Active Learning Metamodelling for R-NEST

Raquel Sánchez-Cauce*, Christoffer Riis†, Francisco Antunes†, David Mocholí*,
Oliva G. Cantú Ros*, Francisco Câmara Pereira†, Ricardo Herranz*, Carlos Lima Azevedo†

*Nommon Solutions and Technologies
Madrid, Spain
raquel.sanchez@nommon.es

†DTU Management
Machine Learning for Smart Mobility
{chrrii,franant,camara,climaz}@dtu.dk

***Abstract*—The computational cost of realistic air traffic simulations is a barrier for a comprehensive assessment of new ATM concepts and solutions, which, in practice, restricts the simulations to a limited number of scenarios, often insufficient to obtain conclusive results. So, a goal for a comprehensive exploration of the simulation space should be finding its most informative instances. This can be done by means of active learning metamodelling, which can be used to translate a complex simulation model into a metamodel, allowing a more efficient exploration of the simulation input-output space.**

This work presents two metamodels developed within the SESAR ER4 SIMBAD project for one of the state-of-the-art ATM simulation tools, R-NEST. The metamodels were trained using the active learning technique through the metamodelling framework developed by the SESAR ER4 NOSTROMO project. The training process with this tool is also described in the paper.

***Keywords*—ATM; active learning; metamodelling; R-NEST; Gaussian Process**

I. Introduction

In many cases, microsimulation models are usually the only feasible and reliable way to assess the performance impact of new ATM concepts and solutions at network-wide level. However, when embedded with enough detail, their computational cost is a barrier for a comprehensive assessment of those solutions, and, in practice, simulations are necessarily restricted to a limited number of scenarios, often insufficient to obtain conclusive results. Hence, a goal when exploring the simulation space should be picking only the most informative instances. This can be done through active learning metamodelling.

An integrated approach of active learning and simulation metamodelling can be used to translate a complex simulation model into a metamodel, i.e., an analytical input/output function that iteratively approximates the results of a more complex function defined by the simulation model itself, improving computational tractability and interpretability of results by allowing a more efficient exploration of the simulator's behaviour.

This work presents the metamodels developed by the SESAR ER4 SIMBAD project for the R-NEST simulation tool using the active learning metamodelling technique. These metamodels were trained using the SESAR ER4 NOSTROMO metamodelling framework, developed through a metamodelling Application Programming Interface (API) [1]. This work is therefore the result of a collaboration between both SESAR projects.

The metamodels proposed were validated with the Demand and Capacity Balancing (DCB) SESAR's solution. The first metamodel is defined for a fixed day, and the second metamodel extends the previous one to a complete AIRAC by combining the metamodelling methodology with the selection of representative traffic samples methodology defined in [2]. The results obtained show the potential of this combined metamodelling approach.

II. Background

In this section, we briefly review the essential elements comprising the adopted methodology, namely, active learning, simulation metamodels, and Gaussian Processes.

A. Active Learning

Artificial intelligence (AI), in the form of statistical and machine learning models, is often data-hungry because it requires vast amounts of data to perform well. However, data can be expensive, mainly if a human must manually provide the label or if the data comes from a computationally costly simulator. First, we must be clear that in the literature, a data set consists of data points, and a data point can refer to both a *labelled* data point with an input and output $(\boldsymbol{x}, y)$, and to an *unlabelled* data point only consisting of the input $\boldsymbol{x}$. If the data point comes from a simulator, the unlabelled data point $\boldsymbol{x}$ is typically trivial to obtain, but the labelled data point $(\boldsymbol{x}, y)$ can only be obtained by actually running the simulator for the input $\boldsymbol{x}$ to get the output y.

Active learning reduces the amount of data needed by carefully focusing on only spending time on labelling a data point if it adds essential information to the model [3]. It should be seen in contrast to labelling a data set in a single go, e.g., making a grid with data points to label [4]. The acquisition function of the active learning can indeed be based on diversity in the input space, but more often, it is an advantage to look at the outputs of the model instead. In particular, it has been found that choosing to label the data point where the model is very uncertain about the predictions is an efficient strategy to increase the model's performance with the smallest amount of labelled data points [5].

As seen in [6], an arbitrary active learning strategy can be defined by the quintuple $(\mathcal{L}, \mathcal{U}, \mathcal{M}, \mathcal{O}, \mathcal{Q})$. First, $\mathcal{L}$ is the labelled data set used for training. The set of unlabelled data points is represented by $\mathcal{U}$. Generally, the number of unlabelled points is much higher than the labelled ones. Note that $\mathcal{U}$ encompasses the area of exploration within the input space. $\mathcal{M}$ is the machine learning model. Depending on the nature of the problem being modelled, it can be a classification or a regression model, which in turn affects the nature of labels in the set $\mathcal{L}$ of being discrete or continuous, respectively. The oracle is represented by $\mathcal{O}$. In this work, the oracle role will be played by R-NEST. Finally, $\mathcal{Q}$ is the query function that encodes the strategies and criteria for finding and selecting the most informative instances from $\mathcal{U}$ to be added to $\mathcal{L}$.

B. Simulation Metamodelling

A simulation metamodel [7] is any type of model that is used to approximate the unknown input-output mapping inherently defined by the simulation model, essentially serving as a surrogate or proxy with respect to the associated simulator. Although simulation models are simplified representations of the real-world system, they can still be, and often are, complex and detailed enough to yield significant inconveniences in practice. The most common shortcoming is their tendency to exhibit expensive simulation runs. Furthermore, the size and range of the input variable space can make it difficult to efficiently study and explore the behaviour of computer simulations as a whole, even with current computing technologies.

