A and S_e^B are the fields of stress tensors obtained for wave-A and wave-B, respectively. Due to this procedure, a complete simulation requires solving 4 different load conditions: the two approximated regulatory waves, and the two factoring waves. Figure 2.3 shows the shear and bending moment of the factoring waves, and the resulting correction coefficients. At the extremities of the ship, the linear systems from Equation 2.2 become ill-conditioned and the corresponding coefficients diverge. In practice, the designers simply discard the correction when the coefficients grow beyond a threshold proportional to the length of the ship.

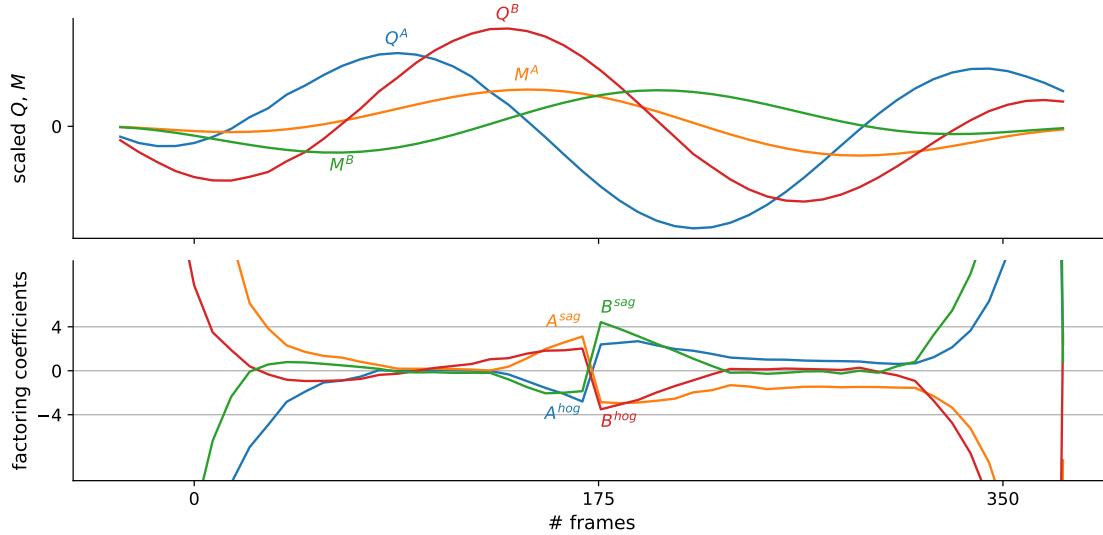


Figure 2.3: The shear and bending moment for the additional loads due to factoring correction are represented in the top plot. For confidentiality reasons, numerical values on the vertical axis are omitted. Shear and bending moment values are scaled independently to enable qualitative comparison. Negative abscissae correspond to coordinates behind the stern. The bottom plot shows the resulting coefficients for the post-processing of the hogging and sagging FEA results.

The structural failures considered in the preliminary design phase are the yielding and buckling of the shell elements. The yielding state of each shell is obtained by comparing the components of the stress tensor with the corresponding critical thresholds, as well as computing the von Mises criterion σ_{VM} :

$$|\sigma_i| \leq 245 \text{ N/mm}^2, \quad \text{for } i \in \{x, y, z\}, \quad (2.4a)$$

$$|\tau_i| \leq 153 \text{ N/mm}^2, \quad \text{for } i \in \{xy, xz, yz\}, \quad (2.4b)$$

$$\sigma_{VM} = \sqrt{\frac{(\sigma_x - \sigma_y)^2 + (\sigma_y - \sigma_z)^2 + (\sigma_z - \sigma_x)^2}{2}} + 3(\tau_{xy}^2 + \tau_{xz}^2 + \tau_{yz}^2) \leq 307 \text{ N/mm}^2, \quad (2.4c)$$

where the critical values refer to high strength structural steel (AH36) [8] with a tolerance factor already applied. An element is marked as yielded if any of the conditions in Equation (2.4) is violated.

The buckling state is similarly obtained from the aggregation of 11 usage factors, computed for each combination of element and load condition according to formulae by the classification societies [125, 57]. An element is marked as buckled if at least one usage factor exceeds the critical threshold, for at least one load condition. We remark that even when based on the linearized buckling formulation, the expressions for the usage factors are non-linear both in the stress tensor components and in the geometrical features of the shells being considered. Moreover, we only consider linear buckling where the elements fail independently, instead of converging to the failure state by simulating the removal of buckled elements.

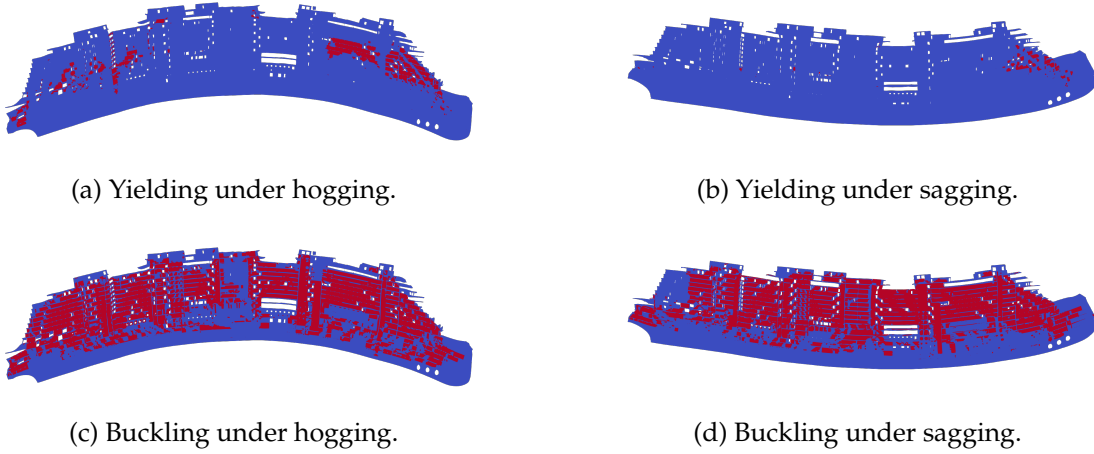


Figure 2.4: Yielding and buckling under hogging and sagging load conditions. Elements colored in red fail the structural stability tests mandated by the regulatory entities.

Another indicator of structural integrity is the vertical deflection, computed as the maximum absolute vertical displacement of a representative mesh node, where higher values indicate less resilient ships.

In this work, the three [QoIs](#) computed from the [FEA](#) results that describe the structural integrity of a ship are the number of yielded and buckled elements, and the vertical deflection. Figure 2.4 represents the yielding and buckling state of the elements, together with the displacements due to the hogging and sagging load conditions, of a configuration with low structural integrity.

The designers can control the ship's structural behavior by assigning the thickness of the primary structural elements, while the secondary stiffeners are considered fixed. Thicker decks and bulkheads directly increase the ship's stiffness against the load conditions, decreasing the insurgence of yielding and buckling as well as the vertical deflection. During the preliminary design phase, the designers can tolerate a limited amount of yielded and buckled elements, since in further phases of the design process these phenomena can be corrected with more involved solutions. Rather than resorting to change the

configuration of secondary stiffeners, or change the geometry of the model, in this work we consider critical thresholds on the number of yielded and buckled elements as optimization constraints. Moreover, we model buckled elements as having an additional stiffener automatically applied, which slightly increases the total mass of the configuration. Additionally, the vertical deflection can be constrained by setting a upper critical threshold.

A crucial trade-off emerges: increasing the structural performance of a ship comes primarily at the cost of increasing its total mass, which in turn determines more expensive and longer building process, and higher operational expenses.

Moreover, the allocation of thicknesses needs to also respect constraints on the [COG](#), in particular the value of the [vertical center of gravity \(VCG\)](#). The [VCG](#) must not exceed a critical value, which is computed from the ship's geometry, to prevent heeling over or capsizing.

The list of [QoIs](#) is thus composed of five scalars: number of yielded and buckled elements, vertical deflection, total mass, and [VCG](#). Figure 2.5 shows the relationship between different [QoIs](#). The vertical deflection, the number of yielded elements, and the number of buckled elements increase as the total mass of the ship decrease, as expected. However, even if decreasing the vertical deflection requires a large increase in mass, the behavior of yielded and buckled totals shows that it is possible to find allocations of thicknesses that provide good compromises between structural integrity and steel usage. The [VCG](#), however, does not present a clear trade-off with the mass and thus constitutes a constraint that is always active.

We remark that changing the mass distribution of the model would require a reorganization of the loads for the simulations. This correction is not taken into account in this work, and is left as a manual post-processing of the configuration obtained after optimization.

In principle, the designer could specify a different thickness for each element and exercise extreme control on the structural behaviors by precisely allocating higher thickness values to elements with higher stresses. However, in the physical world the ship is composed by welding together sheets of steel about $2.5\text{ m} \times 15\text{ m}$, which constrain the corresponding elements to share the same thickness. In the computational mesh, these elements groups are called patches and denote the finest level of thickness parameterization of the structural problem. Even so, a large ship defines tens of thousands of patches, which would constitute an unwieldily large search space. The designers thus define a minimum number of parameterized sections, or parameters, which each comprise a large number of patches, to balance the complexity of the search space and the expressiveness of the model.

In designing a parameterization, the first criterion is to separate patches that have different orientation and regulatory minimum thickness. The most common subdivisions, which relates to the functional role of the structural elements, are: bottom, double bottom, accommodation decks, machinery decks, internal longitudinal bulkheads, external longitudinal bulkheads, and the shell.

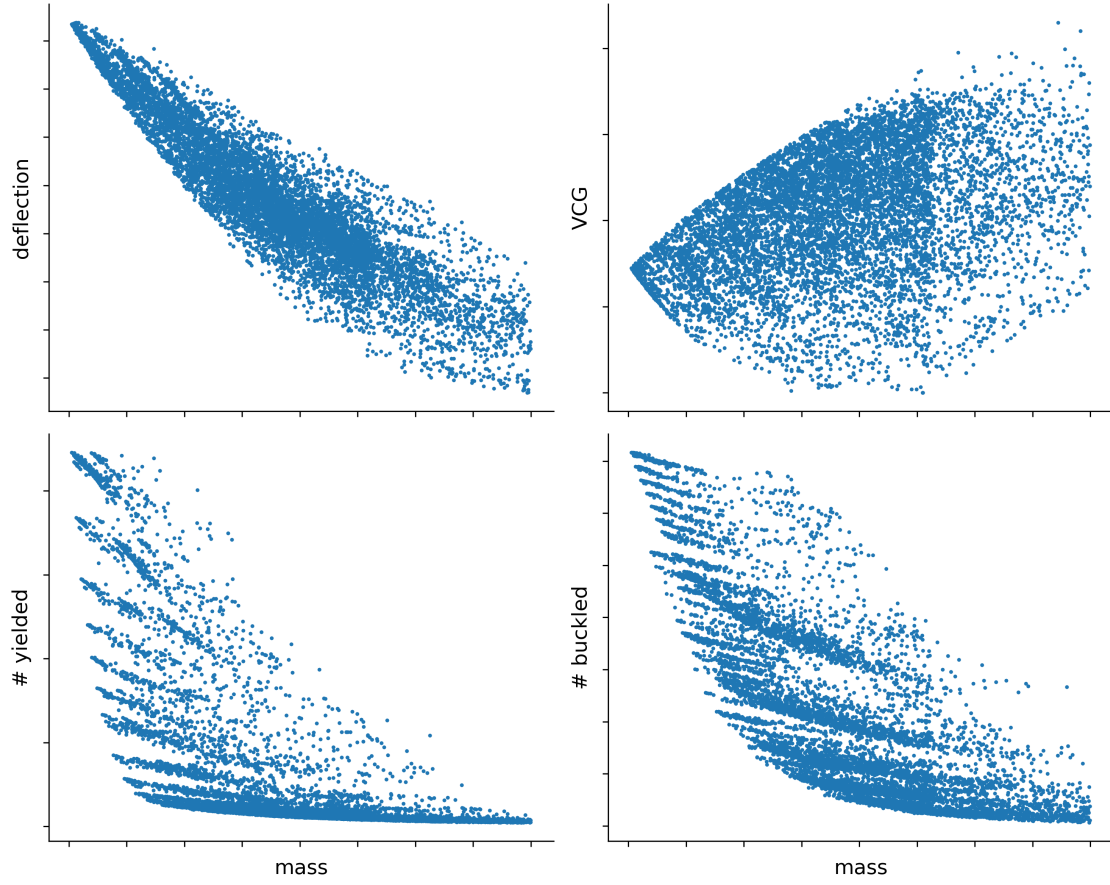


Figure 2.5: Trade-offs between different QoIs. Axes are muted due to confidentiality of data.

Figure 2.6 compares the base model, the patches subdivision, and the initial parameterization of a full ship.

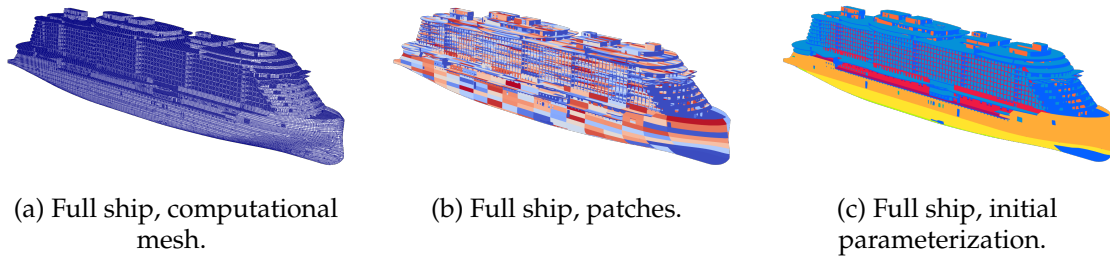


Figure 2.6: Computational mesh of a full ship on the left, the patches subdivision at the center, and a coarse parameterization on the right. Dark blue patches and parameterized elements correspond to elements with fixed thickness.

More refined parameterizations separate patches whose elements sustain much different tensions and loads, in order to allocate structural mass in a more effective manner and only strengthen the parts which would otherwise undergo structural failure. A final source of complexity in the choice of thickness for the parameterized sections is that the domain of possible values is discrete: the designers are limited, at least in the preliminary design

phase, to the sheet sizes commonly available on the market. Typical values range from 5 to 8 mm in increments of 0.5 mm, and from 9 to 20 mm in increments of 1 mm. More rare is the use of higher values, up to 25 mm, which is usually only used for the bottom or the more complex local structures such as staircases with multiple doors.

The thickness of primary structural elements is the only type of parameter considered in this thesis, but we remark that other works in the literature also consider changing the spacing and size of the secondary stiffeners. This type of change requires either to rebuild the computational mesh, which is unfeasible with the full-scale models that we consider in this thesis, or the use of more involved formulations for the [FEA](#) [162].

2.2 Notation

We use $\mathbf{x}$ to denote a configuration of thicknesses, where $x_i \in \mathcal{D}_i$ is the thickness value associated to the i -th parameterized section and $\mathcal{D}_i$ is the corresponding set of feasible values. Denoting the number of parameterized sections as n_{dim} , the domain of possible configuration is thus $\mathcal{D} = \times_{i=1}^{n_{\text{dim}}} \mathcal{D}_i$, where $\times$ denotes the iterated Cartesian product.

The vector of [QoIs](#) computed from a parameter configuration $\mathbf{x}$ is thus

$$\mathbf{f}(\mathbf{x}) = [n_y(\mathbf{x}), n_b(\mathbf{x}), f_{\text{deflection}}(\mathbf{x}), f_{\text{mass}}(\mathbf{x}), \text{VCG}(\mathbf{x})], \quad (2.5)$$

where $n_y(\mathbf{x})$ and $n_b(\mathbf{x})$ are the number of yielded and buckled elements, respectively, computed according to Section 2.1. Other components of the COG can be added if the geometry of the model dictates so. The term $f_{\text{deflection}}(\mathbf{x})$ represents the vertical deflection, while the mass is computed as

$$f_{\text{mass}}(\mathbf{x}) = m_{\text{fixed}} + \mathbf{d} \cdot \mathbf{x} + m_{\text{bar}} n_b(\mathbf{x}), \quad (2.6)$$

where m_{fixed} represents the mass of the elements not controlled by any parameter. The vector $\mathbf{d}$ collects the linear density of each parameterized section, where the i -th component is given by the ratio of total mass of the shell elements assigned to the i -th section, divided by the thickness value. m_{bar} is the average mass of an additional stiffener, which is considered automatically applied to every element that exhibits buckling. To measure the quality of a configuration in terms of mass, it is useful to exclude m_{fixed} and consider the lower bound of the mass associated to the parameterized elements. Using $\mathbf{x}_{\text{LB}}$ as the configuration of minimum thicknesses, we define

$$m_{\text{param. LB}} = \mathbf{d} \cdot \mathbf{x}_{\text{LB}}, \quad (2.7)$$

and during optimization we consider the gap of a configuration's parameterized mass and this lower bound, that is

$$m_{\text{gap}}(\mathbf{x}) = 100 \frac{f_{\text{mass}}(\mathbf{x}) - m_{\text{fixed}} - m_{\text{param. LB}}}{m_{\text{param. LB}}}. \quad (2.8)$$

This transformation provides a more expressive representation of the discrepancy from the ideal value, which corresponds to 0. As a remark, we note that the computation of

structural mass can be easily adapted to compute the cost of the ship. A simple approach is the multiplication by a factor representing the cost of steel per unit of mass, and the inclusion of manual labor cost for the m_{bar} term of Equation (2.6).

The critical thresholds for the number of yielding and buckling phenomena are denoted by y_{crit} and b_{crit} , respectively, so that the corresponding constraints are

$$n_y(\mathbf{x}) \leq y_{\text{crit}}, \quad (2.9)$$

$$n_b(\mathbf{x}) \leq b_{\text{crit}}. \quad (2.10)$$

Likewise, in case the vertical deflection is constrained, we define d_{crit} and write

$$f_{\text{deflection}}(\mathbf{x}) \leq d_{\text{crit}}. \quad (2.11)$$

We compute the COG for the configuration $\mathbf{x}$ on the model at rest, ignoring the stiffeners applied to buckled elements, without the need of a FEA simulation. For the VCG, we write

$$\text{VCG}(\mathbf{x}) = \frac{\text{VCG}_{\text{fixed}} m_{\text{fixed}} + \sum_i \text{VCG}_i \mathbf{d}_i \mathbf{x}_i}{m_{\text{fixed}} + \mathbf{d} \cdot \mathbf{x}}, \quad (2.12)$$

where $\text{VCG}_{\text{fixed}}$ is the VCG of elements not controlled by any parameter, and VCG_i is the VCG of the i -th parameterized region. The exclusion of the small and sparse mass of additional stiffeners is an effective approximation, providing a trivial computation with a negligible error. The other components of the COG use analogous definitions. VCG is positive due to design conventions, so the designer need to constrain it only from above

$$\text{VCG}(\mathbf{x}) \leq \text{VCG}_{\text{crit}}. \quad (2.13)$$

For some ships, other components of the COG must be constrained as well. The transversal center of gravity (TCG) is usually subject to a constrain on its absolute value

$$|\text{TCG}(\mathbf{x})| \leq \text{TCG}_{\text{crit}}, \quad (2.14)$$

when the ship presents large asymmetries between its left and right side, either due to structures or to the placement of the machinery.

The longitudinal center of gravity (LCG) can be constrained to be in an interval around the center of the ship as

$$\text{LCG}_{\text{LB}} \leq \text{LCG}(\mathbf{x}) \leq \text{LCG}_{\text{UB}}. \quad (2.15)$$

2.3 Numerical test cases

This section presents the ship models on which the techniques developed in this thesis will be validated. As an illustrative test case, we use a simplified midship model discussed in Section 2.3.1. Although inexpensive to validate via FEA, this model is comparable with previous works in the literature and constitutes one of the earliest artifacts validated by

the regulatory entities. The base model was modified to obtain a large number of variants, in order to exhibit a variety of different, but related, structural behaviors.

Section 2.3.2 presents the full ship models used as large-scale applications. These ships are actual commercial models that Fincantieri S.p.A. has built or that are currently in the design phase, and constitute realistic use cases.

2.3.1 Midship section variants

The base midship section model, depicted in Figure 2.7, represents a quarter of the central section of a full ship, and spans a length of 95.2 m, 15.5 m width, and 36.55 m height. During simulation, the model is reflected through the xz and yz planes.

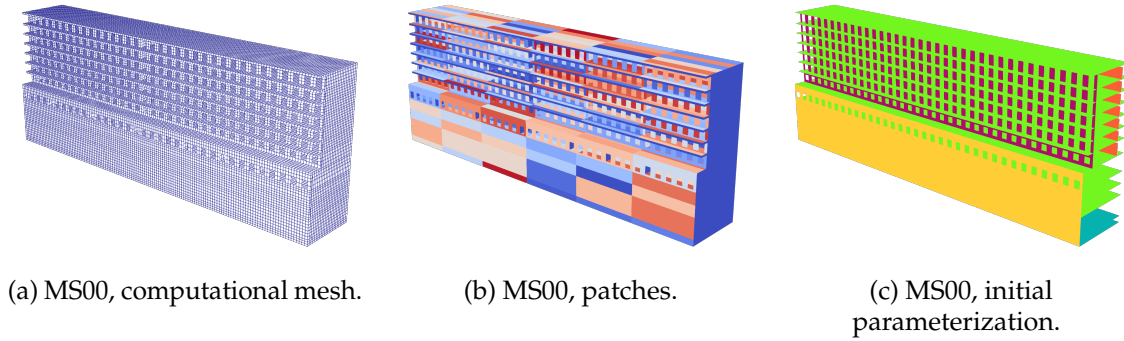


Figure 2.7: Full view of the midship section on the left, the division into patches in the middle, and the initial parameterized sections on the right.

In this study, the critical values for the yielding due to the stresses and the von Mises criterion are lower than the regulatory standard, as the low complexity and scale of this model would otherwise result in no yielded elements. Figure 2.8 shows the von Mises criterion for x_{LB} , under hogging and sagging, where the red elements correspond to a critical value of 184 N/mm^2 . The corresponding reduced thresholds for yielding are 147 N/mm^2 for direct stresses and 91 N/mm^2 for shear stresses.

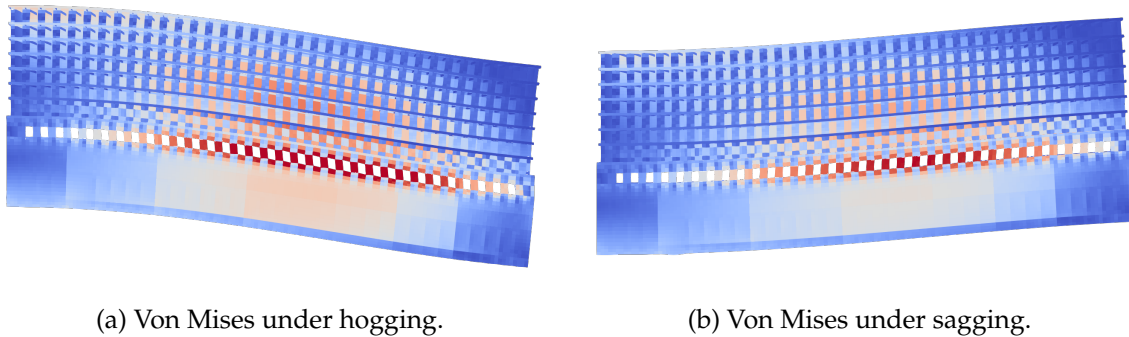


Figure 2.8: Von Mises criterion for the midship section under hogging load condition, on the left, and sagging on the right.

Table 2.1 presents the details of the parameterization of the model, along with the thickness domains.

Table 2.1: Parameters description of the midship section test cases. Thickness values are in mm.

P. section	Description	# patches	Default	Min.	Max.	$ \mathcal{D}_i $
$\mathbf{x}_1$	bottom and inner bottom	72	14	12	20	14
$\mathbf{x}_2$	decks from 2 to 12	396	5	5	15	5
$\mathbf{x}_3$	external bulkheads	30	10	8	15	14
$\mathbf{x}_4$	internal bulkheads	42	5	5	15	8
$\mathbf{x}_5$	shell plating	42	8	8	15	8

A NASTRAN run on this model, on a node of Fincantieri S.p.A.’s compute cluster facility, takes about 1 minute.

Since we require a population of different models, but still related to each other, to be able to evaluate techniques based on [machine learning \(ML\)](#), the designers provided a large number of variants of the midship section. To produce a variant of the base model, the designers replicate or eliminate the structural elements which naturally extend along the dimensions being changed. For the x axis, the length is modified in steps of 5.44 m by extruding decks, walls, and shells. This determines the addition of a new cell in the structure of secondary stiffeners. For the y axis, the modification is limited to the widening of the lower section (enclosed by the shell) and uses a step of size 0.67 m. Finally, for the z axis, the model is changed by the addition or removal of entire decks, modifying the height in steps of 3.06 m.

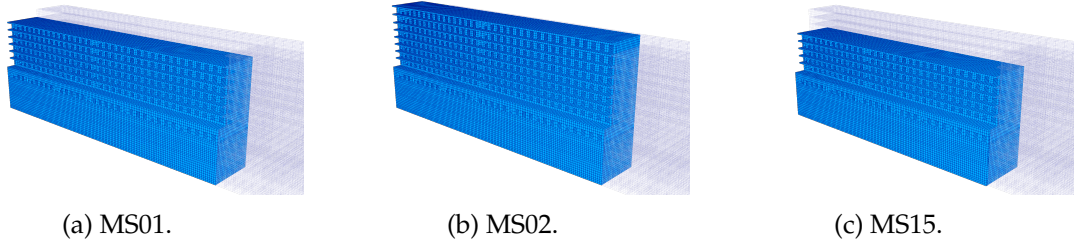


Figure 2.9: Three midship section variants models, depicted in light blue, and the default model represented with transparency. On the left, MS01 with the top deck removed. At the center, MS02 with one extra deck. On the right, MS15 with a shorter length, wider bottom section and two decks removed.

The parameterized sections are extruded accordingly, so that all variants share the same set of decision variables, albeit with different proportions of assigned patches. We remark that these changes are fundamentally different from a simple scaling of the model, and give rise to vastly different structural behaviors. Figure 2.9 shows three different variants, depicted together with the default model (in transparency) to highlight their differences.

Table 2.2 presents the details of the base midship section and the 20 variants that are considered in this study. The default model is denoted MS00, while models MS01 through MS06 each add or remove a single structural element along the axes. Models MS07 through

Table 2.2: Midship section variants features.

Code	Δ_x	Δ_y	Δ_z	# elements	# patches	m_{fixed} [t]	$m_{\text{param. LB}}$ [t]
MS00	0	0	0	75192	582	975.55	1108.21
MS01	0	0	-1	69688	534	920.55	1035.41
MS02	0	0	1	80696	594	1027.81	1181.01
MS03	0	-1	0	73947	582	957.46	1086.15
MS04	0	1	0	76437	582	992.52	1130.27
MS05	-1	0	0	71186	582	927.28	1044.89
MS06	1	0	0	79198	679	1021.77	1171.54
MS07	1	-1	1	83695	693	1062.30	1225.18
MS08	-1	1	0	72369	618	944.09	1065.68
MS09	-1	0	-1	65982	534	875.11	976.25
MS10	1	1	-1	74701	665	982.24	1117.90
MS11	2	0	2	96569	749	1216.84	1397.10
MS12	1	2	-1	76008	665	1000.49	1141.22
MS13	-1	2	-1	72178	570	955.05	1079.53
MS14	2	0	0	84209	679	1092.67	1234.87
MS15	-2	1	-2	58095	522	786.40	874.17
MS16	1	0	-2	67590	567	907.87	1020.04
MS17	-1	0	-2	62939	486	849.03	942.84
MS18	-2	2	1	73908	630	948.42	1085.12
MS19	2	2	2	99347	791	1256.01	1446.26
MS20	-1	2	0	73552	618	960.90	1086.48

MS20 were randomly sampled in the full domain using a maximin approach.

The resulting meshes cover a range between 58095 and 99347 shell elements, with number of patches between 534 and 791. Overall, the m_{fixed} values range from 786.40 and 1256.01 t, while the $m_{\text{param. LB}}$ go from 874.17 to 1446.26 t.

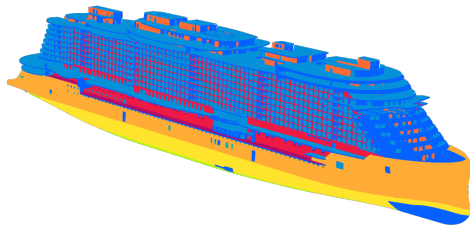
For all variants, the designers set $y_{\text{crit}} = 200$ and $b_{\text{crit}} = 4000$. For the base model MS00, the upper limit for the vertical deflection is set to 140 mm and the $\text{VCG}_{\text{crit}} = 15$ m. The critical values for the other variants are adapted according to the changes in geometry and are discussed in Section A.1.

2.3.2 Full ship fleet

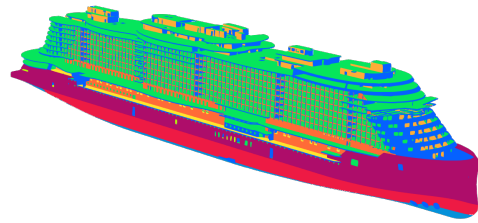
This dataset is composed of full ship models from multiple classes, with different dimensions and features such as number of decks and architectural elements (indoor and outdoor activity areas, technical environments, cabin sizes, etc.).

Figure 2.10 shows the entire group of ships, and Table 2.3 shows the details of each model. Due to confidentiality of industrial data, the dimensions are shown as relative to the dataset's mean and the ratio between $m_{\text{param. LB}}$ and m_{fixed} is reported in percentage.

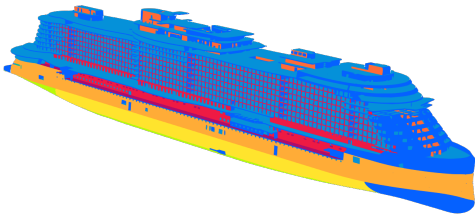
Two main groups can be identified, based on size and proportions. The first group comprises FS0, FS1, FS2, FS7, and FS9, which are medium to large ships. They all have above average length, and are much wider and taller compared to the remaining ships.



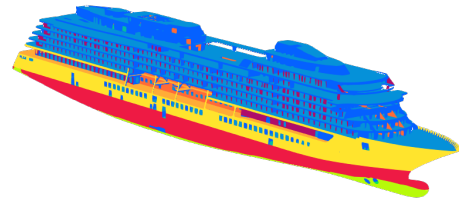
(a) FS0 with 9 parameters.



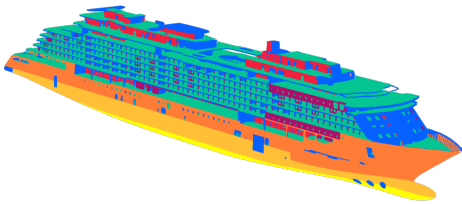
(b) FS1 with 9 parameters.



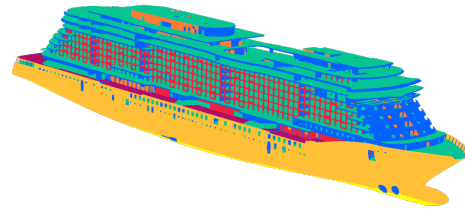
(c) FS2 with 9 parameters.



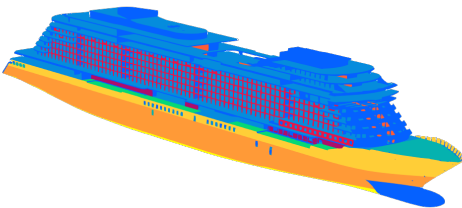
(d) FS3 with 9 parameters.



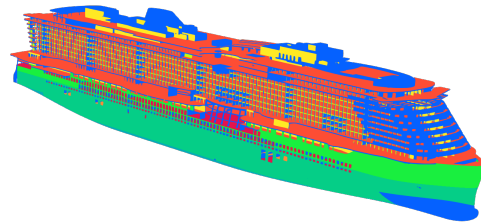
(e) FS4 with 8 parameters.



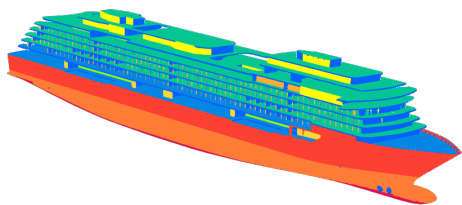
(f) FS5 with 8 parameters.



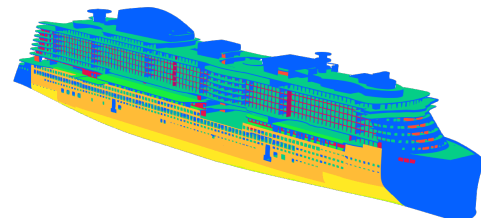
(g) FS6 with 10 parameters.



(h) FS7 with 11 parameters.



(i) FS8 with 12 parameters.



(j) FS9 with 11 parameters.

Figure 2.10: The full ship models, with their initial parameterizations. Each color corresponds to a different parameterized section, except for dark blue which represents elements with fixed thickness. The models are not to scale.

Table 2.3: Details of the full ship models, for their basic parameterization.

Code	Length	Width	Height	# elements	# patches	n_{dim}	$\frac{m_{\text{param. LB}}}{m_{\text{fixed}}} [\%]$
FS0	1.01	1.13	1.09	693237	7127	9	86.32
FS1	1.10	1.13	1.09	761700	8517	9	66.44
FS2	1.18	1.13	1.16	821483	8972	9	82.63
FS3	0.82	0.74	0.78	352400	3870	9	83.24
FS4	0.92	0.80	0.86	404059	4477	8	92.95
FS5	0.88	0.81	0.86	439859	3579	8	129.91
FS6	0.97	0.81	0.91	485736	5006	10	71.45
FS7	1.18	1.39	1.29	1342584	9965	11	71.26
FS8	0.84	0.81	0.80	385484	3322	12	81.61
FS9	1.11	1.24	1.15	986323	7836	11	63.76

The computational meshes reflect this, with the smallest FS0 composed of 693237 elements and the largest FS7 almost doubling the count at 1342584. In terms of patches, the range is more contained and spans from 7127 to 9965. The smaller group is composed of FS3, FS4, FS5, FS6, and FS8, all having less than 500000 elements. The range of elements counts goes from 352400 to 485736, while the patches go from 3579 to 5006. A noticeable difference between the two groups concerns the geometry of the bow: the smaller ships have a very pronounced bulbous bow, with the shell receding a lot, while the larger ships have an almost vertical bow profile.

The dataset includes two groups of sister ships, which is representative of a common industrial design process where an initial design serves as the basis for a sequence of derivative ships. The first group consists of FS0, FS1, and FS2, while the second comprises FS5 and FS6. Within each group, the sister ships show an increase in length relative to the original design.

The NASTRAN solve times for the smaller ships range from 5 to 7 minutes, while the larger ships require from 16 minutes to one full hour.

Since the values chosen for the thresholds are considered confidential data by Fincantieri S.p.A., they will not be disclosed for this dataset.

2.4 Conclusions

This chapter introduced the structural analysis problem that forms the focus of this thesis, describing the FEA setup used to evaluate the structural response of cruise ships under extreme wave loading. In the context of the preliminary design phase, the designers aim to identify a configuration of steel thicknesses for the primary structural elements (decks, bulkheads, etc.) able to withstand the loads of navigation, but at the same time, reduce the total mass of the ship.

Five QoIs summarize the results of a simulation: other than the total mass of the ship f_{mass} , which will be the objective during optimization, we use the VCG and the three measures of structural resilience n_y , n_b , and $f_{\text{deflection}}$ to constrain the possible configurations. The notation for the QoIs, the domain, the configurations of steel thicknesses and the constraints were then established.

Finally, we presented the test cases for the numerical evaluation of the techniques that will be developed. As an inexpensive but illustrative test case, we use a simplified midship section and a moderate number of variants obtained by addition or removal of decks and longitudinal or transverse sections. For realistic full-scale test cases, we selected a group of 10 commercial designs that is representative of industry practices. The models range from small vessels to very large ships, and the dataset contains two sets of sibling ships.

The high computational cost of FEA, that even for the midship sections requires several seconds, prevents the use of automated procedures for mass optimization. Chapter 3 will present the model order reduction approach that will enable all the computational techniques developed in this thesis.

SURROGATES CONSTRUCTION

As seen in Section 2.3.2, the scale of the full ship test cases implies that, even for the smaller instances, a designer needs to wait several minutes to obtain the results of a [finite element analysis \(FEA\)](#) simulation. This cost in terms of time, although not high compared to other types of industrial analyses, is nonetheless an obstacle for an efficient design process. On the one hand, when a simulation results in unsatisfying [quantities of interest \(QoIs\)](#), the time investment has been a waste and the designers are steered towards more conservative designs instead of freely experimenting with alternative solutions. On the other hand, the large latency makes operating small changes cumbersome and error-prone due to the necessary context switch. Moreover, modern commercial software often imposes strict licensing terms, which limit the number of concurrent analyses and restrict the hardware resources that can be used to accelerate the computation.

[Model order reduction \(MOR\)](#) [39, 22, 20, 21, 30] is the paradigm of approximating an expensive-to-compute function with a much cheaper surrogate, trading off the accuracy of the results with the possibility of greatly increasing the rate of evaluation. The use of such [reduced order models \(ROMs\)](#) in place of the expensive [FEA](#) or [computational fluid dynamics \(CFD\)](#) solvers during the simulation-based design process gives rise to the so-called surrogate-based modeling, design and optimization literature [149, 73, 38, 85, 173].

In this work, we use [proper orthogonal decomposition \(POD\)](#) for the dimensionality reduction of the stress tensor field, together with [Gaussian process regression \(GPR\)](#) for the latent space prediction of the [POD](#) coefficients and vertical deflection. The predicted stress tensors are post-processed to compute the yielding and buckling states of the full-scale mesh with great accuracy and low computational cost. Thanks to these surrogates, the designers can test new configurations in quasi-real time, and automated optimization becomes feasible.

This chapter is structured as follows: Section 3.1 discusses different approaches to [MOR](#), and presents the details of [POD](#), [GPR](#), and a multi-fidelity alternative in the [auto-regressive non-linear Gaussian process \(NARGP\)](#). Section 3.6 shows the numerical results on some representative test cases from Section 2.3.

3.1 Introduction

The fundamental paradigm of MOR is based on the separation between an offline phase, where expensive simulation data is collected and analyzed to construct a predictor, and the online phase where the computationally cheaper predictor is used in place of the original simulator. A broad distinction among the approaches that fall under the umbrella of MOR can be traced between techniques for dimensionality reduction and methods for regression.

Dimensionality reduction tackles the problem of compressing high-dimensional data, such as large vectors representing 2D and 3D fields, to a much lower dimensional representation. Efficient mappings between the original and latent spaces then enable the construction of predictors defined on a much smaller number of degree of freedom, possibly greatly simplifying the regression task.

Classical dimensionality reduction techniques rely on restriction of the full-scale data to a linear manifold, with a vector base derived from an eigenvalue problem on the empirical correlation of simulation data. The discrete POD, principal components analysis, and Karhunen-Loeve expansion (KLE) are all based on the singular value decomposition (SVD) of a snapshot matrix [194, 30, 207]. Linear projection methods have seen wide application to engineering problems, due to their simplicity and efficiency. However, advection and strong non-linearity in the behavior of the system often require too many latent dimensions, as captured by the Kolmogorov n-width [146], limiting the efficacy of such methods.

Non-linear projection methods, such as the use of kernels for the computation of correlations [165, 164], hyper-reduction approaches [64, 206, 13, 36], or deep learning approaches such as autoencoders [157], have been applied to more complex dynamics.

Complementary to the problem of learning compact representations is the task of approximating the relationship between the system inputs and the corresponding quantities of interest. Depending on the application, this may involve learning a mapping from the parameter space to scalar outputs or reduced representations, or approximating the evolution operator that maps initial and boundary conditions to the system response.

Historically, the simplest approaches relied on the construction of polynomial approximations, commonly referred to as response surface method (RSM) [106]. More flexible regression techniques have subsequently emerged, including radial basis function (RBF) interpolation [205] and other non-parametric methods. Among these, GPR [155, 85], also known as Kriging, has become one of the most widely adopted techniques in engineering applications due to its robustness, its ability to provide uncertainty estimates alongside predictions, and its effectiveness in data-scarce settings.

A particularly important class of reduced-order models arises when the governing equations of the high-fidelity system are explicitly available. In this setting, dimensionality reduction can be combined with projection techniques to restrict the original system to a low-dimensional subspace while preserving its physical structure. This family of methods is commonly referred to as reduced basis (RB) [92] methods and has achieved considerable

success in computational mechanics and parametric [partial differential equations \(PDEs\)](#).

Data-driven approaches are particularly attractive in industrial settings, as they can be applied to proprietary simulation software without requiring access to the underlying implementation. This provides maximum flexibility while avoiding the development, maintenance, and validation of specialized in-house solvers. As discussed in Chapter 2, the high-fidelity solver considered in this work is MSC NASTRAN, a commercial closed-source finite element code.

To effectively support the design process, the optimization pipeline must provide fast predictions not only for scalar [QoIs](#), but also for spatially distributed quantities such as stress tensors and buckling usage factors. The availability of such surrogates enables the interactive exploration of design alternatives without the need for additional high-fidelity simulations.

3.2 Proper orthogonal decomposition

[POD](#) aims to decompose a field, dependent on multiple variables, into a linear combination of a set of basis vectors and coefficients dependent on a separate subset of the original variables. The method is applied to snapshots data collected from high-fidelity simulations, and the construction of the vector base guarantees optimality (in the sense of Frobenius norm) of the reconstruction error when the least important basis elements are discarded. The most common application is to the data-driven dimensionality reduction of spatial fields obtained from the numerical solution of parametrized [PDEs](#).

In our setting, we consider the prediction of the stress tensor components parameterized by the thickness values of the parameterized sections. A single snapshot consists of 6 scalar fields, associating a component of the stress tensor to the centroid of each shell element in the computational mesh. With [POD](#), we construct a spatial basis defined on the elements' centroids, while the thickness configuration only determines the values of the coefficients.

We denote as $\mathbf{S} \in \mathbb{R}^{n \times m}$ the real matrix that collects the snapshots $\mathbf{s}$ of a scalar field, so that n is the number of degrees of freedom and m is the number of snapshots, with $m \ll n$. The $\mathbf{S}_{ij}$ element depends on the i -th degree of freedom and the j -th thickness configuration $\mathbf{x}^{(j)}$ in the sampling plan that generated $\mathbf{S}$.

Following [30], we take the economy [SVD](#) of $\mathbf{S}$ to write

$$\mathbf{S} = \mathbf{U} \mathbf{\Sigma} \mathbf{V}^T, \quad (3.1)$$

where $\mathbf{U} \in \mathbb{R}^{n \times m}$ and $\mathbf{V} \in \mathbb{R}^{m \times m}$ are column-orthonormal matrices, and $\mathbf{\Sigma} \in \mathbb{R}^{m \times m}$ is a diagonal matrix. The columns of $\mathbf{U}$ are the left singular vectors of $\mathbf{S}$, that is, the eigenvectors of the empirical covariance matrix $\mathbf{S} \mathbf{S}^T$. Conversely, the columns of $\mathbf{V}$ are the right singular vectors of $\mathbf{S}$ and the eigenvectors of $\mathbf{S}^T \mathbf{S}$, which represents the covariance of the snapshots. The two empirical covariance matrices share the non-zero eigenvalues, and their square roots give the singular values that form the diagonal of $\mathbf{\Sigma}$, ordered by decreasing module. The decomposition is unique, up to a reordering of the columns of $\mathbf{U}$ and $\mathbf{V}$ associated to

equal singular values, or a change of sign in the columns of $\mathbf{U}$ and $\mathbf{V}$. The matrix $\mathbf{U}$ can be interpreted as a basis for the columns of $\mathbf{S}$, ordered by decreasing contribution to the reconstruction.

Indeed, dimensionality reduction is obtained by considering only the first $r < m$ elements of the basis, to write

$$\mathbf{S} \approx \tilde{\mathbf{S}} = \tilde{\mathbf{U}}\tilde{\Sigma}\tilde{\mathbf{V}}^T, \quad (3.2)$$

where $\tilde{\mathbf{U}} \in \mathbb{R}^{n \times r}$ is the matrix formed by the r leftmost columns of $\mathbf{U}$, $\tilde{\Sigma} \in \mathbb{R}^{r \times r}$ is the diagonal matrix of the r largest singular values in descending order, and $\tilde{\mathbf{V}} \in \mathbb{R}^{m \times r}$ the matrix with the r leftmost columns of $\mathbf{V}$. The error obtained in the reconstruction of the snapshot matrix with the truncated modes can be quantified, in Frobenius norm, as

$$\|\mathbf{S} - \tilde{\mathbf{S}}\|_F^2 = \|\mathbf{S} - \tilde{\mathbf{U}}\tilde{\mathbf{U}}^T\mathbf{S}\|_F^2 = \sum_{i=r+1}^m \sigma_i^2, \quad (3.3)$$

and $\tilde{\mathbf{S}}$ coincides with the optimal linear reconstruction. This reconstruction error is commonly referred to as "residual energy" [194]. Equation (3.3) is often used to choose r so that the residual energy is below a relative threshold, usually below 1%. Other criteria for the choice of r can be the given as a constraint on the relative magnitude of the first discarded singular value, e.g., $\sigma_{r+1} \leq \sigma_1/100$, or derived from more sophisticated assumptions on the data [30].

To each snapshot $\mathbf{s}(\mathbf{x}^{(j)})$ is associated a vector of reduced coefficients $\mathbf{c}(\mathbf{x}^{(j)})$ such that

$$\mathbf{s}(\mathbf{x}^{(j)}) \approx \tilde{\mathbf{s}}(\mathbf{x}^{(j)}) = \tilde{\mathbf{U}}\mathbf{c}(\mathbf{x}^{(j)}). \quad (3.4)$$

For applications where the constraints on $\mathbf{s}$ are given in an analytical and treatable form, we can impose Equation (3.4) with an equality instead of an approximation, to restrict the solutions to an r -dimensional manifold embedded in the n -dimensional space. A common setting for this intrusive approach is the **RB** method [92], which in the case of the **FEA** discretization of static linear elasticity simply projects the stiffness matrix and right hand side unto the reduced space.

When such intrusive approaches are not viable, it is necessary to first obtain the reduced coefficients via a projection of the original snapshot

$$\mathbf{c}(\mathbf{x}^{(j)}) = \tilde{\mathbf{U}}^T \mathbf{s}(\mathbf{x}^{(j)}), \quad (3.5)$$

then construct a regression from the parametric domain to the reduced space.

3.3 Gaussian process regression

GPR is a non-parametric regression technique based on the interpretation of data as a multivariate Gaussian vector, where the covariance is a kernel function fitted to the data. Predictions are obtained by conditioning the expectation on the observed data, with more advanced kernels accounting for the possibility of epistemic noise, periodic domains,

multiple fidelity levels, combinatorial domains, etc. [GPR](#) is the preferred tool for Bayesian optimization, thanks to its uncertainty quantification capabilities, and this application has driven much of the developments in kernels [107, 11] and numerical implementations.

This section follows the expositions from [113] and [85].

Let $\mathbf{y}$ be a vector of observations $\mathbf{y}_i = f(\mathbf{x}^{(i)})$ for $i = 1, \dots, m$, with $\mathbf{y} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{K})$. The vector of marginal means $\boldsymbol{\mu}$ and the covariance matrix $\mathbf{K}$ are defined as:

$$\boldsymbol{\mu}_i = \mathbb{E}[\mathbf{y}_i] = \mathbb{E}[f(\mathbf{x}^{(i)})], \quad (3.6)$$

$$\mathbf{K}_{ij} = \mathbf{K}_{ji} = \mathbb{E}[(\mathbf{y}_i - \boldsymbol{\mu}_i)(\mathbf{y}_j - \boldsymbol{\mu}_j)] = \mathbb{Cov}[\mathbf{y}_i, \mathbf{y}_j] = \mathbb{Cov}[f(\mathbf{x}^{(i)}), f(\mathbf{x}^{(j)})]. \quad (3.7)$$

The conditional distribution of a subset of observations L , on past outcomes H , can be obtained by leveraging the joint distribution

$$\begin{bmatrix} \mathbf{y}_H \\ \mathbf{y}_L \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_H \\ \boldsymbol{\mu}_L \end{bmatrix}, \begin{bmatrix} \mathbf{K}_{HH} & \mathbf{K}_{HL} \\ \mathbf{K}_{LH} & \mathbf{K}_{LL} \end{bmatrix}\right), \quad (3.8)$$

to obtain

$$\mathbf{y}_{L|H} \sim \mathcal{N}(\boldsymbol{\mu}_{L|H}, \mathbf{K}_{L|H}), \quad (3.9)$$

where the conditioned terms are expressed in terms of mean and covariance matrix of the process' distribution as

$$\boldsymbol{\mu}_{L|H} = \boldsymbol{\mu}_L + \mathbf{K}_{LH}\mathbf{K}_{HH}^{-1}(\mathbf{y}_H - \boldsymbol{\mu}_H), \quad (3.10)$$

$$\mathbf{K}_{L|H} = \mathbf{K}_{LL} - \mathbf{K}_{LH}\mathbf{K}_{HH}^{-1}\mathbf{K}_{HL}. \quad (3.11)$$

The crucial component of a [GPR](#) is the construction of two functions $\mu(\mathbf{x}^{(i)})$ and $\text{kern}(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$ to generate the entries of $\boldsymbol{\mu}$ and $\mathbf{K}$, respectively, so that past observations are reproduced with high confidence and extrapolation beyond the observed instances gives a coherent probability distribution. The mean function $\boldsymbol{\mu}$, which encodes prior knowledge about the output, is usually set to zero. Indeed, in Equation (3.10) the covariance-dependent terms act as a correction, weighting the observed deviations from the prior.

For the covariance function, which is required to generate a symmetric semi-definite positive matrix, a popular choice is the squared exponential kernel with [automatic relevance determination \(ARD\)](#):

$$\mathbf{K}_{ij} = \text{kern}(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \sigma^2 \exp\left(-\frac{1}{2} \sum_d \frac{(\mathbf{x}_d^{(i)} - \mathbf{x}_d^{(j)})^2}{l_d^2}\right), \quad (3.12)$$

where σ^2 is a scaling factor, and l_d is the length scale associated to the dimension d of the parametric domain. [ARD](#) is a primary factor for the efficacy of [GPR](#) for higher n_{dim} [26, 96]. Dimensions for which variations in the input have little influence on the output are associated with large length scales, decreasing their effect on the covariance. Conversely, dimensions in which small input variations produce significant changes in the output are assigned shorter length scales, restricting the influence on the output to nearby samples.

An alternative kernel function, which generalizes Equation (3.12), is the Matérn kernel

$$\text{kern}_{M\nu}(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\sqrt{2\nu} \sqrt{\sum_d \frac{(\mathbf{x}_d^{(i)} - \mathbf{x}_d^{(j)})^2}{l_d^2}} \right)^\nu K_\nu \left(\sqrt{\sum_d \frac{(\mathbf{x}_d^{(i)} - \mathbf{x}_d^{(j)})^2}{l_d^2}} \right), \quad (3.13)$$

where ν constrains the smoothness of the interpolation and is usually chosen to be 1.5 or 2.5, Γ is the gamma function generalizing the factorial, and K_ν is the modified Bessel function of the second kind.

The values of the hyper-parameters for Equations (3.12) or (3.13) are chosen by maximizing the log-likelihood on the training data. Let $\mathbf{K}_\theta$ denote the covariance matrix obtained by the chosen kernel, whose hyperparameters are collected in the vector θ . For the squared exponential kernel, $\theta = [\sigma, l_1, \dots, l_{n_{\text{dim}}}]$. For the case of zero prior, with $\mathbf{X} = [\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}]$, the log-likelihood reads as follows

$$\log p(\mathbf{y} | \mathbf{X}, \theta) = -\frac{m}{2} \log 2\pi - \frac{1}{2} \log |\mathbf{K}_\theta| - \frac{1}{2} \mathbf{y}^T \mathbf{K}_\theta^{-1} \mathbf{y}, \quad (3.14)$$

where $|\mathbf{K}_\theta|$ is the determinant of the covariance matrix.

Once the hyper-parameters have been selected, the values in θ are kept constant and the term $\mathbf{K}_{HH}^{-1}(\mathbf{y}_H - \mu_H)$ in Equation (3.10) can be computed and stored for online querying of the GPR. During a query, the computation of the conditioned mean requires only evaluating the kernel function between the snapshots samples and the query points. Figure 3.1 represents the prediction of a GPR for the Forrester function [73], showing how the addition of a new sample to the training data modifies the outcome.

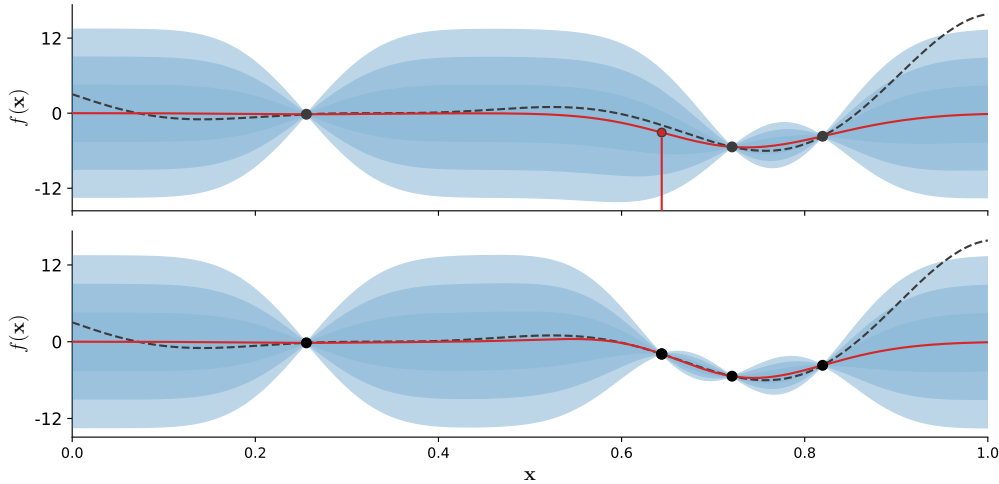


Figure 3.1: Representation of a GPR prediction before and after its update. The training samples are shown as black dots, with the sample in red absent from the training in the top figure, and included in the bottom one. The true function is represented with a dashed black line, while the GPR predictions are in red. The confidence level corresponding to σ , 2σ , 3σ are represented in increasingly lighter shades of blue.

For the prediction of the vector of POD coefficients $\mathbf{c}(\mathbf{x})$, a possible approach is to fit a separate GPRs for each component, for a total of r sets of hyper-parameters. As a cheaper,

but less powerful alternative, a single vector-valued **GPR** can be obtained by making all components share the same hyper-parameters set.

