PAPER • OPEN ACCESS

Integrated environment for machine learning-aided heat transfer optimisation in internal flows: hammerhead software

To cite this article: D M Segura Galeana *et al* 2026 *Mach. Learn.: Sci. Technol.* **7** 015011

View the article online for updates and enhancements.

MACHINE LEARNING
Science and Technology

OPEN ACCESS

RECEIVED
16 October 2025

REVISED
9 December 2025

ACCEPTED FOR PUBLICATION
22 December 2025

PUBLISHED
14 January 2026

PAPER

Integrated environment for machine learning-aided heat transfer optimisation in internal flows: hammerhead software

D M Segura Galeana[1,*], A Liptak[2], A J Gil[1], M G Edwards[1], A Dubas[3] and A Davis[3]

[1] Zienkiewicz Institute for Modelling, Data and AI, Faculty of Science and Engineering, Swansea University, Swansea, United Kingdom
[2] Diamond Light Source, Harwell Science and Innovation Campus, Didcot, United Kingdom
[3] UK Atomic Energy Authority, Culham Campus, Abingdon, United Kingdom
[*] Author to whom any correspondence should be addressed.

E-mail: d.m.m.seguragaleana@swansea.ac.uk, alexander.liptak@diamond.ac.uk and a.j.gil@swansea.ac.uk

Keywords: machine learning, computational fluid dynamics, neural networks, heat transfer, Gaussian processes

Abstract

The optimisation of the cooling system performance in a tokamak reactor requires the accurate analysis of the thermo-hydraulic interactions under complex flow regimes. Although computational fluid dynamics (CFD) provides high-fidelity insight at a lower cost than experimental testing, optimisation remains computationally prohibitive due to the large number of simulations required. To address this, we introduce Hammerhead, the first open-source, transparent and fully integrated machine learning (ML)-CFD framework with automated high-fidelity database generation and modular multi-surrogate modelling for optimisation in pipe flow heat and mass transfer problems. The software provides seamless Python-OpenFOAM integration for automatic high-fidelity model database building and multi-surrogate model comparison from 3 ML algorithms with modular architecture within Python infrastructure. For the high-fidelity database, the software constructs a parametric space using up to four shape parameters to deform pipe wall geometries and perform the conjugate heat transfer simulations. The surrogate models are constructed on the basis of radial basis function interpolation, feed-forward neural networks, and Gaussian processes, and trained on the high-fidelity database to approximate thermo-hydraulic responses efficiently. The framework provides a comparative environment for systematically assessing surrogate model accuracy and reliability within the same optimisation workflow. The approach offers a practical tool that increases the feasibility of design optimisation on small datasets for cooling system applications. On a database of up to 400 cases for a 2D domain, a geometry with thermo-hydraulic performance enhancement of over 500% with respect to that of a smooth surface pipe at $Re = 1000$ was found with the aid of surrogate modelling, validated through high-fidelity simulation.

1. Introduction

Cooling systems are critical across a wide range of engineering applications, from laptops to aircraft engines and thermal power plants. They rely on fluid flow to prevent devices from exceeding thermal limits or to extract energy from heat sources. While most cooling systems are designed to maximise thermal extraction with little consideration of hydraulic loss, maximising the energy output of power plant cooling systems typically requires optimising the coupled thermal [1] and hydraulic mechanisms of the flow [2]. The trade-off between these mechanisms often implies that increasing thermal power output comes at the cost of higher hydraulic losses, or vice versa [3], leading to a multi-objective optimisation (MOO) problem with non-unique solutions.

The thermo-hydraulic performance (THP) of pipe flows in cooling systems can be optimised through a range of methods involving numerous parameters. Previous studies, often focusing on either thermal or hydraulic effects in isolation, have shown that carefully designed surface roughness geometries can enhance efficiency for the targeted effect [4–6]. In this work, we address the MOO problem for both

THP aspects, with emphasis on shape parametrisation, where geometric parameters control the roughness of the fluid-facing surface of the pipe.

The effect of surface roughness on pipe flow THP has been widely investigated through experiments and high-fidelity computational fluid dynamics (CFD) [7, 8]. However, when applied to optimisation, experimental approaches face significant practical challenges, while CFD rapidly becomes computationally prohibitive as the parameter space expands. In such cases, surrogate modelling using machine learning (ML) [9] can provide an effective alternative, operating on limited datasets and enabling efficient exploration of parametric spaces to identify optimal solutions [10].

ML-aided CFD has been employed to construct digital twins [11], predict flow variables [12], estimate quantities of interest [13], and address optimisation problems [10, 14]. Within the engineering domain, surrogate models have been applied to material property prediction [15], stress distribution analysis [16], and exploration of high-dimensional parametric spaces [17], amongst other tasks. Despite their success, these studies are typically resource-intensive, application-specific, and rely on ML models trained on data without *a priori* knowledge, potentially leading to data bias -as reported in previous studies [18, 19] and undermining the model's performance and generalisation.

Optimising THP in pipe flows therefore requires a methodology that combines high-fidelity CFD with surrogate modelling. The rise of ML underscores the need to integrate advanced algorithms with high-fidelity models to accelerate engineering design. This motivates the development of a unified tool that bridges high-fidelity data generation and surrogate modelling, guided by *a priori* knowledge, to achieve more efficient design optimisation. To address these challenges, commercial software like Ansys OptiSLang [20], Cadence [21] and CONVERGE Studio [22] have integrated ML models into their sensitivity analysis process. These tools are designed to reduce the computational cost of parametric design optimisation whilst improving the efficiency of parameter exploration. However, these licensed solutions operate largely as black boxes: users have limited control over the underlying ML methodology, they have no external CFD solver compatibility, and modular extensibility is essentially absent. Furthermore, their optimisation pipelines are primarily tailored to locating local or global minima of the sensitivity problem, thereby discarding potentially valuable insights obtainable from analysing the broader parameter space from the available high-fidelity datasets.

As an alternative to these proprietary environments and a first step towards a community driven, fully integrated CFD-ML generalised framework, we present Hammerhead, an open-source software package for CFD-driven, ML-aided THP optimisation.

Hammerhead, developed in Python 3, provides a robust and user-friendly environment for semi-automated THP optimisation. It includes both a graphical user interface for interactive use and a parallelised, console-based interface designed for high-performance computing (HPC) [23] environments. Key features include geometric design with shape and flow parametrisation, automated body-fitted mesh generation, and seamless integration with OpenFOAM [24] for high-fidelity CFD simulations. To enable efficient surrogate modelling, Hammerhead processes simulation data using proper orthogonal decomposition (POD) [25], and supports a modular architecture for radial basis function (RBF) interpolation [26], neural networks (NNs) [27, 28], and Gaussian processes (GPs) [29–31]. In addition, it provides tools for THP evaluation and shape optimisation through inverse analysis techniques.

The aim of this research is to provide the research community with a novel open-source toolkit that fully integrates CFD high-fidelity data generation with modular multi-surrogate modelling to automate the optimisation of cooling systems' geometry design for THP enhancement. The software is limited to simplified surface roughness for low Reynolds numbers in the range $Re = [500, 8000]$, but its modular framework permits adjustments for new geometries, domains and other software with low user cost. The paper introduces the reader to Hammerhead's functionality and presents the mathematical formulation as a basis for the high-fidelity and surrogate modelling processes incorporated in the software. In summary, the key novelties that Hammerhead provides the research community with are:

- Open-source software for CFD design optimisation, intended for community-driven development.
- Fully integrated CFD-ML framework.
- Automated high-fidelity database generation.
- Seamless integration with OpenFOAM.
- Multi-surrogate modelling with modular architecture.
- End-to-End THP optimisation workflow.

The paper is broken down as follows. Section 2 briefly outlines the numerical model corresponding to the conjugate heat transfer problem. In this section, the objective functions required for the optimisation are also presented. Next, section 3 describes the reduced order and ML modelling approach based on POD, RBF, NN and GP. Specifically, the focus of POD is that of feature extraction, whilst the rest of algorithms are used as approximation methods to interpolate data features. Then, section 4 presents the Hammerhead software package, developed to automate the high-fidelity data generation and surrogate model training. This includes a brief explanation of Hammerhead's main function and data processing capabilities, where information from the high-fidelity simulations is transformed into multi-parameter and single- and/or multi-output tensors for the surrogate model training. Finally, in section 5 we conclude the work carried out outlining its strengths, current limitations and paths of prospective research.

2. Governing equations and numerical methods

This section introduces the mathematical framework that underpins the high-fidelity model and MOO functions used in this work. Description of the computational domain, governing equations, physical assumptions, interface and boundary conditions are presented. As the work focuses on optimising the energy output of a cooling system, the proposed MOO functions are given by the flow dissipation rate and advected heat flux, which are naturally coupled, and are evaluated at the inlet and outlet of the modelled pipe. Under the assumption of stationary flow, the OpenFOAM solver `chtMultiRegionSimpleFoam` [32] performs a steady-state simulation of the numerical model: a domain with modular surface geometry using Hammerhead's interface, coupled to OpenFOAM's body-fitting mesh generation tool `blockMesh`.

2.1. Mathematical formulation

The problem considered is formulated as that of conjugate heat transfer, where the flow is assumed to be Newtonian, steady-state, and incompressible. The conservation of mass, linear momentum and energy, under the assumption of constant thermophysical properties [7, 33] are

$$\nabla \cdot \boldsymbol{v} = 0 \qquad \text{in } \Omega_\mathrm{f} \tag{1a}$$

$$\rho \nabla \cdot (\boldsymbol{v} \otimes \boldsymbol{v}) = -\nabla p + \nabla \cdot \left[\mu \left(\nabla \boldsymbol{v} + (\nabla \boldsymbol{v})^\mathrm{T}\right)\right] \qquad \text{in } \Omega_\mathrm{f}, \tag{1b}$$

$$\nabla \cdot (\rho \boldsymbol{v} h) = \nabla \cdot (k_d \nabla T) + \nabla \cdot (p \boldsymbol{v}) \qquad \text{in } \Omega_\mathrm{f}, \tag{1c}$$

where $\boldsymbol{v}$ denotes the velocity field, ρ is the fluid density, p is the pressure field, μ is the dynamic viscosity of the fluid, h is the specific enthalpy, k_d is the (isotropic) thermal diffusivity and T is the temperature, all defined in the fixed control volume of the fluid region Ω_f. The solid region Ω_s for the conjugate heat problem is described only by the thermal energy equation [34] and interface conditions between the fluid and the solid as

$$\nabla \cdot (k_d \nabla T) = 0 \qquad \text{in } \Omega_\mathrm{s}, \tag{2a}$$

$$T|_{\Omega_\mathrm{f}} = T|_{\Omega_\mathrm{s}}, \quad (k_d \nabla T)|_{\Omega_\mathrm{f}} \cdot \boldsymbol{n}|_{\Gamma_\mathrm{sf}} = (k_d \nabla T)|_{\Omega_\mathrm{s}} \cdot \boldsymbol{n}|_{\Gamma_\mathrm{sf}} \qquad \text{on } \Gamma_\mathrm{sf}, \tag{2b}$$

where Γ_sf is the surface interface between solid and fluid regions and $\boldsymbol{n}|_{\Gamma_\mathrm{sf}}$ is the interface normal vector. The Reynolds number $Re = \frac{\rho \bar{v} L}{\mu}$ and Prandtl number $Pr = \frac{\mu}{\rho \alpha}$ are given in terms of a representative length L, the dynamic viscosity of the fluid μ, the thermal diffusivity of the fluid $\alpha = \frac{k_d}{\varsigma_p \rho}$, with ς_p the specific heat coefficient, and the averaged flow velocity defined as $\bar{v} = \dot{V}/A$, where $\dot{V}$ is the volumetric flow rate and A is the cross-sectional area. The problem considered encompasses a range of values for Re and a single value for Pr, namely, $Re = [500, 8000]$ and $Pr = 0.2414$. This Prandtl value was selected to reflect the thermophysical properties of Helium, as it is considered suitable for a fusion-relevant cooling system [35, 36]. The verified regime of Reynolds numbers is considered as an exploratory research that includes both laminar and turbulent flows with a non-prohibitive computational cost; this is used to study the influence of surface roughness on the flow and benchmark the surrogate models' performance within both regimes.