Simulation metamodels can then be employed to minimise the computational drawbacks posed by time-consuming simulation runs by jointly exploiting their approximate nature, functional simplicity, and fast computing. Being approximations of the underlying simulation functions, the metamodels' design and general performance can achieve balanced trade-offs between computational speed and controlled accuracy loss, depending on their ultimate objectives. Another feature of metamodels is that their respective functional structures are generally known and analytically defined, as opposed to those of most simulators. It is worthwhile noting that, although the average arbitrary simulator is oftentimes comprised of a plethora of internal analytic expressions and logical relationships, it can be externally treated as a single 'black-box' function with no clear mathematical formula. Nevertheless, an 'emergent behaviour', resulting from its inner interactions and dynamics that evolve over time can be directly observed. Metamodels aim at mimicking precisely this output behaviour as a function of the simulation inputs. Metamodels have previously been used in the field of transportation, but only recently within the area of ATM [8].

C. Gaussian Processes

The purpose of a metamodel is to have a fast approximation of the simulator, but it is equally essential that the underlying model of the metamodel can model complex functions alongside efficiently using small and medium-sized data sets. Additionally, active learning will be more efficient if the model provides some uncertainty measures because that helps the construction of a metamodel that minimises the uncertainty of its predictions and thus increases the accuracy of the approximation. For those reasons, the Gaussian Processes (GPs) are generally the underlying model used to create the metamodels because they are flexible, work well with relatively small data sets, and provide uncertainty estimates [5]. In short, a GP is a kernel method and can be used for regression problems, in a similar way as linear regression can be used. However, since the method is less known than linear regression, we will in the following formally define what a GP is while keeping the practitioner in mind. For an thorough description of a GP, we refer the readers to [9]. Due to their modelling flexibility and Bayesian formalism, GPs are common choices for both active learning and metamodelling tasks.

At first, to fit a GP - as our metamodel - to approximate the simulator, we need a data set $\mathcal{L}$, consisting of labelled simulations. We denote each simulation in the data set as a data point, which is specified by the d input variables $\boldsymbol{x}$ and the output of the simulator y (here it is a scalar, but in other cases it could be a vector). Given that there are N simulations in the data set, we formally write $\mathcal{L} = (X, \boldsymbol{y}) = \{\boldsymbol{x}_i, y_i\}_{i=1}^{N}$. In contrast to linear regression, a GP is a non-parametric function, and thus there are no parameters as such. Instead, a GP is given by some latent function values $\boldsymbol{f}$, such that they - added by a Gaussian noise ε - give the corrupted observation $y_i = f_i + \varepsilon_i,\ \varepsilon_i \in \mathcal{N}(0, \sigma_\varepsilon^2)$. A GP is fully specified by a mean function $m(\cdot)$ and a kernel $k(\cdot, \cdot)$. It is common practice to ignore the mean function and set it equal to zero, such that the GP is fully specified by the kernel $K_{\boldsymbol{\theta}}$, which is parameterised with the hyperparameters $\boldsymbol{\theta}$.

The kernel can take many forms, e.g., linear or polynomial, but one of the most common choices is the Radial-basis function (RBF) kernel, which is flexible and smooth. The RBF kernel is given as $k_{\text{RBF}}(\boldsymbol{x}, \boldsymbol{x}') = \sigma_k^2 \exp\left(-||\boldsymbol{x} - \boldsymbol{x}'||^2 / 2\ell^2\right)$, where σ_k^2 is the output variance and ℓ is the length scale. The intuition is that the output variance models the amplitude of the outputs, whereas the length scale is a measure of correlation between the outputs of nearby inputs. Often the kernel is defined without Gaussian noise, but we can add it to the kernel with an indicator variable, as $\sigma_\varepsilon^2 \mathbb{I}_{\{\boldsymbol{x}=\boldsymbol{x}'\}}$, such that the hyperparameters of the kernel are $\boldsymbol{\theta} = (\boldsymbol{\ell}, \sigma_k, \sigma_\varepsilon^2)$. We optimise the hyperparameters by maximizing the marginal likelihood estimate (MLE): $\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \left(\log \mathcal{N}(\boldsymbol{y} | \boldsymbol{0}, \boldsymbol{\theta} + \sigma_\varepsilon^2 \mathbb{I})\right)$. Then the prediction of the output for a new input $X^\star$ is given by the predictive posterior $p(\boldsymbol{f}^\star | \boldsymbol{\theta}, \boldsymbol{y}, X, X^\star) = \mathcal{N}(\boldsymbol{\mu}^\star, \boldsymbol{\Sigma}^\star)$, where the predicted mean and covariance are, respectively, $\boldsymbol{\mu}^\star = K_{\boldsymbol{\theta}}^\star \left(K_{\boldsymbol{\theta}} + \sigma_\varepsilon^2 \mathbb{I}\right)^{-1} y$ and $\boldsymbol{\Sigma}^\star = K_{\boldsymbol{\theta}}^{\star\star} - K_{\boldsymbol{\theta}}^\star \left(K_{\boldsymbol{\theta}} + \sigma_\varepsilon^2 \mathbb{I}\right)^{-1} K_{\boldsymbol{\theta}}^{\star\top}$, with $K_{\boldsymbol{\theta}}^{\star\star}$ being the covariance matrix between the test inputs, and $K_{\boldsymbol{\theta}}^\star$ being the covariance matrix between the training and test inputs.

