Computers and Fluids 214 (2021) 104759

Contents lists available at ScienceDirect

Computers and Fluids

journal homepage: www.elsevier.com/locate/compfluid

Embedding data analytics and CFD into the digital twin concept

Roberto Molinaro[a], Joel-Steven Singh[b], Sotiris Catsoulis[c], Chidambaram Narayanan[b], Djamel Lakehal[b,*]

[a] ETH Zurich, Seminar for Applied Mathematics, Zurich, Switzerland
[b] AFRY Switzerland, Advanced Modelling & Simulation AMS, Zurich, Switzerland
[c] ETH Zurich, Computational Science & Engineering Laboratory, Zurich, Switzerland

ARTICLE INFO

Article history:
Received 19 May 2020
Accepted 6 July 2020
Available online 9 October 2020

Keywords:
Fluid flow simulations
Data analytics
Machine learning
Data-driven models

ABSTRACT

Computer-Aided Engineering (CAE) has supported the industry in its transition from trial-and-error towards physics-based modelling, but our ways of treating and exploiting the simulation results have changed little during this period. Indeed, the business model of CAE centers almost exclusively around delivering base-case simulation results with a few additional operational conditions. In this contribution, we introduce a new paradigm for the exploitation of computational physics data, consisting in using machine learning to enlarge the simulation databases in order to cover a wider spectrum of operational conditions and provide quick response directly on field. The resulting product from this hybrid physics-informed and data-driven modelling is referred to as Simulation Digital Twin (SDT). While the paradigm can be equally used in different CAE applications, in this paper we address its implementation in the context of Computational Fluid Dynamics (CFD). We show that the generation of Simulation Digital Twins can be efficiently accomplished with the combination of the CFD tool *TransAT* and the data analytics platform *eDAP*.

1. Introduction

The fulgurant progress in data science will play an important role besides the traditional scientific computing branch, and may even outperform it in some niche areas where the CAE models are limited in their predictive performance, or simply computationally expensive. A new activity has recently appeared consisting of deploying free-to-use data in order to predict trends using machine learning, with applications covering market analysis as well as operational maintenance and process optimization in industry.

Machine learning has recently found a fertile ground also in the broad field of fluid mechanics due to the large volumes of data. The main source of fluid data is direct measurements conducted in laboratories or from in-situ sensors. The other source of data in the field is obviously CFD. The volume of simulation data has been increasing at unpreceded paces and it is in this particular segment where ML is gaining in importance.

In a recent review paper [2], the authors note that ML has been used in fluid mechanics in two main directions: (I) fundamental understanding and practical prediction via modeling and simulations, and (II) flow optimization and control. In flow modelling and simulation (I), ML algorithms have been used to (Ia) identify and extract specific flow features and topologies, and to (Ib) predict the flow dynamics. In (Ia), the so-called dimensionality reduction is employed in order to mimic the key flow features and its dominant patterns [3]. In (Ib), on the other hand, ML algorithms are used to learn the solutions of ODEs and PDEs governing the system [4–7], including deriving discretization and solver parameterization, increasing the efficiency of the numerical methods, accelerating the resolution of the PDEs [8,9], or to reduce the computational cost of large-scale problems [10–12] such as uncertainty quantification, shape optimization, inverse problem, etc, and even to help find the best set of computational parameters and stabilize the convergence of the simulation [13]. In the same context, ML can be used to help upgrade closure laws for the pertinent physics, including turbulence [14], boiling heat transfer [15], etc.

In flow optimization and control (II), ML algorithms have been used under various forms and methodologies, e.g. genetic algorithms, neural networks, stochastic control, etc [16].

The categorization can be expanded to add a third pillar (III), with reference to establishing data-driven models (DDMs) using field data, with the objective of accurately mimicking the underlying physics controlling field assets. However, leveraging on field data for the purpose of digitally duplicating every system is not straightforward: raw data are noisy, often incomplete, and require

* Corresponding author.
E-mail address: djamel.lakehal@afry.com (D. Lakehal).

https://doi.org/10.1016/j.compfluid.2020.104759
0045-7930/

considerable pre-processing before becoming useful for the construction of DDMs.

In the present contribution, we propose to complement the third pillar by exploiting simulation data instead in order to build approximate models which allows *on-line* extrapolation of Figures of Merit (FoM) for a broader range of operational conditions compared to what is achievable with CFD alone, without necessarily having to solve the underlying system of equations. The paradigm has been embedded in what we refer to as *Simulation Digital Twins*.

FoMs are processed for selected operational conditions and the collection over all the simulations is stored in a database, denoted here as *Synthetic Flow Databases*, for further use as input data for predictive modelling. We have developed for this purpose a dedicated data analytics tool, *eDAP*, to facilitate building databases from CFD results and perform data modelling, using various types of machine learning algorithms. The workflow includes two main steps: physics-driven simulations using *TransAT* CFD tool [17] and its Parametric Optimizer, combined with data driven modelling enabled by *eDAP*.

The paradigm proposed here deserves a comparison with the Model Order Reduction (MOR) that has proven lots of theoretical and practical success at solving similar tasks [3]. One should indeed consider that the evaluation of a trained learning model for new inputs is extremely fast when compared to the resolution of the underlying system of differential equations, allowing real-time response on field. Moreover, this approach, not only applies to any type of problem (non-linear and multi-dimensional) but also allows the design of very accurate approximate models. On the other hand, for nonlinear systems, MOR techniques are not well developed, and some of the most commonly used ones (i.e. POD/Galerkin models) are particularly expensive, and might result in poor surrogates. Finally, the approach described here is completely non-intrusive, and, in fact, it has been developed on top of standalone platforms and can be extended to any computational science context. Therefore, the approach might represent a valuable alternative to the model order reduction. It should be mentioned though that the training procedure is the result of a possibly non-convex optimization problem which is not yet well understood, not guaranteed to converge and, depending on the model capacity, might entail high computational cost.

In summary, our strategy should offer clear advantages, including:

- a straightforward procedure to build physic-informed data-driven models, overcoming some of the issues related to the construction of DDM (such as the collection and pre-preprocessing of the data), thanks to the efficient coupling of tools (*TransAT* CFD, Parametric Optimizer and *eDAP*) conceived exactly for this purpose. To the best of our knowledge, building surrogate models based on CFD data is not offered yet in the data analytics platforms available in the market;
- the design of accurate models embedding either FoMs or the entire flow field for an extremely wide range of operational conditions and accessible on-line for faster decisions;
- cost effective strategy of multiplying initial CFD modelling efforts through rapid deployment of parametric study using automation tools;
- an intelligent way of making use of the simulation data for a comprehensive analysis of the problem.

The paper is organized as follows: Section 2 provides an introduction of the tools used to perform CFD simulations and assemble so-called *Synthetic Flow Databases* serving as the support for the data-analytics part. The next section describes the tools and processes needed for data-driven modelling, including *eDAP*, short for engineering Data Analytics Platform [1]. The paper concludes with the description of the process resulting in Simulation Digital Twins for practical selected CFD applications, including the settling in a secondary water clarifier, the free fall of a cylinder filled with a non-Newtonian fluid, and a fire in a train travelling in a tunnel.

2. Physics-informed modelling

2.1. TransAT CFD/CMFD

CFD is the sub-branch of CAE dealing with fluid flow and thermal processes, wherein transport of mass, momentum and energy is modelled via the following equations:

$$\frac{\partial \rho}{\partial t} + \frac{\partial (\rho u_j)}{\partial x_j} = 0 \tag{1}$$

$$\frac{\partial (\rho u_i)}{\partial t} + \frac{\partial (\rho u_i u_j)}{\partial x_j} = -\frac{\partial p}{\partial x} + \frac{\partial}{\partial x_j}\left(\mu \frac{\partial u_i}{\partial x_j}\right) + S_U \tag{2}$$

$$\frac{\partial (\rho \Phi)}{\partial t} + \frac{\partial (\rho \Phi u_j)}{\partial x_j} = \frac{\partial}{\partial x_j}\left(\Gamma \frac{\partial (\rho \Phi)}{\partial x_j}\right) + S_\Phi \tag{3}$$

where u_i and P are the main fluid-flow variables, namely the velocity and the pressure, and ρ and μ are the fluid density and viscosity, respectively. In the scalar equation, ϕ denotes a generic passive scalar (temperature or concentration), and Γ is the diffusivity coefficient. The last terms in Eqs. (2) and (3) account for sources of momentum and any other quantity, e.g. energy. To treat multiphase flows, the above equations can be modified to reflect the nature of the flow considered [18].

The CFD software *TransAT* is based on finite-volume method, using the Immersed Surfaces Technique (IST) for multidimensional meshing, enhanced with Block-Mesh Refinement (BMR) to refine the mesh near the walls. The tool can be operated on Windows and Linux operating systems [19]. *TransAT* is particularly suitable for complex, multiphase and multicomponent flow systems, with phase change heat transfer. Additional models accounting for specific physical phenomena can be added using User Defined Functions. More details concerning the tool and its features can be found in [17].

