EXPLORATORY RESEARCH

D5.1 ATM Performance Metamodels - Preliminary Release

Deliverable ID:	D5.1
Dissemination Level:	PU
Project Acronym:	NOSTROMO
Grant:	892517
Call:	H2020-SESAR-2019-2
Topic:	SESAR-ER4-26-2019
Consortium Coordinator:	CRIDA
Edition date:	04 April 2022
Edition:	00.00.03
Template Edition:	02.00.05

EUROPEAN PARTNERSHIP

Co-funded by the European Union

Authoring & Approval

Authors of the document

Name / Beneficiary	Position / Title	Date
DTU	Task Member	10/03/2022
ISA	Task Member	10/03/2022
UoW	WP Leader	10/03/2022

Reviewers internal to the project

Name / Beneficiary	Position / Title	Date
UoW	WP Leader	14/03/2022
UPC	Task Member	14/03/2022
CRIDA	Task Member	14/03/2022

Reviewers external to the project

Name / Beneficiary	Position / Title	Date
N/A		

Approved for submission to the SJU By - Representatives of all beneficiaries involved in the project

Name / Beneficiary	Position / Title	Date
CRIDA	Project Coordinator	04/04/2022

Rejected By - Representatives of beneficiaries involved in the project

Name and/or Beneficiary	Position / Title	Date
N/A		

Document History

Edition	Date	Status	Name / Beneficiary	Justification
00.00.01	11/03/2022	Draft	UoW	First draft for partners review
00.00.02	14/03/2022	Draft	UoW	Version ready for submission
00.00.03	04/04/2022	Final Draft	CRIDA	Update with new template

NOSTROMO

NEXT-GENERATION OPEN-SOURCE TOOLS FOR ATM PERFORMANCE MODELLING AND OPTIMISATION

This deliverable is part of a project that has received funding from the SESAR Joint Undertaking under grant agreement No 892517 under European Union's Horizon 2020 research and innovation programme.

Abstract

This deliverable presents the results obtained with the meta-modelling process presented in D3.1 and D3.2 applied to the two micromodels (or simulators), Mercury and FLITAN, themselves implementing concepts from four SESAR solutions, PJ01.01, PJ07.02, PJ08-01, and PJ02.08.

The objective of the meta-modelling process is explained briefly again in the introduction, in particular with respect to performance assessment. The rationale for the selection of the SESAR solutions implemented in the simulators are briefly explained too.

The simulators are presented in two distinct chapters. First, a general presentation of each simulator is given, with past challenges and development, before explaining the development steps carried out to implement the concepts from the chosen solutions. Domain research questions that could be answered by these implementations are highlighted along the way.

The meta-modelling process is then briefly explained again, followed by the results obtained with the two simulators, in distinct sections. The results highlight the performance of the meta-model with respect to approximating the output of the micromodels, but not the performance of the models themselves with respect to the research questions, which will be explored in WP7 instead.

The deliverable closes with some considerations on the meta-modelling performance and next steps for this line of work.

Table of Contents

List of Tables

List of Figures

1 Introduction

From a key performance assessment point of view, modelling can be a very strong tool to forecast the efficiency or the potential downfalls of new processes in the air transportation system before they are deployed. Models are sometimes very specialised by nature, able to precisely estimate important KPIs tackled by these new processes. However, KPIs are typically highly interdependent, due to the complexity of the air transportation system and its actors. Including more systems and actors in existing models becomes quickly prohibitive from the computational point of view and thus prevents from having a systemic assessment of these new processes.

Active learning is a possible solution to this issue. By allowing a machine learning model (the metamodel) to learn functional relationships between input and output of a low-level model (the micromodel)[1], one can hope to generate predictions with a much higher speed while retaining a satisfactory forecast accuracy. In NOSTROMO we explore this possibility of metamodelling by considering two micro-simulators that can answer hypothetical research questions linked to existing SESAR solutions. This deliverable presents the results of this metamodelling process applied to the modelling of different solutions by the simulators Flitan and Mercury.

Indeed, we assume in this deliverable that some research questions linked to solutions PJ01.01, PJ07.02, PJ08-01, and PJ02.08 have been raised, as explained in D4.1 [1] (and further explained in this deliverable). Hypothetically, these research questions could be answered by the micromodels, Flitan and Mercury. We are interested in this deliverable in how well these questions would be answered by the metamodelling process in terms of quality and speed of computation. Hence, the main objective of this deliverable is to assess the metamodelling process from a technical point of view. The analysis of results of the micromodels themselves, from ATM domain-knowledge point of view, will be examined in WP7 deliverables.

The metamodelling process, as explained in D3.2 [2], relies on querying a micromodel with specific input (in a "smart" way) and getting the output of this micromodel in order to approximate it. In NOSTROMO, we used Flitan and Mercury as test cases for the metamodelling process. Both simulators have different strengths when it comes to modelling. As a result, NOSTROMO has selected a small number of SESAR solutions (PJ01.01 and PJ07.02 for Mercury, PJ08-01, and PJ02.08 for Flitan) for which research questions could in principle be answered by the two simulators (see D4.1 for other criteria of selection on the solutions).

Flitan is particularly well suited to research questions related to airspace or runway configuration. As a result, concepts from solutions PJ08-01 "Management of Dynamic Airspace Configurations" and PJ02.08 "Traffic Optimisation of Single and Multiple Runway Airports" can be modelled by the simulator. Section 2.1 explains in detail the past and current development of Flitan in relation to these two solutions. It includes a detailed explanation of the concepts taken from these.

[1] Note that in this deliverable we will use "micromodel" and "simulator" quite interchangeably.

Mercury has been developed with a strong emphasis on airline cost models and passenger tracking. It is thus more suited to modelling concepts from solutions PJ07.02 – AU Fleet Prioritisation and Preferences (UDPP) and PJ01.01 – Extended arrival management with overlapping AMAN operations and interaction with DCB and CTA. Section 2.2 presents the concepts inspired by these solutions that were implemented and tested in Mercury. It shows the potential value of these concepts and explains the kind of research questions that can be answered with the new Mercury development.

The metamodelling process, at the heart of NOSTROMO and this deliverable, has been explained previously in deliverables from WP3. In section 3, we explain again briefly the main concepts behind it.

Section 4 presents the results obtained with this process. The results are organised per simulator and then per solution. A section for Flitan is present at the beginning to explain the extra steps that were needed in order to train the metamodel – mainly encoding the input in a way usable by the training procedure. The main focus of the results is on the quality of the metamodelling compared with the bare output of the simulators. Note that for reasons explained in sections 3.1 and 4.1, the results of solution PJ02.08 could not be produced. These results will instead be included in deliverable D5.2.

Finally, section 5 draws some conclusions on the simulator development and the metamodelling process and highlights the next steps.

1.1 Acronyms

Table 1: List of Acronyms

Acronyms	Definition
3D	three dimensional
ACC	area control centre
AMAN	arrival manager
ANSP	air navigation service provider
AOC	airline operating centre
ATC	air traffic control
ATCO	air traffic controller
ATFM	air traffic flow management
ATS	air traffic services
CASA	Computer Assisted Slot Allocation
CDM	collaborative decision making
CODA	Central Office for Delay Analysis
CTA	calculated time of arrival
CTO	computed time over
DAC	dynamic airspace configuration

Acronyms	Definition
DCB	demand capacity balancing
DDR2	demand data repository (version 2)
DMAN	departure manager
EAMAN	Extended Arrival Manager
ETO	estimated time over
EU	European Union
FDR	Flight Departure Reordering
FP7	Framework Programme 7
FPFS	first planned first served
GP	Gaussian process
IATA	International Air Transport Association
ICAO	International Civil Aviation Organisation
IFPS	Integrated Initial Flight Plan Processing System
ISTOP	
KPI	key performance indicator
NN	neural network
RECAT	wake turbulence re-categorisation
SESAR	single European sky ATM research
SFP	Selective Flight Protection
TOD	Time of Descent
TMA	terminal manoeuvring area
UDPP	user driven prioritisation process
WP	work package

2 Micromodelling process

2.1 Flitan core concept

This document provides relevant information to understand Flitan core concept as well as the various software enhancement carried out to support various European projects.

Flitan executes as a standalone server where scenarios can be loaded, processed and evaluated in a consistent and repeatable manner.

2.1.1 Background

2.1.1.1 TITAN project

Flitan was initially developed to support the turnaround analysis for an EU FP7 project called TITAN that involved several major European partners. The original name of Flitan was indeed TITAN. It was developed to investigate future operational concepts relating to aircraft turnaround. In simple terms, the turnaround process involves a set of activities that apply to aircraft in the time period between arriving at any given airport and leaving for a subsequent operation. However, instead of considering turnaround as a black-box process represented by a sequence of operations that are required from the time that an aircraft is stationary at the airport with the wheel chocks in place, to the time that the chocks are removed and the aircraft is cleared to leave by ATC, the TITAN model provides a flexible and detailed capability that is able to capture and simulate turnaround processes that are made up of a complex set of often interdependent processes.

2.1.1.2 Flite project

Subsequently, TITAN was adapted to support the SESAR WP-E Flite project with the aim to investigate FlightPath 2050 4-hrs door-to-door target. Given the fundamental changes introduced into TITAN core concepts in order to support Flite project, the tool was renamed Flitan.

To realistically evaluate the ability of European ATM system to handle the different traffic growths for the year 2050 based on EUROCONTROL Challenges of growth, ISA extended the scope of TITAN to include the airborne segment of the ATM network. As part of the software development, Flitan included the following features:

- Modelling of the ATM network as a connected system of nodes where each node is characterised by its occupancy time function and capacity
- Modelling of flight time distribution in the TMA and the En-route in order to predict both the estimated time over and transition time at each node.
- Use of statistical data to derive mean and variance of time distribution in the en-route.
- Use of statistical data to derive mean and variance of time distribution in the TMA.
- Feed these results back to the network-level input file system.

ISA also developed time distribution functions for more than 250 aircraft models for various ATM nodes using a variety of fast-time simulation exercises, analytical tools and available data.

2.1.1.3 NOSTROMO project

In support of NOSTROMO project, Flitan is being again enhanced to support the simulation analysis of two SESAR solutions namely;

- PJ.08-01 - Management of dynamic airspace configurations,
- PJ.02-08 - Traffic optimisation on single and multiple runway airports.

In the framework of the NOSTROMO project, many enhancements have been introduced into Flitan:

- The ability to use ALL-FT+ as the primary traffic data information,
- The ability to read airspace information using the Demand Data Repository (DDR2),
- The ability to model both airspace and runway configurations,
- The ability to activate both airspace and runway configurations according to planned schedule,
- The ability to compute the 4D flight sector crossings,
- The ability to model dependent runways operations,
- The ability to balance demand and capacity,
- The ability to generate various metrics in support of post simulation analysis for both SESAR solutions.

The ultimate goal of Flitan in support of NOSTROMO project is to assess a large set of combinations of both airspace and runway configurations in order to train the metamodel to select the best configurations for a given traffic demand.

2.1.2 Flitan simulation engine

Basically, Flitan abstracts the ATM system as a network of nodes and connectors where flights travel safely and efficiently. At the highest level of abstraction, each node represents an ATM entity be it an airport, sector, etc. A node is primarily characterised by its occupancy time distribution function which defines the nominal time for the process it represents to be accomplished (e.g. the mean time to use a node) and the possible variance in that time. A Flitan connector is responsible for routing flights between adjacent nodes.

A flight starts its journey at an origin node (e.g. departure airport) and passes through a series of nodes and connectors until it reaches the final node (e.g. arrival airport). At each node, its time distribution function is sampled to determine the time needed for the flight to use the node. The time distribution is defined for a city pair and a specific aircraft model. For example, all the flights with the same aircraft model travelling between the same city pair will share the same distribution functions.

Since the implementation of SESAR solution PJ08-01, Flitan uses a full 4D trajectory flight model. Consequently, the time distribution function of a sector is precisely computed from the flight's sector crossings profile. So, the time a flight will need to use a sector is the difference between the sector exit time and the sector entry time.

The underlying Flitan simulation engine is based on a generic Business Process Modelling (BPM) capability and is seeded using a set of flight trajectories, airport and airspace data. Once active, each flight progresses through the network from node to the next node via connectors until it reaches its final destination.

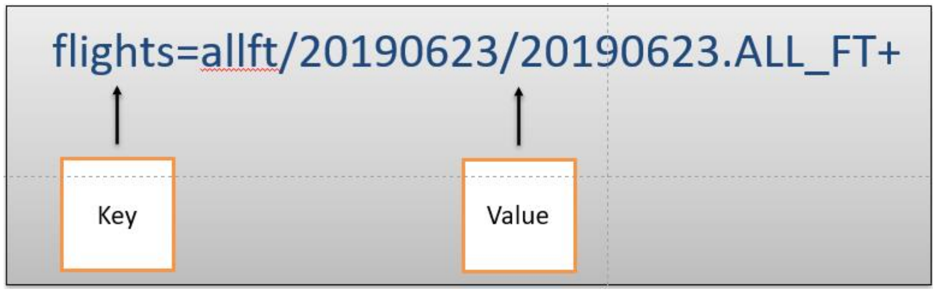
The simulation engine is designed using a discrete event simulation engine- a type of simulation engine where events within the simulation are scheduled to occur at particular time in the future and each event is tied to a piece of code to be executed once that time arrives. It is comprised of a clock, an event list and the event scheduling and dispatch system.