III. Methodology

In this section, we give an overview of the metamodelling methodology developed within NOSTROMO [1] and depicted in Figure 1. Several requirements should be satisfied before

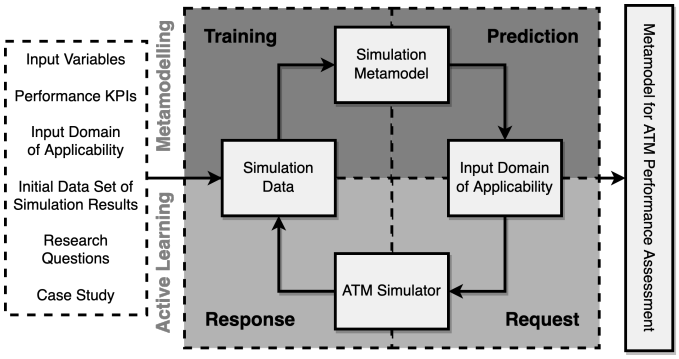

Figure 1. NOSTROMO Methodology

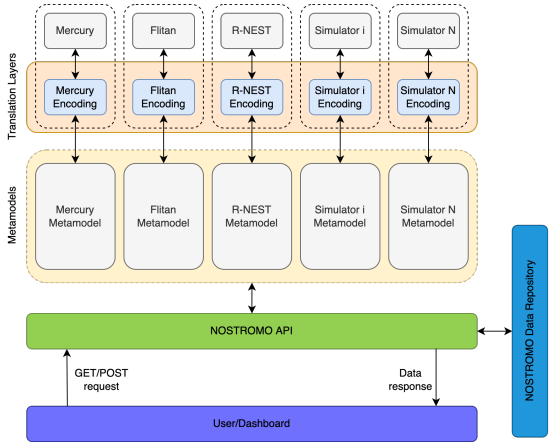

Figure 2. NOSTROMO API.

proceeding with this methodology. The first step is to define the research questions and the case study. From a modelling perspective, this will lead to the specification of the simulation input variables $\boldsymbol{x}$ and Key Performance Indicators (KPIs) or other outputs of interest y and, consequently, the input domain $\mathcal{U}$ of applicability in which we want to explore the simulator's behaviour. Additionally, due to the iterative nature of active learning, an initial training data set $\mathcal{L}$ should be built, essentially comprising a small set of simulation results generated according to a certain sampling strategy and based on the set of simulation variables we choose to work with.

The methodology is split into two stages, active learning and metamodelling, which then unfold into four sequential steps: (i) training, the metamodel is fitted to the simulation data $\mathcal{L}$; (ii) prediction, the fitted metamodel is used to predict over the simulation input domain of applicability $\mathcal{U}$; (iii) request, based on some query function $\mathcal{Q}$ (e.g. maximum predictive variance), new unlabelled data points $\boldsymbol{x}'$ are selected to be run by the simulator, being the oracle $\mathcal{O}$ in this case (remember that these points are comprised of points for which the corresponding true output values are unknown); (iv) response, the simulator provides new simulation outputs y' corresponding to the points from the third step, which are then added to the current training set $\mathcal{L}$. These steps are repeated cyclically until a stopping criterion is satisfied. This criterion can be defined, for example, as a function of the metamodel's performance, such as accuracy or error reduction, or simply by number of iterations to be performed with respect to the available time, budget and resources.

A. NOSTROMO API

Figure 2 depicts the general architecture of the metamodelling API developed in the context of NOSTROMO. Mercury [10] and Flitan (ISA Software), are two simulators used for the case studies evaluation within the project itself. This work focuses however on the results obtained from the integration of R-NEST. The NOSTROMO API assumes a REST architecture, enabling the creation of interactive applications. Within this architecture, the user can send data requests and receive back data from the server via an JSON object. The user requests are based on the HTTP protocol, which in turn provides a set of basic operations such as, checking the available simulator, specify the applicability domain (search grid), input variables and KPIs for metamodelling, and submit a metamodelling request.

Another important point to mention is the potential asynchronous nature of some API requests. Due to the runtimes of both the metamodel fitting and the simulator execution itself (due to the sequential active learning queries), the final metamodelling results are unlikely to be immediately available after the user request. On the contrary, a reasonable amount of computational time is indeed between the user's request and the effective delivery of the corresponding results. Therefore, the API will be logging the results directly to the NOSTROMO repository. The user is then able to check whether the submitted request has finished and finally obtain the results by looking into the repository.

In the current version of the API, the Translation Layer (TL) is implemented within a preliminary stage of the metamodelling process, essentially dealing with data preparation. Some simulators, whose input/output variables are of categorical (string-based) nature, cannot be directly used by most machine learning metamodels, only handling numerical or quantitate data. On top of that, the simulation data might be scattered across multiple log files and organised according to a structure not readily compatible with the usual matrix form. For these reasons, the TL generally encompasses data collection, conversion, and merging, and it is applied prior to the metamodelling itself. In the case of R-NEST this step was deemed unnecessary as the original data structure is already compatible with the developed infrastructure. For more details on the NOSTROMO API, as well as on the underlying methodology, refer to [1].