3.4 Multi-fidelity regression

Auto-regressive non-linear Gaussian process with active subspaces (NARGPAS) is a multi-fidelity regression technique proposed in [159]. In a multi-fidelity setting, a regressor is trained on a combination of data with varying levels of accuracy, for example: the highest fidelity could come from a **FEA** simulation with a very fine mesh and complex constitutive assumptions, while a lower fidelity level could use a **RB** surrogate. A common assumption is that lower fidelity levels provide datasets with more samples, since the generation of less accurate data is generally cheaper. By modeling the relationship between the fidelity levels, it is possible to leverage this discrepancy in data availability and achieve a much higher accuracy than possible with the lower fidelity data alone. In the **GPR** literature, the deep **Gaussian process (GP)** approach constructs a sequence of **GPs** where the mean function is obtained from the immediately lower level.

An extension is the **NARGP** proposed in [144], where instead of simply recursively propagating the mean from lower levels, the kernel at each level combines the covariance of the lower-level predictions with the covariance of the current level. Let

$$S_q = \{\mathbf{x}^{q,i}, y^{q,i}\}_{i=1}^{n_q} \subset \mathcal{D} \times \mathbb{R}, \quad \text{for } q = 1, \dots, p, \quad (3.15)$$

where q denotes the fidelity level, with $q = 1$ being the lowest, and $y^q = f_q(\mathbf{x})$ is the observation at the corresponding accuracy. As an additional assumption, let the datasets be structured hierarchically

$$\pi(S_p) \subset \pi(S_{p-1}) \subset \dots \subset \pi(S_1), \quad (3.16)$$

where π is the projection unto the first n_{dim} coordinates. The covariance kernel function at level $q > 1$ is given by three terms separate terms

$$k_q(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = k_q^\rho(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) k_q^f(f_{q-1}(\mathbf{x}^{(i)}), f_{q-1}(\mathbf{x}^{(j)})) + k_q^\delta(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}), \quad (3.17)$$

where k_q^δ models the intra-level covariance and the product $k_q^\rho k_q^f$ models the inter-level interaction. Each term has separate hyper-parameters to be fitted via likelihood maximization. This approach increases the complexity of the construction and the outputs are no longer Gaussian distributed, but the resulting model is able to express more complex functions.

In [159], **NARGP** is used to include the assumption of low-dimensional **active subspace (AS)** in the original input space. **AS**, discussed in [45, 44], is a widely used technique for the dimensionality reduction of parameter spaces which leverages the covariance matrix of gradients in a similar way to **POD**. In particular, **AS** constructs a basis for the gradients of the function of interest using **SVD** and discards the least energetic modes.

By constructing a gradients snapshot matrix $M \in \mathbb{R}^{n_{\text{dim}}, m}$ with the i -th column given by $\nabla f(\mathbf{x}^{(i)})$, let $\mathbf{W}$ be the matrix collecting the r most energetic AS modes obtained from SVD. The retained modes are able to maximally reconstruct the directions of highest variability, while variations along the discarded modes result in little variation of f .

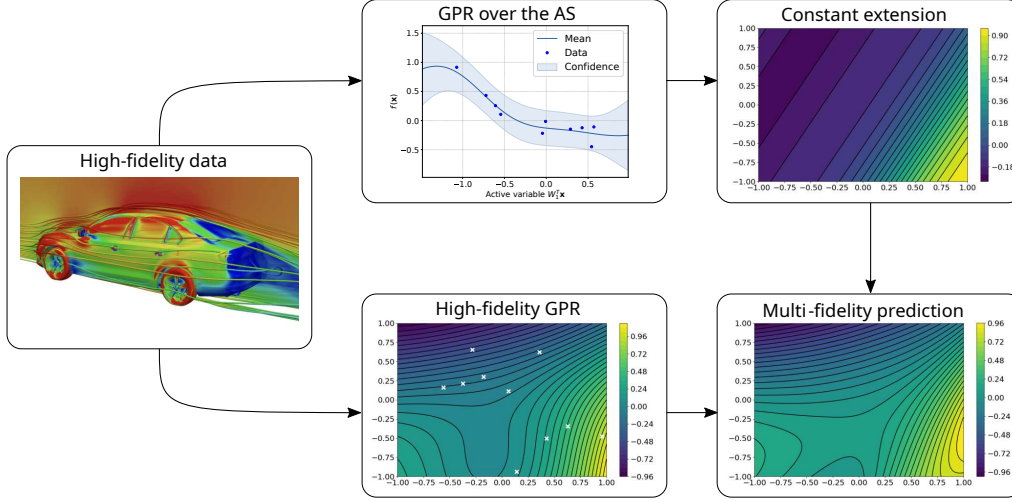


Figure 3.2: Representation of NARGPAS on an automobile simulation. Image courtesy of the authors of [159].

NARGPAS first constructs a GPR on the data from the highest fidelity level, and uses it to compute the gradients corresponding to the train samples. The gradients are used to compute the AS modes, of which only the first is retained. As a second and lowest fidelity level, NARGPAS trains a single-input GPR using the projection of the training data unto the first AS mode. This correspond to a constant extension of the prediction along the directions orthogonal to the AS mode, as a consequence of the approximation of low variability.

NARGP was used in an early version of the optimization pipeline [190], achieving good results in term of accuracy on both a simplified midship section and a full ship.

3.5 Full surrogate

To construct a surrogate for the stress tensor field, we use POD for dimensionality reduction and GPR for the prediction of reduced coefficients. For each combination of load condition and field, we construct a separate POD and vector-valued GPR, for a total of 12 surrogates. To obtain n_y and n_b , rather than training a simple GPR, we post-process the full-scale stress tensor field of each load condition and aggregate the results. Although more computationally expensive, the vectorized computation of yielding and buckling criteria achieves much lower prediction error. The vertical deflection is instead well predicted by a simple GPR. For all GPRs, we use the squared exponential kernel from Equation (3.12) with ARD. In the case of reduced coefficients, for each combination of load condition and stress tensor component, we use vector-valued GPRs that share the kernel hyper-parameters

across all **POD** modes. This approach results in a slightly higher prediction error, but is much cheaper in computation time compared to fitting separate hyper-parameters for each mode.

Regarding the dimensionality reduction, we observe that the decay of singular values consistently achieves relative residual energy, quantified by Equation (3.3), below 1% for rank r higher than, but close to, the number of parameters. This behavior results in an effective reduction and a cheap training phase for the **GPRs**. Additionally, the construction of the **GPRs** for each stress tensor component and load condition is embarrassingly parallel, once the tensor fields are decomposed and the basis for each component is stored.

Overall, we obtain a surrogate for the full set of **QoIs** that is cheap to construct and evaluate, suitable for our optimization pipeline.

3.6 Numerical results

In this section, we evaluate our full surrogate on two test cases from Chapter 2: the default midship section MS00 is studied in Section 3.6.1 and the full ship FS6 in Section 3.6.2. We consider different sizes of snapshots database and parameterizations, to verify the robustness of the proposed approaches as the optimization proceeds. For each test case, we discuss the selection of the **POD** truncation rank, and present the trade-offs between accuracy and computational cost of multiple predictors for the reduced coefficients.

The implementation of **GPR** comes from GPy [84], and the **POD-GPR** framework used is from EZyRB [56]. All values are obtained using a full node of Fincantieri S.p.A.'s HPC facility, equipped with an Intel Xeon Gold 6354 CPU (3.00 GHz) featuring 36 cores and 512 GB of RAM.

3.6.1 Midship section MS00

We apply **POD** to the high-fidelity snapshots of the separate components of the stress tensor. Figures 3.3 shows the **POD** modes associated to the three most energetic singular values of σ_x and τ_{xz} , computed under the hogging load condition.

For the choice of truncation rank, we analyze the singular values decay from a random sampling and choose the last rank above the 1% threshold of the first singular value. This choice leads to good reconstruction errors, and is robust with respect to the increase in high-fidelity samples due to the infill criteria from the optimization results. Figure 3.4 shows the singular values decay, for all load conditions and stress tensor components. We consider the initial parameterization with 5 parameters, with the snapshots database at the initial sampling (21 high-fidelity configurations) and at the end of the optimization (96 configurations). Even after the addition of 75 configurations via the infill criteria of the optimizations, the proposed criterion would still select the same truncation rank. We remark that in our experience, the ideal truncation rank is close to the number of parameterized sections, with the difference usually between 1 and 5.

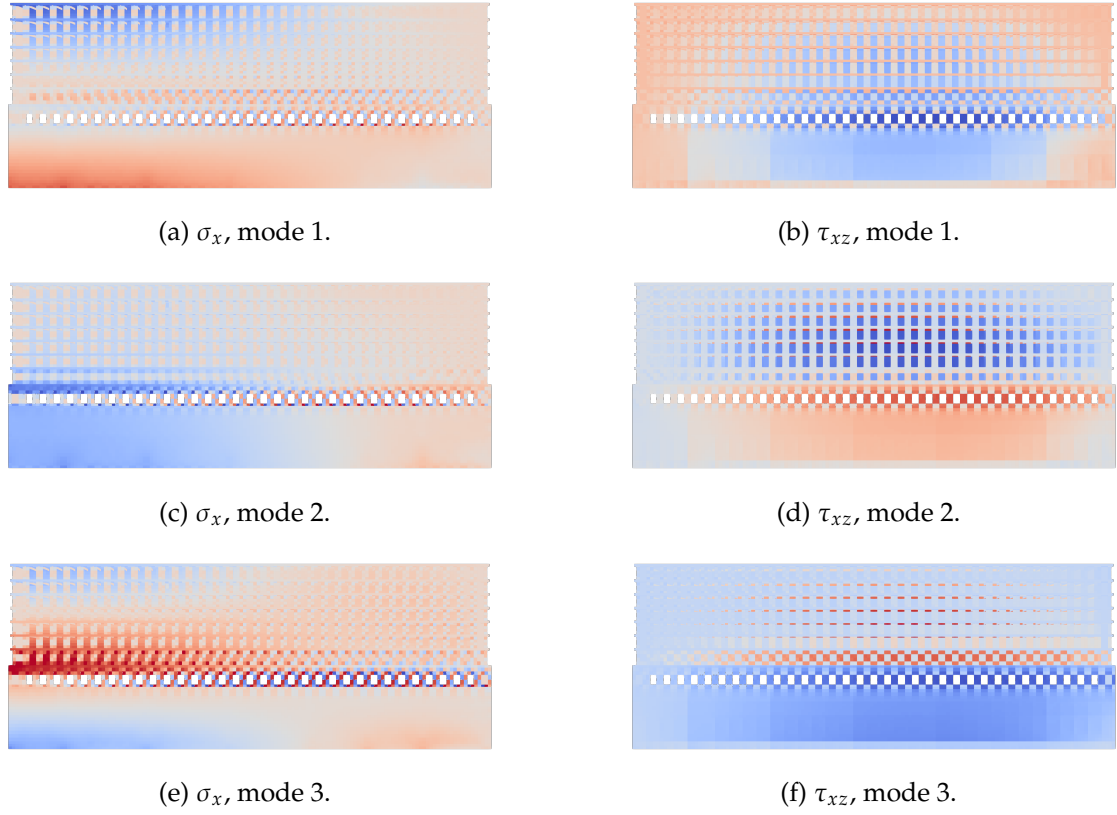


Figure 3.3: Side view of the midship section MS00, depicting the most energetic POD modes from top to bottom, for σ_x (left) and τ_{xz} (right) under hogging. The color scale is symmetric, with blue for negative values and red for positive values.

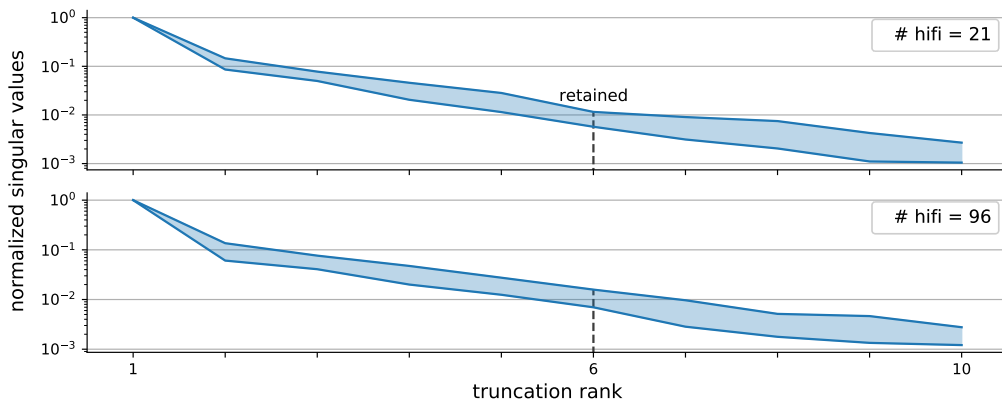


Figure 3.4: Evolution of the range of normalized singular values decay of the stress tensor components. The values come from the high-fidelity database before and after optimization, for the midship section MS00 with 5 parameterized sections.

Figure 3.5 compares the prediction errors on n_y and n_b for different ranks and number of high-fidelity database, obtained from 5-fold cross-validation (5-CV). Due to the large range of these QoIs, we report the error relative to the chosen critical threshold. As the number of high-fidelity samples increases, the prediction errors increase on lower truncation ranks, but the one selected previously consistently scores among the best.

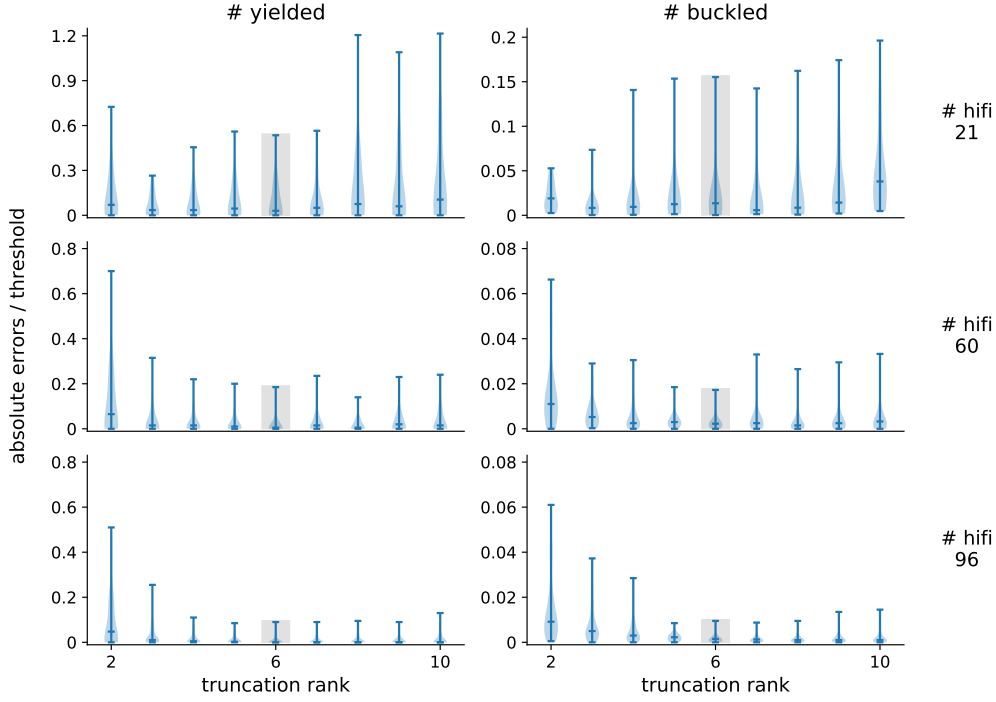


Figure 3.5: Comparison of the 5-CV errors on n_y and n_b , for different database sizes and ranks. The predictions are relative to MS00 with 5 parameterized sections. The plots highlight the median error and the gray box identifies the chosen truncation rank.

To motivate the choice of predictor for the reduced coefficients, we compare multiple approaches in terms of the resulting error. We employ RBF with a multiquadric kernel, GPR with a single squared exponential kernel shared across coefficients (vector-GPR), GPR with a separate kernel for each coefficients (multi-GPR), and NARGPAS. All GPR kernels use ARD, and the hyper-parameters are fitted via log-likelihood maximization using low-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS), generating 20 random initial configurations and retaining the best configuration.

We study the 5-CV relative error, in L2 norm, of the fields used in the prediction of the QoIs. We monitor the stress tensor and the von Mises criterion which are used to obtain the yielding state, as well as the 11 buckling usage factors. We first consider the high-fidelity database of the initial parameterization of MS00, after optimization. Figure 3.6 shows that the multi-GPR approach leads to the best predictions, followed by the vector-GPR, the NARGPAS, and finally the RBF. Overall, the GPR-based surrogates have very good errors that almost always fall below the 0.02 threshold. The most difficult field to reproduce is the usage factor related to $uf_{x+\tau}$, for which both vector-GPR and multi-GPR have a tail

well over the 0.02 threshold.

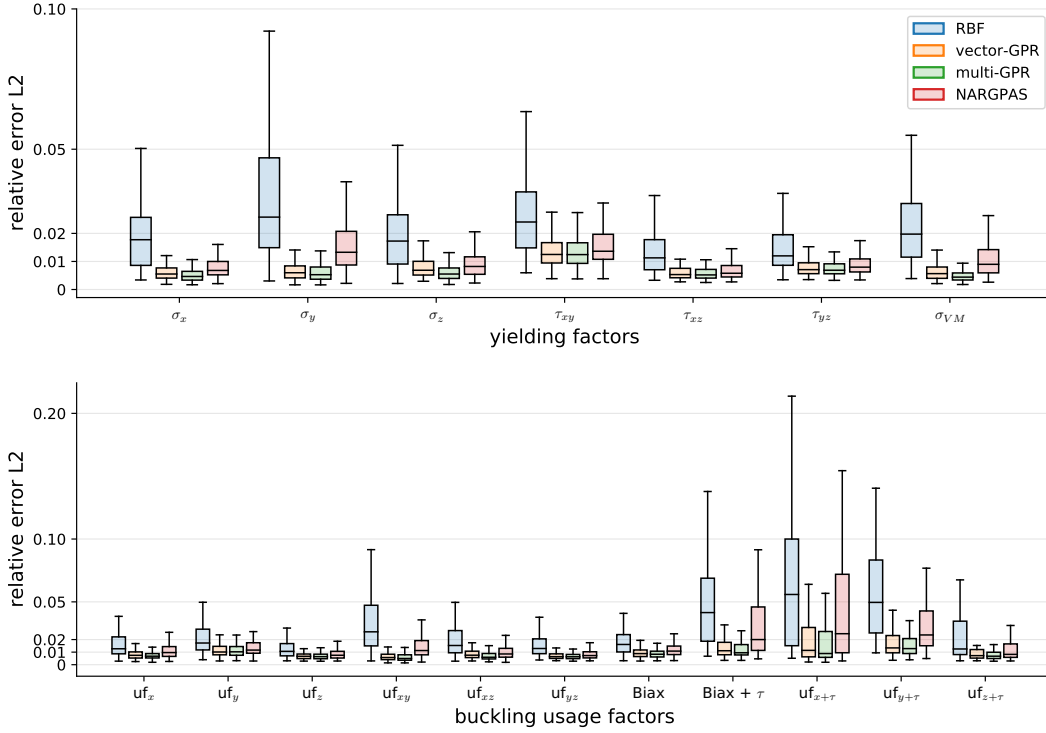


Figure 3.6: Comparison of the 5-CV errors on the stress tensor components and their post-processed results, with medians highlighted, for different predictors of reduced coefficients. The values are relative to the hogging condition, simulated on MS00 with 5 parameterized sections.

Figure 3.7 repeats the comparison on a refined parameterization with 17 sections, truncation rank 18, and 352 configurations in the high-fidelity database at the end of optimization. In this case, the ranking of the predictors remains the same, but the errors reflect the increased complexity of the problem. **NARGPAS** shows much higher errors on the stress tensor components, possibly due to higher difficulty in fitting of the non-linear kernel to the larger and higher-dimensional database. On the 4 last buckling usage factors $\text{Biax}+\tau$, $uf_{x+\tau}$, $uf_{y+\tau}$, and $uf_{z+\tau}$, both vector-**GPR** and multi-**GPR** record errors above 0.02.

However, the good performances of multi-**GPR** come at a high computational cost. While in the case of the basic parameterization the difference in **GPR** training time is negligible, the multi-**GPR** approach becomes infeasible in the case of larger high-fidelity database and higher number of parameterized sections. On the finer parameterization, the training time for the multi-**GPR** associated to a single stress tensor component is about 1 hour, while a vector-valued **GPR** only requires 5 minutes. Since the trade-off in terms of accuracy is limited, we choose this approach for our full surrogates.

Finally, we compare the distributions of the 5-CV errors on n_y and n_b , relative to the respective critical values, obtained via a scalar **GPR** and via our full surrogate. Figure 3.8 shows that, for both initial and refined parameterized sections, at the largest database size,

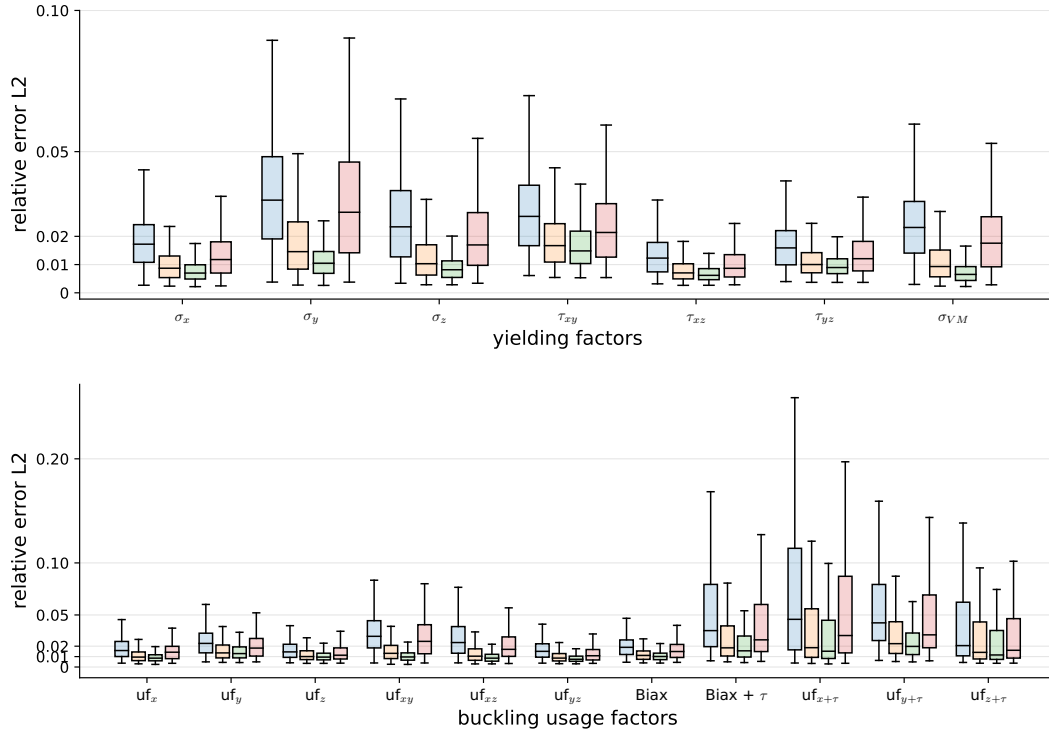


Figure 3.7: Comparison of the 5-CV errors on the stress tensor components and their post-process results, with medians highlighted, for different predictors of reduced coefficients. The values are relative to the hogging condition, simulated on MS00 with 17 parameterized sections.

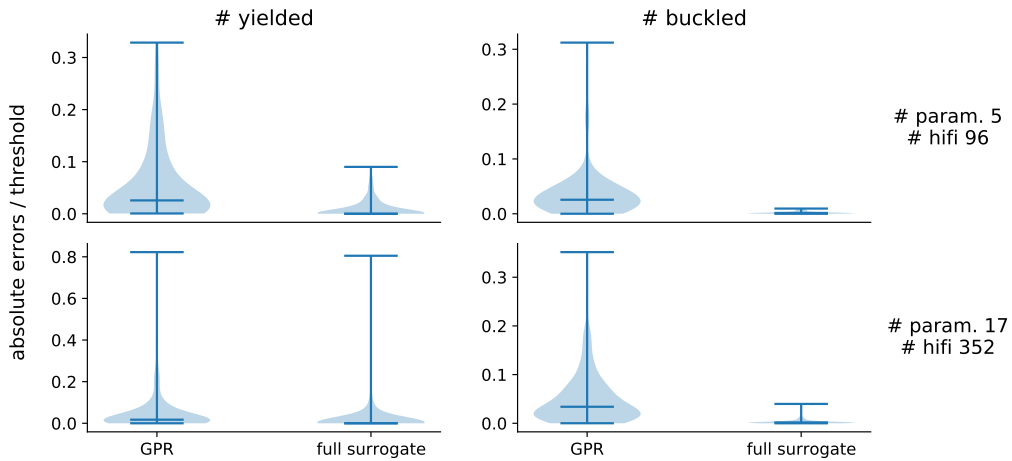


Figure 3.8: Comparison of the 5-CV errors on n_y and n_b , with medians highlighted, for different prediction approaches. The values are relative to MS00, with 5 and 17 parameterized sections.

the full surrogates show a much lower error and warrants its adoption in the optimization pipeline.

Regarding the prediction of the vertical deflection, the simple GPR approach model achieves R^2 score above 0.99 for both the initial and refined parameterizations, measured with 5-CV. In terms of mean absolute percentage error (MAPE), the two models score 0.43% and 0.47% respectively.

Overall, the prediction of the complete set of QoIs on MS00, amortized over 1000 queries executed in parallel, takes on average 0.005 s for both the initial and fine parameterization.

3.6.2 Full ship FS6

We now repeat the analyses of the previous section on a full ship model, FS6, with a basic parameterization of 10 parameterized sections. Although this ship is of medium size, it has much greater variety of structural elements compared with MS00. The most energetic POD modes for σ_x and τ_{xz} , under the hogging load condition, are shown in Figure 3.9.

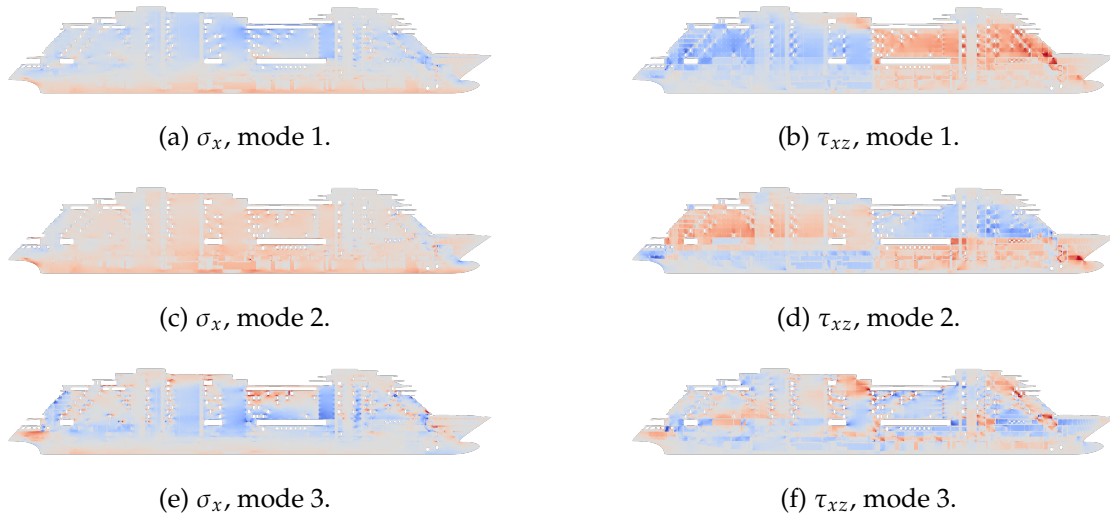


Figure 3.9: Side view of the full ship FS6, depicting the most energetic POD modes from top to bottom, for σ_x (left) and τ_{xz} (right) under hogging. The color scale is symmetric, with blue for negative values and red for positive values.

Due to the increase in complexity, we use a conservative choice for the truncation rank. We retain the first 16 modes, instead of 13 as informed by the criterion proposed in the previous section. Figure 3.10 shows the singular values decay measured on the initial high-fidelity database (21 configurations) and at the end of the optimization (114 configurations). As with the simpler MS00, the criterion's choice remains the same on the two databases. We remark that the initial high-fidelity database size was purposely chosen to be very small, since the active learning component of the optimization pipeline would grow it to refine the surrogates.

Figure 3.11 reports the errors distribution for n_y and n_b , for different truncation ranks and the two databases considered. Although the chosen truncation rank leads to worse

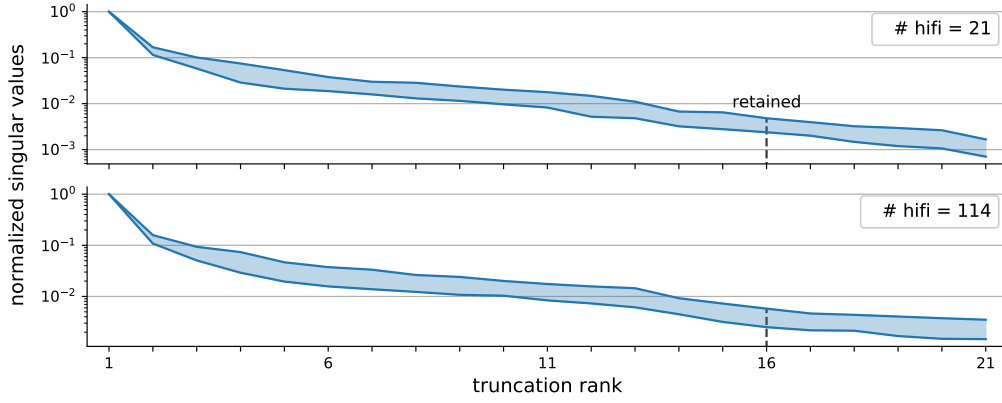


Figure 3.10: Evolution of the range of normalized singular values decay of the stress tensor components. The values come from the high-fidelity database before and after optimization, for the full ship FS6 with 10 parameterized sections.

error distributions on the initial database, after optimization the choice of a higher rank does not lead to a large difference.

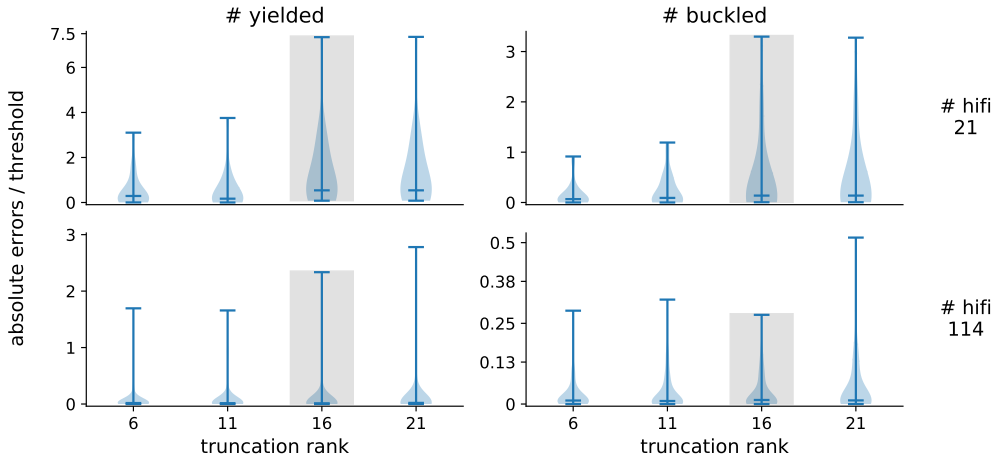


Figure 3.11: Comparison of the 5-CV errors on n_y and n_b , for different database sizes and ranks. The predictions are relative to FS6 with 10 parameterized sections. The plots highlight the median error and the gray box identifies the chosen truncation rank.

We then compare different predictors for reduced coefficients, for the initial parameterization and the full database, in Figure 3.12. As for MS00, the multi-GPR approach achieves the best results in terms of accuracy, followed by vector-valued GPR, RBF, and NARGPAS. The single-fidelity GPR-based predictors consistently score under the 0.05 error threshold for the stress tensor components, von Mises criterion, and the first buckling usage factors. The buckling usage factor uf_{yz} , which considers the shear in the yz plane, is zero for all degrees of freedom in FS6. This is due to the regulatory minimum thickness of the transverse bulkheads, and other transverse structural elements, being set at the higher end of the commercially available thicknesses. In this case, the designers decided to not compute the buckling for such elements and use instead a lower threshold for b_{crit} during

optimization.

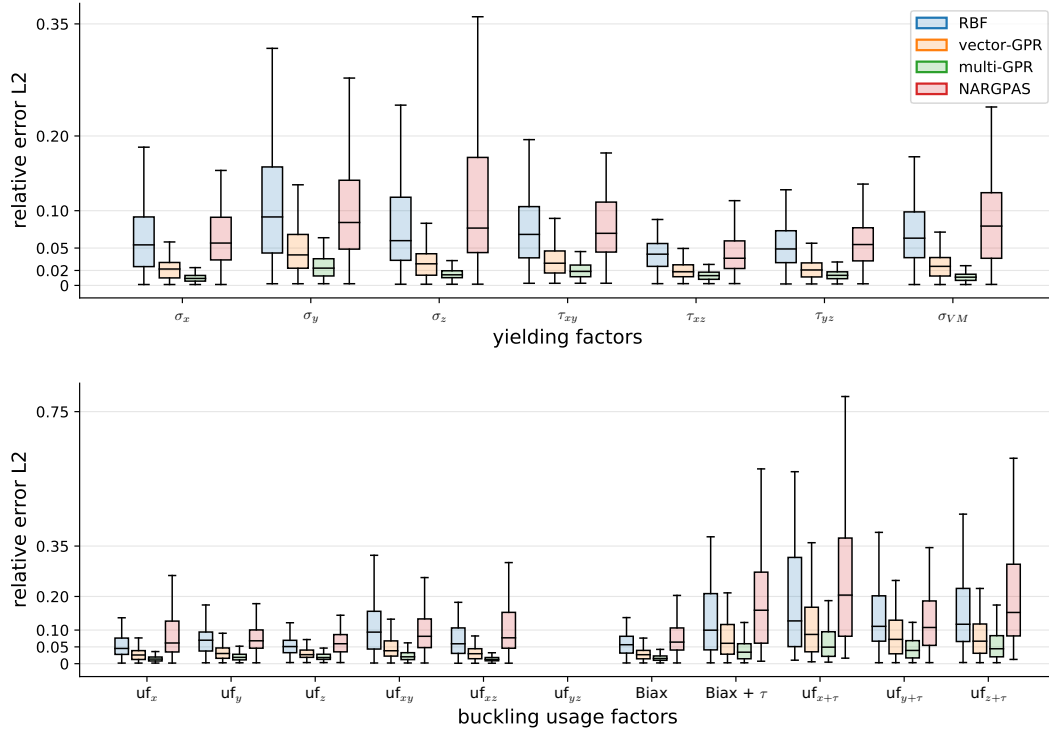


Figure 3.12: Comparison of the 5-CV errors on the stress tensor components and their post-processed results, with medians highlighted, for different predictors of reduced coefficients. The values are relative to the hogging condition, simulated on the full ship FS6 with 10 parameterized sections.

Figure 3.13 is obtained from the refined parameterization with 40 parameterized sections, on a database of size 617. Differently from the previous case, **NARGPAS** shows heavily degraded performances possibly due to the increase in database size, with the worst predictions reaching above the 0.5 threshold in relative error. For this reason, **NARGPAS** is not shown in the plot. Although less pronounced as with **NARGPAS**, all other models register a decrease in accuracy. The disparity in computational effort in constructing the multi-**GPR** compared with the vector-valued **GPR**, on this larger database, is staggering: the multi-**GPR** of a single field takes about 10 hours to be fitted, instead of the 10 minutes required by the simpler predictor.

Finally, Figure 3.14 compares the performance of the single **GPR** and the full surrogate approach for n_y and n_b . As with MS00, the more involved approach achieves much lower prediction errors.

Regarding the prediction of the vertical deflection, the simple **GPR** approach model achieves an R^2 score of 0.94 on the initial parameterization, and 0.98 on the refined one. In terms of **MAPE**, the two models score 3.48% and 1.49%, respectively.

A prediction of the complete set of **QoIs** on FS5, amortized over 1000 queries executed in parallel, takes on average 0.034s for the model with 10 parameterized sections and 0.036s for the one with 40 parameters.

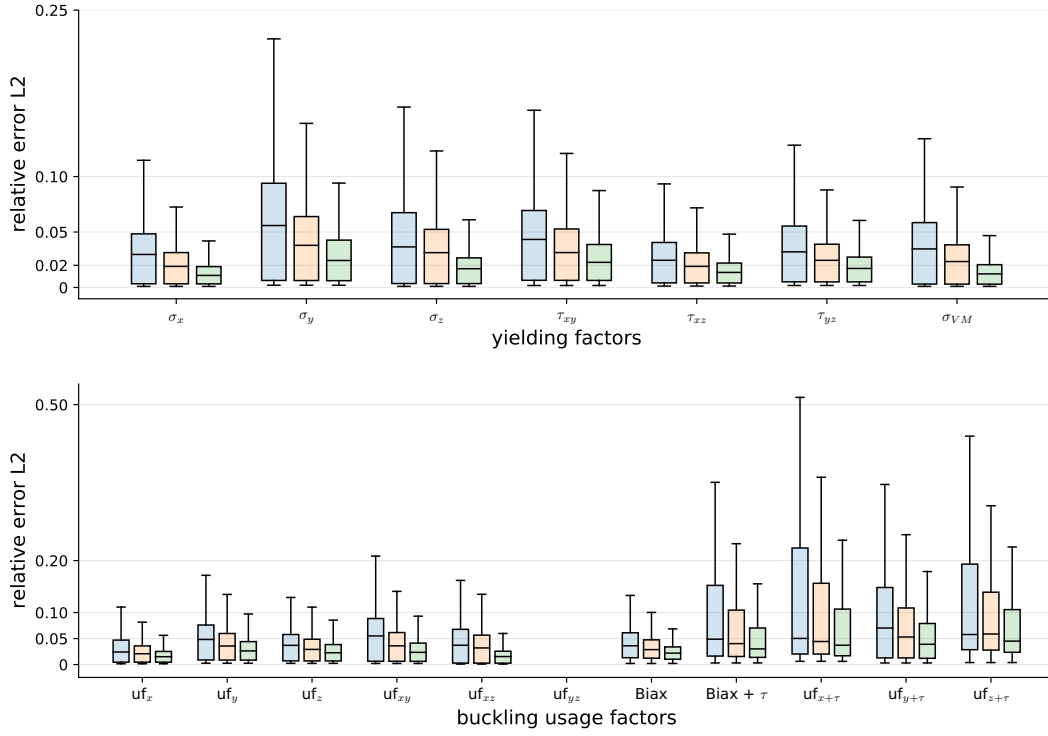


Figure 3.13: Comparison of the 5-CV errors on the stress tensor components and their post-processed results, with medians highlighted, for different predictors of reduced coefficients. The values are relative to the hogging condition, simulated on the full ship FS6 with 40 parameterized sections.

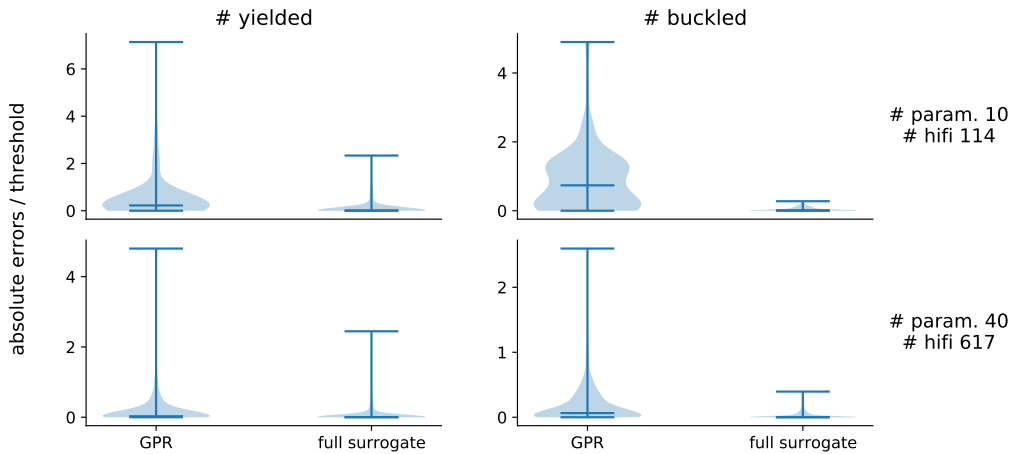


Figure 3.14: Comparison of the 5-CV errors on n_y and n_b , with medians highlighted, for different prediction approaches. The values are relative to FS6, with 10 and 40 parameterized sections.

3.7 Conclusions

This chapter presented the full surrogate at the core of the optimization pipeline presented in Chapter 1.

We use **POD** for the dimensionality reduction of the stress tensor components, with truncation rank chosen based on the singular values decay. We propose to select the last rank for which, considering all the fields of interest, the highest normalized singular value is above 0.01. For the prediction of the reduced coefficients of each load condition and stress tensor component, we select a vector-valued **GPR**. In this setup, a single squared exponential kernel with **ARD** is used for all the elements of the coefficients vector. The alternative approach, using separate **GPRs**, achieves lower prediction errors, but the extremely high construction time makes it infeasible for the optimization pipeline.

The full surrogate computes n_y and n_b via aggregation of the full-scale yielding criteria and the buckling usage factors, with the vertical deflection obtained via a simple scalar **GPR**. This approach achieves a high computational efficiency in the many-query regime, with the amortized time for a prediction of the full **QoIs** requiring a few milliseconds for MS00, and a few hundredths of a second for FS6.

Chapter 4 will present the developed multi- and single-objective optimizations leveraging the full surrogates.

SURROGATE-ASSISTED OPTIMIZATION

This chapter discusses the optimization step of the pipeline from Chapter 1, leveraging the full surrogates presented in Chapter 3 instead of the high-fidelity simulation from Chapter 2 to evaluate the [quantities of interest \(QoIs\)](#). With an infill criterion, only the best candidates are then added to the snapshots database to update the surrogates and continue the search.

First, multi-objective optimization is used to increase the accuracy of the surrogates close to the [Pareto front \(PF\)](#), which provides the designers with best trade-offs between [QoIs](#). The ability to obtain a global representation of the model’s capabilities is crucial for the design process, since it guides the choices of constraint thresholds for the following mass optimization. Due to the computational efficiency of our full surrogates, we use the [non-dominated sorting genetic algorithm III \(NSGA-III\)](#) [54] for unconstrained multi-objective optimization. As infill criterion, we propose a simple approach that leverages the covariance kernels of the underlying [Gaussian process regressions \(GPRs\)](#) to select the samples most relevant to the accuracy of the [PF](#).

For single-objective optimization, we compare multiple approaches from the literature on black-box and derivative-free optimization [113, 43]. In particular, we use [Bayesian optimization \(BO\)](#) [85, 24] and metaheuristics such as [particle swarm optimization \(PSO\)](#) [105] and [differential evolution \(DE\)](#) [116], hybridized with a simpler direction-based approach. A key element of the implementation is the ability to handle the feasibility constraints on the [QoIs](#) and the discrete nature of $\mathcal{D}$, for which we leverage a rounding strategy based on an [integer linear program \(ILP\)](#) [166].

In Section 4.1 we briefly review the literature on surrogate-assisted optimization and discuss its integration in the optimization pipeline. In Section 4.2 we present [NSGA-III](#) and our infill criterion. The different approaches to single-objective optimization and constraint handling are discussed in Section 4.3. In Section 4.4 we apply our multi-objective infill criterion on the coarse parameterizations of a simplified midship section and a full ship, and compare multiple approaches to single-objective optimization on the refined parameterization of a full ship. The final conclusions are drawn in Section 4.5.

4.1 Introduction

Armed with the efficient surrogates discussed in Chapter 3, the designer can now evaluate new configurations in the design space with minimal time-to-solution. In principle, the search for the thickness configuration that leads to the minimal mass while respecting the criteria of structural resilience described in Chapter 2 can be conducted in semi-real time, with the designers inspecting the changes in [QoIs](#) and field outputs to decide which decision variable to change. However, rather than relying only on their expertise and intuition, the search for the optimal ship design can be augmented via the automation of an optimization procedure, able to fully leverage the computational resources available nowadays.

The literature on numerical optimization methods is vast and constantly evolving. Usually, the first choice is to leverage the gradient of the objective function, to iteratively refine a configuration along the direction of maximal decrease. Many first order methods integrate additional heuristics [110], and have achieved great success in conjunction with automated differentiation in the context of deep learning [141]. Quasi-Newton methods, which extend gradient descent with an approximation of the Hessian such as [low-memory Broyden–Fletcher–Goldfarb–Shanno \(L-BFGS\)](#) [217], are a popular choice in scientific computing tasks [197, 84]. Differentiable constraints can be integrated via the Lagrangian, leading to methods such as the augmented Lagrangian [27] and interior point methods [199]. A classic approach is the construct of a surrogate linear or quadratic model inside a [trust region \(TR\)](#) [83, 31, 129, 43], updating the [TR](#) and the surrogate as the incumbent moves and accuracy of the model changes. An issue with these approaches is that they are based on a local refinement of a starting configuration, and are susceptible to becoming stuck in local minima. Nonetheless, this is often mitigated by the use of repeated refinement of random initial configurations, which is a strong baseline [216].

In [4], the [reduced basis \(RB\)](#) method was employed alongside hyper-reduction and [radial basis function \(RBF\)](#) interpolation to optimize complex [partial differential equation \(PDE\)](#)-constrained problems. The adjoint method [82] was employed in [40] to the optimization of wings under flutter constraints and an [arbitrary Lagrangian Eulerian \(ALE\)](#) system. In our case, NASTRAN could provide the Jacobians of the stress tensors with respect to the thickness of the parameterized sections by solving the adjoint problem, but the post-processing to obtain n_y and n_b is non-smooth and non-convex. Combined with the discrete nature of the parametric domain, we prefer black-box modeling of the constrained [QoIs](#).

Another approach that leverages both differentiation and data-driven surrogation is [BO](#) [85, 174, 24, 95]. [BO](#) requires a surrogate that is able to produce [uncertainty quantification \(UQ\)](#) of the objective function, most often a [GPR](#), and optimizes an acquisition function which takes into account both the predicted objective and the associated uncertainty. This technique is able to escape local optima, although its efficacy on high-dimensional domains and discrete decision variables remains an area of active research [51, 63, 140, 96]. [BO](#) has

seen wide adoption for the optimization of expensive black-box functions, especially in the context of hyper-parameters tuning for deep learning [175, 2, 132].

Under the assumption of an objective cheap to evaluate, multiple algorithms directly construct a set of directions to explore, such as the classic Powell’s method, Hooke-Jeeves and generalized pattern search [113]. Other algorithms, such as Nelder-Mead or simplex method [43] or the related Box’s method [9] that handles convex constraints, employ a structure of multiple evaluation points which are updated according to sophisticated rules. The [Dividing RECTangles \(DIRECT\)](#) algorithm [99, 35] uses a subdivision schema to explore the domain under assumption of Lipschitz objective. These techniques, however, are tailored to continuous domains and do not translate easily to discrete variables.

Other than using deterministic search directions, metaheuristics [188, 62] include a stochastic element in the generation of configurations to evaluate. Although this class of procedures does not guarantee optimality, its efficacy in providing high quality solutions for hard optimization problems, often defined on combinatorial domains, has determined widespread adoption as heuristic procedures. One of the foremost examples is simulated annealing [111, 195], which generates coordinates changes at random and accepts worsening moves with decreasing probability.

However, the most common family of metaheuristics is that of the population-based procedures, in which we can separate evolutionary computation and nature-inspired dynamics. The common denominator is the maintenance of a population of configurations rather than a single element, and that the interaction between different individuals generates the coordinates update that explore the domain.

In the evolutionary computation approach, which include [genetic algorithm \(GA\)](#) [177], [evolutionary strategy \(ES\)](#) [10] and [DE](#) [147, 48, 212], the fundamental idea is to replicate the effects of natural selection through a series of operators such as selection and crossover. In a broad sense, these metaheuristics iteratively select elite individuals and combine their parameter values to generate new configurations, effectively interpreting the parameters as genetic material.

Nature-inspired procedures, instead, treat the individuals in the population as geometric coordinates and define an update step that mimics the population dynamics seen in birds flocking, or other groups of animals. The most famous exponent of this family is [PSO](#), where individuals are attracted by both the position of the best known solution, and their own best configuration. [Ant colony optimization \(ACO\)](#) [61, 60] introduces an indirection layer: the individuals contribute to a globally known score of coordinate changes and select their move accordingly. This imitates the pheromone trails left by ants, which progressively promote paths leading to the optimum.

A crucial element of the optimization problem is the enforcement of constraints, which in our case take two forms: inequalities on structural [QoIs](#) and on the thickness configuration vector itself, and discreteness of the domain.

The discrete domain is a deceptively simple formulation: although in principle it decreases the complexity of the search, having a finite number of configurations to

choose from, it takes away the information of local behavior that comes from the gradient and Hessian. Heuristics that can employ effective discrete updates of the configuration coordinates have an advantage, while procedures that rely on a continuous domain need to integrate a rounding step. In the special case of linear objective and linear constraints, but discrete domain, [ILP](#) have a long history of developments and specialized solution techniques via branch and bound, and simplex or interior points methods, which result in practical and efficient methods for small and medium problem instances [1, 178]. In particular, it is possible to recast the rounding of an invalid configuration to a valid discrete point, feasible for the constraints on the [center of gravity \(COG\)](#), as an [ILP](#) with a simple structure and small size. This efficient rounding routine can be integrated in distance-based algorithms such as [PSO](#), and provides a post-processing step for [BO](#).

The inequalities on the structural [QoIs](#) exhibit non-convex and non-smooth behavior, and as shown in Chapter 3 are difficult to surrogate directly. A simple workaround is the integration of the constraints violation into the objective function, an approach known as penalty method [113, 43]. Even if this method opens to the use of unconstrained algorithms, the drawback lies in the careful choice of penalty coefficients, since they must properly model the trade-off between the entity of the violation and the objective.

The study of interaction between different [QoIs](#) goes beyond the mere selection of penalty terms: identifying the configurations with the best trade-off between competing quantities, such as total mass and structural [QoIs](#), can inform the decision-making and the planning of the entire design process. Multi-objective optimization aims to determine, or approximate, the [PF](#) and the corresponding [Pareto set \(PS\)](#) which provide such trade-offs. Although this problem can be recast in terms of repeated single-objective optimization [113] either via constrained searches [35] or via scalarization [214], population-based metaheuristics are the most popular choice [71]. [Non-dominated sorting genetic algorithm \(NSGA\)](#) is a family of [GAs](#) [53, 54] that constitutes the first choice in many applications [203]. Multi-objective [BO](#) has recently been explored, but remains limited by the complex computation of appropriate acquisition functions on higher dimensions [112, 63, 49, 131, 86, 50].

4.2 Multi-objective optimization

In this section, we first review the notation for multi-objective optimization and motivate our choice of [NSGA-III](#) as optimization algorithm. Then, we present the details of the algorithm and our proposed infill criterion, tailored to the full surrogate.

We aim to find the [PF](#) in the space of [QoIs](#) collected by $\mathbf{f}$ from Equation (2.5), which will instruct the choice of y_{crit} , b_{crit} , d_{crit} . The structural [QoIs](#) that will be constrained are indeed in competition with f_{mass} , which will be the target of single-objective optimization. By inspecting the unconstrained [PF](#), the designers can choose the critical thresholds that allow for a light ship, but limit the number of structural failures to be corrected in future design phases.

In multi-objective optimization, a configuration $\mathbf{x}^*$ is said to dominate $\mathbf{x}$, with notation $\mathbf{x}^* > \mathbf{x}$, when the objective functions evaluated on $\mathbf{x}^*$ take no worse values compared to $\mathbf{x}$, and at least one objective takes a strictly better value [176]. Formally, assuming that all members of $\mathbf{f}$ must be minimized, we write

$$\mathbf{x}^* > \mathbf{x} \Leftrightarrow \mathbf{f}_i(\mathbf{x}^*) \leq \mathbf{f}_i(\mathbf{x}) \forall i, \exists j : \mathbf{f}_j(\mathbf{x}^*) < \mathbf{f}_j(\mathbf{x}), \quad (4.1)$$

with $i, j \in \{1, \dots, 5\}$. A dominated configuration is of no interest in the optimization since there exists another configuration that performs no worse on all **QoIs**, and is strictly better for at least one **QoI**.

PS is the set of all non dominated configurations

$$\text{PS} = \{\mathbf{x}^* \in \mathcal{D} : \nexists \mathbf{x} \in \mathcal{D} \text{ such that } \mathbf{x} > \mathbf{x}^*\}, \quad (4.2)$$

and the **PF** is the image of **PS** through $\mathbf{f}$:

$$\text{PF} = \{\mathbf{f}(\mathbf{x}^*) : \mathbf{x}^* \in \text{PS}\}. \quad (4.3)$$

For usual $\mathcal{D}$, the exact **PF** and **PS** can not be determined due to the high number of possible configurations and **QoIs**. We settle on finding an approximation of **PF**, and refining the surrogates' accuracy on the configurations close to the **PS** via an appropriate infill criterion. The optimization and infill will be repeated in the usual surrogate-based approach [149], until either a budget is exhausted or the infill criterion shows limited improvement.

For the choice of optimization procedure, we adapt the **NSGA** family of **GA**. The population-based approach enables parallel evaluation of a large batch of configurations, fully leveraging the hardware resources. Alternative approaches present significant drawbacks: scalarization or constrained optimization add a computational overhead compared to direct evaluation of the population, and **UQ**-based approaches require Monte Carlo evaluation of complex integrals.

4.2.1 Non-dominated sorting genetic algorithm-III

Classic **GAs** for single-objective optimization are designed around the concept of the objective function representing the fitness for survival and mating success of an individual, and the domain coordinates constitute its genome. This point of view drives the design of the various operators that implement the update of the population. The selection operator identifies the parents-to-be in the population; the crossover operator generates new genomes from the combination of two or more parents; the mutation operator introduces alien genome features in the newly conceived individuals; and finally the environmental selection operator removes the less fit individuals. The generic structure is outlined in Algorithm 1.

In the multi-objective case, the fitness must represent the degree of domination that an individual exercises in the population. The **NSGA** family of algorithms constructs a sequence of population subsets, where each subset contains the **PS** of the union of itself

Algorithm 1 Template of a genetic algorithm.