2.2. TransAT parametric optimizer

In all CAE disciplines, the sensitivity of the simulation results to minor variations in the operating conditions and material properties is of significant importance. Real applications require tools capable of returning results for more than a few operating conditions. *TransAT* is equipped with a batch tool for multiparametric simulations, significantly reducing the overhead for the user. Parametric simulation studies can therefore be conducted for specification of the physics, the numerical parameters, initial and boundary conditions, material properties, embedded CAD objects, etc. Once the batch of simulations is executed, the tool transfers the selected results automatically to the *eDAP* platform to create databases for analysis and predictive modelling.

3. Data-driven modelling

3.1. eDAP platform architecture

eDAP is a data archival and analysis tool, based on an SQL engine for fast access to information, combined with a user-friendly interface for data analysis, visualization, and predictive modelling. The platform is conceived with the purpose of creating a simple workflow that guides the user from the creation of a database to predictive modelling. The platform is structured in the form of distinct applications created for the analysis of large-scale systems:

each application is filled with data representative of a specific system component and contributes to assembling the database for that specific system.

3.2. Synthetic flow databases

eDAP can deal with all sorts of data for treatment, including those delivered by CFD. In this particular context, fluid flow and thermal processes are first simulated for selected key operating conditions. FoMs are defined and determined for each simulated scenario using appropriate reduction of the flow results. Out of the simulated flow cases, *Synthetic Flow Databases* are created, which encompass the selected FoMs. Depending on the application field, FoM can for example represent the aerodynamics coefficients in aerospace engineering, pressure losses in mechanical engineering, separation efficiency in process engineering, runout length in hydraulics, etc.

3.3. Machine learning

The end objective of *eDAP* is to perform predictive modelling on Synthetic Flow Databases. In particular, we make use of the specific class of *supervised machine learning* to reconstruct or interpolate simulation data and create physics-based data-driven models [21].

The target of machine learning algorithms is to approximate a function

$$f : X \rightarrow Y, \tag{4}$$

$X \subseteq \mathbb{R}^m$, $Y \subseteq \mathbb{R}$, based on a set of samples $\{x_i, y_i\}$, $i = 1 : n$. The vast majority of machine learning algorithms requires the resolution of a minimization problem, reading:

$$\hat{\theta} = \arg\min_{\theta} \mathcal{L}(\theta), \tag{5}$$

where $\mathcal{L}(\theta)$ is a proper loss function and θ the set of parameters defining the approximate model. The loss function mostly used in regression problems is the *mean squared error*:

$$\mathcal{L}(\theta, x) = \sum_{i=0}^{n} (y_i - \hat{y}(x_i, \theta))^2, \tag{6}$$

where $\hat{y}(x_i, \theta)$ is the response predicted by the surrogate model.

As is usual in supervised learning, the available dataset is split into three subsets: *training, validation* and *testing* sets. The entire process of finding the best set of fitting parameters θ is usually denoted as *model training*, and is performed on the training dataset. The model hyperparameters, namely higher-level parameters chosen before the model training, are defined through *cross-validation*. This is usually performed by training the learning model for different hyperparameters configurations and selecting the optimal one according to some *selection criterion*, for instance the value of the loss function on the validation test. Eventually, the performances on the optimal model are evaluated on the testing set. Note that validation and testing sets are two distinct and independent sets, and the latter one is not involved in the selection of the model hyperparameters. The described procedure is applied in the numerical examples reported in Section 5.

It is customary in supervised learning to add a regularization term to the loss function to avoid the overfitting of the data. The optimization problem can be reformulated as follows:

$$\hat{\theta} = \arg\min_{\theta} (\mathcal{L}(\theta, x) + J(\theta)), \tag{7}$$

with $J(\theta) = \lambda ||\theta||^p$, $1 \leq p < \infty$. Usually, L^1 or L^2 regularization ($p = 1$ or 2) is adopted.

The end-user can rely on the following learning algorithms implemented in *eDAP*: (1) Polynomial Regression [20,21], (2) Multivariate Adaptive Regression Spline (MARS) [22] (3) Random Forest [20,21], and (4) Artificial Neural Network [23].

3.3.1. Polynomial regression

Polynomial regression is a parametric algorithm that assumes the existence of a polynomial relationship between the input and the response. In the simple one-dimensional case ($m = 1$) the relationship reads:

$$y(x, \theta) = \theta_0 + \theta_1 x + \ldots + \theta_m x^m, \tag{8}$$

where coefficients θ_i are unknown and need to be estimated. This is achieved by minimizing (6) with

$$\hat{y}(x_i, \theta) = \theta_0 + \theta_1 x_i + \ldots + \theta_m x_i^m. \tag{9}$$

One disadvantage of the polynomial regression is the assumption made about the structure of the mapping function f that makes the model less flexible compared to non-parametric models. However, its simplicity guarantees a better interpretation of the data.

3.3.2. Multi-variate adaptive regression splines

MARS is learning model intended to perform flexible non-linear modelling of highly dimensional data [22]. It is a non-parametric approach that models the function $y = f(X)$ as a sum of basis functions $h(X)$ [20]:

$$f(X) = \beta_0 + \sum_{i=1}^{k} \beta_m h(X). \tag{10}$$

The basis functions can be:

- hinge functions: $h(X) = \max(0, X - c)$ or $h(X) = \max(0, c - X)$, where c is called *knot* point;
- or product of different hinge functions.

Once the basis functions are chosen, the expansion coefficients are computed using standard linear regression. The algorithm consists of two steps:

- *Forward Pass*. At each stage, a new basis functions pair is added. This consists in the product of a term already present in the basis succession and a new hinge function:

  $$\hat{\beta}_{k+1} h_l(X)(X - c) + \hat{\beta}_{k+2} h_l(X)(c - X). \tag{11}$$

  The term $h_l(X)$ resulting in the largest decrease of the training error is chosen. The process is performed until the maximum number of terms is reached or the threshold value of the error achieved.
- *Backward Pass*. The less effective terms are progressively removed until the best model is found using the method of Generalized Cross Validation, or GVC [22].

In practice, we fix a rather low value of the error threshold and we select the optimal maximum value of terms with cross-validation.

3.3.3. Random forest

Random Forest is a decision tree based algorithm that overcomes the problem of data over-fitting in classical decision algorithms. The idea behind can be summarized as follows [20]:

- Divide the training set into B smaller subsets (*bootstrap* technique);
- Build decision trees on each bootstrapped training sample. Each time a split in the tree is considered, a random number of features m, smaller than the total one, is picked, and the split is allowed to use only the extracted features;
- Compute the final predicted value as the average of the values obtained by each bootstrapped sample.

The construction of a decision tree consists of two steps [21]:

- The training space is divided into M non-overlapping regions R_i, $i = 1 : M$;

- For every observation falling in the region R_m the same prediction $\hat{y}_m = \sum_{x_i \in R_m} y_i$ is made.

The algorithm for growing decision trees searches for the optimal space subdivision with a *top-down greedy* approach on each bootstrapped training sample, with the aim of minimizing an *ad-hoc* loss function. In the first place, consider a splitting variable X_j and a splitting point x_s and build the regions R_1 and R_2:

$$R_1(X_j, x_s) = \{X | X_j < x_s\}, \quad R_2(X_j, x_s) = \{X | X_j > x_s\} \tag{12}$$

Then, we search for X_j and x_s as solutions of the optimization problem:

$$\min_{X_j, x_s} \left\{ \min_{c_1} \sum_{x \in R_1(X_j, x_s)} (y - c_1)^2 + \min_{c_2} \sum_{x \in R_2(X_j, x_s)} (y - c_2)^2 \right\}. \tag{13}$$

For any j and s, the inner optimization problem is solved by

$$c_1 = \sum_{x_i \in R_1(X_j, x_s)} y_i, \quad c_2 = \sum_{x_i \in R_2(X_j, x_s)} y_i. \tag{14}$$

The best pair (X_j, x_s) can be determined by running through all the inputs. This process is iterated for each new region, leading to the optimal subdivision of the sample space. The fundamental parameter controlling the performance of the model is the total number of trees employed, mostly tuned with cross-validation.

3.3.4. Neural network

A feed-forward neural network consists of an input layer, an output layer, and different hidden layers [23]. Each layer is defined by a specific number of neurons, connected by synapses that are mathematically modelled by functions σ, called *activation functions* (Fig. 1). In various problems, ReLU function is usually employed, reading:

$$\sigma(x) = \max(0, x). \tag{15}$$

The intricate structure of the network provides a great flexibility that allows to model almost any type of complex function. To describe how a network works, let us consider the architecture shown in Fig. 1, with L different hidden layers.

Each neuron at the layer l is connected to all the neurons of the previous level $l-1$, and the connections are formulated as follows:

$$v_i^{(l)} = \sigma(\xi_i^{(l)}) = \sigma\Big(\underbrace{\sum_{j=1}^{n^{(l)}} W_{ij}^{(l)} v_j^{(l-1)} + b_j^{(l)}}_{\xi_i^{(l)}} \Big) \tag{16}$$

Here, $n(l)$ denotes the number of neurons at layer l, $W \in \mathbb{R}^{n^{(l)} \times n^{(l-1)}}$ is the matrix of weight connecting hidden layer $l-1$ to layer l, and $v^{(l)} \in \mathbb{R}^{n^{(l)}}$ is the vector variable encoded into the neurons of layer l. The connections are feed-forwarded from the input layer up to the output, and, in the end, the output variables $\hat{y}$ are obtained as functions of the input variables and the weights connecting each layer. The model parameters $(W, b) = \theta$ are estimated by minimizing the cost function $\mathcal{L}(\theta, x)$, defined as in (6). The essential element in the optimal design of a neural network is the tuning of the model hyperparameters, including the loss function, the optimization algorithm, the regularization, and the width and depth of the network. Lye et al. [10] recently proved that the careful selection of the model hyperparameters can lead to a high level of compression, minimizing the prediction error. However, the approximation theory currently developed on neural network does not provide explicit information about how an efficient network is to be designed, making the task of constructing it rather challenging.