The scheduling-dispatch system acts upon the head of the event list which is always equal to the current time. Each piece of code that is tied to the event being processed is able to create new events on the schedule either at the current time, or at some time in the future. Once all events at the current time are exhausted the clock steps to the next discrete point in time (the next position in the list) and events that are scheduled at this new time in the simulation are executed. A simulation continues to run until there are no events left in the system to dispatch, at which point the simulation is considered to be complete.

An in-depth discussion of the software enhancement to Flitan simulation engine can be found in the sections describing Flitan implementation of SESAR solutions PJ08-01 and PJ02-08.

2.1.2.1 Data

Flitan uses a properties data file to provide key simulation and functionality settings that together define a particular scenario. The keys are predefined and their values allow Flitan to access the file containing the information in order to build the data model that correctly maps to the key. For example, the line in the following figure instructs Flitan to construct the flight object model using the provided file.



```
flights=allft/20190623/20190623.ALL_FT+
```

Figure 1. Instruction line in Flitan.

The value part mostly points to a file, however, there are situations where the value points to ISO date as in the case of the start and end Flitan keys. The table below provides a list of keys used by Flitan.

Table 2. List of keys used by Flitan.

Key	Description
airports	This key provides the list of airports to be simulated in Flitan
flights	This key provides the list of flights to be simulated in Flitan. Flitan reads ALL-FT+ format version 5

EUROPEAN PARTNERSHIP

Co-funded by the European Union

Key	Description
airspaceDelayLookup	The airspaceDelayLookup key points to a file containing information related to the time distribution in the en-route airspace. Since the implementation of SESAR solution PJ08-01, Flitan derives this information directly from the flights 4D trajectory data. Therefore, this key will be phased out in the coming Flitan version.
arrivalTimeDelayLookup	This key points to a file that provides a list of arrival TMA transit time distributions based on departure airport, arrival airport and aircraft model
departureTimeDelayLookup	This key points to a file that provides a list of departure TMA transit time distributions based on departure airport, arrival airport and aircraft model
log	This key points to an output file to be used to store various Flitan logging information. The file is mainly used for debugging information which is mainly useful for developers when looking for issues that may require fixing.
model	The key model should always be equal to v2Model to instruct Flitan to use enhancement introduced to support NOSTROMO project
runwayOccupancyLookup	This key provides a list of runway occupancy time distributions based on departure airport, arrival airport and aircraft model
runwayDependencyLookup	This key points to a file that provides a list of runway pairs with dependent operations. The pair of runways have to coordinate their operations. This is a new key introduced to support SESAR solution PJ02-08
runwayConfiguration	This key points to a file that provides a list of runway configurations and their global capacity. This is a new key introduced to support SESAR solution PJ02-08
runwayConfigurationCapacity	This key points to a file that provides the runway configuration time-based capacity. This is a new key introduced to support SESAR solution PJ02-08
runwayOpeningScheme	This key points to a file that provides a list of runway configuration activation plan. This is a new key introduced to support SESAR solution PJ02-08
start	This key provides the simulation start time. For example, start=2019-06-22T00:00:00
end	This key provides the simulation end time. For example, end=2019-06-22T23:59:59
taxiTimeLookup	This key points to a file that provides a list of taxi time occupancy time distributions based on departure airport, arrival airport and aircraft model

Given that the DDR2 has a large set of files, describing them in the properties data file using keys will be cumbersome. Therefore, Flitan looks for a directory under the scenario directory called 'rnest' to read all the DDR2 data files. Flitan uses DDR2 file extensions to determine which file will be used to build a specific object model. For example, a file with the .cos extension will be used to build the airspace configuration opening scheme object model, while a file with .spc will be used to build the airspace object model.

For a description of the list of DDR2 file extensions used by Flitan , please refer to the section dealing with Flitan implementation of SESAR solution PJ08-01.

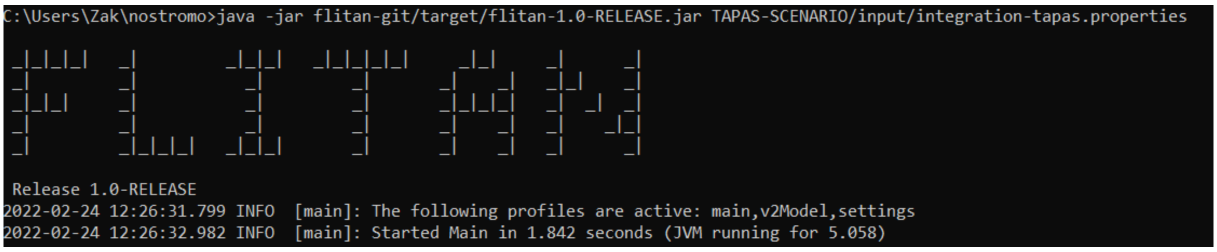
For a description of the output files, please refer to the sections describing Flitan implementation of SESAR solutions PJ08-01 and PJ02-08.

2.1.2.2 Running Flitan

Flitan executes as a standalone server where scenarios can be loaded, processed and evaluated in a consistent and repeatable manner. The scenario is defined using a properties data file.

Flitan accepts only a single argument which is the scenario properties data file. Here is how to run Flitan to execute a scenario which its properties data file is called integration-scenario.properties:

java -jar flitan-1.0-RELEASE.jar integration-scenario.properties



```
C:\Users\Zak\nostromo>java -jar flitan-git/target/flitan-1.0-RELEASE.jar TAPAS-SCENARIO/input/integration-tapas.properties

 _|_|_|_|   _|         _|_|_|   _|_|_|_|_|     _|_|     _|      _|
 _|         _|           _|         _|       _|    _|   _|_|    _|
 _|_|_|     _|           _|         _|       _|_|_|_|   _|  _|  _|
 _|         _|           _|         _|       _|    _|   _|    _|_|
 _|         _|_|_|_|   _|_|_|       _|       _|    _|   _|      _|

 Release 1.0-RELEASE
2022-02-24 12:26:31.799 INFO  [main]: The following profiles are active: main,v2Model,settings
2022-02-24 12:26:32.982 INFO  [main]: Started Main in 1.842 seconds (JVM running for 5.058)
```

Figure 2. Flitan execution window.

When Flitan starts execution, it logs a variety of messages to allow users monitor its progress as well as to track error messages. Flitan may raise an exception that completely stops its execution if the error is considered fatal to the completion of the simulation. Flitan is very robust, and it is therefore very unlikely to encounter a crash if the scenario is well set up and all the files are correctly prepared. We run thousand of simulation iterations with Flitan in a row without experiencing a single crash.

2.1.3 Dynamic airspace configuration

This section provides the Flitan implementation of the SESAR Solution PJ-08.01 "Dynamic Airspace Configurations". The main aim of this solution is to allow Air Navigation Service Providers (ANSP) to organise, plan and manage airspace configuration in a flexible manner that increases capacity and reduces delays without impacting traffic trajectories.

SESAR solution PJ08-01 is articulated around four key concepts:

1. ***Dynamically designed sectors*** tailored to specific flow patterns,
2. ***Dynamic sector configurations*** adapted to respond flexibly to available air traffic controller (ATCO) resources, changes to traffic demand, and performance objectives,

Co-funded by the European Union

3. ***Dynamic mobile areas*** which are temporary reserved volumes of airspace designed to segregate the military activities from the civil air traffic,

4. ***Fully integrated CDM process*** between civil and military actors enabled by automated support tools.

The scope of the PJ08-01 implementation in Flitan is essentially focused on the second key concept "Dynamic sector configurations" and their assessment. Dynamic sector design and sector capacities computation are not part of the Flitan implementation of PJ.08-01.

In implementing SESAR solution PJ.08-1, Flitan targets two specific objectives:

1. Introduce airspace configurations functions into the Flitan simulation engine,

2. Train the metamodel through the assessment of a large set of possible combinations of sectors.

The ultimate goal is that for a given traffic demand the metamodel will be able to select and provide a ranking of suitable sector configurations for a given traffic demand.

2.1.3.1 Fundamental Concepts

2.1.3.1.1 Elementary Sector

An elementary sector is 3D airspace designed as a unit volume of air traffic control, it cannot be split further down. However, the combination of adjacent elementary sectors can form other controllable sectors. The design of elementary sectors is an important step in the dynamic airspace configuration process:

- They are designed in both lateral and vertical dimensions to meet traffic flow and airspace complexity,

- They are designed also to adapt to seasonal specificity (i.e, Morning/Evening, Weekday/Weekend, Summer/Winter, etc),

- They are designed to balance the workload among controllers.

2.1.3.1.2 Configured Sector

A configured sector is 3D airspace volume designed for air navigation service (ANS) provision tasks. They are constructed by combining several adjacent elementary sectors. During the airspace configuration process, configured sectors have capacities and time spans.

2.1.3.1.3 Airspace Sector Configuration

Several possibilities of combining elementary sectors that cover the whole Air Traffic Control (ATC) Centre exist. Any of these possibilities is called an airspace configuration with the constraint that all the resulting configured sectors are formed of adjacent elementary sectors and that resulting sectors they cover the whole ATC Centre.

Example:

Consider the following example where a fictional ATC Centre is composed of four elementary sector, as depicted in the figure below.

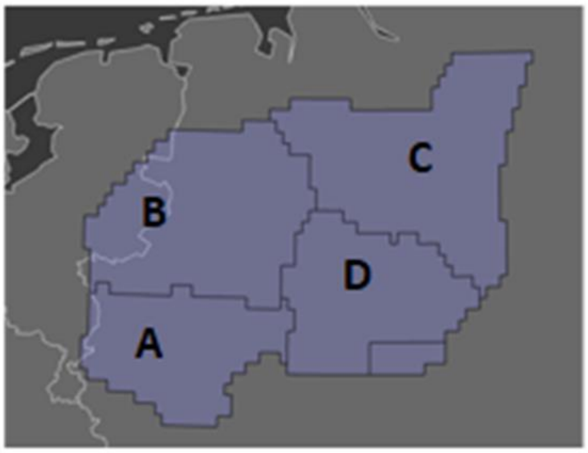


Figure 3. Fictional ATC centre with four elementary sectors.

The table below highlights the possible airspace sector configuration for this ATC Centre. In the table, the sign "+" denotes the combination of elementary sectors into configured sectors. For example, configuration CNF1 is composed of a single sector which is the result of combining elementary sectors A, B, C and D.

Table 3. Possible configurations for fictional four-sector ATC centre.

Sector Configuration ID	Configured Sectors
CNF1	A + B + C + D
CNF2A	A + B
	C + D
CNF2B	A + B + D
	C
CNF2C	A + D
	B + C
CNF3A	A
	B
	C + D
CNF3B	A + B
	C
	D
CNF4	A
	B
	C
	D

2.1.3.1.4 Opening Scheme

An opening scheme defines an orderly sequence of airspace configurations and their corresponding activation periods. An ATC Centre usually has several opening schemes that can be activated to meet specific objectives (i.e. ATC resources availability, seasonality and traffic demand).

Example: From the above fictional ATC Centre, Table 4 and Figure 4 represent possible opening schemes.

Table 4. Example of opening scheme.

Sector Configuration ID	Start of Activation Period	End of Activation Period
CNF1	00:00	03:29
CNF2A	03:30	06:59
CNF3B	07:00	13:59
CNF2B	14:00	17:59
CNF3A	18:00	21:59
CNF2A	22:00	23:59

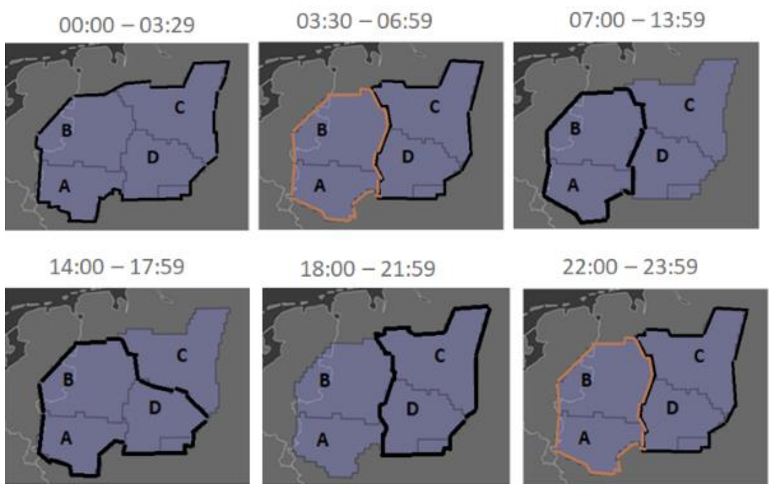


Figure 4. Example of opening scheme.

2.1.3.1.5 Demand Capacity Balancing (DCB)

When traffic demand is higher then the available capacity, flights are delayed. The process of allocating new entry times to upcoming flights in order to respect declared capacity is called Demand Capacity Balancing (DCB).