2.1.1. Objective functions

As the MOO problem is approached in this work from the perspective of energy output optimisation, the viscous dissipation rate $\dot{D}$ and advected heat flux $\dot{Q}$ are considered the appropriate objective functions [1, 3], defined as

$$\dot{D} = f_\mathrm{f}(\boldsymbol{v}, p) = -\int_{\partial \Omega_\mathrm{f}} (p + \rho k)\, \boldsymbol{v} \cdot \boldsymbol{n}\, \mathrm{d}\Gamma_\mathrm{f}, \tag{3}$$

$$\dot{Q} = f_\mathrm{h}(\boldsymbol{v}, T) = \int_{\partial \Omega_\mathrm{f}} (\rho \varsigma_p T)\, \boldsymbol{v} \cdot \boldsymbol{n}\, \mathrm{d}\Gamma_\mathrm{f} - \dot{D}, \tag{4}$$

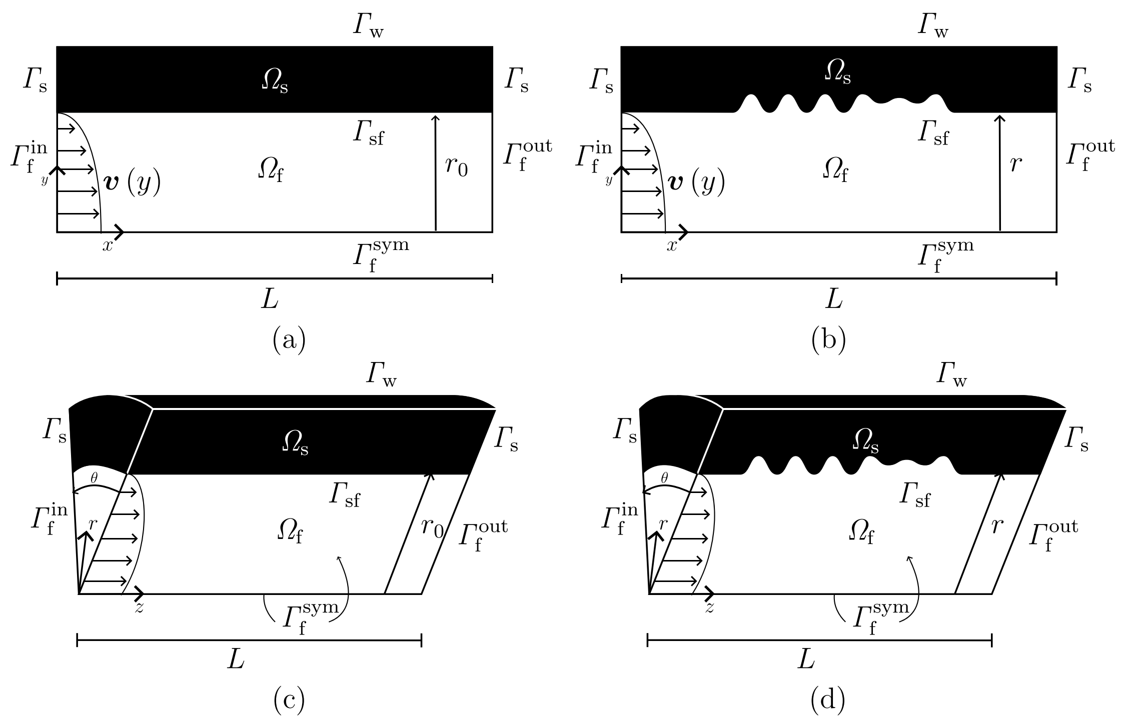


Figure 1. Domains of internal flow in a pipe represented by: (a) 2D flow over a *smooth* surface; (b) 2D flow over a *rough* surface; (c) axisymmetric flow over *smooth* surface; (d) axisymmetric flow over *rough* surface.

where $k = \frac{1}{2}\boldsymbol{v}\cdot\boldsymbol{v}$ is the kinetic energy and $\partial\Omega_f$ is the fluid bounding surface. Due to the nature of wall boundary conditions (i.e. $\boldsymbol{v}\cdot\boldsymbol{n} = 0$), (3) and (4) are only evaluated at the inlet and outlet boundaries, thus reducing the data size and computational cost, making it suitable for large databases. In addition to this, the description of advected heat flux from (4) already incorporates (3), thus the *global maximum* of $\dot{Q}$ from (4) within range of optimisation for $\dot{D}$ is considered to be the THP metric, removing the need for a sensitivity analysis to evaluate the MOO problem. The THP metric $\overline{\dot{Q}} = \dot{Q}_r/\dot{Q}_0$ is computed from (4), where $\dot{Q}_0$ refers to the baseline *smooth* pipe case, and $\dot{Q}_r$ represents an individual *rough* pipe case.

2.2. Computational domain

In this study, the computational domain of the pipe is treated through two distinct configurations: a two-dimensional cross-section and an axisymmetric wedge, as illustrated in figure 1. Both configurations include fluid Ω_f and solid Ω_s regions, the latter to address the conjugate heat transfer problem. The fluid bounding surface $\partial\Omega_f$ incorporates the part where symmetric boundary conditions are applied, that is Γ_f^{sym}, and the solid bounding surface $\partial\Omega_s$ incorporates the pipe outer wall Γ_w. Hammerhead has been designed to consider the geometry of the flow-facing surface as a double harmonic function, which in the simpler two dimensional case, can be expressed as

$$r(x) = r_0 + \sum_{i=1}^{N_h} A_i \left(\cos\left(\pi k_i x\right) + 1\right), \tag{5}$$

where r_0 is the baseline radius of the pipe, N_h is the number of harmonics, and A_i, k_i represent the amplitude and wavenumber the of ith harmonic, respectively. Currently, Hammerhead supports a maximum of $N_h = 2$ harmonics. As a result, the computational domain mesh is generated using OpenFOAM's `blockMesh` tool, with fully structured meshes applied to all regions, and a maximum grid resolution for $y+ = 1$ [4] at $Re = 9000$ [37]. Shared geometric parameters featuring in both two-dimensional cross-section and axisymmetric wedge domains are pipe length $L = 3$ m, pipe radius $r = 0.2$ m, and wall thickness $t = 0.06$ m. The axisymmetric wedge has an additional parameter: the wedge angle $\theta = 20°$.

[4] Dimensionless wall distance used to estimate grid resolution for the required level of accuracy in wall-bounded flows.

Table 1. Mesh parameters and maximum non-orthogonality of smooth and micro-structured surface examples. The 2D and wedge domains share these parameters on the x- and y- axes, while the 2D uses a single cell in the z-axis and the wedge has 8 cells.

h	Δy	y-cell count	x-cell count	z-cell count	y-expansion ratio
0.01	$7.14e^{-4}$	100	1100	$\{1, 8\}$	1.02
		Maximum non-orthogonality [°]			
		$A_1 = 0$ m	$A_1 = 0.001$ m	$A_1 = 0.01$ m	
		0	29.3	44.9	

Table 2. Average measured rates of convergence for the numerical solution for each flow variable field.

Domain	Re	q_{v_x}	q_{v_y}	q_{v_z}	q_T	q_p
2D	8000	1.495	1.2		1.7	2.18
Wedge	500	1.595	2.4	1.165	1.58	2.38

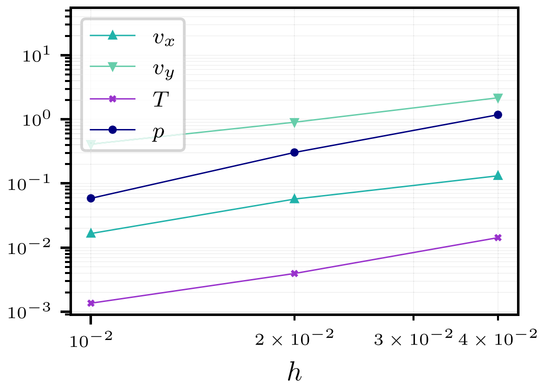


Figure 2. Relative norm L^2 error $||\varepsilon_r||_{L^2}$ against h for flow over a 2D smooth surface in $Re = 8000$.

Fully developed laminar flow and heat transfer are assumed at the inlet, with zero-gradient boundary conditions imposed for pressure and temperature. A constant pressure of zero is applied at the outlet, with zero-gradient boundary conditions imposed for velocity and temperature. Non-slip conditions are enforced at the wall, and a zero-gradient condition is applied to all variables at the symmetry boundaries. The temperature applied on the outer wall Γ_w is considered constant, with an initial solid/fluid temperature ratio $\frac{T_f}{T_s} = 0.6$, Pr $= 0.2414$, and thermal conductivity ratio $\frac{k_{d,f}}{k_{d,s}} = 21.6652$.

The mesh parameters are shown in table 1, and computations are performed with the OpenFOAM solver `chtMultiRegionSimpleFoam` [24, 32] using the SIMPLE algorithm [38] and a combination of first- and second-order schemes. The model grid independence assessment via mesh refinement has been carried out in [39], using 2 levels of coarser grids and a finer reference grid, with refinement ratios of 2 between each level. Examples of the average measured rates of convergence based on a relative norm L^2 error $||\varepsilon_r||_{L^2}$ found in this study are presented in table 2, and the convergence slope between each grid refinement level for the 2D domain is shown in figure 2. The expected rate of convergence is between 1 and 2 for the first and second order schemes selected. These results show enough accuracy and are regarded as mesh independent.

3. Surrogate modelling

This section presents an overview of the POD method employed as a dimensional reduction technique, followed by a brief description of the ML algorithms available in Hammerhead. The equations defined here outline the fundamental principles of each algorithm and are implemented using the open-source packages `SciPy` [40], `PyTorch` [41] and `GPyTorch` [42], to provide modular-architecture RBF, NN and GP classes, respectively.

POD is used to reduce the computational complexity of datasets, being noteworthy due to its widespread use for feature extraction [25, 43, 44], where in recent years the method has been applied to parametric problems [12, 45–47]. However, to produce a surrogate model, this method requires integration with other algorithms such as RBF interpolation, which is attractive for complex parametric spaces as it uses a linear combination of kernel-based type functions to estimate high dimensionality problems [12, 26, 48] and has verified accuracy [12, 46, 49].

Competing algorithms that can be used to produce surrogate models like NNs or GPs are more complex and computationally expensive than RBF, but the increasing abundance of data and processing power have allowed these to outperform simple interpolation methods. For NNs, architectures such as deep, convolutional and recurrent networks allow the models to accurately approximate non-linear problems [9, 27, 50–52], and similarly, the advantage of GPs [29, 53, 54] as stochastic processes is their ability to adaptively model parametric regions of interest.