4. Simulation digital twins

Before elaborating on the machine learning algorithms employed (performance and limitations), we summarize first the workflow leading to what we refer to as Simulation Digital Twin (SDT), or the concept of creating a virtual model using simulation data.

4.1. The workflow

The workflow leading to a typical SDT is described below (see also Fig. 2). For a given thermal-fluid flow problem with defined geometry, fluid properties and flow or operating conditions, the process follows these steps:

1. Create a robust simulation setup for a base condition and simulate and analyze the results to determine the key FoMs. Use this setup as the template for the TransAT Parametric Optimizer;

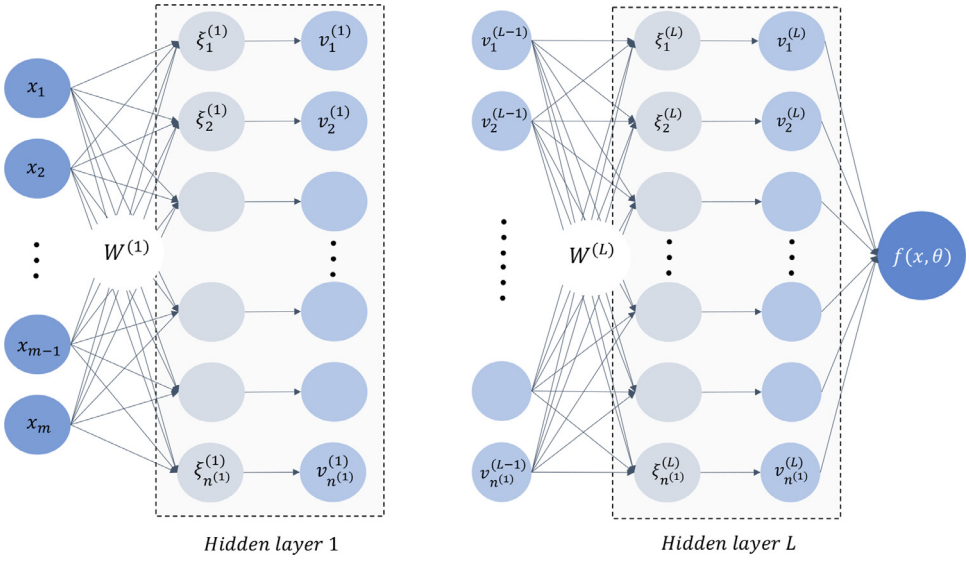


Fig. 1. General artificial neural network architecture.

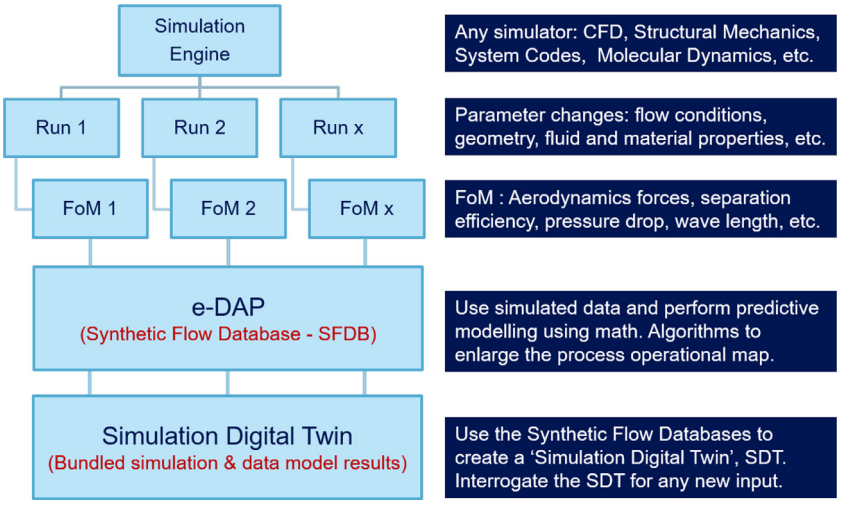


Fig. 2. Workflow for Simulation Digital Twin via Synthetic Flow Databases.

Table 1
Mean absolute error for the test example.

	Streamwise Vel. U	Temp. T
Regression	0.0018	0.04
MARS	0.0053	0.11
Random Forest	0.00015	0.0012
Neural Network	0.0044	0.0042

2. Design the scope of the parametric study and define the study using the Parametric Optimizer;
3. Perform the simulations in the parametric study for the chosen set of conditions;
4. Collect the simulation data into a database. TransAT Parametric Optimizer tool allows to directly collate the data which can be imported into *eDAP*;
5. Apply selected learning algorithms to the Synthetic Flow Databases in order to build SDT.

Once the SDT is built, it can at anytime be updated by adding new simulation data and quickly interrogated for additional operating conditions (Fig. 2).

4.2. ML algorithms testing & validation

In a first step, the ML algorithms available in *eDAP* have been tested for several CFD problems, with variable complexity. The idea behind was to select the most accurate ones for further use in the cases discussed in the application section.

Among these validation cases, we discuss here the results of a 2D heated-channel flow. Different flow scenarios have been simulated with *TransAT* to create its Synthetic Flow Database, using the Reynolds number (Re) as the sole input variable. The objective is to realize a SDT to retrieve velocity and temperature values at any spatial location for any Re: $U = U(x, y, Re)$ and $T = T(x, y, Re)$. Polynomial regression, multivariate regression spline, random forest, and neural network have been tested separately and compared. The models performance are assessed by computing the *mean absolute error* (MAE) over the samples of the testing set (Table 1)[1]:

$$e = \frac{1}{N_{test}} \sum_{i=1}^{N_{test}} (|y_i - \hat{y}_i|). \tag{17}$$

[1] The samples data are normalized between 0 and 1 through a linear transformation.

Polynomial regression results in rather poor generalization capabilities, due to the parametric nature of the model. Interestingly, while MARS has shown a good performance in other cases tested (not discussed here), it behaves worse than simple regression for the 2D channel flow. On the other hand, neural network and random forest return significantly accurate approximations of the functions of interest (further details are reported in [24,25]).

4.3. Selection of models hyperparameters

As mentioned in Section 3.3, the proper training of machine learning algorithms requires the specification of several hyperparameters in order to achieve reasonable accuracy in building a model. To this end, we perform cross-validation as described in Section 3.3. Specifically, cross-validation is used for the selection of the maximum number of terms added in the forward step of MARS and the number of splitting trees for the random forest. Within the context of neural networks, we fix a priori a reference architecture and adopt key results achieved by Lye, Mishra and Ray [10] regarding the selection of regularization type, loss function, optimizer, and learning rate. In this regard, the authors proved that L^2 regularization, ADAM algorithm with learning rate $\eta = 0.01$, and mean squared error loss function are suitable choices for a large variety of similar CFD problems. The tuning of the regularization parameter λ is realized by cross-validation starting from this configuration. In summary, the problem reduces to the selection of one hyperparameter per model. We choose as selection criterion the value of the MAE (17) estimated on the validation set,

$$e = \frac{1}{N_{val}} \sum_{i=1}^{N_{val}} (|y_i - \hat{y}_i|), \tag{18}$$

and select among the hyperparameter configurations reported in Table 2 the one resulting in the lowest value of (18).

Table 2
Configurations of hyperparameters used for cross-validation.

	Model Hyperparamters
Max terms (MARS)	20, 30, 40, 50
Trees (RF)	200, 400, 700, 1000, 1500
Reg. Param. (NN)	10^{-7}, 10^{-6}, 10^{-5}, 10^{-4}

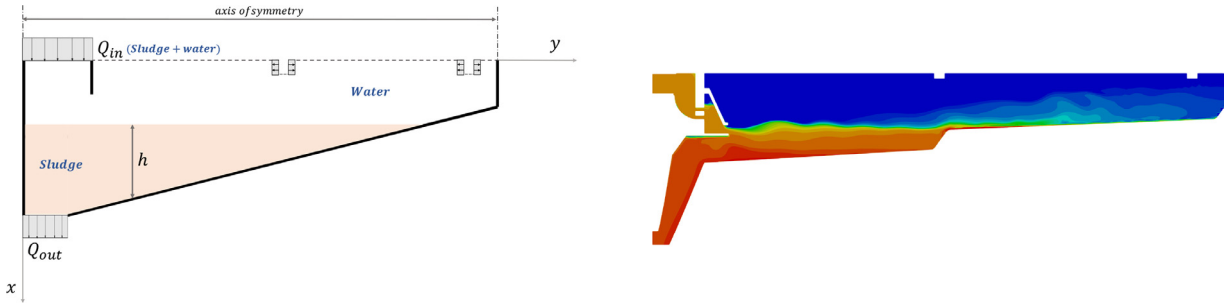


Fig. 3. *Secondary Water Clarifier* case study: schematic configuration Fig. 3a and simulation results of the sludge concentration Fig. 3b.