Input: initial population $\mathbf{X}$, fitness function f , crossover and mutation functions, maximum population p , number of generations q **Output:** final population $i \leftarrow 0$ **while** $i < q$ **do** $\mathbf{X}^{\text{parents}} \leftarrow \text{select from } \mathbf{X} \text{ according to } f$ $\mathbf{X}^{\text{children}} \leftarrow \text{crossover of } \mathbf{X}^{\text{parents}}$ $\mathbf{X}^{\text{children}} \leftarrow \text{mutation of } \mathbf{X}^{\text{children}}$ $\mathbf{X} \leftarrow \mathbf{X}^{\text{parents}} \cup \mathbf{X}^{\text{children}}$ $\mathbf{X} \leftarrow p \text{ survivors, the best performing individuals from } \mathbf{X} \text{ according to } f$ $i \leftarrow i + 1$ **end while****return** $\mathbf{X}$

Algorithm 2 Non-dominated sorting.

Input: population $\mathbf{X} = \{\mathbf{x}^{(i)}\}_{i=1}^m$ **Output:** list L of non-dominated layers as set of indices $L \leftarrow \text{empty list}$ $R \leftarrow \{i = 1, \dots, m\}$ **while** $|R| > 0$ **do** $\mathbf{X}^R \leftarrow \{\mathbf{x}^{(i)} \mid i \in R\}$ $F \leftarrow \{i \mid \mathbf{x}^{(i)} \text{ is not dominated in } \mathbf{X}^R\}$ $L \leftarrow L \cup \{F\}$ $R \leftarrow R \setminus F$ **end while****return** L

and the following subsets. The construction of the non-dominated sort is implemented in Algorithm 2.

For the parents selection operator, we simply return the entire population. As crossover operator, we use two-point crossover and a variant of interpolation crossover where we take the element-wise minimum or maximum of the parents' features. We select a pair of parents and apply one of the two crossover operators, with both selection performed at random. This operation is iterated until we generate children in number equal to the current population. For the mutation operator, we choose at random whether to decrease or increase the features of an individual, or to generate a completely new one with a much lower probability. We remark that crossovers and the mutation operators are tailored to the discrete domain.

The selection of the fittest individuals is performed by taking the union of non-dominated subsets, starting from the first, until the limit of population size is reached. If the number of selected individuals exceeds the limit, the subset with the most dominated individuals is down-sampled with the niching strategy from [NSGA-III](#) [54], to prevent overcrowding in the [QoIs](#) space. Algorithm 3 summarizes the procedure.

Algorithm 3 NSGA-III selection strategy.

Input: population $\mathbf{X} = \{\mathbf{x}^{(i)}\}_{i=1}^m$, list $L = \{F_1, F_2, \dots\}$ of non-dominated layers as set of indices, $\mathbf{f}$ vector of objective functions, $\mathbf{H} \in [0, 1]^{q \times |\mathbf{f}|}$ reference points on the plane $\sum_i y_i = 1$, p number of individuals to select

Output: S set of selected points as indices

```

 $S \leftarrow \emptyset$ 
 $i \leftarrow 1$ 
while  $|S| + |F_i| \leq p$  do
     $S \leftarrow S \cup F_i$ 
     $i \leftarrow i + 1$ 
end while
if  $|S| < p$  then
    evaluate  $\mathbf{Y} = \{\mathbf{y}^{(j)} = \mathbf{f}(\mathbf{x}^{(j)}) \mid j \in S \cup F_i\}$ 
    rescale  $\mathbf{Y}$  to  $[0, 1]^{|\mathbf{f}|}$ 
    construct  $q$  straight lines passing through  $\mathbf{0}$  and a point of  $\mathbf{H}$ 
    associate each individual in  $S$  to the closest line
    build  $\mathbf{c} \in \mathbb{N}^q$  that counts the individuals associated to each line
    build  $\pi : F_i \rightarrow \{1, \dots, q\}$  association to the closest line
    compute  $P_j = \{k \in F_i : \pi(k) = j\}$  for  $j = 1, \dots, q$ 
    while  $|S| < p$  do
         $j = \operatorname{argmin}_{P_j \neq \emptyset} |\mathbf{c}_j|$ 
        select random  $k \in P_j$ 
         $S \leftarrow S \cup \{k\}$ 
         $\mathbf{c}_j \leftarrow \mathbf{c}_j + 1$ 
         $P_j \leftarrow P_j \setminus \{k\}$ 
    end while
end if
return  $S$ 

```

4.2.2 Infill

Due to the use of surrogates, the final **PF** is only an approximation of the true set of non-dominated individuals. An infill criterion is needed to select a reduced number of individuals to be validated by the high-fidelity solver, so that further multi-objective optimizations yield a more accurate **PF**.

The approximated **PS** comprises the high-fidelity samples $\mathbf{X}^H$, that contributed to the construction of the surrogates, and the low-fidelity samples $\mathbf{X}^L$ that can be passed to the high-fidelity solver to increase the accuracy of the surrogates. For an individual to be chosen, we require that its addition maximally improves the predictions at the location of the remaining low-fidelity members of the **PF**. This approach is similar to [160], which adapts the idea of maximizing the conditioned uncertainty reduction from [122] and the integration of an uncertainty measure over a domain of interest from [42], to the case of **GPR**. This principle has been developed in terms of entropy [91, 16, 183] for multi-objective optimization for **GPRs** that directly model the objectives. We choose to measure the

covariance between the samples in the reduced coefficient space, since the corresponding **GPRs** have been shown to provide accurate predictions from which the **QoIs** are derived. Instead of explicitly estimating the reduction in uncertainty as in previous works [160], the increment in covariance only requires the evaluation of the kernel and is thus cheap compared to Equation (3.11). We also need an aggregation strategy, since the procedure considers **GPRs** organized in a multi-dimensional matrix.

The infill procedure starts by constructing the covariance matrices between the elements in $\mathbf{X}^L$, and between those in $\mathbf{X}^L$ and $\mathbf{X}^H$. We aggregate over load conditions and stress tensor components by taking the maximum, then sum over the reduced coefficients to obtain $\mathbf{C}^{LL} \in \mathbb{R}^{n \times n}$, symmetric, and $\mathbf{C}^{LH} \in \mathbb{R}^{n \times m}$.

Let $\{\mathbf{K}_{LL}^{(l,s,c)}\}$ be the set of covariance matrices of the **GPR** corresponding to the reduced coefficient c of the stress tensor component s under the load condition l , evaluated between the samples in $\mathbf{X}^L$. Then, the element (i, j) of the aggregation matrix is computed as

$$\mathbf{C}_{ij}^{LL} = \sum_c \max_{l,s} \left[\left(\mathbf{K}_{LL}^{(l,s,c)} \right)_{ij} \right], \quad \forall i, j = 1, \dots, n, \quad (4.4)$$

giving a scalar measure of the mutual information between the samples in $\mathbf{X}^L$. The same expression is used for the construction of $\mathbf{C}^{LH}$. A vector Δ is built by computing, for the i -th sample in $\mathbf{X}^L$, the total positive relative increase over the maximum high-fidelity contribution to the rest of the low-fidelity **PF** as

$$\Delta_i = \frac{\sum_{j \neq i} (\mathbf{C}_{ij}^{LL} - \max_h \mathbf{C}_{jh}^{LH})_+}{\sum_{j \neq i} \max_h \mathbf{C}_{jh}^{LH}}, \quad \forall i = 1, \dots, n, \quad (4.5)$$

where $(\cdot)_+ = \max(\cdot, 0)$. The high-fidelity set is then enriched by adding the i^* -th sample determined by

$$i^* = \underset{i}{\operatorname{argmax}} \Delta. \quad (4.6)$$

The choice of the aggregation functions in Equation (4.4) can be viewed as constructing the best case scenario for the mean covariance increase across the stress tensor components. If more than one configuration can be selected for the high-fidelity validation, the matrices $\mathbf{C}^{LL}$ and $\mathbf{C}^{LH}$ can be updated by simulating the addition of the sample i^* to the high-fidelity set, as summarized in Figure 4.1. In particular, the i^* -th row (and column) is removed from $\mathbf{C}^{LL}$, its elements are used to build the column $m + 1$ in $\mathbf{C}^{LH}$, and finally the i^* -th row of $\mathbf{C}^{LH}$ is removed. The resulting matrices have thus size $n - 1 \times n - 1$ and $n - 1 \times m + 1$ respectively, and the selection can take place after computing the updated Δ vector.

As a stopping criterion for the optimization-infill loop, we can monitor the Δ of the first selection and terminate when it falls below a small threshold. Another stopping criterion can be defined on the size of the high-fidelity database, or on the number of infill operations.

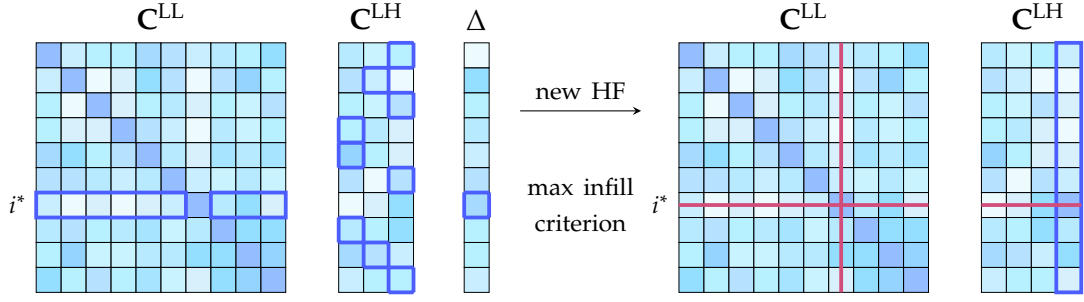


Figure 4.1: On the left, position i^* in the low-fidelity PF is selected as maximizer of the infill criterion Δ evaluated on matrices $\mathbf{C}^{\text{LL}}$ and $\mathbf{C}^{\text{LH}}$ using Equation (4.5). The addition of the i^* -th sample to the high-fidelity set is simulated with the removal of row (and column) i^* from $\mathbf{C}^{\text{LL}}$, its addition to $\mathbf{C}^{\text{LH}}$, followed by the removal of row i^* . The next sample is selected using the updated matrices.

4.3 Single-objective optimization

At this point, the designers have acquired a global view of the model's QoIs via the PF, and focus on the minimization of the total mass. In this setting, the acceptable thresholds for the structural QoIs n_y , n_b , and $f_{\text{deflection}}$ have been determined by taking into account the manual work that will be required to reduce them in the following design phases. Likewise, the designers have set the critical values on the components of the COG, usually limited to VCG_{crit} .

The fundamental difference between the two sets of constraints is that the structural QoIs are computed via the evaluation of the full surrogate, through a non-smooth, non-convex procedure. We choose a soft-constraint approach typical of black-box constraints, integrating the violation of the critical thresholds as a quadratic penalty term in the objective function. The objective for mass optimization thus becomes

$$f(\mathbf{x}) = f_{\text{mass}}(\mathbf{x}) + f_{\text{pen}}(\mathbf{x}), \quad (4.7)$$

with

$$f_{\text{pen}}(\mathbf{x}) = c_y(n_y(\mathbf{x}) - y_{\text{crit}})_+^2 + c_b(n_b(\mathbf{x}) - b_{\text{crit}})_+^2, \quad (4.8)$$

where we use $(\cdot)_+ = \max(\cdot, 0)$. The terms c_b and c_y are positive conversion factors from the unit of measure of the penalized QoI, squared, to the unit of measure of the QoI being minimized.

The basic optimization problem reads as follows

$$\min f(\mathbf{x}) = m_{\text{fixed}} + \mathbf{d} \cdot \mathbf{x} + m_{\text{bar}} n_b(\mathbf{x}) + f_{\text{pen}}(\mathbf{x}), \quad (4.9)$$

$$\begin{aligned} \text{s.t.} \quad & \sum_{i \in I} (\text{VCG}_i - \text{VCG}_{\text{crit}}) \mathbf{d}_i x_i \leq (\text{VCG}_{\text{crit}} - \text{VCG}_{\text{fixed}}) m_{\text{fixed}}, \\ & \mathbf{x} \in \mathcal{D}, \end{aligned} \quad (4.10)$$

where only a constraint on vertical center of gravity (VCG) is shown for brevity.

To incorporate the constraints on the **COG** components and the discrete nature of the design space into optimization algorithms that do not natively support such constraints, we implement a repair procedure based on an **ILP**. The procedure is formulated as the problem of finding the feasible configuration $\mathbf{x} \in \mathcal{D}$ that is closest to a given infeasible candidate $\bar{\mathbf{x}}$, while satisfying the prescribed constraints on the **COG** components. Thus, we write an assignment problem minimizing $\|\bar{\mathbf{x}} - \mathbf{x}\|^2$. We introduce a binary variable for each valid pair of parameterized section and thickness, that takes value 1 when the assignment is active. A single thickness can be chosen per parameterized section. The **ILP** reads as follows:

$$\begin{aligned}
 \min \quad & \sum_{i \in I} \sum_{t \in \mathcal{D}_i} x_{it} |t - \bar{x}_i|^2, \\
 \text{s.t.} \quad & \sum_{t \in \mathcal{D}_i} x_{it} = 1, \\
 & \sum_{i \in I} \sum_{t \in \mathcal{D}_i} x_{it} (\text{VCG}_i - \text{VCG}_{\text{crit}}) \mathbf{d}_i t \leq (\text{VCG}_{\text{crit}} - \text{VCG}_{\text{fixed}}) m_{\text{fixed}}, \\
 & x_{it} \in \{0, 1\},
 \end{aligned} \tag{4.11}$$

$\forall i \in I,$

$\forall i \in I, \forall t \in \mathcal{D}_i,$

where x_{it} is 1 if thickness t is assigned to parameterized section i , and $I = \{1, \dots, n_{\text{dim}}\}$ is the set of indices of all parameterized sections. The constraints on **transversal center of gravity (TCG)** and **longitudinal center of gravity (LCG)** can be added with analogous expression to the one for the **VCG**.

This problem requires $n_{\text{dim}} \times \mathcal{T}$ variables, up to 3 constraints on **COG** components, and n_{dim} special ordered set constraints of type 1 [15]. In practice, the solution of such instances poses no problem to modern solvers [72], especially since Problem 4.11 can be created once and efficiently recycled for multiple $\bar{\mathbf{x}}$ by only updating the coefficients of the objective function. The projection approach is reminiscent of the feasibility pump [70] developed for the quick search of feasible solution to mixed integer linear programs.

To further constrain the search, it is possible to derive another linear inequality from Equation (4.7). Given the incumbent $\mathbf{x}^*$, we have that the candidates of interest $\mathbf{x}$ need to satisfy

$$f(\mathbf{x}^*) \geq m_{\text{fixed}} + \mathbf{d} \cdot \mathbf{x} + m_{\text{bar}} n_{\text{b}}(\mathbf{x}) + f_{\text{pen}}(\mathbf{x}) \geq m_{\text{fixed}} + \mathbf{d} \cdot \mathbf{x}, \tag{4.12}$$

since $n_{\text{b}}(\mathbf{x}) \geq 0$ and $f_{\text{pen}}(\mathbf{x}) \geq 0$. The left hand side of Equation (4.12) is updated as the search progresses, providing an effective restriction of the search space. We will use the term "active linear constraint" to denote the set of linear constraints related to feasibility (that is, imposed on the **COG**) and Equation (4.12) which only restricts the search space.

Another routine that will be used throughout this section is the selection of an initial population of configurations, where the bias towards optimal values of the objective function is balanced with a sufficient variety of coordinates values. Single-objective optimization always starts from a population of previously evaluated configurations, consisting of the high-fidelity database and the latest **PF**. It is then natural to identify not only the best known solution, commonly called incumbent, but to also select a subset

of configurations to initialize the actual optimization algorithms such as [BO](#) and [PSO](#). In Algorithm 4, we assume an already evaluated initial population and constrain the Hamming distance among the chosen samples to promote diversity. The Hamming distance counts the number of different elements in two sequences and can be defined as

$$d_{\text{Hamming}}(\mathbf{x}, \bar{\mathbf{x}}) = \sum_i^{n_{\text{dim}}} 1 - \delta_{x_i \bar{x}_i}, \quad (4.13)$$

where $\delta_{xy} = 1$ if and only if $x \neq y$. For our application, we impose that the Hamming distance is $\geq n_{\text{dim}}/2$.

Algorithm 4 Down-sampling procedure for single-objective populations.

Input: feasible samples $\mathbf{X}$, objective function f , number of output samples n_{pop} , discrete domain $\mathcal{D}$

Output: initial population $\mathbf{S}$

```

 $\mathbf{S} \leftarrow \emptyset$ 
while  $|\mathbf{S}| < n_{\text{pop}} \wedge |\mathbf{X}| > 0$  do
     $\mathbf{x} \leftarrow \operatorname{argmin}_{\mathbf{x} \in \mathbf{X}} f(\mathbf{x})$  such that  $d_{\text{Hamming}}(\mathbf{x}, \mathbf{s}) \geq n_{\text{dim}}/2 \forall \mathbf{s} \in \mathbf{S}$ 
     $\mathbf{S} \leftarrow \mathbf{S} \cup \{\mathbf{x}\}$ 
     $\mathbf{X} \leftarrow \mathbf{X} \setminus \{\mathbf{x} \in \mathbf{X} : d_{\text{Hamming}}(\mathbf{x}, \mathbf{s}) < n_{\text{dim}}/2 \forall \mathbf{s} \in \mathbf{S}\}$ 
end while
if  $|\mathbf{S}| < n_{\text{pop}}$  then
     $\mathbf{T} \leftarrow \text{maximin sampling of } n_{\text{pop}} - |\mathbf{S}| \text{ samples from } \mathcal{D}$ 
     $\mathbf{S} \leftarrow \mathbf{S} \cup \mathbf{T}$ 
end if
return  $\mathbf{S}$ 

```

Having discussed our approaches to constraints handling and population initialization, the rest of this section will present the actual procedures used for optimization.

4.3.1 Principal directions search

For a simple greedy heuristic, we combine the fixed search directions approach of Hooke-Jeeves and the line-search of the classic Powell's method to obtain the [principal directions search \(PDS\)](#). First, starting from an initial point $\bar{\mathbf{x}}$, we enumerate all other feasible $\mathbf{x}$ along the principal axes. The population obtained can be represented as

$$\mathbf{X}^{\text{PDS}} = \{\mathbf{x} \in \mathcal{D} : d_{\text{Hamming}}(\mathbf{x}, \bar{\mathbf{x}}) = 1 \wedge \mathbf{g}(\mathbf{x}) \leq \mathbf{0}\}, \quad (4.14)$$

where $\mathbf{g}(\mathbf{x}) \leq \mathbf{0}$ represents the active linear inequality constraints. Then, the objective function is evaluated on $\mathbf{X}^{\text{PDS}}$ and if a new incumbent is found, it becomes the next $\bar{\mathbf{x}}$ and the search is repeated. If the incumbent is not updated, the next $\bar{\mathbf{x}}$ can be drawn at random. The optimization process is terminated after reaching either a prescribed number of iterations or a specified time limit.

The entire process is detailed in Algorithm 5.

[PDS](#) provides a strongly exploitative heuristic to quickly refine a configuration. At each step, $|\mathbf{X}^{\text{PDS}}| \leq \sum_{i=1}^{n_{\text{dim}}} |\mathcal{D}_i| - 1$ different configurations are evaluated, but the $\mathbf{g}$ constraints

Algorithm 5 principal directions search.

Input: initial sample $\bar{\mathbf{x}}$, objective function f , **COG** linear inequality constraints $\mathbf{g}(\mathbf{x}) \leq \mathbf{0}$, discrete domain $\mathcal{D}$

Output: incumbent $\mathbf{x}^*, y^*$

```

 $\mathbf{x}^*, y^* \leftarrow \bar{\mathbf{x}}, f(\bar{\mathbf{x}})$ 
while not finished do
  construct  $\mathbf{X}^{\text{PDS}}$  as in Equation (4.14)
   $\mathbf{y} \leftarrow f(\mathbf{X}^{\text{PDS}})$ 
  if  $\min\{\mathbf{y}\} < y^*$  then
    update  $\mathbf{x}^*, y^*$ 
     $\bar{\mathbf{x}} \leftarrow \mathbf{x}^*$ 
  else
    generate random  $\bar{\mathbf{x}}$ 
  end if
end while
return  $\mathbf{x}^*, y^*$ 

```

often reduce the evaluation load. In practice, medium sized problem instances result in a few hundreds of configurations to evaluate per iteration, enabling a good scaling of the parallel evaluation of the full surrogates. However, the rigid restriction of the search directions severely limits the exploration capabilities and the escape from local minima, making **PDS** not suited for global optimization.

4.3.2 Bayesian optimization

BO aims to optimize an expensive black box objective function, using a low number of function evaluations [85, 174]. This approach relies on a surrogate to both decrease the computational cost of a query, and perform **UQ** of the prediction. Rather than directly optimizing the objective's predictions, **BO** maximizes an acquisition function that takes into account both the objective and its uncertainty, representing the uncertain gain. This achieves a balance between exploitation, which refers to minimization of the objective with low uncertainty and close to known configurations, and exploration of areas with high uncertainty. The configuration selected via acquisition is then validated with the expensive original objective function, and added to the training set of the surrogate to refine and repeat the search. The final optimum is obtained by selecting the best sample in the training set of the surrogate.

In this section we refer to **BO** implemented using **GPR**, although other **UQ**-capable surrogates have been used in the literature [24, 95]. We review some of the most popular acquisition function: negative lower confidence bound, the expected improvement, and the probability of improvement. **Negative lower confidence bound (NLCB)** [179] is an exploratory acquisition function for minimization problems, expressed as

$$\alpha_{\text{NLCB}}(\mathbf{x}) = -(\mu(\mathbf{x}) - \beta\sigma(\mathbf{x})), \quad (4.15)$$

where μ is the mean and σ is the standard deviation from the [GPR](#) posterior, with $\beta \geq 0$ weighting optimistically the contribution from the estimated uncertainty.

[Expected improvement \(EI\)](#) [213] is an exploratory acquisition function defined as

$$\alpha_{\text{EI}}(\mathbf{x}, y^*) = \mathbb{E}[(y^* - \mu(\mathbf{x}))_+] = (y^* - \mu(\mathbf{x}))\Phi\left(\frac{y^* - \mu(\mathbf{x})}{\sigma(\mathbf{x})}\right) + \sigma(\mathbf{x})\phi\left(\frac{y^* - \mu(\mathbf{x})}{\sigma(\mathbf{x})}\right), \quad (4.16)$$

where y^* denotes the best known value of the objective function, ϕ and Φ are the probability density function and the cumulative distribution function, respectively, for the standard normal.

[Probability of improvement \(PI\)](#) [100] is an exploitative acquisition function defined as

$$\alpha_{\text{PI}}(\mathbf{x}, y^*) = \Phi\left(\frac{y^* - \epsilon - \mu(\mathbf{x})}{\sigma(\mathbf{x})}\right), \quad (4.17)$$

where $\epsilon > 0$ prevents the acquisition function from selecting candidates excessively close to the best-known solution. A representation of a search step in [BO](#), using different acquisition functions, is given in Figure 4.2.

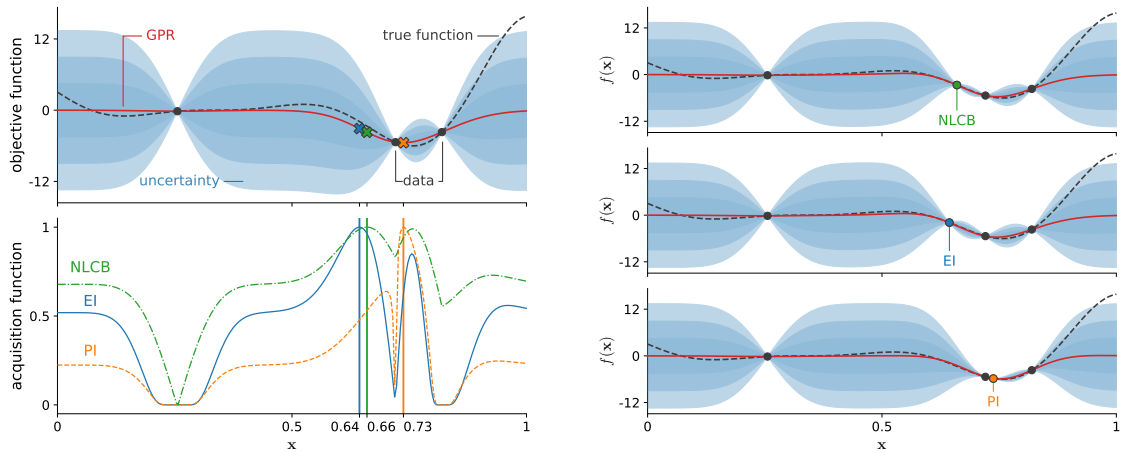


Figure 4.2: On the left, the GPR prediction with its uncertainty on the top and the acquisition functions at the bottom. On the right, the updated GPR after the addition of the previously selected sample, for each acquisition function.

The optimization of the acquisition function is carried out with quasi-Newton methods that can integrate the linear inequalities discussed in this chapter, as mentioned in Section 4.1. However, most [BO](#) implementations only allow continuous variables or impose a trivial rounding of the configuration found by the acquisition step. A common pitfall of such approach is that repeated acquisition might return the same configuration or an infeasible one, since the rounding ignores any information beyond the definition of $\mathcal{D}$. To address this behavior, we can adapt Problem (4.11) to enforce both feasibility search constraints, and reject the starting configuration. Problem (4.18) is the result of this

approach and reads

$$\begin{aligned}
 & \min \sum_{i \in I} \sum_{t \in \mathcal{D}_i} x_{it} |t - \bar{x}_i|^2, \\
 & \text{s.t.} \quad \sum_{t \in \mathcal{D}_i} x_{it} = 1, \\
 & \quad \sum_{i \in I} \sum_{t \in \mathcal{D}_i} x_{it} (\text{VCG}_i - \text{VCG}_{\text{crit}}) \mathbf{d}_i t \leq (\text{VCG}_{\text{crit}} - \text{VCG}_{\text{fixed}}) m_{\text{fixed}}, \\
 & \quad \sum_{i \in I} \sum_{t \in \mathcal{D}_i} x_{it} \mathbf{d}_i t \leq f(\mathbf{x}^*) - m_{\text{fixed}}, \\
 & \quad \sum_{i, t | \bar{x}_i = t} x_{it} \leq n_{\text{dim}} - 1, \\
 & \quad x_{it} \in \{0, 1\},
 \end{aligned} \tag{4.18}$$

$\forall i \in I, \quad \forall t \in \mathcal{D}_i,$

where Equation (4.12) has been added and $\sum_{i, t | \bar{x}_i = t} x_{it} \leq n_{\text{dim}} - 1$ ensures that the starting point $\bar{\mathbf{x}}$ is invalidated. If the rounded candidate is still a duplicate, we apply a random disturbance and repeat Problem (4.18) until an actual unvisited configuration is generated, or a maximum number of attempts is reached. Other approaches to **BO** on discrete domains use specialized kernels such as in [79, 130], but these are more suited to categorical variables rather than ordinals as in our setup. Another option is to use probabilistic models other than **GPR**, such as tree-structured Parzen estimators [24, 201], random forests [95], or sparse quadratic models [11, 51]. However, both the **UQ** and acquisition then rely on large random sampling, and often rely on segregated optimization on subsets of the variables.

As an additional heuristic, the acquisition function is switched whenever the optimization fails to improve the incumbent solution for a prescribed number of consecutive iterations. This strategy dynamically shifts the focus of the search between exploration and exploitation. For the stopping criterion, we can specify a maximum number of iterations or a time limit. The complete procedure is reported in Algorithm 6.

We remark that the initial $\mathbf{X}$ is obtained via Algorithm 4 on the union of the high-fidelity database, the **PF** from multi-objective optimization, and the set of configurations explored during the latest single-objective optimization, if available.

4.3.3 Particle swarm optimization

PSO is a widely used nature-inspired metaheuristic [74, 76, 215], due to the robustness of its performance on a wide variety of applications, the simplicity of its implementation, and the ability to leverage parallelism.

The main idea of the algorithm is to reproduce the flocking behavior of birds, where each individual is primarily drawn to the position of a leader, but is also influenced by its own personal history. Combined with randomness both in the starting configurations and during the update of the individuals, this simple rule is able to effectively explore the landscape of complex functions. The evolution of a population $\mathbf{X} \in \mathbb{R}^{m \times n_{\text{dim}}}$ is expressed in

Algorithm 6 Specialized BO implementation.

Input: initial configurations $\mathbf{X}$, objective function f , active linear inequality constraints $\mathbf{g}(\mathbf{x}) \leq \mathbf{0}$, discrete domain $\mathcal{D}$

Output: incumbent $\mathbf{x}^*$, y^*

```

evaluate  $\mathbf{y} = f(\mathbf{X})$ 
 $\mathbf{x}^* \leftarrow \operatorname{argmin}_{\mathbf{x} \in \mathbf{X}} f(\mathbf{x})$ 
 $\alpha \leftarrow \text{one of } \{\alpha_{\text{NLCB}}, \alpha_{\text{EI}}, \alpha_{\text{PI}}\}$ 
while not finished do
  train a GPR on  $\mathbf{X}, \mathbf{y}$ 
   $\mathbf{x} \leftarrow \operatorname{argmax}_{\mathbf{x}} \alpha(\mathbf{x})$  subject to  $\mathbf{g}(\mathbf{x}) \leq \mathbf{0}$ 
  while  $\mathbf{x}$  is infeasible or  $\mathbf{x} \in \mathbf{X}$  or iteration limit reached do
     $\mathbf{x} \leftarrow \text{solution to Problem (4.18)}$ 
  end while
  if  $f(\mathbf{x}) < f(\mathbf{x}^*)$  then
     $\mathbf{x}^* \leftarrow \mathbf{x}$ 
  end if
   $\mathbf{X} \leftarrow \mathbf{X} \cup \{\mathbf{x}^*\}$ 
   $\mathbf{y} \leftarrow \mathbf{y} \cup \{f(\mathbf{x}^*)\}$ 
  if  $f(\mathbf{x}^*)$  did not decrease in the last 100 iterations then
     $\alpha \leftarrow \text{another one of } \{\alpha_{\text{NLCB}}, \alpha_{\text{EI}}, \alpha_{\text{PI}}\}$ 
  end if
end while
return  $\mathbf{x}^*$ 

```

terms of a velocity field $\mathbf{V} \in \mathbb{R}^{m \times n_{\text{dim}}}$, which gets updated at each step through

$$\mathbf{V}^{t+1} = \mathbf{V}^t + c_1 \mathbf{r}_1 \odot (\mathbf{X}^{\text{pbest}} - \mathbf{X}^t) + c_2 \mathbf{r}_2 \odot (\mathbf{x}^* - \mathbf{X}^t), \quad (4.19)$$

where t denotes the iteration, c_1 and c_2 are hyper-parameters in $(0, 1)$, $\mathbf{r}_1$ and $\mathbf{r}_2$ are random vectors drawn from $[0, 1]^{n_{\text{dim}}}$, and $\odot$ is the elemental product operator. The coordinates of each individual's best objective evaluation are stored in $\mathbf{X}^{\text{pbest}}$, while the incumbent's position is $\mathbf{x}^*$. The coordinates are updated via

$$\mathbf{X}^{t+1} = \mathbf{X}^t + \mathbf{V}^{t+1}, \quad (4.20)$$

then the objective function is evaluated to update $\mathbf{X}^{\text{pbest}}$ and $\mathbf{x}^*$. Usually, the velocity field is subject to decay via multiplication with another hyperparameter $\omega \in (0, 1)$, which itself is decreased as the algorithm progresses.

The basic formulation of Equation (4.19) assumes continuous coordinates and does not include constraints. Domain bounds and linear inequalities can be easily enforced via a projection step, the addition of a penalty term, or by disabling the evaluation of infeasible configurations, but the support for discrete variables requires a more involved approach. Popular options include naive rounding, repair procedures, or reformulation of the velocity and position updates in terms of combinatorial structures [215, 33, 143, 41, 204]. In our application, we let [PSO](#) update the continuous coordinates $\mathbf{X}$ in the convex hull of $\mathcal{D}$, and evaluate the objective function on their repaired counterpart $\mathbf{X}^{\text{eval}}$ obtained

through Problem 4.11. Configurations in $\mathbf{X}$ that violate the active linear constraints are overwritten by the corresponding repaired elements of $\mathbf{X}^{\text{eval}}$.

Algorithm 7 Particle swarm optimization.

Input: initial feasible population $\mathbf{X}$, objective function f , $c_1, c_2, w_{\text{decay}} \in (0, 1)$, active linear inequality constraints $\mathbf{g}(\mathbf{x}) \leq \mathbf{0}$, discrete domain $\mathcal{D}$

Output: incumbent $\mathbf{x}^*, y^*$

$\mathbf{y} \leftarrow f(\mathbf{X})$

store per-particle best $\mathbf{X}^{\text{pbest}}, \mathbf{y}^{\text{pbest}}$

select $\mathbf{x}^*, y^*$ global best

initialize random initial velocities $\mathbf{V} \in (-1, 1)^{|\mathbf{X}| \times n_{\text{dim}}}$

$w \leftarrow 1$

while not finished **do**

 initialize random vectors $\mathbf{r}_1, \mathbf{r}_2$

 update $\mathbf{V}$ via Equation (4.19)

$\mathbf{X} \leftarrow \mathbf{X} + \mathbf{V}$

$\mathbf{X}^{\text{eval}} \leftarrow$ projection of $\mathbf{X}$ unto $\mathcal{D}$

 repair each $\mathbf{x} \in \mathbf{X}^{\text{eval}}$ that violates $\mathbf{g}(\mathbf{x}) \leq \mathbf{0}$ via Problem 4.18, update $\mathbf{X}$

 update $\mathbf{X}^{\text{pbest}}, \mathbf{y}^{\text{pbest}}, \mathbf{x}^*, y^*$

if y^* has changed **then**

 run local search to refine $\mathbf{x}^*$

 update $\mathbf{X}^{\text{pbest}}, \mathbf{y}^{\text{pbest}}, \mathbf{x}^*, y^*$

end if

$\mathbf{V} \leftarrow w\mathbf{V}$

▷ velocities decay

$w \leftarrow w_{\text{decay}}w$

end while

return $\mathbf{x}^*, y^*$

As an additional heuristic, when **PSO** finds a new global best configuration $\mathbf{x}^*$, we refine it with a single round of **PDS**. This hybrid approach accelerates the exploration of high-quality areas of the domain, and the results are propagated to all the population thanks to the velocity update.

The complete **PSO** routine is summarized in Algorithm 7, and Figure 4.3 represents a possible state of the algorithm. We remark that the constraint from Equation (4.12) is not used in **PSO**. In our ablation experiments, the addition of the constraints induced a slower convergence that we conjecture is due to a decrease in the diversity of the population following the repair step.

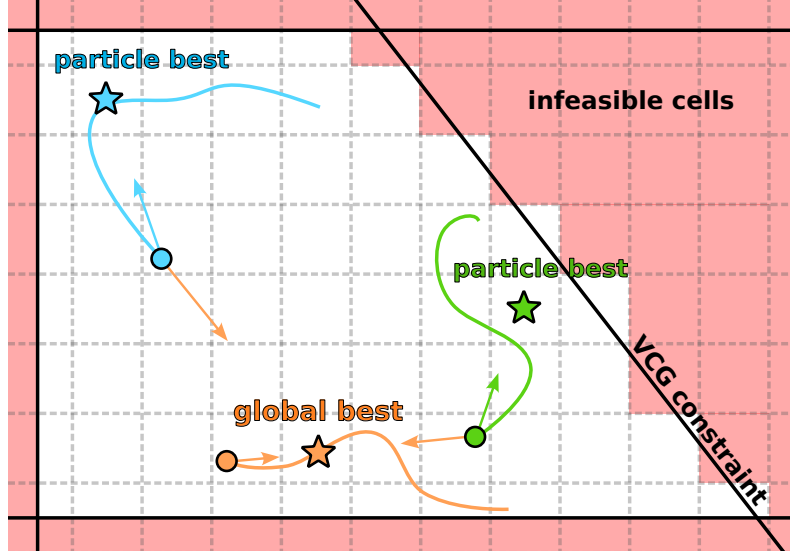


Figure 4.3: Representation of the velocity update calculation in particle swarm optimization. The grid cells correspond to the areas of the continuous domain that evaluate with the same $\mathbf{x}^{eval}$, the black lines represent the active linear constraints and domain bounds, and the cells in red are associated to infeasible $\mathbf{x}^{eval}$. The particles are represented with circles trailing a path of past positions, and the stars of the corresponding color show the associated $\mathbf{x}^{p^{best}}$. In this case, the global best $\mathbf{x}^*$ is in orange. The arrows show the components of the velocity update: the orange arrows are given by $\mathbf{x}^* - \mathbf{x}$ and the colored arrows come from $\mathbf{x}^{p^{best}} - \mathbf{x}$.

4.4 Numerical results

This section presents the evaluation of the methods described in this chapter. Section 4.4.1 discusses the multi-objective optimization of the base midship section MS00, in its initial parameterization of 5 parameters, studying the convergence of the infill criterion.

Section 4.4.2 focuses on the full ship FS6, and repeats the study on the multi-objective optimization of the base model with 10 parameters. The single-objective routines presented in this chapter are then compared on the refined 40-parameters model, which provides a sufficiently complex setting.

All values are obtained using a full node of Fincantieri S.p.A.'s HPC facility, equipped with an Intel Xeon Gold 6354 CPU (3.00 GHz), featuring 36 cores and 512 GB of RAM. All algorithms are implemented in Python and leverage the `multiprocessing` package.

4.4.1 Midship section MS00

To monitor the effect of our multi-objective criterion, we focus on the surrogate [QoIs](#) of the low-fidelity individuals in each [PF](#). We perform a sequence of 9 [NSGA-III](#) runs, each performing 10 iterations with population size 2000 and infill size 9. From each [PF](#), we select the individuals that enter the high-fidelity database in a future infill and compare the difference between the surrogate n_y , n_b and their high-fidelity values.

Figure 4.4 shows the decreasing infill criterion values and the behavior of the [QoIs](#)

surrogate errors. The errors quickly decrease for both **QoIs**, showing the efficacy of the proposed infill in refining the accuracy close to the **PF**. We observe that the fifth **PF** appears to produce an outlier in the first rank of the infill criterion values. The increase in the estimated value of the selected sample could come from the discovery of a new subset of the **PF**, either due to the progress in the exploration or the refinement of the surrogates.

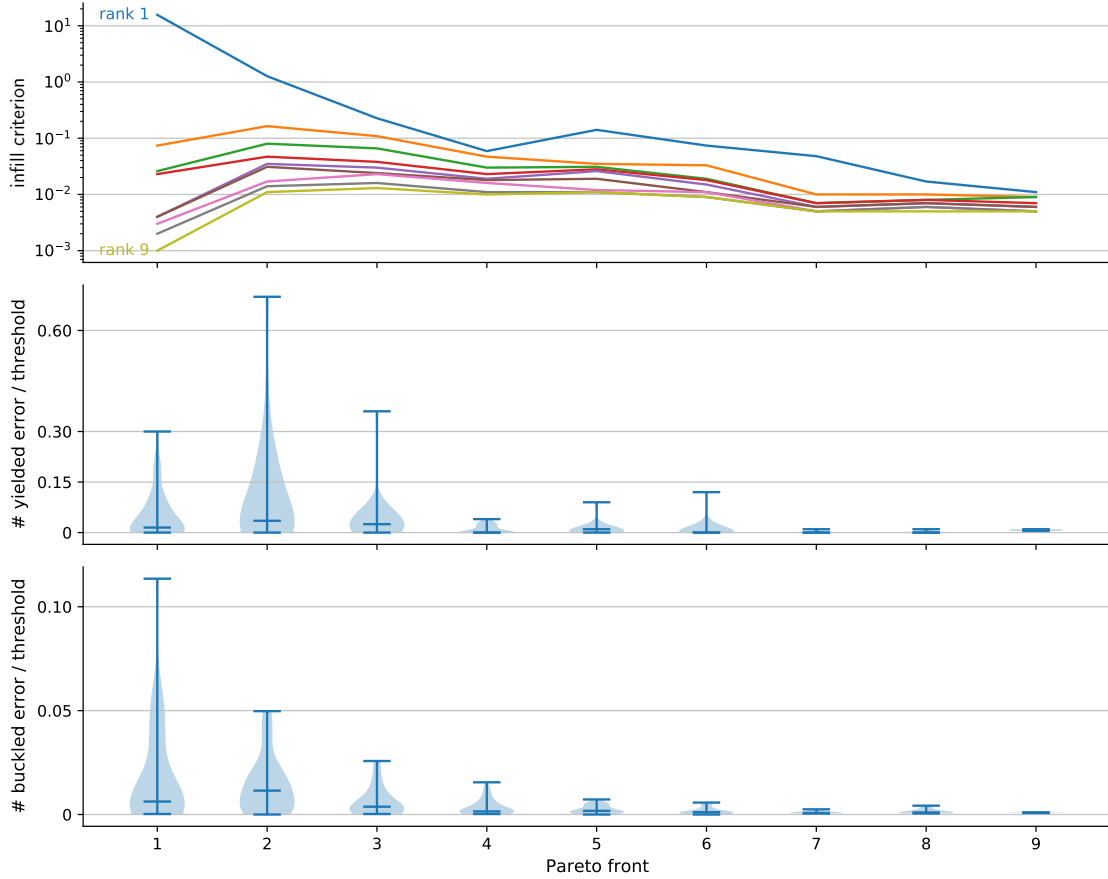


Figure 4.4: Comparison of multi-objective infill criterion values and structural QoIs errors corrected by the high-fidelity simulation, during repeated multi-objective optimization of MS00. Different colors of the criterion values denote the rank of the selected sample in the infill. The error plots highlight the median of the distribution.

Figure 4.5 shows the evolution of the approximated **PF**.

4.4.2 Full ship FS6

We repeat the analysis of the multi-objective infill criterion on the full ship FS6, for its initial parameterization with 10 parameters. We collect 9 **NSGA-III** runs, each performing 10 iterations with population size 2000 and infill size 9.

Figure 4.6 is similar to the midship section case, although the initial value of the infill criterion is much higher. In this more complex case, the descent of the infill criterion values and the errors are smoother, although the error values remain much higher relative to the critical thresholds.

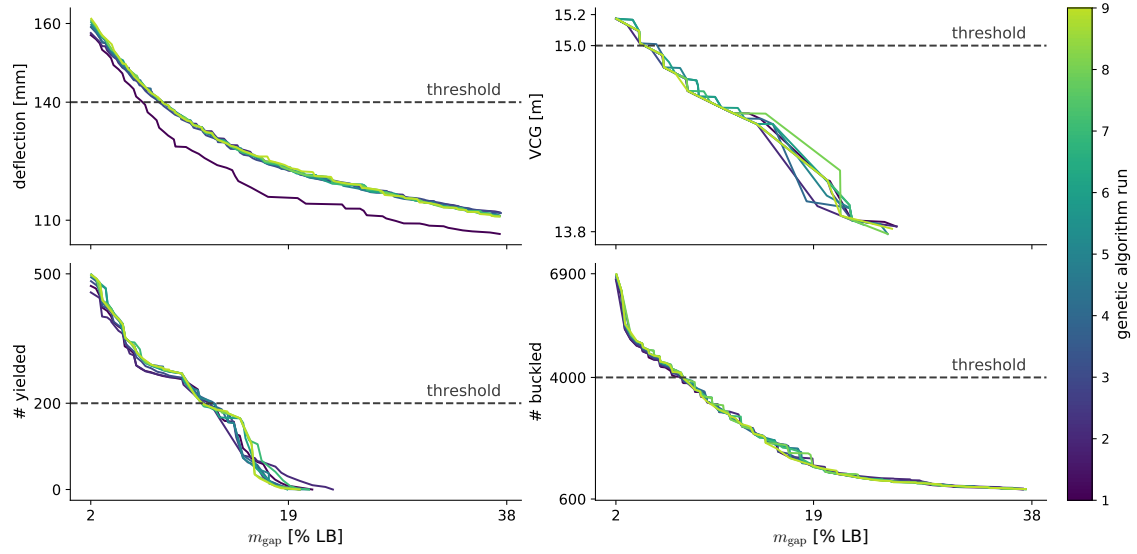


Figure 4.5: Evolution of the surrogate PF in subsequent runs of the genetic algorithm, for MS00. Each plot depicts the 2-dimensional PF computed from the projection of the full 5-dimensional PF.

Then, we use the most refined parameterization of FS6, with 40 parameters, to compare multiple approaches to black-box single-objective optimization. A single linear inequality constraint is defined on [VCG](#), and we start the optimization from 3 different high-fidelity databases to simulate different stages in the pipeline execution. The objective function evaluations are obtained from the full surrogates constructed in a previous pipeline execution.

For our comparison, we select two [BO](#) implementation: Algorithm 6, for [GPR-based BO](#), and Optuna, [2] for its [tree-structured Parzen estimator \(TPE\)](#). The [GPR](#) implementation uses GPy [84], while the acquisition-update loop is implemented with Emukit [136, 135]. The [GPR](#) uses [automatic relevance determination \(ARD\)](#) and the kernel is fitted using [L-BFGS](#) [217] using 20 random restarts and 1000 iterations. The acquisition is optimized using `trust-constr` [31] from SciPy’s [197] `minimize` package, starting from the incumbent with a limit of 1000 iterations. Problem 4.18 is implemented using `python-mip` [161] and uses the coin-or branch-and-cut solver [72] as backend.

Optuna is a popular open-source framework for hyper-parameters optimization, and support mixed-variable optimization domains. The [BO](#) functionality of Optuna uses [TPE](#) to estimate the probability of a sample to place in the upper or lower quantiles of the objective values. Maximization of this probability then drive the exploration towards high-quality areas of the domain. Optuna supports arbitrary constraints, implemented in a weak sense through a penalization mechanism during sample selection, as well as mixed domains and batched acquisition of new samples.

We then use NGOpt from Nevergrad [154, 23] as a generic option for black-box optimization. Nevergrad is an open-source framework for derivative-free optimization, which supports mixed-variable domains and arbitrary constraints, and provides an

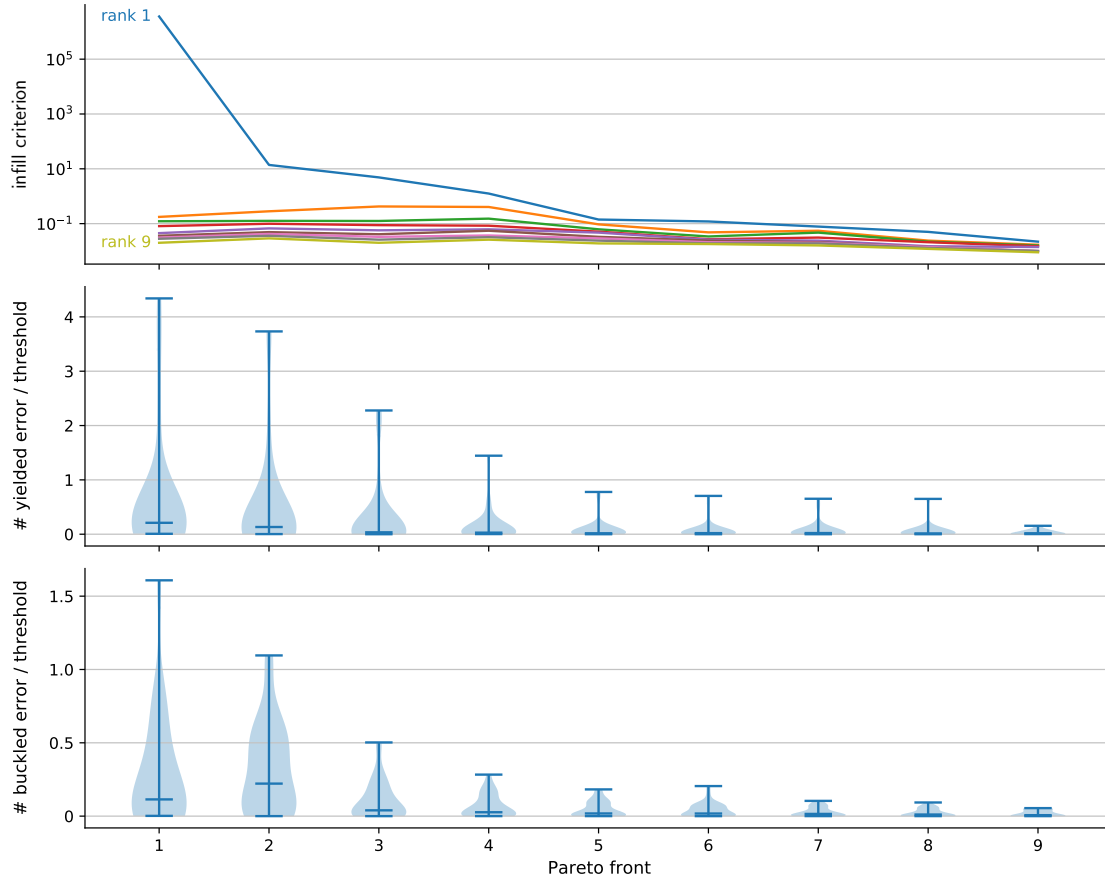


Figure 4.6: Comparison of multi-objective infill criterion values and structural QoIs errors corrected by the high-fidelity simulation, during repeated multi-objective optimization of FS6. Different colors of the criterion values denote the rank of the selected sample in the infill. The error plots highlight the median of the distribution.

automated selection of the optimization strategy. NGOpt takes the definition of the domain and an estimate of the computational budget, and assembles a sequence of stages from the vast catalog of optimization algorithms implemented in Nevergrad, such as evolutionary strategies, GA, and more. Constraints that are cheap to evaluate, such as our linear inequalities, are enforced via rejection sampling. On this test case, NGOpt selects a (1, 1) evolutionary strategy [10] which supports batched evaluation of the objective function.

Finally, we consider two implementations of population-based metaheuristics: PSO as described in Algorithm 7 and DE [147, 48, 212]. PSO uses $c_1 = c_2 = 0.5$ and $\omega_{\text{decay}} = 0.99$. For the DE implementation, we use the classic "best1bin" crossover strategy which combines each individuals with the best configuration in the population. All constraints are enforced by Problem 4.11 in a repair step.

We use three different high-fidelity databases as starting points for the optimization, selected from a previous execution of the pipeline on FS6. The smallest database contains only the configuration with minimum thicknesses, the second database consists of

30 configurations, and the largest one contains 84 samples. For each algorithm, the initial sample or population is generated from the available high-fidelity database using Algorithm 4. A common random seed is used to generate 100 initial individuals, ensuring fairness. The runtime limit of the optimization is 10 minutes, and each run is repeated 10 times.

Figure 4.7 compares the evolution of the best known optima, expressed using Equation (2.8), emphasizing the mean and the minimum-maximum range across the different runs. The first incumbent value comes from the initial random sampling in Algorithm 4, and gives $m_{\text{gap}} = 42.8\%$. Optuna shows the weakest performance and is not able to decrease the initial optimum. BO is able to aggressively decrease the incumbent in the first few iterations, but quickly becomes unable to decrease it further. This suggests an issue with the exploratory abilities of the two Bayesian optimization approaches, with neither of the two acquisition strategy able to match the other algorithms. NGOpt, DE, and PSO all exhibit similar behavior and smooth descent. In this case, NGOpt appears to stop decreasing the incumbent earlier than the others, although all three algorithms become almost flat at the end of the run. Overall, PSO achieves the best optimum with 2.3% gap from DE in terms of averages.

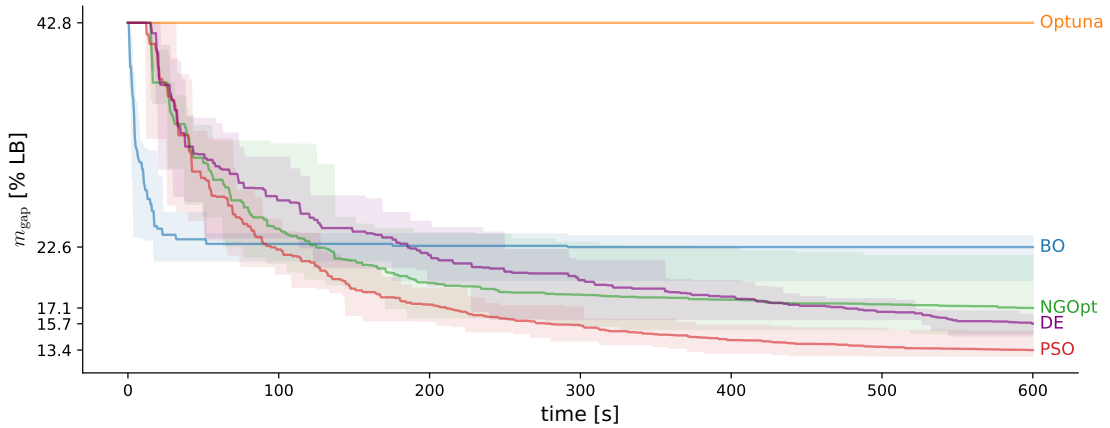


Figure 4.7: Comparison of single-objective optimization algorithms, starting from a single high-fidelity sample. The test case is FS6 with 40 parameterized sections.

In the second experiment, shown in Figure 4.8, the starting incumbent of $m_{\text{gap}} = 33.7\%$ comes from the high-fidelity database with 30 samples. The behavior is similar to the previous case, but this time Optuna is able to slightly decrease the incumbent. BO ends in the same range as the previous case, while all other algorithms are able to achieve better optima and maintain the same ranking.

Finally, Figure 4.9 compares the results starting from an initial incumbent of $m_{\text{gap}} = 16.9\%$, obtained from 84 high-fidelity samples. Differently from the previous cases, DE is unable to reliably decrease the incumbent, although it shows a large difference between the best and worst cases. BO and Optuna both fail to decrease the initial optimum. NGOpt benefits from the lower starting incumbent, but is still unable to match PSO. PSO achieves a

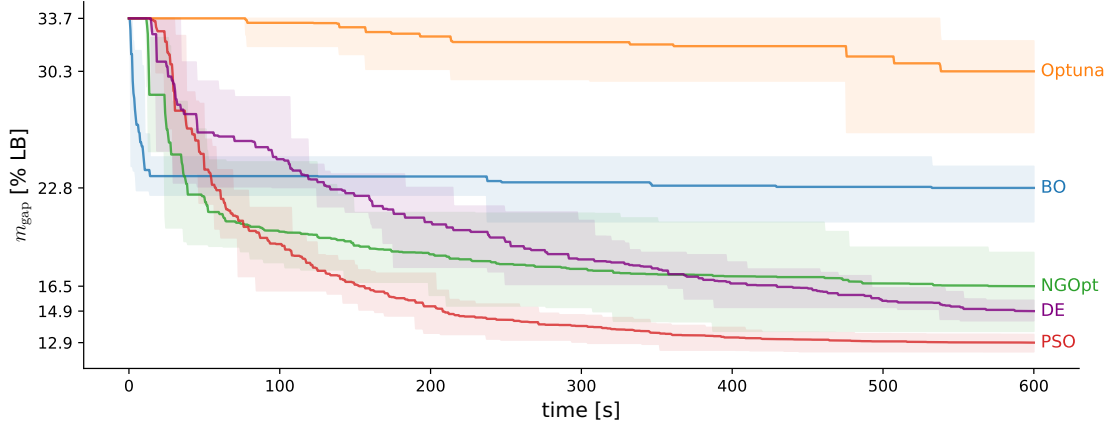


Figure 4.8: Comparison of single-objective optimization algorithms, starting from 30 high-fidelity samples. The test case is FS6 with 40 parameterized sections.

mean optimum very close to the previous case, but with a much larger difference between the minimum and maximum value.