IV. Experiments

This section introduces the ATM simulation tool used, describes the two metamodels developed for it, and explains how they are implemented. For more details on the metamodels definition and implementation, we refer to [11]. Finally, the technical details of the active learning metamodels are given.

A. R-NEST tool

R-NEST (Research NEtwork Strategic Tool) is a simulation tool developed by EUROCONTROL for research activities aimed at evaluating advanced ATM concepts. It combines dynamic Air Traffic Flow and Capacity Management (AT-FCM) simulation with airspace design and capacity planning functionalities, while sharing the same basis as NEST, the Network Manager operational tool for capacity planning. R-NEST allows the dynamic simulation of network operations and the prediction of different types of delays, enabling the assessment at network level of the impact of the local implementations of new ATM concepts.

R-NEST allows the use of innovative algorithms for:

- airspace design processes, such as the estimation of fuel consumption or the creation of new elementary sectors;
- capacity management processes, such as the creation of new configurations that minimise the overload or the estimation of the workload based on potential conflicts and crossing durations;
- advanced ATFCM concepts, such as the implementation of Short Term ATFM measures (STAMs) or the calculation of ATFCM delays.

B. R-NEST metamodels definition

Two R-NEST metamodels are defined. The first one is a simpler model aimed at exploring the simulator input-output space; while the second one is a more complex extension of the former with additional input variables to cover a broader time range.

1) First metamodel: The first metamodel proposed estimates the average departure delay (punctuality PUN1 KPI of the SESAR Performance Framework [12]) of the flights crossing a selected region and the flights per air traffic controller (ATCo) hour in duty (cost-efficiency CEF2 KPI of the SESAR Performance Framework) in that region for a fixed day.

The metamodel is tested using the DCB SESAR's solution. This solution considers the implementation of the Dynamic Airspace Configuration (DAC) together with STAM measures. DAC aims to improve the airspace capacity by allowing Air Navigation Service Providers (ANSPs) to organise, plan, and manage airspace configurations with the granularity and flexibility required to respond to the changes in traffic demand. While STAM measures seek to smooth ATCO workload by reducing traffic complexity and peaks, through the short-term application of minor ground delays, re-routings and flight level capping to a limited number of flights.

The implementation of DCB in R-NEST is twofold. First, the improved configuration optimiser (ICO) algorithm is used to implement DAC in a region. Then, the STAM measures are implemented through the STAM simulation. This simulation provides the trajectories of the simulated flights, as well as a series of performance metrics of them.

ICO is an opening scheme (OS) optimisation algorithm developed and refined by the EUROCONTROL Experimental Centre that aims to minimise overload, the number of control positions used, and the number of configuration changes, in that order. This algorithm provides, among other performance metrics, the controller position hours and the overload sum.

The selected input variables of the metamodel are a subset of the inputs of this algorithm (while the rest are left with the default values). Specifically, the inputs of the metamodel are the two minimum opening duration parameters of the ICO algorithm:

- Configurations: integer for the minimum opening duration of the configuration in the configuration of the OS.
- Sectors: integer for the minimum opening duration of the sector in the configuration of the OS.

Once the ICO algorithm is applied, a STAM simulation is run using the updated OS. This simulation is run with the default values.

Hence, the first metamodel proposed takes as inputs the minimum opening duration of both the configuration and sector in the configuration of the OS and as outputs the PUN1 and CEF2 KPIs, i.e., the datapoints for this metamodel are of the form

$$(\text{configurations, sectors; PUN1, CEF2}),$$

where the semi-colon separates the input variables from the output variables.

2) Extended metamodel: The second R-NEST metamodel expands the previous one to extend the KPIs prediction for an entire AIRAC instead of just for a fixed day. For that, the hourly entry counts of the day in the selected geographical region are also considered as input variables of the metamodel to characterise each day. Hence, the input variables of the previous metamodel are extended with this 24-dimension vector, while the output variables remain the same, i.e., the datapoints for this metamodel are of the form

$$(\text{configurations, sectors, entry counts}_0, \ldots, \text{entry counts}_{23}; \text{PUN1, CEF2}),$$

where the semi-colon separates the input variables from the output variables and entry counts_i denotes the entry counts at hour i.

C. R-NEST metamodels implementation

1) First metamodel: The spatial scope of the analysis is focused on the Bordeaux ACC, specifically the lower and east cluster of the Bordeaux ACC (LFBBCTAE in R-NEST) is selected, and only the set of flights crossing this region are considered.

The metamodel is implemented for the day 5th July 2019. Regarding the input space, both input variables (i.e., configurations and sectors) take values from 10 to 300 minutes in steps of 10 minutes. All the possible combinations of these variables are considered, with the constraint that the sectors variable should be greater than or equal to the configurations variable (as stated in the R-NEST user manual). Additionally, the combination $(0, 0)$ was also included as input point in the metamodel to represent the situation where DAC is not implemented (i.e., the ICO algorithm is not applied and the

original OS is used). In total, this metamodel has 466 input points (i.e., combinations of the input variables).

2) Extended metamodel: As this metamodel is an extension of the previous one, we keep the same spatial scope (the lower and east cluster of the Bordeaux ACC). Here, the 7º AIRAC of 2019 (that includes the days from 20th June to 17th July) is selected as temporal scope, in order to keep the same time period as the previous metamodel. The aim of this metamodel is therefore to estimate both KPIs for every combination of the OS parameters and every day of that particular AIRAC.