5. Case studies

We present next three concrete applications realized with the combination of *TransAT* and *eDAP*, by reference to the workflow described in the section above. In light of the previous results, in the following, we only employ MARS, random forests and neural networks, whereas linear regression is used as a reference baseline.

5.1. Secondary water clarifier

The efficiency of secondary water clarifiers is crucial for the overall performance of wastewater treatment plants. Its operation is determined by complex interactions between flow and settling, including stratification and flocculation processes. The success of the clarifier operation is strongly dependent on the flow features, and the relevance of supervised learning to learn outputs, such as the sludge height blanket, as a function of the flow features is evident. The height blanket should always be under control so that the sludge does not reach the free surface and flows aboard the tank.

5.1.1. Flow physics and modelling

The problem has been simulated following to a large extent the workflow described by Lakehal et al. [26]. Meshing is based on the Immersed Surfaces Technology, whereby the geometry is embedded in a Cartesian grid comprising 100 × 268 cells. A sketch of the computational domain is shown in Fig. 3a. Two-dimensional axisymmetric simulations have been performed, using the following boundary conditions (BCs): the inflow massflow rate and concentration are imposed, the pump BC at the bottom are set according to the specified recirculation factor, and pressure BC is fixed at the outlets. Turbulence is modelled using the conventional $k-\varepsilon$ model modified to account for buoyancy effects. The fluid rheology is accounted for using a Bingham type of model.

A typical CFD result of the water clarification and sludge settling in the tank is shown in Fig. 3b. The prediction shows a strong scalar stratification at the bottom of the vessel, near the drainage, and a less compact blanket in the outermost part of it.

5.1.2. Parametric study

The objective is to explore the efficiency of the clarifier for different flow conditions, such as the sludge inflow concentration and the mixture massflow rate. To this end, a parametric study has been performed, with the aid of a total of 30 simulations covering a relatively broad range of conditions: 10 values of the inflow sludge concentration C_{in} equally spaced between 7.125 and 37.5kg/L, and 3 values of the mixture massflow rate Q_{in} ranging from 1.14 to 4.56kg/s.

Table 3
Number of samples used for the numerical examples.

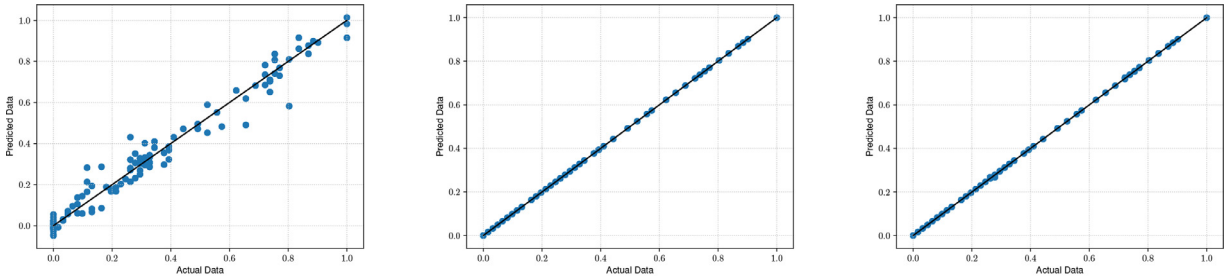
	Training	Validation	Testing
Waterclarifier	9954	2488	1383
Viscous Free Flow	180	20	100
Train in Tunnel	112,076	28,019	1416

5.1.3. Predictive modelling

Here the aim is to explore the relationship between the sludge blanket height and the inflow concentration and massflow rate. MARS, RF and NN have been used for the learning of the map $h = h(y, C_{in}, Q_{in})$, where h denotes the sludge blanket height, defined as the x-location where the molar fraction of sludge becomes negligible (Fig. 3a). The selection of the models hyperparameters is performed according to Section 4.3. We use 72% of the entire dataset to train the model, the 18% for the model selection with the cross-validation, and the remaining 10% as testing set to report the training results (see Table 3, the dataset include also y-spatial Cartesian grid points). For the MARS algorithm, a maximum number of 50 terms, provided by cross-validation, has been used. However, the forward phase concludes before reaching the training error threshold and no improvement is achieved enlarging the maximum number of terms [22]. The underlying mapping exhibits indeed features that MARS is not capable of learning. In the case of the RF, the optimal value of the number of estimators is 200. The best performing neural network model, on the other hand, consists of an architecture of 4 layers and 24 nodes, L^2 regularization and parameter $\lambda = 10^{-6}$. As in the rest of this work, ADAM algorithm with learning rate $\eta = 0.01$ is employed.

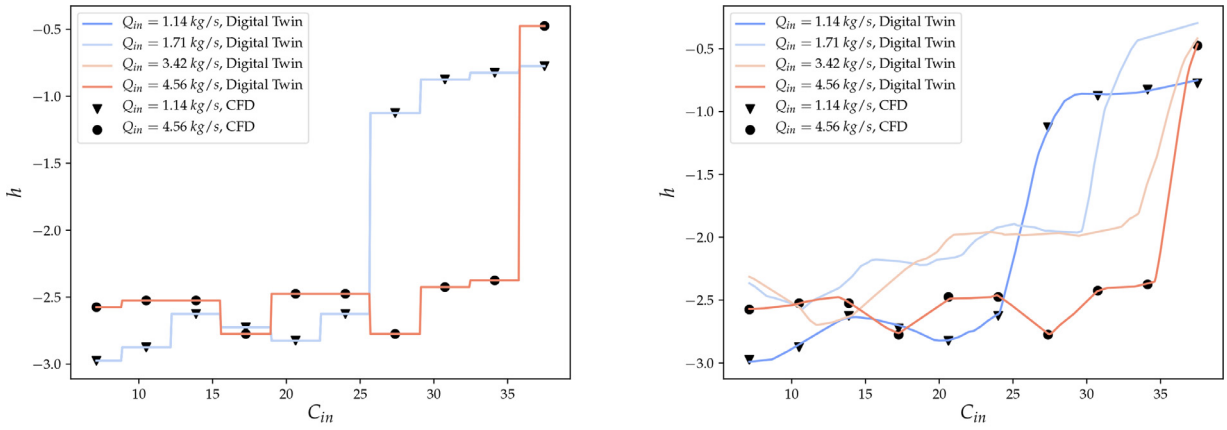
The results of the training for h are shown in Fig. 4. The testing errors for the learning models are 0.0366 for MARS, $7.7{\cdot}10^{-16}$ for the random forest and 0.00048 for the neural network. All of them are rather small compared to the baseline value (0.197). The total elapsed time to perform the entire cross-validation and train the models amounts to 58s, 16s and 2855s, for the three algorithms, respectively, which is extremely low compared to the computational time that the generation of the dataset entails (several hours for one single realization of the operational conditions.) Here, the random forest outperforms the other algorithms in terms of error and computational time, leading to surprisingly accurate solutions. However, a closer analysis of the algorithms reveals an important insight.

NN and RF have been used to perform prediction on new data. The results are shown in Fig. 5a-b for the radial location $y = 5.0471$ m: the continuous lines represent the values predicted by the models, whereas the black dots and triangles correspond to the data stored in the CFD database. We observe that the random forest provides a stepwise representation of the blanket height h



(a) Multivariate Regression Splines **(b)** Random Forest **(c)** Neural Network

Fig. 4. Machine learning algorithms performance in the *Secondary Water Clarifier* case study.



(a) Random Forest **(b)** Neural Network

Fig. 5. The sludge height blanket h is plotted as a function of the sludge inflow concentration C_{in} for different values of the mixture mass flow rate Q_{in} at the radial location $y = 5.0471$. The black triangles and dots correspond to the CFD data stored in the database for $\dot{m} = 1.14$ and 4.56 kg/s. The lines are given by Random Forest model (on the left) and NN model (on the right).

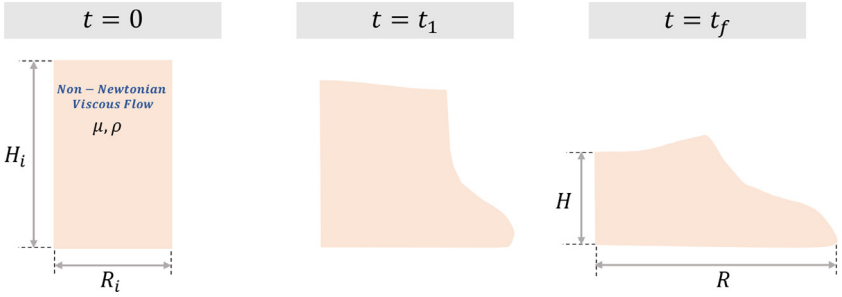


Fig. 6. *Viscous Free Flow of a Cylinder* schematic configuration.

as a function of *both* the inflow flow rate and sludge concentration. In fact, the lines corresponding to flowrates 1.71 kg/s and 3.42 kg/s coincide with the ones corresponding to 1.14 kg/s and 4.56 kg/s, that already exist in the CFD database. In other words, for some input location $(y, C_{in}, Q_{in})^*$ of interest (not contained in the CFD dataset), the model will return the CFD result corresponding to the input point (y, C_{in}, Q_{in}) "closest" to $(y, C_{in}, Q_{in})^*$. Therefore, despite the extraordinary performance of the random forest on the testing set, the algorithm does not provide any additional information compared to the CFD simulations. This is not the case for NN since the model returns an interpolated value between the simulated data.