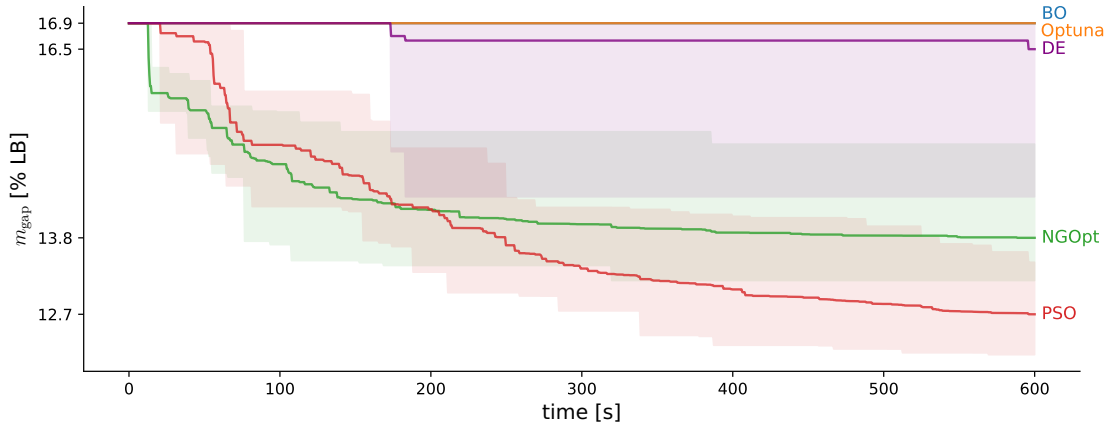


Figure 4.9: Comparison of single-objective optimization algorithms, starting from 84 high-fidelity samples. The test case is FS6 with 40 parameterized sections.

Overall, **PSO** emerges as the more robust and scalable option among the evaluated algorithms, since it consistently achieves the best optimum values on all sizes of the high-fidelity database. Optuna appears to be heavily penalized by the weak enforcement of constraints and subpar exploratory capabilities. **BO**, while successful in finding good configurations within few iterations, is unable to progress much further and suffers on larger sizes of the high-fidelity database. **DE** shows strong exploratory capabilities on smaller databases, but is not reliable on the largest one. NGOpt provides robust performances, benefiting from lower starting points and reliably decreasing the incumbent on all database sizes, but is weaker than our **PSO** implementation in all tests.

4.5 Conclusions

This chapter discussed suitable approaches to multi- and single-objective surrogate-assisted optimization of the structural problem described in Chapter 2, taking into account the discrete domain and the linear constraints on the components of the [COG](#).

Unconstrained multi-objective optimization is carried out with [NSGA-III](#) due to its effectiveness and ability to leverage the efficient surrogates constructed in Chapter 3. We presented an infill criterion, defined on the [GPR](#) components of the surrogates, that selects the best candidates to refine the surrogates' predictions close to the [PF](#). Numerical results on the coarse parameterization of the midship section MS00 and the full ship FS6 show the effectiveness of our approach in refining the predictions of the approximate [PF](#).

For the single-objective setting, effective constraint handling constitutes a crucial factor to achieve high-quality solutions. We implemented an [ILP](#)-based repair procedure capable to simultaneously enforce the discrete nature of the decision variables and the linear constraints imposed on the components of the [COG](#). The proposed repair procedure can be easily integrated in continuous-space optimization routines, significantly improving their effectiveness. On a realistic test case, defined on the full ship FS6 with high parameter count, our [PSO](#) implementation hybridized with [PDS](#) consistently achieves the best results compared to other custom approaches and established frameworks alike. Model-based alternatives such as [GPR](#)-based [BO](#), although effective in quickly finding good configurations, show insufficient exploratory abilities compared to stochastic metaheuristic algorithms.

Chapter 5 will present a problem refinement procedure that enables the pipeline to surpass the results obtained during optimization, through the adaptation of the problem parameterization to the emerging structural responses close to the optimum.

RESPONSE-BASED REPARAMETERIZATION

This chapter introduces a reparameterization procedure, able to refine the optimization problem from Chapter 2 via the addition of new decision variables to better control the structural behavior of the model.

After the optimization pipeline terminates and an optimal configuration for the starting parameterization has been found, we analyze the structural [quantities of interest \(QoIs\)](#) of configurations close to the optimum. By studying the structural behaviors at the patch level, we construct a series of optimization problems to split the assignments of patches and decision variables, thus enhancing the formulation of the optimization problem. From the resulting subset of patches, we build a new parameterization that is able to achieve lower structural mass, while enforcing the constraints on yielding and buckling behavior. Thanks to the splitting approach, the reparameterization operation builds a hierarchy of decision variables and allows to retain the high-fidelity database built by the previous execution of the pipeline. After a small sampling of the larger domain and the construction of new surrogates, the optimization pipeline can be executed on a more difficult, but more expressive, problem formulation.

Section 5.1 motivates the need for a parameterization refinement procedure and the intuition behind our approach. The actual procedure is presented in Section 5.2, and the numerical results on the midship section MS00 and full ship FS6 are discussed in Section 5.3. The final remarks are presented in Section 5.4.

5.1 Introduction

The formulation of optimizations problems is a crucial element of obtaining high-quality solutions in an effective manner. Problems with a large number of decision variables can, in principle, exercise a very precise control on the objective function and the quantities subject to constraints, allowing a sufficiently powerful optimization procedure to efficiently reach the optimal configuration of parameter values.

In our setting, this translates to the definition of a separate decision variable for each patch in the model, which could reach ten thousands for a full ship. The designer would be able to perfectly allocate the thickness so that the minimum value is assigned to patches with low stresses, and higher thicknesses are only used for patches that incur in higher tensions. However, navigating high-dimensional spaces with little to no known structure, such as in black-box optimization, often incurs in high computational costs due to the evaluation of large parts of the domain that do not results in any effect on the objective function. Conversely, a low-dimensional formulation where a single decision variable influences patches with very different loads is inefficient: if the chosen thickness is low, the number of yielded and buckled elements will be large and if instead the thickness is high, the extra mass on unloaded patches is a misallocation of resources.

Relying on the designers' intuition and experience to correctly map the ship's patches to decision variables is a flawed approach. On the one hand, the ships often present very different geometrical features and structural interactions compared with past designs, invalidating the experience of the designer. On the other hand, much of the naval engineering literature on global structural resilience is based on prismatic beam theory, which naturally ingrained in the practitioners a very simplistic model of stress distribution. Indeed, the earliest application of the optimization pipeline [190] only assigned decision variables to a subset of the decks, and at most separated the aft, center and fore sections. An example of this approach is shown in Figure 5.1 for FS0.

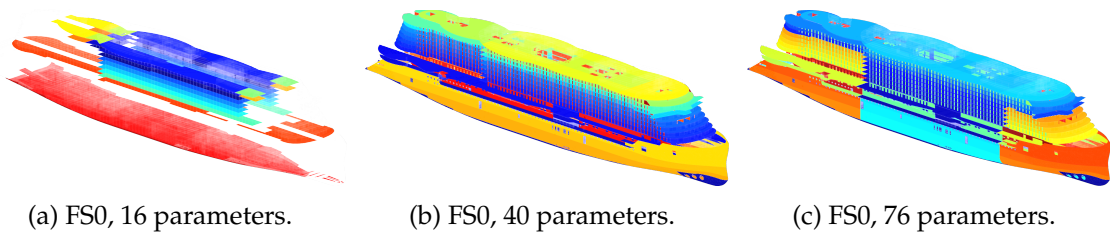


Figure 5.1: Three parameterizations of FS0, produced by the designers. On the left, 16 decision variables are assigned to the higher accommodation decks and the bottom. At the center and the right, the parameterizations extend to the shell and the bulkheads, and use 40 and 76 parameters respectively. Different colors denote different parameterized sections, and the unparameterized parts of the ship are transparent.

To overcome these limitations, we aim to design an automated procedure that is able to correctly assess the differences and similarities between the structural behavior of the

patches and produce an effective mapping of the ship's sections to a limited number of decision variables.

The first option is to start from a very fine parameterization, and reduce the number of decision variables. In the scientific literature, parameter space reduction seeks to remove the challenges associated to high-dimensional domains by identifying low-dimensional manifolds, that retain the expressivity of the original parameterization with a much lower number of decision variables. A classic application is in shape optimizations using free-form deformation [169], where the decision variables correspond to the nodal displacement of the control points. Among the proposed approaches, we mention [active subspace \(AS\)](#) [45, 120, 55, 158] and parametric model embedding [171, 59, 172], and generative models based on deep neural networks such as autoencoders [109, 17]. However, due to the large number of patches in a full ship model, the number of samples necessary for an effective analysis of the parametric space would be infeasible to generate, even during the offline phase. As solution, we could let the designers create a parameterization with a moderately large number of dimensions, and apply parameter space reduction. In our experience, this approach relies on the intuition and experience of the designers and does not produce good results when applied to complex global models.

Our approach to produce an effective parameterization is to start from a coarse initial proposal, mapping large groups of patches to a low number of decision variables, and study the structural responses to appropriately modify the set of parameters. To generate the initial set of decision variables, as mentioned in Chapter 2, we subdivide the patches based on the regulatory minimum thickness and their orientation, taking also into account the construction plan. After the execution of the optimization pipeline, the optimal configuration is examined to assess the efficacy of the parameterization. A typical outcome is represented in Figure 5.2, where the vast majority of elements are sufficiently strengthened by a thickness of 6 mm and only a handful require a higher value.

The designers could then split the original grouping of the patches to create two separate decision variables, and execute a new optimization to select the best values. However, three issues arise from this approach. First, the designer has no easy way to determine with certainty which patches should be assigned to a new different decision variable. Second, the setup of a new problem is cumbersome and error prone. Third, executing the pipeline from scratch loses all the valuable information accumulated during the optimization of the coarse model.

We propose an automatic refinement of the parameterization based on the solution of a set of [integer linear programs \(ILPs\)](#) for the minimization of penalized mass, which will determine a division of each group of patches according to the optimal thickness found. This approach has been inspired by a similar idea in the context of multiscale finite elements [18], where the number of expensive problems to be solved at the finer level was reduced via a clustering approach. By determining a limited number of centroids in the space of boundary conditions of the sub-elements, the authors approximated the solution to the other members of the cluster and achieved great reduction of computational cost.

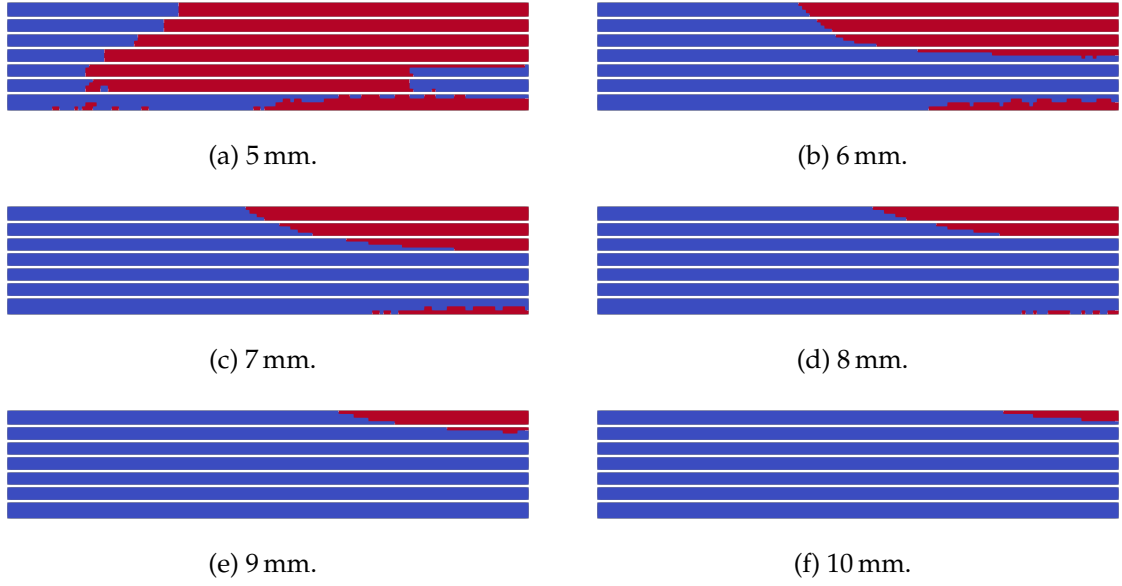


Figure 5.2: Buckling behavior for x_4 in MS00, obtained by modifying the value of the parameter while keeping the remaining thicknesses fixed at their optimal values. Red indicates buckled elements.

5.2 Reparameterization

In each parameterized section, we aim to find a thickness assignment which uses a prescribed number of different values and minimizes the total mass, penalized by yielded and buckled elements. If successful in reducing the mass of the parameterized section compared to the current parameterization, each thickness value then identifies a better assignment of patches to decision variables.

Let P_i be the set of patch indices assigned to the i -th parameterized section x_i , and $\mathcal{D}_i$ the set of possible parameter values. Additionally, let y_{pt} be the number of elements in patch $p \in P_i$, when $x_i = t$ that yield under any of the load conditions, and let b_{pt} be the analogue quantity for buckling. We assume that each patch is independent of the rest of the ship, so that y_{pt} and b_{pt} combine linearly and can be computed (using surrogates, and high-fidelity evaluations when available) by enumerating $t \in \mathcal{D}_i$ while keeping the rest of the parameter values fixed. The values of y_{pt} and b_{pt} can be determined thanks to the full-scale approach taken to the computation of n_y and n_b , where we aggregate over each patch instead of the full mesh.

The binary decision variables x_{pt} express the assignments of patches to parameter values, with $x_{pt} = 1$ only if patch p has thickness t . For consistency, for each patch p , it is required that exactly one of the x_{pt} variables is nonzero. The upper bound on the [vertical center of gravity \(VCG\)](#) takes the same form used in Equation (2.13). For the objective function, we aim to minimize the total mass of the patches in P_i , their reinforcement bars, and penalty terms given by the sum of squares of y_{pt} and b_{pt} . Finally, we fix the actual number of different parameter values $n_{\text{clusters}} \geq 2$ through the additional decision variables

u_t and its consistency constraints.

We obtain the following optimization problem:

$$\begin{aligned}
\min \quad & \sum_{p \in P_i} \sum_{t \in \mathcal{D}_i} x_{pt} (\mathbf{d}_p t + m_{\text{bar}} b_{pt} + c_y y_{pt}^2 + c_b b_{pt}^2), \\
\text{s.t.} \quad & \sum_{t \in \mathcal{D}_i} x_{pt} = 1, & \forall p \in P_i, \\
& x_{pt} \leq u_t, & \forall p \in P_i, \forall t \in \mathcal{D}_i, \\
& \sum_{p \in P_i} x_{pt} \geq u_t n_{\min}, & \forall t \in \mathcal{D}_i, \\
& \sum_{t \in \mathcal{D}_i} u_t = n_{\text{clusters}}, \\
& \sum_{p \in P_i} \sum_{t \in \mathcal{D}_i} x_{pt} (\text{VCG}_p - \text{VCG}_{\text{crit}}) \mathbf{d}_p t \leq (\text{VCG}_{\text{crit}} - \text{VCG}_{\text{res}}) m_{\text{res}}, \\
& x_{pt} \in \{0, 1\}, & \forall p \in P_i, \forall t \in \mathcal{D}_i, \\
& u_t \in \{0, 1\}, & \forall t \in \mathcal{D}_i,
\end{aligned} \tag{5.1}$$

where $\mathbf{d}_p$ is the linear density of patch p , VCG_p is the **VCG** of patch p , and VCG_{res} and m_{res} are the **VCG** and mass, respectively, of all elements not controlled by the current parameter. The minimum number of patches in an assignment is $n_{\min} \geq 1$. This prevents the generation of clusters, and thus new decision variables, that would have a negligible effect on the total mass.

The total number of binary variables is $(|P_i| + 1)|\mathcal{D}_i|$ and the number of constraints is $|P_i||\mathcal{D}_i| + |P_i| + |\mathcal{D}_i| + 2$. A representation of the procedure is depicted in Figure 5.3, considering a simplified section composed of 6 patches.

We remark that the objective functions in Problem (5.1) and Equation (4.7) are fundamentally different in the way they handle the penalization of yielded and buckled elements, as the former uses a sum of squares and the latter uses the square of a sum.

In Problem (5.1), the objective function can be split in the contributions from each patch to the physical mass, due to thickness and additional stiffeners, and the penalization of failed elements as

$$\sum_{p \in P_i} \sum_{t \in \mathcal{D}_i} x_{pt} \left(\underbrace{\mathbf{d}_p t}_{\text{patches mass}} + \underbrace{m_{\text{bar}} b_{pt}}_{\text{reinforcement bars}} + \underbrace{c_y y_{pt}^2 + c_b b_{pt}^2}_{\text{per-patch penalty}} \right), \tag{5.2}$$

which is linear in the decision variables x_{pt} . The proper reformulation of Equation (4.7) in terms of x_{pt} requires instead the quadratic programming expression

$$\sum_{p \in P_i} \sum_{t \in \mathcal{D}_i} x_{pt} \left(\underbrace{\mathbf{d}_p t}_{\text{patches mass}} + \underbrace{m_{\text{bar}} b_{pt}}_{\text{reinforcement bars}} \right) + f_{\text{pen}}(x), \tag{5.3}$$

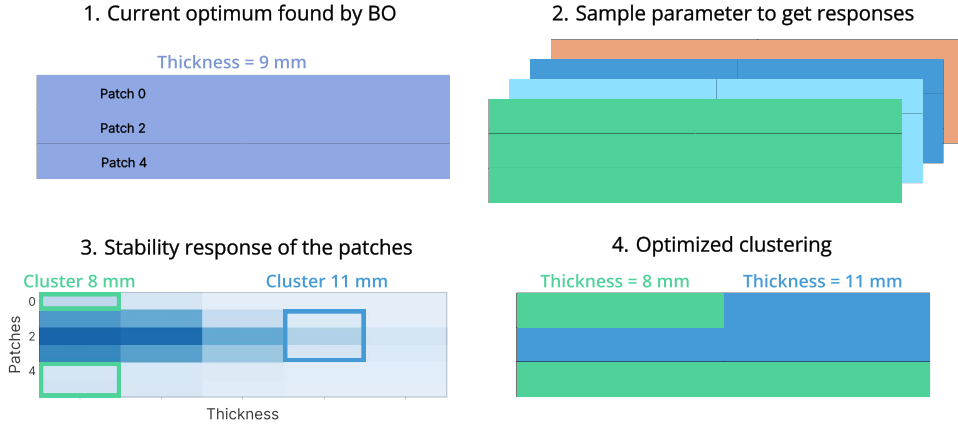


Figure 5.3: The reparameterization procedure applied to a simplified deck composed of 6 patches. The patch colors in steps 1, 2, and 4 represent different thicknesses. In step 1, all patches have the same thickness as selected by the single-objective optimization. In step 2, different thicknesses are evaluated to obtain the structural QoIs of each patch. In step 3, the structural responses determine the reparameterization problem, and its optimal solution clusters the patches in two groups. In step 4, the optimized thicknesses are assigned to the patches and the new mapping to the decision variables is thus obtained.

where the penalty term for the binary decision variables, corresponding to Equation (4.8), is

$$f_{\text{pen}}(x) = c_y \underbrace{\left(\sum_{p \in P_i} \sum_{t \in \mathcal{D}_i} x_{pt} y_{pt} \right) - y_{\text{crit}}}_{\text{global yielding penalty}}^2 + c_b \underbrace{\left(\sum_{p \in P_i} \sum_{t \in \mathcal{D}_i} x_{pt} b_{pt} \right) - b_{\text{crit}}}_{\text{global buckling penalty}}^2. \quad (5.4)$$

The thresholds y_{crit} and b_{crit} in Equation (5.4) need to be adapted to the current s -th parameterized section, as Equation (4.8) uses global values. One option would be subtracting the yielded and buckled elements totals of the other sections from the critical values, but the combination of the separate problems could still exceed the global constraints. As an alternative, Problem (5.1) needs to be reformulated as a global optimization problem, optimizing all the parameterized sections at once, which is impractical for large models. In practice, the usage of Problem (5.1) leads to a more severe penalization of failure phenomena in each patch, so that the updated model will offer better trade-offs during the optimization.

By solving Problem (5.1) for each parameterized section, and possibly for increasing n_{clusters} , we obtain multiple different subdivisions of the mapping from patch to variables. For each section, the solutions that do not improve upon the trivial solution for $n_{\text{clusters}} = 1$ by a user-specified amount are discarded. At most one refinement can be applied to each parameterized section, including the trivial one which leaves it unchanged. Moreover, the designers specify a maximum number of parameters for the model. The selection of the optimal set of subdivisions can be obtained by the solution of a knapsack problem [166], where the cost of a refinement is the number of clusters and its value is given by Problem (5.1).

Since the construction of the **VCG** constraint assumes that all the elements outside the current group of patches do not change mass, the parameter configuration obtained by joining the optimized assignments might violate the global **VCG** constraint. Indeed, the **VCG** constraint in each local Problem (5.1) has slack dependent on the current optimum, so in an extreme case each section with **VCG** above the global threshold could increase its **VCG** contribution, and the opposite for sections with **VCG** below the global threshold. The actual changes in **VCG** contribution could then violate the global **VCG** constraint. Even so, the goal of the procedure is to obtain an expressive parameterization: the locally-optimal thickness assignments will be overwritten by the following optimization procedure, which enforces the global **VCG** constraint.

Once the optimal selection of clusters has been chosen, the i -th parameterized section associated to $\mathbf{x}_i$ is also partitioned in a set of patch clusters $\{\mathcal{P}_j\}_{j=1}^m$ with $1 \leq m \leq n_{\text{clusters}}$. The patches in each cluster $\mathcal{P}_{j>1}$ will be assigned to a new parameter $\mathbf{x}_k$, with feasible domain $\mathcal{D}_i$, decreasing the number of patches assigned to $\mathbf{x}_i$. With the new parameterization, the configurations generated with the coarse model can still be retrieved by enforcing the constraints

$$\mathbf{x}_i = \mathbf{x}_k \quad \forall k \in \mathcal{K}_i, \quad (5.5)$$

where $\mathcal{K}_i$ collects the indices of the $m - 1$ parameters generated for the clusters $\mathcal{P}_{j>1}$. This feature is crucial for the efficiency of the optimization procedure, as all the previous high-fidelity simulations can be translated to the new parameterization without the need for a new simulation. Indeed, all the high-fidelity samples collected so far can be viewed as coming from a constrained version of the current parameter space, where Equation (5.5) reduced the feasible configurations. Due to this consideration, the surrogates must be rebuilt on a sampling where Equation (5.5) is not active, so that the effects of the new parameterization are properly observed. For this task, we perform a maximin sampling on the updated domain, considering the translated high-fidelity database as fixed.

At this point, the surrogates are rebuilt on the updated high-fidelity samples and the optimization, both multi- and single-objective, can take place on a more expressive model. With this approach, the parameterization is adapted as the optimization progresses and is thus able to overcome the biases that the designers could have introduced in the creation of the initial model.

5.3 Numerical results

This section presents the application of the reparameterization procedure to the midship section MS00 and the full ship FS6. In both cases, we start with the coarse parameterization and perform multi-objective and single-objective mass optimization. From the optimal configuration, we set an upper limit to the number of total parameterized sections and apply three independent refinement approaches:

- iterative: we alternate the automated procedure from Section 5.2 with $n_{\text{clusters}} = 2$, followed by the optimization the updated model,
- one-shot: we refine the parameterization a single time to the maximum number of parameters, and optimize the updated model,
- designer: we let the designers refine the parameterization to the maximum number of parameters, and optimize the updated model.

All values are obtained using Fincantieri S.p.A.'s HPC facility, composed of 3 identical nodes equipped with an Intel Xeon Gold 6354 CPU (3.00 GHz), featuring 36 cores and 512 GB of RAM.

Section 5.3.1 presents the results on the midship section MS00, while Section 5.3.2 repeats the analysis on the full ship FS6.

5.3.1 Midship section MS00

The optimization pipeline for MS00 starts with an initial maximin sampling of size 21, and the full surrogates are built using truncation rank 6 as discussed in Chapter 3.

Multi-objective optimization alternates [non-dominated sorting genetic algorithm III \(NSGA-III\)](#) and the infill procedure described in Section 4.2, until the value of the criterion for the first selected candidate falls below 0.05 or a maximum number of high-fidelity configurations have been added to the database. The infill size is 9, chosen to maximize the computational resources available.

The designers then select the critical threshold for the structural [QoIs](#) as $y_{\text{crit}} = 200$, $b_{\text{crit}} = 4000$ and $d_{\text{crit}} = 140$ mm. Single objective optimization first loops between Bayesian optimization, using Algorithm 6, and infill of the 3 best configurations found, until the search is unable to provide better candidates. Then, the optimization is progressed using [principal directions search \(PDS\)](#) from Algorithm 5. In this case, the optimization ends with optimum $m_{\text{gap}} = 15.44\%$ and a high-fidelity database of size 96.

At this point, the optimum configuration is analyzed for the parameterization refinement, as in Section 5.2. We first focus on the iterative approach, where we set $n_{\text{clusters}} = 2$ and $n_{\text{min}} = 1$.

Figure 5.4 shows the results on $\mathbf{x}_4$, where the procedure selects thicknesses 6 and 9 mm to minimize the penalized mass of the patches. We remark the behavior of the procedure on the top-right corner of $\mathbf{x}_4$: the three patches are characterized by very high tensions, as seen previously in Figure 5.2, and would require a very high value of the parameter to reduce their buckling. The evaluation of the trade-off between higher thickness and lower structural [QoIs](#) results in the critical patches being assigned to the group of patches with lower thickness.

On MS00, the reparameterization creates 5 new parameters, for a total of 10. With the updated model and the conversion of the high-fidelity database, we select a small set of configurations on the enlarged domain to produce new surrogates. We perform

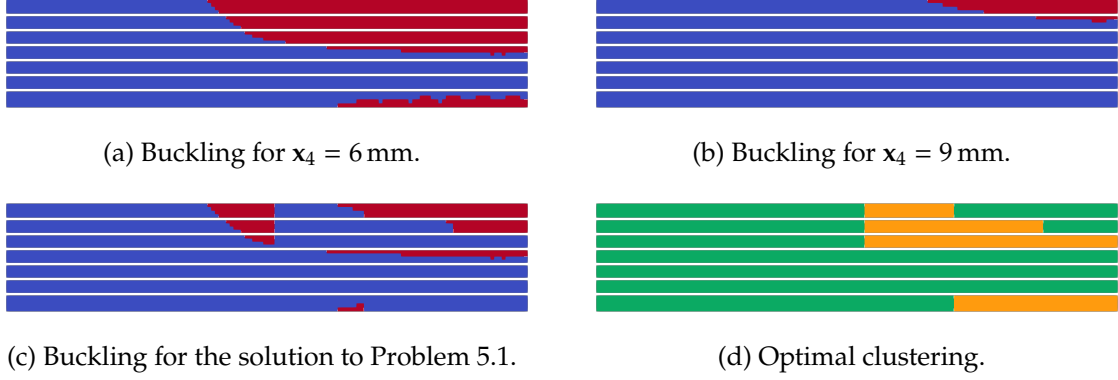


Figure 5.4: The reparameterization of x_4 in MS00 selects two thickness values, 6 and 9 mm. The buckling behavior for a single parameter is represented by the figures on top, with red indicating buckled elements. Figure 5.4c shows the solution to Problem 5.1, and Figure 5.4d shows the corresponding grouping of patches.

a maximin sampling of the new domain, adding 19 new configurations other than the optimal thicknesses selected during Problem 5.1. To construct the new surrogates, we require a truncation rank for the [proper orthogonal decomposition \(POD\)](#): following the discussion from Chapter 3, we choose to increment the previous rank by the number of newly created parameters. Figures 5.5 and 5.6 demonstrate the robustness of the choice by showing the evolution of the singular values decay and comparing the error distributions of n_y and n_b before and after the optimization phase, for multiple truncation ranks.

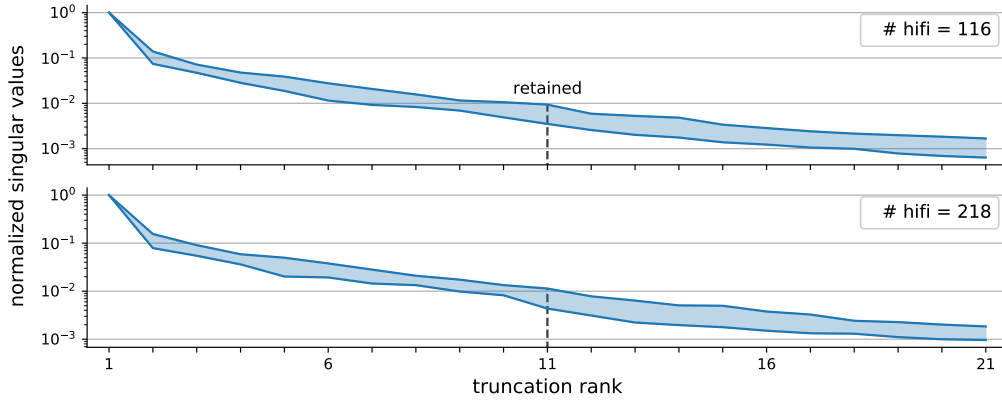


Figure 5.5: Evolution of the minimum and maximum for the normalized singular values decay of the stress tensor components, for the midship section with 10 parameters. Values are reported for the high-fidelity database before and after optimization.

The optimization of MS00 with 10 parameters ends with $m_{\text{gap}} = 7.70\%$ and high-fidelity database of size 218. The large decrease of m_{gap} proves that the reparameterization was successful in adapting the decision variables to the structural behavior of the model, achieving a much lower mass for the same structural constraints.

The reparameterization is then repeated a second time, adding 7 new parameters for a total of 17. We extend the high-fidelity database with the same approach described

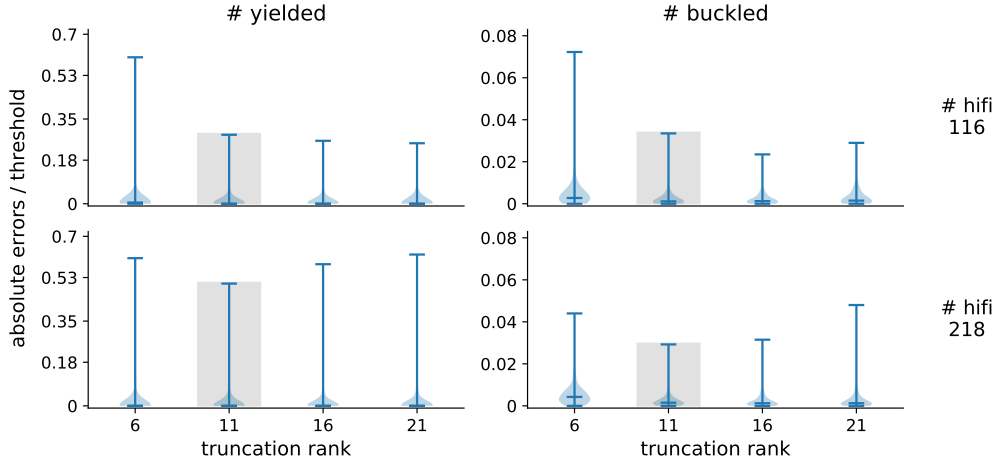


Figure 5.6: Evolution of the 5-CV error distribution for the midship section with 10 parameters. Values are reported for the high-fidelity database before and after optimization. The gray box identifies the chosen truncation rank.

before, adding 20 new configurations. The following optimization is unable to achieve m_{gap} much lower than the 10 parameters model, with the best configuration reaching 7.57% with a high-fidelity database of size 352. Figures 5.7 and 5.8 show the evolution of the singular values decay and compare the error distributions before and after optimization, for multiple truncation ranks.

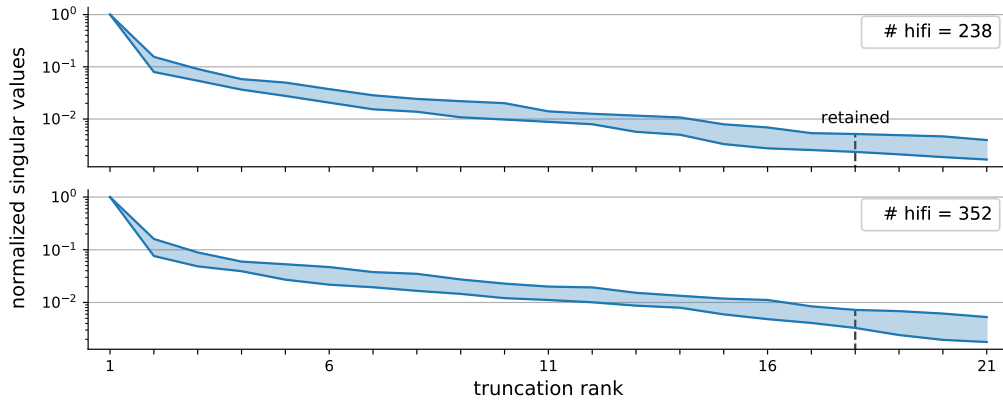


Figure 5.7: Evolution of the minimum and maximum for the normalized singular values decay of the stress tensor components, for MS00 with 17 parameters. Values are reported for the high-fidelity database before and after optimization.

Due to the limited beneficial effect of the last parameterization refinement, we do not refine the model further.

To study the reparameterizations' effects on the [QoIs](#), we can compare the [Pareto fronts](#) (PFs) of the different models as shown in Figure 5.9. The largest improvement in PF quality is attributed to the first reparameterization, as seen in the plots for the number of yielded and buckled elements. The updated PF intercepts the thresholds at a much lower total mass value, indicating that the subsequent single-objective optimizations could find much

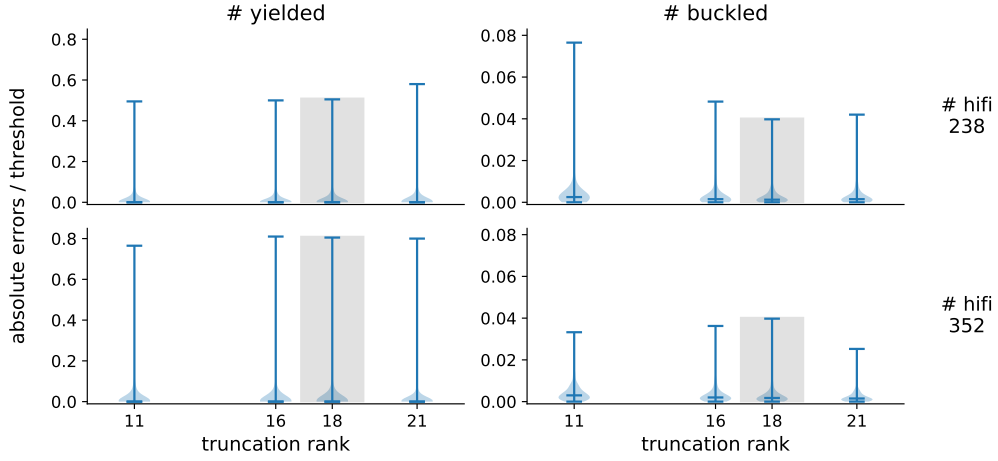


Figure 5.8: Evolution of the 5-CV error distribution for MS00 with 17 parameters. Values are reported for the high-fidelity database before and after optimization. The gray box identifies the chosen truncation rank.

lighter configurations with no additional drawbacks.

The largest improvement in PF quality is attributed to the first reparameterization, as seen in the plots for the number of yielded and buckled elements. The updated PF intercepts the thresholds at a much lower total mass value, indicating that the subsequent single-objective optimizations could find much lighter configurations with no additional drawbacks. The second refinement does not produce a significant change in the PFs and thus the procedure is stopped.

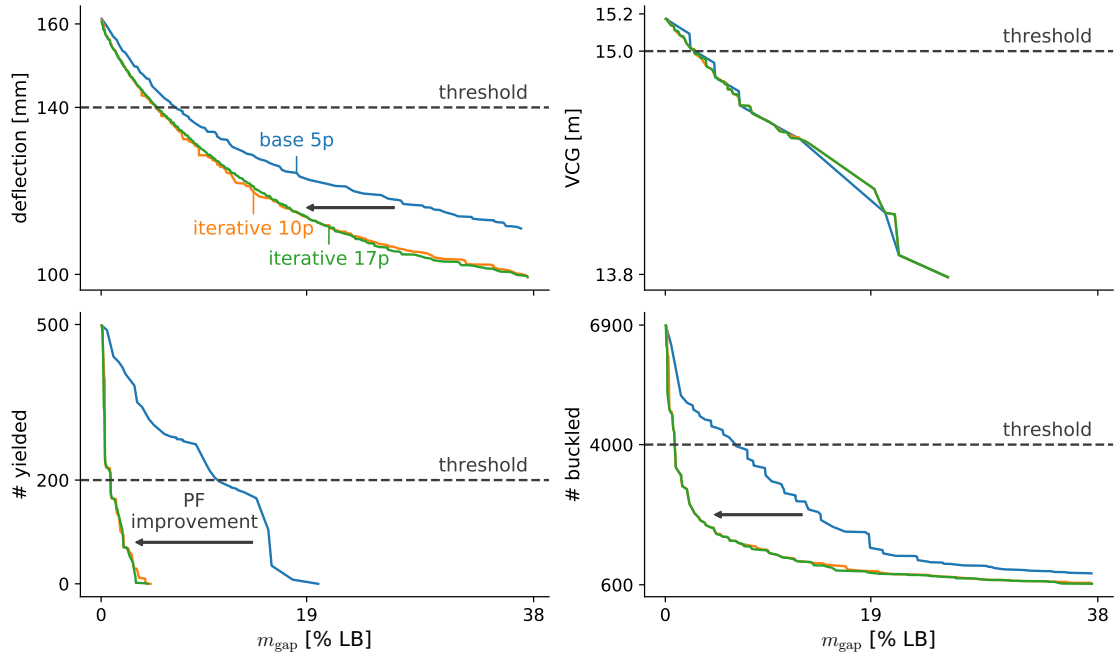


Figure 5.9: Evolution of the surrogate PF for MS00, for different parameterizations.

The evolution of the single-objective incumbent is represented in Figure 5.10, where

the values of earlier samples are not shown due them being too large. We remark that the long stretches of constant incumbent are due to the multi-objective infills.

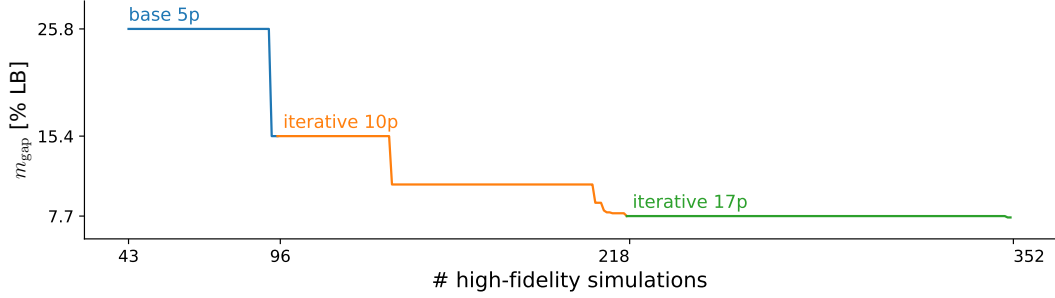


Figure 5.10: Evolution of the optimal high-fidelity m_{gap} of MS00, for the iterative reparameterization approach.

As anticipated, we now evaluate the other reparameterization approaches, "one-shot" and "designer", to obtain two different parameterization with 20 parameters. The results of the optimizations, in terms of high-fidelity m_{gap} , are shown in Figure 5.11 and the detailed data of the optimal configurations are shown in Table 5.1.

Overall, all the largest parameterizations achieve similar m_{gap} , but with different computational effort and results for the structural QoIs. Iterative reparameterization shows a much faster decrease, with the 10 parameters model reaching a comparable objective value to the others, but requiring a much lower number of high-fidelity simulations. The second reparameterization, with 17 parameters, decreases the optimum only by a negligible amount. Both the one-shot and the designer approaches result in a much higher number of high-fidelity evaluations, giving a negative trade-off compared to the 10 parameters model. We remark that all refined models achieve values of n_y much lower than the threshold, while in terms of n_b only the finer models of the iterative and one-shot approaches are able to render the corresponding constraint inactive.

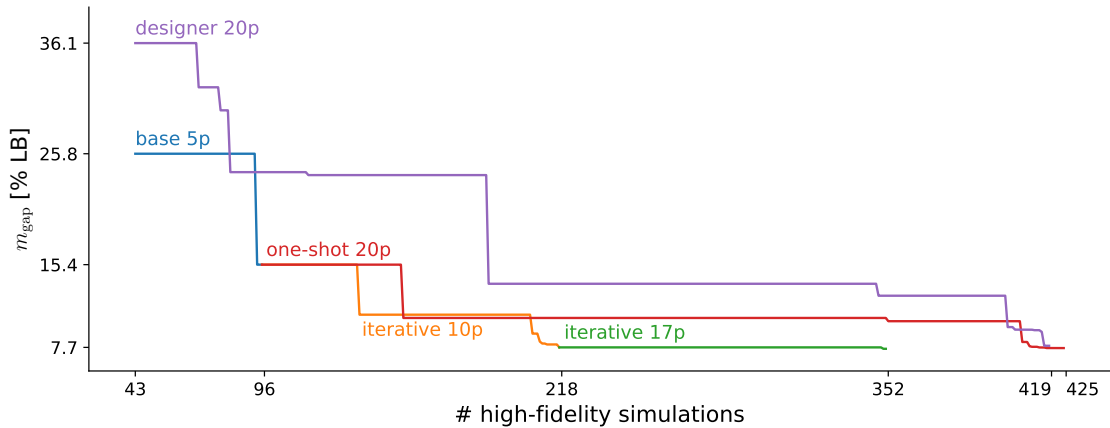


Figure 5.11: Evolution of the optimal high-fidelity m_{gap} of MS00, for different reparameterization approaches.

Table 5.1: Optimization results for different parameterizations of MS00.

Parameterization	m_{gap} [% LB]	n_y	n_b	Deflection [mm]	VCG [m]	# hifi
base, x_{LB}	> 1000	395	6211	150	14.79	-
base 5 p.	15.44	196	3994	138	14.93	96
iterative 10 p.	7.70	33	3984	140	14.99	218
iterative 17 p.	7.57	11	3563	140	14.99	352
one-shot 20 p.	7.64	5	2729	140	14.99	419
designer 20 p.	7.86	22	4004	140	14.99	425

5.3.2 Full ship FS6

We now repeat the study from the previous section on the full ship FS6. The optimization pipeline starts with an initial maximin sampling of size 21, and the full surrogates are built using truncation rank 16 as seen in Chapter 3. Multi- and single-objective optimization are carried out in the same fashion as the previous case, achieving optimum m_{gap} 51.58% and a high-fidelity database of size 114. We remark that this case does not enforce a critical threshold for the vertical deflection.

The iterative reparameterization with $n_{\text{clusters}} = 2$ and $n_{\text{min}} = 1$ leads to 3 model refinements, that progressively increase the number of decision variables to 16, 27 and finally the upper limit of 40. Figure 5.12 shows the σ_x component of the stress tensor for the optimal configurations of the 40 parameters model, and the evolution of the parameters mapping through the different refinements. As the number of decision variables increase, the assignments of the patches match more closely the increase in tension of the deck's elements. The fore and aft, which sustain very low tension, are all assigned to the same decision variable and the optimization consistently selects a low thickness. The sides of the empty spaces require instead more stiffness, and this reflects on the procedure adding progressively smaller parameters in the proximity of the larger one, limiting the use of higher thicknesses to the most critical areas.

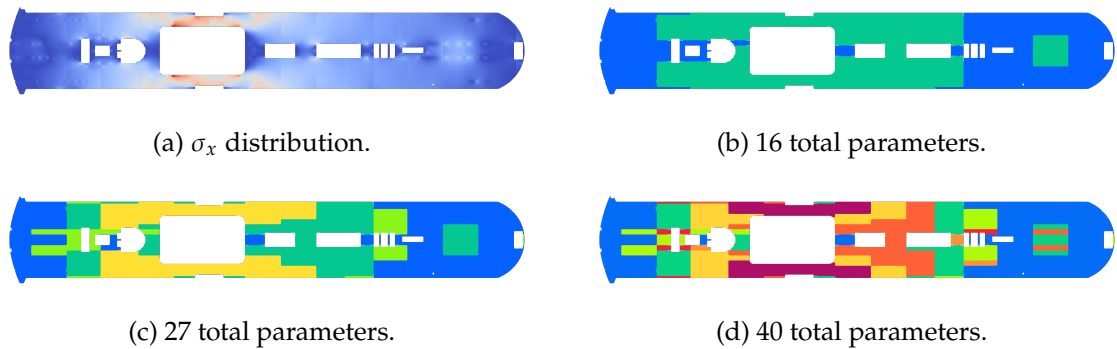


Figure 5.12: Details of the reparameterization of deck 14. Figure 5.12a shows the absolute value of the σ_x from the stress tensor under sagging condition, with red representing the largest values and blue 0. Figures 5.12b, 5.12c and 5.12d show the assignments to different decision variable for the 16, 27 and 40 parameters models, respectively.

To verify the choice of truncation rank, we monitor the singular values decay and the error distribution on n_y and n_b for different sizes of the high-fidelity database of the model with 16 parameters, for different ranks. Figures 5.13 and 5.14 confirm the robustness of the criterion discussed in Chapter 3.

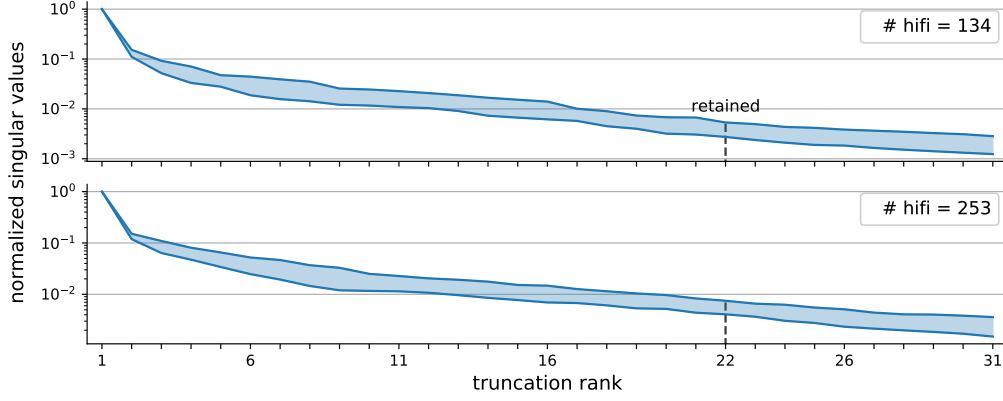


Figure 5.13: Evolution of the minimum and maximum for the normalized singular values decay of the stress tensor components, for FS6 with 16 parameters. Values are reported for the high-fidelity database before and after optimization.

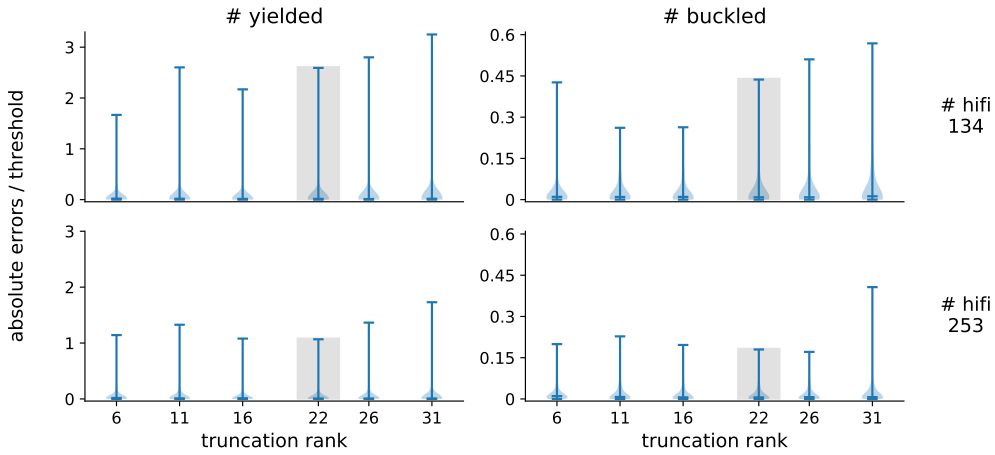


Figure 5.14: Evolution of the 5-CV error distribution for FS6 with 16 parameters. Values are reported for the high-fidelity database before and after optimization. The gray box identifies the chosen truncation rank.

We compare the surrogate PFs of the models obtained with the iterative approach in Figure 5.15. The effectiveness of the reparameterization is most evident on n_y and n_b , where the 16 and 27 parameters models are able to achieve much better trade-offs with the ship's mass. As in the previous case, the first reparameterization has the most pronounced impact on the QoIs.

Figure 5.16 shows the evolution of the optimal high-fidelity m_{gap} during the iterative reparameterizations. Compared with the previous case in Figure 5.10, the first two increases in the number of parameters also determine a large drop of the objective function. This is

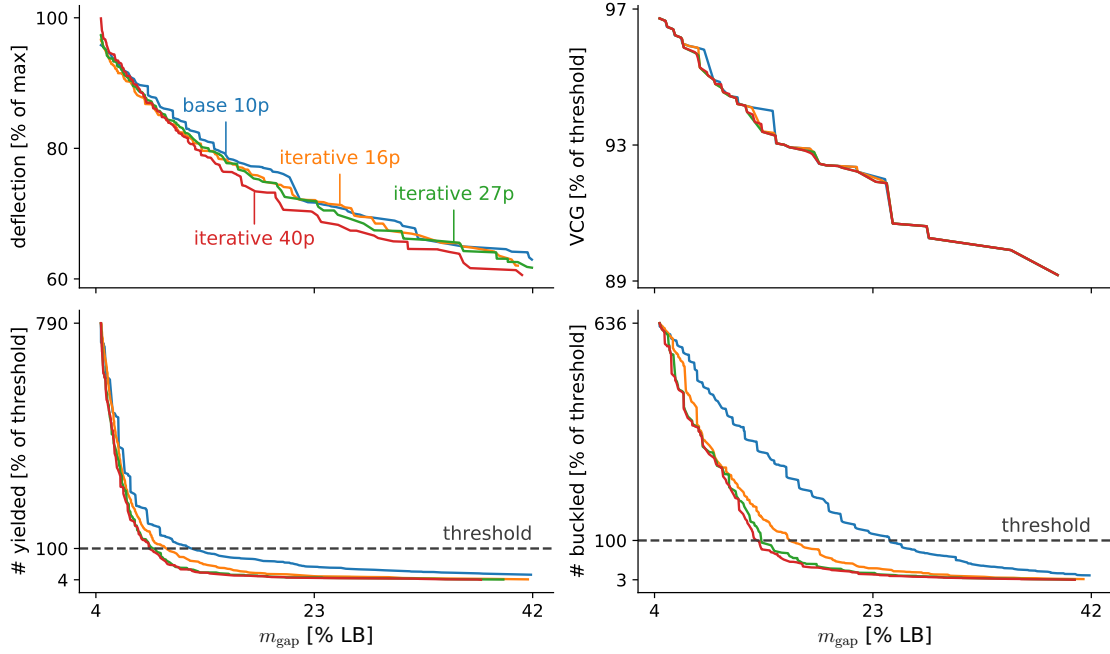


Figure 5.15: Evolution of the surrogate PFs for FS6, for different parameterizations of the iterative approach. The threshold for the VCG constraint is not shown, since the projected PF lies well below its value.

due to the addition of the thickness configuration from Problem 5.1 to the high-fidelity database, that in this setting is not penalized by a constraint on the deflection.

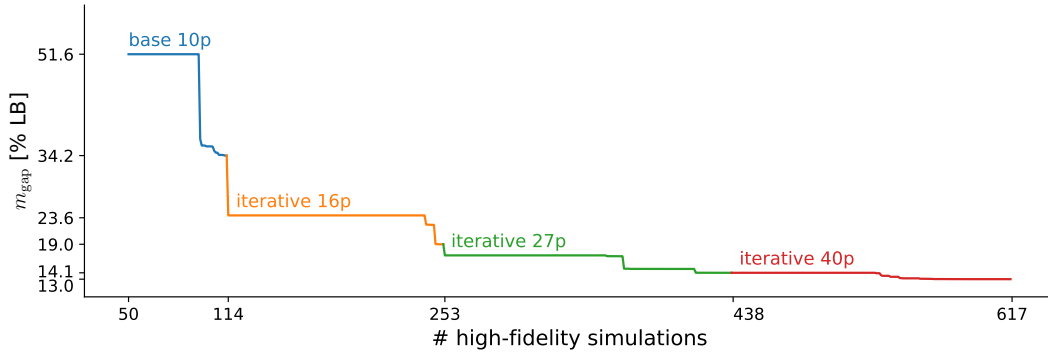


Figure 5.16: Evolution of the optimal high-fidelity m_{gap} of FS6, for the iterative reparameterization approach.

As for the previous case, we also apply the one-shot and designer approaches to obtain two models with 40 parameters. The evolution of optimal high-fidelity m_{gap} for each model is shown in Figure 5.17, and the details of the optima are shown in Table 5.2.

The main difference from the previous case is in the performance of the designer model, which converges earlier and with a much higher m_{gap} compared to every refined model. The one-shot approach shows a large initial decrease due to the solution of Problem 5.1, but the following optimization is unable to beat the 27 parameters model from iterative

approach. The iterative reparameterization results in the best m_{gap} , although the last refinement is only able to decrease the incumbent by about 1%.

Regarding the **QoIs**, we remark that only the refined models from the iterative approach activate the constraint on n_y , while all other optima result in a number of yielded elements close to 50% of the critical threshold. The constraints on n_b and **VCG** are instead active for all optimal configurations.