2.1.3.2 Solution Design and Implementation

Flitan has been developed to support network assessment and executes as a standalone server in which scenarios are loaded, processed, and evaluated in order to provide analytical views of each of the involved network nodes.

Co-funded by the European Union

2.1.3.2.1 Airspace Node

2.1.3.2.1.1 Before PJ08-01

One of the Flitan network node is the so-called Airspace Node which emulates the operations in the enroute airspace. Before Flitan implementation of solution PJ08-01, this node was used as a direct link between departure TMAs and arrival TMAs with enough capacity to handle any traffic demand.

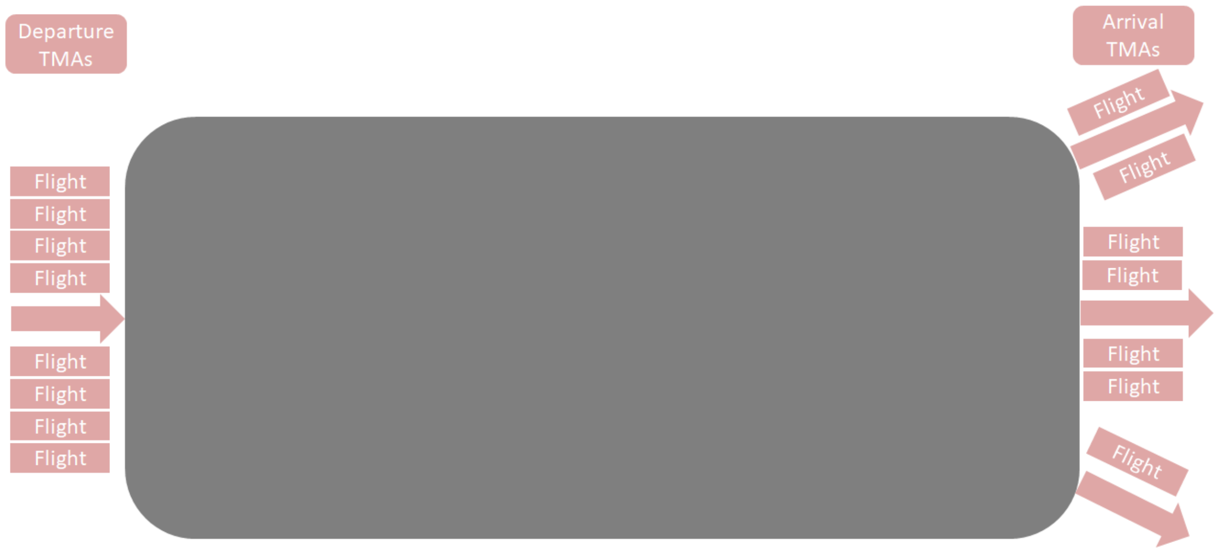


Figure 5. Flitan airspace node prior to PJ08-01 implementation.

The Airspace Node uses a well-calibrated set of time distributions to determine the amount of time a specific flight will spend in the enroute airspace. Several fast-time simulations were performed in order to derive a database of time distributions for more than 250 different aircraft types based on departure and arrival airports. The time distributions can be expressed as:

- a normal distribution has a mean and standard deviation values; when used, travel times of all flights with the specific aircraft type linking the specific city-pair are randomly selected based on the normal distribution values,
- a fixed distribution has only one constant value; when used, all travel times of all flights with the specific aircraft type linking the city-pair are equal to this constant value.

Prior to a flight's entry to the airspace node, Flitan will pick up the time distribution associated with the candidate flight using as the search attributes the flight's departure and arrival airports as well as its aircraft type. Once the time distribution is allocated, Flitan will use it as the flight's travel time from departure terminal maneuvering area (TMA) until the arrival TMA.

Table 5. Example of flight time draw from distributions.

Time distributions (milliseconds)	Description
EBBR;LEBL; A320;distribution:normal 5887000 180000	The flights operated by A320 aircraft departing from EBBR and arriving to LEBL will use this random time distributions where the mean flight time is 5887 seconds and standard deviation is 180 seconds..

Time distributions (milliseconds)	Description
LIMC;EGLL;MD82;distribution:fixed 4778000	The enroute flight times of all MD82 aircraft departing from LIMC and arriving to EGLL are equal to 4778 seconds.

The travel time for any node in Flitan is based on suitable time distributions including;

- taxi in and taxi out time,
- runway occupancy time,
- departure and TMA travel time,
- enroute travel time.

Since the time distributions were used to compute travel times, the original flight object model was also very basic and did not use 4D trajectory information to derive time information. Only the flight schedule information were used:

- estimated departure time,
- callsign,
- departure airport,
- departure runway,
- arrival airport,
- arrival runway,
- aircraft model,
- ICAO Wake Category,
- RECAT Wake Category,
- entry flight level,
- cruise level,
- exit level.

Flitan is capable of simulating flight operations by simply using the flight schedule information and time distributions for each Flitan node as shown below:

1. Flights are added to Flitan sequencer object in order based on their departure times,
2. When a flight scheduled departure time is reached, Flitan performs the following tasks in time order as the flight progresses from the departure airport to arrival airport:
    a. Estimate the flight taxi out time,

b. Estimate the flight take-off time taking into account the runway departure queue,

c. Block runway while in use,

d. Estimate the departure TMA travel time,

e. Estimate the enroute travel time,

f. Estimate the arrival TMA travel time,

g. Estimate the runway landing time,

h. Block runway while in use,

i. Estimate the taxi-in time.

This was the original Flitan simulation engine set up that was successfully used to model complex studies such as FlightPath 2050.

2.1.3.2.1.2 Modelling PJ08-01

The original Flitan simulation engine set up was enough under some very specific simulation assumptions like FlightPath 2050 where enroute capacity was assumed to be large enough to handle at least 25 million flights a year. However, in order to model PJ08-01 SESAR solution, the original simulation engine is no longer sufficient as we need to model both airspace configurations and 4D flight trajectories as proper Flitan object models.

Airspace Configuration

PJ08-01 is centered on the concept of dynamic airspace configuration (DAC), which allows ANSPs to organise, plan and manage airspace configurations with enough flexibility to respond to changes in traffic demand.

To be able to assess airspace configurations, there is a need for Flitan to model at least the following elements:

- Elementary sector,
- Configured sector,
- Capacity,
- Sector configuration,
- Opening scheme.

With all these new object models in place, it will be possible to assess many configurations with the ultimate goal to deploy the optimum one.

4D flight trajectories

One of the key elements in assessing sector configurations is the ability to provide an improved traffic prediction in order to allocate efficiently the resources that meet actual traffic demand.

The past Flitan flight object model which is based only on flight schedule information does not provide for a mechanism to accurately predict demand within a specific airspace volume.

4D trajectories offer the opportunity to improve traffic demand prediction and therefore allocate adequately resources to meet traffic demand and ultimately optimise sector configurations.

New model

For this purpose, airspace node needs to be populated with elementary sectors (the building block of any airspace volume) and 4D flight trajectories (not just flight schedule information).

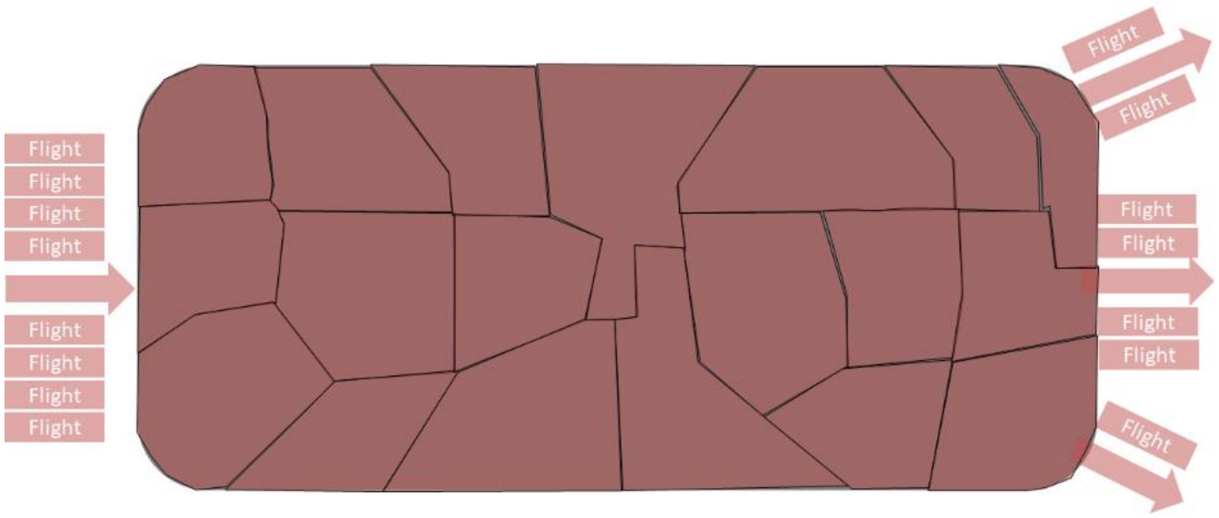


Figure 6. Flitan Airspace node to accommodate PJ08-01.

In this new airspace node model, flight enroute travel times are no longer dependent on calibrated and pre-defined time distributions. They are actually dependent on the exact times of flight trajectories crossing the elementary sectors.

Once all flights have been loaded into memory, Flitan will then perform all sector crossing calculation for all flights and elementary sectors in the simulation. For a specific flight, Flitan determines the travel time within any elementary sector by computing the difference between elementary sector's exit time and elementary sector's entry time. Therefore, the travel time within the airspace node for a defined flight is the sum of the travel times of all crossed elementary sectors.

Every sector configuration allows to deploy a different sectorisation by scheduled time, where existing sectors may be grouped or split into different sectors. The advantage of using elementary sectors as the basic volume unit of sector crossing calculation is that configured sectors (i.e. sector deployed by a specific sector configuration) entry and exit times can be determined by a simple process as explained here. During the simulation run, an elementary sector changes its owner as sector configurations are activated and/or deactivated. Every sector configuration is composed of configured sectors and each configured sector is composed of one or many elementary sectors. Therefore, at any time during the simulation run, we can determine the flight's entry time to and exit time from a configured sector by requesting these times from the elementary sectors they own, sorted them by order and extract the first time as the entry time and the last one as the exit time. However, sector crossing calculation would have been very expensive if the calculation has to be performed at each sector configuration activation.

2.1.3.2.1.3 Flitan workflow

Two sources of information are used in support of the dynamic airspace configuration (see Figure 7)

- Airspace data in Demand Data Repository (DDR2) format,
- Traffic data in ALL-FT+ format (version 5).

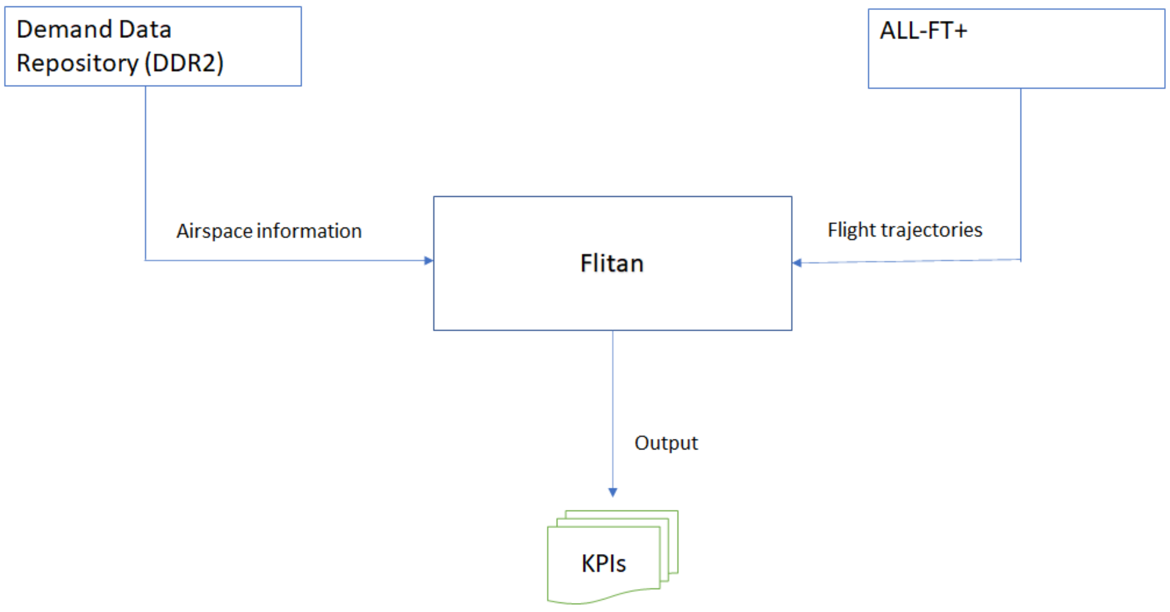


Figure 7. Flitan workflow.