The key part in this metamodel implementation consists in selecting a big enough set of representative days to train it with, able to capture all the different behaviours and provide information about the patterns present in the region. This is needed because using the 28 days of the AIRAC, with all their possible OS configurations, would be really inefficient (even in a AL context). For that, the methodology proposed in [2] for the selection of representative traffic samples in a year is applied to the Bordeaux ACC.

After this process, seven clusters were obtained (see Fig. 3, where each day is coloured by cluster belonging). To select the traffic samples, the days with the highest silhouette score of the AIRAC in each cluster were identified. Table I shows the days selected.

TABLE I. REPRESENTATIVE DAYS FOR EACH CLUSTER

Cluster	Cluster id	Day
Dark blue	0	10/07/2019
Orange	1	30/06/2019
Red	2	29/06/2019
Brown	3	15/07/2019
Pink	4	20/06/2019
Light blue	6	02/07/2019

Note that the yellow-olive cluster only contains one day (10th May 2019), which does not belong to the selected AIRAC, hence, this cluster is not included in the metamodel.

For each day, the same combinations of the ICO parameters as before are considered, i.e., 466 possible combinations. Hence, the training space has a total of 2796 points (6×466), while the complete input space has $13,048$ points (28×466).

D. Technical details of the active learning metamodels

The active learning metamodels are created by the NOSTROMO API using a GP for each output with an RBF kernel and zero mean function. All inputs X were normalised to the unit cube, and the outputs $\boldsymbol{y}$ were standardised to have zero mean and unit variance. The hyperparameters were optimised with gradient descent based on the MLE. New data points were queried based on the query function $\mathcal{Q}_{var}$ given by the maximisation of the mean variance across the outputs [8]. In practice, only a training data set $\mathcal{L}$ is needed to create a metamodel, but for evaluation it is recommended to have a small test set $\mathcal{L}_{test}$ as well. Since we also compare the performance of the query function $\mathcal{Q}_{var}$ to random sampling $\mathcal{Q}_{rand}$, we create a small validation set $\mathcal{L}_{val}$ (by removing some points from $\mathcal{L}$), potentially allowing us to choose between the two query functions $\mathcal{Q}_{var}$ and $\mathcal{Q}_{rand}$. All results are averaged across 100 experiments using the same data.

The performance of the metamodel is assessed using the root mean squared error (RMSE) and mean absolute percentage error (MAPE) metrics. For the training process of the first metamodel, 73 points were used. The training set contains 45 points, the validation set ($\mathcal{L}_{val}$) contains 13 points, and the test set ($\mathcal{L}_{test}$), 15 points. While a total of 80 points were used for the training process of the extended metamodel, 65 for training and 15 for validation. Regarding the testing data, two different test sets were created, one with 13 points containing only the representative days identified (referred to as test set with representative days) and another one with 28 points containing different days of the AIRAC (referred to as test set with external days). This way, we can assess how the metamodel has learnt for the representative days and its ability to generalise to the whole AIRAC. This provides an assessment of the complete methodology used.

In both cases, the AL process ends when all the points of the training set have been explored.

V. RESULTS

A. First metamodel

Fig. 4 shows the comparison of the RMSE of the predictions on the validation set of the PUN1 (top) and CEF2 (bottom) variables per iteration of the AL cycle. Each blue line represents a repetition of the training process, and the mean and median are shown in red and yellow, respectively. The column in the right shows the comparison between the mean RMSE of both query strategies. As can be seen, the mean RMSE of the variance sampling is smaller than the one of the random sampling in every iteration of the AL cycle, showing that the points selected with this criterion provides more information to the learning algorithm.

Regarding the two columns on the left, it can be seen that the variance sampling produces more stable results, as the error curves of each repetition are more similar among them and the band that enclose them is narrower. Hence, the model trained with this query function was finally selected.

Next, the predictive performance of the trained model is assessed on the test set. Table II shows the mean RMSE and MAPE of the predictions for each output of the metamodel. As can be observed, the predictive errors are pretty low, especially for the CEF2 variable.

TABLE II. PREDICTIVE ERROR OF THE R-NEST METAMODEL

	PUN1	CEF2
RMSE	2.45	0.14
MAPE	0.125	0.008

Finally, Fig. 5 compares the actual values of the test set (in red) with the predictions of the metamodel (in black). The prediction is shown together with its predictive standard deviation (grey vertical bar). The first column shows the PUN1 values