3.1. Proper orthogonal decomposition

In optimisation problems, the computational cost of the models scale up when the dimensionality of the problem is increased. A feature extraction method is proposed to approximate lower-rank matrices of the high-fidelity database as a higher dimensional matrix, to reduce the inlet and outlet boundary datapoints and eliminate mesh dependence for the surrogate models. Consider a set of high fidelity solutions $\mathbf{Y}_i = \mathbf{Y}(\boldsymbol{\omega}_i) \in \mathrm{I\!R}^{N_\mathrm{d}}$, $i = 1 \ldots N_\mathrm{s}$, each corresponding to a different set of parameters $\boldsymbol{\omega}_i \in \mathrm{I\!R}^{N_\mathrm{s}}$, arranged into the form of a snapshot matrix $\mathbf{D} \in \mathrm{I\!R}^{N_\mathrm{s} \times N_\mathrm{d}}$ as

$$\mathbf{D} = \left[\mathbf{Y}_1, \mathbf{Y}_2, \ldots, \mathbf{Y}_{N_\mathrm{s}}\right]^\mathrm{T}. \tag{6}$$

The kth rank truncation of the matrix $\mathbf{D}$ can be computed through a singular value decomposition [25, 44] so that $\mathbf{D} = \mathbf{H}\boldsymbol{\Sigma}\mathbf{G}^\mathrm{T}$, where $\mathbf{H} \in \mathrm{I\!R}^{N_\mathrm{s} \times N_\mathrm{s}}$ and $\mathbf{G}^\mathrm{T} \in \mathrm{I\!R}^{N_\mathrm{d} \times N_\mathrm{d}}$ are unitary matrices and $\boldsymbol{\Sigma} \in \mathrm{I\!R}^{N_\mathrm{s} \times N_\mathrm{d}}$ is a block diagonal matrix with uncorrelated singular values arranged in decreasing order. Snapshot matrix $\mathbf{D}$ is then truncated by

$$\mathbf{D} \approx \mathbf{D}_k = \mathbf{H}_k \boldsymbol{\Sigma}_k \mathbf{G}_k^\mathrm{T}, \tag{7}$$

where k represents the number of singular values retained for the reconstruction. A low-dimensional matrix $\mathbf{P}_k = \mathbf{H}_k \boldsymbol{\Sigma}_k \in \mathrm{I\!R}^{N_\mathrm{s}} \times \mathrm{I\!R}^k$ is constructed within a subspace of the high-dimensional matrix $\mathbf{D}$ and represents the original matrix $\mathbf{D}$ in a lower dimension $k \lll N_\mathrm{d}$ [43, 55], used to perform the surrogate modelling computations at lower computational cost.

3.2. ML algorithms

Three ML algorithms area available in Hammerhead and will be described in what follows.

3.2.1. Radial basis function

The interpolated solution $\mathbf{Y}(\boldsymbol{\omega})$ can be obtained by exploiting the lower-rank matrix $\mathbf{P}_k$ interpolated through basis functions of the form

$$\mathbf{Y}(\boldsymbol{\omega}) = \sum_{j=1}^{k} \mathrm{w}_j \varphi\left(||\boldsymbol{\omega} - \boldsymbol{\omega}_j||\right) \mathbf{P}_k \mathbf{E}_j, \tag{8}$$

where $\mathbf{Y}(\boldsymbol{\omega})$ denotes the solution interpolation for parameters $\boldsymbol{\omega}$, φ is a pre-established kernel function, $\mathbf{E}_j$ is a unit vector in the jth direction, and w_j is an unknown weighting coefficient that needs to be found through optimisation. If the data point on features $\boldsymbol{\omega}_j$ is recognised as the *centre* of the kernel φ, the available data in the snapshot matrix can be directly used to interpolate the response function $\mathbf{Y}(\boldsymbol{\omega})$ [56]. The linear system is solved for the weight coefficients w_j with respect to the available data matrix $\mathbf{D}_k$.

3.2.2. Neural network

Consider a feedforward network of L layers with N neurons per layer. Each layer $l = 1, \ldots$,L has a weight matrix $\mathbf{W}^l \in \mathrm{I\!R}^{N^l} \times \mathrm{I\!R}^{N^l - 1}$, a bias unit vector $\mathbf{b}$ related to the weight matrix, supplemented by the input vector of *features*$\boldsymbol{\omega}$, and the output vector of hypothesis $\hat{\mathbf{h}}$ [9, 27]. The forward computations of each layer are

$$\mathbf{z}^l = \mathbf{W}^l \mathbf{g}^l \left(\mathbf{z}^{l-1} + \mathbf{b}\right) = \sum_{i=1}^{N} \mathbf{w}_i^l \mathrm{g}_i^l \left(\mathrm{z}_i^{l-1} + \mathbf{b}^l\right), \tag{9}$$

where

$$\mathbf{z}^0 = \boldsymbol{\omega}, \quad \mathbf{z}^{\mathrm{L}} = \hat{\mathbf{h}}. \tag{10}$$

The individual functions of the vector function $\mathbf{g}^l = \left[g_1^l, \ldots, g_N^l\right]^{\mathrm{T}}$ are referred to as the *activation functions* of each unit [50, 51]. The chosen equation for all g_i^l for the ith neuron is the *sigmoid*. The hypothesis $\hat{\mathbf{h}}$ then becomes the NN's approximation of $\mathbf{Y}$ from (6). The weight coefficients $\mathbf{W}$ are obtained through backpropagation [50, 57] using the mean square error J gradient for multilayer networks

$$J\ (\mathbf{W}) = \frac{1}{N_{\mathrm{s}}} \sum_{q=1}^{N_{\mathrm{s}}} ||\mathbf{Y}_q\ \left(\boldsymbol{\omega}_q\right) - \mathbf{y}_q\ \left(\boldsymbol{\omega}_q, \mathbf{W}\right)||^2. \tag{11}$$

3.2.3. GP

The Gaussian distribution [29, 58] of a real process $\mathbf{f}(\boldsymbol{\omega})$ is defined by

$$\mathbf{f}(\boldsymbol{\omega}) \sim \mathcal{N}\left(\boldsymbol{\mu}(\boldsymbol{\omega}), \mathbf{K}(\boldsymbol{\omega}, \boldsymbol{\omega}')\right), \tag{12}$$

where μ is the mean function, and $\mathbf{K}$ is the covariance function or *kernel*. Let $\mathbf{y}$ be the known values of function $\mathbf{f}$ from the observed data at $\boldsymbol{\omega}$, where noisy observations can be accounted for with the error $\epsilon \sim \mathcal{N}\left(\mathbf{0}, \boldsymbol{I}\sigma^2\right)$ as

$$\mathbf{y}(\boldsymbol{\omega}) = \mathbf{f}(\boldsymbol{\omega}) + \epsilon, \tag{13a}$$

$$\mathbf{y}(\boldsymbol{\omega}) \sim \left(\boldsymbol{\mu}(\boldsymbol{\omega}), \mathbf{K}(\boldsymbol{\omega}, \boldsymbol{\omega}') + \sigma^2 \boldsymbol{I}\right). \tag{13b}$$

where σ^2 is the variance parameter. Consider $\mathbf{f}_*$ as a set of function values at new location points $\boldsymbol{\omega}_*$ from a test set. The conditional distribution of $\mathbf{f}_*$ with a given $\mathbf{f}$ is computed using the multivariate normal theorem [54]

$$\mathbf{f}_* | \mathbf{y} \sim \mathcal{N}\left(\bar{\mathbf{f}}, \operatorname{cov}(\mathbf{f}_*)\right), \tag{14}$$

where

$$\bar{\mathbf{f}} := \mathbb{E}\left[\mathbf{f}_* | \boldsymbol{\omega}, \mathbf{y}, \boldsymbol{\omega}_*\right] = \mathbf{K}_*^{\mathrm{T}} \left(\mathbf{K} + \sigma^2 \boldsymbol{I}\right)^{-1} \mathbf{y}, \tag{15}$$

$$\operatorname{cov}(\mathbf{f}_*) = \mathbf{K}_{**} - \mathbf{K}_*^{\mathrm{T}} \left(\mathbf{K} + \sigma^2 \boldsymbol{I}\right)^{-1} \mathbf{K}_*. \tag{16}$$

In the previous equations, $\boldsymbol{\mu}(\boldsymbol{\omega}_i), i = 1, \ldots, n_s$ corresponds to the training means and analogously, $\boldsymbol{\mu}_*$ to the test means, and the covariance terms are $\mathbf{K} = (\boldsymbol{\omega}, \boldsymbol{\omega}')$ for training set, $\mathbf{K}_* = (\boldsymbol{\omega}, \boldsymbol{\omega}_*)$ for training-test set, and $\mathbf{K}_{**} = (\boldsymbol{\omega}_*, \boldsymbol{\omega}_*')$ for test set [29, 54]. This is the posterior distribution for any random set of cases $\boldsymbol{\omega}_*$ when a training set $\boldsymbol{\omega}$ is available, and constitutes the central equation for GP predictions. The mean and covariance functions are parametrised and updated through a process similar to the mean square error gradient based on the log marginal likelihood $\mathrm{L_J}$ [53]

$$\mathrm{L_J} = \log p\left(\mathbf{f} | \boldsymbol{\omega}, \mathbf{W}\right) = -\frac{1}{2} \mathbf{f}^{\mathrm{T}} \mathbf{K}^{-1} \mathbf{f} - \frac{1}{2} \log |\mathbf{K}| - \frac{n_s}{2} \log 2\pi. \tag{17}$$

The benchmark of these algorithms has been carried out in [39], using the 2D pipe cross-section examples within $Re = [500,\ 8000]$ regimes. Multiple models were constructed with varying architectures (different kernels for RBF and GP, variations on neuron and layer count for NNs), amount of input features (2 and 4 for shape parameter analysis, 3 and 5 when including Re as a parameter), output dimension size (single point from dissipation rate and advected heat flux, 100 points from inlet and outlet boundary mesh points, or the low-dimensional representation of the mesh points resulting from POD) and training set size. A summary of the *a priori* considerations and the metrics used to analyse the models' performance is presented in table 3. Dimension sensitivity refers to the scalability of the model architecture and their performance characteristics when increasing the input and output dimensions. The convergence and over-fitting metrics reference the resulting mean square error J and maximum relative error $\mathrm{Er_{max}}$ from the training and validation sets, respectively. An example case study resulting from this benchmark is presented in section 4.2.

Table 3. Comparison of performance metrics of the selected algorithms. *These metrics are based on the 2D cross-section of a pipe within $Re = [500,\ 8000]$ on databases of 600 cases, training with 65% of the total. **Computational cost of each algorithm depends on amount of features (Ranging from 2 to 5 in the analysis), output size (1 if $\dot{Q}$, 5 if $\mathbf{P}$ or 100 if the mesh datapoints are used) and database size (usually 65% of 600 cases; however, GP on 5 features uses only 10% due to the memory cost of the feature covariance matrix).

	RBF	NN	GP
Method	Linear kernel based interpolation	Non-linear statistical learning	Kernel based predictive probability distribution
Dimension sensitivity	Memory cost increases quadratically with the amount of data points in the matrix [59]	Networks' complexity increases non-linearly with the amount of features and output size	Cubic computational complexity dependent on amount of features [52]
Convergence J^* Over-fitting $Er_{\max}{}^*$	$\approx 10^{-25}$ $\approx 10^{-1}$	$\approx 10^{-6}$ $\approx 10^{-2}$	$\approx 10^{-2}$ $\approx 10^{-1}$
Computational cost (training)**	$[1,\ 15]$ s	$[5,\ 2425]$ s	$[1,\ 3000]$ s

4. Software package

Hammerhead has been developed as a software package to semi-automate the THP optimisation process of cooling systems based on shape and flow parameters using a mix of high-fidelity simulations and surrogate models. Hammerhead's functions can be executed either sequentially as an integrated workflow, or independently, as shown in figure 3, provided the necessary input data for each component is available. Below is a list of the main functions implemented in Hammerhead, along with a brief description of each step.