2.1.3.2.1.4 Airspace data

The Demand Data Repository (DDR2) is used as the primary source of airspace data to construct the Flitan DAC object model. Flitan processes many of DDR2 files (see the below table); however, there are two categories of DDR2 files are that of most important to Flitan DAC object model:

- Airspace environment category (see Table 6) is composed of information that defines ATC centres, their corresponding elementary sectors,
- Capacity category.

Table 6. Airspace environment information.

Key	Description	DDR2 File name extension	Category
AIRBLOCK	Provides a 2D definition of the airblocks used in the scenario.	.gar	AIRSPACE
AIRSPACE	Describes how elementary sectors can be collapsed (combined) for use in different airspace configurations.	.spc	AIRSPACE

Key	Description	DDR2 File name extension	Category
CAPACITY	Defines the (hourly) sector and traffic volume capacities for the scenario and the time periods for which those capacities apply.	.ncap	CAPACITY
CONFIGURATION	Provides the set of available airspace configurations for use by ATC in the scenario.	.cfg	AIRSPACE
CONTROLLER_COUNT	Provides information on the number of available ATC controllers for each of the airspace configurations in the scenario.	.ncnc	AIRSPACE
PEAK_OCCUPANCY	Defines the peak occupancy counts that should not be exceeded at any time in the scenario. For any period that the max occupancy is exceeded OTMV max occupancy alerts occur.	.nocp	CAPACITY
SCHEMA	Defines the start and end times when specific sector configurations are used in the scenario.	.cos	AIRSPACE
SECTOR	Defines how elementary sectors are constructed from airblocks.	.gsl	AIRSPACE
SUSTAINED_OCCUPANCY	Defines the maximum sustained occupancy for any sector/traffic volume in the scenario. If the max sustained occupancy is exceeded for more than n-periods in a given window an OTMV sustained capacity alert will occur.	.nocs	CAPACITY
TWENTY_MINUTE_CAPACITY	Defines the (20-minute) sector and traffic volume capacities for the scenario and the time periods for which those capacities apply.	.ncap20	CAPACITY

Flitan uses internally a key-extension mapping to associate a specific DDR2 file to a specific Flitan DAC object model. For example, to construct elementary sector object model, Flitan will load and process a file with .gsl extension. If two files exist with the same extension then only one file will be picked up to construct the object model. Therefore, Flitan users must ensure that there is only a unique file for each DDR2 extension in the simulated DDR2 folder.

2.1.3.2.1.5 ALL-FT+ data

Prior to the implementation of PJ08-01 SESAR solution, Flitan used a very basic flight object model that contains mainly scheduled information. In order to implement DAC, a much more complex flight object model was required. After analysis of various flight format, we finally settled for the ALL-FT+ data format.

The Table 7 shows the main ALL-FT+ fields used by Flitan to construct its flight object model.

Table 7. ALL-FT+ fields used by Flitan.

Field Number	Description
1	Departure airport ICAO code
2	Arrival airport ICAO code
3	Aircraft Identifier
5	Aircraft model
7	Integrated initial flight plan processing system (IFPS) Identifier
18	Estimated Off Block Time
79	Departure runway
80	Arrival runway
85	Flight's 4D profile that contains a sequence of points defined by latitude, longitude, altitude and estimated time over.

For the sake of uniqueness, Flitan constructs the callsign of each flight by concatenating the aircraft identifier (field #3) and IFPS identifier (field #7) using "-" as a separator.

2.1.3.2.1.6 Flitan KPI Files

Flitan generates several files that contain information that allows subject matter experts to derive indicators about the evaluated scenario. Flitan also generates a large file that is mainly useful for Flitan software developers when looking to understand and fix issues (if they appear).

Average open sector file

This file provides an indication of the average number of open positions in a unit of control during the whole simulation run. The file format is as follow:

- unit of control name,
- average number of open positions.

Configuration delay file

This file captures the total delay incurred by the activation of a specific airspace configuration. The file format is as follow:

- unit of control refers to the air traffic control centre where the configuration was deployed such as LECMCTAN,
- airspace configuration name refers to an airspace configuration that has been activated during the simulation run,
- number of opened positions refer to the number of sectors that have been deployed at the activation of the airspace configuration,

- total delay in seconds.

Configuration delay periods file

This file contains more or less the same information as the configuration delay file. It has been generated at the request of DTU university to include also the period of the airspace configuration activation. The file format is as follow:

- unit of control name,
- airspace configuration name ,
- start time of the activation of the airspace configuration,
- end time of the activation of the airspace configuration,
- total delay in seconds.

Flight delay file

This file provides entry information into an open position for each flight: the estimated time over and computed time over the position. The file format is as follow:

- Callsign,
- unit of control name,
- airspace configuration name,
- start of activation time,
- end of activation time,
- configured sector name,
- estimated time over,
- computed time over,
- delay (in seconds).

Sector load file

This file provides sector throughput aggregated over a 20-minute bins from the start to the end of activation period. The file format is as follow:

- unit of control name,
- airspace configuration name,
- configured sector name,
- start of activation time,
- end of activation time

- sequence of flight numbers over a 20-minute period.

Airspace info file

This file is used mainly to check which configurations were activated during the simulation run, their periods of activation, the deployed configured sectors, their constituent elementary sectors and the applied capacity. The file format is as follow:

- unit of control,
- airspace configuration name,
- number of configured sectors,
- list of configured sectors,
- list of constituents elementary sectors for each configured sector,
- list of capacity for each configured sector,
- list of activation periods.

Flitan DAC Simulation Engine

The main steps in Flitan assessment of a given opening scheme are as follow:

1. Read all DDR2 airspace information:
    - elementary sectors,
    - configured sectors,
    - sectors declared capacities,
    - airspace configurations,
    - opening schemes,
2. Read ALL-FT+ flight trajectories information for the corresponding day of simulation,
3. Compute when flights are expected to cross elementary sectors,
4. Initialise Flitan clock time to the start time of the simulation,
5. Enter the Flitan event loop,
    - Extract current event,
    - if the event type is an airspace entry event then perform the following sub tasks:
        i. Identify the current active configuration,
        ii. Identify the configured sector based on active configuration and the crossed elementary sector,

iii. Analyse the demand,

iv. Compare demand to the available capacity,

v. Make any necessary adjustment to the expected times over the configured sector to solve the demand/capacity imbalances,

- Exit the event loop when end time of simulation is reached; otherwise go back to the beginning of the loop to process the next event.

6. Output KPIs metrics in terms of delays and sector throughput.

Flitan uses a simple mechanism to adjust entry times over configured sectors in order to balance traffic demand and capacity as explained below:

1. Every configured sector has its own slot allocation list divided into periods of 20-minute length. These periods are referred to here as 20-minute bin.
2. When a flight is planned to enter a specific elementary sector, Flitan will use the flight Estimated Time Over (ETO) to identify the configured sector that currently owns the elementary sector.
3. If the flight is already in the configured sector slot allocation list then no action is performed. This case can arise when the current and the previous elementary sectors crossed by the flight belong to the same configured sector.
4. Otherwise, Flitan will use the flight's ETO to determine the 20-minute bin that will host the flight.
5. In the case that the identified 20-minute bin has already reached its capacity then Flitan will check the next 20-minute bin for slot availability and the process will continue until a 20-minute bin with enough slot is found..
6. Flitan will then allocate a time slot to the flight and this time will become the flight's Computed Time Over (CTO).

2.1.4 Traffic Optimisation on single and multiple runway airports

This section provides the Flitan implementation of the SESAR solution PJ02-08 "Traffic optimisation on single and multiple runway airports". The main aim of this solution is to enhance airport throughput operations by ensuring that runways operate at their optimum capacity.

SESAR solution PJ02-08 is focused on two key concepts:

1. An integrated runway sequence function to balance arrival flights and departure flights on single runway, dependent runways or parallel runways
2. Use of a runway manager for airports with more than one runway to plan the optimal runway configuration

The aim is to describe the enhancement introduced into Flitan that translates the PJ02-08 requirements into a suitable software framework capable of modelling runway sequencing and runway configurations functions for any type of airport (e.g. single or multiple runway airports) as well as supporting functions to assist in traffic demand prediction, runway occupancy times, and dependent runways operations.

2.1.4.1 Concepts

Airports manage at least one runway to handle departing and arriving aircraft. Busiest airports usually have a very complex structure of runways that are operated under varying wind conditions, daytime or night time operations, inbound or outbound traffic, off peak operations, etc. Additionally, to mitigate safety risk and environmental issues (i.e. noise for the communities surrounding the airport), airports are assigned a maximum number of operations. As a result, airports are required to operate using various runway configurations throughout the day in order to optimise their operations.

For example, Amsterdam Schiphol airport has six runways but only three runways are active at any moment and their configurations change to accommodate traffic pattern. This requires switching runways operations to accommodate either inbound or outbound traffic (i.e. two runways can be used for departures and one runway for arrival and vice versa depending on the peak direction).

Wind direction is another factor that impacts runway configurations. Landings and take-offs are typically conducted into the wind. For instance, when the wind is from the north, north runway configurations will be preferred over others, and landings and take-offs will be performed in a south-to-north direction.

A runway can be unavailable for use because of crosswinds. The ICAO specifies that a runway should not be used if the magnitude of the crosswind exceeds predefined values. The magnitude of crosswind can be computed by simply multiplying the speed of the prevailing wind by the sine of the angle between the wind direction and the runway centerline. In practice, airports have runways in various directions to ensure a higher level of usability.

Runways also can be used in segregated mode (used only for departure or arrival), can be used from both runway ends, or exclusively from one runway end.

Runway configuration, when it is done in a timely manner, can lead to a substantial increase in the amount of traffic an airport can handle. However, the high amount of possible runway configurations makes it difficult even for a skilled subject matter expert to assess and put in place the most optimal configuration. One of the biggest challenges in this respect are:

- selecting the adequate set of runways;
- considering the operation dependencies among the selected runways;
- analysing the environmental, safety and throughput implications;
- determining the orientation of the runways to achieve optimum operations;
- developing a timely schedule of activation and a transition plan that will allow operations to continue smoothly without a decrease in capacity during the runway reconfigurations.

Long-range planning is also a factor to be considered during the runway configurations as too many activation of runway configurations on short time intervals may lead to capacity decrease and considerable safety risks.

The next section provides an overview of the fundamental concepts that impacts runway planning without going into much detail. In all cases, the principal objective is to draw a comprehensive picture of the elements essential in maximizing airport throughput.

2.1.4.1.1 Runway layout

Of special relevance to the runway configurations is the geometric layout of runways which largely dictates the orientation of inbound and outbound traffic and therefore may critically affect operations.

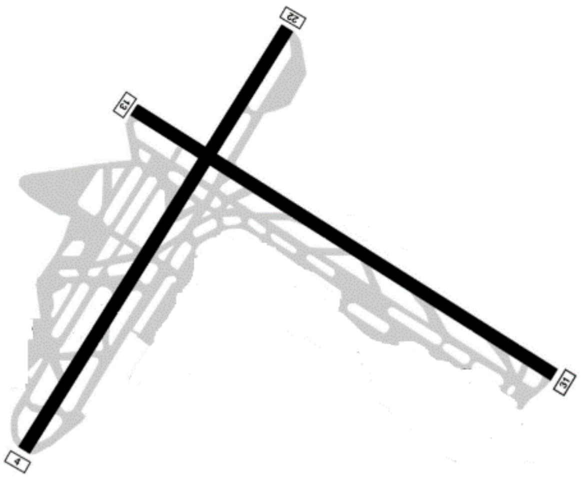


Figure 8. Example of runway configuration.

Every runway end is identified by a two-digit number, which indicates its azimuth in the direction of operations to the nearest 10 degrees. For example, a runway end with a magnetic azimuth of 224 degrees is designated as "Runway 22". The opposite end of runway is designated as "Runway 04", and the runway is referred to as "Runway 04-22". The identification at the two ends of any runway will always differ by 18.

In the case of parallel runways, the runway end designation can be supplemented by letter R (for right), L (for left), or C (for centre) to distinguish between the runways.

2.1.4.1.2 Dependent runways operations

Although the number of simultaneously active runways contributes to the enhancement of airport throughput, the geometric characteristic of the active runways is another critical parameter in selecting the right runway configurations. The proximity and orientation of runways are factors that can increase their operation dependency. Obviously, the less dependency between active runways, the more airport throughput, which subsequently leads to an optimum runway configuration.

At major airports, the runway system can consist of two or more runways that have particular geometric layout that limit their simultaneous operations: operations on a runway have to be coordinated with operations on the adjacent runways. This is not an issue at single-runway airports.

Parallel runways

Parallel runways consist of two or more runways whose centre lines are parallel. The selection of runway configuration may be impacted by the operation mode of the parallel runways:

- Simultaneous parallel approaches,

- Simultaneous parallel departures,
- Segregated parallel approaches/departures,
- Semi-mixed parallel operations,
- Mixed mode parallel operations.