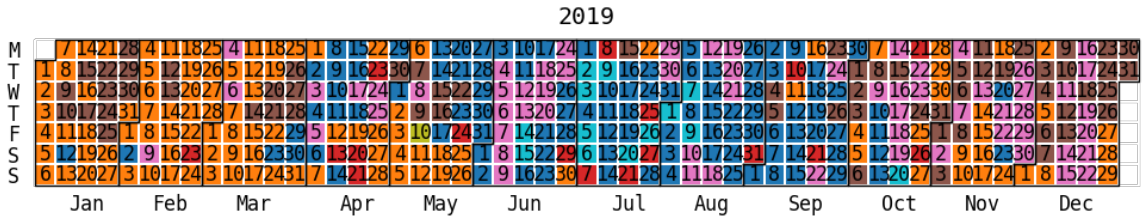

Figure 3. Traffic patterns for the Bordeaux ACC

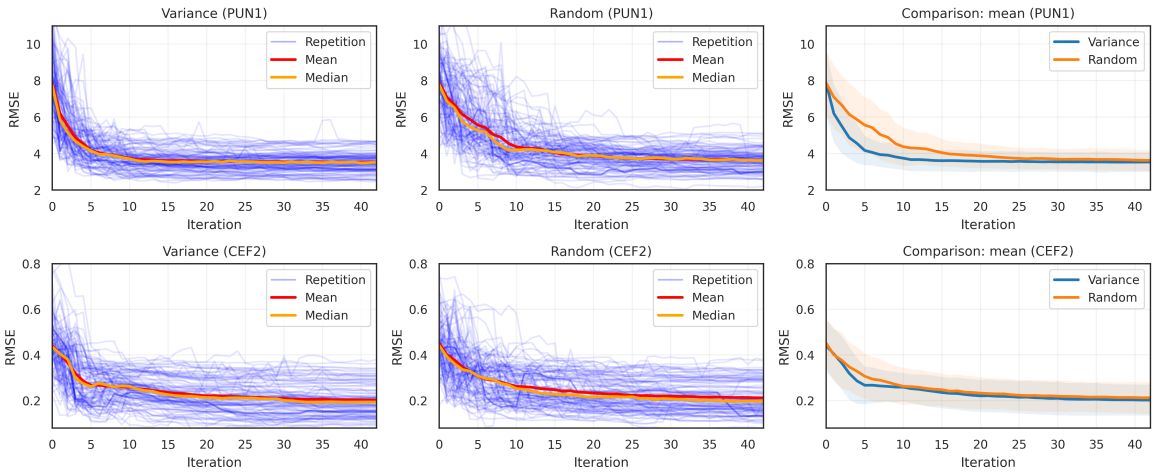

Figure 4. RMSE of the predictions of the PUN1 (top) and CEF2 (bottom) variables per iteration for the two query functions considered ($\mathcal{Q}_{var}$ and $\mathcal{Q}_{rand}$). Each repetition is a metamodel with a random subset of the data used for validation. The two plots to the right are the mean performance $\pm$ 1 standard deviation.

with respect to each input variable, while the second column depicts the CEF2 values with respect to each input variable. As shown in the figure, the predicted values for the CEF2 variable are very close to the true ones. Moreover, in most of the cases the actual values are inside the standard deviation interval of the predictions, which in turn are small. This implies that the metamodel is confident about its predictions. Regarding the PUN1 variable, the predictions are not as accurate, however, in many cases the actual value also belongs to the predictive standard deviation interval, which implies that the metamodel has learnt to model the behaviour of the variable.

B. *Extended metamodel*

This metamodel was trained following the same process as the first metamodel. After comparing its performance for both query functions, the query function $\mathcal{Q}_{var}$ was selected as it produced more regular and stable results (as already observed in Section V-A). Once it is trained, its predictive performance is assessed on the two test sets created. Table III shows the predictive RMSE and MAPE for each output variable for the test set with representative days and the test set with external days. As can be seen, the errors for the PUN1 variable are high in both cases, although they are higher in the set with representative days. Regarding the CEF2 variable, the predictions are very accurate in the first case and less accurate (although still good) in the second case. As one may expect,

	PUN1	CEF2
RMSE	5.66	0.24
MAPE	0.306	0.015

(A) REPRESENTATIVE DAYS

	PUN1	CEF2
RMSE	3.23	0.61
MAPE	0.247	0.035

(B) EXTERNAL DAYS

TABLE III. PREDICTIVE ERROR FOR THE TEST SET WITH THE REPRESENTATIVE AND EXTERNAL DAYS OF THE EXTENDED R-NEST METAMODEL

the overall predictive results are worse on the test set with external days, as these are new for the model.

Fig. 6 shows the comparison of the actual values and the predictions (in black) for the test set with representative days. The standard deviation of the prediction is also depicted (grey vertical bar). Each actual point is coloured by cluster belonging (where the cluster id is the same used in Table I). As before, each output variable is plotted against each input variable. The predictive results for the CEF2 variable are pretty accurate, as already observed in Table III (A), with most of the actual points inside the standard deviation interval of their predictions, which in turn are small. This means that the metamodel provides confident and accurate predictions for that variable. While the predictions for the PUN1 variable are worse, although for many points their actual value belongs to the predictive standard deviation interval as well.

Fig. 7 shows the same information for the test set with the external days. Note that for this dataset, only four clusters