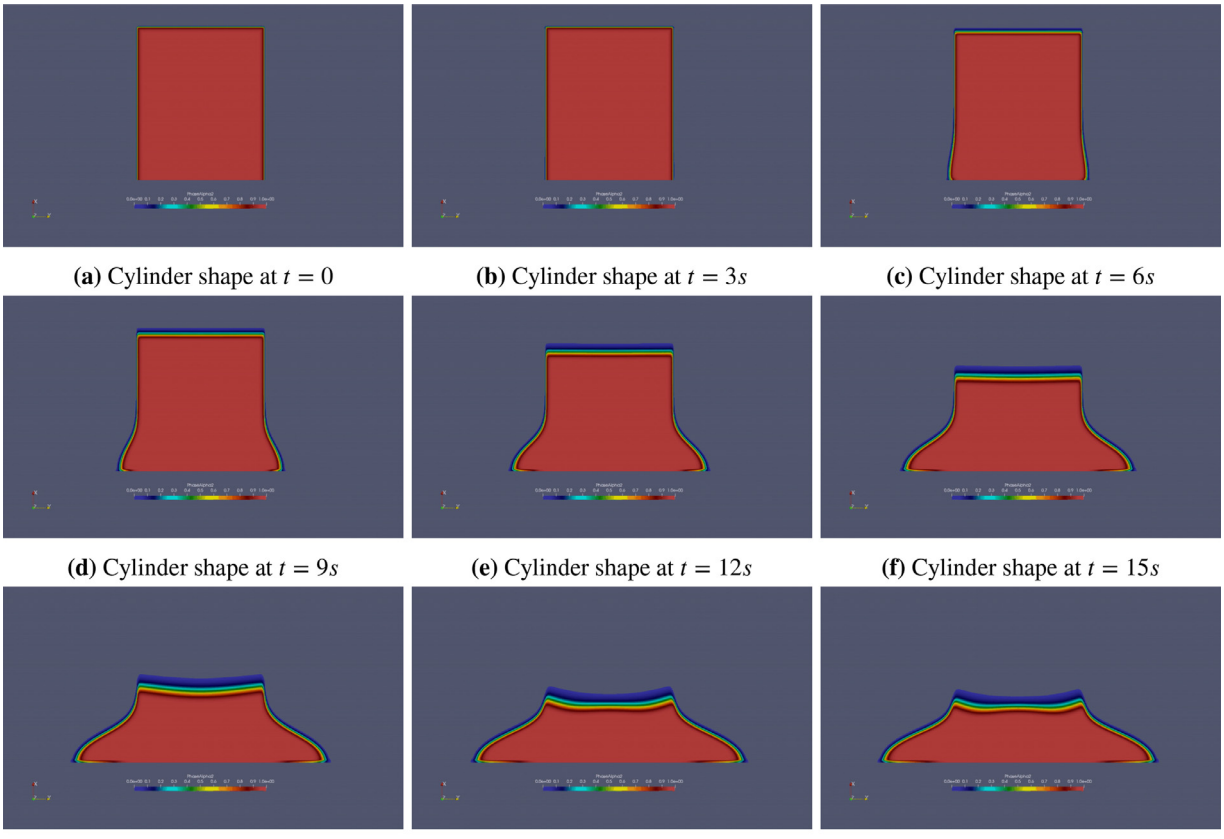
In summary, the user of this result can interrogate the SDT (without performing new CFD simualtions) for any new operational condition, e.g. what would be the loading of the settling tank in case of rains or when inflow has changed due for example to an increase of population in the area..

5.2. Viscous free flow of a cylinder

The case study involves a cylinder made up of a viscous fluid that is initially in standing position, and is then simply let to freely flow until it reaches a final configuration [27].

5.2.1. Flow physics and modelling

The simulations have been carried out by taking advantage of the axisymmetry of the cylinder. The computational domain, shown in Fig. 6, consists of a rectangle of length 0.22 m and height 0.17 m, discretized with a Cartesian grid of 245 × 130 cells. The extent to which the simulations have been carried out is based on a non-dimensional time used across all simulations, indicating that the cylinder has reached steady-state. The boundary conditions are as follows: a no-slip wall boundary condition on the side of the cylinder's base, and symmetry conditions on the rest of the domain's boundaries.



(a) Cylinder shape at $t = 0$ **(b)** Cylinder shape at $t = 3s$ **(c)** Cylinder shape at $t = 6s$

(d) Cylinder shape at $t = 9s$ **(e)** Cylinder shape at $t = 12s$ **(f)** Cylinder shape at $t = 15s$

(g) Cylinder shape at $t = 17s$ **(h)** Cylinder shape at $t = 19s$ **(i)** Cylinder shape at $t = 21s$

Fig. 7. Simulation results for the *Viscous Free Flow of a Cylinder* case study.

Table 4
Experiment-Simulation comparison for similar inputs.

	Final radius [mm]	Final height [mm]
CFD	108.64	42.10
Experiment	118.50	40.00

Table 5
Extremum values for parameter variation, and constants.

	Re	*Fr*	L_r	*Bm*	*n*
Variation boundaries	[50:340]	[1:2]	[1:15]	[1:30]	[0.05:0.95]
Operating conditions	130.0	1.41	2.4	5.46	0.35

The material of the cylinder is modeled according to the non-Newtonian Herschel-Bulkley model, which includes a yield stress τ_0 for initiating the fluid motion:

$$\tau = \tau_0 + K\dot{\gamma}^n,$$

where n is the power-law value, $\dot{\gamma}$ is the shear rate, and K is the consistency coefficient [28]. Gravity is included, but no turbulence modelling is employed here. The homogeneous phase averaging model is used to treat the multiphase flow, where one phase is the material itself and the other is air.

Before proceeding with the parametric study, a "reference" test case was performed using similar inputs as in [27]. The results are shown in the Fig. 7: the fluid body falls and splashes on the surface until it reaches a final position and shape. The comparison between CFD and measured results is highlighted in Table 4.

5.2.2. Parametric study

The objective of this study is to explore the I/O behavior of the cylinder on account of several different input parameters, such as initial height H_i, initial radius R_i, material density ρ, K, n and τ_0. However, in order to reduce the complexity of the problem, and ensure that identical points on the input space are not accidentally visited, a contraction of the input space has been performed by grouping the input parameters into four non-dimensional numbers:

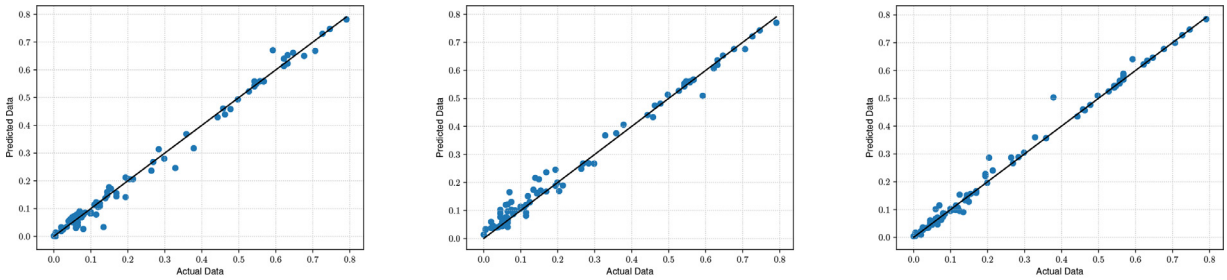
$$L_r = H_i/R_i, \quad Re = \frac{\rho U H_i}{\mu}, \\ Bm = \frac{\tau_0 H_i}{\mu U}, \quad Fr = \frac{U}{\sqrt{gH_i}}, \tag{19}$$

where U is obtained after specifying Froude number *Fr*. To these numbers, we add the power-law coefficient n to model parametrization. The input space has been explored in the following fashion: 150 simulations have been run by varying one parameter at a time (with others kept at operating conditions) and another 150 using randomly-generated inputs. This method of exploration was chosen as it allows us to explore the search space efficiently, given that some inputs are more influential than others, while the random sampling aids in exploring yet unseen regions of the input space. We also note that it allows us bypass the Cartesian spacing method which produces a combinatorial explosion on the number of simulations to run. The input parameters vary in intervals as presented in Table 5.

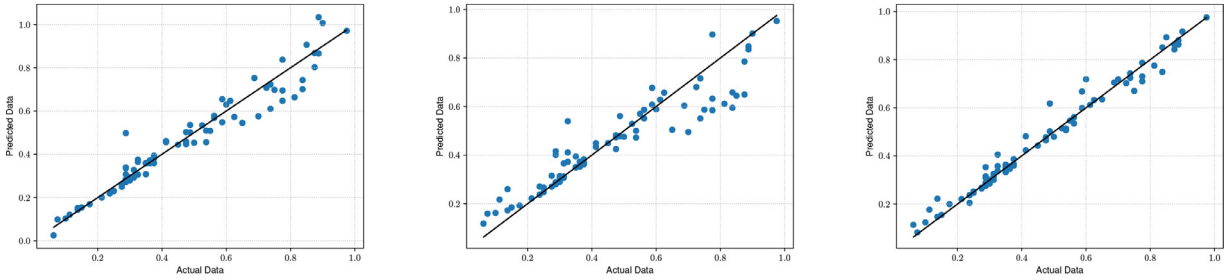
5.2.3. Predictive modelling

We aim to investigate the effect of the input parameters (*Re, Fr,* L_r, *Bm* and n) on the following FoMs: the final radius R and the final height H of the cylinder shape.