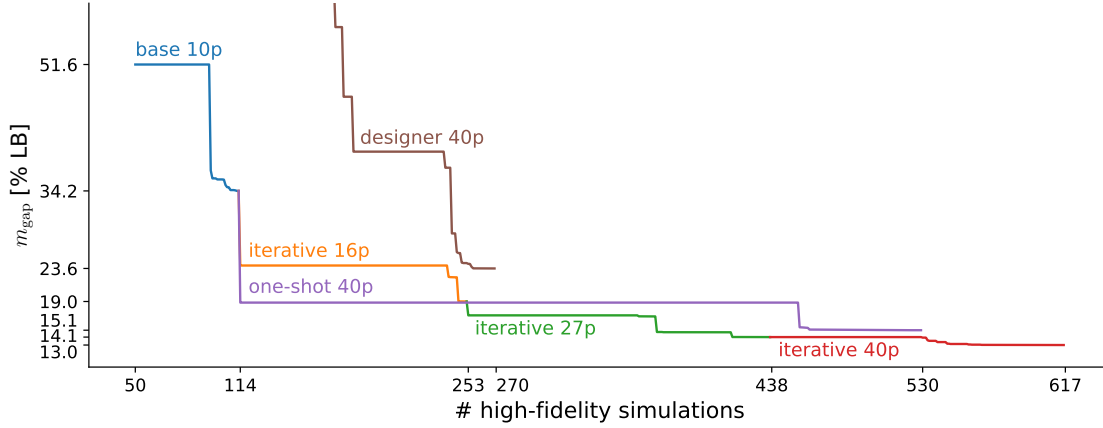


Figure 5.17: Evolution of the optimal high-fidelity m_{gap} of FS6, for different reparameterization approaches.

Table 5.2: Optimization results for different parameterizations of FS6.

Parameterization	m_{gap} [% LB]	n_y [% crit]	n_b [% crit]	VCG [% crit]	# hifi
base, $\mathbf{x}_{\text{LB}}$	> 1000	789.75	636.38	96.26	-
base 10 p.	34.21	46.25	100.38	99.35	114
iterative 16 p.	19.01	56.00	99.88	99.76	253
iterative 27 p.	14.12	98.25	99.29	99.71	438
iterative 40 p.	13.03	98.25	100.01	99.88	617
one-shot 40 p.	15.07	51.50	99.87	99.85	530
designer 40 p.	23.55	55.00	100.15	99.99	270

5.4 Conclusions

This chapter introduced an automated parameterization refinement procedure for the optimization problem described in Chapter 2.

To obtain an effective parameterization, able to finely control the structural behavior of the model while using a low number of decision variables, we propose to refine a coarser parameterization by leveraging the results of its optimization.

The surrogates for the structural [QoIs](#) enable the formulation of auxiliary mass optimization problems defined at the patch level, and the resulting thickness allocations instruct the construction of a new set of decision variables. The hierarchical structure of the refined parameterizations allow to retain the high-fidelity database and to efficiently progress the optimization.

The numerical results on a simplified midship section a full-scale ship show that the reparameterization is able to achieve good results in terms of structural mass, while constraining the structural [QoIs](#). In particular, the approach based on iterative refinement is superior to both the one-shot approach and the baseline produced from the designers. Although showing diminishing returns as the number of parameters increase, the iterative reparameterization approach is able to streamline the initial design phase with minimal intervention.

Chapter 6 will adapt the parameterization procedure to use the knowledge from past optimized ships in place of the full surrogates, enabling the refinement of the optimization problem without the need for the construction of a full high-fidelity database.

TRANSFER LEARNING-BASED REPARAMETERIZATION

This chapter presents a one-shot reparameterization procedure that modifies the approach from Chapter 5 to leverage the results from past optimizations on different ships, without constructing a high-fidelity database or full surrogates for the current ship.

The extraction of useful knowledge from different tasks is the core idea of [transfer learning \(TL\)](#), and constitutes an important resource when data is scarce or expensive to accumulate. At the beginning of the optimization pipeline, the designers have at most access to the simulation results for the default configuration, which motivated the choice of coarse parameterization discussed in Chapter 2 and the refinement procedure developed in Chapter 5. By using the results from past optimizations, the pipeline could begin with a more effective initial formulation without accumulating a large number of high-fidelity simulations during optimization of the coarse problem. A trade-off between the quality of the optimum and the size of the actual high-fidelity database is to be expected, but transfer learning could provide a fast way to produce acceptable configurations in a shorter time frame.

Section 6.1 motivates the need for a faster reparameterization and reviews the basic principles of transfer learning. The approach chosen to achieve the one-shot reparameterization is presented in Section 6.2, and the numerical results on are discussed in Section 6.3. The final remarks are presented in Section 6.4.

6.1 Introduction

In the previous Chapter 5, we constructed a reparameterization procedure that, based on the structural [quantities of interest \(QoIs\)](#) aggregated at the patch level, adapts the assignment of patches and decision variables to obtain a more expressive optimization problem. The pipeline is then able to achieve much lower total ship mass, while respecting the constraints on structural resilience.

However, this approach requires a set of accurate full surrogates, and thus the construction of a high-fidelity database during the execution of the optimization pipeline on the coarse model. In this chapter, we constrain our reparameterization procedure to be available at the moment of problem formulation, when the designers have only access to the computational mesh and the initial coarse parameters definition. In place of the high-fidelity database from a random sampling or from the optimization pipeline, we aim to extract the knowledge necessary to the parameterization refinement from the results of past optimizations.

Transfer learning [137] is a branch of machine learning focused on producing a useful model for a current task, also called target, for which labeled data is scarce or absent, by only training on the results of past task, also called source. The authors of [137] propose a taxonomy to establish the relationship between source and current tasks, based on their specification, the degree of data availability, and the form taken by the knowledge being transferred. We find ourselves in the sequential learning setting, since the source tasks (the ships already optimized) have a fixed amount of data available. Moreover, our case falls under the definition of transductive learning, meaning that the source and current domains are different but related, and the actual tasks are the same. Indeed, the mapping from patches to decision variables in Problem 5.1 is computed at the patch level, based on y_{pt} and b_{pt} which are themselves computed at the patch level. The suggested approach, which is denoted as domain adaptation, can be implemented via instance transfer or feature-representation transfer.

Instance transfer consists in the transformation of elite solutions of a source task into solutions to the current task, and is frequently used in the context of population-based optimization algorithms [88, 133, 187]. However, in our case the solution is represented by the mapping from patches to decision variables, which does not support an obvious encoding that can be transferred through different mesh topologies and sets of variables.

Feature-representation transfer aims to obtain a unified representation of the two domains, and solve the current task leveraging this artificial domain. The ideal approach would then be the construction of a model that predicts the full y_{pt} and b_{pt} fields, given only the definition of the patches, followed by the solution of Problem 5.1. However, the training of such a model would require massive amounts of data due to the necessity of generating multiple fields for each source ship.

A different idea is the construction of a different problem, whose structure is informed by the past tasks, that can produce good heuristic solutions for the current task. In [68], a

population of solutions for the capacitated vehicle routing problem is averaged to extract a distance scaling matrix, that is in turn used in conjunction with a greedy algorithm to construct solutions for a capacitated arc routing problem. A similar approach is used in [145], where a number of solutions to a direct current optimal transmission problem are available for different load conditions. To speed up the solution for unseen load conditions, the network configurations of the most similar past solutions are combined to heuristically generate or refine the constraints for the current task.

Following this philosophy, we focus on finding an alternative to the computation of the terms y_{pt} and b_{pt} in Problem 5.1, leveraging the data available at the patch level.

6.2 Reparameterization

The data accessible for the current and past tasks come from the definition of the computational mesh and the [finite element analysis \(FEA\)](#) results. In the case of the current task, the [FEA](#) results are limited to those computed with the lower bounds of $\mathcal{D}$, as discussed in Section 2.1 with regard to the wave-factoring approach. Table 6.1 summarizes the data available at the patch level, with the exception of the optimal thickness which is unavailable for the current task.

Table 6.1: Features of a single patch.

Features	Description
centroid	Coordinates of the centroid, absolute
centroid (relative)	Coordinates of the centroid, relative to ship's length
normal	Components of the patch's normal vector
area	Total area of the patch
borders	Number of contiguous patches
thickness bounds	Minimum and maximum possible thickness
optimal thickness	Optimal thickness of the patch
simulated thickness	Actual simulated thickness
n_y, n_b	Number of yielded and buckled elements

Instead of modeling a predictor for y_{pt} and b_{pt} as unifying representation of the patches, and solving Problem 5.1 unchanged, we choose to train a predictor for the optimal thickness itself and modify the reparameterization problem accordingly.

Let $\hat{t}_p$ be the optimal thickness predicted for patch p by a [machine learning \(ML\)](#) model

trained on past tasks. Then the TL-based reparameterization problem is given by

$$\begin{aligned}
 & \min \sum_{p \in P_i} \sum_{t \in \mathcal{D}_i} x_{pt} (\mathbf{d}_p t + c(\hat{t}_p - t)_+^2), \\
 & \text{s.t.} \quad \sum_{t \in \mathcal{D}_i} x_{pt} = 1, & \forall p \in P_i, \\
 & \quad x_{pt} \leq u_t, & \forall p \in P_i, \forall t \in \mathcal{D}_i, \\
 & \quad \sum_{p \in P_i} x_{pt} \geq u_t n_{\min}, & \forall t \in \mathcal{D}_i, \\
 & \quad \sum_{t \in \mathcal{D}_i} u_t = n_{\text{clusters}}, \\
 & \quad \sum_{p \in P_i} \sum_{t \in \mathcal{D}_i} x_{pt} (\text{VCG}_p - \text{VCG}_{\text{crit}}) \mathbf{d}_p t \leq (\text{VCG}_{\text{crit}} - \text{VCG}_{\text{res}}) m_{\text{res}}, \\
 & \quad x_{pt} \in \{0, 1\}, & \forall p \in P_i, \forall t \in \mathcal{D}_i, \\
 & \quad u_t \in \{0, 1\}, & \forall t \in \mathcal{D}_i,
 \end{aligned} \tag{6.1}$$

where c is a penalty coefficient with unit of measure t/mm^2 . We remark that $\hat{t}_p$ needs not to be an element of $\mathcal{D}_i$, since the solution assigns feasible thickness values by design. Problem 6.1 can then be used in place of Problem 5.1 in the reparameterization procedure described from the previous section.

Due to the tabular nature of the patch features listed in Table 6.1, we employ gradient boosted tree regressors [75]. In this approach, the predictions are obtained by the additive combination of the outputs from an ensemble of regression trees, constructed in an incremental fashion. Each tree is trained to correct the errors of its predecessors, thereby improving the accuracy of the model.

A regression tree is a hierarchy of nodes: the internal nodes partition the input features into disjoint intervals, associated to children nodes, and the terminal nodes (leaves) output a prediction value. In its simplest form, each internal node evaluates a condition of the form $\mathbf{x}_i < s$, where s is a value to be chosen during training. Depending on whether the condition is satisfied, the evaluation is routed to one of the child nodes until a leaf is reached. The construction of a regressor tree starts from the root node, and uses a greedy approach. Multiple splits are computed, consisting of a feature and a threshold, to divide the outputs in two subsets and associate to each its mean (in the case of mean square error loss) as prediction value. The split that results in the lowest prediction error is integrated as internal node, with the two means assigned to its children as leaf nodes. The procedure continues by expanding the leaf with highest error, until a stopping criterion is met such as reaching a maximum tree depth or insufficient decrease in the prediction error. In tree boosting, the target values of a new tree are computed as the difference between the training targets and the predictions from the trees already constructed. Gradient tree boosting extends tree boosting by considering a generic differentiable loss and leveraging its gradient and Hessians to select the optimal splits. We refer to [75] and [37] for the details. Figure 6.1 shows a simplified representation of a prediction using tree boosting.

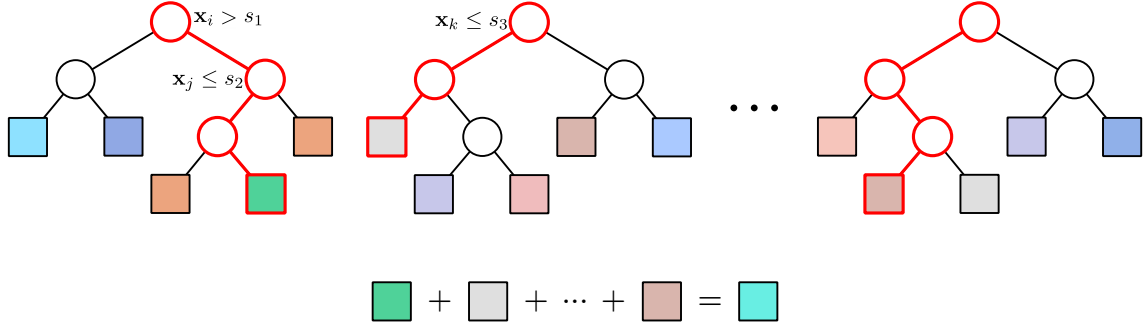


Figure 6.1: Schematic representation of a prediction using tree boosting. Each tree has different splits, which determine different paths to the leaf value, highlighted in red. The final output is the summation of all predictions, with the trees beyond the first acting as correctors to the first element of the ensemble.

A limitation of this approach is that the topological information from the mesh connectivity is discarded. Alternative models such as [graph neural networks \(GNNs\)](#) [163], tailored to learning tasks where data is defined on graphs, could maximize the use of knowledge from past ship designs. However, early experiments with [GNNs](#) have shown significantly higher sensitivity to the choice of network architecture and training hyper-parameters, suggesting that either our database is insufficient to obtain a good model, or that excessive care must be put in the design of the network’s architecture and its training. Since the proposed optimization pipeline is intended for practical integration into an industrial design workflow, the guarantee of robustness and predictability of the training procedure is a strong requirement. Due to this, the pipeline implements a hyper-parameter tuning step of the boosted decision trees via grid search, iterating on a pre-defined set of options and validating the results via [10-fold cross-validation \(10-CV\)](#).

The loss function used to create the regression trees is [mean squared error \(MSE\)](#), while hyper-parameter tuning is carried out by minimization of the number of misclassified patches. We consider a patch to be misclassified if its predicted thickness differs more than 0.25 mm from the true value, which is half the minimum difference between the feasible thicknesses we consider in this study. Although this alternative loss is weaker than [MSE](#), it is an indicator of the number of outliers that Problem 6.1 will be forced to cluster to an incorrect value. In the case of underperforming training, the reparameterization process is more robust in handling bounded absolute errors, where the patches will be clustered together anyway, than facing a group of outliers that could divert the cluster results. [MSE](#) could hide the effect due to the high number of patches in the largest parameterized sections, while the ratio of misclassification correctly represents this behavior.

The optimal thickness predictor needs to be re-trained, and its hyper-parameters re-tuned, every time a new ship design is finalized and added to the knowledge base. The reparameterization procedure itself, however, has total cost comparable to a single high-fidelity evaluation or other phases of the optimization pipeline.

6.3 Numerical results

This section presents the evaluation of the TL-based reparameterization to the midship section variants and the set of full ships described in Chapter 2.

To construct a database of optimized configurations, we followed the steps detailed in Section A. All models were optimized with the routines from Chapter 4, using a fixed number of multi-objective optimizations with [non-dominated sorting genetic algorithm III \(NSGA-III\)](#) and iterating a combination of [principal directions search \(PDS\)](#) and [particle swarm optimization \(PSO\)](#) for mass optimization until no new candidates could be found. From the optimum configuration, the response-based reparameterization from Chapter 5 refined the problem and the optimization was repeated, until the new optimum achieved a very low m_{gap} or showed no significant decrease with respect to the previous parameterization. The optima of the refined ships constitute the database of past tasks for training of the TL-based reparameterization models.

All values are obtained using Fincantieri S.p.A.'s HPC facility, composed of 3 identical nodes equipped with an Intel Xeon Gold 6354 CPU (3.00 GHz), featuring 36 cores and 512 GB of RAM. The gradient boosted tree regressors are implemented via the `xgboost` [37] library, and the hyper-parameter tuning uses `scikit-learn`'s [142] grid search for the [10-CV](#) implementation.

Section 5.3.1 presents the results on the midship section MS00, while Section 5.3.2 repeats the analysis on the full ship FS6.

6.3.1 Midship section variants

The dataset is composed of three types of models:

- the default midship section MS00,
- MS01 through MS06, which add or remove a single component compared to MS00,
- MS07 through MS20, which present more differences with MS00.

This structure enables the evaluation of the reparameterization in two common scenarios. For the first case, we build a test set containing only the variants MS01 through MS06: these models are close to MS00 which will be part of the train set, so this setup will evaluate multiple cases of the current task being very similar to one of the past tasks. In the second case, the test set contains the models MS00 through MS06 which will evaluate the reparameterization when none of the past tasks are close to the current one.

In both cases, we train the `xgboost` model on the train set and use Problem 6.1 with $n_{\text{clusters}} = 2$, $n_{\text{min}} = 1$ and maximum $n_{\text{dim}} = 10$. No threshold on the minimum decrease of the objective function was used when assembling final parameterization. This was chosen from the observation that more refined parameterizations did not achieve much better optima than the models with $n_{\text{dim}} = 10$, as shown in Section A.1. The results will be compared on models that have undergone a single parameterization refinement.

Section 6.3.1.1 presents the results of the first test case, where the current tasks are all related to the past task MS00. Section 6.3.1.2 discusses the results on the second test case, where the differences between past and current tasks are more pronounced.

6.3.1.1 Close past tasks

In this experiment, the `xgboost` model is trained on the optimal configurations of ships MS00 and MS07 through MS20. The test ships MS01 through MS06 are all obtained from a single variation of the default model MS00, which is part of the train data, and should thus prove an easier test for the predictive model. Hyper-parameter tuning of `xgboost` achieves 10-CV MSE of 0.49 and 2.45 on the train and validation folds, respectively. As for the ratios of misclassified patches, the train folds averages 10.9% while the validation folds reaches 25.4%. Table 6.2 compares the results of the TL-based reparameterization with those from the first step of the iterative approach.

Table 6.2: Midship section variants optima - default configuration in training data.

Code	Δ_x	Δ_y	Δ_z	Method	n_{dim}	m_{gap} [% LB]	# hifi	Gain	
								m_{gap}	# hifi
MS01	0	0	-1		5	25.82	60		
				iterative	9	14.18	120		
				xgboost	10	15.28	69	-1.10	51
MS02	0	0	1		5	5.17	60		
				iterative	9	4.08	117		
				xgboost	10	3.54	60	0.54	57
MS03	0	-1	0		5	10.58	60		
				iterative	10	7.97	122		
				xgboost	10	8.64	63	-0.67	59
MS04	0	1	0		5	21.28	60		
				iterative	10	8.92	119		
				xgboost	10	8.94	60	-0.02	59
MS05	-1	0	0		5	8.44	60		
				iterative	10	4.84	122		
				xgboost	10	4.77	60	0.07	62
MS06	1	0	0		5	23.88	60		
				iterative	10	10.84	119		
				xgboost	10	12.30	66	-1.46	53

Variants MS02 and MS05 outperform the iterative reparameterization, and MS04 performs only marginally worse than the reference. The worse results are observed on MS03, MS01 and MS06, but the difference in m_{gap} from the references are relatively small for all case. Overall, the reparameterizations are able to outperform the base 5-parameters models, while employing a lower number of high-fidelity evaluations compared to the references. Even though the reference solutions are penalized, in this metric, by the additional sampling of the reparameterization and the repeated rounds of multi-objective optimization, these steps are necessary to explore the updated parametric domain and are

thus crucial to the design process. Indeed, in a practical workflow the designers would inspect the [Pareto front \(PF\)](#) approximation to establish the threshold for the optimization constraints, a decision that could drastically change between parameterizations. For this reason, the iterative reparameterization procedure can not omit the infill steps of multi-objective optimization, and the capacity of the [TL](#)-based reparameterization to achieve similar performance with less [FEA](#) evaluations constitutes a strong point for its adoption.

6.3.1.2 No close past task

In this experiment, the model was trained on the optimal configurations of ships MS07 through MS20. The test ships differ from all train configurations for at least 2 variations, with the exception of MS08, MS09 and MS14 which can be obtained with a single change from other models in the test set. Hyper-parameter tuning of `xgboost` achieves [10-CV MSE](#) of 0.44 and 2.57 on the train and validation folds, respectively. As for the ratios of misclassified patches, the train folds averages 10.6% while the validation folds reaches 25.5%. Table 6.3 compares the results of the [TL](#)-based reparameterization with those from the first step of the iterative approach.

Table 6.3: Midship section variants optima - default configuration missing from training data.

Code	Δ_x	Δ_y	Δ_z	Method	n_{dim}	m_{gap} [% LB]	# hifi	Gain	
								m_{gap}	# hifi
MS00	0	0	0		5	15.44	60		
				iterative	10	7.50	125		
				xgboost	10	8.32	66	-0.82	59
MS01	0	0	-1		5	25.82	60		
				iterative	9	14.18	120		
				xgboost	10	15.14	72	-0.96	48
MS02	0	0	1		5	5.17	60		
				iterative	9	4.08	117		
				xgboost	10	3.71	63	0.37	54
MS03	0	-1	0		5	10.58	60		
				iterative	10	7.97	122		
				xgboost	10	8.68	66	-0.71	56
MS04	0	1	0		5	21.28	60		
				iterative	10	8.92	119		
				xgboost	10	9.00	60	-0.08	59
MS05	-1	0	0		5	8.44	60		
				iterative	10	4.84	122		
				xgboost	9	4.48	58	0.36	64
MS06	1	0	0		5	23.88	60		
				iterative	10	10.84	119		
				xgboost	10	12.18	63	-1.34	56

The results are in line with the previous experiment, with variants MS02 and MS05

outperforming the references, and MS04 achieving m_{gap} very close to its counterpart. Variants MS01, MS03 and MS00 perform worse than the references, and their negative gains are all close to 1%. The worse performing model is MS06, but the difference from the reference is similar to the value achieved in the previous case, suggesting that the `xgboost` model is intrinsically less robust on this variant. The adoption of TL-based reparameterization thus provides an acceptable trade-off between solution quality and number of high-fidelity evaluations, even when the predictor for the optimal thickness shows high errors during training.

6.3.2 Full ship fleet

Differently from the midship section variants, the full ships have been optimized with different approaches and constraint thresholds, as well using different n_{dim} and n_{clusters} during reparameterization. Moreover, there is not a single default model to which all other ships are related. There are, however, two sets of related ships: FS1 and FS2 are obtained from FS0 via the lengthening of the middle section, and the same relationship applies to FS5 and FS6.

We select four ships as test instances: FS0 which is of medium size, and three ships of small size FS3, FS4 and FS5. Since the dataset contains only 10 instances, for each test case we train the `xgboost` model on the remaining 9 ships. Due to the construction of the dataset, FS0 counts two related ships in the train dataset, and FS5 a single one.

For all test cases, we set the maximum $n_{\text{dim}} = 25$ and compare three approaches to optimization and TL-based reparameterization:

- $n_{\text{clusters}} = 2$: starts with the TL-based reparameterization, performs the usual pipeline optimization, followed by the response-based reparameterization and a final execution of the pipeline,
- $n_{\text{clusters}} = 4$: a one-shot approach limiting the refinement of a single coarse section,
- $n_{\text{clusters}} = n_{\text{dim}}$ (unlimited): one-shot reparameterization with no limit on the split of a single section.

This setup investigates whether a parameterization of medium complexity obtained with the TL-based procedure leads to an acceptable trade-off in terms of final optimum and number of simulations, compared to the more accurate response-based approach.

6.3.2.1 Test ship FS0

The FS0 ship is the shorter sister of FS1 and FS2, sharing with them the cross section as well as the aft and fore geometries. The optimization of the initial model with 9 parameterized sections leads to $m_{\text{gap}} = 38.93\%$, but the following reparameterization with $n_{\text{clusters}} = 4$ manages to reach $m_{\text{gap}} = 23.20\%$ by introducing 9 new decision variables and executing a total of 163 high-fidelity simulations. The second reparameterization of

the iterative approach adds 12 decision variables, but has only a marginal gain in terms of $m_{\text{gap}} = 22.13\%$. However, the number of high-fidelity simulations required by this configuration increases to 265. Hyper-parameter tuning of xgboost achieves 10-CV MSE of 0.001 and 2.836 on the train and validation folds, respectively. As for the ratios of misclassified patches, the train folds averages 0.03% while the validation folds reaches 43%, confirming overfitting. The results are summarized in Table 6.4 and represented in Figure 6.2.

Table 6.4: FS0 - comparison of optima.

Method	n_{clusters}	n_{dim}	m_{gap} [% LB]	# hifi
base		9	38.93	67
iterative	4	18	23.20	163
iterative	4	30	22.13	265
xgboost	2	14	29.19	79
xgboost + iterative	2	18	25.58	160
xgboost	4	25	27.03	96
xgboost	unlimited	25	26.09	105

Reflecting the poor performance of the predictor, none of the approaches tested was able to achieve optima comparable to the references, with the hybrid approach obtaining the best $m_{\text{gap}} = 25.58\%$ only after response-based procedure. This is comparable to the first reparameterization of the reference, which also required a similar number of high-fidelity evaluations. The unlimited approach is a close second at $m_{\text{gap}} = 26.09\%$, and the middle approach achieves $m_{\text{gap}} = 27.03\%$. However, the last two approaches require 60 less high-fidelity evaluations compared to the hybrid and reference methods. Overall, the unlimited approach shows the best trade-off.

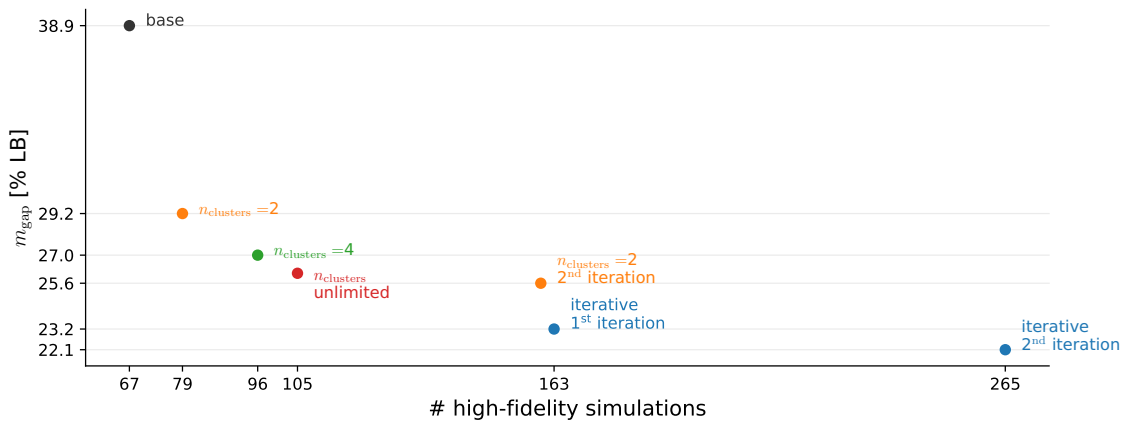


Figure 6.2: Comparison of the optima reached for FS0 with different parameterizations.

6.3.2.2 Test ship FS3

Ship FS3 is part of the group of smaller ships, for which only two iterations of pipeline were performed. Hyper-parameter tuning of xgboost achieves 10-CV MSE of 0.0027 and 2.72

on the train and validation folds, respectively. As for the ratios of misclassified patches, the train folds averaged 0.8% while the validation folds achieved 42.3%, again showing overfitting.

The results are summarized in Table 6.5 and represented in Figure 6.3.

Table 6.5: FS3 - comparison of optima.

Method	n_{clusters}	n_{dim}	m_{gap} [% LB]	# hifi
base		9	12.06	67
iterative	4	15	6.35	139
xgboost	2	12	7.73	64
xgboost + iterative	2	14	7.04	137
xgboost	4	21	7.16	91
xgboost	unlimited	25	6.86	99

The relative results between the tested approaches is analogous to the previous case: all reparameterized models performed better than the initial model, but none of the TL-based optima was able to reach the m_{gap} of the reference. The unlimited run was able to surpass the other two TL-based experiments, achieving $m_{\text{gap}} = 6.86\%$ with 99 high-fidelity evaluations, thus placing close to the optimum of the reference with $m_{\text{gap}} = 6.35\%$, but which required 139 simulations. Interestingly, the TL-based model with $n_{\text{clusters}} = 2$ required 64 high-fidelity samples to converge to an optimum much better than the initial model, for a lower number of simulations. In this case too, the unlimited approach appears to offer the best tradeoff between quality and computational effort.

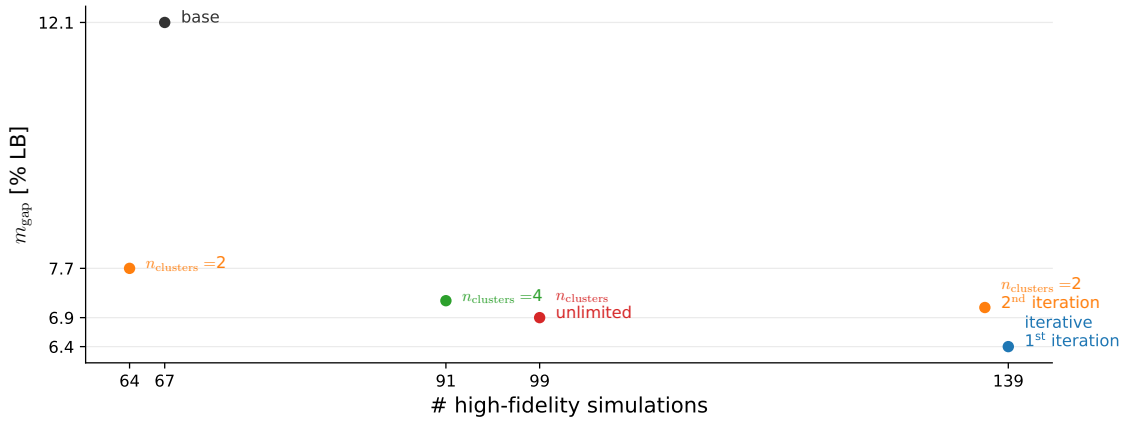


Figure 6.3: Comparison of the optima reached for FS3 with different parameterizations.

6.3.2.3 Test ship FS4

Ship FS4 is another smaller ship, for which only two iterations of pipeline were performed. Hyper-parameter tuning of xgboost achieves 10-CV MSE of 0.09 and 2.38 on the train and validation folds, respectively. As for the ratios of misclassified patches, the train folds

averaged 16.1% while the validation folds achieved 43.5%. The results are summarized in Table 6.6 and represented in Figure 6.4.

Table 6.6: FS4 - comparison of optima.

Method	n_{clusters}	n_{dim}	m_{gap} [% LB]	# hifi
base		8	13.69	56
iterative	4	14	8.93	126
xgboost	2	13	9.58	69
xgboost + iterative	2	16	9.29	140
xgboost	4	22	9.00	87
xgboost	unlimited	25	8.69	93

Although the relative differences between the TL-based models are similar to the results of the other test cases, this is the only case in which a TL-based model was able to surpass the reference: the unlimited approach achieves $m_{\text{gap}} = 8.69\%$ with 93 high-fidelity evaluations, compared to $m_{\text{gap}} = 8.93\%$ with 126 simulations. It must be noted, though, that all reparameterized models were quite close in terms of m_{gap} .

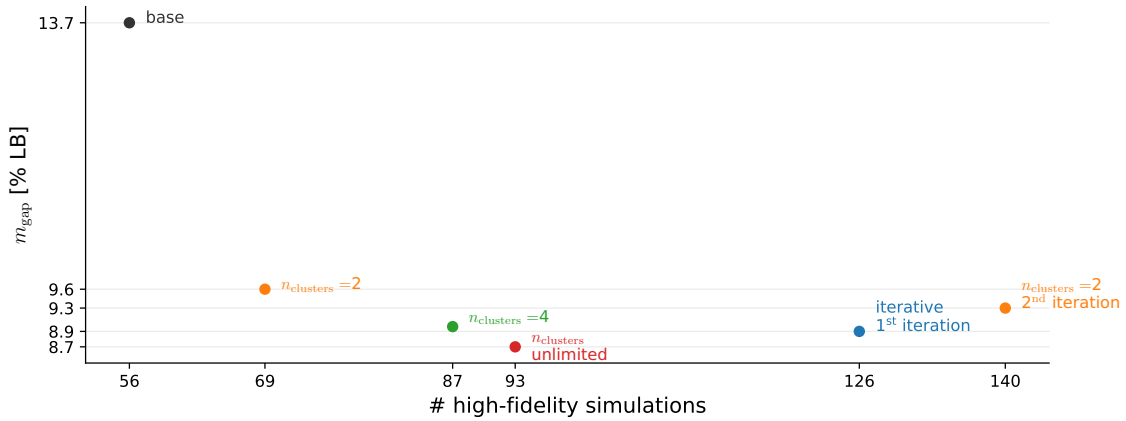


Figure 6.4: Comparison of the optima reached for FS4 with different parameterizations.

6.3.2.4 Test ship FS5

The FS5 ship is the shorter version of FS6. Hyper-parameter tuning of xgboost achieves 10-CV MSE of 1.15 and 3.20 on the train and validation folds, respectively. As for the ratios of misclassified patches, the train folds averaged 23.1% while the validation folds achieved 43.6%. The results are summarized in Table 6.7 and represented in Figure 6.5.

In this experiment, none of the TL-based model achieve a better optimum than the reference $m_{\text{gap}} = 10.76\%$. The unlimited model and the first TL-based parameterization with $n_{\text{clusters}} = 4$ results in very close m_{gap} s of 12.18 and 12.20 respectively, both with 96 high-fidelity evaluations. The best TL-based model is obtained after the response-based reparameterization, but its $m_{\text{gap}} = 11.53\%$ requires an even higher number of high-fidelity evaluations than the reference.

Table 6.7: FS5 - comparison of optima.

Method	n_{clusters}	n_{dim}	m_{gap} [% LB]	# hifi
base		8	16.78	59
iterative	4	17	10.76	135
xgboost	2	14	13.03	71
xgboost + iterative	2	18	11.53	149
xgboost	4	25	12.20	96
xgboost	unlimited	25	12.18	96

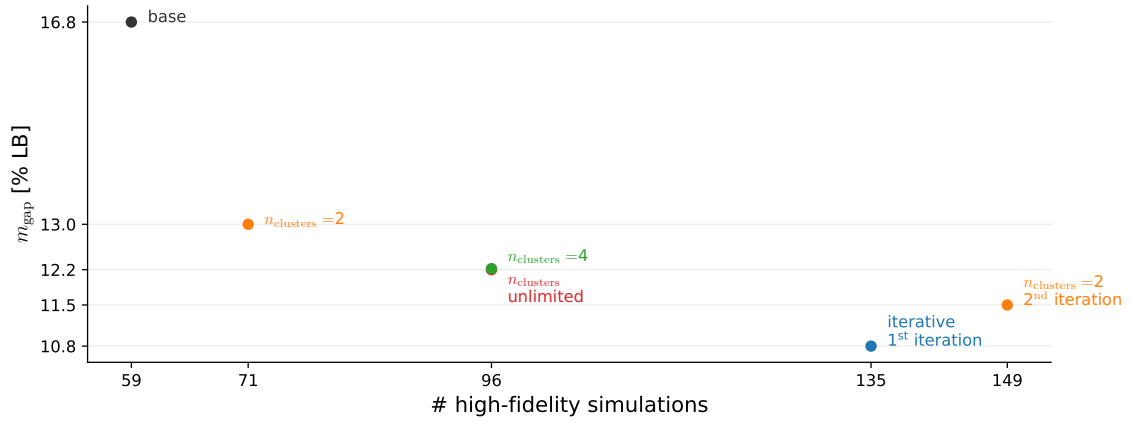


Figure 6.5: Comparison of the optima reached for FS5 with different parameterizations.

Overall, the TL-based reparameterization with $n_{\text{clusters}} = n_{\text{dim}}$ provides an effective trade-off between solution quality and total number of FEA simulations on smaller ships, but suffers a decrease in quality on more complex models such as FS0.

6.4 Conclusions

This chapter presented a one-shot reparameterization routine, based on [TL](#), that is able to refine an optimization problem with a single [FEA](#) thanks to the effective use of a database of past optimized ships.

The Problem 5.1 was analyzed through the lens of transductive learning, leading to a domain adaptation of its formulation. The result is Problem 6.1, where the estimation of optimal thicknesses is used as unified representation of the patches' structural behavior in place of the [QoIs](#) returned by the full surrogates. The `xgboost` predictor for the optimal thickness is constructed in the offline phase, on the data from past optimized ships, and generalizes to new geometries without requiring the construction of a high-fidelity database.

The numerical experiments on a set of simplified midship section variants, which compared models refined via [TL](#)-based reparameterization and the response-based approach, show promising trade-offs between the quality of the optima and the number of [FEA](#) performed during optimization. The evaluation on two settings, whether the database of past ships contains a model close to the one being optimized or not, showed consistent results.

On the full ship dataset, multiple approaches to one-shot reparameterizations were compared to determine the optimal criteria for the refinement. Although the predictor for the optimal thickness showed signs of overfit, the results on multiple instances of smaller ships show that the refinements that only limit the number of total parameters produce optima close to the reference solutions, but with a lower number of [FEA](#) simulations. A test on a more complex model of medium size showed a trade-off of lower quality.

Nevertheless, the approach consistently reduced the number of required high-fidelity evaluations, making the optimization workflow more suitable for preliminary design phases characterized by strict computational and time constraints.

Following the successful application of [TL](#), Chapter 7 will adopt a [TL](#) approach to initialize the populations for the optimization metaheuristics with elite configurations from past ships, accelerating the exploration of high-quality areas of the domain.

POPULATION TRANSFER

In Chapter 6, an approach based on [transfer learning \(TL\)](#) was able to extract useful knowledge from a database of past ships and produce refined parameterizations of new and unseen models, without the construction of a high-fidelity database and full surrogates.

Motivated by the positive results, we now aim to integrate [TL](#) in the optimization phase of the pipeline. We will construct a procedure for the transfer of high-quality configurations from past optimizations to initialize the populations of our multi- and single-objective metaheuristics. Our objective is to enhance the exploratory capabilities of the optimization routines by quickly evaluating large amounts of promising configurations, instead of relying only on the rules of the population update.

Section 7.1 reviews the existing literature and discusses several possible approaches to the problem. Our proposed transfer procedure is presented in Section 7.2. The numerical results are discussed in Section 7.3, where we evaluate the transfer on multi- and single-objective optimization for a group of midship section variants and two complex full ship instances. Section 7.4 draws the final remarks.

7.1 Introduction

In this chapter, we aim to enhance the multi- and single-objective optimization procedures of the surrogate-based pipeline. The results from Chapter 6 suggest that the tasks corresponding to different ships are sufficiently similar to extract useful knowledge to be used on new unseen models.

In the context of optimization, the integration of general machine learning has seen much diversification in recent years. Once relegated to the implementation of surrogates for the evaluation of objectives and constraints, the rise in computational power and availability of data has extended its application to many different aspects of optimization [19, 34, 28]. More specifically to metaheuristic algorithms, [186, 187, 104] expand and specialize the aforementioned concepts. Common trends are the tuning of hyper-parameters of complex black-box solvers [121, 34], model transfer [119] and the construction of optimization policies [152, 198] via reinforcement learning.

Focusing on population-based metaheuristics such as [genetic algorithm \(GA\)](#) and [particle swarm optimization \(PSO\)](#), the issue of individuals initialization complements the trends described above [87, 186, 187, 104], often under the names of "population seeding" or "case-based initialization". A detailed discussion of the use of case-based initialization of a [GA](#) is in [118], where the authors inject elite configurations from past tasks defined on the same domain and evaluate the effect of task similarity on the optimization.

We discard approaches based on hyper-parameter tuning or policy-based optimization of [non-dominated sorting genetic algorithm III \(NSGA-III\)](#) and [PSO](#) due to the robustness shown across multiple test cases, and choose not to study model transfer since our full surrogates are cheap enough to construct. Instead, we focus on the transfer of populations to leverage the large amount of records from the [NSGA-III](#) and [PSO](#) logs. The major issue to overcome is the heterogeneous nature of the domains, which differ both in number of parameterized sections and the underlying computational mesh.

Little details are present in the literature on how to approach the heterogeneous domains setting, since it often requires problem-specific modeling. The transfer between the populations of different optimization tasks is the key element of evolutionary multi-task optimization [88, 133, 210]. In multi-task optimization, multiple populations are evolved to each optimize a different task, in the context of a single [GA](#). The exchange of information is carried out in the crossover operator, and a variety of approaches have been developed to process individuals from heterogeneous domains. The simplest approach rescales the respective domains and embeds the individuals in a single space, relying on a correspondence between the dimensions of the original domains [88]. In [211], the author propose to construct an affine transformation between domains by leveraging the empirical covariance matrices of the respective [Pareto fronts \(PFs\)](#). Another approach to transfer of [PF](#) from past tasks is through denoising autoencoders [67, 69], constructing a new autoencoder every time the transfer takes place. Separate from optimization, techniques for heterogeneous [TL](#) have been discussed in [200, 52], although the focus was on classification

problems.

However, the previous approaches are all based on the evaluation of the objective functions, or classification labels, rather than leveraging a-priori similarity between tasks, limiting their application to the context of co-optimization or to the online phase of optimization. In [202], the authors propose a to enhance a [tree-structured Parzen estimator \(TPE\)](#) for [Bayesian optimization \(BO\)](#) by using a measure of task similarity to leverage past optimizations. By assuming that the objective function of similar tasks correlate and by integrating a suitable kernel in the density estimation of the [TPE](#), the proposed approach was able to overcome many baselines on multiple hyper-parameter tuning tasks in a multi-objective setting.

For our approach to population transfer, we will recast the problem of mapping configurations of thicknesses between different domains in terms of correlation between parameterized sections, using the tools of [Gaussian process regression \(GPR\)](#). We will leverage a similarity metric defined on the problem definition, using the computational mesh, the definitions of the parameterization and the domain, and the results of the simulation of the minimum thicknesses. The final population transfer procedure will not require the evaluation of configurations by the target task, enabling the transfer at any point during the optimization pipeline.

7.2 Transfer procedure

We aim to construct a transformation that, given a configuration of parameter values $\mathbf{x}^A \in \mathcal{D}^A$ for the ship A , builds a configuration $\mathbf{x}^B \in \mathcal{D}^B$ for the ship B . Since our purpose is to leverage the exploration of $\mathcal{D}^A$ from previous optimizations, we desire to transfer the increase of thickness above the lower admissible values, as it encapsulates the concept of a "direction" in domain space. Let us consider a pair of elements in the two configurations: $\mathbf{x}_i^B$ and $\mathbf{x}_j^A$. When the associated parameterized section are related, for example they both correspond to the bottom decks of the respective ships, if $\mathbf{x}_j^A$ is much higher than its lower bound so should be $\mathbf{x}_i^B$. On the opposite, when the controlled sections are unrelated, the value of $\mathbf{x}_j^A$ should have no influence on $\mathbf{x}_i^B$ during the transfer.

The same relationship applies to sections in the same ship: parameterized sections that are similar should increase their thickness together. We want to build a transfer function that captures correlation between different configuration elements, so that the transfer of elite configurations between similar ships results in configurations close to the optimum of the receiving ship.

First, let us consider $\mathbf{x}^A$ as being the evaluation of an implicit function χ on the metadata associated to the parameterized sections of ship A . If we could evaluate χ on the metadata of the sections of ship B , we would obtain $\mathbf{x}^B$ and thus, a transfer of the original configuration. But since we only know χ on the metadata of sections in A , its evaluation requires leveraging the structure of sections' metadata across different ships and the intuitions presented at the beginning of this section.

We propose to construct a [GPR](#) to evaluate χ on the metadata of sections in B , obtaining $\mathbf{x}^B$ as the mean of a set of random variables conditioned on the observed $\mathbf{x}^A$. To specify the kernel function, we leverage the geometrical similarity of the mesh elements associated to each section to obtain a dissimilarity metric in place of the usual euclidean distance used in most stationary kernels.

Let $\mathcal{U}$ be the set of metadata of all possible parameterized sections, of all possible ships. An element $u \in \mathcal{U}$ is composed of all the information available at the beginning of the optimization pipeline:

- the mesh elements assigned to the element's decision variable, with their coordinates $X^u \subset \mathbb{R}^3$,
- the admissible thickness values,
- the stress tensor values corresponding to the lower bound, under hogging and sagging conditions $S^u \subset \mathbb{R}^{12}$.

Thus, a metadata record for the parameterized section s of a ship is the triple

$$\mathcal{U}^s = \left(S \in \mathbb{R}^{|E^s| \times 3}, \mathcal{D}^s, Q \in \mathbb{R}^{|E^s| \times 12} \right), \quad (7.1)$$

where E^s is the set of elements assigned to section s and $\mathcal{D}^s \subset \mathcal{R}_+$ is a discrete set of admissible thickness values. The aforementioned thickness function is $\chi : \times_{s=1}^n \mathcal{U}^s \mapsto \mathbb{R}_+^n$, where n is the number of parameterized sections being evaluated. Thus we have

$$\chi(u^A) = \mathbf{x}^A, \quad \chi(u^B) = \mathbf{x}^B. \quad (7.2)$$

We model the evaluation of χ as the observation of a random variable, for which we choose prior mean μ equal to the minimum admissible thickness for the given parameterized sections

$$\mu(\mathbf{x}^A) = \mathbf{x}_{LB}^A, \quad \mu(\mathbf{x}^B) = \mathbf{x}_{LB}^B. \quad (7.3)$$

instead of adopting the usual null prior.

Additionally, we assume to have a covariance kernel function $k : \mathcal{U} \times \mathcal{U} \mapsto \mathbb{R}$ that is used to generate positive definite covariance matrices between the associated configuration elements. We impose that the kernel function k depends only on the metadata and not on the actual χ . The covariance matrix between $\mathbf{x}^B$ and $\mathbf{x}^A$ is then composed of elements such that in position i, j we find:

$$k(u_i^B, u_j^A) = \mathbf{K}(u_i^B, u_j^A)_{ij} = \mathbb{Cov} \left[\mathbf{x}_i^B, \mathbf{x}_j^A \right], \quad (7.4)$$

where

$$\mathbb{Cov} \left[\mathbf{x}_i^B, \mathbf{x}_j^A \right] = \mathbb{E} \left[(\mathbf{x}_i^B - \mu(\mathbf{x}_i^B))(\mathbf{x}_j^A - \mu(\mathbf{x}_j^A)) \right]. \quad (7.5)$$

Then, applying the properties of [GPRs](#), we have that

$$\chi(u^B) = \mathbf{x}^B \sim \mathcal{N}(\mu(\mathbf{x}^B | \mathbf{x}^A), \Sigma(\mathbf{x}^B | \mathbf{x}^A)), \quad (7.6)$$

where Σ is a covariance matrix. By expanding, we find

$$\mathbf{x}^B \approx \mu(\mathbf{x}^B|\mathbf{x}^A) = \mathbf{x}_{LB}^B + \mathbf{K}(u^B, u^A)\mathbf{K}(u^A, u^A)^{-1}(\mathbf{x}^A - \mathbf{x}_{LB}^A), \quad (7.7)$$

$$\Sigma(\mathbf{x}^B|\mathbf{x}^A) = \mathbf{K}(u^B, u^B) - \mathbf{K}(u^B, u^A)\mathbf{K}(u^A, u^A)^{-1}\mathbf{K}(u^A, u^B). \quad (7.8)$$

In order to choose a kernel function k , we can first assume to have a proper measure of dissimilarity $d : \mathcal{U} \times \mathcal{U} \mapsto \mathbb{R}_+$ that can be used in a classic kernel such as the squared exponential from Equation (3.12) or the Matérn from Equation (3.13) in place of the Euclidean distance with [automatic relevance determination \(ARD\)](#)

$$d(u, u') = \sqrt{\sum_i \frac{(u_i - u'_i)^2}{l_i^2}}. \quad (7.9)$$

We rewrite the squared exponential as

$$k_{SE}(u, u') = \sigma^2 \exp\left(-\frac{d(u, u')^2}{2l^2}\right), \quad (7.10)$$

and the Matérn kernel as

$$k_{Mv}(u, u') = \sigma^2 \frac{2^{1-v}}{\Gamma(v)} \left(\sqrt{2v} \frac{d(u, u')}{l}\right)^v K_v\left(\sqrt{2v} \frac{d(u, u')}{l}\right), \quad (7.11)$$

where l is now a scalar length scale associated to the dissimilarity d . To proceed, we need to specify a distance function d in metadata space, as well as the hyper-parameters required by the chosen kernel.

Our metadata space $\mathcal{U}$ does not allow for the use of the Euclidean distance, the sets of coordinates in our domain have different cardinality and do not translate to simple vectors. To define a suitable dissimilarity metric, we can leverage the coordinates of the mesh elements and the available [quantities of interest \(QoIs\)](#), namely, the evaluation of the stress tensors for $\mathbf{x}_{LB}$. These datasets have different cardinality, and require techniques tailored to discrete sets.

In [78], Bayesian optimization is applied to a sensor placement problem. The configurations are collections of sensor coordinates, where the covariance kernel function is constructed with the earth mover distance from the optimal transport literature. However, this approach is practical only for limited numbers of points, since it requires the solution of an optimal transport problem that does not scale well to the tens of thousands of elements in our parameterized sections.

The same scalability issue affects approaches based on the pairwise covariance computation between elements of sets, used in other approaches to [BO](#) on sets [80, 107, 108].

For our case, we aim to identify an inexpensive, but descriptive, dissimilarity metric between point clouds, and post-process it for use in the covariance computation. To this end, we choose the bi-directional Chamfer distance. First, let $S \subset \mathbb{R}^3$ be a point cloud and

$$c(x, S) = \operatorname{argmin}_{s \in S} \|s - x\|_2 \quad (7.12)$$

be the function that, given a reference point $x \in \mathbb{R}^3$, returns the closest point in S . Then, the bi-directional Chamfer distance between $S^A, S^B \subset \mathbb{R}^3$ is defined as

$$d_{\text{CH}}(S^A, S^B) = \frac{1}{|S^A|} \sum_{a \in S^A} \|a - c(a, S^B)\|_2 + \frac{1}{|S^B|} \sum_{b \in S^B} \|b - c(b, S^A)\|_2. \quad (7.13)$$

To better capture the functional similarity between parameterized sections, we rescale the coordinates of S^A and S^B so that the axis-aligned bounding box is the $[0, 1]^3$ cube. In this way, we facilitate the overlap of sections that share the same relative position (e.g., top decks, shells, etc.) in the two meshes. The computation of the d_{CH} can be implemented efficiently by leveraging space partitioning trees for the nearest neighbor retrieval in $c(x, S)$, leveraging logarithmic search complexity. Moreover, the trees need to be computed only once for each section, and can be reused in the case of multiple transfers.

In Equation (7.13), the Euclidean distance is used as both relevance metric, for the selection of the nearest neighbor, and dissimilarity metric between the so obtained pair of points. We can extend the Chamfer distance formulation to allow for a different dissimilarity metric that, for example, compares the elements' stress tensor components in the simulation of minimum thickness. Let $r : S \mapsto \mathcal{Q}$ be the mapping between the mesh element's coordinates and its [QoIs](#) from domain $\mathcal{Q}$. Then the generalized bi-directional Chamfer distance between two point clouds, augmented by the respective [QoIs](#) functions, is given by

$$d_{\text{CH}}(S^A, r^A, S^B, r^B) = \frac{1}{|S^A|} \sum_{a \in S^A} \|r^A(a) - r^B(c(a, S^B))\|_2 + \frac{1}{|S^B|} \sum_{b \in S^B} \|r^B(b) - r^A(c(b, S^A))\|_2. \quad (7.14)$$

If the [QoIs](#) functions r^A, r^B are the identity, we obtain Equation (7.13). In our setting, we wish to consider as [QoIs](#) the structural response described in Equation (7.1). Compared to the dissimilarity from the euclidean distance of elements' coordinates, the use of structural responses is much more sensitive to the orientation of the elements: most of the stress tensors will have only three non-zero entries, whose positions reflect the plane that the element resides in. Thus, elements that are orthogonal will evaluate to a very high euclidean distance in terms of stress tensors, achieving the goal of de-correlating parameterized sections that are functionally different, while the original formulation of the Chamfer distance does not reveal this discrepancy.

However, the use of either form of d_{CH} in place of the Euclidean distance in Equation (7.10) or (7.11) can result in a covariance matrix that is not positive semi-definite. Indeed, a valid dissimilarity metric for such kernels is required to be conditionally negative definite [164]. As a workaround, we solve a [multi-dimensional scaling \(MDS\)](#) problem [29] to fit a set of coordinates as embedding for the parameterized sections, by approximating the distance matrix computed using d_{CH} . The Euclidean distance computed with the embedding coordinates will then be used to generate positive semi-definite covariance matrices for the [GPR](#) inference.

Let $n_u = n_{\text{dim}}^A + n_{\text{dim}}^B$ be the total number of parameterized sections from ship A and B . Then let $d_{\text{CH}} \in \mathbb{R}_+^{n_u \times n_u}$ be the distance matrix computed using d_{CH} on all pairs of

parameterized sections' metadata. Through [MDS](#), we aim to find the set of coordinates $X \in \mathbb{R}^{n_u \times n_u}$ such that the stress

$$\sqrt{\sum_{i \neq j=1}^{n_u} (d_{CHij} - \|X_i - X_j\|_2)^2} \quad (7.15)$$

is minimized. We choose n_u as embedding dimension to saturate the fitting capabilities of [MDS](#). In practice, even with this large embedding space, the stress is not able to reach 0 and the relative error for the distance matrix is contained between 5 and 20% in Frobenius norm. Convergence of the error is empirically observed to manifest already at embeddings of dimension 5-6, independently of n_u .

As choices for the kernels' hyper-parameters, we set the length-scale l equal to the standard deviation of the distances between embedded coordinates X . As for the scaling coefficient σ , it is set to 1 due to its elision in the computation of the transfer matrix in Equation (7.7) and its irrelevant scaling effect in Equation (7.8).

At this point, we compute the transfer matrix $M(u^B, u^A) = \mathbf{K}(u^B, u^A)\mathbf{K}(u^A, u^A)^{-1}$ where $M \in \mathbb{R}^{n_{\text{dim}}^B \times n_{\text{dim}}^A}$. By design, M is constructed only from the metadata of the source and destination parameterized sections, with no dependency on the actual χ : this implies that the same matrix is repeatedly used by all the configurations of the population from the source ship. Then, the transfer of multiple configurations $\mathbf{x}^A$ to the corresponding $\mathbf{X}^B$ using Equation (7.7) involves only a matrix-matrix product and vector sums. The repair Problem 4.18 is used to enforce both $\mathbf{x}^B \in \mathcal{D}^B$ and the specific constraints on the center of gravity of ship B. Finally, we remove all duplicates from the transferred, repaired population.

An issue with this transfer approach is that the rows of M might sum to a total higher than 1, in which case the final thickness in the receiving ship could be higher than the highest thickness in the source configuration. Rather than letting multiple source configurations be repaired to reach the upper bound of multiple members, we scale the sum of row elements to be 1 to increase the population's diversity.