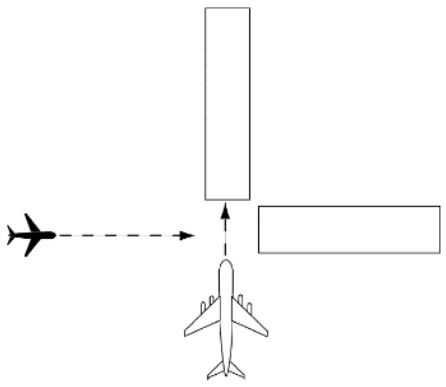


Figure 9. Example of intersecting runways.

Intersecting runways

Airports with intersecting runways are more difficult to optimise. Operations on intersecting runways may impact the deployment of an optimum runway configurations as operations need to be coordinated. From the figure, it is clear that when both runways are active, arrivals on both runways are dependent operations and cannot take place at the same time. Additionally, the capacity of the runway pair varies depending on the direction in which the operations take place (i.e. the landings on the other runways ends are independent operations and may be the preferred choice for landings if possible).

2.1.4.1.3 Runway length

The runway length is an important factor in determining which runway should be assigned to an aircraft type. It is known that heavy aircraft requires longer distances for the take-off and landing. The wind direction also impacts these distances: flying facing the wind will require shorter distances while a tailwind will require longer distances.

The relationships between the required runway length and aircraft weight and the prevailing wind are quiet obvious; but other factors affect the runway length required for the landing or take-off of any aircraft on any given day:

- Weather temperature - High temperatures create lower air densities, resulting in lower output of thrust and reduced lift, this increasing runway length required.
- Airport elevation - The higher the elevation of an airport, the lower the air density and therefore the longer will be the runway required
- Runway slope - An airplane taking off on uphill slope requires more distance than one on a level or downhill slope.

- Surface condition - a wet runway will increase the runway length required, especially on landing
- Flap settings

Consequently, runway length should be considered when assigning a runway to an aircraft. For example, at Schiphol airport, the length of the runway 04-22 is short and is used exclusively for general aviation traffic.

2.1.4.1.4 Runway sequence function

The objective of the runway sequence function is to allocate runways and usage time to aircraft in a way that maximize the airport throughput. This involves which runways will be used (if more than one runway exist), and in which direction. To select the active runways and the directions of operations, a combination of criteria may be used such as maximizing airport throughput, minimizing noise, reducing the impact of weather conditions, etc. Traffic mix is another important factor in determining the right sequence at a specific runway. In major airport, traffic mix may involve a variety of aircraft models from light to heavy which may lead to deviation from the standard first-come, first-served (FCFS) queueing model.

2.1.4.2 Solution design and implementation

SESAR solution PJ02-08 focuses on two key concepts:

1. An integrated runway sequence function to balance arrival flights and departure flights on single runway, dependent runways or parallel runways
2. Use of a runway manager for airports with more than one runway to plan the optimal runway configuration

Flitan has already an integrated runway sequence function to balance both arrival and departure flights on airport runways. The core function of the Flitan sequence function is to assign flights to runways in order to balance the workload among the available runways. In this manner, Flitan prevents overloading one runway and under utilizing another. Another aspect of Flitan sequencing function worth mentioning is that once a runway is assigned to a flight, its order will not change in the runway queue (i.e. first-come, first-served).

However, a runway configuration manager function was not available in Flitan prior to the implementation of PJ02-08.

2.1.4.2.1.1 Flitan objective of implementing PJ02-08

In implementing SESAR solution PJ02-08, Flitan targets three specific objectives:

1. Introduce both the runway configuration manager and runway dependency operations functions into the Flitan simulation engine
2. Enhance Flitan sequencing function
3. Train the metamodel through the assessment of a large set of possible combinations of runway configurations

The ultimate goal is that for a given traffic demand the metamodel will be able to select and provide a ranking of runway configurations to airport decision-makers.

2.1.4.2.1.2 Flitan Runway configuration modelling

For an airport with several runways, it is important to distinguish between the number of runways at the airport and the number of runways that are active at any given time. For example, Amsterdam Schiphol airport has six runways but no more than three runways are ever active at the same time. Each operational combination of runways is called a runway configuration.

Consequently, a runway configuration is the process of planning and selecting a specific set of runways to be operated at any given time. The selection of active runways are usually done to meet specific objectives such as traffic demand, weather conditions, resource constraints, etc. An airport can change runway configurations several times during an operational day.

In Flitan, a runway configuration is defined by its name, active runways and type of operations. The type of operations is either:

1. ARRIVAL when the runway is used exclusively for landings,
2. DEPARTURE when the runway is used exclusively for departures,
3. MIXED when the runway is used for both take-offs and landings,
4. INACTIVE when the runway is not used.

For example, to describe Madrid Barajas north configuration where runways 36R and 36L are used for take-offs while runways 32R and 32L are used for landings, Flitan uses the definition depicted in Figure 10.

```
LEMD;NORTHCONF;R;14R-32L;32L;ARRIVAL
LEMD;NORTHCONF;R;14L-32R;32R;ARRIVAL
LEMD;NORTHCONF;R;18R-36L;36L;DEPARTURE
LEMD;NORTHCONF;R;18L-36R;36R;DEPARTURE
LEMD;NORTHCONF;R;14R-32L;14R;INACTIVE
LEMD;NORTHCONF;R;14L-32R;14L;INACTIVE
LEMD;NORTHCONF;R;18R-36L;18R;INACTIVE
LEMD;NORTHCONF;R;18L-36R;18L;INACTIVE
```

Figure 10. Runway configurations description.

In the above example, the fields are as follow:

1. field #1 is the airport ICAO code
2. field #2 is the runway configuration name
3. field #3 (the letter 'R') indicates that the definition is related to runway configuration
4. field #4 is the runway identifier
5. field #5 is the runway end

6. field #6 is the operation type

2.1.4.2.1.3 Flitan Runway configuration capacity modelling

A change in runway configuration results in the change in the runway system capacity. Therefore, Flitan provides two methods to define the runway configuration capacity.

Global capacity

The global capacity allows to define the runway configuration independently of the operation time. It has three capacities:

1. Maximum of arrivals per hour,
2. Maximum departures per hour,
3. Global movements per hour.

Flitan defines the global capacity of a runway configuration by also specifying whether the runway configuration is optimum. For example, Madrid airport has two runway configurations: north configuration and south configuration. Flitan defines the Madrid Barajas airport runway configuration capacity as follow:

```
LEMD;NORTHCONF;C;48;52;100;;yes

LEMD;SOUTHCONF;C;48;52;100;;no
```

In the above example, the fields are as follow:

1. field #1 is the airport ICAO code
2. field #2 is the runway configuration name
3. field #3 (the letter 'C') indicates that the definition is related to runway configuration capacity
4. field #4 is the maximum arrivals per hour
5. field #5 is the maximum departures per hour
6. field #6 is the global movement per hour
7. field #7 is used for comments
8. field #8 is either yes or no to indicate whether the configuration is optimum

Time-based capacity

The time-based capacity is similar to the global capacity however the capacity varied with time. Below is an example of time-based capacity for Madrid north configuration:

```
#1    ;#2    ;#3    ;#4    ;#5    ;#6;    #7

LEMD;00:00;04:59;NORTHCONF;20;20;38

LEMD;05:00;05:59;NORTHCONF;19;29;
```

LEMD;06:00;21:59;NORTHCONF;48;52;

LEMD;22:00;22:59;NORTHCONF;28;22;

LEMD;23:00;23:59;NORTHCONF;20;20;38

In the above example, the fields are as follow:

1. field #1 is the airport ICAO code
2. field #2 is the start time of the capacity enforcement
3. field #3 is the end time of the capacity enforcement
4. field #4 is the runway configuration name
5. field #5 is the maximum arrivals per hour
6. field #6 is the maximum departures per hour
7. field #7 is the global movement per hour

When the global movement per hour in the time-based capacity is not defined, Flitan will compute it as the sum of maximum departures per hour and maximum arrivals per hour.

It should be noted that the time-based capacity has precedence over the global capacity. The global capacity is used only when no time-based capacity is found for the operational time under analysis

2.1.4.2.1.4 Flitan Runway configuration activation plan modelling

Flitan uses a runway configuration activation plan to select a specific runway configuration. The activation plan indicates to Flitan which runways should be selected and what capacities should be applied for arrivals and departures during the activation period.

Amsterdam Schiphol airport (EHAM) has four runway configurations: Off-peak, Outbound-peak, Inbound-peak and Night-peak. One can define EHAM runway configuration activation plan as follows:

#1   ;#2   ;#3   ;#4

EHAM;00:00;04:30;NIGHTPEAK

EHAM;04:30;05:00;OFFPEAK

EHAM;05:00;05:20;OUTBOUNDPEAK

EHAM;05:20;07:20;INBOUNDPEAK

The fields are:

1. field #1 is the airport ICAO code
2. field #2 is the runway configuration activation start time
3. field #3 is the runway configuration activation end time

4. field #4 is the runway configuration name

2.1.4.2.1.5 Flitan Runway dependency modelling

Runways layout may impact airport operations in crucial ways. For example, when it comes to runways that either intersect physically or the projection of their centerlines intersect, operations on one runway may block operations on the other runway and as a consequence, operations on both runways involved cannot take place at the same time. Therefore operations need to be coordinated on these runways. Clearly, as shown in the figure under section ‘intersecting runways’, the coordination may not involve operations from both runway ends but only operations from/to the runway ends which are closer to the intersection.

Flitan allows the modelling of the runway dependency operations using information about the involved pair of runways, the type of operations, and the directions of operations as follow:

AIRPORT-ID;RUNWAY;RUNWAY-END;OPERATION-TYPE;AIRPORT-ID;OTHER-RUNWAY;OTHER-RUNWAY-END;OTHER-RUNWAY-OPERATION-TYPE

1. field #1 is an airport ICAO code
2. field #2 is a runway identifier of the designated airport in field #1
3. field #3 is a runway end of the designated runway in field #2
4. field #4 identifies the operation in the runway from the specified runway end that may block the operations in the other runway
5. field #5 is the airport ICAO code
6. field #6 is the other runway identifier
7. field #7 is a runway end of the other runway
8. field #8 identifies the operation in the runway from the specified runway end that may block operations in the other runway

The field #3 is optional and when it is not specified then Flitan will consider that the operation (field #4) from runway (field #2) without regard to the runway ends will block operation (field #8) on the other runway (field #6) from runway end (field #7).

The field #7 is optional and when it is not specified then Flitan will consider that the operation (field #8) from runway (field #6) without regard to the runway ends will block operation (field #3) on the other runway (field #2) from runway end (field #3).

The runway operations that may block operations on the other runway are as follow:

- ARRIVAL
- DEPARTURE
- MIXED

2.1.4.2.1.6 Flitan simulation of PJ02-08

Once all the input data is read (see the input data section), Flitan will perform the following main steps in order to assess a runway configuration activation plan:

1. Identify and activate the current runway configuration - As the simulation clock advances, Flitan will determine from the runway configuration activation plan which runway configuration to activate at the current simulation time.

2. Select the set of runways based on the active runway configurations - Once the current runway configuration is found, Flitan will select the set of active runways as well as for each selected runway, Flitan will determine the runway operation orientation by selecting the appropriate runway ends

3. Determine the runway configuration capacity - Given the current runway configuration, Flitan will identify the maximum allowed movement per hour for arrivals, departures and global. If a time-based capacity is found, Flitan will use those capacities otherwise it will use the global capacity

4. Assign runway to balance workloads among runways - As flights are planned for either departures or arrivals, Flitan will assign an active runway as well as a runway end to each flight. The runway assignment is done tom preserve a workload balance between active runways.

5. Consider the runway dependency operations - Before a flight is allowed to perform its operation , Flitan will perform a situation awareness to determine whether an operation on a runway is blocking the current operation.

6. Estimate runway time usage and occupancy time - Given the current runway configuration, its capacities and possible runway dependency operations, Flitan will either allow a flight to continue its operation or wait in the runway queue until all the conditions are met. Once a flight is granted the runway usage, the runway is blocked during a selected time. The selected time is mainly based on the flight aircraft model

Once the simulation is completed, Flitan outputs a set of metrics (see the output data section) that together provide a clear assessment of a given runway configuration activation plan. These metrics will be used to train the metamodel with the ultimate goal that the metamodel will be capable to predict and select a combination of runway configurations that meet a given traffic demand without human intervention.

2.1.4.2.1.7 Input Data

In order to model PJ02-08 SESAR solution, Flitan introduces a new set of input files as described in Table 8 below.

2.1.4.2.1.8 Output Data

In support of post simulation analysis of SESAR solution PJ02-08, Flitan generates a new set of log files that together provide a clear assessment of a runway configuration activation plan. Each generated log file contains information about a specific Key Performance Indicator (KPI), see Table 9.

Table 8. Input data.

Input File	Description
Traffic data file	This file includes the trajectory as well as the schedule information for each of the flights intended to be simulated through Flitan. The traffic data file uses the ALL-FT+ in version 5 format
Runway dependency file	The runway dependency file contains information related to runway pairs where their operations need to be coordinated
Runway configuration file	The runway configuration file contains information related to the definition of a given runway configuration at a given airport. This file provides also the runway configuration global capacity. Flitan uses this information to select the active runways, their directions of operations and capacity
Runway configuration time-based capacity	The runway configuration time-based capacity contains runway configuration capacity segregated by time intervals
Runway configuration activation plan	The runway configuration activation plan contains schedule information in regard to the activation of runway configurations

Table 9. Flitan output files.