- **High-fidelity database population**: a database is built using OpenFOAM's `chtMultiRegionSimpleFoam` solver with flow and shape parameters based on *Re* and equation (5), which can be set programmatically or via the 'High-Fidelity Model settings' window shown in figure 4. By default, `chtMultiRegionSimpleFoam` is the only solver supported by Hammerhead, with compatibility verified for OpenFOAM versions `v2106` and `v2212`. However, Hammerhead's modular architecture enables straightforward adaptation for other solvers or OpenFOAM versions through minor modifications to the source code or configuration files.
- **Tensor data update**: hammerhead uses the high-fidelity database from the previous step to convert the available data into tensors optimised for surrogate modelling, as detailed in section 4.1.
- **Surrogate model training**: the ML algorithms RBF, NN, and GP are trained using the processed tensor data. Each model is implemented with a modular architecture that can be selected via the 'Surrogate Model settings' interface presented in figure 5. Hammerhead includes the boundaries of the parametric space in the training set, while the remaining parameter space is randomly partitioned into training and validation sets based on the specified `validation split` parameter. This approach helps minimise the risk of data bias and extrapolation.
- **Optimal parameter search**: the optimisation process uses the trained surrogate models to predict the shape and flow parameters corresponding to the *global maximum* of $\overline{\dot{Q}}$ as THP metric within the specified parametric space bounds.
- **Plotting and data visualisation**: the trained models are used to generate THP predictions, assess ML architecture performance, and track the evolution of the THP optimisation process. ML performance analysis is further divided into individual model evaluations, relative error metrics (figure 8), and architectural benchmarking (figure 6). Prediction results include THP distributions and variable profiles 10). Figures illustrating examples of additional model statistics are available in the supplementary material.

All computations are parallelised using Python's `multiprocessing` package to utilise the available system resources, with the exception of surrogate model training, where multithreading is handled automatically by the `PyTorch` package.

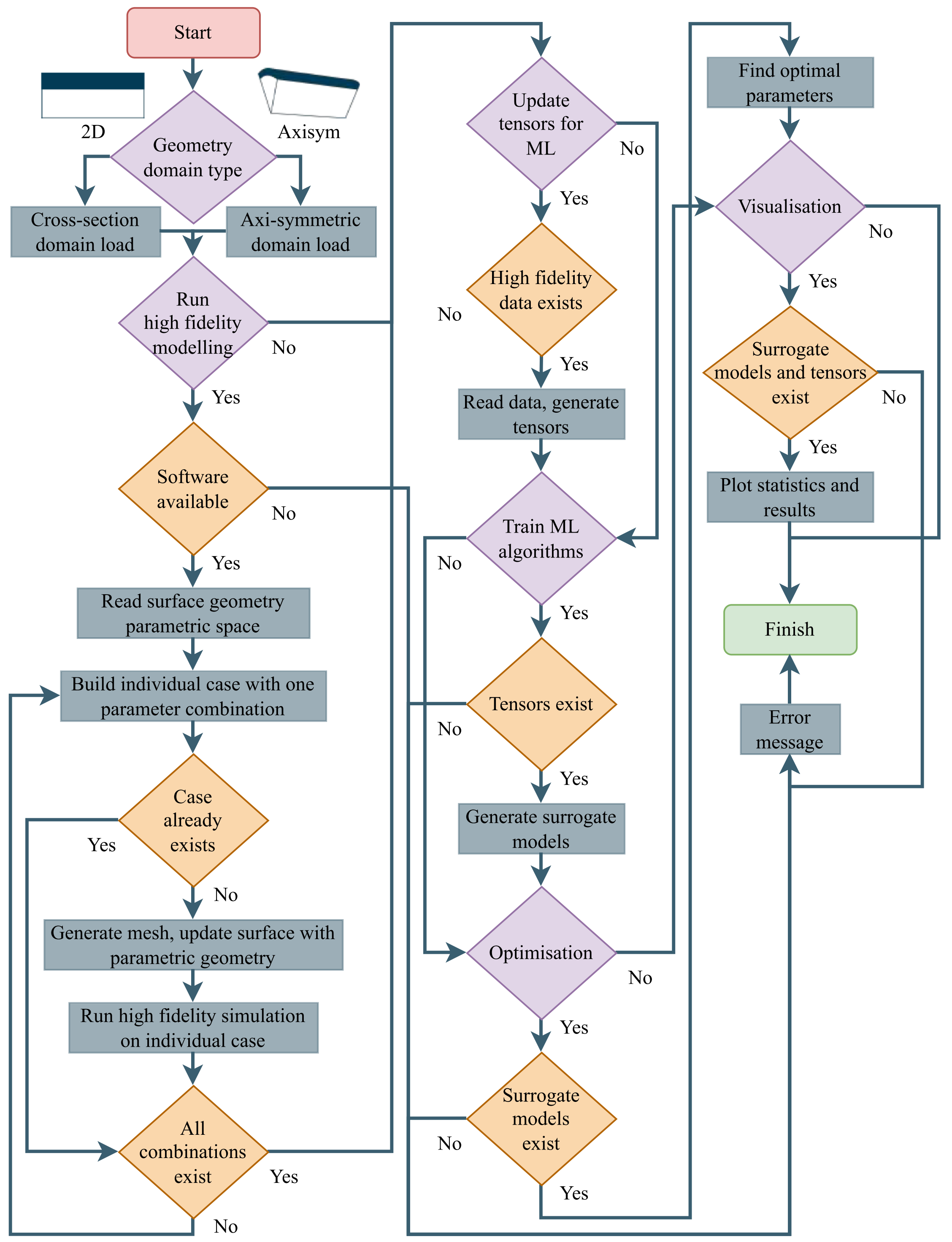
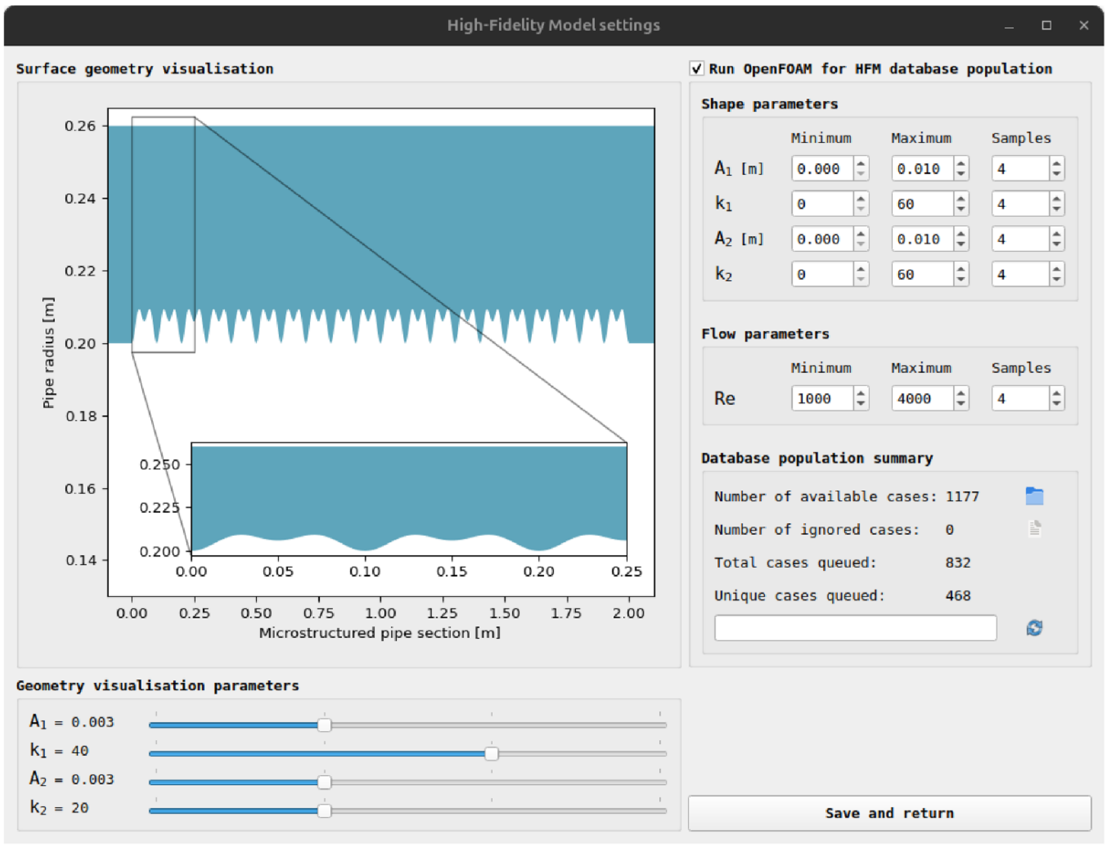

Start
2D
Axisym
Geometry domain type
Cross-section domain load
Axi-symmetric domain load
Run high fidelity modelling
No
Yes
Software available
No
Yes
Read surface geometry parametric space
Build individual case with one parameter combination
Case already exists
Yes
No
Generate mesh, update surface with parametric geometry
Run high fidelity simulation on individual case
All combinations exist
Yes
No
Update tensors for ML
No
Yes
High fidelity data exists
No
Yes
Read data, generate tensors
Train ML algorithms
No
Yes
Tensors exist
No
Yes
Generate surrogate models
Optimisation
No
Yes
Surrogate models exist
No
Yes
Find optimal parameters
Visualisation
No
Yes
Surrogate models and tensors exist
No
Yes
Plot statistics and results
Finish
Error message

Figure 3. Hammerhead flowchart, where software decisions are marked as yellow, user inputs are purple and Hammerhead functions handle the actions in blue [39].

4.1. Data processing for surrogate models

Hammerhead transforms data from the high-fidelity database into two sets of `PyTorch` tensors: input of *features* and output of *results*. The input tensors contain harmonic amplitude A_i and wavenumber k_i as shape parameters, and the Reynolds number Re as a flow parameter. The dimension of these tensors depends on the number of harmonics N_h and the inclusion of Re as an input feature, with sizes defined as



High-Fidelity Model settings

[✓] Run OpenFOAM for HFM database population

Shape parameters

	Minimum	Maximum	Samples
A_1 [m]	0.000	0.010	4
k_1	0	60	4
A_2 [m]	0.000	0.010	4
k_2	0	60	4

Flow parameters

	Minimum	Maximum	Samples
Re	1000	4000	4

Database population summary

Number of available cases:	1177
Number of ignored cases:	0
Total cases queued:	832
Unique cases queued:	468