To showcase the steps of the process, we simulate the construction of the transfer matrix for a pair of full-scale ships from Chapter 2. We utilize FS4 as the source ship, and FS5 as the target. Both ships use their base parameterization, giving 9 parameterized section for the source FS4 and 8 sections for the target FS5.

Figure 7.1 illustrates the Chamfer distance evaluated between the source and target sections, constructed on the scaled coordinates of the mesh elements.

The good approximation of the Chamfer distance via the embedding obtained from [MDS](#) can be observed in Figure 7.2.

Figure 7.3 illustrates the covariance matrix, computed with the squared exponential kernel applied to the coordinates from the [MDS](#) embedding.

Finally, we inspect the transfer matrix from FS4 to FS5. Using Equation (7.7) and scaling the rows so that the elements sum to 1, we obtain the matrix represented in Figure 7.4.

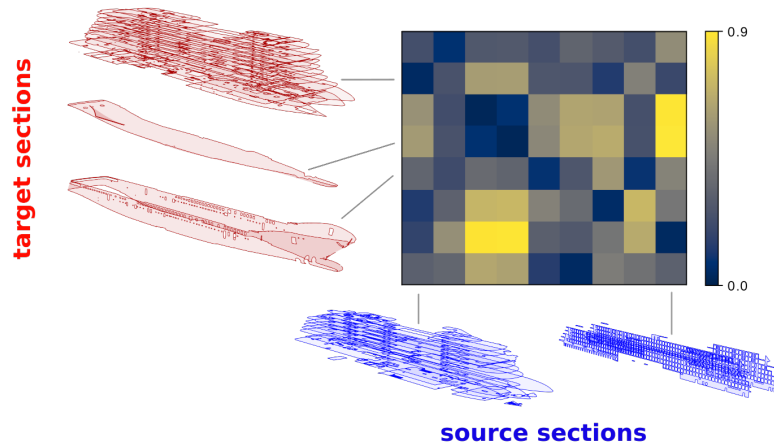


Figure 7.1: Chamfer distance matrix, with the columns corresponding to sections of the source ship (in blue), and the rows corresponding to sections of the target ship (in red). Lower values for the matrix entries correspond to more similar sections, as can be seen between the entry corresponding to the sets of accommodation decks, while the highest distance is observed between external bulkheads and the bottom parts of the shell.

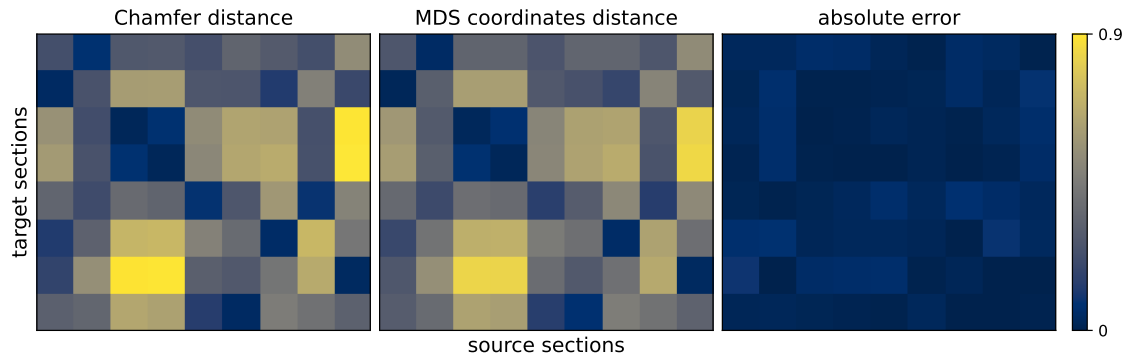


Figure 7.2: Comparison between the Chamfer distance computed on the scaled coordinates, the distance between the embedding coordinates computed via MDS, and the corresponding absolute error.

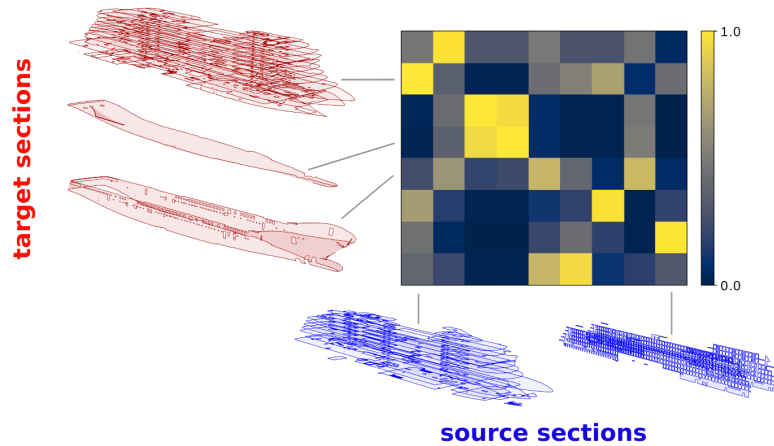


Figure 7.3: Covariance matrix between the source and target sections. The highest values are found for the entries corresponding to the pairs with lower Chamfer distance.

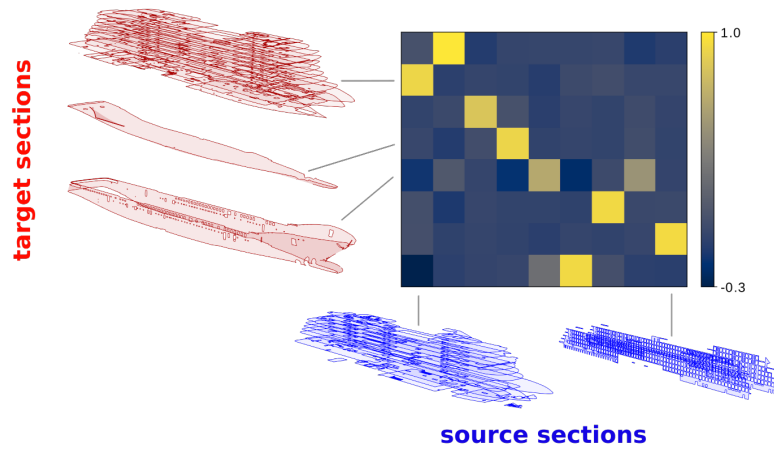


Figure 7.4: Transfer matrix between the source and target sections. Most rows have a single entry with a value close to 1.

Figure 7.5 highlights the case of two source sections contributing to the same target section. In this case, the shell of source ship is divided in two parameterized sections, which contribute almost equally to the value associated to the shell of the target ship.

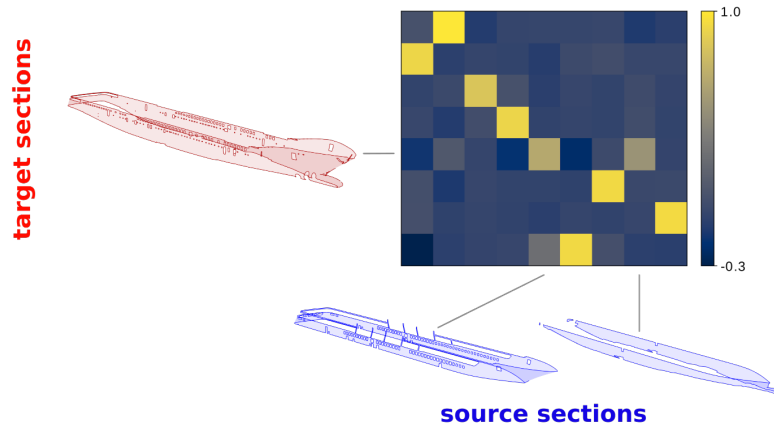


Figure 7.5: Transfer matrix between the source and target sections. The source shell is divided in two sections, which have very similar contribution to the output for the single shell section of the target ship.

The entire process is summarized in Algorithm 8.

The transfer procedure is inexpensive even for ships that have a large number of elements, due to the low complexity of the Chamfer distance. Moreover, the computation of MDS embeddings, covariance matrices, and the repair step can leverage independence between each pair of source and destination ships, achieving a high degree of parallelism.

Algorithm 8 Population transfer.

Input: population $\mathbf{X}^A$, lower bounds $\mathbf{x}_{LB}^A$, mesh elements coordinates S_j^A assigned to the j -th decision variable from ship A , lower bounds $\mathbf{x}_{LB}^B$, mesh elements coordinates S_i^B assigned to the i -th parameterized section of ship B , feasibility linear inequality constraints $\mathbf{g}(\mathbf{x}^B) \leq \mathbf{0}$ and discrete domain $\mathcal{D}^B$ for ship B , covariance kernel function k

Output: population $\mathbf{X}^B$ of feasible configurations
rescale S^A and S^B to be in the $[0, 1]^3$ cube
compute the dissimilarity matrices, using Equation (7.14)
solve the MDS to obtain embedding coordinates X^A and X^B
compute the transfer matrix M using k with Equation (7.7)
optionally, scale the rows of M
 $\mathbf{X}^B \leftarrow \mathbf{x}_{LB}^B + (\mathbf{X}^A - \mathbf{x}_{LB}^A)M^T$
repair each row of $\mathbf{X}^B$ with Problem 4.18
remove duplicated rows from $\mathbf{X}^B$
return $\mathbf{X}^B$

7.3 Numerical results

This section presents the evaluation of the proposed transfer procedure for multi- and single-objective optimization, for a group of midship sections and full ships.

For each dataset, we first select the combination of transfer hyper-parameters that results in the best trade-off of m_{gap} and number of transferred configurations. With the chosen hyper-parameters, we perform the transfer to initialize [NSGA-III](#) and [PSO](#), comparing the resulting optimization with a baseline random initialization.

All values are obtained using Fincantieri S.p.A.'s HPC facility, composed of 3 identical nodes equipped with an Intel Xeon Gold 6354 CPU (3.00 GHz), featuring 36 cores and 512 GB of RAM.

Section 7.3.1 presents the evaluation on the midship dataset, while Section 7.3.2 discusses the results on two full ships.

7.3.1 Midship section variants

We first aim to identify the combination of transfer hyper-parameters that achieves the best initialization of multi- and single-objective populations on the target ship. In particular, we consider the capability of a transferred population to contain a configuration close to the known optimum of the target ship. Given the 21 midship section variants, the validation set provides 14 best m_{gap} values. For each ship in the validation set, we transfer a population composed of the best configurations from the single-objective optimization from each different variant. After transformation, retrieve the best configuration from the aggregation of all populations so obtained. The difference between the reference m_{gap} and the transferred value becomes an element of a vector of size 14, of which we take the l^2 norm to measure the quality of the transfer. We denote this quantity as gaps loss.

A different gaps loss score is collected for each combination of hyper-parameters:

- number of top configurations from the source populations, varying from 100 to 5000,
- choice of r in the computation of the Chamfer distance generalization from Equation (7.14), either the elements' coordinates or the stress tensor components
- choice of covariance kernel, either the squared exponential from Equation (7.10) or Matérn 5/2 from Equation (7.11),
- whether the rows of the transfer matrix are scaled or not.

As the criterion for the selection of the hyper-parameters, we aim to identify the best trade-off between the number of unique transferred configurations and the resulting gaps loss score. In the implementation of Algorithm 8, we use the [MDS](#) interface from `scikit-learn` [142].

Figure 7.6 shows the results of the experiments, where we select 2500 as the best source population size due to diminishing returns from further increment.

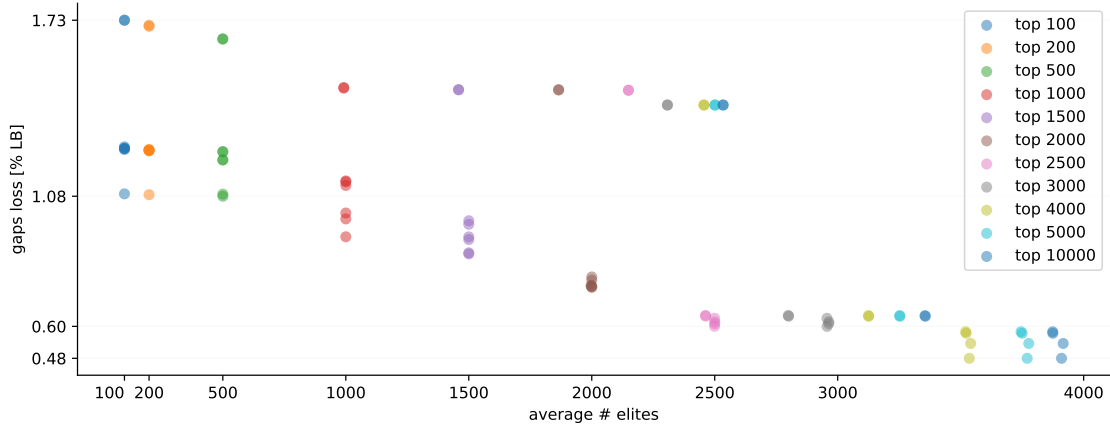


Figure 7.6: PF of gaps loss and average number of unique transferred configurations for population transfer, evaluated on the training set of midship section variants.

We note that the chosen threshold is quite high, in the sense that the resulting selections include configurations that have m_{gap} far from the source ship's optimum. However, the transferred results benefit from the inclusion of these sub-optimal configurations, possibly reproducing the effects described in [118] with seeding initial populations in case of negative transfer. We remark that the discrepancy between the selection size and transferred size is due to the repair procedure, which removes duplicates.

For the chosen selection size, we analyze the score of individual hyper-parameters combinations in Table 7.1.

The results show two outliers in the combination of stress tensor components as [QoIs](#) and non-scaling of the rows of the transfer matrix. The best combination is obtained with the use of coordinates, Matérn kernel and non-scaling of the transfer matrix, although all transfers that use the coordinates as QoI achieve similar scores of gaps loss. We thus fix

Table 7.1: Comparison of gaps loss for different combinations of transfer hyper-parameters, for the validation set of midship section variants.

d_{CH} QoIs	Kernel	Scaled	Gaps loss [% LB]
coordinates	squared exp.	raw	0.627
		scaled	0.614
	Matérn	raw	0.597
		scaled	0.607
stress tensors	squared exp.	raw	1.468
		scaled	0.635
	Matérn	raw	1.469
		scaled	0.637

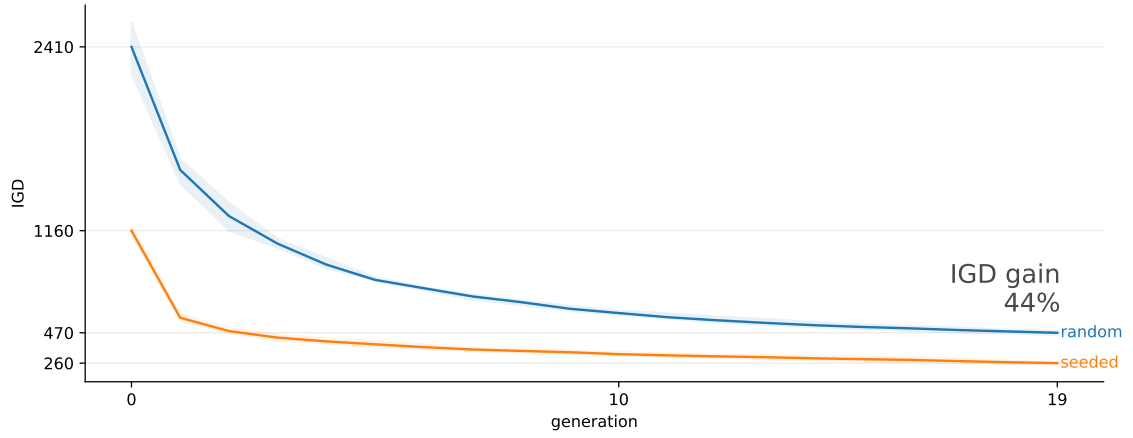
d_{CH} to be computed on the coordinates, the covariance kernel to be the Matérn 5/2, and the scaling of the transfer matrix rows to be skipped.

We now perform the actual multi-objective optimization on the 7 test midship section variants. For each target model, we use the transfer procedure on the best known PF of the 20 source variants and merge the resulting configurations. We create an initial population for NSGA-III by random selection of 2000 individuals from this merged dataset. We then evolve the population for 20 generations, and compare it to the evolution from a randomized maximin sampling of the domain as baseline. As a performance metric, we use the *inverse generational distance* (IGD) from the ideal PF to the current population, given by

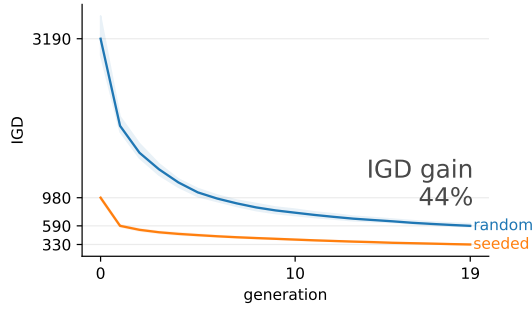
$$\text{IGD}(S^*, S) = \frac{1}{|S^*|} \sum_{s \in S^*} \|\mathbf{f}(s) - \mathbf{f}(c(s, S))\|, \quad (7.16)$$

which corresponds to the directed Chamfer distance from the ideal PF S^* to the candidate PF S . We use the vector of QoIs $\mathbf{f}$ from Equation (2.5), rescaled to the unit hypercube $[0, 1]^5$. To compute S^* for a test midship section variant, we collect all populations generated in the 20 runs of NSGA-III and compute a global PF. Other performance indicators can be found in the literature [181, 86, 117], but due to the construction of our ideal PF, it is guaranteed that the candidates S^A are dominated, thus the evolution of the IGD fully summarizes the performance of the approaches.

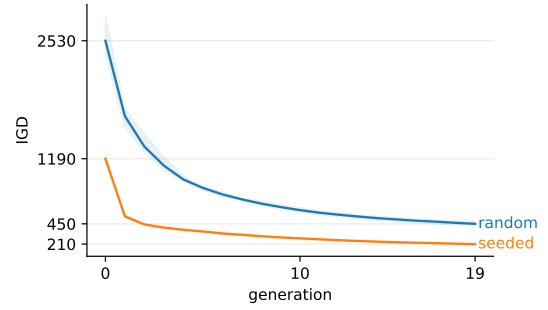
Figure 7.7 shows the IGD evolution for the test dataset. The shaded areas represent the difference between the best and worst values, while the lines follow the average across the 10 runs.



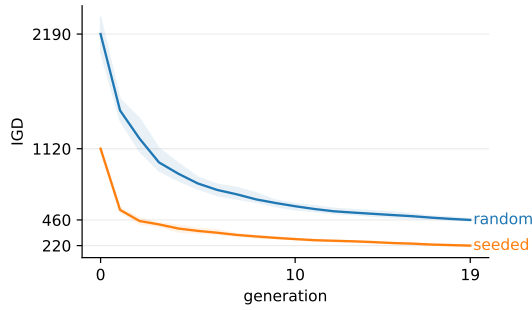
(a) MS00.



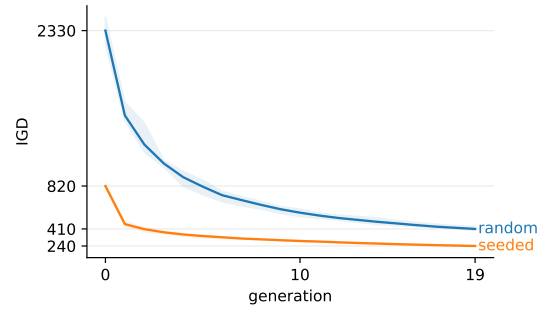
(b) MS01.



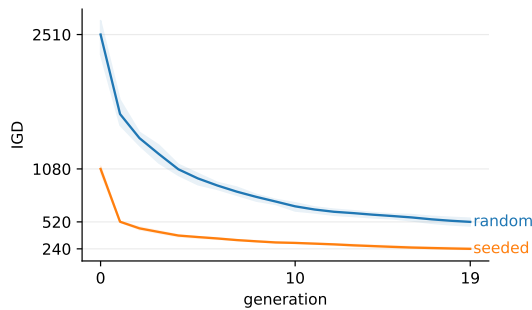
(c) MS02.



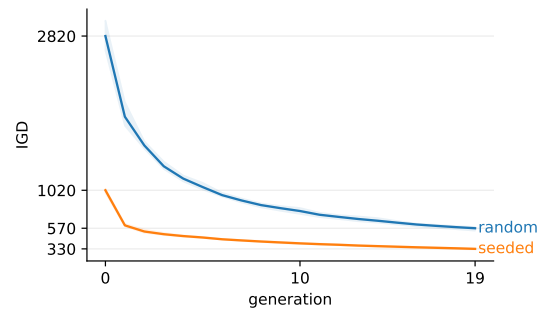
(d) MS03.



(e) MS04.



(f) MS05.



(g) MS06.

Figure 7.7: Evolution of IGD during NSGA-III for the midship section variants MS00 through MS06. The baseline random initialization is in blue, while the evolution of the seeded population is in orange. The shaded area shows the minimum and maximum values measured on 10 repeated runs.

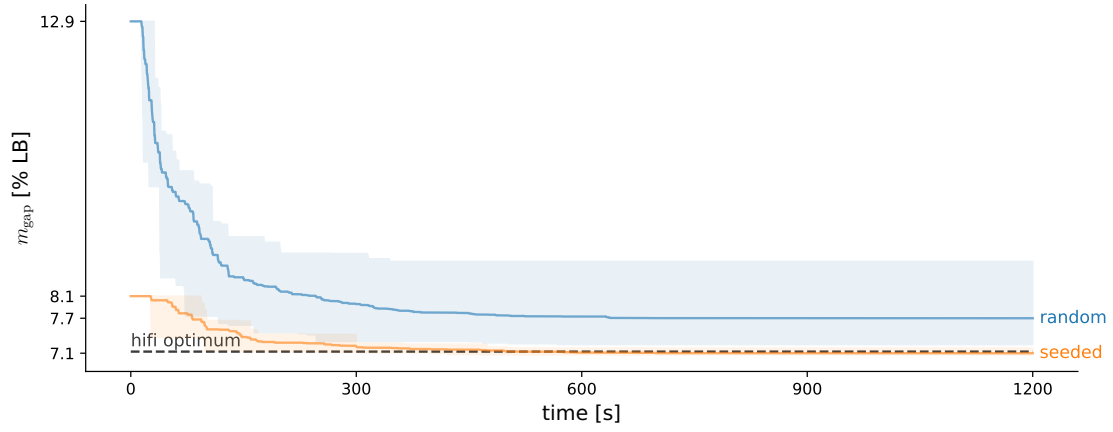
For all test cases, the seeded population starts from a much lower value of [IGD](#) compared to the random baseline. As the generations advance, the advantage diminishes but is never overcome: even at the end of a run, the seeded populations achieve [IGD](#) scores that are around half of those from the baseline. These results encourage the use of our transfer procedure to initialize the population of [NSGA-III](#). We remark that this seeding procedure is inexpensive, even with a large number of configurations involved. Indeed, the transfers themselves can be carried out in an offline phase, leveraging the high degree of parallelism due to independence of source variants, while in the online phase the only operation is accessing the stored transferred populations.

We now study the effect of transfer learning for the initialization of the population of [PSO](#). In this case, we select the 2500 configurations with the best m_{gap} from the latest optimization history of each source midship section variant different from the target. For each test variant, we merge the transferred populations and pass the result as input to Algorithm 4 to generate the initial population for [PSO](#).

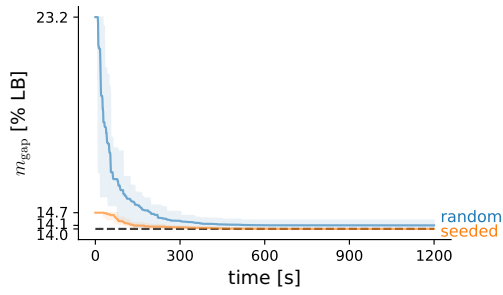
Figure 7.8 shows the evolution of the [PSO](#) incumbent for the test dataset. The reference optimum is represented by the dashed line. The shaded areas represent the difference between the best and worst values, while the lines follow the average across the 10 runs.

For all test cases, the seeded population provides an initial incumbent that is closer to the best known optimum compared to the incumbent from the baseline. In almost all cases, the seeded runs are able to reach the optimum in the allotted time. However, the random baseline is often able to overcome the large initial gaps and reach final m_{gap} values close to the optimum. In the optimization of MS00, the random baseline shows a larger difference between best and worst case, with the mean incumbent over 0.6 % over the optimum. On MS03, both setups show convergence of the average incumbent close to 7.8, and only the randomized baseline was able to close in on the best optimum of 7.56. On MS05, the seeded population is able to outperform the best known optimum, although the difference is less than the termination threshold. Overall, these results suggest that the transfer learning approach is able to provide performant configurations that can quickly be refined to very good quality configurations.

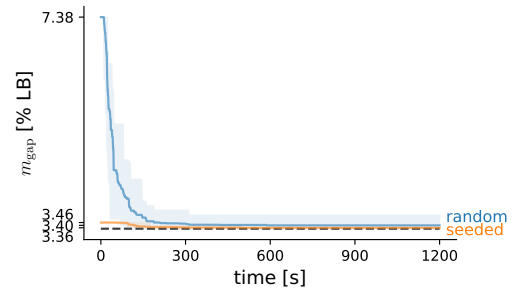
However, we remark that Algorithm 4 requires the evaluation of m_{gap} of the entire seeding population, a computation that could require a computational effort of the same order of magnitude of the [PSO](#) execution.



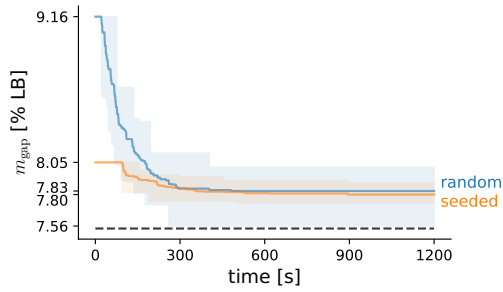
(a) MS00.



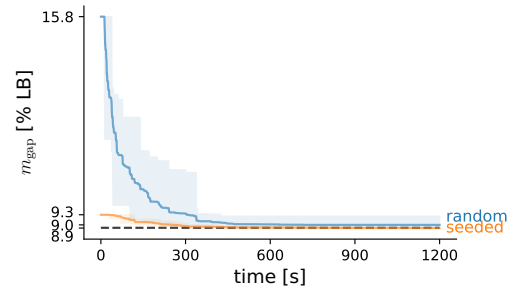
(b) MS01.



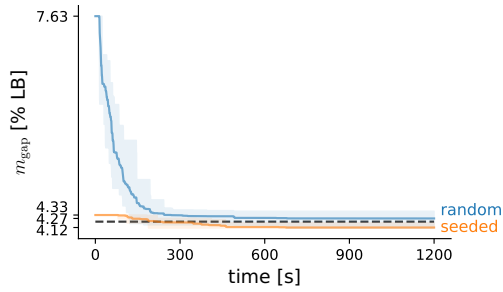
(c) MS02.



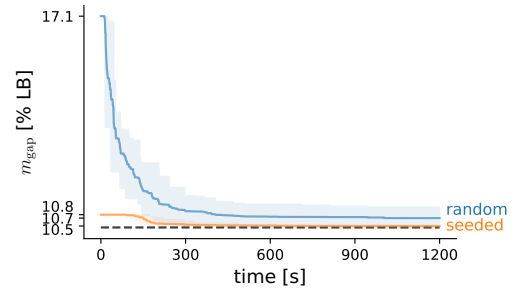
(d) MS03.



(e) MS04.



(f) MS05.



(g) MS06.

Figure 7.8: Evolution of m_{gap} during PSO, for the midship section variants MS00 through MS06. The baseline random initialization is in blue, while the evolution of the seeded population is in orange. The shaded area shows the minimum and maximum values measured on 10 repeated runs. The dashed line represents the reference optima from the database.

7.3.2 Full ship fleet

Since the effectiveness of the population-transfer approach was demonstrated on the midship section variants, which represent geometrically similar tasks, here we choose FS7 and FS9 as the target tasks. The difference in dimensions and proportions, compared to the rest of the dataset, marks them as out of distribution samples.

First, we select the transfer hyper-parameters with the same approach used in Section 7.3.1, to achieve the best trade-off between the number of unique feasible configurations and the resulting gaps loss. Figure 7.9 shows the evaluation of the different hyper-parameters combinations.

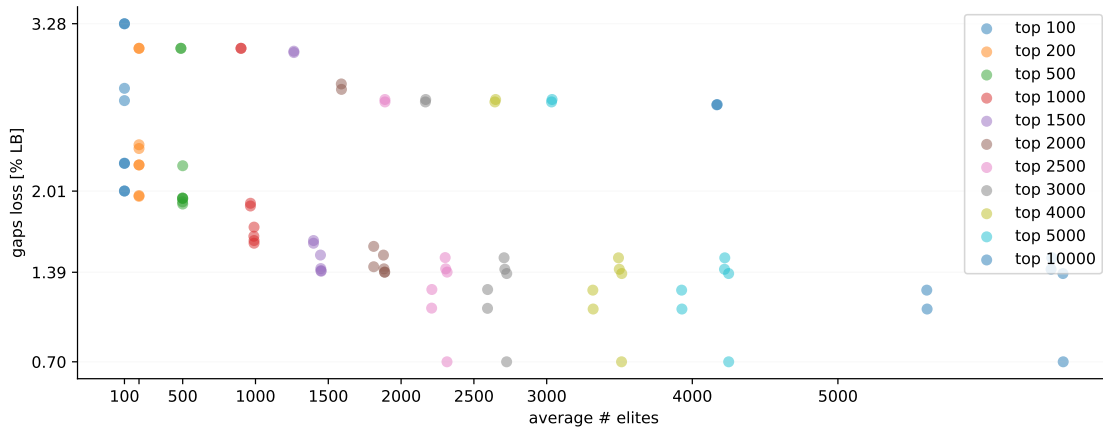


Figure 7.9: PF of gaps loss and average number of unique transferred configurations for population transfer, evaluated on the training dataset of full ships.

Again, the best trade-off is achieved for population size 2500, with higher values unable to further decrease the gaps loss. Table 7.2 collects the detailed scores for the different combinations of hyper-parameters.

Table 7.2: Comparison of gaps loss for different combinations of transfer hyper-parameters, for the validation set of full ships.

d_{CH} QoIs	Kernel	Scaling	Gaps loss [% LB]
coordinates	squared exp.	raw	1.498
		scaled	0.702
	Matérn	raw	1.412
		scaled	1.387
stresses	squared exp.	raw	2.687
		scaled	1.255
	Matérn	raw	2.706
		scaled	1.112

On this dataset, the scores show higher variety than in the previous case, with a single combination achieving a much better gaps loss than the rest using elements' coordinates as d_{CH} QoIs, squared exponential kernel, and scaled transfer matrix rows. The worst behavior

is still found for the same combinations using stress tensor components as d_{CH} QoIs and non-scaled transfer matrix rows, while all remaining experiments result in similar scores.

We now perform the same multi-objective optimization experiment as the previous section. Figures 7.10 and 7.11 show the IGD evolution for the transferred population and the random baseline, for FS7 and FS9 respectively.

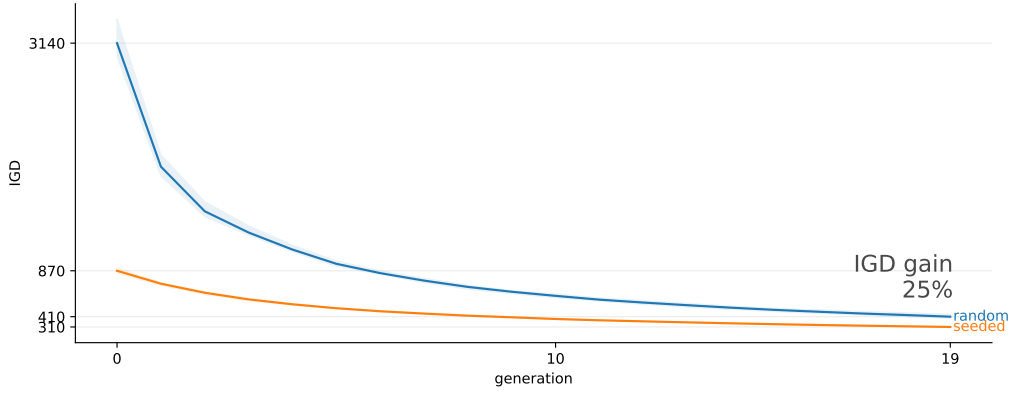


Figure 7.10: Evolution of IGD during NSGA-III for FS7. The baseline random initialization is in blue, while the evolution of the seeded population is in orange. The shaded area shows the minimum and maximum values measured on 10 repeated runs.

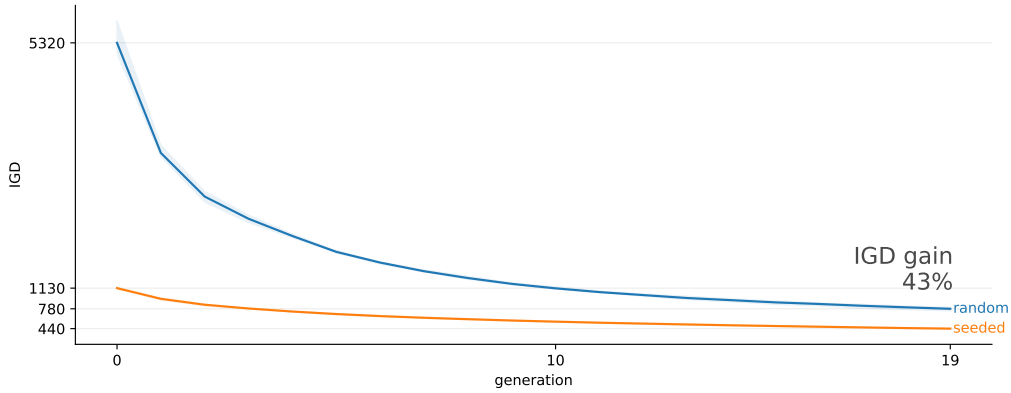


Figure 7.11: Evolution of IGD during NSGA-III for FS9. The baseline random initialization is in blue, while the evolution of the seeded population is in orange. The shaded area shows the minimum and maximum values measured on 10 repeated runs.

Even on these out-of-distribution cases, the seeded population shows faster convergence to a lower IGD compared to the random baseline.

As for the single-objective optimization, Figures 7.12 and 7.13 show the comparison of the incumbent evolutions for FS7 and FS9 respectively.

In the case of FS7, the average seeded run is able to reach the best known optimum, while the average baseline approach gets within 0.3 gap points. The optimization of FS9 proves more difficult: the average seeded run beats the baseline approach by 0.5 gap points, but is not able to overcome a 1.3 points gap from the best known optimum.

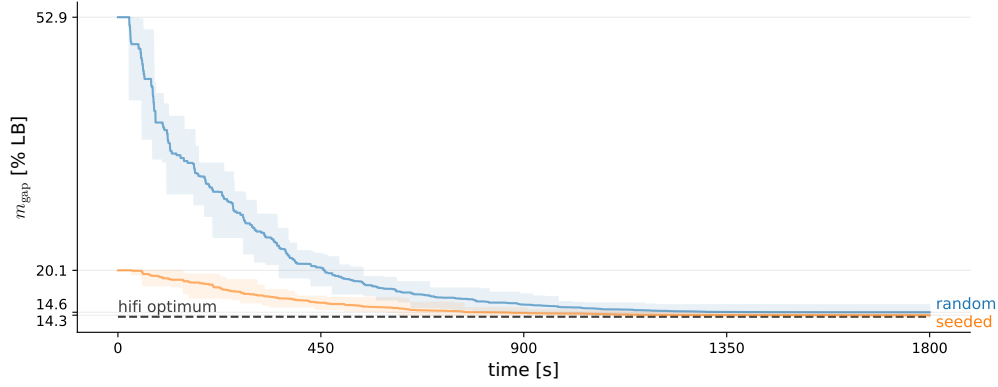


Figure 7.12: Evolution of m_{gap} during PSO, for FS7. The baseline random initialization is in blue, while the evolution of the seeded population is in orange. The shaded area shows the minimum and maximum values measured on 10 repeated runs. The dashed line represents the reference optimum from the database.

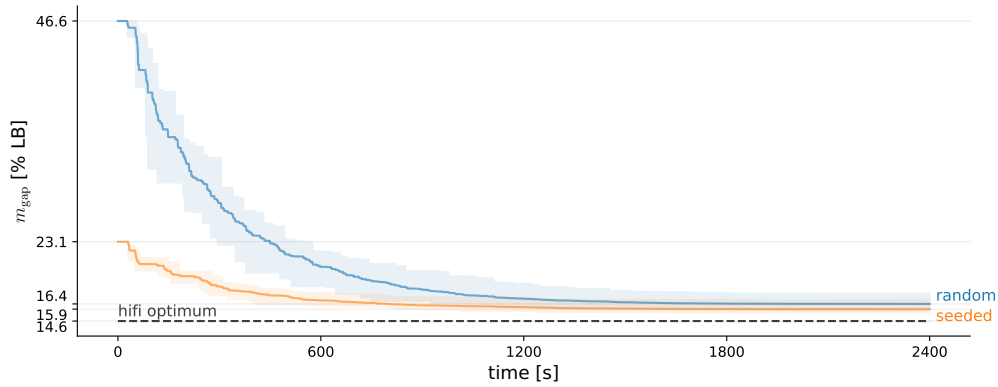


Figure 7.13: Evolution of m_{gap} during PSO, for FS9. The baseline random initialization is in blue, while the evolution of the seeded population is in orange. The shaded area shows the minimum and maximum values measured on 10 repeated runs. The dashed line represents the reference optimum from the database.

7.4 Conclusions

This chapter introduced a population transfer approach for the initialization of metaheuristics algorithms, able to leverage the configurations from past optimizations on different ships. The proposed methodology frames the transfer problem in terms of Gaussian process regression and defines the covariance in terms of the similarity between the metadata available at the start of the optimization pipeline. By computing the Chamfer distance between the parameterized sections of the computational mesh, the transfer procedure can be performed offline without the evaluation of structural [QoIs](#) for the target ship.

The numerical experiments on both the midship section variants and the full-scale ship models demonstrate that the proposed strategy significantly accelerates the convergence of multi-objective optimization, while its impact on single-objective optimization is limited.

The next chapter departs from the structural optimization problem to tackle the life-cycle dimension of the design process, by applying the digital twins paradigm to structural health monitoring.

STRUCTURAL DIGITAL TWINS

The integration of all aspects of the ship's life-cycle during the design, from construction to dismantling, constitutes a necessary step to tackle modern requirements by industry and governments with a holistic view [139, 98, 208, 81]. Paraphrasing from the definition given in [126], a [digital twin \(DT\)](#) is a set of virtual constructs that mimics a real-world phenomenon and is able to predict its evolution, is dynamically updated from its physical counterpart, and informs decisions that realize value. During the design phase, the designers can leverage the [DT](#) by simulating past historical conditions and evaluate the current ship under realistic loads, with the same data assimilation procedure available during operations [103, 124, 192].

However, the development and validation of a [DT](#) for complex engineering systems remain challenging, due to the lack of standardization and the difficulty in the integration of multiple interacting components, often problem- and process-specific. [Probabilistic graphical models \(PGMs\)](#) [114] offer an elegant framework to handle the complexity of [DT](#) design, but the lack of high-level software libraries slows their adoption.

In this chapter we present `pgmtwin`, a novel open-source Python library for the development and the analysis of [DT](#), based on [PGMs](#). By providing clear interfaces and powerful base routines, as well as versatile ready-to-use components, `pgmtwin` streamlines the implementation of complex [DTs](#) that integrate the user's own simulations, sensor analysis, and decision-making components. The library is demonstrated on the [structural health monitoring \(SHM\)](#) of a simplified cantilever beam.

8.1 Introduction

Being able to simulate and monitor the entire life-cycle of a vessel from design, to manufacturing, to the operational period, and finally to its dismantling, is necessary to address the growing requirements for the reduction of emissions and resource consumption, as well as the increase in the efficiency of operating costs [139, 98, 208, 81].

During the design phase, the simulation against historical data can help reveal and correct issues that would not be identified by usual validation practices. The operational phase can benefit from the continuous monitoring of the ship's performance to identify unexpected degradation of the materials and systems, as well as proposing changes in the current regime to increase efficiency. During manufacturing and dismantling, the shipyard can detect anomalies in the structure's behavior and respond accordingly, increasing security and preventing loss of material and time [115].

Past approaches to holistic design mainly focus on incorporating the uncertainties of the operations phase, using stochastic programming [58] or simulating a voyage from few historical data [127]. However, these approaches consider the ship in operation as a passive component of the simulation and can not account for, or validate, the decision-making aspect of the ship's life-cycle.

Thanks to the advancements in computational power and modeling capabilities, the adoption of DTs throughout the vessel's life provides a versatile framework to address the demands from industry and regulatory entities, and to develop a holistic design process. Quoting from [126], a DT is

a set of virtual information constructs that mimics the structure, context, and behavior of a natural, engineered, or social system (or system of-systems), is dynamically updated with data from its physical twin, has a predictive capability, and informs decisions that realize value. The bidirectional interaction between the virtual and the physical is central to the digital twin.

From a practical point of view, a proper DT system is thus characterized by a loop of communication flows: from physical to digital via sensors reading and data assimilation, from the digital asset to the decision-making component by the inspection of the digital state, and from the decision-making component to the physical asset via actions. Figure 8.1 represents a DT system.

Although the primary applications of DTs are in the field of control, such as SHM of infrastructures [192, 191], autonomous navigation of unmanned vehicles [189], and administration of medical treatments [102], the framework is equally applicable during the design phase. By substituting the generation of sensor data with a simulation or historical data, the same DT can be used to assess the model's configuration, calibrate the inverse problem and validate the decision making component.

However, the adoption of DTs in the naval industry is still in its early stages. Most of the applications in the literature are tied to the operations phase, while manufacturing

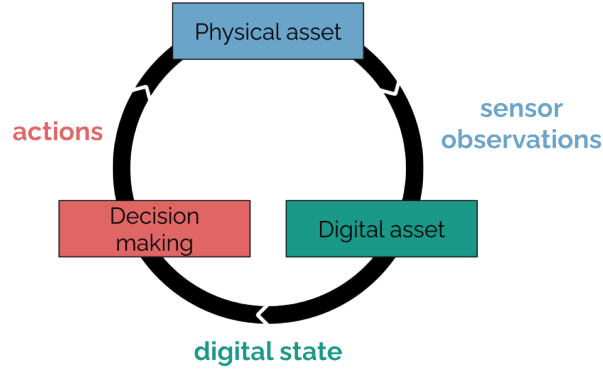


Figure 8.1: Representation of a DT, combining a physical asset, a digital asset, and a decision-making component. The exchange of information is obtained via the generation and communication of different data from one component to the next.

and dismantling are the least considered aspects of the ship’s life-cycle [124, 123]. With respect to the design phase, the literature acknowledges the potential improvements that **DTs** can provide, but has been so far limited to theoretical frameworks [209, 138] and lacks concrete implementations.

One of the main challenges for the adoption of **DTs** in the design phase is represented by the complexity of their simulation, which in the case of a full cruise ship requires orchestrating numerous interacting subsystems. **PGMs** [114] provide a powerful framework to address this complexity, as demonstrated in [103, 192, 196, 189, 191]. Through the explicit modeling of the causal dependencies between subsystems, **PGMs** enable the efficient propagation of uncertainty from the simulations to the data assimilation components, allowing the evaluation of multiple scenarios within a single, unified framework. Even so, current implementations of **PGMs** such as pgmpy [6, 7] or Pyro [25] provide only low-level building blocks, requiring users to have extensive expertise in these frameworks to develop complex **DTs**.

We propose a new open-source Python library to facilitate the design, analysis and simulation of **DT** systems based on **PGMs**, pgmtwin [66]. Built on top of the pgmpy library, pgmtwin provides high-level interfaces that easily accommodate case-specific business logic via composition, while standard core procedures organize the update of entities, the assimilation of sensor readings, and the decision-making. Following a clear workflow, the users first focus on the definition of states and actions, and integrate their own simulation and inverse problem. The decision-making component can be supplied by the users, or obtained via standard **reinforcement learning (RL)** [182] frameworks [150, 193]. The library offers out-of-the-box support for an **SHM DT** adapted from [192], but advanced users can easily extend the **PGM** and entities to support more complex setups.

The rest of this chapter is organized as follows. In Section 8.2 we introduce the **PGM** framework, and Section 8.3 describes the workflow of pgmtwin. A showcase of the library, we implement and evaluate an **SHM** application for a cantilever beam in Section 8.4. The final remarks are given in Section 8.5.

8.2 Probabilistic graphical models

In this section we follow the notation from [114]. Let $\mathcal{X} = \{X_i\}_{i=1}^n$ be a set of random variables. Then let P be a joint probability distribution for $\mathcal{X}$. A **PGM** is a graph representation of the conditional relationship between the elements of $\mathcal{X}$, where each X_i for $i = 1, \dots, n$ is associated to a node, and the arcs define the dependencies between the random variables. In particular, a **Bayesian network (BN)** is a **PGM** that uses a **directed acyclic graph (DAG)** $\mathcal{G}$ and where to each node X_i is associated a **conditional probability distribution (CPD)** $P(X_i | \text{Pa}_{X_i}^{\mathcal{G}})$, with $\text{Pa}_{X_i}^{\mathcal{G}}$ denoting the set of nodes in $\mathcal{G}$ that are the origin of the arcs entering X_i . The elements of $\text{Pa}_{X_i}^{\mathcal{G}}$ are called the parents of X_i . In a **BN**, nodes without any parents have an a-priori **CPD**.

The joint probability P factorizes according to the structure of $\mathcal{G}$ as

$$P(\mathcal{X}) = \prod_{i=1}^n P(X_i | \text{Pa}_{X_i}^{\mathcal{G}}). \quad (8.1)$$

The graph structure and Equation (8.1) enable the automated application of Bayes theorem to efficiently perform inference from acquired evidence, without the need for manual analysis and ad-hoc inference procedures. When some of the variables are observed, their influence is propagated to their children nodes via the **CPDs** and the computation of marginalized probability distributions, for the nodes of interest, can be optimized via dynamic programming [46]. An example of such inference procedure is the variable elimination from [114].

A **dynamic Bayesian network (DBN)** extends the concept of a **BN** by considering $\mathcal{X}$ as collecting multiple series of random variables, for which a **BN** act as a template structure, indexed by time. In this setup, each time slice has a set of associated variables, which re-create the **BN** of the template. Given a **BN** with the previous notation, X_i is the template for a series of variable instances $\{X_i^{(t)}\}_{t \geq 0}$ and each arc (X_i, X_j) generates a corresponding arc among the instances $(X_i^{(t)}, X_j^{(t)})$ associated to its vertices.

Then, the joint distribution P on the set of variables from time 0 to time T becomes

$$P(\mathcal{X}^{(0:T)}) = P(\mathcal{X}^{(0)}) \prod_{t=0}^{T-1} P(\mathcal{X}^{(t+1)} | \mathcal{X}^{(0:t)}), \quad (8.2)$$

where $\mathcal{X}^{(0:T)}$ is the subset of variables associated to slices from 0 to T . It is common to assume that a system is Markovian, from which Equation (8.2) simplifies to

$$P(\mathcal{X}^{(0:T)}) = P(\mathcal{X}^{(0)}) \prod_{t=0}^{T-1} P(\mathcal{X}^{(t+1)} | \mathcal{X}^{(t)}), \quad (8.3)$$

reducing the communication between frame t and the next to the arcs $P(\mathcal{X}^{(t+1)} | \mathcal{X}^{(t)})$.

Figure 8.2 shows a **DBN** tailored to a generic **DT** system, adapted from [192]. At time t , a physical state $S^{(t)}$, usually unobservable, is influenced by the action performed at the previous timestep $U_{\text{PREV}}^{(t)}$ and determines the sensor observations $O^{(t)}$. The information of the sensors is processed by an inverse problem, retrieving an estimation of the digital state used by the **DT** in the node $D_{\text{IP}}^{(t)}$. The $D_{\text{IP}}^{(t)}$ is separate from the actual digital state, since we can often obtain a **CPD** for $P(D_{\text{IP}}^{(t)} | D_{\text{IP}}^{(t-1)})$, for example pre-computing a confusion matrix in

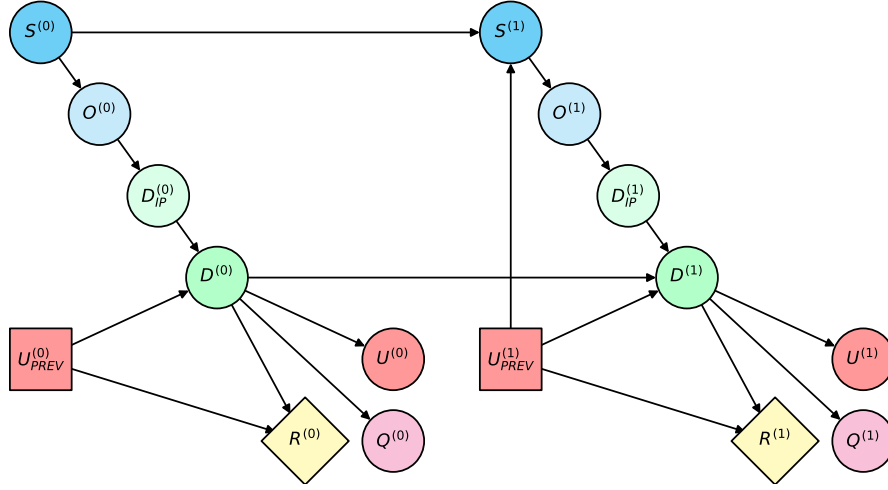


Figure 8.2: The first two slices of a DBN for a generic DT system. Circles denote random variables, squares denote actions taken, and diamonds identify the reward.

case of discrete states. The combination of $D_{IP}^{(t)}$, $U_{PREV}^{(t)}$ and, if available, the previous digital state $D^{(t-1)}$ provides the assimilation of the sensor information in the current digital state. It is common to define a reward function, computed from $U_{PREV}^{(t)}$ and $D^{(t)}$, to quantify the efficacy of the decision-making component of the DT, as well as other [quantities of interest \(QoIs\)](#) $Q^{(t)}$. Finally, the probability of a specific action being performed on the physical state given the estimation of $D^{(t)}$ is $U^{(t)}$, and the actual action performed is observed in $U_{PREV}^{(t+1)}$. We remark the presence of two nodes related to the action choice: $U_{PREV}^{(t)}$ indicates an observed value and is always provided as evidence during inference, while $U^{(t)}$ is retrieved as a probability distribution over the possible actions.

At each instant, a DT queries the sensors and performs the inverse problem to obtain a probability distribution of the inverse problem's result. In the simplest case, this result is represented by a confidence interval or a Gaussian in the continuous case, or a normalized count vectors in the case of discrete random variables. The estimation of the digital state requires the combination of three arcs, which are conceptually separated in two relationships. First, the CPD that relates the inverse problem to the current digital state distribution captures the known bias and inaccuracies of the procedure, and its construction requires, for example, estimation on historical data. Second, the CPD of the digital state conditioned on the previous action and the previous digital state models the a-priori knowledge of the update mechanism. The combination of these CPDs can be carried out under an independence assumption, carried out by simple multiplication. The arc between the digital state and the action represents the decision-making component of the DT, often implemented via RL [182]. A common choice is to determine a tabular policy in the case of discrete digital state and action space, which also translate to tabular CPDs, but recently the use of neural network policies has become the norm [150].

The graph structure allows to potentially extend the DT system to integrate separate digital states and multiple inverse problems, enabling modular DTs [90, 89] with minimal

intervention in the inference procedure. Other area of research recently enabled by the use of [PGMs](#) are the integration of multi-scale simulations in the evolution of the digital state [148] and the formal verification of an entire [DT](#) system under unreliable communications [94].

8.3 `pgmtwin`

[PGMs](#) constitute a powerful framework for the implementation of [DTs](#), but the construction of structures like the one in Figure 8.2, the integration of case-specific components for the sensor reading, inverse problem solution, and decision-making component constitutes an obstacle to widespread adoption.

With `pgmtwin`, we aim to address the difficulties in setting up a practical [DT](#) system based on [PGMs](#). The focus of the library is on the implementation of foundation entities and processing loops in a core module, which are then customized via composition (also known as the "strategy" design pattern [77]).

The user is only required to define the basic metadata (states and actions) and provide the implementation of the business logic (physical asset update and observation, inverse problem, reward) which gets combined by the library's entities and used in standardized procedures. `pgmtwin` provides the update loop of the physical and digital assets via the `gym.Env` class from the widely used Gymnasium library [193], enabling the integration of standard [RL](#) frameworks such as Stable-Baselines3 [150] for the decision-making component of the [DT](#) system. The library `pgmpy` [6, 6] provides the implementation of the [PGM](#) and the inference procedures, while `pgmtwin` adds routines to streamline the construction and analysis.

Since the specific structure of a [PGM](#) is application specific and depends on the action and domain definitions, `pgmtwin` provides a `toolkits` module with ready-to-use specialized classes and the corresponding [PGM](#)-enabled environment, for customization via composition. The `toolkits` module also implements specific simulation loops and procedures for the visualization of the results, facilitating a structured approach to the evaluation of the [DT](#) system.

Figure 8.3 represents the workflow for the implementation of a new [DT](#) system with `pgmtwin`. In this scenario, we assume that the user has already implemented a simulation of the physical asset, which is an observable process that is influenced by actions applied to it. The same workflow applies to the case where the [DT](#) system is deployed in the real world and instead of a simulation, the observations are obtained via sensors and the actions have physical consequences.

The first step is the discretization of the digital states and the available actions using the `DiscreteDomain` and `BaseAction` classes. At the moment, `pgmtwin` only supports discrete states and actions, with `DiscreteDomain` providing routines for bookkeeping and conversions, and `BaseAction` implementing the update. In particular, the `ActionCPD` class specifies the transitions between states via a [CPD](#) matrix.