Output File	Key Performance Indicator (KPI)	Description
Flight runway delay	Efficiency	This file provides the delay incurred by a given runway for each simulated flight
Runway throughput file	Capacity	This file provides the runway hourly throughput
Runway delay file	Capacity	This file provides the total delay for each simulated runway

2.2 Mercury development

In this section we describe the general purpose of Mercury, a simulator which has been developed over several years for specific purposes. We then move on to describe the changes carried out for NOSTROMO in order to model the implementation of Solutions PJ01.01 and PJ07.02.

2.2.1 General description of Mercury

Mercury is a simulator developed over several years which is able to produce detailed network-wide performance assessment, in particular regarding passenger mobility in Europe.

Mercury is implemented as an event-driven simulator. The underlying model can be seen as a Monte Carlo simulation, sampling distributions (delays, missed connections, etc.) based on causal rules representing actual processes of the air transportation system (e.g. if passenger delay is bigger than a given threshold, an airline incurs costs for compensation and assistance to the passenger). It can also be seen as an agent-based model, and this paradigm has been followed closely when it was designed. A series of agent instances sends messages back and forth and reacts to events. Their memory is private, i.e. they have attributes that cannot be accessed by other agents. This opens the way to modelling imperfect knowledge and eases the implementation of rules of thumb and approximated decision-making processes, more realistic than full ‘hyper-rational’ agents in general.

The scope of the simulator is to model individual aircraft throughout one day of operation, including turnaround processes, tracking the passengers on board, as well as passenger connections. The passengers in Mercury are modelled through passenger processes, simulating for instance connecting times for individual passengers, connecting options, etc. The flight is described in terms of times it takes to complete different processes (taxi, take-off, cruise, etc). The simulation of the en-route phase is approximated using actual flight plans and delay distributions. The fuel consumption can be assessed with quite a good precision, thanks to BADA models, but a full trajectory optimiser is not included in this version. Hence, the simulator relies heavily on historical flight data to sample navigation times[2].

Different types of agents are present in the system, sometimes instantiated multiple times (e.g. airline operating centre), sometimes once (e.g. Network Manager). We describe the most important ones succinctly in the following.

2.2.1.1 Airline Operating Centres (AOC)

This agent is the most important in the simulation and the most advanced. It is tasked with following its flights and passengers, making decisions when disruptions hit, providing information to the Network Manager agent if needed, etc.

[2] A trajectory optimiser is in general required when one wants to modify an aircraft trajectory. For instance, modifying the speed of the aircraft in general changes the descent profile. In NOSTROMO we are relying on past data instead (for this second iteration), using similar aircraft that underwent similar changes in order to rebuild trajectories. Note that in the third iteration of the model, another process will be used, based on BADA4 model sampling.

AOC decision-making process is based on usage of detailed cost functions, estimated using [13] and representing the best guess we have on the costs to airlines. The costs include fuel, explicit passenger direct costs (compensation/duty of care), implicit maintenance and crew costs, curfew costs, etc. The cost function of a flight (the cost as a function of the arrival time for instance) is deterministic and features typically various 'jumps', corresponding to passengers missing their next connections or curfew infringement (potentially on the next rotations).

The cost function is used for different processes, like the two mechanisms described thereafter.

2.2.1.2 Flight

The flight agent is tasked with estimating different phases of flights, drawing randomly delays where needed, based on historical distributions. It is mainly communicating with its AOC. For instance, the AOC may provide to the flight its cost function to be able to make informed decision on speed in advanced mechanisms.

2.2.1.3 Ground Airport

The ground airport takes mostly care of two processes. First, it manages the connecting passengers, drawing random values for their connecting times, sampled from a distribution calibrated on past data. This connecting time is compared to the minimum connecting times, available for each airport and type of connection (national-international, international-international, national-national). Second, it does the same for the aircraft, drawing at random values for their turnaround times. This distribution is also calibrated on past data. The departure delay is then applied to the next flight if the turn-around time is too high.

2.2.1.4 DMAN and AMAN

These agents track explicitly the departures and arrivals at their airport and ask for ATFM regulations if the traffic needs to be regulated. Depending on their nature (see new mechanisms thereafter 2.2.3.2), they may communicate to the flights en-route to ask them to change their speed.

2.2.1.5 Network Manager

The Network Manager is responsible for considering flight plan submissions from the airlines and managing ATFM regulations. It checks with DMAN and AMAN if the estimated traffic would surpass the capacity in their respective airports and arranges for flights to be delayed, according to different rules (see mechanisms thereafter). Note that on top of regulations for traffic, regulations can be randomly sampled from historical data, to simulate loss of capacity due to other events, like weather.

2.2.1.6 Other considerations

Passengers are bundled in groups that share common itineraries, and these groups are handled by the other agents, like the AOC and the ground airport. These groups are dynamic, i.e. in case of missed connections they can be split if their respective passengers need to board different planes. Note that in this simulator the passengers do not have to make any decision and are thus not considered as agents.

Finally, we highlight two important limitations of the simulator:

- The crew is not explicitly considered by the model. In other words, the airline does not take into account crew rotations. It only considers an approximated cost for the airline to operate with delay, which takes into account the average cost of dealing with disrupted crew processes. This is due to the difficulty to get historical crew rosters from airlines.

- Apart from the runways themselves, there is no explicit model of the airspace. In other words:
  - the model does not track through which sectors a flight is going, and thus cannot issue regulations based on traffic on these pieces of airspace. En-route ATFM regulations are thus simulated as random delays.
  - no tactical traffic control takes place. We do not model controllers, and the trajectory is not modified based on their potential actions. Instead, we used distribution of "delays", which modify the flight times between navigation points, extracted from historical data.

As highlighted above, the simulator relies heavily on historical data, that it uses in particular to create and then sample various distributions. The results in this deliverable have been obtained with a calibration of the model on the 12th of September 2014, for which we have all the required data, and the data sources are shown in Table 10.

Table 10. Data used by Mercury.

Data source	Main usage	Reference
DDR2	Used to get the set of flights, origin-destination, routes, aircraft type, estimated cruise wind, distributions on climb and descent profiles, requested nominal cruise speeds and flight levels, companies, alliances, airspace structure, ATFM regulations	[3]
Cost of delay report	Used to compute cost of delay function	[4]
IATA Summer Season 2010 from CODA	taxi times	[5]
DDR2	minimum turnaround times, minimum connecting times	[6]
CODA	non-ATFM delays	[7]
Paxis, GDS	For passenger itineraries, including fares and class	[6]
Innovata (Cirium)	Flight schedules, for scheduled times of arrival and departure	-

For the third iteration of the model (to be presented in D5.2), the project will use the same kind of data, updated to match the 14th of September 2018, thus provided a more up to date estimation of the KPIs.

More details on the Mercury simulator can be found in [8].

2.2.2 PJ01.01 - Flight Arrival Coordination

2.2.2.1 The role of the Arrival Manager

As explained in D4.1 [1], PJ01.01 was selected as a case study for Mercury. Indeed, the simulator is well suited to model decisions made by airlines, thanks to its detailed cost model. This case study has also been selected to highlight the potential importance of systemic effects, such as network knock-on effects, triggered by delays incurred during the arrival phase.

Flight arrival coordination is an important present and future ATM feature both for airlines and the corresponding control systems. Indeed, arrival flights need to be arranged into a sequence at arrival by the arrival manager, making sure that safety is enforced, but also that the runway throughput is as high as possible given its capacity. Indeed, contrary to en-route control, where the focus is almost only on separating aircraft, i.e. safety, arrival at the airports also includes optimisation considerations. Since runway capacity cannot be infringed, any traffic overload translates into delay, usually materialised by a holding stack, whereby flights wait for a slot to land. Hence, the arrival manager needs to ensure safety while decreasing delays/optimising runway throughput.

Arrival managers use various strategies, but all consider a certain horizon, in time or space, where they start building up a queue of flights in arrival. An important parameter of this process is the size of this horizon, because getting information early means that the AMAN can build its queue more efficiently. In practice, uncertainties may be so big that increasing the horizon may not be useful, and, more importantly, the AMAN cannot give arbitrary commands (for instance a slowdown command) to a flight, in particular because this command may not be consistent with other commands from controllers in intermediate sectors. Equally important, for the airline changing the speed has a cost. Indeed, by increasing their speed, they will burn more fuel, and by decreasing it, they may have passenger connection issues, for example (cost of delay).

In summary, the future of arrival coordination must include the following considerations:

- An extended horizon, in order to build an efficient queue at arrival.
- Airspace user preferences (e.g. related to their costs). This is related to business trajectories, where airlines are part of the decision-making process leading to their final trajectory.
- Other sectors' overloading. This is related to 4D trajectories, with consistent constraints throughout the airspace leading to a full specification of the trajectory, as opposed, currently, to mostly independent safety control processes.
- Uncertainty in the 4D trajectory, directly linked to the (maximum) efficiency of the queue optimisation.

The research questions that Mercury – and the meta-modelling process – could tackle are related to these considerations. In NOSTROMO, we decided to focus on the extension of the horizon, the airline preferences, and the uncertainty. The study of the en-route sector overload is less adapted to Mercury, given its lack of modelling precision in the airspace.

In NOSTROMO, we extended the capabilities of Mercury by implementing an advanced form of AMAN that can monitor flights in a larger area, give commands earlier than in reality, and taking into account the airline cost model.

The following section describes the implementation in Mercury, as simulated for the second iteration. They only use delay minimisation, but in a smarter way for the advanced implementation, taking into account reactionary delay (one of the most important elements of the cost of delay).

2.2.2.2 Description of the Extended Arrival Manager

The E-AMAN manages an explicit queue of slots for the airport. The algorithm optimises the arrival sequence between the planning and the tactical horizon considering a particular objective function.

The final sequencing (from tactical horizon to runway) is out of scope of the optimisation.

Optimisation inside E-AMAN

When a flight enters the planning horizon, all flights which are located in the scope of the arrival manager, i.e., between the planning and the execution horizon, are re-optimised, i.e., assigned to the slots which are either planned or available, considering a given optimisation function which depends on the level of the mechanism.

The optimisation of all the flights within the E-AMAN every time a flight enters or exits the system ensures that the best sequence is maintained within the arrival manager with respect to the optimisation function, and that a flight may slow down to absorb part of the delay saving some fuel if delay is assigned by the E-AMAN. However, this re-optimisation does not consider an update on the position and expected delays/costs of the flights already inside the E-AMAN process, i.e., reuses the cost function for those flights as provided when entering the E-AMAN.

As the amount of delay that can be absorbed (by slowing down) is very limited, only the flight which enters the arrival manager considers this speed and TOD variation. Note that only available or planned slots are considered in the optimisation and once a landing slot has been assigned to a flight which reaches the tactical horizon it is fixed. Note that in some cases, the slot which has been planned at the planning horizon for a given flight might not be still available when it reaches the execution horizon. Finally, at both horizons, the arrival capacity at the airport is considered to ensure that the arrival sequence respects airport throughput.

Use of delay by flights

At the planning horizon, a flight which triggers this optimisation, i.e., which enters the arrival manager, receives the amount of delay that it is expected to experience and tries to absorb as much delay as possible by slowing down (saving some fuel). At the tactical horizon, a flight will be issued with a slot (assigned as the output of another re-optimisation) and the required delay (if any) will be performed as holding.

Airports without E-AMAN

For airports which are considered to have an E-AMAN, a simple arrival manager located at 100 NM from the airport is considered, and a first-in first-out approach modelled. The assigned delay is done as holding. This ensures that the arrival capacity at the airport is not exceeded.

Airports with E-AMAN

In this case the focus is on minimising the total expected delay that the flight will experience at the airport (arrival and departure). This is done considering arrival and expected reactionary delay, which act as a better proxy for the cost of the airline.

When a flight enters the planning horizon, the information on the expected delay that the flight will have (both arrival and reactionary) per slot available (i.e., not assigned yet to a flight in the tactical horizon) is communicated to the E-AMAN.

The E-AMAN uses this information to optimise the sequence aiming at minimising the total delay.

2.2.3 PJ07.02 - Airspace user prioritisation for hotspots

2.2.3.1 Hotspot solving

‘Hotspots’ are areas of the air traffic network that are (likely to be) stressed due to high levels of traffic. Indeed, based on the traffic forecast from flight plan submissions, the area control centres and the Network Manager monitor whether the traffic will be over capacity for any portion of airspace, at specific times. If this is the case, the Network Manager can issue an ATFM regulation, i.e. a restriction on the number of flights that can enter a given volume of airspace for a given time. The flights that were supposed to cross this zone during the regulation duration are then explicitly assigned ATFM slots. A departure delay is issued to the flight based on this slot (ATFM delay). Note that airlines may also resubmit a new flight plan in order to avoid the regulated zone (although this may not be very effective).