Geometry visualisation parameters

A_1 = 0.003
k_1 = 40
A_2 = 0.003
k_2 = 20

Save and return

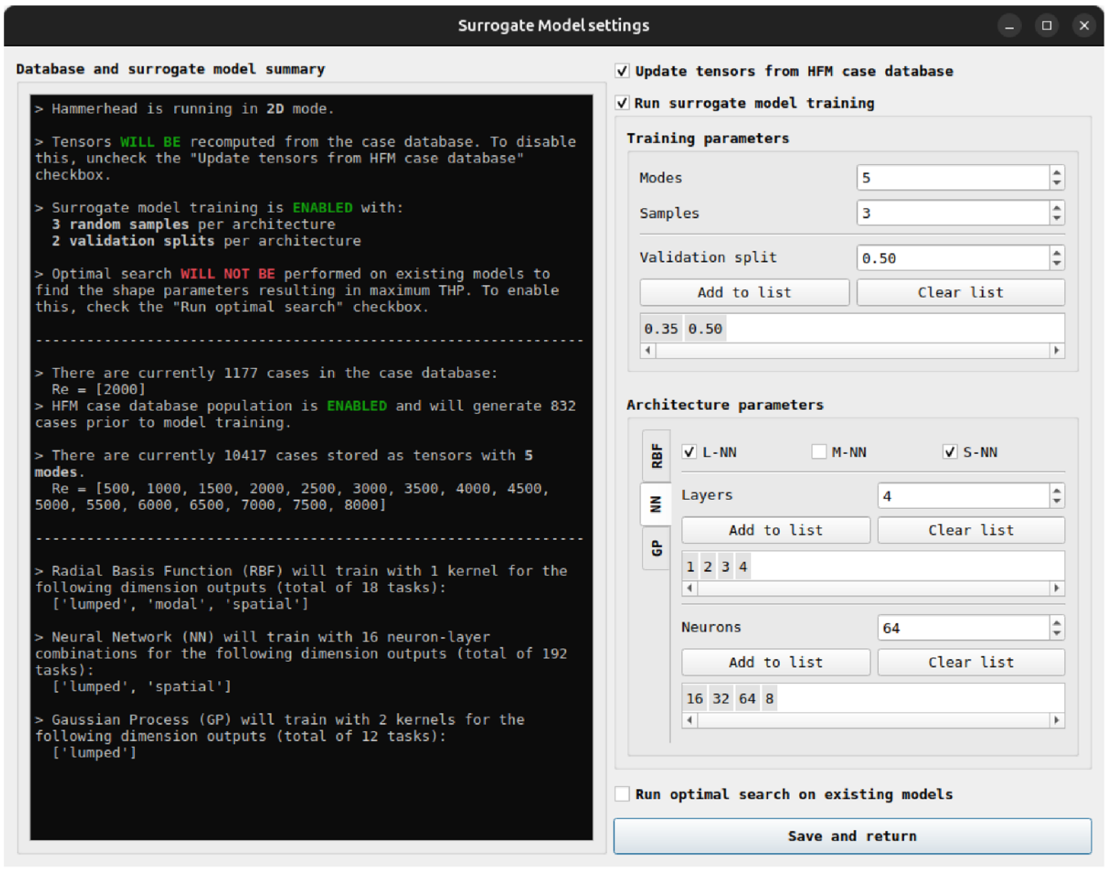
Figure 4. Hammerhead 'High-Fidelity Model settings' window. This interface allows for the selection of amplitudes, wave numbers, and Reynolds number, which are used to generate the pipe surface geometry based on (5) and to update the inlet velocity profile. The window also shows the number of cases currently available in the database, and the number of new cases queued for generation.

$$\omega_2 = [A_1, k_1], \qquad \text{when } Re = \text{constant}, N_\text{h} = 1, \tag{18a}$$
$$\omega_3 = [A_1, k_1, Re], \qquad \text{when } N_\text{h} = 1, \tag{18b}$$
$$\omega_4 = [A_1, k_1, A_2, k_2], \qquad \text{when } Re = \text{constant}, N_\text{h} = 2, \tag{18c}$$
$$\omega_5 = [A_1, k_1, A_2, k_2, Re], \qquad \text{when } N_\text{h} = 2. \tag{18d}$$

The output tensors are associated with one of three dimensions N_x, depending on how the data is processed prior to training the surrogate models. The *lumped* (L-) $\mathbf{x}^{\dot{Q}}$ is the only single-output tensor, with $N_x = 1$, containing the dissipation rate f_f and advected heat flux f_h from (3) and (4), respectively[5]. The *spatial* (S-) $\mathbf{x}^{\Gamma_\text{f}}$ is a mesh-dependent multi-output tensor, with $N_x = 100$, containing the grid information of inlet and outlet boundary values of the flow variables. Lastly, the *modal* (M-) is a mesh-independent multi-output tensor $\mathbf{x}^\mathbf{P}$ extracted from the columns of matrix $\mathbf{P}_k$, with $1 \leqslant N_x \leqslant 100$. The resulting tensor shapes are

$$\mathbf{x}^{\dot{Q}} = [f_\text{f}, f_\text{h}], \qquad \text{where } N_x = 1 \tag{19a}$$
$$\mathbf{x}^{\Gamma_\text{f}} = \left[{v_x}^\text{in}, T^\text{in}, p^\text{in}, {v_x}^\text{out}, T^\text{out}, p^\text{out}\right], \qquad \text{where } N_x = 100, \tag{19b}$$
$$\mathbf{x}^\mathbf{P} = \left[\mathbf{P}^\text{in}_{v_x}, \mathbf{P}^\text{in}_{T}, \mathbf{P}^\text{in}_{p}, \mathbf{P}^\text{out}_{v_x}, \mathbf{P}^\text{out}_{T}, \mathbf{P}^\text{out}_{p}\right], \qquad \text{where } 1 \leqslant N_x \leqslant 100. \tag{19c}$$

[5] Dissipation rate and advected heat flux are treated as independent variables, resulting in two lumped output tensors and therefore two surrogate models, each with a one-dimensional output.



Database and surrogate model summary

```
> Hammerhead is running in 2D mode.

> Tensors WILL BE recomputed from the case database. To disable
this, uncheck the "Update tensors from HFM case database"
checkbox.

> Surrogate model training is ENABLED with:
  3 random samples per architecture
  2 validation splits per architecture

> Optimal search WILL NOT BE performed on existing models to
find the shape parameters resulting in maximum THP. To enable
this, check the "Run optimal search" checkbox.

--------------------------------------------------------------

> There are currently 1177 cases in the case database:
  Re = [2000]
> HFM case database population is ENABLED and will generate 832
cases prior to model training.

> There are currently 10417 cases stored as tensors with 5
modes.
  Re = [500, 1000, 1500, 2000, 2500, 3000, 3500, 4000, 4500,
5000, 5500, 6000, 6500, 7000, 7500, 8000]

--------------------------------------------------------------

> Radial Basis Function (RBF) will train with 1 kernel for the
following dimension outputs (total of 18 tasks):
  ['lumped', 'modal', 'spatial']

> Neural Network (NN) will train with 16 neuron-layer
combinations for the following dimension outputs (total of 192
tasks):
  ['lumped', 'spatial']

> Gaussian Process (GP) will train with 2 kernels for the
following dimension outputs (total of 12 tasks):
  ['lumped']
```

[✓] Update tensors from HFM case database

[✓] Run surrogate model training

Training parameters

Modes	5
Samples	3
Validation split	0.50

0.35 0.50

Architecture parameters

[✓] L-NN [ ] M-NN [✓] S-NN

Layers	4

1 2 3 4

Neurons	64

16 32 64 8

[ ] Run optimal search on existing models

Figure 5. Hammerhead 'Surrogate Model settings' window. This interface allows for the selection of ML algorithms, output dimensions, and architectural configurations for surrogate model training. A summary of each active algorithm, global model parameters and the available training data are also provided.

4.2. Outcomes of example case

An example outcome obtained with Hammerhead is the optimisation of surface roughness geometric parameters for a 2D pipe cross-section to maximise $\dot{Q}$ within $Re = 1000$. An in-depth analysis of the high-fidelity data containing information of around 10,000 cases within $Re = [500,\ 8000]$ and a full benchmark of surrogate models were carried out in [39]. For this example, an initial benchmark of a NN trained on the low-rank matrix approximation of the boundary values with 65% of 600 high-fidelity model cases, is presented in figure 6. This study allowed us to understand the architecture best fit to approximate our specific problem, which is a network of 3 layers and 128 neurons per layer with a cross-validation as shown in figure 7.

A better understanding of the relative error between the surrogate model approximations and the high fidelity data across the data points is visible in figure 8, which presents a comparison with the GP surrogate model. The M-NN model shows a best fit to the data points, but the GP shows the confidence region even in points not available in the high-fidelity database. The M-NN model was used to predict the boundary values of the smooth surface case (baseline) alongside a gradient-based inverse analysis to produce a new set of geometric parameters that would optimise the net power output in the form of $\dot{Q}$ from (4), as shown in figure 9. The flow variable profiles of resulting boundary approximations of the baseline and optimised surfaces are shown in figure 10. The overlined variables $\overline{\phi}$ are the normalised values of each quantity of interest from (3) and (4) as $\overline{\phi} = \phi_{\mathrm{m}}/\phi_0$[6], where the subscripts m and 0 represent each case with surface roughness and the baseline smooth surface case, respectively. A high-fidelity model was built using such parameters to confirm the model's accuracy and credibility of the findings. The high-fidelity boundary values of the baseline and optimised surfaces are shown in figure 11.

[6] The definition of f_{h} in the figures neglects the dissipation rate to observe the impact that both dissipation and advected heat have in the net power output $\dot{Q}$.

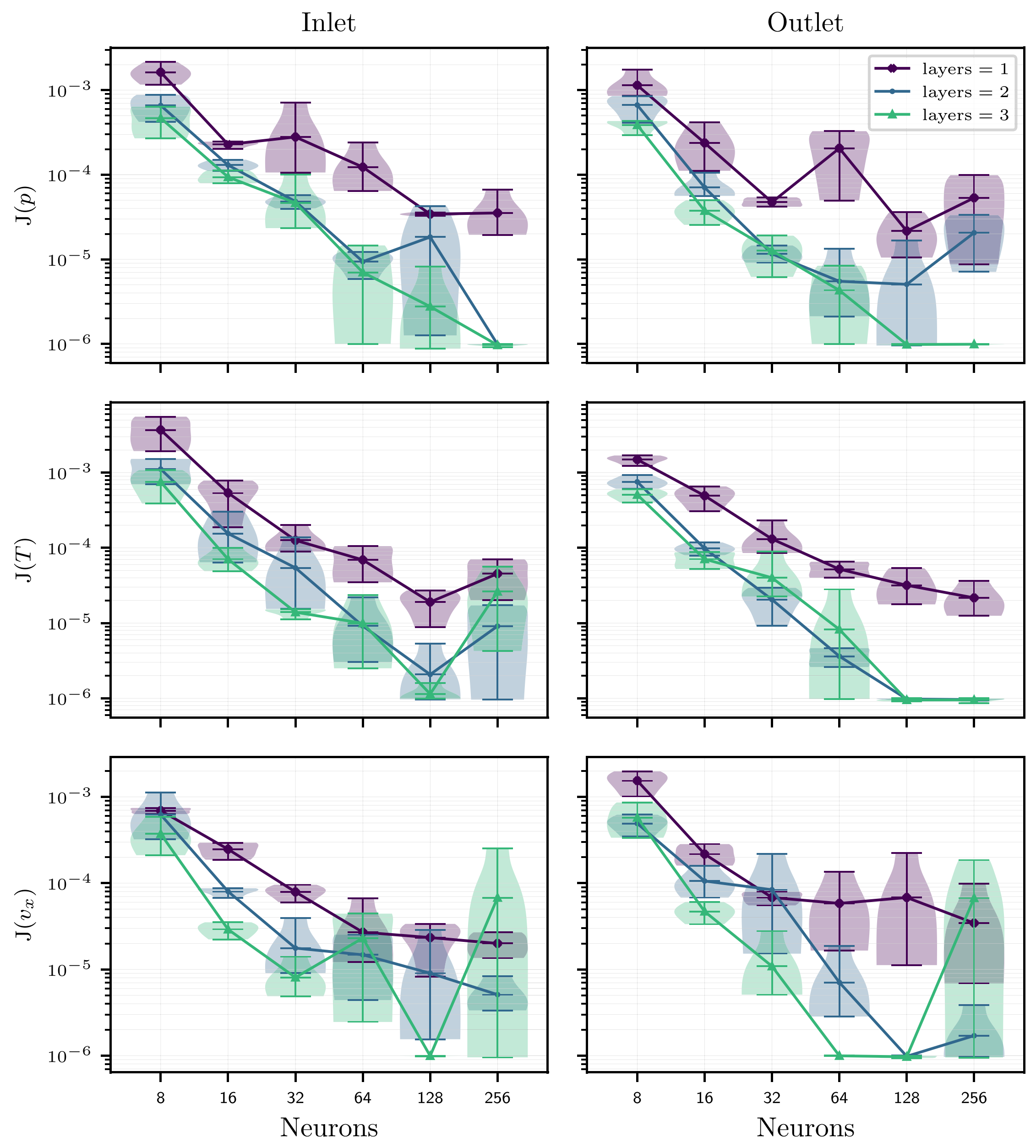


Figure 6. Benchmarking of surrogate model architectures using the training *loss* J as the evaluation metric, defined as the mean squared error between the surrogate model predictions and the training data (M-NN, 35% validation split, $Re = 1000$, $A_i = [0, 0.01]$ m, $k_i = [0, 60]$, 5 random initialisations per architecture). The examples shown correspond to multi-output models, where the benchmark plots display inlet and outlet flow variables, pressure p, temperature T, and axial velocity v_x. For single-output neural network models, the predictions instead are the dissipation rate f_f (3) on the left, and advected heat flux f_h (4) on the right, in place of the flow variables.