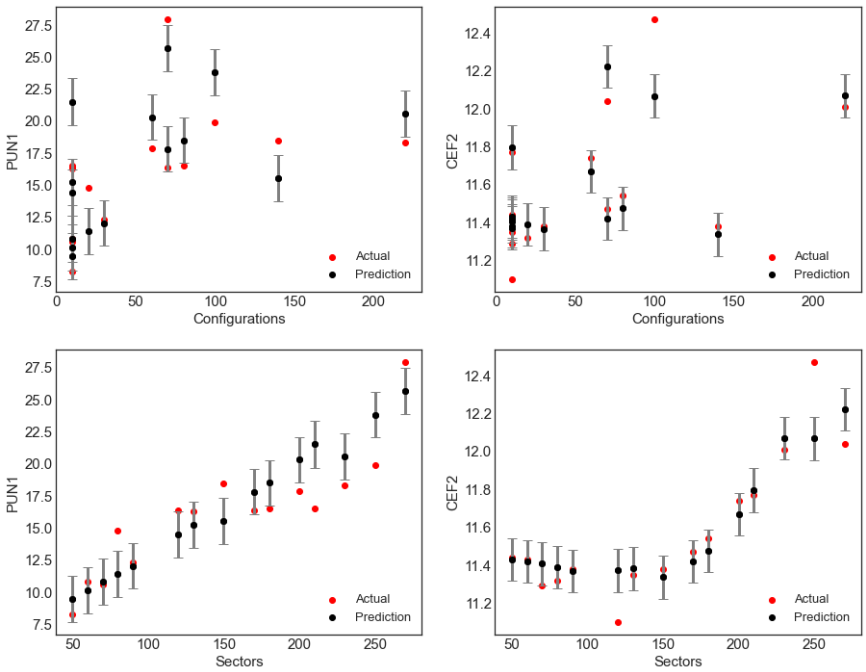

Figure 5. Predictive performance of the R-NEST metamodel on the test set.

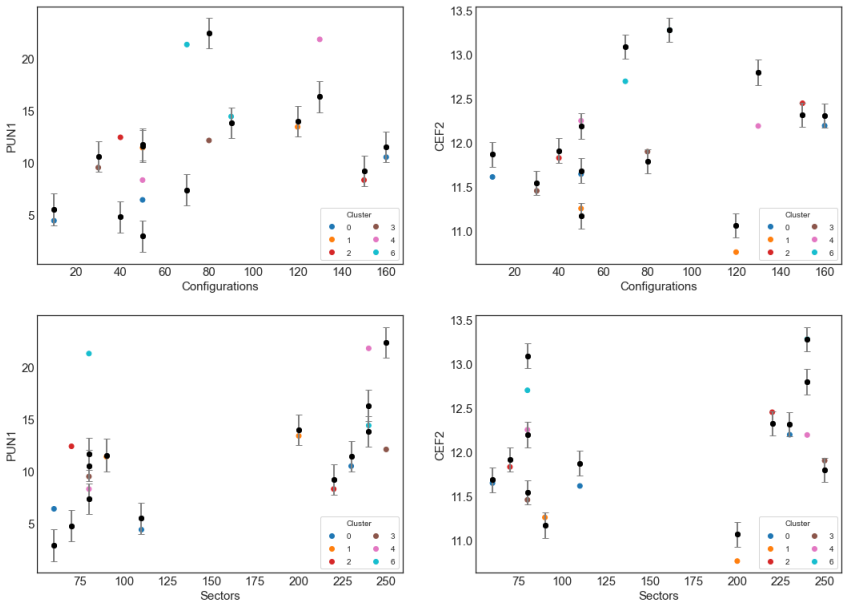

Figure 6. Predictive performance of the extended R-NEST metamodel on the test set with the representative days.

are considered, as clusters 1 and 3 only contain one day in the 7º AIRAC, which were used for training. In this case, the predictive standard deviation intervals are much larger, implying a greater uncertainty in the predictions of the metamodel. Moreover, the actual value of very few points lies in this interval. This suggests that the metamodel is not able to accurately generalise to other days of the AIRAC. Nevertheless, it manages to model the behaviour and inertia of the variables.

VI. Discussion and Conclusions

This paper shows the metamodels defined in the SESAR ER4 SIMBAD project for the R-NEST tool and the NOSTROMO active learning metamodelling framework used to train them.

Two metamodels were defined and the results obtained show the potential of the methodology presented. The first metamodel was defined for one single day, 5th July 2019, and was trained with 45 points. Its predictive performance is very accurate, with a mean predictive error under 1% for

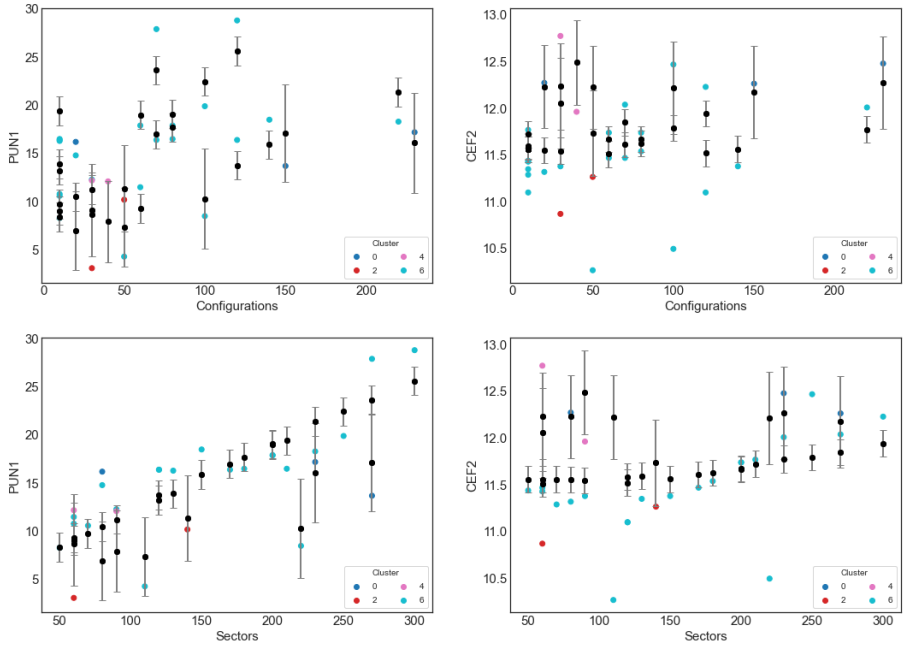

Figure 7. Predictive performance of the extended R-NEST metamodel on the test set with the external days.