The way flights are currently assigned to ATFM slots is using the so-called “First-Planned First served” CASA algorithm (FPFS) [9]. CASA assigns each flight to the first available slot, starting from the earliest one. This algorithm is used because it is considered as fair among flights and airlines, but also because it minimises the total delay assigned to flights in the regulation.

However, the FPFS rule fails to take into account an important dimension for airlines: cost. Indeed, flights have usually very different importance for an airline. Some, for instance, carry many connecting passengers, which may miss their next flight. On the contrary, some flights may carry a small number of non-connecting passengers, or do not have another rotation before the following day. As a consequence, delays will have very different effects on them in terms of costs, which is crucial to airlines.

Various rules and mechanisms have been suggested to improve the situation, but they all come down to flight swapping, i.e. the ability to swap flights present in the regulation queue. This is the focus of the study in PJ07.02, the User-Driven Prioritisation Process (now solution #57, ready for industrialisation).

A first improvement over the FPFS rule is to let the airlines ***swap their own slots***, i.e move flights back and forth in their slots, without interacting with other airlines. This can be done through different automated processes, such as Flight Departure Reordering (FDR) , Selective Flight Protection(SFP), and ‘Margins’, etc., which aim at helping the airline to make the best decision. Among these processes, some of them are still in the deployment phase (Margins), others are active only at some airports (FDR) while others are more widely in operation (SFP).

Whilst swapping is efficient in some cases, it lacks the capacity to find good solutions when airlines have a small number of flights in the regulation (the issue of so-called low-volume users). For this reason, different projects are exploring the possibility to trade or at least exchange slots across airlines. In ER-4, the projects BEACON and SlotMachine are studying such mechanisms. We decided to use some concepts developed in BEACON – of which two of the consortium members are part – to test the meta-modelling process. See BEACON deliverable D3.1 for more details about the mechanisms [10].

The goal in NOSTROMO is to assess the efficiency of these potential future mechanisms – once again more efficiently thanks to the meta-modelling process. We restricted ourselves to regulations hitting arrival airports, for two reasons: a) to avoid having to model the behaviour of airlines changing their flight plan to avoid en-route regulations and b) because it is the initial scope of UDPP, and thus allows an easier comparison with the previous state.

2.2.3.2 Mechanisms tested in the second iteration of NOSTROMO

The mechanisms we considered in this second iteration of the model are shown in Table 11.

Table 11. Mechanisms tested.

Mechanism	**Main principles**
FPFS	Serves as baseline to compare the other mechanisms; current mechanism in use, minimises the total delay assigned to flights in a regulation
UDPP	Optimal allocation intra-airlines. It represents the efficiency in the current situation, if all airlines use the UDPP tools in an efficient way.
ISTOP	Developed in BEACON, it uses information akin to what is passed to UDPP to rebuild an approximated normalised cost function and suggests suitable slot swaps among airlines.
NNBound	Uses the true cost functions of the airlines, it finds the best allocation, given that no airline loses in terms of cost. This is used for benchmarking, since it represents the maximum efficiency if local “fairness” is enforced (nobody loses).
GlobalOptimum	Uses the true cost functions from the airlines, it finds the best allocation overall. This is used for benchmarking, since it represents the maximum theoretical efficiency reachable by any mechanism.

The important mechanism that we want to test in this second iteration of the model is ISTOP. In this mechanism, the Network Manager asks every airline in the regulation to provide two parameters called “margin” and “jump” (note that names are not definitive and come from the current state of the ER-4 BEACON project). Notionally, one can think of this parameter as the location and the height, respectively, of the first jump in the flight’s cost function (see Figure 11). This first jump is typically well-known to the operator, and represents an important piece of information on the flight. They are also related to parameters provided by airlines in some UDPP mechanisms: the ‘margin’ is almost the same as the ‘time not after’ present in the Margin mechanism, and the ‘jump’ is closely related to the priority of the flight, as featured in the UDPP prioritisation mechanism. These are thus quite intuitive and already known, to some extent, by operators using the present-day or near-future UDPP capabilities. Moreover, note that the jumps provided do not need to be given in absolute terms, but relative to each other, which is far easier (and, once again, akin to priorities).

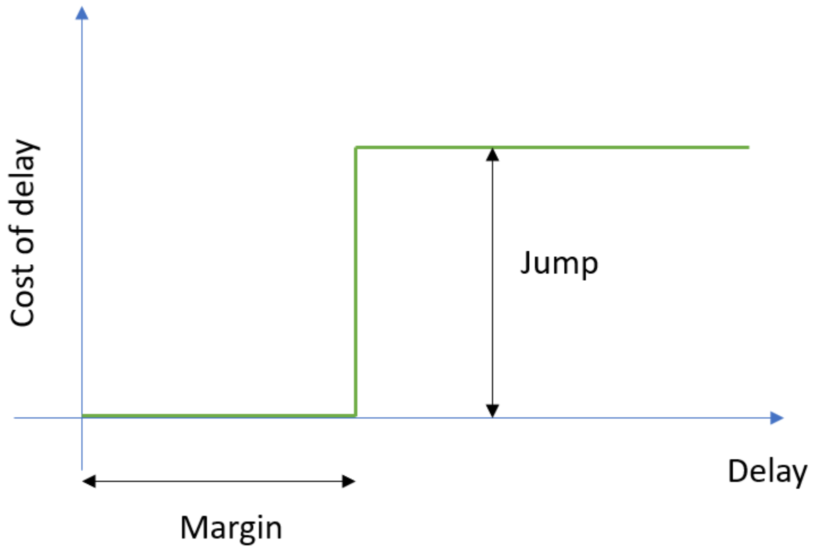


Figure 11: Approximated cost function used by the UDPP mechanisms, with the two parameters, margin and jump.

The reason for that lies in the next step of the algorithm, where the Network Manager rebuilds approximated, normalised cost functions for each flight in the regulation. This cost functions are then used to find suitable swaps between flights, with pairs (or triples) of slots being exchanged between airlines, resulting in a smaller cost for both airlines. In theory, these swaps are then suggested to the airlines, who may or may not accept them. In NOSTROMO, for simplicity, we assumed that the airlines always accept the swaps suggested.

2.2.4 Simulations

As mentioned previously, we use the 12th of September 2014 to calibrate the model. In order to have faster simulation time, we also decided to reduce the geographical scope of the model and consider only the flights to and from Charles De Gaulle airport – with all corresponding information on passenger connections and turnaround. Another reason for this selection is to have decent statistics when it comes to ATFM regulations. Indeed, regulations are quite rare in a normal day of operation, or at least not big enough so that advanced UDPP mechanisms could be used. For the same reason, we considered a scenario where exogenous delays have been artificially inflated, in order to stress the system and have a better assessment of the mechanisms.

In order to answer the research questions and find interesting relationship between variables – harder to model and thus more interesting from the metamodelling point of view – we selected the input variables shown in Table 12 to be studied by the metamodelling process. The table includes the possible values, practical range, and default values of the parameters, all used during the metamodelling process.

Table 12. Mercury input variables.

Variable	Name in model	Description	Theoretical range	Practical range	Default
Fuel price	fuel_price	Price of one kg of fuel - Continuous	[0, infty)	[0, 5]	1
Hotspot solver	hotspot_solver	Type of solver in the hotspot	['globaloptimum', 'nnbound', 'udpp_merge', udpp_merge_istop']	NA	'udpp_merge_istop'
Planning horizon	eaman_planning_horizon	Distance horizon where the EAMA tries to optimize the arrival, in NM.	(100, infty)	[100, 1000]	200
Cruise uncertainty	cruise_uncertainty_sigma	Deviation in the aircraft speed during cruise	[0, infty)	[0, 10]	1
Turn-around time scale	alpha_tat_mean	Scaler of mean of the distribution of turn-around times	[0, infty)	[0, 10]	1
Minimum connecting time scale	alpha_mct	Scaler of mean of the distribution of passenger minimum connecting times	[0, infty)	[0, 10]	1
Claim rate	claim_rate	Proportion of passengers claiming compensation	[0, 1]	[0, 1]	0.14

Finally, we decided to select the performance indicators included in the Table 13 as output variables to be learnt by the metamodel. They represent the most interesting variables that can be extracted from the model related to the two mechanisms described above. The table also shows a priority number that could be used by the metamodellers to incrementally improve their model.

Co-funded by the European Union

Table 13.. Mercury output variables.

Variable	Description	Name in model	Priority
Holding time	Holding time incurred by the flight at arrival	m3_holding_time	2
Assigned delay at planning horizon	Delay assigned the E-AMAN at the first horizon of command.	eaman_planned_assigned_delay	3
Planned absorbed delay	Delay can the E-AMAN planned to be absorbed by the flight (via speed change)	eaman_planned_absorbed_air	2
Extra tactical delay	Actual delay incurred by the flight	eaman_extra_arrival_delay	1
Assigned delay at tactical horizon	Delay assigned by the E-AMAN at the second horizon	eaman_tactical_assigned_delay	3
Diff in tactical and assigned delay	Difference between the actual delay and the assigned delay	eaman_diff_tact_planned_delay_assigned	1
Saved REAL cost in regulation w.r.t. FPFS allocation	Ratio of the cost incurred by the airlines with a given mechanism after allocation with respect to cost incurred in FPFS	ratio_cost	1
Saved DECLARED cost in regulation w.r.t. FPFS allocation	Ratio of the cost incurred by the airlines, as declared through their approximated cost functions, with a given mechanism after allocation with respect to cost incurred in FPFS	ratio_cost_approx	1

Co-funded by the European Union

3 Metamodelling process

Simulation constitutes a well-known and established tool to model complex real-world systems. However, despite its clear practical advantages, simulation models, when embedded with enough detail and realism, can become computationally expensive to run. This shortcoming may pose a hindrance to the exploration of their input-output behaviour. ATM research, particularly the sub-field that deals with the simulation of air traffic systems, is no stranger to such concerns.

To tackle the above-mentioned computational challenges, simulation metamodels [10], [11], [12] can be employed to approximate the underlying function inherently defined by the simulation model and used as a modelling proxy for the latter. In turn, this allows for a reasonable number of exhausting computer experiments to be bypassed during the exploration process.

Additionally, the problem of expensive simulation runs has a clear resemblance with modelling scenarios where labelled data tends to be particularly difficult or time-consuming to acquire. In such situations, active learning [13], [14] emerges as a powerful learning paradigm aiming at attaining high prediction performance with as few data points as possible. Remember that labelled data point corresponds to a single tuple encompassing both input values and output results. In our specific context, a label is nothing more than an output simulation result.

In this project, we adopt an integrated approach that employs active learning strategies on top of simulation metamodelling processes aiming at reducing the computational burden often associated with exhaustive and systematic simulation analysis. Eventually, the ultimate goal is to provide ATM researchers and practitioners with an auxiliary tool to explore the output behaviour of simulation models in a more insightful and computationally efficient manner. For details on this methodology, as well as on its elementary constituents, please refer to the deliverable D3.1 [15].

Figure 12 depicts an overview of the metamodelling process underlying the results presented herein.

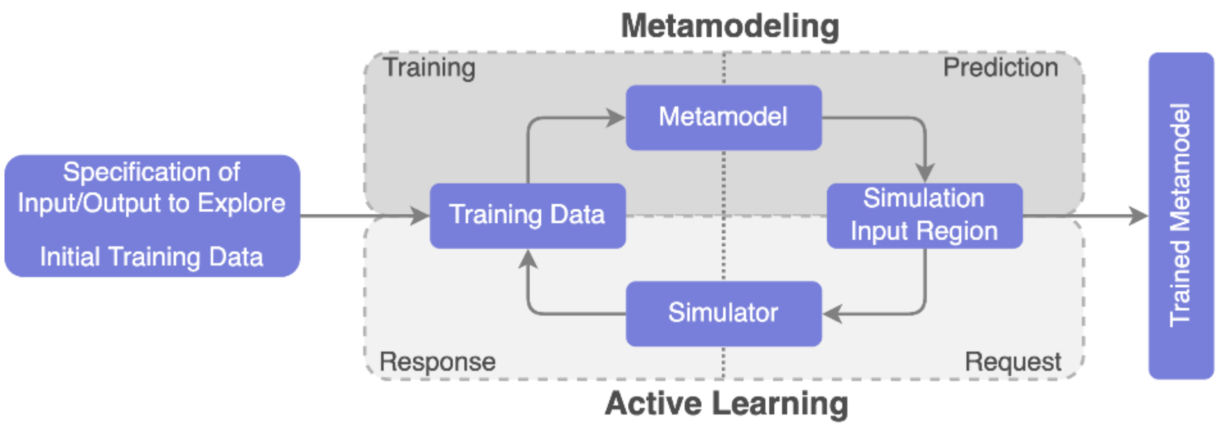


Figure 12. Active learning-based metamodelling process overview.

This process has at least two starting requirements, namely, the specification of the input-output variables necessary to study a particular solution, and, due to its iterative nature, an initial training data set. The latter comprises a small set of simulation results generated according to the specified simulation variables of interest and a certain sampling strategy such as the Latin hypercube sampling [16]. Additionally, the metamodel itself should be chosen a priori. While the Gaussian Processes (GP) regression framework, also known as Kriging, is the traditional and still common choice for metamodelling, other machine learning approaches can be tested, such as Neural Networks (NN). In fact, in the document, we present initial results using both modelling approaches.