The computational cost of a single high-fidelity simulation has a range of $[2,\ 6]$ h, resulting in an average CPU cost of 2400 h for 600 simulations, or 1560 h considering just the training set of 65% the total amount of cases. However, the training cost of this model was around 1600 s, with the inverse problem resulting in CPU time cost of ≈ 30 s, and each prediction has a cost of $\lll 1$ s. The net power output values found in the high-fidelity data prior to using the surrogate model were $\overline{\dot{Q}} = 4.79$ compared to $\overline{\dot{Q}} = 5.818$, over 100% increase in power output optimisation within low dissipation rate regime $\overline{f_f}$.

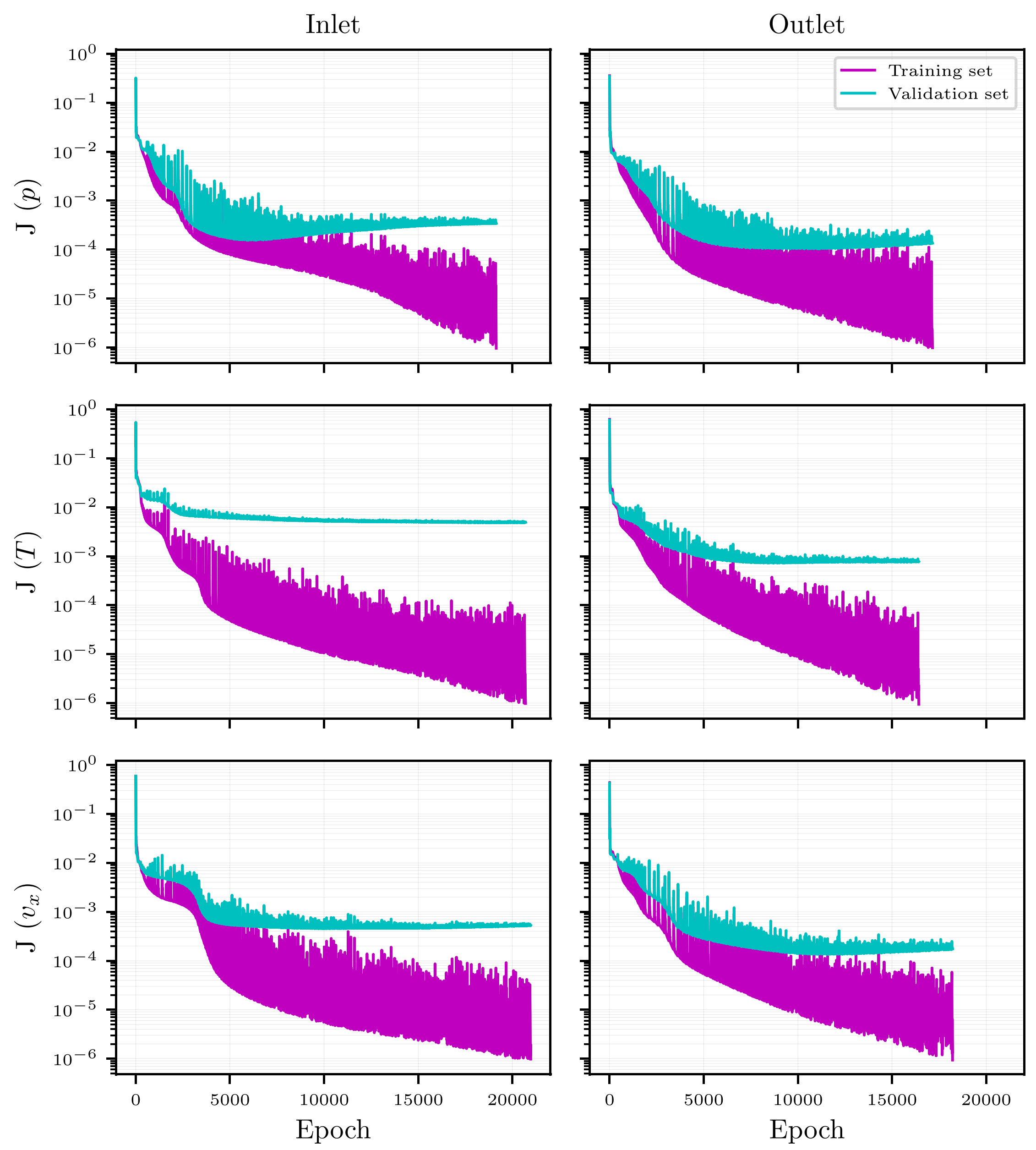


Figure 7. Surrogate models training and validation *loss J*, defined as the mean squared error between the surrogate model predictions and the training/validation data (M-NN, 3 layer, 128 neurons, 35% validation split, $Re = 1000$, $A_i = [0, 0.01]$ m, $k_i = [0, 60]$). The example shown corresponds to an M-NN, which shares the display with S-output models where the loss plots show inlet and outlet flow variables, pressure p, temperature T, and axial velocity v_x. For models with L-output, the predictions instead are the dissipation rate f_f on the left, and advected heat flux f_h on the right.

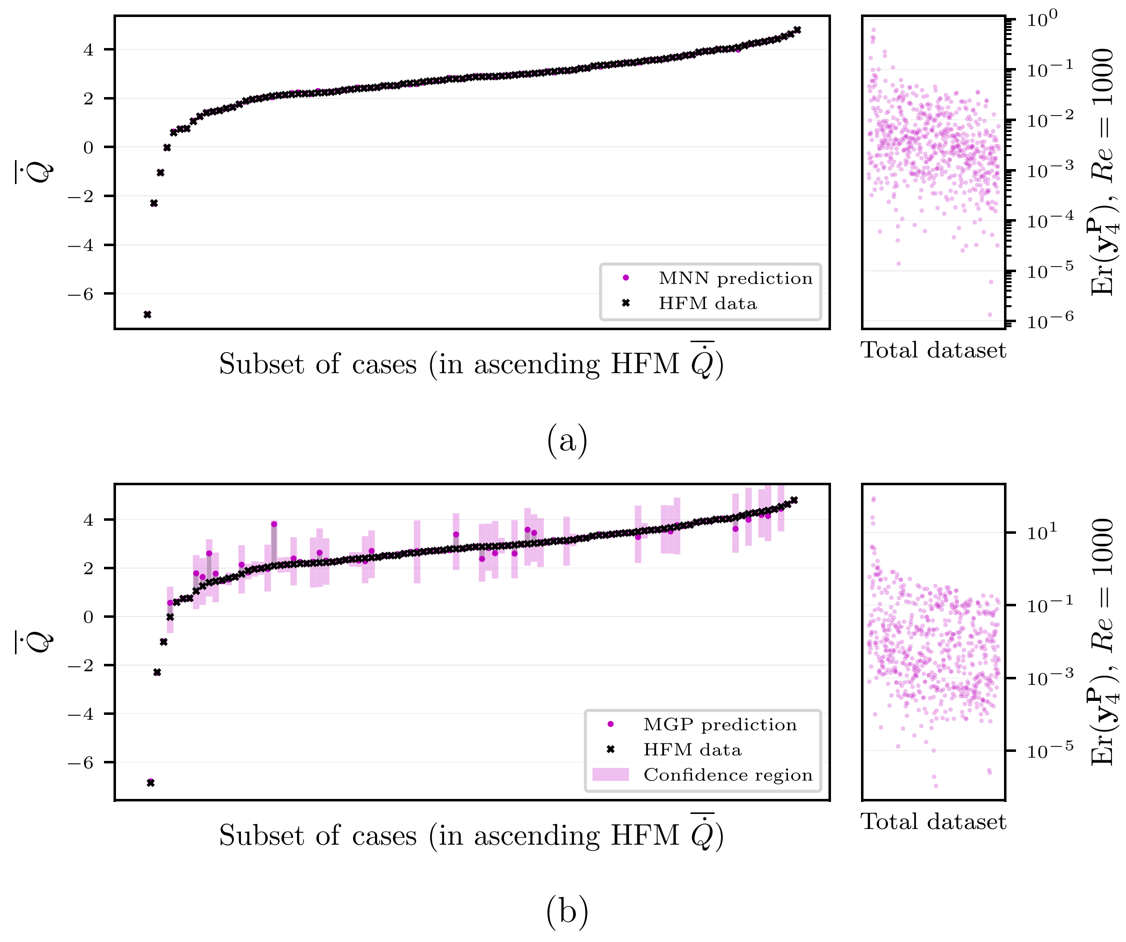


Figure 8. Left plot: THP $\overline{\dot{Q}}$ surrogate model prediction (pink dot) with the corresponding high-fidelity data (black cross), from a subset of 100 sampled cases in ascending value of $\overline{\dot{Q}}$ on models trained at 35% validation split, $Re = 1000$, $A_i = [0, 0.01]$ m, $k_i = [0, 60]$ with algorithm and architecture (a) M-NN, 3 layers with 128 neurons each and (b) M-GP, Matern kernel. GP predicts the *confidence region* from the normal distribution of the kernel (semi-transparent pink bars) alongside the *mean* predictions (pink dots), while NN and RBF report mean predictions only. Right plot: Relative error between the THP $\overline{\dot{Q}}$ surrogate model prediction and the corresponding high-fidelity data, for all available cases.

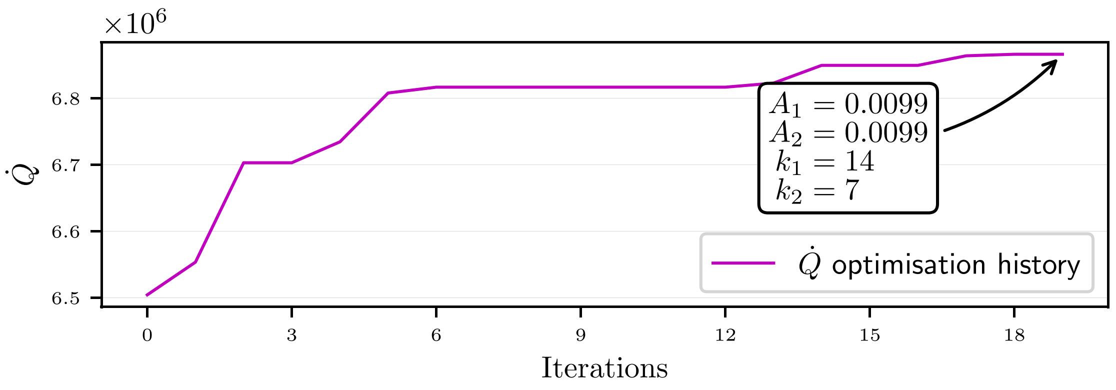


Figure 9. THP $\dot{Q}$ maximisation history plot (M-NN, 3 layers, 128 neurons per layer, 35% validation split, $Re = 1000$, $A_i = [0, 0.01]$ m, $k_i = [0, 60]$). The optimal features corresponding to the maximum $\dot{Q}$ are also reported: for this model, as $A_1 = 0$, the harmonic does not contribute to the geometry of the pipe roughness, regardless of the associated value of the k_1 parameter.