the CEF2 variable and under 13% for the PUN1 variable. In order to extended this metamodel to the rest of the days of the 7º AIRAC, an extended metamodel was defined including as input variable the 24-dim vector of the hourly entry counts of the day in the Bordeaux ACC. To find the days to train this model with, the methodology for the identification of representative traffic patterns and traffic samples described in [2] was followed. This procedure led to six representative days, which were used to train the metamodel. The predictive performance of this metamodel is evaluated on two different test sets, one containing the same days used to train it and another one containing different days of the AIRAC. The former test set allows the assessment of the ability to generalise of the model, while the later, the assessment of the combined approach followed.

The predictive results for the test set with representative days are very good for the CEF2 variable (predictive error under 2%), for which most of the true values are inside the standard deviation interval of the prediction. The results for the PUN1 variable are worse (predictive error of around 30%).

Regarding the test set with external days, the results are worse in general. Even though the predictive error for the PUN1 variable is smaller than for the test set with representative days, it is still high (around 25%). Nevertheless, results for the CEF2 variable, despite being worse (3.5% of predictive error), are still good. The fact that the predictive results for the PUN1 variable are worse for both metamodels may be due to the high variability of this variable, which may require more training points to fully capture its behaviour.

In the extended metamodel case, for both variables the actual values of most of the points lie outside the standard deviation interval of the prediction, which in turn are large in many cases. This implies that the metamodel is not able to properly generalise to other days with the provided training set. At this point, it is important to highlight that the training and validation sets contain 80 points, while the complete input space of the metamodel consists in $13,048$ points. So, these datasets may not contain enough information for the metamodel to extract the global structure of the complete input-output space. Nevertheless, the results obtained show the potential of this approach as, from only 80 points corresponding to the six representative days found, it is able to get the inertia for the rest of the days of the AIRAC.

As future work, the performance of the extended R-NEST metamodel should be improved, to refine and further demonstrate the presented metamodelling approach. For that, we propose two possible solutions. In first place, the training set should be enlarged with more points. In addition to that, more representative days should be included in the training set, considering for that the days of the AIRAC with the lowest silhouette score per cluster (as suggested in [2]). This way, the model would be trained with a more complete sample of the behaviour and patterns of the ACC, which should lead to a better performance.

ACKNOWLEDGMENT

This work was supported by the SIMBAD and NOSTROMO projects, both funded by SESAR Joint Undertaking through the European Union's Horizon 2020 research and innovation programme under grant agreements Nos 894241 and 892517, respectively. The Eurocontrol R-NEST team is also greatly acknowledged for their support and guidance.

AUTHORS CONTRIBUTION

The authors confirm contribution to the paper as follows: study conception and design: R. Sánchez-Cauce, C. Riis, F. Antunes, D. Mocholí, R. Herranz and O.G. Cantú Ros; data collection: R. Sánchez-Cauce and C. Riis; analysis and interpretation of results: all; draft manuscript preparation: R. Sánchez-Cauce, C. Riis and F. Antunes. All authors reviewed the results and approved the final version of the manuscript.

REFERENCES

[1] NOSTROMO Consortium, *D3.4: Final Metamodelling Methodology*. NOSTROMO Project, Deliverable D3.4, WP3, 2020.

[2] SIMBAD Consortium, *D4.1: Methodologies and Algorithms for the Selection of Representative Traffic Samples*. SIMBAD Project, Deliverable D4.1 Version 01.00.00, October 2022.

[3] B. Settles, "Active learning literature survey," University of Wisconsin–Madison, Computer Sciences Technical Report 1648, 2009.

[4] T. J. Santner, B. J. Williams, and W. I. Notz, *The Design and Analysis of Computer Experiments*. Springer, 2019.

[5] R. B. Gramacy, *Surrogates*. Chapman and Hall/CRC, mar 2020.

[6] X. Wang and J. Zhai, *Learning With Uncertainty*. CRC Press, 2016.

[7] L. W. Friedman, *The simulation metamodel*. Springer Science & Business Media, 2012.

[8] C. Riis, F. Antunes, G. Gurtner, F. C. Pereira, L. Delgado, and C. M. L. Azevedo, "Active learning metamodels for atm simulation modeling," *Proceedings of the 11th SESAR Innovation Days*, 2021.

[9] C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*. MIT press Cambridge, MA, 2006, vol. 2, no. 3.

[10] L. Delgado, G. Gurtner, P. Mazzarisi, S. Zaoli, D. Valput, A. Cook, and F. Lillo, "Network-wide assessment of atm mechanisms using an agent-based model," *Journal of Air Transport Management*, vol. 95, p. 102108, 2021.

[11] SIMBAD Consortium, *D5.1: Active Learning Metamodelling*. SIMBAD Project, Deliverable D5.1 Version 01.00.00, September 2022.

[12] PJ19.04 Consortium, *PJ19.04: Performance Framework (2019)*. PJ19.04 Project, Deliverable D4.7 Version 01.00.01, November 2019.