To that end, we follow the same procedure as in the previous case study for the hyperparameter selection of MARS, RF and



(a) Multivariate Regression Splines **(b)** Random Forest **(c)** Neural Network



(d) Multivariate Regression Splines **(e)** Random Forest **(f)** Neural Network

Fig. 8. Machine learning algorithms performance in the *Viscous Free Flow of a Cylinder* case study for the radius R (Fig. 8a–c) and the height H of the final shape (Fig. 8d–f).

NN. The number of training, validation, and testing samples are reported in Table 3. The resulting optimal number of maximum terms used in the forward phase of MARS is 20 for both the FoMs. In the case of the RF, the number of trees ensuring the smallest value of the MAE on the validation set is 1500 for the radius R and 200 for the height H of the cylinder. As far as NN is concerned, we employ the same reference architecture used in the *Water Clarifier* example with regularization parameter $\lambda = 10^{-7}$, 10^{-6} for R and H (respectively), obtained from cross-validation.

The mean absolute error (17) achieved on the testing set by the algorithms for the observables of interest are 0.015 and 0.034 for MARS, 0.018 and 0.046 for RF and 0.01 and 0.02 for NN (Fig. 8). All the algorithms show indeed a very good accuracy, with rather small prediction errors, smaller than the baseline (0.083 and 0.05), despite the high dimensionality of the input parameter space.

The learning models obtained provide an implicit function to the FoMs that can be used to interpolate the simulation results to any operating point falling in the range of the parametric study. The prediction results realized with NN are presented as 2D contour maps in Fig. 9. This particular case is a validation of our debris-flow and snow-avalanche simulation solutions that we develop for natural hazard prediction. The end user (generally the mountains and forestry safety authorities) could use the platform to create SDT's for particular areas where snow avalanches or debris falls are expected to occur. This is actually one of our present projects with the Swiss authorities.

In the light of the results achieved for these case studies, in the next example, we will only employ neural networks, as they have proven significantly higher generalization capability and accuracy compared to the other algorithms.

5.3. Train in a tunnel

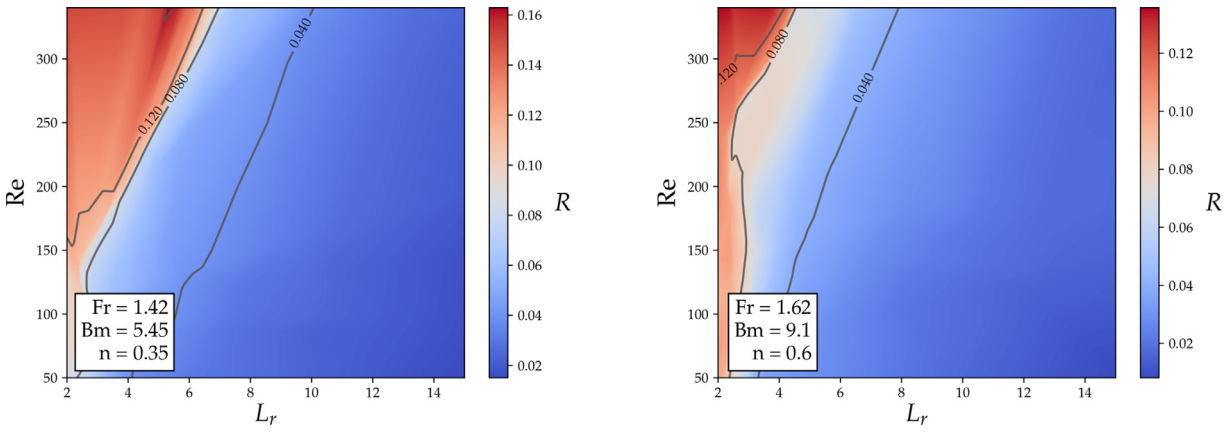
A case study has been performed concerning a fire emergency in a train driving gear inside a base tunnel. The fire is managed with the help of fans located along the tunnel and activated at the fire ignition. Events like this can be particularly dangerous for passengers if the ventilation is not properly handled. It has been shown that an insufficiently strong one can lead to the propagation of smoke in the evacuation direction [29]. The objective is to create a sufficiently complete database consisting of probable scenarios reflecting (i) various positions of the train in the tunnel, (ii) various fire intensities, and (iii) variable ventilation fan flowrates. An in-depth analysis of the problem can help design save evacuation plans, but it would require a large number of expensive CFD simulations due to a large number of parameters at play. Here, the role of machine learning becomes obvious: even with a reasonable number of CFD runs, the SDT of the tunnel should provide a wider range of operational conditions than what is feasible via simulation alone.

5.3.1. Flow physics and modelling

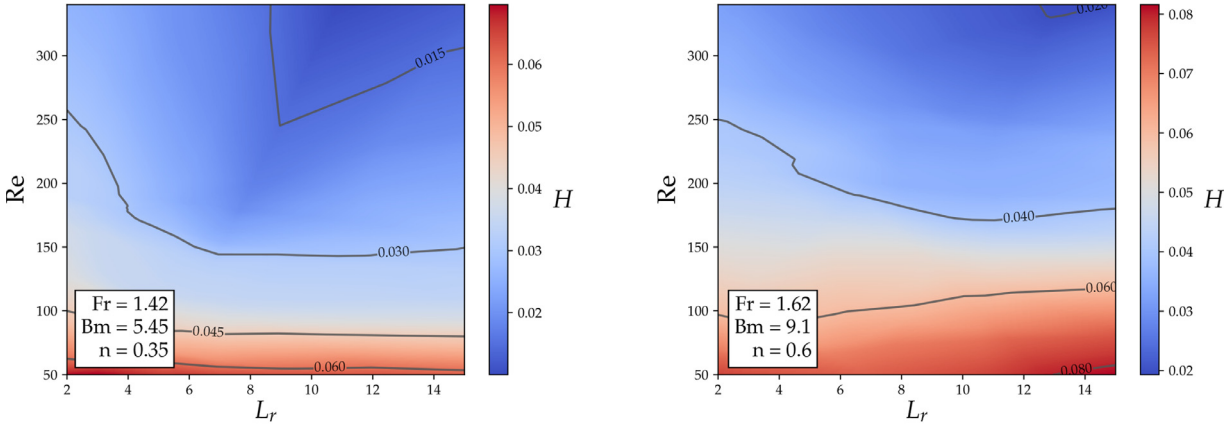
The simulations have been conducted in two dimensions only, considering the configuration described in Fig. 10a. The computational domain consists of a straight tunnel section of 200 m and a train of 50 m length positioned at different locations. The train and tunnel are represented by geometry data extracted from a CAD tool. Meshing is based on the Immersed Surfaces Technology, with a Cartesian grid comprising about 400 $\times$ 80 cells.

The flow of compressible air in the tunnel is treated as transient, where Eqs. (1)–(3) are solved and marched in time. A transport equation for smoke is also solved in the form of Eq. (3). Turbulence is modelled using the conventional $k-\varepsilon$ model with modifications for buoyancy effects. Wall functions are employed to treat the near-wall frictional effects. The fire model consists of a volumetric source of heat and smoke (S_Φ) at the location of one of the train driving gears.

The ventilation jet-fan is modelled as a volumetric momentum source added to the corresponding transport equation. Three equidistant jet fans (150 cm $\times$ 50 cm) are located along the tunnel ceiling. A delay between the start of the fire and the reaching of jet fans full operation capacity is needed to model the fire ignition and propagation scenario. Pressure outflow conditions are used at the open ends of the tunnel. Figs. 11a–c display contours of smoke concentration, fluid temperature and velocity at three positions of the train in the tunnel, including one case where the

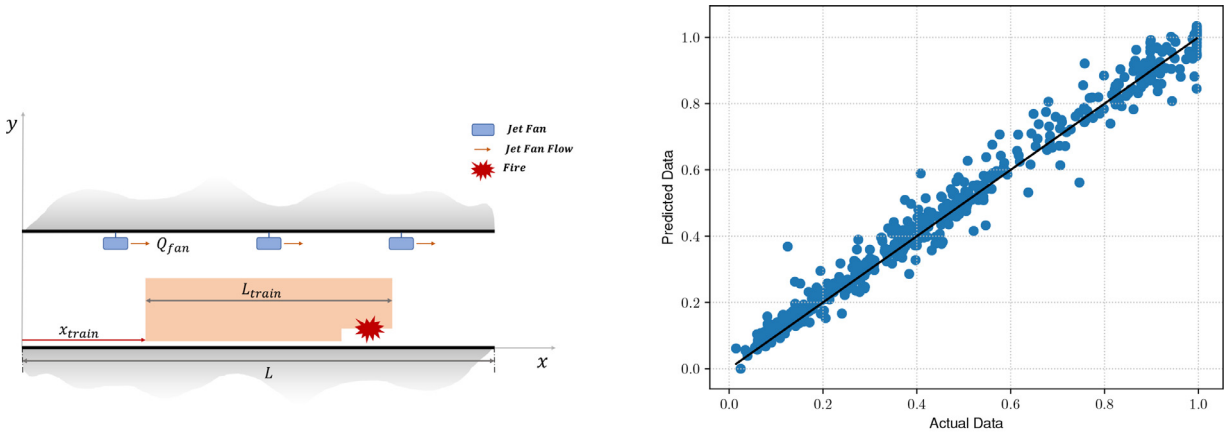


(a) Final radius of the shape R as a function of Re and L_r for different operating conditions



(b) Final height of the shape H as a function of Re and L_r for different operating conditions

Fig. 9. Predictions of the optimal network model for the two FoMs of the *Viscous Free Flow of a Cylinder*.