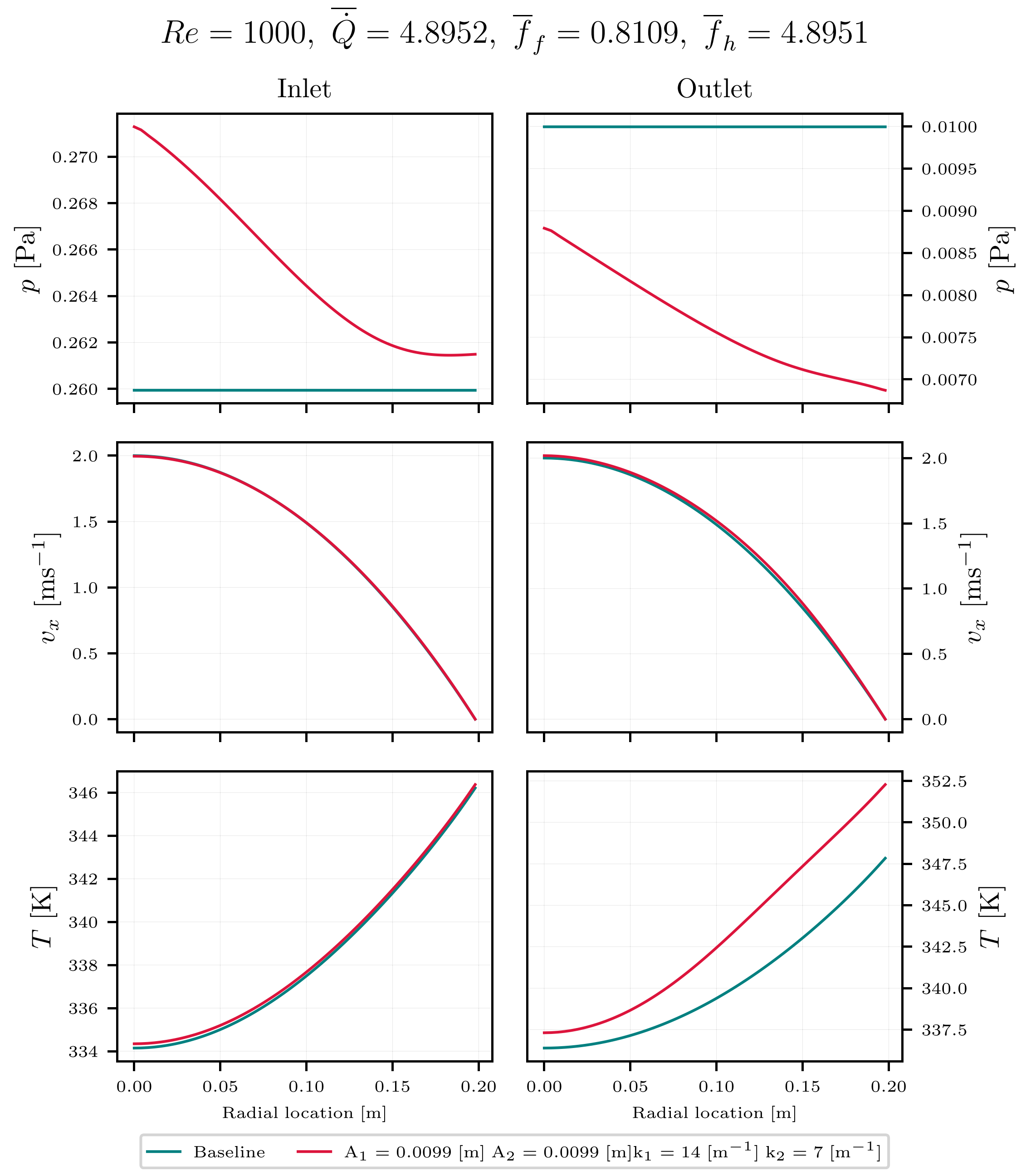


Figure 10. From top to bottom, profile plots of pressure p, stream-wise velocity v_x and temperature T fields of 2D flow over a rough surface plate with highest $\dot{Q}_m$ vs baseline $\dot{Q}_0$ predicted by the M-NN surrogate model at $Re = 1000$.

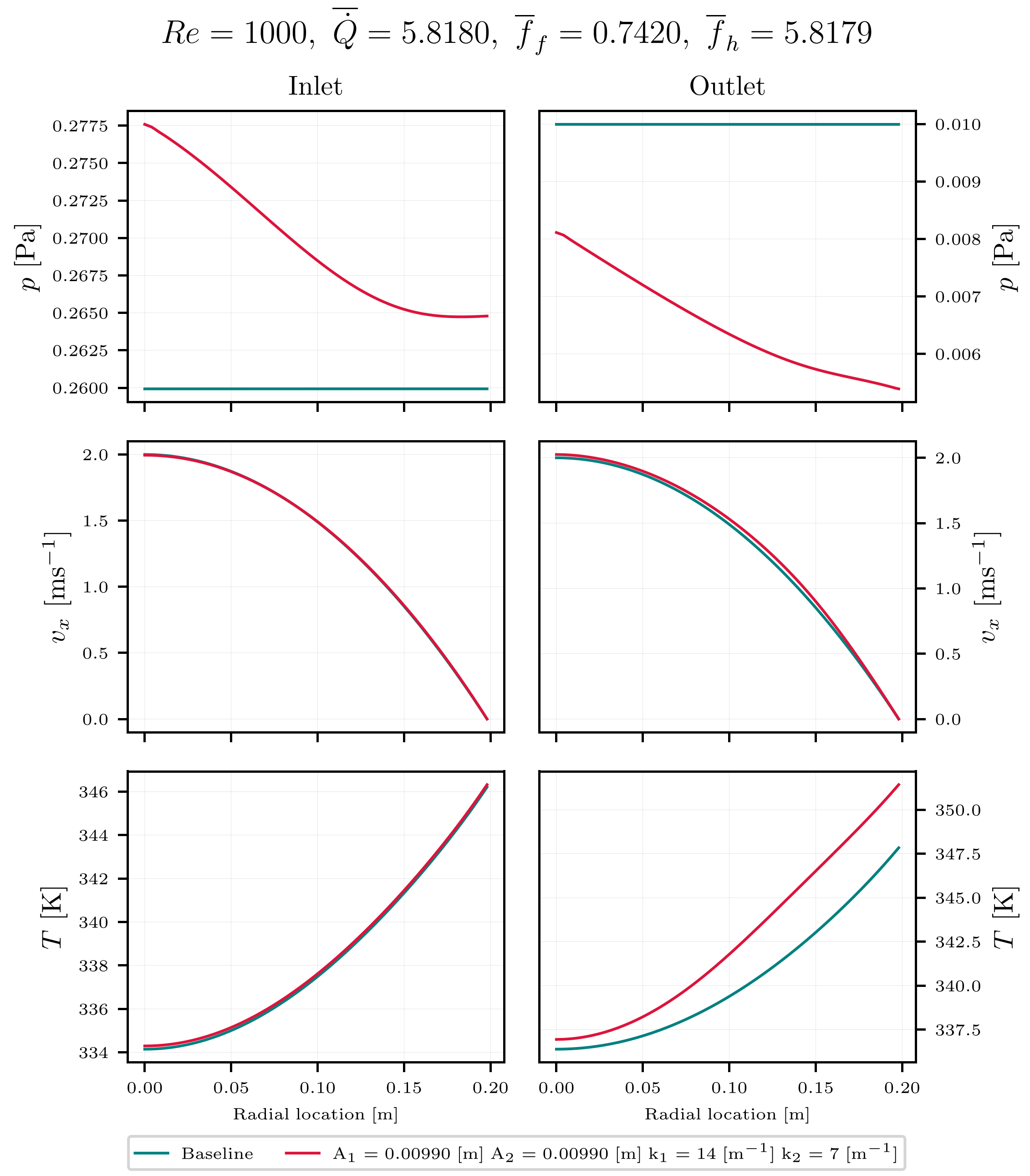


Figure 11. From top to bottom, high fidelity verification of profile plots of pressure p, stream-wise velocity v_x and temperature T fields of 2D flow over a rough surface at $Re = 1000$.

5. Conclusions

This paper has introduced Hammerhead, an open-source software package designed to automate ML-aided heat and mass transfer optimisation in pipe systems. Developed in Python 3, Hammerhead bridges the CFD and ML domains by automating the generation of a high-fidelity database and integrating it into the training of modular surrogate models. It also provides functionality for benchmarking the performance of different surrogate modelling approaches. The high-fidelity database is generated in OpenFOAM `chtMultiRegionSimpleFoam`, leveraging the body-fitted mesh generation capabilities of `blockMesh`, which facilitate seamless integration and automation with Python. Surrogate models are trained using RBF, NN, or GP algorithms, with Hammerhead's modular ML classes supporting multiple architectures. To reduce the risk of ill-posed predictions, the training dataset is pre-allocated to include the parametric space boundaries as an *a priori* constraint, independent of the randomised validation split. This results in an efficient and unified framework for optimisation and surrogate model development, while reducing the time required for mesh generation and high-fidelity data production.

With the aid of surrogate modelling we were able to find a geometry with $\dot{Q}$ optimisation of over 500% while the maximum $\dot{Q}$ optimisation observed with high-fidelity on the database of 600 cases was around 480%, resulting in a computational cost of 2400 h. This suggest an offline CPU gain of 35% for the example analysed here. The in-depth analysis of the high-fidelity data from the exploratory study, as well as a full benchmark of the surrogate models used in the optimisation process were carried out in [39] and will be presented in a follow-up article.

5.1. Potential impact

The production of Hammerhead gives rise to potential implications:

- Fully integrated CFD-ML framework for data generation with modular multi-surrogate THP optimisation.
- Open-source and modular alternative to commercial tools, giving users full transparency and control over the modelling process.
- Automated high-fidelity database generation with body-fitted meshes based on shape/flow parametrisation and CFD execution, both through seamless integration with OpenFOAM, enabling reproducible high-fidelity simulations without user intervention.
- Multi-surrogate modelling with modular architecture, supporting interchangeable surrogate models, including POD-based feature extraction for pre-processing of the data, RBF interpolation, NNs, and GPs, allowing users to tailor the surrogate strategy to the problem and benchmark their approach within the same workflow.
- Parametric knowledge based modelling, designed to prioritise the boundaries of the parameter space to reduce data bias and improve the robustness and generalisation of surrogate models.
- HPC-ready and user-friendly interfaces, including a graphical interface for interactive exploration and a console interface suitable for HPC environments.
- End-to-End THP optimisation workflow, providing tools for THP evaluation and inverse analysis for parameter optimisation based on the high-fidelity and surrogate modelling approximations.
- Designed for community-driven development, built entirely in Python and openly accessible, Hammerhead offers a flexible foundation that can be extended to new geometries, flow regimes, surrogate models, and external software with minimal user cost.

5.2. Limitations and future work

At present, Hammerhead has limited compatibility: it is specifically tailored to OpenFOAM `chtMultiRegionSimpleFoam` solver and does not natively support alternative solvers, mesh configurations, or geometries beyond those defined by the implemented harmonic function, unless modifications are made to the source code. Although the ML algorithms and parametric space are modular, their configurations are semi-fixed and predefined within the software. Extending compatibility to a broader range of high-fidelity modelling methods and surrogate approaches across higher-dimensional parameter spaces will be the focus of future research.

Furthermore, although Hammerhead can identify optimal geometries, it does not yet update the high-fidelity database automatically with these optimised parameters. Instead, the user must manually run additional simulations for verification, and subsequently retrain the surrogate model if further optimisation is required. Incorporating this feature would further reduce the computational cost of the design process. Overall, Hammerhead represents an important step toward a generalised toolkit for automated, ML-aided engineering design optimisation-one that is intended to evolve into a method-agnostic framework in future developments.

Data availability statement

No new data was created or analysed in this study. Data pertaining verification and validation of the high-fidelity models as well as the use case of the software is available upon user request. The source code is available on GitHub at https://github.com/Dani-Darko/Hammerhead [60] (commit `cd59923` at date of manuscript submission).