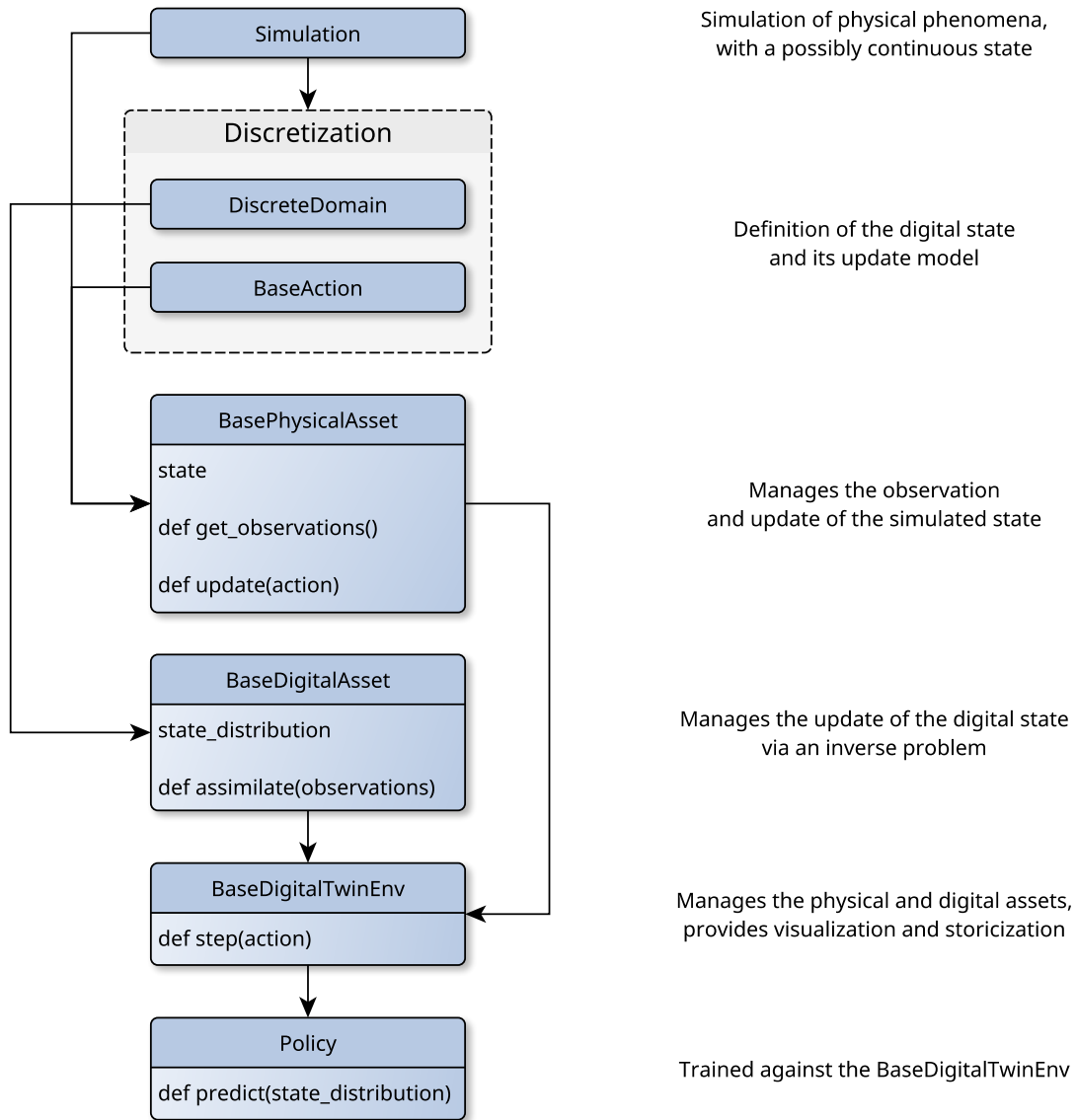


Figure 8.3: Implementation diagram of a new DT system using `pgmtwin`.

The simulation and the actions inform the customization of the `BasePhysicalAsset` class, which is responsible for holding the (possibly continuous) state of the simulation, updating it according to the supplied actions, and generating observations. Each of these functionalities are implemented by a specific component, that specifies the interface to be filled by the user's logic. For example, the user extends the `ObserverComponent` to implement the observation of the sensor values, which are in general different from the physical state, often with an efficient surrogate of the original simulation.

The observations generated are passed to the `BaseDigitalAsset`, which is responsible for the estimation of the probability distribution of the digital state. In the main implementation of the base class, `InverseProblemConfusionDigitalAsset`, the user supplies a `Callable[[np.ndarray], np.ndarray]` and a confusion matrix for the uncertainty

quantification of the inverse problem.

The `BasePhysicalAsset` and the `BaseDigitalAsset` are then used to initialize a `BaseDigitalTwinEnv`, which encapsulates the simulation and assimilation procedures. This approach separates the assets' operations from the decision-making component of the `DT`, enabling the use of popular frameworks for `RL`. As mentioned before, this class manages the `PGM` inference by combining the inverse problem, the previous state, and the history of actions. Due to the `PGM`-specific inference operations, the toolkits also provide standardized loops for simulating, logging, and visualize the evolution of the `DT` system.

The implementation of a `Policy` class completes the `DT` system. `pgmtwin` is compatible with `Stable-Baselines3`'s models, but also provides a `TabularPolicy` class that implements Q-learning and value-iteration [182].

The component-based approach and the minimal method signatures help to write reusable, independent behaviors and to easily compare multiple `DT` configurations in a unified manner. Particular care was taken to reduce coupling between components and to ensure reproducibility via explicit initialization of all configurable elements. The use of inheritance is only necessary for the more complex aspects of the `DT`, such as a new `PGM` topology, but the effort remains limited even in these use cases.

8.4 Numerical results

The `shm` toolkit, based on the work in [192], is the basis for the following section. Our application is intended as a showcase of `pgmtwin`'s functionalities, and thus is much less complex. We will follow the workflow from Figure 8.3 to compare a baseline `DT` system, which performs data assimilation with just a pre-computed confusion matrix for the inverse problem, and one that instead leverages the `DBN` from Figure 8.2.

All results are obtained with version `v0.0.1-alpha` of `pgmtwin`, available at the repository in [66]. The environment was built with the provided environment `yaml` file and by installing the full version of the library. All data for the figures and the numerical values are obtained with the execution of the script `examples/basic_cantilever/oneshot.sh`.

8.4.1 Structural health monitoring of a cantilever beam

We aim to implement a structural `DT` for the `SHM` of a cantilever beam. The beam is modeled as a 4 m-long prism with a square cross section of side length 0.3 m. The material is assumed to be concrete, with a density of 2500 kg/m^3 , a Young's modulus of 30 GPa, and a Poisson's ratio of 0.2. A pressure of 60 KPa is applied to the top face at the free end of the beam, over a square area equal to the cross-section. The beam undergoes structural degradation, modeled as a reduction of the Young's modulus in either one the two longitudinal halves. Only one section of the beam can be damaged at any one time, with the entity of reduction ranging from 25% to 85%. The severity of the simulated

damage is intentionally severe, to facilitate the demonstration of the framework. The physical state of the system is thus defined by two scalars: the location of the damage, and its level. We represent the location with three labels: 0 indicates the undamaged state, 1 represents damage in the half closest to the clamped end, and 2 indicates damage in the half with the free end. The second number is the percentage of Young's modulus reduction. We envision three possible maintenance actions that can be applied to the beam: "do nothing" which risks further degradation; a "minor repair" that slightly reduces the degradation, and a "perfect repair" that fully restores the Young's modulus.

The high-fidelity simulation of the beam is carried out with a [finite element analysis \(FEA\)](#) simulation, modeled under the assumptions of static linear elasticity, implemented in FEniCSx [12, 168, 167, 3]. We build a high-fidelity database of 400 simulations, sampling the parametric domain at random. Figure 8.4 shows three examples of simulation results.

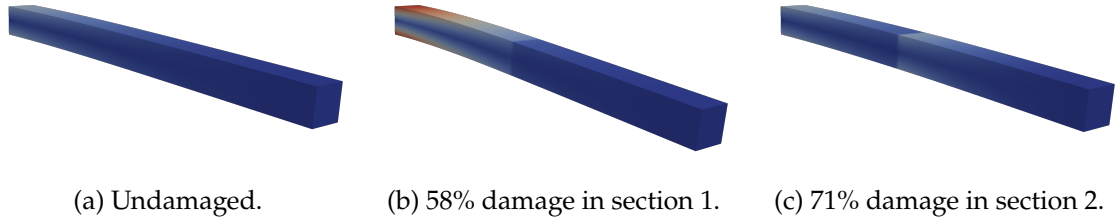


Figure 8.4: Strain of the deformed cantilever beam under three different levels of damage. No damage on the left, 58% damage in the section closest to the clamped end at the center, and 71% of damage in the section further from the clamped end on the right. The colors represent the magnitude of the strain, with blue being 0 and red indicating the largest values.

Following the schema in Figure 8.3, we first construct a discretization of the domain and the actions' effects. To represent the digital state, consisting of a discrete level of damage and its location, we use the `SingleDamageDomain` class specifying 7 levels of damage: 0%, 30%, 40%, 50%, 60%, 70%, and 80%. Since the damage can only be present in a single section at a time, the domain consists of 13 different states. The translation from continuous physical state and discrete digital state simply selects the closest level of damage, while the damage location is taken identical to the physical state.

The three maintenance actions are modeled via stochastic transition tables. When the "do nothing" action is chosen, the beam remains at the current level of damage with probability 0.5, otherwise the damage increases to the next level. If the beam is undamaged, it has equal probability of applying damage to the first or second half. The "minor repair" action reduces the level of damage by one level with probability 0.75, otherwise the damage remains unchanged. When the beam undergoes "perfect repair", all damage is removed. Figure 8.5 represents the three maintenance actions.

To simulate a realistic monitoring setup, we place 4 displacement sensors on the bottom side the beam at increasing distance from the clamped end: 1.6 m, 2.25 m, 3.1 m, and 4 m. Figure 8.6 shows the positions of the sensors.

To construct a fast surrogate of the high-fidelity simulation, we use [Gaussian process](#)

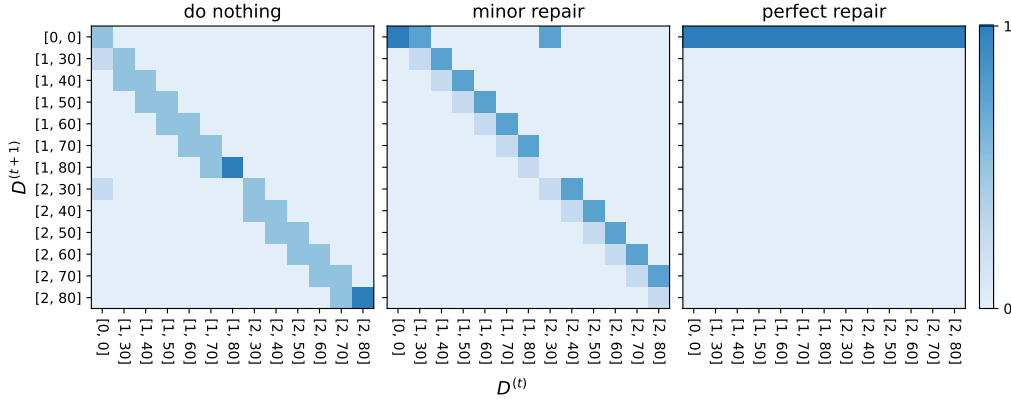


Figure 8.5: Transition matrices of the SHM maintenance actions. The "do nothing" and "minor repair" maintenance actions risk degrading the structural integrity of the beam, while "perfect repair" guarantees full restoration. Darker shades of blue represent higher probability of transition.

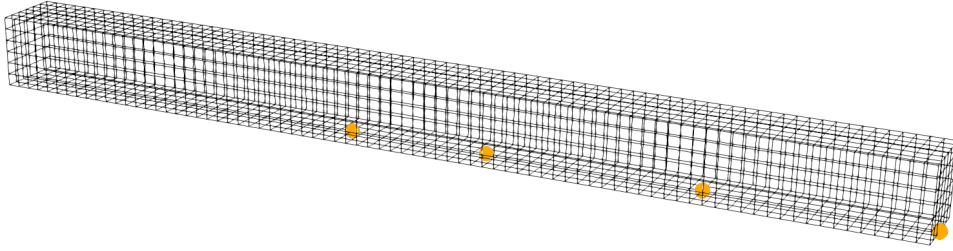


Figure 8.6: Placement of the displacement sensors on the cantilever's mesh.

regression (GPR) from SciPy [197] to reconstruct the sensors' readings given the physical state. For the inverse problem, we use a two-layer approach based on `xgboost` [37]. First, we generate 2000 instances of sensor readings via the surrogates. Then, we train a first `xgboost` model to classify the damage location based on the four vertical displacements. For each damage location, we train a separate `xgboost` model to discern the level of damage. During training, the sensors are corrupted with the addition of Gaussian noise centered in 0 and with standard deviation 0.003. This correspond to a signal-to-noise ratio of around 12.22 for the sensor closest to the clamped end, and 45.57 for the one at the free end. In the online phase, we first identify the damaged section and then, if not undamaged, detect the level of damage using the corresponding model. Using an 80-20 train/test split, the first model achieves 94.50% accuracy on the test set, the model for the damage located close to the clamped end obtains 96.83%, and the final model achieves only 56.28%. We evaluate the classifier on the original high-fidelity data, replicated and corrupted by noise, and obtain the confusion matrix represented in Figure 8.7. The confusion matrix shows the difficulty in correctly discerning the damage in the section close to the free end of the beam, which sustain much lower deformation. On this high-fidelity dataset, the overall accuracy of the inverse problem is 73.28%.

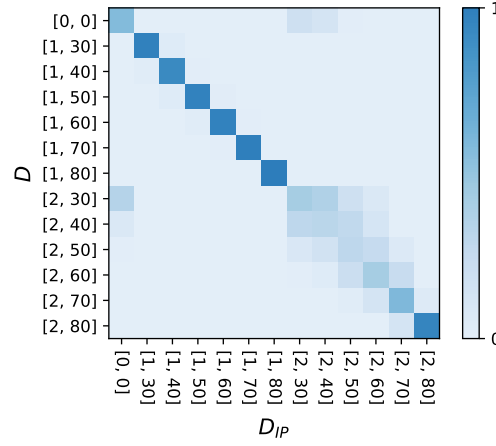


Figure 8.7: Confusion matrix for the SHM inverse problem. Darker shades of blue indicate a higher score associated to the truth and prediction combination. Low damage on the section farther from the clamped end results in lower displacement values and a more difficult discernment of the actual D .

To complete the simulation, we use the `ActionUpdateComponent` from the core package. This component applies the chosen action's transition table to the discretized physical state, selecting the updated state according to the resulting probability distribution.

At this point, we construct the decision-making component of the `DT`. We first construct a reward function that evaluates the profit associated to the damage level of the beam, and establish a cost for each maintenance action. A time slice where the beam is undamaged beam gives a profit of 1, while the loss due to the level of damage d , expressed in percentage points, is computed as $e^{0.07d}$. The "do nothing" action has a profit of 10, while the "minor repair" and the "perfect repair" cost 30 and 100, respectively. We construct a `TabularPolicy` via Q-learning, performing 1 million iterations divided in episodes of 100 time slices. The resulting policy is represented in Figure 8.8.

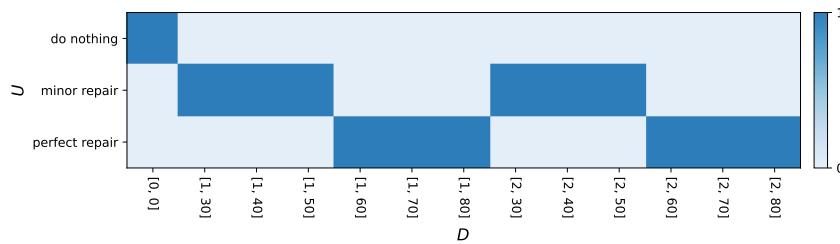


Figure 8.8: Tabular policy for the SHM, obtained via Q-learning with the stochastic `ActionUpdateComponent`. Darker shades of blue represent higher probability of choice.

We can now simulate the complete `DT` system, evaluating the accuracy in the estimation of the digital state and the resulting choice of actions. As a baseline, we implement a `DT` where the assimilation relies only on the confusion matrix of the inverse problem. We simulate $T = 100$ time slices, starting from the undamaged state, and corrupt the sensors with the same Gaussian noise used previously. The results are shown in Figure 8.9.

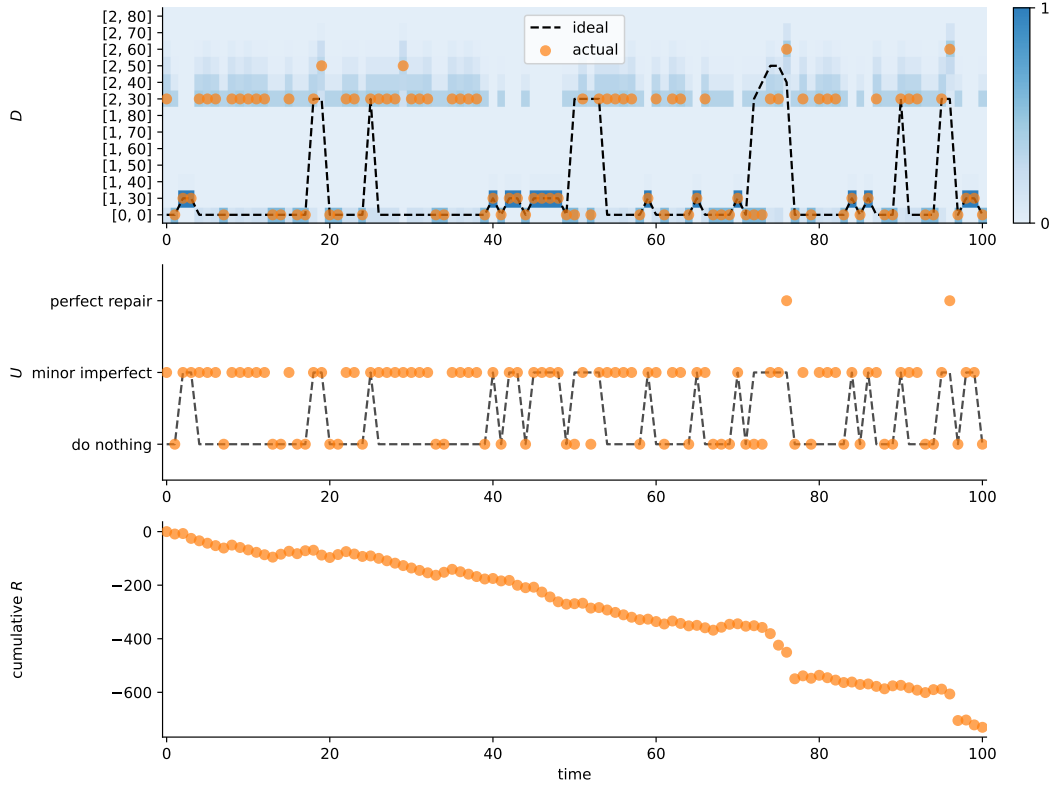


Figure 8.9: Simulation of the baseline SHM system, using sensors corrupted by Gaussian noise, stochastic physical asset update, and tabular policy. The top plot represents the estimation of the digital state, the plot in the middle compares the action selected and the one that would be chosen with perfect information, while the bottom plot shows the evolution of the cumulative reward. Darker shades of blue represent higher confidence associated to the assimilated state.

As a performance measure, we define the assimilation accuracy as the percentage of the number of slices in which the physical state matches the digital state with the highest probability. The choice of action is assessed in a similar way, with the corresponding accuracy computed as the percentage of slices in which the evaluation of the policy to the discretized physical state match the one on the estimated digital state. The assimilation reaches an accuracy of 53.47% and the correct choice of action is achieved 56.44% of the time. The assimilation tends to overestimate the damage level, often confounding the undamaged state with the lower level of damage in the section farther from the clamped end. This phenomenon results in overuse of the "minor repair" action and consequent decrease of the cumulative reward, which in this simulation reaches -730.61.

For the PGM-based DT, we use the DBN from Figure 8.2. Since we only consider the estimation from the inverse problem, the digital state, and the actions, the actual DBN used by the DT is represented in Figure 8.10.

The CPDs for the DBN are obtained from the actions' transition matrices, the confusion matrix of the inverse problem, and the tabular policy. In particular, we combine the actions' matrices and the confusion matrix by assuming independence, so that the DBN

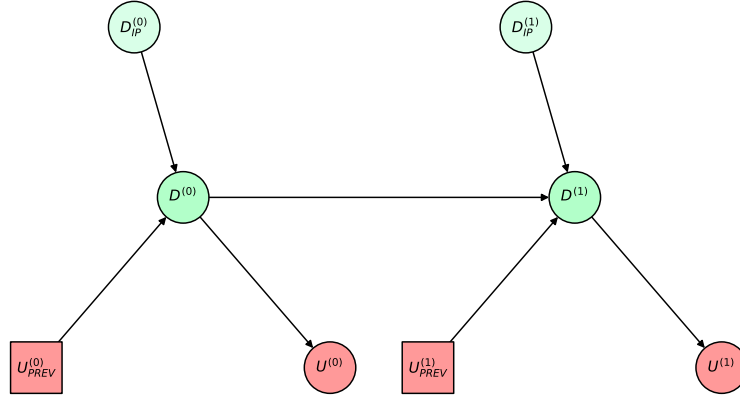


Figure 8.10: DBN for the assimilation. The evidence nodes $D_{IP}^{(1)}$, $D^{(0)}$, and $U_{PREV}^{(1)}$ render the rest of the full DBN superfluous for the inference of $D^{(1)}$.

for $P(D^{(1)} \mid D_{IP}^{(1)}, D^{(0)}, U_{PREV}^{(1)})$ is a large matrix with 13 rows and $13 \times 13 \times 3$ columns. We remark that the creation of the large, dense matrix could be bypassed by a suitable implementation of the `CPD` class for `pgmpy`, but the library does not support such interface at the moment.

To determine the inference results, `pgmtwin` can perform either a random sampling of the $D_{IP}^{(1)}$ and $D^{(0)}$ distributions, or use a weighted sum of the values with highest probability. In this case, we use the weighted sum approach to maintain a low computational cost and select the two most probable evidences. We repeat the simulation with the new `DT` and presents the results in Figure 8.11.

Thanks to the `DBN`, the assimilation reaches a much higher accuracy of 80.20%, while the share of correct choice of action achieves 82.18%. The final value of the cumulative reward is -267.38, a significant improvement on the previous case. We remark that the first action chosen by the `DT` is an unjustified "perfect repair", which determines a large decrease of the reward. This is caused by the absence of information on the initial state, with the `DT` using a uniform distribution of the digital state as prior information.

Repeating the simulations on a larger time window of 10000 slices, the differences between the two approaches are maintained, but reduced. The baseline approaches increases its assimilation accuracy to 59.33% and the correct choice of action rate to 61.75%, while the `PGM`-based `DT` achieves 72.52% and 78.31%, respectively. The cumulative rewards are -41657.68 and -34203.40, respectively, confirming the superiority of the `PGM`-based approach. Table 8.1 collects the results of the experiments.

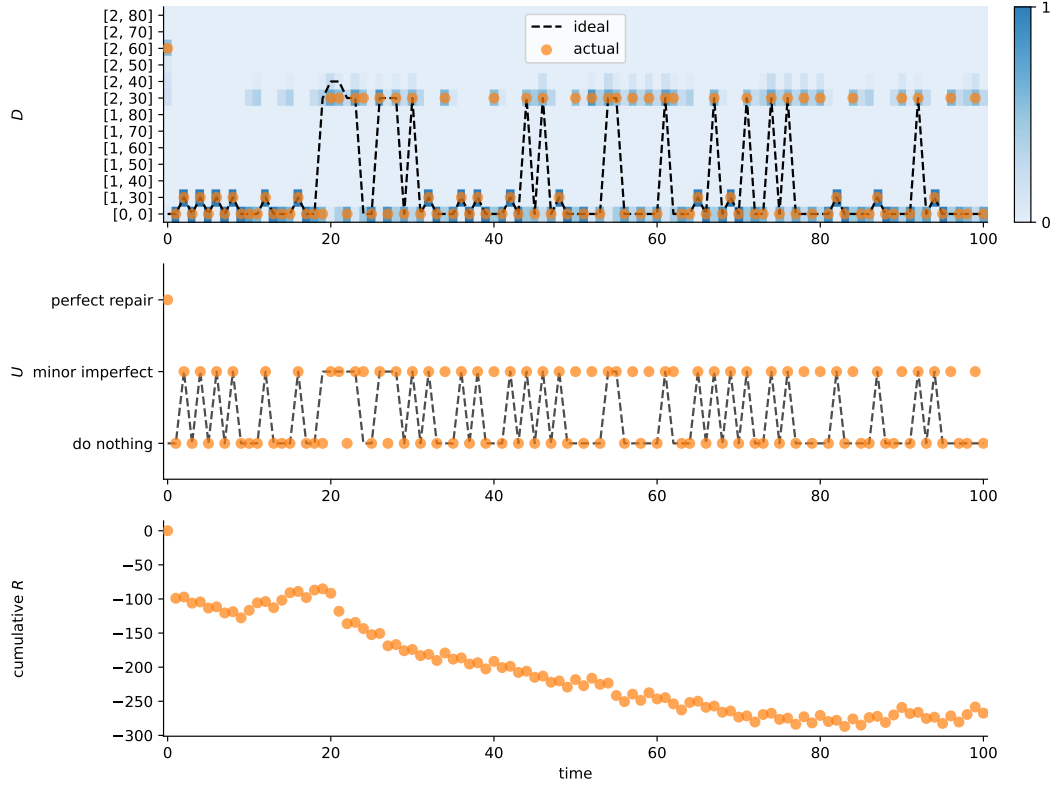


Figure 8.11: Simulation of the SHM PGM-based system, using 2 samples for PGM inference, Gaussian sensor noise, stochastic physical asset update, and tabular policy. The top plot represents the estimation of the digital state, the plot in the middle compares the action selected and the one that would be chosen with perfect information, while the bottom plot shows the evolution of the cumulative reward. Darker shades of blue represent higher confidence associated to the assimilated state.

Table 8.1: Comparison of the baseline DT and the PGM-based version, for different numbers of total time slices.

	$T = 100$			$T = 10000$		
	Assimilation	Action	Reward	Assimilation	Action	Reward
baseline	53.47%	56.44%	-730.61	59.33%	61.75%	-41657.68
PGM	80.20%	82.18%	-267.38	72.52%	78.31%	-34203.40

8.5 Conclusions

This chapter introduced `pgmtwin`, a new open-source Python library for the implementation and analysis of [DT](#) systems based on [PGMs](#).

The component-based architecture of the library determines a streamlined workflow in the construction of a new case, with clear interfaces for the customization by the user's business logic. First, the discretization of the states and the possible actions constitute the base elements for the [DT](#) system. The user implements the components that customize the physical asset update and observation, as well as the inverse problem, with total independence from the [DT](#) loop. The decision-making component uses the widely used interfaces from the `Gymnasium` and `Stable-Baselines3`, making `pgmtwin` interoperable with the [RL](#) ecosystem.

Specific implementations are provided in the `toolkits` module, such as the `shm` module demonstrated in this chapter. The complete implementation of a [DT](#) for the structural health monitoring of a cantilever beam was discussed and evaluated, following the workflow of `pgmtwin`. The results show a significant increase in performance for the [PGM](#)-based approach compared to a naive implementation, achieving a much more accurate assimilation of the simulated sensor data and higher cumulative reward over short and long time frames.

`pgmtwin` is a first step to enable the adoption of [DTs](#) in the industrial design process, thanks to its versatility and the ease of customization.

CONCLUSIONS

In this thesis we have developed a pipeline for the structural optimization of cruise ships in the preliminary design phase, reducing the total mass of the vessel while reducing the number of structural failures. The pipeline leverages data-driven [model order reduction \(MOR\)](#) and metaheuristic optimization algorithms, and provides a procedure for the refinement of the optimization problem. We extended the pipeline through the application of [transfer learning \(TL\)](#), obtaining a one-shot parameterization refinement procedure and an effective population transfer to initialize the metaheuristics. All components of the pipeline have been evaluated on the optimization of a family of simplified midship sections, and a group of full-scale commercial ships. Finally, we introduced `pgmtwin`, an open source Python library to streamline the implementation of [digital twins \(DTs\)](#).

More specifically, the contributions of this thesis are summarized as follows. We have comprehensively assessed the accuracy of our proposed surrogates, based on [proper orthogonal decomposition \(POD\)](#) and [Gaussian process regression \(GPR\)](#), followed by non-linear post-processing, for multiple combinations of [POD](#) truncation rank and sizes of the high-fidelity database. The proposed surrogates provide good predictions of the stress tensor fields, the number of yielded and buckled elements, and the vertical deflection, and are robust to the increase of samples as the optimization extends the high-fidelity database.

For the optimization of the surrogates, we selected [non-dominated sorting genetic algorithm III \(NSGA-III\)](#) and [particle swarm optimization \(PSO\)](#). We proposed a new infill criterion for the approximation of the [Pareto front \(PF\)](#), and enhanced the single-objective [PSO](#) and [GPR](#)-based [Bayesian optimization \(BO\)](#) on linearly constrained discrete domains, using a repair procedure based on an [integer linear program \(ILP\)](#). [PSO](#) was further modified with hybridization with [principal directions search \(PDS\)](#), outperforming other black-box approaches and established frameworks on a full-scale test case.

To overcome the complexity in defining an effective parameterization of the optimization problem, we developed a data-driven parameterization refinement procedure based on a series of [ILPs](#) and the predictions from our surrogates. The iterative refinement was able to outperform a high-parameter formulation from expert designers on a full-scale

ship, showing that the adaptation of the problem to the emerging structural behavior is an effective strategy.

In order to reduce the number of high-fidelity [finite element analysis \(FEA\)](#) simulations of the pipeline, we employed [TL](#) to leverage the results from past optimizations. We used the domain adaptation approach to formulate a one-shot parameterization refinement, lifting the requirement for surrogates and a large high-fidelity database. Evaluated on a family of midship sections and full-scale ships, the procedure was able to achieve good trade-off in the number of [FEA](#) simulations and the final mass. For the enhancement of multi- and single-objective optimization, we proposed a population transfer procedure based on [GPR](#) to accelerate the exploration of unseen domains. The transfer of the [PFs](#) and the single-objective optimization histories from past ships was shown to greatly accelerate the convergence to the ideal [PF](#) and the optimum on unseen ships, both for the midship sections and full-scale vessels.

Finally, we introduced `pgmtwin` to streamline the implementation and analysis of [DTs](#) based on [probabilistic graphical models \(PGMs\)](#). We showcased the workflow for the implementation of a [DT](#) for the [structural health monitoring \(SHM\)](#) of a simplified cantilever beam, where the use of [PGM](#) for the data assimilation outperformed an approach that only evaluated the accuracy of the inverse problem.

9.1 Future perspectives

We identify multiple directions for the extension of our work.

Thanks to the data-driven formulation of the [reduced order models \(ROMs\)](#) and the optimization algorithms used, the pipeline can be used in other structural optimization applications with minimal changes. For example, the optimization of other ship classes would only require the adaptation of the buckling criteria, or in the case of floating off shore wind turbines, the load conditions that model the wind loads would only increase the number of [ROMs](#) constructed. A further extension of the pipeline capabilities would be the inclusion of different steel grades and the sizing of the secondary stiffeners.

Moving towards the holistic design approach [139] envisioned by the industry and academia, other aspects of the vessel could be integrated and optimized jointly. The propulsion system, the [computational fluid dynamics \(CFD\)](#) properties of the hull, and the seakeeping qualities can be evaluated from a unified parameter set, determining an increase of decision variables, constraints, and objective functions to consider during multi-objective optimization. The integration of life-cycle [quantities of interest \(QoIs\)](#) of the ship, such as fatigue, [SHM](#), fuel consumption, and carbon emissions evaluated on historical mission data and synthetic scenarios thanks to a [DT](#) approach, could contribute as well to the holistic optimization.

Other components of the pipeline can be generalized to different contexts. The response-based parameterization refinement approach can be adapted to other applications where the contributions to the objective function can be evaluated at the lowest level of

parameterization. The population transfer methodology, developed for the thicknesses of similar structural elements, can be translated to the transfer of other fields defined on computational meshes, under the assumption of a link between the correlation of the [QoIs](#) and the similarity of the mesh features.

BIBLIOGRAPHY

- [1] T. Achterberg and R. Wunderling. “Mixed Integer Programming: Analyzing 12 Years of Progress”. In: *Facets of Combinatorial Optimization*. Ed. by M. Jünger and G. Reinelt. Berlin, Heidelberg: Springer, 2013, pp. 449–481. ISBN: 978-3-642-38188-1. DOI: 10.1007/978-3-642-38189-8_18 (cit. on p. 42).
- [2] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama. “Optuna: A next-generation hyperparameter optimization framework”. In: *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery and data mining*. SIGKDD’19. 2019, pp. 2623–2631. DOI: 10.1145/3292500.3330701 (cit. on pp. 2, 41, 57).
- [3] M. S. Alnæs, A. Logg, K. B. Ølgaard, M. E. Rognes, and G. N. Wells. “Unified form language: A domain-specific language for weak formulations of partial differential equations”. In: *ACM Transactions on Mathematical Software* 40.2 (2014), pp. 1–37. ISSN: 0098-3500, 1557-7295. DOI: 10.1145/2566630 (cit. on p. 123).
- [4] D. Amsallem, M. Zahr, Y. Choi, and C. Farhat. “Design optimization using hyper-reduced-order models”. In: *Structural and Multidisciplinary Optimization* 51.4 (2015), pp. 919–940. ISSN: 1615-147X, 1615-1488. DOI: 10.1007/s00158-014-1183-y (cit. on p. 40).
- [5] J. Andric, P. Prebeg, and V. Zanic. “Multi-level Pareto supported design methodology-application to RO-PAX structural design”. In: *Marine Structures* 67 (2019), p. 102638. ISSN: 09518339. DOI: 10.1016/j.marstruc.2019.102638 (cit. on p. 5).
- [6] A. Ankan and A. Panda. “pgmpy: Probabilistic Graphical Models using Python”. In: *Proceedings of the 14th Python in Science Conference*. SciPy. Austin, Texas: SciPy, 2015, pp. 6–11. DOI: 10.25080/Majora-7b98e3ed-001 (cit. on pp. 117, 120).
- [7] A. Ankan and J. Textor. “pgmpy: a Python toolkit for Bayesian networks”. In: *Journal of Machine Learning Research* 25.1 (2024), pp. 1–8. ISSN: 1532-4435 (cit. on p. 117).
- [8] ASTM International Committee A01 on Steel, Stainless Steel and Related Alloys. *Specification for Structural Steel for Ships*. DOI: 10.1520/A0131_A0131M-19 (cit. on p. 9).

- [9] O. B. Augusto and A. Kawano. "A mixed continuous and discrete nonlinear constrained algorithm for optimizing ship hull structural design". In: *Ocean Engineering* 25.9 (1998), pp. 793–811. ISSN: 00298018. DOI: 10.1016/S0029-8018(97)10011-7 (cit. on pp. 5, 41).
- [10] T. Back, U. Hammel, and H.-P. Schwefel. "Evolutionary computation: comments on the history and current state". In: *IEEE Transactions on Evolutionary Computation* 1.1 (1997), pp. 3–17. ISSN: 1089778X. DOI: 10.1109/4235.585888 (cit. on pp. 41, 58).
- [11] R. Baptista and M. Poloczek. *Bayesian Optimization of Combinatorial Structures*. 2018. DOI: 10.48550/arXiv.1806.08838 (cit. on pp. 25, 52).
- [12] I. A. Baratta, J. P. Dean, J. S. Dokken, M. Habera, J. S. Hale, C. N. Richardson, M. E. Rognes, M. W. Scroggs, N. Sime, and G. N. Wells. *DOLFINx: The next generation FEniCS problem solving environment*. 2023. DOI: 10.5281/ZENODO.10447666 (cit. on p. 123).
- [13] M. Barrault, Y. Maday, N. C. Nguyen, and A. T. Patera. "An 'empirical interpolation' method: application to efficient reduced-basis discretization of partial differential equations". In: *Comptes Rendus. Mathématique* 339.9 (2004), pp. 667–672. ISSN: 1631-073X, 1778-3569. DOI: 10.1016/j.crma.2004.08.006 (cit. on p. 22).
- [14] A. Bayatfar, J. Mishaël, R. Warnotte, and P. Rigo. "An integrated framework for ship structural optimisation in contract design phase". In: *Annals of "Dunărea de Jos" University of Galati, Fascicle XI, Shipbuilding* 42 (2019), pp. 109–116. ISSN: 12214620, 26683156. DOI: 10.35219/AnnUGalShipBuilding.2019.42.15 (cit. on p. 5).
- [15] E. M. L. Beale and J. J. H. Forrest. "Global optimization using special ordered sets". In: *Mathematical Programming* 10.1 (1976), pp. 52–69. ISSN: 0025-5610, 1436-4646. DOI: 10.1007/BF01580653 (cit. on p. 48).
- [16] S. Belakaria, A. Deshwal, and J. R. Doppa. "Max-value Entropy Search for Multi-Objective Bayesian Optimization". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Vol. 32. NeurIPS'19. Curran Associates, Inc., 2019 (cit. on p. 45).
- [17] I. Ben-Yelun, M. E. F. Idrissi, J. Mounayer, S. Rodriguez, and F. Chinesta. *Rank Reduction AutoEncoders for Mechanical Design: Advancing Novel and Efficient Data-Driven Topology Optimization*. 2026. DOI: 10.48550/arXiv.2601.23269 (cit. on p. 65).
- [18] M. A. Benaïmeche, J. Yvonnet, B. Bary, and Q.-C. He. "A k-means clustering machine learning-based multiscale method for anelastic heterogeneous structures with internal variables". In: *International Journal for Numerical Methods in Engineering* 123.9 (2022), pp. 2012–2041. ISSN: 0029-5981, 1097-0207. DOI: 10.1002/nme.6925 (cit. on p. 65).

- [19] Y. Bengio, A. Lodi, and A. Prouvost. “Machine learning for combinatorial optimization: A methodological tour d’horizon”. In: *European Journal of Operational Research* 290.2 (2021), pp. 405–421. ISSN: 03772217. DOI: 10.1016/j.ejor.2020.07.063 (cit. on p. 96).
- [20] P. Benner, S. Grivet-Talocia, A. Quarteroni, G. Rozza, W. Schilders, and L. M. Silveira, eds. *Model Order Reduction - 1 - System- and Data-Driven Methods and Algorithms*. De Gruyter, 2021. ISBN: 978-3-11-049896-7. DOI: 10.1515/9783110498967 (cit. on p. 21).
- [21] P. Benner, S. Grivet-Talocia, A. Quarteroni, G. Rozza, W. Schilders, and L. M. Silveira, eds. *Model Order Reduction - 2 - Snapshot-Based Methods and Algorithms*. De Gruyter, 2021. ISBN: 978-3-11-067149-0. DOI: 10.1515/9783110671490 (cit. on p. 21).
- [22] P. Benner, M. Ohlberger, A. Cohen, and K. Willcox, eds. *Model Reduction and Approximation: Theory and Algorithms*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2017. ISBN: 978-1-61197-481-2. DOI: 10.1137/1.9781611974829 (cit. on p. 21).
- [23] P. Bennet, C. Doerr, A. Moreau, J. Rapin, F. Teytaud, and O. Teytaud. “Nevergrad: black-box optimization platform”. In: *ACM SIGEVOlution* 14.1 (2021), pp. 8–15. DOI: 10.1145/3460310.3460312 (cit. on p. 57).
- [24] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl. “Algorithms for hyper-parameter optimization”. In: *Proceedings of the 25th International Conference on Neural Information Processing Systems*. NeurIPS’11. Red Hook, NY, USA: Curran Associates Inc., 2011, pp. 2546–2554. ISBN: 978-1-61839-599-3 (cit. on pp. 39, 40, 50, 52).
- [25] E. Bingham, J. P. Chen, M. Jankowiak, F. Obermeyer, N. Pradhan, T. Karaletsos, R. Singh, P. Szerlip, P. Horsfall, and N. D. Goodman. “Pyro: deep universal probabilistic programming”. In: *J. Mach. Learn. Res.* 20.1 (2019), pp. 973–978. ISSN: 1532-4435 (cit. on p. 117).
- [26] M. Binois and N. Wycoff. “A Survey on High-dimensional Gaussian Process Modeling with Application to Bayesian Optimization”. In: *ACM Transactions on Evolutionary Learning and Optimization* 2.2 (2022), pp. 1–26. ISSN: 2688-299X, 2688-3007. DOI: 10.1145/3545611 (cit. on p. 25).
- [27] E. G. Birgin and J. M. Martínez. *Practical Augmented Lagrangian Methods for Constrained Optimization*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2014. ISBN: 978-1-61197-335-8. DOI: 10.1137/1.9781611973365 (cit. on p. 40).
- [28] A. Bolufé-Röhler and D. Tamayo-Vera. “Machine Learning for Enhancing Meta-heuristics in Global Optimization: A Comprehensive Review”. In: *Mathematics* 13.18 (2025), p. 2909. ISSN: 2227-7390. DOI: 10.3390/math13182909 (cit. on p. 96).
- [29] I. Borg and P. J. F. Groenen. *Modern Multidimensional Scaling: Theory and Applications*. 2nd ed. Springer Series in Statistics. Dordrecht: Springer-Verlag New York Inc, 2007. ISBN: 978-0-387-25150-9 (cit. on p. 100).

- [30] S. L. Brunton and J. N. Kutz. *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. 2nd ed. Cambridge: Cambridge University Press, 2022. ISBN: 978-1-00-908951-7. DOI: 10.1017/9781009089517 (cit. on pp. 1, 21–24).
- [31] R. H. Byrd, M. E. Hribar, and J. Nocedal. “An Interior Point Algorithm for Large-Scale Nonlinear Programming”. In: *SIAM Journal on Optimization* 9.4 (1999), pp. 877–900. ISSN: 1052-6234, 1095-7189. DOI: 10.1137/S1052623497325107 (cit. on pp. 40, 57).
- [32] Y. Cai and J. Jelovica. “Adaptive Constraint Handling in Optimization of Complex Structures by Using Machine Learning”. In: *ASME 2021 40th International Conference on Ocean, Offshore and Arctic Engineering*. Virtual, Online: American Society of Mechanical Engineers, 2021, V002T02A037. ISBN: 978-0-7918-8512-3. DOI: 10.1115/OMAE2021-62304 (cit. on p. 5).
- [33] Y. Cai and J. Jelovica. “Structural Optimization of Ships: Benchmark Study of Metaheuristic Algorithms and Constraint Handling Approaches”. In: *ASME 2022 41st International Conference on Ocean, Offshore and Arctic Engineering*. Hamburg, Germany: American Society of Mechanical Engineers, 2022, V002T02A068. ISBN: 978-0-7918-8586-4. DOI: 10.1115/OMAE2022-79199 (cit. on pp. 5, 53).
- [34] L. Calvet, J. D. Armas, D. Masip, and A. A. Juan. “Learnheuristics: hybridizing metaheuristics with machine learning for optimization with dynamic inputs”. In: *Open Mathematics* 15.1 (2017), pp. 261–280. ISSN: 2391-5455. DOI: 10.1515/math-2017-0029 (cit. on p. 96).
- [35] E. F. Campana, M. Diez, G. Liuzzi, S. Lucidi, R. Pellegrini, V. Piccialli, F. Rinaldi, and A. Serani. “A multi-objective DIRECT algorithm for ship hull optimization”. In: *Computational Optimization and Applications* 71.1 (2018), pp. 53–72. ISSN: 0926-6003, 1573-2894. DOI: 10.1007/s10589-017-9955-0 (cit. on pp. 41, 42).
- [36] S. Chaturantabut and D. C. Sorensen. “Nonlinear Model Reduction via Discrete Empirical Interpolation”. In: *SIAM Journal on Scientific Computing* 32.5 (2010), pp. 2737–2764. ISSN: 1064-8275, 1095-7197. DOI: 10.1137/090766498 (cit. on p. 22).
- [37] T. Chen and C. Guestrin. “XGBoost: A Scalable Tree Boosting System”. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. SIGKDD’16. San Francisco, California, USA: Association for Computing Machinery, 2016, pp. 785–794. ISBN: 9781450342322. DOI: 10.1145/2939672.2939785 (cit. on pp. 84, 86, 124).
- [38] K. Cheng, Z. Lu, C. Ling, and S. Zhou. “Surrogate-assisted global sensitivity analysis: an overview”. In: *Structural and Multidisciplinary Optimization* 61.3 (2020), pp. 1187–1213. ISSN: 1615-147X, 1615-1488. DOI: 10.1007/s00158-019-02413-5 (cit. on p. 21).

-
- [39] F. Chinesta, A. Huerta, G. Rozza, and K. Willcox. "Model Reduction Methods". In: *Encyclopedia of Computational Mechanics, Second Edition*. New York: John Wiley & Sons, Ltd., 2017, pp. 1–36. doi: 10.1002/9781119176817.ecm2110 (cit. on pp. 1, 21).
 - [40] Y. Choi, G. Boncoraglio, S. Anderson, D. Amsallem, and C. Farhat. "Gradient-based constrained optimization using a database of linear reduced-order models". In: *Journal of Computational Physics* 423 (2020), p. 109787. issn: 00219991. doi: 10.1016/j.jcp.2020.109787 (cit. on p. 40).
 - [41] S. Chun, Y.-T. Kim, and T.-H. Kim. "A Diversity-Enhanced Constrained Particle Swarm Optimizer for Mixed Integer-Discrete-Continuous Engineering Design Problems". In: *Advances in Mechanical Engineering* 5 (2013), p. 130750. issn: 1687-8132, 1687-8140. doi: 10.1155/2013/130750 (cit. on p. 53).
 - [42] D. A. Cohn. "Neural Network Exploration Using Optimal Experiment Design". In: *Neural Networks* 9.6 (1996), pp. 1071–1083. issn: 08936080. doi: 10.1016/0893-6080(95)00137-9 (cit. on p. 45).
 - [43] A. R. Conn, K. Scheinberg, and L. N. Vicente. *Introduction to derivative-free optimization*. MPS-SIAM series on optimization 8. Philadelphia, Pa: SIAM, 2009. isbn: 978-0-89871-668-9. doi: 10.1137/1.9780898718768 (cit. on pp. 39–42).
 - [44] P. G. Constantine. *Active subspaces: emerging ideas for dimension reduction in parameter studies*. Siam spotlights 02. Philadelphia: siam, Society for Industrial and Applied Mathematics, 2015. isbn: 978-1-61197-385-3 (cit. on pp. 2, 27).
 - [45] P. G. Constantine, E. Dow, and Q. Wang. "Active Subspace Methods in Theory and Practice: Applications to Kriging Surfaces". In: *SIAM Journal on Scientific Computing* 36.4 (2014), A1500–A1524. issn: 1064-8275, 1095-7197. doi: 10.1137/130916138 (cit. on pp. 2, 27, 65).
 - [46] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms*. Fourth edition. Cambridge, Massachusetts London, England: The MIT Press, 2022. isbn: 978-0-262-04630-5 (cit. on p. 118).
 - [47] J. Cui and D. Wang. "Application of knowledge-based engineering in ship structural design and optimization". In: *Ocean Engineering* 72 (2013), pp. 124–139. issn: 00298018. doi: 10.1016/j.oceaneng.2013.06.013 (cit. on p. 5).
 - [48] S. Das and P. N. Suganthan. "Differential Evolution: A Survey of the State-of-the-Art". In: *IEEE Transactions on Evolutionary Computation* 15.1 (2011), pp. 4–31. issn: 1089-778X, 1941-0026. doi: 10.1109/TEVC.2010.2059031 (cit. on pp. 41, 58).
 - [49] S. Daulton, M. Balandat, and E. Bakshy. *Parallel Bayesian Optimization of Multiple Noisy Objectives with Expected Hypervolume Improvement*. 2021. doi: 10.48550/arXiv.2105.08195 (cit. on p. 42).

- [50] S. Daulton, D. Eriksson, M. Balandat, and E. Bakshy. “Multi-Objective Bayesian Optimization over High-Dimensional Search Spaces”. In: *Proceedings of the Thirty-Eighth Conference on Uncertainty in Artificial Intelligence*. PMLR. Proceedings of Machine Learning Research, 2022 (cit. on p. 42).
- [51] E. Daxberger, A. Makarova, M. Turchetta, and A. Krause. “Mixed-Variable Bayesian Optimization”. In: *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*. IJCAI’20. 2020, pp. 2633–2639. doi: 10.24963/ijcai.2020/365 (cit. on pp. 40, 52).
- [52] O. Day and T. M. Khoshgoftaar. “A survey on heterogeneous transfer learning”. In: *Journal of Big Data* 4.1 (2017), p. 29. issn: 2196-1115. doi: 10.1186/s40537-017-0089-0 (cit. on p. 96).
- [53] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. “A fast and elitist multiobjective genetic algorithm: NSGA-II”. In: *IEEE Transactions on Evolutionary Computation* 6.2 (2002), pp. 182–197. doi: 10.1109/4235.996017 (cit. on p. 42).
- [54] K. Deb and H. Jain. “An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point-Based Nondominated Sorting Approach, Part I: Solving Problems With Box Constraints”. In: *IEEE Transactions on Evolutionary Computation* 18.4 (2014), pp. 577–601. doi: 10.1109/TEVC.2013.2281535 (cit. on pp. 39, 42, 44).
- [55] N. Demo, M. Tezzele, A. Mola, and G. Rozza. “Hull Shape Design Optimization with Parameter Space and Model Reductions, and Self-Learning Mesh Morphing”. In: *Journal of Marine Science and Engineering* 9.2 (2021), p. 185. doi: 10.3390/jmse9020185 (cit. on p. 65).
- [56] N. Demo, M. Tezzele, and G. Rozza. “EZyRB: Easy Reduced Basis method”. In: *The Journal of Open Source Software* 3.24 (2018), p. 661. doi: 10.21105/joss.00661 (cit. on p. 29).
- [57] Det Norske Veritas. *Rules for Classification of Ships*. Online; accessed on 2026-08-06. 2017. url: <https://rules.dnv.com/servicedocuments/dnvpmp/> (cit. on pp. 1, 5, 6, 9).
- [58] M. Diez and D. Peri. “Two-stage Stochastic Programming Formulation for Ship Design Optimisation under Uncertainty”. In: *Ship Technology Research* 57.3 (2010), pp. 172–181. issn: 0937-7255, 2056-7111. doi: 10.1179/str.2010.57.3.003 (cit. on p. 116).
- [59] P. Díez, A. Muixí, S. Zlotnik, and A. García-González. “Nonlinear dimensionality reduction for parametric problems: A kernel proper orthogonal decomposition”. In: *International Journal for Numerical Methods in Engineering* 122.24 (2021), pp. 7306–7327. issn: 0029-5981, 1097-0207. doi: 10.1002/nme.6831 (cit. on p. 65).
- [60] M. Dorigo, M. Birattari, and T. Stutzle. “Ant colony optimization”. In: *IEEE Computational Intelligence Magazine* 1.4 (2006), pp. 28–39. issn: 1556-603X. doi: 10.1109/MCI.2006.329691 (cit. on p. 41).

-
- [61] M. Dorigo and C. Blum. “Ant colony optimization theory: A survey”. In: *Theoretical Computer Science* 344.2-3 (2005), pp. 243–278. issn: 03043975. doi: 10.1016/j.tcs.2005.05.020 (cit. on p. 41).
 - [62] J. Dréo, ed. *Metaheuristics for hard optimization: methods and case studies*. Berlin: Springer, 2006. isbn: 978-3-540-23022-9 (cit. on p. 41).
 - [63] D. Eriksson, M. Pearce, J. R. Gardner, R. Turner, and M. Poloczek. *Scalable Global Optimization via Local Bayesian Optimization*. 2020. doi: 10.48550/arXiv.1910.01739 (cit. on pp. 40, 42).
 - [64] R. Everson and L. Sirovich. “Karhunen–Loève procedure for gappy data”. In: *Journal of the Optical Society of America A* 12.8 (1995), p. 1657. issn: 1084-7529, 1520-8532. doi: 10.1364/JOSAA.12.001657 (cit. on p. 22).
 - [65] L. Fabris, M. Tezzele, C. Busiello, M. Sicchiero, and G. Rozza. “Data-Driven Parameterization Refinement for the Structural Optimization of Cruise Ship Hulls”. In: *International Journal for Numerical Methods in Engineering* 126.24 (2025), e70201. issn: 0029-5981, 1097-0207. doi: 10.1002/nme.70201 (cit. on p. 2).
 - [66] L. Fabris, M. Tezzele, and M. Torzoni. *pgmtwin*. <https://github.com/pgmtwin/pgmtwin>. 2026 (cit. on pp. 117, 122).
 - [67] L. Feng, Y.-S. Ong, S. Jiang, and A. Gupta. “Autoencoding Evolutionary Search With Learning Across Heterogeneous Problems”. In: *IEEE Transactions on Evolutionary Computation* 21.5 (2017), pp. 760–772. issn: 1089-778X. doi: 10.1109/TEVC.2017.2682274 (cit. on p. 96).
 - [68] L. Feng, Y.-S. Ong, A.-H. Tan, and I. W. Tsang. “Memes as building blocks: a case study on evolutionary optimization + transfer learning for routing problems”. In: *Memetic Computing* 7.3 (2015), pp. 159–180. issn: 1865-9284, 1865-9292. doi: 10.1007/s12293-015-0166-x (cit. on p. 82).
 - [69] L. Feng, L. Zhou, J. Zhong, A. Gupta, Y.-S. Ong, K.-C. Tan, and A. K. Qin. “Evolutionary Multitasking via Explicit Autoencoding”. In: *IEEE Transactions on Cybernetics* 49.9 (2019), pp. 3457–3470. issn: 2168-2267, 2168-2275. doi: 10.1109/TCYB.2018.2845361 (cit. on p. 96).
 - [70] M. Fischetti, F. Glover, and A. Lodi. “The feasibility pump”. In: *Mathematical Programming* 104.1 (2005), pp. 91–104. issn: 0025-5610, 1436-4646. doi: 10.1007/s10107-004-0570-3 (cit. on p. 48).
 - [71] C. M. Fonseca and P. J. Fleming. “An Overview of Evolutionary Algorithms in Multiobjective Optimization”. In: *Evolutionary Computation* 3.1 (1995), pp. 1–16. doi: 10.1162/evco.1995.3.1.1 (cit. on p. 42).