(a) **(b)**

Fig. 10. *Train in Tunnel* case study: schematic configuration Fig. 10a and neural network performance Fig. 10b.

ventilation fans are not active. The results suggest indeed different smoke spreading scenarios based on train locations.

5.3.2. Parametric study

A parametric study consisting of 144 simulations has been performed considering different fire and ventilation scenarios:

- 3 locations of the train, x_{train}: 25 m, 75 m, and 125 m;
- 8 values of fire intensity: $1 \leq I_{fire} \leq 8MW$;
- 6 values of jet fan flowrates: $0 \leq Q_{fan} \leq 40m^3/s$.

5.3.3. Predictive modelling

Finally, machine learning is used to complete the operational map representing any train position, any jet fan flowrate and any

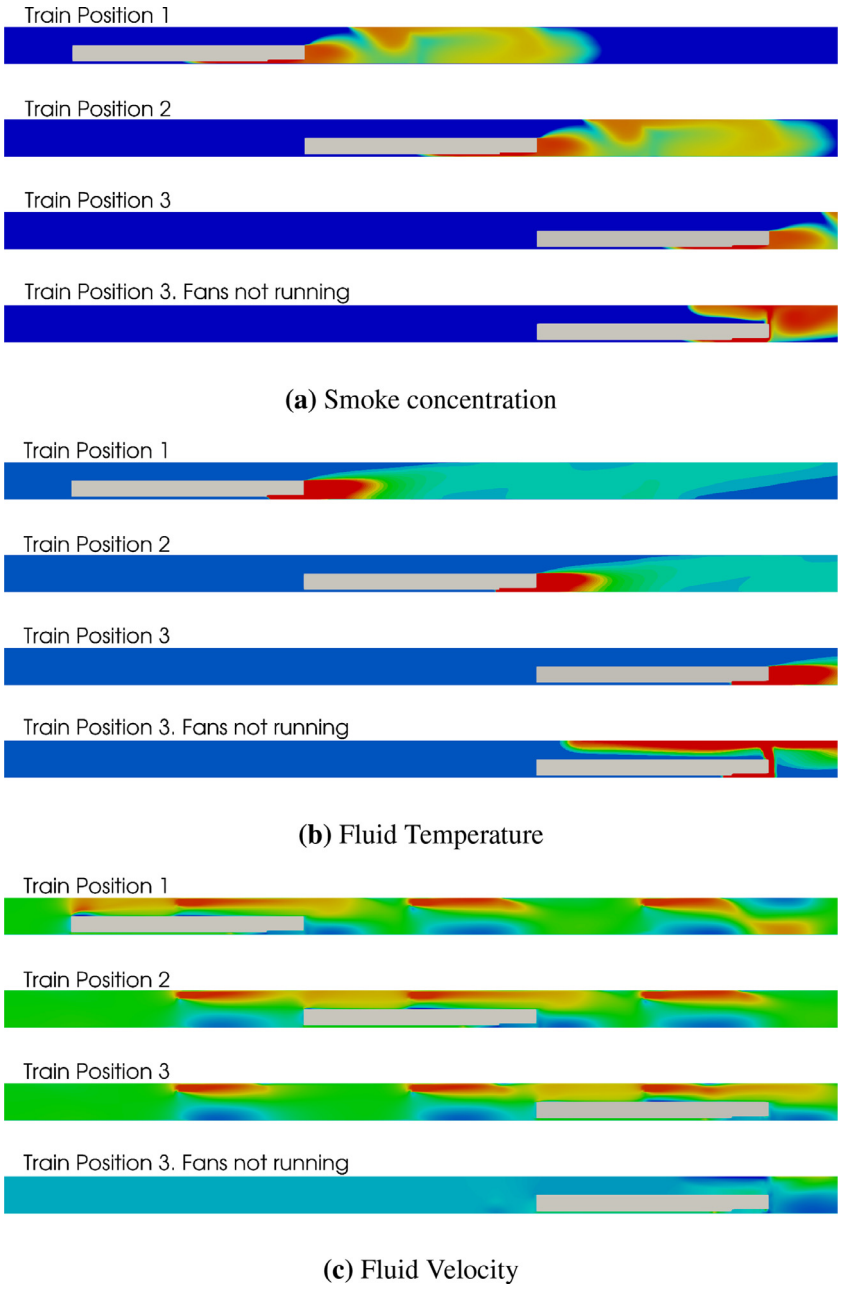


Fig. 11. CFD simulation results of the smoke concentration (Fig. 11a), and fluid temperature and velocity (Fig. 11b, c) for the *Train in Tunnel* case study.

fire intensity. In this context, the so-called visibility coefficient (S) is selected as the FoM:

$$S = 6T_g/83000F_{CO_2}, \tag{20}$$

where T_g is the temperature of the gas mixture in Kelvin and F_{CO_2} is the volume fraction of CO_2.

A neural network has been employed to reconstruct the map,

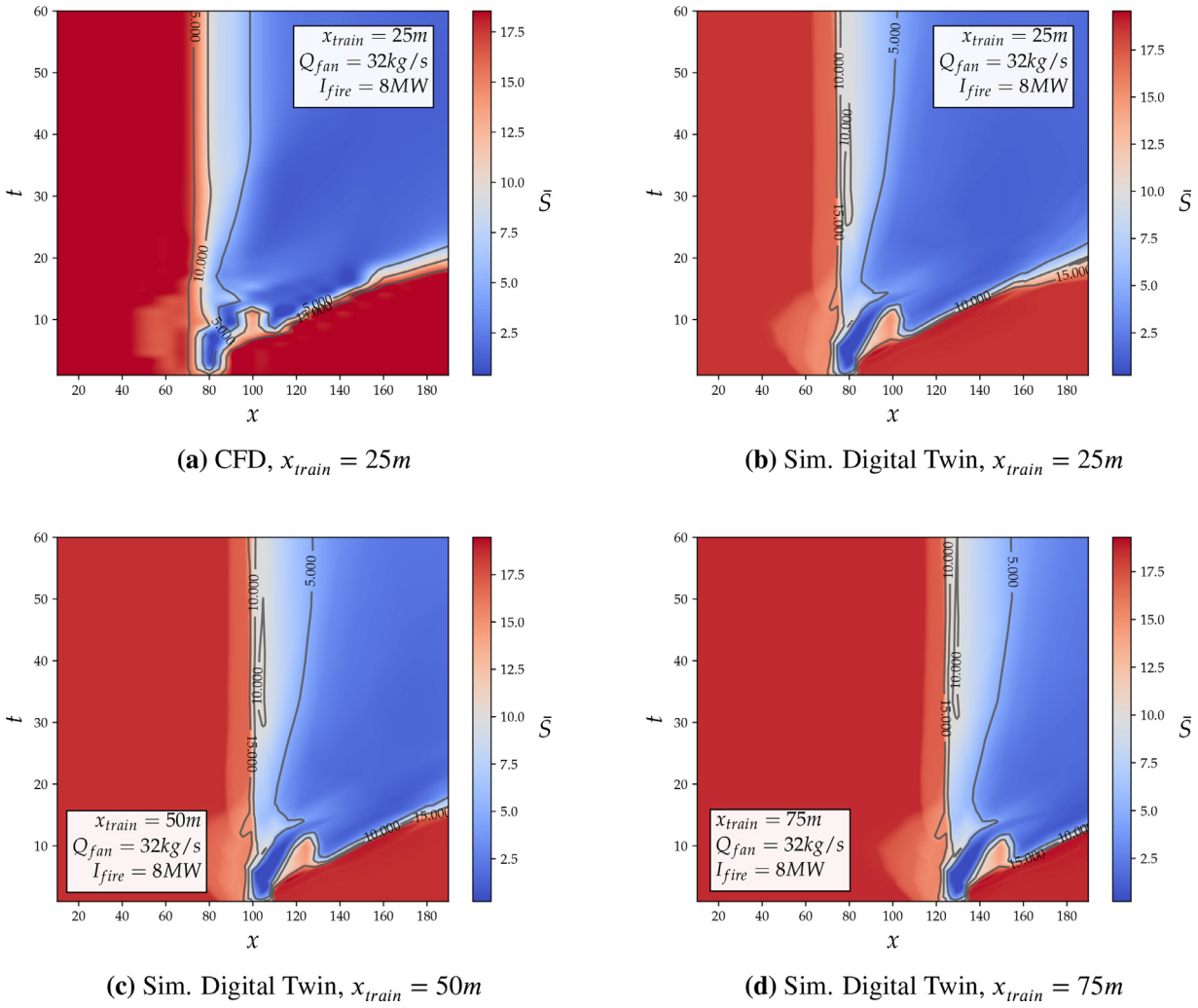
$$\bar{S} = \bar{S}(t, x, I_{fire}, Q_{fan}, x_{train}), \tag{21}$$

where t and x represent the time and the longitudinal coordinate of the tunnel, and $\bar{S}$ is the average visibility coefficient along the tunnel cross-section. The dataset (including also longitudinal and temporal profiles) has been split into training, validation and testing sets, corresponding to 79%, 10% and 1% of the entire dataset, respectively. Contrary to the previous cases, in the present one, we use cross-validation to define most of the network hyperparameters, including learning rate, regularization type, and architecture. We choose the best configuration among L^1 or L^2 regularization, $\lambda = 0, 0.001, 0.01, 0.1, 1$[2], $\eta = 0.001, 0.01$ and 4 hidden layers with 12, 24 or 48 neurons.