Acknowledgments

The research was developed through the ESPRC DTP ADDED project with the financial support of RCUK Energy Programme, Grant Number EP/T012250/1, and ERDF via the Welsh Government.

Author contributions

D M Segura Galeana 0009-0007-0758-0262
Conceptualization (lead), Formal analysis (equal), Investigation (lead), Methodology (lead), Software (lead), Validation (lead), Writing – original draft (lead)

A Liptak 0000-0003-1597-4193
Software (supporting), Visualization (supporting), Writing – review & editing (equal)

A J Gil 0000-0001-7753-1414
Conceptualization (supporting), Funding acquisition (lead), Project administration (lead), Resources (supporting), Supervision (lead), Writing – review & editing (equal)

M G Edwards
Conceptualization (supporting), Supervision (supporting)

A Davis 0000-0003-4397-0712
Funding acquisition (lead), Project administration (supporting), Supervision (supporting)

References

[1] Marck G, Nemer M and Harion J-L 2013 Topology optimization of heat and mass transfer problems: laminar flow *Numer. Heat Transfer B* **63** 508–39
[2] Othmer C 2008 A continuous adjoint formulation for the computation of topological and surface sensitivities of ducted flows *Int. J. Numer. Methods Fluids* **58** 861–77
[3] Subramaniam V, Dbouk T and Harion J-L 2019 Topology optimization of conjugate heat transfer systems: a competition between heat transfer enhancement and pressure drop reduction *Int. J. Heat Fluid Flow* **75** 165–84
[4] Soleimani S and Eckels S 2021 A review of drag reduction and heat transfer enhancement by riblet surfaces in closed and open channel flow *Int. J. Thermofluids* **9** 100053
[5] Abdulbari H A, Mahammed H D and Hassan Z B Y 2015 Bio inspired passive drag reduction techniques: a review *ChemBioEng Rev.* **2** 185–203
[6] Rahman M M 2000 Measurements of heat transfer in microchannel heat sinks *Int. Commun. Heat Mass Transfer* **27** 495–506
[7] Moukalled F, Mangani L and Darwish M 2016 *The Finite Volume Method in Computational Fluid Dynamics* vol 113 (Springer)
[8] Zienkiewicz O, Taylor R and Fox D 2014 *The Finite Element Method for Solid and Structural Mechanics* (Elsevier)
[9] Hastie T, Tibshirani R and Friedman J 2009 *The Elements of Statistical Learning* vol 27, 2nd edn (Springer)
[10] Zheng S, Li S, Hu C and Li W 2024 Performance optimization of latent heat storage device based on surrogate-assisted multi-objective evolutionary algorithm and cfd method *J. Energy Storage* **99** 113338
[11] Molinaro R, Singh J-S, Catsoulis S, Narayanan C and Lakehal D 2021 Embedding data analytics and cfd into the digital twin concept *Comput. Fluids* **214** 104759
[12] Balla K, Sevilla R, Hassan O and Morgan K 2021 An application of neural networks to the prediction of aerodynamic coefficients of aerofoils and wings *Appl. Math. Modelling* **96** 456–79
[13] Knight B G and Maki K J 2024 Fast-running CFD calculations of the ONR tumblehome performing maneuvers in calm water and in waves using a Gaussian process regression propeller and rudder model *Ocean Eng.* **303** 117671
[14] Lakshmanan K, Tessicini F, Gil A J and Auricchio F 2023 A fault prognosis strategy for an external gear pump using machine learning algorithms and synthetic data generation methods *Appl. Math. Modelling* **123** 348–72
[15] Conrad F, Wiemer H and Ihlenfeldt S 2025 Leveraging multi-task learning regressor chains for small and sparse tabular data in materials design *Mach. Learn.: Sci. Technol.* **6** 015045
[16] Kedharnath A, Kapoor R and Sarkar A 2024 Prediction of flow stress of ta-w alloys using machine learning *Mach. Learn.: Sci. Technol.* **5** 045007
[17] Meinecke S, Selig M, Köster F, Knorr A and Lüdge K 2024 Data-driven acceleration of multi-physics simulations *Mach. Learn.: Sci. Technol.* **5** 045011
[18] Davariashtiyani A, Wang B, Hajinazar S, Zurek E and Kadkhodaei S 2024 Impact of data bias on machine learning for crystal compound synthesizability predictions *Mach. Learn.: Sci. Technol.* **5** 040501
[19] Langer M F, Pozdnyakov S N and Ceriotti M 2024 Probing the effects of broken symmetries in machine learning *Mach. Learn.: Sci. Technol.* **5** 04LT01
[20] Optimize cfd simulations with just a click (available at: www.ansys.com/en-gb/blog/optimize-cfd-simulations-just-click) (Accessed 13 November 2025)
[21] Fidelity cfd platform | cadence (available at: www.cadence.com) (Accessed 13 November 2025)
[22] Converge cfd software (available at: https://convergecfd.com/) (Accessed 13 November 2025)
[23] Sterling T, Brodowicz M and Anderson M 2018 *High Performance Computing* (Elsevier Science)

[24] Weller H G, Tabor G, Jasak H and Fureby C 1998 A tensorial approach to computational continuum mechanics using object-oriented techniques *Comput. Phys.* **12** 620–31

[25] Chatterjee A 2000 An introduction to the proper orthogonal decomposition *Curr. Sci.* **78** 808–17

[26] Buhmann M D 2003 *Radial Basis Functions* 1st edn (Cambridge University Press)

[27] Linka K, Hillgärtner M, Abdolazizi K P, Aydin R C, Itskov M and Cyron C J 2021 Constitutive artificial neural networks: a fast and general approach to predictive data-driven constitutive modeling by deep learning *J. Comput. Phys.* **429** 110010

[28] Miah S, Sooriyakanthan Y, Ledger P D, Gil A J and Mallett M 2023 Reduced order modelling using neural networks for predictive modelling of 3d-magneto-mechanical problems with application to magnetic resonance imaging scanners *Eng. Comput.* **39** 4103–27

[29] Bousquet O, Luxburg U and Ratsch G 2004 *Advanced Lectures on Machine Learning* vol 3176, ed O Bousquet, U von Luxburg and G Rätsch (Springer)

[30] Ellmer N, Ortigosa R, Martínez-Frutos J and Gil A J 2024 Gradient enhanced Gaussian process regression for constitutive modelling in finite strain hyperelasticity *Comput. Methods Appl. Mech. Eng.* **418** 116547

[31] Ellmer N, Ortigosa R, Martínez-Frutos J, Poya R, Sienz J and Gil A J 2024 Stretch-based hyperelastic constitutive metamodels via gradient enhanced Gaussian predictors *Comput. Methods Appl. Mech. Eng.* **432** 117408

[32] Renze P and Akermann K 2019 Simulation of conjugate heat transfer in thermal processes with open source cfd *Chem. Eng.* **3** 59

[33] Hirsch C 2007 *Numerical Computation of Internal and External Flows* vol 1, 2nd edn (Elsevier)

[34] Pippard A B 1966 *Elements of Classical Thermodynamics: For Advanced Students of Physics* 5th edn (Cambridge University Press)

[35] Hesch K, Boccaccini L V and Stieglitz R 2018 Blankets-key element of a fusion reactor-functions, design and present state of development *Kerntechnik* **83** 241–50

[36] Jackson D, Selander W, Townes B, Leung T, Miller J, Verrall R, Tapping R, Geiger J, Tatone O, Carlick E, Garvey P, Holtslander W, Hastings I and Lane A 1985 A review of fusion breeder blanket technology, part 1 Canadian Fusion Fuels Technology Project *Technical Report* (available at: https://inis.iaea.org/records/6rx6v-xg485) (Accessed January 1985)

[37] Schlichting H and Gersten K 2017 *Boundary-Layer Theory* 7th edn, ed F J Cerra (Springer)

[38] Patankar S and Spalding D 1972 A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows *Int. J. Heat Mass Transfer* **15** 1787–806

[39] Galeana D M S 2025 Computational heat transfer optimisation in nuclear fusion reactors using data-driven machine learning *PhD Dissertation* (Swansea University)

[40] Virtanen P *et al* 2020 Scipy 1.0: fundamental algorithms for scientific computing in python *Nat. Methods* **17** 261–72

[41] Ansel J *et al* 2024 Pytorch 2: faster machine learning through dynamic python bytecode transformation and graph compilation *Proc. of the 29th ACM Int. Conf. on Architectural Support for Programming Languages and Operating Systems, Volume 2* (ACM) pp 929–47

[42] Gardner J R, Pleiss G, Bindel D, Weinberger K Q and Wilson A G 2023 Gpytorch: Blackbox matrix-matrix Gaussian process inference with GPU acceleration *Technical Report* (available at: https://gpytorch.ai)

[43] Buljak V 2012 *Inverse Analyses With Model Reduction* ed K J Bathe (Springer)

[44] Liang Y C, Lee H P, Lim S P, Lin W Z, Lee K H and Wu C G 2002 Proper orthogonal decomposition and its applications-part i: Theory *J. Sound Vib.* **252** 527–44

[45] Ballarin F, D'Amario A, Perotto S and Rozza G 2019 A pod selective inverse distance weighting method for fast parametrized shape morphing *Int. J. Numer. Methods Eng.* **117** 860–84

[46] Bui-Thanh T, Damodaran M and Willcox K 2004 Aerodynamic data reconstruction and inverse design using proper orthogonal decomposition *AIAA J.* **42** 1505–16

[47] Seoane M, Ledger P D, Gil A J, Zlotnik S and Mallett M 2020 A combined reduced order-full order methodology for the solution of 3d magneto-mechanical problems with application to magnetic resonance imaging scanners *Int. J. Numer. Methods Eng.* **121** 3529–59

[48] Toit D W 2008 Radial basis function interpolation *PhD Dissertation* (University of Stellenbosch)

[49] Wang Y, Yu B, Cao Z, Zou W and Yu G 2012 A comparative study of pod interpolation and pod projection methods for fast and accurate prediction of heat transfer problems *Int. J. Heat Mass Transfer* **55** 4827–36

[50] Hagan M T, Demuth H B, Beale M H and Jesús O D 2014 *Neural Network Design* 2nd edn (Martin Hagan)

[51] Shalev-Shwartz S and Ben-David S 2014 *Understanding Machine Learning* (Cambridge University Press)

[52] Bishop C M 2013 *Pattern Recognition and Machine Learning* vol 2 (Springer)

[53] Rasmussen C E and Williams C K I 2006 *Gauss. Process. Mach. Learn.* **11** 32–46

[54] Wang J 2023 An intuitive tutorial to gaussian process regression *Comput. Sci. Eng.* **25** 4–11

[55] Weiss J 2019 A tutorial on the proper orthogonal decomposition *Aiaa Aviation 2019 Forum* (American Institute of Aeronautics and Astronautics) pp 1–21

[56] Fasshauer G E 2007 *Meshfree Approximation Methods With Matlab* vol 6 (World Scientific)

[57] Nilsson N J 1998 Introduction to machine learning, an early draft of a proposed textbook *PhD Dissertation* Stanford University pp 387–99 (available at: https://ai.stanford.edu/ nilsson/MLBOOK.pdf)

[58] Jones D R 2001 A taxonomy of global optimization methods based on response surfaces *J. Global Optim.* **21** 345–83

[59] Rbfinterpolator-scipy v1.14.1 manual (available at: https://docs.scipy.org/doc/scipy/reference/generated/scipy.interpolate.RBFInterpolator.html) (Accessed 23 December 2024)

[60] Segura Galeana D M 2025 Hammerhead (available at: https://github.com/Dani-Darko/Hammerhead/tree/cd599235f8005bc3fbdd70fdc295c2d6526f9087) (Accessed14 October 2025)