- [72] J. Forrest, T. Ralphs, S. Vigerske, H. G. Santos, J. Forrest, L. Hafer, B. Kristjansson, E. Straver, M. Lubin, J. Willem, S. Brito, M. Saltzman, B. Pitrus, and F. Matsushima. *coin-or/Cbc: Release releases/2.10.11*. 2023. DOI: 10.5281/ZENODO.10041724 (cit. on pp. 48, 57).
- [73] A. I. J. Forrester, A. Sóbester, and A. J. Keane. *Engineering Design via Surrogate Modelling: A Practical Guide*. 1st ed. Wiley, 2008. ISBN: 978-0-470-06068-1. DOI: 10.1002/9780470770801 (cit. on pp. 1, 21, 26).
- [74] D. Freitas, L. G. Lopes, and F. Morgado-Dias. “Particle Swarm Optimisation: A Historical Review Up to the Current Developments”. In: *Entropy* 22.3 (2020), p. 362. ISSN: 1099-4300. DOI: 10.3390/e22030362 (cit. on pp. 2, 52).
- [75] J. H. Friedman. “Greedy Function Approximation: A Gradient Boosting Machine”. In: *The Annals of Statistics* 29.5 (2001), pp. 1189–1232. ISSN: 0090-5364. DOI: 10.1214/aos/1013203451 (cit. on p. 84).
- [76] A. G. Gad. “Particle Swarm Optimization Algorithm and Its Applications: A Systematic Review”. In: *Archives of Computational Methods in Engineering* 29.5 (2022), pp. 2531–2561. ISSN: 1134-3060, 1886-1784. DOI: 10.1007/s11831-021-09694-4 (cit. on p. 52).
- [77] E. Gamma, R. Helm, R. E. Johnson, and J. Vlissides. *Design patterns: elements of reusable object-oriented software*. Reading, MA: Addison-Wesley, 2009. ISBN: 978-0-321-70069-8 (cit. on p. 120).
- [78] R. Garnett, M. A. Osborne, and S. J. Roberts. “Bayesian optimization for sensor set selection”. In: *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*. Stockholm Sweden: ACM, 2010, pp. 209–219. ISBN: 978-1-60558-988-6. DOI: 10.1145/1791212.1791238 (cit. on p. 99).
- [79] E. C. Garrido-Merchán and D. Hernández-Lobato. “Dealing with Categorical and Integer-valued Variables in Bayesian Optimization with Gaussian Processes”. In: *Neurocomputing* 380 (2020), pp. 20–35. ISSN: 09252312. DOI: 10.1016/j.neucom.2019.11.004 (cit. on p. 52).
- [80] T. Gärtner, P. A. Flach, A. Kowalczyk, and A. J. Smola. “Multi-instance kernels”. In: *Proceedings of the Nineteenth International Conference on Machine Learning*. Vol. 2. ICML’02 3. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2002, pp. 179–186. ISBN: 1-55860-873-7 (cit. on p. 99).
- [81] H. M. Gaspar, Í. A. Fonseca, L. Sepållä, H. Koelman, and J. J. G. Agis. “Can European Shipyards Be Smarter? A Proposal from the SEUS Project”. In: *Transport Transitions: Advancing Sustainable and Inclusive Mobility*. Ed. by C. McNally, P. Carroll, B. Martinez-Pastor, B. Ghosh, M. Efthymiou, and N. Valantasis-Kanellos. Cham: Springer Nature Switzerland, 2026, pp. 54–60. ISBN: 978-3-032-06762-3. DOI: 10.1007/978-3-032-06763-0_8 (cit. on pp. 1, 115, 116).

-
- [82] M. B. Giles and N. A. Pierce. “An Introduction to the Adjoint Approach to Design”. In: *Flow, Turbulence and Combustion* 65.3/4 (2000), pp. 393–415. issn: 13866184. doi: 10.1023/A:1011430410075 (cit. on p. 40).
 - [83] P. E. Gill, W. Murray, and M. A. Saunders. “SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization”. In: *SIAM Review* 47.1 (2005), pp. 99–131. issn: 0036-1445, 1095-7200. doi: 10.1137/S0036144504446096 (cit. on p. 40).
 - [84] GPy. *GPy: A Gaussian process framework in python*. <https://github.com/SheffieldML/GPy>. 2012 (cit. on pp. 29, 40, 57).
 - [85] R. B. Gramacy. *Surrogates - Gaussian Process Modeling, Design, and Optimization for the Applied Sciences*. Boca Raton: Chapman & Hall/CRC, 2021. isbn: 978-1-00-076652-3 (cit. on pp. 2, 21, 22, 25, 39, 40, 50).
 - [86] A. P. Guerreiro, C. M. Fonseca, and L. Paquete. “The Hypervolume Indicator: Problems and Algorithms”. In: *ACM Computing Surveys* 54.6 (2022), pp. 1–42. issn: 0360-0300, 1557-7341. doi: 10.1145/3453474 (cit. on pp. 42, 106).
 - [87] A. Gupta, Y.-S. Ong, and L. Feng. “Insights on Transfer Optimization: Because Experience is the Best Teacher”. In: *IEEE Transactions on Emerging Topics in Computational Intelligence* 2.1 (2018), pp. 51–64. issn: 2471-285X. doi: 10.1109/TETCI.2017.2769104 (cit. on pp. 2, 96).
 - [88] A. Gupta, Y.-S. Ong, and L. Feng. “Multifactorial Evolution: Toward Evolutionary Multitasking”. In: *IEEE Transactions on Evolutionary Computation* 20.3 (2016), pp. 343–357. issn: 1089-778X, 1941-0026. doi: 10.1109/TEVC.2015.2458037 (cit. on pp. 82, 96).
 - [89] S. Henao-Garcia, M. Kapteyn, and K. E. Willcox. “Federated Digital Twins for Space Systems”. In: *AIAA SCITECH 2026 Forum*. Orlando, FL: American Institute of Aeronautics and Astronautics, 2026. isbn: 978-1-62410-765-8. doi: 10.2514/6.2026-1420 (cit. on p. 119).
 - [90] S. Henao-Garcia, M. Kapteyn, K. E. Willcox, M. Tezzele, M. Castroviejo-Fernandez, T. Kim, M. Ambrosino, I. Kolmanovsky, H. Basu, P. Jirwankar, and R. Sanfelice. “Digital-Twin-Enabled Multi-Spacecraft On-Orbit Operations”. In: *AIAA SCITECH 2025 Forum*. Orlando, FL: American Institute of Aeronautics and Astronautics, 2025. isbn: 978-1-62410-723-8. doi: 10.2514/6.2025-1432 (cit. on p. 119).
 - [91] D. Hernandez-Lobato, J. Hernandez-Lobato, A. Shah, and R. Adams. “Predictive Entropy Search for Multi-objective Bayesian Optimization”. In: *Proceedings of The 33rd International Conference on Machine Learning*. Ed. by M. F. Balcan and K. Q. Weinberger. Vol. 48. ICML’16. New York, New York, USA: JMLR.org, 2016, pp. 1492–1501 (cit. on p. 45).

- [92] J. S. Hesthaven, G. Rozza, and B. Stamm. *Certified Reduced Basis Methods for Parametrized Partial Differential Equations*. SpringerBriefs in Mathematics. Cham: Springer International Publishing, 2016. ISBN: 978-3-319-22469-5. DOI: 10.1007/978-3-319-22470-1 (cit. on pp. 22, 24).
- [93] S. Horne. “MSC/NASTRAN”. In: *Finite Element Systems*. Ed. by C. A. Brebbia. Berlin, Heidelberg: Springer, 1982, pp. 287–294. ISBN: 978-3-662-07231-8. DOI: 10.1007/978-3-662-07229-5_19 (cit. on p. 6).
- [94] L. Huang, U. Topcu, L. R. Varshney, and K. E. Willcox. “Formal Verification of Digital Twins with the Temporal Logic of Actions”. In: *Journal of Computational Physics* 565 (2026), p. 115196. ISSN: 00219991. DOI: 10.1016/j.jcp.2026.115196 (cit. on p. 120).
- [95] F. Hutter, H. H. Hoos, and K. Leyton-Brown. “Sequential Model-Based Optimization for General Algorithm Configuration”. In: *Learning and Intelligent Optimization*. Ed. by C. A. C. Coello. Vol. 6683. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 507–523. ISBN: 978-3-642-25565-6. DOI: 10.1007/978-3-642-25566-3_40 (cit. on pp. 40, 50, 52).
- [96] C. Hvarfner, E. O. Hellsten, and L. Nardi. “Vanilla Bayesian Optimization Performs Great in High Dimensions”. In: *Proceedings of the 41st International Conference on Machine Learning*. Ed. by R. Salakhutdinov, Z. Kolter, K. Heller, A. Weller, N. Oliver, J. Scarlett, and F. Berkenkamp. Vol. 235. ICML’24. Vienna, Austria: JMLR.org, 2024, pp. 20793–20817 (cit. on pp. 25, 40).
- [97] International Maritime Organization. *Marine Environment Protection Committee – 79th session*. Online; accessed on 2026-08-06. 2022. URL: <https://www.imo.org/en/mediacentre/meetingsummaries/pages/mepc-79th-session.aspx> (cit. on p. 1).
- [98] A. Ivaldi, A. Bayatfar, J.-D. Caprace, G. Egorov, S. E. Heggelund, S. Hirakawa, J. M. Kwon, D. Mcgreer, P. Prebeg, R. Sielski, M. Slagmolen, A. Sobey, W. Tang, and J. Wu. “Committee IV.2: Design Methods”. In: *Proceedings of the 21st International Ship and Offshore Structures Congress*. Vol. 1. ISSC’22. Vancouver, Canada: The Society of Naval Architects and Marine Engineers, 2022, D011S001R009. DOI: 10.5957/ISSC-2022-COMMITTEE-IV-2 (cit. on pp. 1, 115, 116).
- [99] D. R. Jones, C. D. Perttunen, and B. E. Stuckman. “Lipschitzian optimization without the Lipschitz constant”. In: *Journal of Optimization Theory and Applications* 79.1 (1993), pp. 157–181. ISSN: 0022-3239, 1573-2878. DOI: 10.1007/BF00941892 (cit. on p. 41).
- [100] D. R. Jones, M. Schonlau, and W. J. Welch. “Efficient Global Optimization of Expensive Black-Box Functions”. In: *Journal of Global optimization* 13 (1998), pp. 455–492. DOI: 10.1023/A:1008306431147 (cit. on p. 51).

-
- [101] T.-H. Joung, S.-G. Kang, J.-K. Lee, and J. Ahn. "The IMO initial strategy for reducing Greenhouse Gas(GHG) emissions, and its follow-up actions towards 2050". In: *Journal of International Maritime Safety, Environmental Affairs, and Shipping* 4.1 (2020), pp. 1–7. ISSN: 2572-5084. DOI: 10.1080/25725084.2019.1707938 (cit. on p. 1).
 - [102] M. G. Kapteyn, A. Chaudhuri, E. A. B. F. Lima, G. Pash, R. Bravo, K. E. Willcox, T. E. Yankeelov, and D. A. Hormuth II. "TumorTwin: a Python framework for patient-specific digital twins in oncology". In: *BMC Medical Informatics and Decision Making* 26.1 (2026), p. 237. ISSN: 1472-6947. DOI: 10.1186/s12911-026-03520-2 (cit. on p. 116).
 - [103] M. G. Kapteyn, J. V. R. Pretorius, and K. E. Willcox. "A probabilistic graphical model foundation for enabling predictive digital twins at scale". In: *Nature Computational Science* 1.5 (2021), pp. 337–347. ISSN: 2662-8457. DOI: 10.1038/s43588-021-00069-0 (cit. on pp. 2, 115, 117).
 - [104] M. Karimi-Mamaghan, M. Mohammadi, P. Meyer, A. M. Karimi-Mamaghan, and E.-G. Talbi. "Machine learning at the service of meta-heuristics for solving combinatorial optimization problems: A state-of-the-art". In: *European Journal of Operational Research* 296.2 (2022), pp. 393–422. ISSN: 03772217. DOI: 10.1016/j.ejor.2021.04.032 (cit. on p. 96).
 - [105] J. Kennedy and R. Eberhart. "Particle swarm optimization". In: *Proceedings of the International Conference on Neural Networks*. Vol. 4. ICNN'95. Perth, WA, Australia: IEEE, 1995, pp. 1942–1948. ISBN: 978-0-7803-2768-9. DOI: 10.1109/ICNN.1995.488968 (cit. on pp. 2, 39).
 - [106] A. I. Khuri and S. Mukhopadhyay. "Response surface methodology". In: *WIREs Computational Statistics* 2.2 (2010), pp. 128–149. ISSN: 1939-5108, 1939-0068. DOI: 10.1002/wics.73 (cit. on p. 22).
 - [107] J. Kim and M. McCourt. "Bayesian Optimization over Sets". In: *6th ICML Workshop on Automated Machine Learning*. ICML'19. 2019 (cit. on pp. 25, 99).
 - [108] J. Kim, M. McCourt, T. You, S. Kim, and S. Choi. *Bayesian Optimization with Approximate Set Kernels*. 2021. DOI: 10.48550/arXiv.1905.09780 (cit. on p. 99).
 - [109] Y. Kim, Y. Choi, and B. Yoo. "Gappy AE: A nonlinear approach for Gappy data reconstruction using auto-encoder". In: *Computer Methods in Applied Mechanics and Engineering* 426 (2024), p. 116978. ISSN: 00457825. DOI: 10.1016/j.cma.2024.116978 (cit. on p. 65).
 - [110] D. P. Kingma and J. Ba. *Adam: A Method for Stochastic Optimization*. 2017. DOI: 10.48550/arXiv.1412.6980 (cit. on p. 40).
 - [111] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. "Optimization by Simulated Annealing". In: *Science* 220.4598 (1983), pp. 671–680. ISSN: 0036-8075. DOI: 10.1126/science.220.4598.671 (cit. on p. 41).

- [112] J. Knowles. “ParEGO: a hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems”. In: *IEEE Transactions on Evolutionary Computation* 10.1 (2006), pp. 50–66. DOI: 10.1109/TEVC.2005.851274 (cit. on p. 42).
- [113] M. J. Kochenderfer and T. A. Wheeler. *Algorithms for Optimization*. The MIT Press, 2019. ISBN: 978-0-262-03942-0 (cit. on pp. 25, 39, 41, 42).
- [114] D. Koller and N. Friedman. *Probabilistic graphical models: principles and techniques*. Adaptive computation and machine learning. Cambridge, Mass.: MIT Press, 2010. ISBN: 978-0-262-01319-2 (cit. on pp. 2, 115, 117, 118).
- [115] Z. Kunkera, T. Opetuk, N. Hadžić, and N. Tošanović. “Using Digital Twin in a Shipbuilding Project”. In: *Applied Sciences* 12.24 (2022), p. 12721. ISSN: 2076-3417. DOI: 10.3390/app122412721 (cit. on p. 116).
- [116] J. Lampinen. “A constraint handling approach for the differential evolution algorithm”. In: *Proceedings of the 2002 Congress on Evolutionary Computation*. Vol. 2. CEC’02. Honolulu, HI, USA: IEEE, 2002, pp. 1468–1473. ISBN: 978-0-7803-7282-5. DOI: 10.1109/CEC.2002.1004459 (cit. on p. 39).
- [117] M. Li, S. Yang, and X. Liu. “A Performance Comparison Indicator for Pareto Front Approximations in Many-Objective Optimization”. In: *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*. GECCO’15. Madrid, Spain: ACM, 2015, pp. 703–710. ISBN: 978-1-4503-3472-3. DOI: 10.1145/2739480.2754687 (cit. on p. 106).
- [118] S. Louis and J. McDonnell. “Learning With Case-Injected Genetic Algorithms”. In: *IEEE Transactions on Evolutionary Computation* 8.4 (2004), pp. 316–328. ISSN: 1089-778X. DOI: 10.1109/TEVC.2004.823466 (cit. on pp. 96, 105).
- [119] Y. Lu, X. Xue, K. Zhang, L. Zhang, G. Chen, C. Cao, P. Liu, and K. C. Tan. *Multi-Task Surrogate-Assisted Search with Bayesian Competitive Knowledge Transfer for Expensive Optimization*. 2025. DOI: 10.48550/arXiv.2510.23407 (cit. on p. 96).
- [120] T. W. Lukaczyk, P. Constantine, F. Palacios, and J. J. Alonso. “Active Subspaces for Shape Optimization”. In: *10th AIAA Multidisciplinary Design Optimization Conference*. National Harbor, Maryland: American Institute of Aeronautics and Astronautics, 2014. ISBN: 978-1-62410-310-0. DOI: 10.2514/6.2014-1171 (cit. on p. 65).
- [121] P. Luszczek, W. M. Sid-Lakhdar, and J. Dongarra. “Combining multitask and transfer learning with deep Gaussian processes for autotuning-based performance engineering”. In: *The International Journal of High Performance Computing Applications* 37.3-4 (2023), pp. 229–244. ISSN: 1094-3420, 1741-2846. DOI: 10.1177/10943420231166365 (cit. on p. 96).
- [122] D. J. C. MacKay. “Information-Based Objective Functions for Active Data Selection”. In: *Neural Computation* 4.4 (1992), pp. 590–604. ISSN: 0899-7667, 1530-888X. DOI: 10.1162/neco.1992.4.4.590 (cit. on p. 45).

- [123] N. S. Madusanka, Y. Fan, S. Yang, and X. Xiang. “Digital Twin in the Maritime Domain: A Review and Emerging Trends”. In: *Journal of Marine Science and Engineering* 11.5 (2023), p. 1021. ISSN: 2077-1312. DOI: 10.3390/jmse11051021 (cit. on pp. 1, 117).
- [124] F. Mauro and A. Kana. “Digital twin for ship life-cycle: A critical systematic review”. In: *Ocean Engineering* 269 (2023), p. 113479. ISSN: 00298018. DOI: 10.1016/j.oceaneng.2022.113479 (cit. on pp. 115, 117).
- [125] A. F. Molland, ed. *The maritime engineering reference book: a guide to ship design, construction and operation*. Oxford: Butterworth-Heinemann, 2008. ISBN: 978-0-7506-8987-8 (cit. on pp. 5–7, 9).
- [126] National Academies of Sciences, Engineering, and Medicine. *Foundational Research Gaps and Future Directions for Digital Twins*. Washington, DC: The National Academies Press, 2024. ISBN: 978-0-309-70042-9. DOI: 10.17226/26894 (cit. on pp. 115, 116).
- [127] L. Nikolopoulos and E. Boulougouris. “A novel method for the holistic, simulation driven ship design optimization under uncertainty in the big data era”. In: *Ocean Engineering* 218 (2020), p. 107634. ISSN: 00298018. DOI: 10.1016/j.oceaneng.2020.107634 (cit. on pp. 1, 116).
- [128] H. Nobukawa and G. Zhou. “Discrete Optimization of Ship Structures with Genetic Algorithms”. In: *Journal of the Society of Naval Architects of Japan* 1996.179 (1996), pp. 293–301. ISSN: 1884-2070, 0514-8499. DOI: 10.2534/jjasnaoe1968.1996.293 (cit. on p. 5).
- [129] J. Nocedal and S. J. Wright. *Numerical optimization*. Second edition. Springer series in operation research and financial engineering. New York, NY: Springer, 2006. ISBN: 978-0-387-30303-1 (cit. on p. 40).
- [130] C. Oh, J. M. Tomczak, E. Gavves, and M. Welling. “Combinatorial Bayesian Optimization using the Graph Cartesian Product”. In: *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. NeurIPS’19. Red Hook, NY, USA: Curran Associates Inc., 2019 (cit. on p. 52).
- [131] H. Ohno. “Analysis of Multi-objective Bayesian Optimization Using Random Scalarizations for Correlated Observations”. In: *SN Computer Science* 2.3 (2021), p. 214. ISSN: 2662-995X, 2661-8907. DOI: 10.1007/s42979-021-00505-y (cit. on p. 42).
- [132] M. Olson, E. Santorella, L. C. Tiao, S. Cakmak, M. Garrard, S. Daulton, Z. J. Lin, S. Ament, B. Beckerman, E. Onofrey, P. Igusti, C. Lara, B. Letham, C. Cardoso, S. S. Shen, A. C. Lin, M. Grange, E. Kashtelyan, D. Eriksson, M. Balandat, and E. Bakshy. “Ax: A Platform for Adaptive Experimentation”. In: *Proceedings of the Fourth International Conference on Automated Machine Learning*. Ed. by L. Akoglu, C. Doerr, J. N. van Rijn, R. Garnett, and J. R. Gardner. Vol. 293. PMLR. Proceedings of Machine Learning Research, 2025, pp. 21/1–25 (cit. on p. 41).

- [133] Y.-S. Ong and A. Gupta. "Evolutionary Multitasking: A Computer Science View of Cognitive Multitasking". In: *Cognitive Computation* 8.2 (2016), pp. 125–142. issn: 1866-9956, 1866-9964. doi: 10.1007/s12559-016-9395-7 (cit. on pp. 82, 96).
- [134] M. Palaversa, P. Prebeg, and J. Andric. "Current State of Development of Ship Structural Design and Optimization Methods". In: *Pomorski zbornik* Special edition.3 (2020), pp. 171–187. doi: 10.18048/2020.00.13. (cit. on p. 5).
- [135] A. Paleyes, M. Mahsereci, and N. D. Lawrence. "Emukit: A Python toolkit for decision making under uncertainty". In: *Proceedings of the Python in Science Conference* (2023) (cit. on p. 57).
- [136] A. Paleyes, M. Pullin, M. Mahsereci, C. McCollum, N. Lawrence, and J. González. "Emulation of physical processes with Emukit". In: *Second Workshop on Machine Learning and the Physical Sciences*. NeurIPS'19. 2019 (cit. on p. 57).
- [137] S. J. Pan and Q. Yang. "A Survey on Transfer Learning". In: *IEEE Transactions on Knowledge and Data Engineering* 22.10 (2010), pp. 1345–1359. issn: 1041-4347. doi: 10.1109/TKDE.2009.191 (cit. on pp. 2, 82).
- [138] A. Papanikolaou, E. Boulougouris, S.-O. Erikstad, S. Harries, and A. A. Kana. "Ship Design in the Era of Digital Transition: A State-of-the-Art Report". In: *International Marine Design Conference* (2024). issn: 3050-4864. doi: 10.59490/imdc.2024.784 (cit. on p. 117).
- [139] A. D. Papanikolaou. "Holistic Approach to Ship Design". In: *Journal of Marine Science and Engineering* 10.11 (2022), p. 1717. issn: 2077-1312. doi: 10.3390/jmse10111717 (cit. on pp. 1, 115, 116, 132).
- [140] L. Papenmeier, L. Nardi, and M. Poloczek. *Bounce: Reliable High-Dimensional Bayesian Optimization for Combinatorial and Mixed Spaces*. 2024. doi: 10.48550/arXiv.2307.00618 (cit. on p. 40).
- [141] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. "PyTorch: an imperative style, high-performance deep learning library". In: *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d. Alché-Buc, E. Fox, and R. Garnett. Vol. 32. NeurIPS'19. Curran Associates, Inc., 2019 (cit. on p. 40).
- [142] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12.85 (2011), pp. 2825–2830. issn: 1532-4435 (cit. on pp. 86, 105).

-
- [143] J. Pelamatti, L. Brevault, M. Balesdent, E.-G. Talbi, and Y. Guerin. “How to Deal with Mixed-Variable Optimization Problems: An Overview of Algorithms and Formulations”. In: *Advances in Structural and Multidisciplinary Optimization*. Ed. by A. Schumacher, T. Vietor, S. Fiebig, K.-U. Bletzinger, and K. Maute. Cham: Springer International Publishing, 2018, pp. 64–82. ISBN: 978-3-319-67987-7. DOI: 10.1007/978-3-319-67988-4_5 (cit. on p. 53).
 - [144] P. Perdikaris, M. Raissi, A. Damianou, N. D. Lawrence, and G. E. Karniadakis. “Non-linear information fusion algorithms for data-efficient multi-fidelity modelling”. In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 473.2198 (2017), p. 20160751. ISSN: 1364-5021, 1471-2946. DOI: 10.1098/rspa.2016.0751 (cit. on p. 27).
 - [145] S. Pineda, J. M. Morales, and A. Jiménez-Cordero. “Learning-assisted optimization for transmission switching”. In: *TOP* 32.3 (2024), pp. 489–516. ISSN: 1134-5764, 1863-8279. DOI: 10.1007/s11750-024-00672-0 (cit. on p. 83).
 - [146] A. Pinkus. *n-Widths in Approximation Theory*. Ergebnisse der Mathematik und ihrer Grenzgebiete, A Series of Modern Surveys in Mathematics 7. Berlin, Heidelberg: Springer Berlin Heidelberg, 1985. ISBN: 978-3-642-69896-5. DOI: 10.1007/978-3-642-69894-1 (cit. on p. 22).
 - [147] K. V. Price, R. M. Storn, and J. A. Lampinen. *Differential Evolution*. Natural Computing Series. Berlin/Heidelberg: Springer, 2005. ISBN: 978-3-540-20950-8. DOI: 10.1007/3-540-31306-0 (cit. on pp. 41, 58).
 - [148] S. Pyrialakos, A. Chaudhuri, and K. E. Willcox. “Multi-Resolution Digital Twins Under Uncertainty With Applications to Multiscale Systems”. In: *AIAA SCITECH 2026 Forum*. Orlando, FL: American Institute of Aeronautics and Astronautics, 2026. ISBN: 978-1-62410-765-8. DOI: 10.2514/6.2026-1799 (cit. on p. 120).
 - [149] N. V. Queipo, R. T. Haftka, W. Shyy, T. Goel, R. Vaidyanathan, and P. Kevin Tucker. “Surrogate-based analysis and optimization”. In: *Progress in Aerospace Sciences* 41.1 (2005), pp. 1–28. ISSN: 03760421. DOI: 10.1016/j.paerosci.2005.02.001 (cit. on pp. 21, 43).
 - [150] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann. “Stable-baselines3: Reliable reinforcement learning implementations”. In: *Journal of machine learning research* 22.1 (2021), pp. 1–8. ISSN: 1532-4435 (cit. on pp. 117, 119, 120).
 - [151] J. Raikunen, E. Avi, H. Remes, J. Romanoff, I. Lillemäe-Avi, and A. Niemelä. “Optimisation of passenger ship structures in concept design stage”. In: *Ships and Offshore Structures* 14.sup1 (2019), pp. 320–334. ISSN: 1744-5302, 1754-212X. DOI: 10.1080/17445302.2019.1590947 (cit. on p. 5).
 - [152] D. Ramos, L. Lacasa, E. Valero, and G. Rubio. *Transfer learning-enhanced deep reinforcement learning for aerodynamic airfoil optimisation subject to structural constraints*. 2025. DOI: 10.48550/arXiv.2505.02634 (cit. on p. 96).

- [153] J. Rapin and O. Teytaud. *Nevergrad - A gradient-free optimization platform*. <https://github.com/FacebookResearch/Nevergrad>. 2018 (cit. on p. 2).
- [154] J. Rapin, M. Gallagher, P. Kerschke, M. Preuss, and O. Teytaud. “Exploring the MLDA benchmark on the nevergrad platform”. In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. GECCO’19. Prague, Czech Republic: Association for Computing Machinery, 2019, pp. 1888–1896. ISBN: 9781450367486. DOI: 10.1145/3319619.3326830 (cit. on p. 57).
- [155] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. Cambridge, Massachusetts: The MIT Press, 2005. ISBN: 978-0-262-25683-4. DOI: 10.7551/mitpress/3206.001.0001 (cit. on p. 22).
- [156] RINA S.p.A. *RINA Rules*. Online; accessed on 2026-08-06. 2025. URL: <https://www.rina.org/en/rules> (cit. on pp. 5, 6).
- [157] L. Rokach, O. Maimon, and E. Shmueli, eds. *Machine Learning for Data Science Handbook: Data Mining and Knowledge Discovery Handbook*. Cham: Springer International Publishing, 2023. ISBN: 978-3-031-24627-2. DOI: 10.1007/978-3-031-24628-9 (cit. on p. 22).
- [158] F. Romor, M. Tezzele, A. Lario, and G. Rozza. “Kernel-based active subspaces with application to computational fluid dynamics parametric problems using the discontinuous Galerkin method”. In: *International Journal for Numerical Methods in Engineering* 123.23 (2022), pp. 6000–6027. ISSN: 0029-5981, 1097-0207. DOI: 10.1002/nme.7099 (cit. on p. 65).
- [159] F. Romor, M. Tezzele, M. Mrosek, C. Othmer, and G. Rozza. “Multi-fidelity data fusion through parameter space reduction with applications to automotive engineering”. In: *International Journal for Numerical Methods in Engineering* 124.23 (2023), pp. 5293–5311. ISSN: 0029-5981, 1097-0207. DOI: 10.1002/nme.7349 (cit. on pp. 27, 28).
- [160] Sambu Seo, M. Wallat, T. Graepel, and K. Obermayer. “Gaussian process regression: active data selection and test point rejection”. In: *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*. Vol. 3. Como, Italy: IEEE, 2000, pp. 241–246. ISBN: 978-0-7695-0619-7. DOI: 10.1109/IJCNN.2000.861310 (cit. on pp. 45, 46).
- [161] H. G. Santos and T. Toffolo. “Mixed integer linear programming with python”. In: *COINOR Computational Infrastructure for Operations Research* (2020) (cit. on p. 57).
- [162] Y. Satish Kumar and M. Mukhopadhyay. “Finite element analysis of ship structures using a new stiffened plate element”. In: *Applied Ocean Research* 22.6 (2000), pp. 361–374. ISSN: 01411187. DOI: 10.1016/S0141-1187(00)00014-6 (cit. on pp. 5, 12).

-
- [163] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. "The Graph Neural Network Model". In: *IEEE Transactions on Neural Networks* 20.1 (2009), pp. 61–80. DOI: 10.1109/TNN.2008.2005605 (cit. on p. 85).
 - [164] B. Schölkopf. "The Kernel Trick for Distances". In: *Proceedings of the 14th International Conference on Neural Information Processing Systems*. Ed. by T. Leen, T. Dietterich, and V. Tresp. Vol. 13. NeurIPS'00. Denver, CO: MIT Press, 2000, pp. 283–289 (cit. on pp. 22, 100).
 - [165] B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. The MIT Press, 2001. ISBN: 978-0-262-25693-3. DOI: 10.7551/mitpress/4175.001.0001 (cit. on p. 22).
 - [166] A. Schrijver. *Theory of Linear and Integer Programming*. Chichester, Weinheim: John Wiley & Sons, 1998. ISBN: 978-0-471-98232-6 (cit. on pp. 2, 39, 68).
 - [167] M. W. Scroggs, I. A. Baratta, C. N. Richardson, and G. N. Wells. "Basix: a runtime finite element basis evaluation library". In: *Journal of Open Source Software* 7.73 (2022), p. 3982. ISSN: 2475-9066. DOI: 10.21105/joss.03982 (cit. on p. 123).
 - [168] M. W. Scroggs, J. S. Dokken, C. N. Richardson, and G. N. Wells. "Construction of Arbitrary Order Finite Element Degree-of-Freedom Maps on Polygonal and Polyhedral Cell Meshes". In: *ACM Transactions on Mathematical Software* 48.2 (2022), pp. 1–23. ISSN: 0098-3500, 1557-7295. DOI: 10.1145/3524456 (cit. on p. 123).
 - [169] T. W. Sederberg and S. R. Parry. "Free-form deformation of solid geometric models". In: *SIGGRAPH Comput. Graph.* 20.4 (1986), pp. 151–160. ISSN: 0097-8930. DOI: 10.1145/15886.15903 (cit. on p. 65).
 - [170] A. Serani and M. Diez. "A Survey on Design-Space Dimensionality Reduction Methods for Shape Optimization". In: *Archives of Computational Methods in Engineering* 33.2 (2026), pp. 1671–1698. ISSN: 1134-3060, 1886-1784. DOI: 10.1007/s11831-025-10349-x (cit. on p. 2).
 - [171] A. Serani and M. Diez. "Parametric model embedding". In: *Computer Methods in Applied Mechanics and Engineering* 404 (2023), p. 115776. ISSN: 00457825. DOI: 10.1016/j.cma.2022.115776 (cit. on pp. 2, 65).
 - [172] A. Serani, G. Palma, and M. Diez. *A nonlinear extension of parametric model embedding for dimensionality reduction in parametric shape design*. 2026. DOI: 10.48550/arXiv.2605.11759 (cit. on p. 65).
 - [173] A. Serani, T. P. Scholcz, and V. Vanzi. "A Scoping Review on Simulation-Based Design Optimization in Marine Engineering: Trends, Best Practices, and Gaps". In: *Archives of Computational Methods in Engineering* 31.8 (2024), pp. 4709–4737. ISSN: 1134-3060, 1886-1784. DOI: 10.1007/s11831-024-10127-1 (cit. on p. 21).

- [174] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas. “Taking the Human Out of the Loop: A Review of Bayesian Optimization”. In: *Proceedings of the IEEE* 104.1 (2016), pp. 148–175. doi: 10.1109/JPROC.2015.2494218 (cit. on pp. 2, 40, 50).
- [175] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. Van Den Driessche, T. Graepel, and D. Hassabis. “Mastering the game of Go without human knowledge”. In: *Nature* 550.7676 (2017), pp. 354–359. issn: 0028-0836, 1476-4687. doi: 10.1038/nature24270 (cit. on p. 41).
- [176] D. Simon. *Evolutionary Optimization Algorithms: biologically-inspired and population-based approaches to computer intelligence*. Hoboken, New Jersey: Wiley, 2013. isbn: 978-0-470-93741-9 (cit. on p. 43).
- [177] S. N. Sivanandam and S. N. Deepa. *Introduction to Genetic Algorithms*. Berlin, Heidelberg: Springer, 2008. isbn: 978-3-540-73189-4. doi: 10.1007/978-3-540-73190-0 (cit. on p. 41).
- [178] D. A. Spielman and S.-H. Teng. “Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time”. In: *Journal of the Association for Computing Machinery* 51.3 (2004), pp. 385–463. issn: 0004-5411. doi: 10.1145/990308.990310 (cit. on p. 42).
- [179] N. Srinivas, A. Krause, S. Kakade, and M. W. Seeger. “Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design”. In: *Proceedings of the 27th International Conference on International Conference on Machine Learning*. ICML’10. Haifa, Israel: Omnipress, 2010, pp. 1015–1022. isbn: 9781605589077 (cit. on p. 50).
- [180] L. Sun and D. Wang. “Optimal structural design of the midship of a VLCC based on the strategy integrating SVM and GA”. In: *Journal of Marine Science and Application* 11.1 (2012), pp. 59–67. issn: 1671-9433, 1993-5048. doi: 10.1007/s11804-012-1106-5 (cit. on p. 5).
- [181] Y. Sun, G. G. Yen, and Z. Yi. “IGD Indicator-Based Evolutionary Algorithm for Many-Objective Optimization Problems”. In: *IEEE Transactions on Evolutionary Computation* 23.2 (2019), pp. 173–187. doi: 10.1109/TEVC.2018.2791283 (cit. on p. 106).
- [182] R. S. Sutton and A. G. Barto. *Reinforcement learning: an introduction*. Second edition. Adaptive computation and machine learning series. Cambridge, Massachusetts: The MIT Press, 2018. isbn: 978-0-262-03924-6 (cit. on pp. 117, 119, 122).
- [183] S. Suzuki, S. Takeno, T. Tamura, K. Shitara, and M. Karasuyama. “Multi-objective Bayesian Optimization using Pareto-frontier Entropy”. In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by H. C. Daumé and A. Singh. Vol. 119. ICML’20. PMLR, 2020, pp. 9279–9288 (cit. on p. 45).

-
- [184] R. Swischuk, L. Mainini, B. Peherstorfer, and K. Willcox. “Projection-based model reduction: Formulations for physics-based machine learning”. In: *Computers & Fluids* 179 (2019), pp. 704–717. ISSN: 0045-7930. DOI: 10.1016/j.compfluid.2018.07.021 (cit. on p. 1).
 - [185] M. Tadros, M. Ventura, and C. G. Soares. “Review of current regulations, available technologies, and future trends in the green shipping industry”. In: *Ocean Engineering* 280 (2023), p. 114670. ISSN: 00298018. DOI: 10.1016/j.oceaneng.2023.114670 (cit. on p. 1).
 - [186] E.-G. Talbi. “Combining metaheuristics with mathematical programming, constraint programming and machine learning”. In: *Annals of Operations Research* 240.1 (2016), pp. 171–215. ISSN: 0254-5330, 1572-9338. DOI: 10.1007/s10479-015-2034-y (cit. on p. 96).
 - [187] E.-G. Talbi. “Machine Learning into Metaheuristics: A Survey and Taxonomy”. In: *ACM Computing Surveys* 54.6 (2022), pp. 1–32. ISSN: 0360-0300, 1557-7341. DOI: 10.1145/3459664 (cit. on pp. 82, 96).
 - [188] E.-G. Talbi. *Metaheuristics: From Design to Implementation*. 1st ed. Wiley, 2009. ISBN: 978-0-470-27858-1. DOI: 10.1002/9780470496916 (cit. on p. 41).
 - [189] M. Tezzele, S. Carr, U. Topcu, and K. E. Willcox. “Adaptive planning for risk-aware predictive digital twins”. In: *Physics-Based and Data-Driven Modeling for Digital Twins*. Ed. by K. Cherifi and I. V. Gosea. Vol. 8. ICIAM2023 Springer Series. Singapore: Springer Nature, 2026, pp. 45–64. DOI: 10.1007/978-981-96-9108-1_3 (cit. on pp. 116, 117).
 - [190] M. Tezzele, L. Fabris, M. Sidari, M. Sicchiero, and G. Rozza. “A multifidelity approach coupling parameter space reduction and nonintrusive POD with application to structural optimization of passenger ship hulls”. In: *International Journal for Numerical Methods in Engineering* 124.5 (2023), pp. 1193–1210. ISSN: 0029-5981, 1097-0207. DOI: 10.1002/nme.7159 (cit. on pp. 5, 28, 64).
 - [191] M. Torzoni, D. Maisto, A. Manzoni, F. Donnarumma, G. Pezzulo, and A. Corigliano. “Active digital twins via active inference”. In: *Engineering Applications of Artificial Intelligence* 174 (2026), p. 114519. ISSN: 09521976. DOI: 10.1016/j.engappai.2026.114519 (cit. on pp. 116, 117).
 - [192] M. Torzoni, M. Tezzele, S. Mariani, A. Manzoni, and K. E. Willcox. “A digital twin framework for civil engineering structures”. In: *Computer Methods in Applied Mechanics and Engineering* 418 (2024), p. 116584. ISSN: 00457825. DOI: 10.1016/j.cma.2023.116584 (cit. on pp. 115–118, 122).
 - [193] M. Towers, A. Kwiatkowski, J. Terry, J. U. Balis, G. D. Cola, T. Deleu, M. Goulão, A. Kallinteris, M. Krimmel, A. KG, R. Perez-Vicente, A. Pierré, S. Schulhoff, J. J. Tai, H. Tan, and O. G. Younis. *Gymnasium: A Standard Interface for Reinforcement Learning Environments*. 2025. DOI: 10.48550/arXiv.2407.17032 (cit. on pp. 117, 120).

- [194] L. N. Trefethen and D. Bau III. *Numerical Linear Algebra*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 1997. ISBN: 978-0-89871-361-9. DOI: 10.1137/1.9780898719574 (cit. on pp. 22, 24).
- [195] P. J. M. Van Laarhoven and E. H. L. Aarts. "Simulated annealing". In: *Simulated Annealing: Theory and Applications*. Dordrecht: Springer Netherlands, 1987, pp. 7–15. ISBN: 978-90-481-8438-5. DOI: 10.1007/978-94-015-7744-1_2 (cit. on p. 41).
- [196] E. Varetto, M. Torzoni, M. Tezzele, and A. Manzoni. "Adaptive digital twins for predictive decision-making: online Bayesian learning of transition dynamics". In: *Data-Centric Engineering* 7 (2026), e33. DOI: 10.1017/dce.2026.10072 (cit. on p. 117).
- [197] 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, Í. 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". In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2 (cit. on pp. 40, 57, 124).
- [198] M. Volpp, L. P. Fröhlich, K. Fischer, A. Doerr, S. Falkner, F. Hutter, and C. Daniel. *Meta-Learning Acquisition Functions for Transfer Learning in Bayesian Optimization*. 2020. DOI: 10.48550/arXiv.1904.02642 (cit. on p. 96).
- [199] A. Wächter and L. T. Biegler. "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming". In: *Mathematical Programming* 106.1 (2006), pp. 25–57. ISSN: 0025-5610, 1436-4646. DOI: 10.1007/s10107-004-0559-y (cit. on p. 40).
- [200] C. Wang and S. Mahadevan. "Heterogeneous domain adaptation using manifold alignment". In: *IJCAI proceedings - International Joint Conference on Artificial Intelligence*. Vol. 22. IJCAI'11. 2011, p. 1541. DOI: 10.5591/978-1-57735-516-8/IJCAI11-259 (cit. on p. 96).
- [201] S. Watanabe. *Tree-Structured Parzen Estimator: Understanding Its Algorithm Components and Their Roles for Better Empirical Performance*. 2025. DOI: 10.48550/arXiv.2304.11127 (cit. on p. 52).
- [202] S. Watanabe, N. Awad, M. Onishi, and F. Hutter. "Speeding Up Multi-Objective Hyperparameter Optimization by Task Similarity-Based Meta-Learning for the Tree-Structured Parzen Estimator". In: *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*. IJCAI'23. Macau, SAR China: International Joint Conferences on Artificial Intelligence Organization, 2023, pp. 4380–4388. ISBN: 978-1-956792-03-4. DOI: 10.24963/ijcai.2023/487 (cit. on p. 97).

-
- [203] Y. Wei, X. Chen, J. Wang, and D. Wan. "Multi-objective hull form optimization utilizing sequential sampling optimization method". In: *Ocean Engineering* 310 (2024), p. 118667. ISSN: 00298018. DOI: 10.1016/j.oceaneng.2024.118667 (cit. on p. 42).
 - [204] Wei-Neng Chen, Jun Zhang, H. Chung, Wen-Liang Zhong, Wei-Gang Wu, and Yu-hui Shi. "A Novel Set-Based Particle Swarm Optimization Method for Discrete Optimization Problems". In: *IEEE Transactions on Evolutionary Computation* 14.2 (2010), pp. 278–300. ISSN: 1941-0026, 1089-778X. DOI: 10.1109/TEVC.2009.2030331 (cit. on p. 53).
 - [205] H. Wendland. *Scattered Data Approximation*. 1st ed. Cambridge University Press, 2004. ISBN: 978-0-521-84335-5. DOI: 10.1017/CB09780511617539 (cit. on p. 22).
 - [206] K. Willcox. "Unsteady flow sensing and estimation via the gappy proper orthogonal decomposition". In: *Computers & Fluids* 35.2 (2006), pp. 208–226. ISSN: 00457930. DOI: 10.1016/j.compfluid.2004.11.006 (cit. on p. 22).
 - [207] C. Wu, Y. Liang, W. Lin, H. Lee, and S. Lim. "A note on equivalence of proper orthogonal decomposition methods". In: *Journal of Sound and Vibration* 265.5 (2003), pp. 1103–1110. ISSN: 0022460X. DOI: 10.1016/S0022-460X(03)00032-4 (cit. on p. 22).
 - [208] W. Wu and J. Ding, eds. *Proceedings of the 22nd International Ship and Offshore Structures Congress (Volume 1): Technical Committee Reports*. ISSC'25. Springer Nature, 2026. ISBN: 978-981-95-2668-0. DOI: 10.1007/978-981-95-2668-0 (cit. on pp. 115, 116).
 - [209] W. Xiao, M. He, Z. Wei, and N. Wang. "SWLC-DT: An Architecture for Ship Whole Life Cycle Digital Twin Based on Vertical–Horizontal Design". In: *Machines* 10.11 (2022), p. 998. ISSN: 2075-1702. DOI: 10.3390/machines10110998 (cit. on p. 117).
 - [210] Q. Xu, N. Wang, L. Wang, W. Li, and Q. Sun. "Multi-Task Optimization and Multi-Task Evolutionary Computation in the Past Five Years: A Brief Review". In: *Mathematics* 9.8 (2021), p. 864. ISSN: 2227-7390. DOI: 10.3390/math9080864 (cit. on p. 96).
 - [211] X. Xue, K. Zhang, K. C. Tan, L. Feng, J. Wang, G. Chen, X. Zhao, L. Zhang, and J. Yao. "Affine Transformation-Enhanced Multifactorial Optimization for Heterogeneous Problems". In: *IEEE Transactions on Cybernetics* 52.7 (2022), pp. 6217–6231. ISSN: 2168-2267, 2168-2275. DOI: 10.1109/TCYB.2020.3036393 (cit. on p. 96).
 - [212] L. Yu, Z. Meng, L. Kong, V. Snasel, and J.-S. Pan. "Surrogate-assisted differential evolution: A survey". In: *Swarm and Evolutionary Computation* 94 (2025), p. 101879. ISSN: 22106502. DOI: 10.1016/j.swevo.2025.101879 (cit. on pp. 41, 58).
 - [213] D. Zhan and H. Xing. "Expected improvement for expensive optimization: a review". In: *Journal of Global Optimization* 78.3 (2020), pp. 507–544. DOI: 10.1007/s10898-020-00923-x (cit. on p. 51).

- [214] Q. Zhang and H. Li. "MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition". In: *IEEE Transactions on Evolutionary Computation* 11.6 (2007), pp. 712–731. DOI: 10.1109/TEVC.2007.892759 (cit. on p. 42).
- [215] Y. Zhang, S. Wang, and G. Ji. "A Comprehensive Survey on Particle Swarm Optimization Algorithm and Its Applications". In: *Mathematical Problems in Engineering* 2015 (2015), pp. 1–38. ISSN: 1024-123X, 1563-5147. DOI: 10.1155/2015/931256 (cit. on pp. 52, 53).
- [216] A. A. Zhiglavski and A. Zhilinskas. *Stochastic global optimization*. Springer optimization and its applications 9. New York: Springer, 2008. ISBN: 978-0-387-74022-5 (cit. on p. 40).
- [217] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal. "Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization". In: *ACM Transactions on mathematical software (TOMS)* 23.4 (1997), pp. 550–560. DOI: 10.1145/279232.279236 (cit. on pp. 40, 57).

DATABASE OF OPTIMIZED SHIPS

This chapter presents the construction of a database of optimized ships, which provides the knowledge base for the approaches based on transfer learning discussed in this thesis. To produce high quality optimal configurations for the test cases discussed in Chapter 2, we run our computational pipeline in an loop of optimization and parameterization refinement leveraging the approach discussed in Chapter 5.

An important difference from Chapter 5 is the optimization phase: the previous approach required a high number of high-fidelity evaluations due to the stopping criteria being defined on the convergence of the multi-objective infill criterion and m_{gap} . To streamline the creation of the database, we instead adopt a simpler schema by performing a fixed number of multi-objective infills and setting a tolerance for the convergence of m_{gap} . We choose to perform 4 multi-objective infills, corresponding to 36 high-fidelity configurations selected for [PF](#) refinement. Moreover, instead of alternating [BO](#) and [PDS](#) for single-objective optimization, in the same optimization step we concatenate a fixed number of [PDS](#) iterations and a time-limited run of [PSO](#). A loop of single-objective optimization and infill ends when the difference between the high-fidelity and the surrogate m_{gap} differ by less than 0.1%.

In Section A.1, we discuss the choice of constraint thresholds for the midship variants and show the results of the optimizations. In Section A.2, discuss the peculiarities of the full ship models and their optimizations, along with the results of their optimizations.

A.1 Midship section variants

The midship section variants are optimized using constraints values determined only by their modifications with respect to the base model. Vertical deflection is adjusted according to the length of the section, increasing by 15 mm for each additional ring, but is left unchanged for shorter variants. The [vertical center of gravity \(VCG\)](#) is adjusted in a similar manner, increasing by 1.5 mm for each additional deck, but is left the same if the decks decrease. The critical values for n_y and n_b are kept the same for all variants.

Table A.1 shows the different optimization constraints and penalty thresholds applied to the midship section variants.

Table A.1: Midship section variants optimization constraints.

Code	Δ_x	Δ_y	Δ_z	n_y	n_b	Deflection [mm]	VCG [m]
MS00	0	0	0	200	4000	140	15.0
MS01	0	0	-1	200	4000	140	15.0
MS02	0	0	1	200	4000	140	16.5
MS03	0	-1	0	200	4000	140	15.0
MS04	0	1	0	200	4000	140	15.0
MS05	-1	0	0	200	4000	140	15.0
MS06	1	0	0	200	4000	155	15.0
MS07	1	-1	1	200	4000	155	16.5
MS08	-1	1	0	200	4000	140	15.0
MS09	-1	0	-1	200	4000	140	15.0
MS10	1	1	-1	200	4000	155	15.0
MS11	2	0	2	200	4000	170	18.0
MS12	1	2	-1	200	4000	155	15.0
MS13	-1	2	-1	200	4000	140	15.0
MS14	2	0	0	200	4000	170	15.0
MS15	-2	1	-2	200	4000	140	15.0
MS16	1	0	-2	200	4000	155	15.0
MS17	-1	0	-2	200	4000	140	15.0
MS18	-2	2	1	200	4000	140	16.5
MS19	2	2	2	200	4000	170	18.0
MS20	-1	2	0	200	4000	140	15.0

For each model, the size of the initial sampling is 20 + the minimum configuration $\mathbf{x}_{LB}$. Each variant is reparameterized 2 times via the response-based approach, setting $n_{clusters} = 2$ and limiting $n_{dim} \leq 20$. The size of the random sampling after each update of the parameterization is equal to 2 times the number of newly created parameters.

Tables A.2 and A.3 shows the evolution of the resulting optima for the test and train datasets, along with the truncation rank, n_{dim} , the final size of the high-fidelity database, and the mean amortized query time of the full surrogate. We separate models MS00-MS06 since these are more closely related and constitute the main test cases for [TL](#)-based approaches.

For most models, the optimized configurations were able to greatly reduce the m_{gap}

Table A.2: Midship section variants optima - test ships.

Code	Δ_x	Δ_y	Δ_z	Rank	n_{dim}	m_{gap} [%]	# hifi	Query time [s]
MS00	0	0	0	8	5	15.44	60	0.005
				13	10	7.50	125	0.005
				20	17	7.15	213	0.006
MS01	0	0	-1	7	5	25.82	60	0.004
				11	9	14.18	120	0.004
				16	14	13.98	190	0.004
MS02	0	0	1	7	5	5.17	60	0.005
				11	9	4.08	117	0.005
				16	14	3.34	190	0.005
MS03	0	-1	0	7	5	10.58	60	0.005
				12	10	7.97	122	0.005
				19	17	7.60	204	0.005
MS04	0	1	0	7	5	21.28	60	0.005
				12	10	8.92	119	0.005
				18	16	8.87	196	0.005
MS05	-1	0	0	7	5	8.44	60	0.005
				12	10	4.84	122	0.005
				20	18	4.24	200	0.004
MS06	1	0	0	7	5	23.88	60	0.005
				12	10	10.84	119	0.005
				20	18	10.46	203	0.005

with respect to the initial parameterization. However, the second reparameterization shows only marginal performance increase over the first, suggesting that for this simple geometries 10 parameters represent the appropriate trade-off between problem expressivity and number of high-fidelity evaluations.

The amortized time for a parallel prediction via the full surrogates range from 0.004 to 0.007 s, enabling the efficient implementation of population-based metaheuristics.

A.2 Full ship fleet

Although the optimal configuration of each ship was obtained via the optimization pipeline, the design process of each ship used different criteria for the choice values for the optimization constraints, reparameterization constraints, and number of iterations. In particular, FS6 and FS7 were optimized using only single-objective optimization procedures, namely [BO](#) and [PDS](#). FS9 saw the first adoption of [PSO](#), which superseded [BO](#) as the primary method for single-objective optimization and was then used on FS0 through FS5, and FS8. Moreover, FS9 was reparameterized with the "one-shot" approach from 11 to 40 parameters. These inconsistencies reflect the practical reality of the design process, in which each project has a different story.

All the larger ships have been reparameterized with 30 or more parameters, and especially for FS0, FS1 and FS2, the m_{gap} of the accepted optima are significantly higher

Table A.3: Midship section variants optima - train ships.

Code	Δ_x	Δ_y	Δ_z	Rank	n_{dim}	m_{gap} [%]	# hifi	Query time [s]
MS07	-1	1	1	7	5	47.40	63	0.005
				12	10	30.93	128	0.005
				17	15	28.55	200	0.006
MS08	-1	1	0	7	5	11.23	63	0.005
				12	10	2.83	128	0.005
				18	16	2.53	202	0.005
MS09	-1	0	-1	7	5	12.14	60	0.004
				11	9	5.08	126	0.004
				16	14	4.64	199	0.004
MS10	1	1	-1	7	5	46.02	60	0.005
				11	9	23.39	120	0.005
				17	15	23.39	201	0.005
MS11	2	0	2	7	5	11.47	60	0.006
				12	10	4.21	119	0.006
				19	17	3.47	192	0.007
MS12	1	2	-1	8	5	50.11	60	0.005
				12	9	24.94	120	0.005
				18	15	24.70	198	0.005
MS13	-1	2	-1	7	5	42.55	60	0.005
				11	9	21.88	117	0.005
				16	14	21.72	193	0.005
MS14	2	0	0	8	5	37.46	60	0.005
				13	10	13.43	119	0.005
				21	18	12.97	203	0.006
MS15	-2	1	-2	7	5	15.45	66	0.004
				11	9	6.56	126	0.004
				16	14	6.23	196	0.004
MS16	1	0	-2	8	5	73.66	60	0.004
				12	9	41.70	120	0.004
				17	14	41.70	187	0.005
MS17	-1	0	-2	7	5	45.36	60	0.004
				11	9	24.67	123	0.004
				17	15	24.24	198	0.004
MS18	-2	2	1	7	5	2.20	60	0.005
				11	9	1.84	120	0.005
				17	15	1.61	192	0.005
MS19	2	2	2	7	5	15.58	60	0.006
				11	9	4.33	117	0.006
				18	16	3.73	188	0.007
MS20	-1	2	0	7	5	11.90	60	0.005
				11	9	3.62	123	0.005
				17	15	3.10	198	0.005

Table A.4: Full ships optima.

Code	Length	Width	Height	Rank	n_{dim}	m_{gap} [%]	# hifi	Query time [s]
FS0	0.98	1.10	1.06	13	9	38.93	67	0.039
				22	18	23.20	163	0.043
				34	30	22.13	265	0.049
FS1	1.08	1.10	1.06	14	9	53.68	61	0.040
				23	18	24.13	148	0.045
				38	33	22.67	277	0.051
FS2	1.15	1.10	1.13	14	9	43.83	61	0.049
				20	15	25.45	136	0.046
				40	35	21.66	257	0.056
FS3	0.81	0.72	0.76	13	9	12.06	67	0.016
				19	15	6.35	139	0.018
FS4	0.90	0.78	0.84	12	8	13.69	56	0.019
				18	14	8.93	126	0.022
FS5	0.86	0.78	0.84	13	8	16.78	59	0.023
				22	17	10.76	135	0.025
FS6	0.95	0.78	0.89	16	10	33.69	30	0.034
				22	16	19.20	62	0.033
				33	27	14.35	160	0.034
				46	40	12.15	321	0.036
FS7	1.16	1.35	1.25	14	11	27.11	68	0.092
				20	17	18.34	177	0.090
				33	30	14.12	389	0.090
FS8	0.83	0.79	0.78	17	12	14.85	138	0.020
				21	16	7.25	215	0.021
FS9	1.08	1.21	1.12	13	11	35.14	65	0.050
				42	40	14.66	277	0.069

than those of the smaller ships. Surprisingly, the two largest ships FS7 and FS9 achieved m_{gap} only slightly higher than FS6, the most difficult of the smaller ships.

The amortized time for a parallel prediction via the full surrogates range from 0.016 to 0.092 s, with larger difference with respect to the midship section variants reflecting the higher diversity of ship sizes.