The final configuration consists of 4 hidden layers with 48 neurons and RELU activation function, L^2 regularization with $\lambda = 7.13 \cdot 10^{-9}$, ADAM optimizer with learning rate $\eta = 0.01$, and the mean squared error as objective function. The achieved prediction error is equal to 0.0131, fairly acceptable for engineering problems (Fig. 10b).

The trained model has been used to perform predictions on new data (Fig. 12). The diagrams show the visibility coefficient for the train locations $x_{train} = 25{,}50$ m and 75 m as a function of the spatial and temporal coordinates. The results prove that the model

[2] The value of the regularization parameter reported has been divided by the number of training and validation samples.



(a) CFD, $x_{train} = 25m$ **(b)** Sim. Digital Twin, $x_{train} = 25m$

(c) Sim. Digital Twin, $x_{train} = 50m$ **(d)** Sim. Digital Twin, $x_{train} = 75m$

Fig. 12. Comparison between the actual Fig. 12a and predicted Fig. 12b visibility coefficient $\bar{S}$ as function of time t and tunnel longitudinal coordinate x, at $x_{train} = 25$m, and Simulation Digital Twins at $x_{train} = 50$, 75 m Fig. 12c-d.

is able to successfully capture the most prominent features of the problem (Fig. 12a-b) and return reasonable predictions for configurations not covered by CFD data as well (Fig. 12c-d). From an engineering operational viewpoint, the advantage is clear: data covering a wide range of conditions are available in an easy and rapidly accessible database, and thus allowing to take timely safety decisions. Indeed, the tunnel safety crew could virtually interrogate the SDT specific to a tunnel to determine the safest escape paths in case of a hypothetical accident, without running new simulations. This is also one of our ongoing projects for the Swiss Gotthard base tunnel.

6. Conclusions

The work proposed here is meant to pave the way towards a new paradigm for CAE data treatment, analysis and solution-based deployment in computational engineering. We have shown that CFD data can be completed with the significant power of machine learning to cover a wider range of operational conditions that would otherwise be too expensive to simulate and build quickly accessible models for real-time response. The key success of our contribution resides in the efficient coupling of the CFD solver *TransAT* and the data platform *eDAP*, which together build a simple workflow for the design of *Simulation Digital Twins*. With this *data intelligence* strategy, the end-user can have access to a rich database comprising of a full picture of the operational processes, with a deeper insight into the physics and mechanisms that are impossible to achieve with a few CFD datasets only.

The paradigm introduced here opens various other horizons related to scientific data (both measurements and simulation) treatment and exploitation. For instance, the various physics-based models could be assessed by comparing their DDMs (referred to as CFD DDMs) over a large operational range, instead of comparing their results for specific conditions at specific points and segments only, as is usually the case. Further, experimental data generally used for model validation through comparison with the simulation can also be represented via their own DDMs, which we can refer to as Exp. DDMs [25]. CFD DDMs can be compared directly with the corresponding Exp. DDMs, and this should be the most comprehensive way of assessing the predictive performance of the physics-based models over a wider range of experimental conditions, providing a high-level classification of their range of applicability.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] eDAP. Website, (www.poyry.com/e-dap).
[2] Brunton SL, Noack BR, Koumoutsakos P. Machine learning for fluid mechanics. Annu Rev Fluid Mech 2020;52:477–508.
[3] Rowley CW, Dawson STM. Model reduction for flow analysis and control. Annu Rev Fluid Mech 2017;49:387–417.
[4] Lagaris IE, Likas A, Fotiadis DI. Artificial neural networks for solving ordinary and partial differential equations. IEEE Trans Neural Netw 1998;9(5):987–1000.
[5] Raissi M, Perdikaris P, Karniadakis GE. Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J Comput Phys 2019;378:686–707.
[6] Raissi M, Karniadakis GE. Hidden physics models: machine learning of nonlinear partial differential equations. J Comput Phys 2018;357:125–41.
[7] Mishra S., Molinaro R.. Estimates on the generalization error of physics informed neural networks (PINNs) for approximating PDEs. 2020a. ArXiv preprint arXiv:2006.16144.
[8] Tompson J, Schlachter K, Sprechmann P, Perlin K. Accelerating Eulerian fluid simulation with convolutional networks. In: Proceedings of the 34th international conference on machine learning, vol. 70. JMLR. org; 2017.
[9] Mishra S.. A machine learning framework for data driven acceleration of computations of differential equations. 2018. ArXiv preprint arXiv:1807.09519.
[10] Lye K.O., Mishra S., Ray D.. Deep learning observables in computational fluid dynamics. 2019a. ArXiv preprint arXiv:1903.03040.
[11] Lye K.O., Mishra S., Molinaro R.. A multi-level procedure for enhancing accuracy of machine learning algorithms. 2019b. ArXiv preprint arXiv:1909.09448.
[12] Mishra S., Molinaro R.. Estimates on the generalization error of physics informed neural networks (PINNs) for approximating PDEs II: a class of inverse problems. 2020b. ArXiv preprint arXiv:2007.01138.
[13] Ghorbel H, Zannini N, Cherif S, Sauser F, Grunenwald D, Droz W, et al. Smart adaptive run parameterization (SArp): enhancement of user manual selection of running parameters in fluid dynamic simulations using bio-inspired and machine-learning techniques. Soft comput 2019;23(22):12031–47.
[14] Ling J. Using machine learning to understand and mitigate model form uncertainty in turbulence models. In: 2015 IEEE 14th International conference on machine learning and applications (ICMLA). IEEE; 2015.
[15] Liu Y, Dinh N, Sato Y, Niceno B. Data-driven modeling for boiling heat transfer: using deep neural networks and high-fidelity simulation results. Appl Therm Eng 2018;144:305–20.
[16] Duriez T, Brunton SL, Noack BR. Machine learning control-taming nonlinear dynamics and turbulence, vol. 116. Heidelberg: Springer; 2017.
[17] TransAT. Website, www.transat-cfd.com.
[18] Lakehal D. On the modelling of multiphase turbulent flows for environmental and hydrodynamic applications. Int J Multiphase Flow 2002;28(5):823–63.
[19] Taylor SJE, Anagnostou A, Kiss T, Terstyanszky G, Kacsuk P, Fantini N, et al. Enabling cloud-based computational fluid dynamics with a platform-as-a-service solution. IEEE Trans Ind Inf 2019;15(1):85–94.
[20] James G, Witten D, Hastie T, Tibshirani R. An introduction to statistical learning, vol. 112. New York: springer; 2013.
[21] Friedman J, Hastie T, Tibshirani R. The elements of statistical learning, vol. 1. New York: Springer series in statistics; 2001.
[22] Friedman J. Multivariate adaptive regression splines. Ann Stat 1991;19(1):1–67.
[23] Goodfellow I, Bengio Y, Courville A. Deep learning. MIT press; 2016.
[24] Molinaro R, Lakehal D. On the paradigm of combining data analytics and CFD. In: Procs., 17th international conference of numerical analysis and applied mathematics (ICNAAM-2019), Symposium 'Numerical Fluids 2019', Sep. 23–28, Rhodes, Greece; 2019. To appear in AIP Proceedings.
[25] Buisson B, Lakehal D. Towards an integrated machine-learning framework for model evaluation and uncertainty quantification. Nucl Eng Des 2019;354:110197.
[26] Lakehal D, Krebs P, Rodi W. Computing shear flow and sludge blanket in secondary clarifiers. J Hydraul Eng 1999;125.3:253–62.
[27] Fornes P, Bihs H, Thakur V, Nordal S. Implementation of non-newtonian rheology for debris flow simulation with REEF3d. E-proceedings of the 37th IAHR World Congress August 13–18, Kuala Lumpur, Malaysia; 2017.
[28] Huang X, García MH. A Herschel-Bulkley model for mud flow down a slope. J Fluid Mech 1998;374:305–33.
[29] Li YZ, Lei B, Ingason H. Study of critical velocity and backlayering length in longitudinally ventilated tunnel fires. Fire Saf J 2010;45(6–8):361–70.