Alternating between the metamodeling and the active learning phases, the integrated methodology is composed of four elementary steps:

1. **Training**: the metamodel is fitted to the simulation data.
2. **Prediction**: the fitted metamodel is used to predict over the simulation input region of interest.
3. **Reques**t: based on some acquisition criteria, new input data points are selected to be run by the simulator.
4. **Response**: the simulator provides new simulation output results corresponding to the points from step 3, which are then added to the current training set.

Steps 1-4 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 predictive variance reduction, or simply the number of iterations to be performed with respect to the available time, budget and resources.

Remember the approximative nature of the metamodel which calls for careful handling of the trade-off between speed, accuracy and computational budget. It is important to recognize and identify the performance threshold from which the mere addition of new training points will not significantly improve the ability of the metamodel to approximate the simulation results. In those cases, and especially from a metamodelling perspective, requesting more simulation results might prove to be a waste of computational resources.

4 Metamodelling results

In this section, we briefly present part of ongoing metamodelling results with respect to solutions PJ.08-01 (Flitan) as well as PJ01.01 and PJ07.02 (Mercury). With respect to Flitan, due to unforeseen data collection difficulties, we only briefly mention solution PJ02-08, referring its results to the upcoming deliverable during the project's 3rd iteration.

Due to the nature of Flitan's categorical input data types, extra steps of data encoding and preparation before metamodelling had to be undertaken. As most statistical and machine learning-based metamodels cannot operate directly with categorical values, in both input and output, appropriate transformation is applied to the original data so that it can be converted to some kind of numeric form. Here we also detail how we conducted such data conversions. Hence, this is a highly tailored encoding solution to handle Flitan's categorical data.

Within the scope of this project, the procedures involved in the kind of steps mentioned above constitute part of what we have deemed as the "translation layer" [17] responsible for converting the simulator's data types and structures into those of the metamodel itself, ensuring a smooth communication link between both. Although subtle and typically not accessible to the end-user, this layer constitutes a fundamental part of the metamodelling process. Its importance is further extended if we take into account the fact that many ATM simulation models use, in fact, categorical variables. Therefore, we should highlight the development of such translation layers as a co-product of the metamodelling process and results.

On the other hand, no real data preparation is required with respect to Mercury as most of it is essentially comprised of numerical values that can be fed directly into the metamodel.

4.1 Flitan

In this section, we detail the translation layer for Flitan's categorical variables and present some results from ongoing experiments, focusing on solution PJ.08-01.

The experiments reported herein used a version of Flitan loaded with real daily traffic data dated from 21/06/2019 to 23/06/2019 and for North Madrid Air Traffic Control (LECMCTAN). The DAC analysis itself was limited to 22/06/2019, although it trivially encompasses the flights that departed on 21/06 and arrived at LECMCTAN on 22/06 and, similarly, that departed on the latter, arriving only at their destination on 23/06. For this single day, a metamodel was trained and used to virtually explore around 4 million opening schemes.

4.1.1 Data encoding

As oftentimes occurs with other ATM simulation models implementations, totally or partially, Flitan does not follow the required specifications of the data format specified in [2]. Due to the categorical (string-based) nature of some of Flitan's input data, data conversion schemes to numerical types are required for the type of metamodels being used in this project. Consequently, an extra step for data preparation and encoding is necessary prior to the metamodelling process itself.

PJ.08-01 Solution

This solution regards the implementation of Dynamic Airspace Configuration (DAC). The main objective of DAC is to allow the adjusting of capacity to meet available Air Traffic Control (ATC) resources, changes in traffic demand and various performance indicators. DAC itself involves several fundamental concepts, namely (for details see section 2.1.3):

- elementary and configured sectors,
- sector configuration,
- opening scheme (schedule of sector configurations),
- capacity,
- demand,
- and demand capacity balancing (DCB),

which can be generally regarded as inputs for this particular DAC implementation. In terms of outputs, Flitan provides performance metrics such as the average number of open sectors, configuration and flight delays, and sector loads.

In this document, we focus on the opening schemes as input data and the configuration delays as the output performance metric. For a single operation day and fixed traffic demand, the idea was to establish a relationship between the opening scheme and configuration delay via metamodel, and to eventually be able to predict the latter and to allow a more efficient exploration of the simulator's behaviour while minimizing the corresponding computational burden.

Table 14 depicts an example of an opening scheme as it is loaded into Flitan. As stated earlier, an opening scheme is a schedule of sector configurations. These configurations are then referenced by their code name. From this data, we can obtain which configuration was active and when. For example, configuration "CNF8B1" was active from 05h30m to 12h59m on 22/06/2019.

Table 14. Format of an opening scheme for a single day

Date	Unit of Control	Start Time	End Time	Configuration Name
22/06/2019	LECMCTAN	00:00	03:29	CNF1A
22/06/2019	LECMCTAN	03:30	04:59	CNF5A
22/06/2019	LECMCTAN	05:00	05:29	CNF5A
22/06/2019	LECMCTAN	05:30	12:59	CNF8B1
22/06/2019	LECMCTAN	13:00	20:29	CNF8B1
22/06/2019	LECMCTAN	20:30	21:29	CNF8B1
22/06/2019	LECMCTAN	21:30	23:59	CNF5A
22/06/2019	LECMCTAS	00:00	03:29	CNF1A
22/06/2019	LECMCTAS	03:30	04:59	CNF5A2
22/06/2019	LECMCTAS	05:00	05:29	CNF5A1

Co-funded by the European Union

Date	Unit of Control	Start Time	End Time	Configuration Name
22/06/2019	LECMCTAS	05:30	08:59	CNF8A2
22/06/2019	LECMCTAS	09:00	11:29	CNF8A1
22/06/2019	LECMCTAS	11:30	20:29	CNF8A2
22/06/2019	LECMCTAS	20:30	21:29	CNF6D2
22/06/2019	LECMCTAS	21:30	22:59	CNF3C
22/06/2019	LECMCTAS	23:00	23:59	CNF3C

Each sector configuration can be broken down into a particular list of configured sectors, each being further associated with a specific combination of elementary sectors. This information is stored in the airspace information log, mostly as categorical data. A partial example of this file is given in Table 15.

Table 15. Format of an airspace information log file.

Unit of Control	Configuration Name	Number of Configured Sectors	Configured Sectors	Elementary Sectors	Capacity (20-minute period)
LECMCTAN	CNF1A	1	LECMR1I	LECMASL +LECMASU +LECMBLL +LECMBLU +LECMDGL + +LECMDGU + +LECMPAL +LECMPAU +LECMSAO+LECMSLE	6
LECMCTAN	CNF5A	5	LECMASI LECMBLI LECMDGI LECMPAI LECMSAN	LECMASL+LECMASU LECMBLL+LECMBLU LECMDGL+LECMDGU LECMPAL+LECMPAU LECMSAO+LECMSLE	12 15 13 12 11
LECMCTAN	CNF8B1	8	LECMASL LECMASU LECMBLI LECMDGL LECMDGU LECMPAL LECMPAU LECMSAN	LECMASL LECMASU LECMBLL+LECMBLU LECMDGL LECMDGU LECMPAL LECMPAU LECMSAO+LECMSLE	12 12 15 13 15 12 15 11

Column "Elementary Sectors" refers to the combination of elementary sectors that compose each of the sectors under "Configured Sectors". For example, the configuration with name "CNF5A" corresponds to the following elementary sector configuration: "LECMASL+LECMASU" (LECMASI), "LECMBLL+LECMBLU“ (LECMBLI), "LECMDGL+LECMDGU“ (LECMDGI), "LECMPAL+LECMPAU" (LECMPAI), and "LECMSAO+LECMSLE“ (LECMSAN). Each sum ("+") means that the corresponding involved elementary sectors are combined into a single sector.

Given a specific daily traffic demand, the performance of each configured sector can be assessed via flight delays, among other metrics, as mentioned above. Table 16 provides an example of a configuration delay file providing the total flight delay per configuration.

Table 16. Format of a configuration delay file.

Unit of Control	Configuration Name	Configuration Delay (in seconds)
LECMCTAN	CNF8A	8629
LECMCTAN	CNF5A	480
LECMCTAN	CNF1A	2020
LECMCTAN	CNF3C	2335
LECMCTAN	CNF4B	6245
LECMCTAN	CNF9A2	9121
LECMCTAN	CNF9A1	1901

In order to infer a functional relationship via simulation metamodelling, the contents summarily represented in previous three tables must be properly prepared, parsed, merged and finally converted to quantitative values so that they can be finally handled by the metamodel itself. To this end, we made use of dummy variables to encode the indication of the activation status of each elementary section: “0” for inactive, and “1” otherwise. Hence, a conversion layer (encoding) was generated univocally linking sector configurations, elementary sectors, and dummy variables. Table 17 on next page provides a partial overview of this mapping.

Finally, we are in conditions to convert Flitan’s original data to numeric-based values. Note, however, that this corresponds only to one solution which was highly tailored to the specific problem at hand and the simulator’s characteristics. Other encoding solutions could also have been adopted and explored. Table 18 provides an example of the final data format to which the metamodel is fitted.

Table 17. Data encoding from combined elementary sectors names to dummy variables.

Unit of Control	Configuration Name	Elementary Sectors	LECMASL	LECMASL + LECMASU	LECMASL+ LECMASU+ LECMBLL+ LECMBLU+ LECMDGL+ LECMDGU+ LECMPAL+ LECMPAU+ LECMSAO+ LECMSLE	LECMASL + LECMASU + LECMBLL+ LECMBLU + LECMSAO +LECMSLE	…	LECMDGL+ LECMDGU	LECMDGL+ LECMDGU+ LECMPAL+ LECMPAU
LECMCTAN	CNF1A	LECMASL+LECMASU+LECMBLL+LECMBLU+LECMDGL+LECMDGU+LECMPAL+LECMPAU+LECMSAO+LECMSLE	0	0	1	0	…	0	0
LECMCTAN	CNF2A	LECMDGL+LECMDGU+LECMPAL+LECMPAU LECMASL+LECMASU+LECMBLL+LECMBLU+LECMSAO+LECMSLE	0	0	0	1	…	0	1
LECMCTAN	CNF7G1	LECMASL LECMASU LECMBLL LECMBLU LECMDGL+LECMDGU LECMPAL+LECMPAU LECMSAO+LECMSLE	1	0	0	0	…	1	0
…	…	…	…	…	…	…	…	…	…

Table 18. An example of the explored metamodel data format (not exhaustive due to space restrictions, only the most relevant columns are displayed).

Date	Config. Name	Start Time	End Time	LECMASL	LECMASL+ LECMASU	LECMASL+LECMASU +LECMBLL+LECMBLU +LECMDGL+LECMDGU+ LECMPAL+LECMPAU+ LECMSAO+LECMSLE	…	Configuration Delay
22/06/2019	CNF1A	00:00	03:29	0.0	0.0	1.0	…	2020.0
22/06/2019	CNF5A	03:30	04:59	0.0	1.0	0.0	…	60.0
…	…	…	…	…	…	…	…	…

Co-funded by the European Union

PJ02-08 Solution

In implementing SESAR solution PJ02-08, Flitan targets three specific objectives:

1. Introduce the runway configuration manager and runway dependent operations functions into the Flitan simulation engine
2. Enhance Flitan sequencing function
3. Train the metamodel through the assessment of a large set of possible combinations of runway configurations

By training metamodel with enough simulation data points via active learning, the ultimate goal is that for a given traffic demand, the metamodel will be able to select and provide a ranking of runway configurations without human intervention. The experiment is essentially identical to solution PJ08-01, except that sector configurations are replaced by runway configurations. Therefore, due to the categorical nature of this input, another specific translation layer will be necessary for PJ02-08.

In order to achieve this goal, a large collection of runway configurations of a given airport are required. At this stage, we are still looking to find suitable runway configurations data and their associated activation plan that will allow us to carry out the simulations for a set of European airports.

4.1.2 Results

PJ.08-01 Solution

One of the objectives of employing a metamodel within the implementation of this solution is to explore a multitude of opening schemes and assess their corresponding output performances by bypassing the burden of systematic and exhausting simulation runs in an approximative manner. Given a rather small set of training observations, the metamodel should be able to generalise and explore the remaining input space, predicting the associated output performance values with reasonable accuracy and controlled prediction error.

Currently, the generation of opening schemes over a single day is based on random combinations of pre-defined time windows and sector configurations selected by observation of historical data as follows:

```
opening_scheme_lecmctan = [
{'period': ['00:00', '03:29'], 'config': ['CNF1A']},
{'period': ['03:30', '04:59'], 'config': ['CNF4A', 'CNF5A', 'CNF5B']},
{'period': ['05:00', '05:29'], 'config': ['CNF5A', 'CNF6A', 'CNF6B', 'CNF6C', 'CNF7A']},
{'period': ['05:30', '08:59'], 'config': ['CNF8A', 'CNF8B1', 'CNF8B2', 'CNF9A1', 'CNF9A2']},
{'period': ['09:00', '10:59'], 'config': ['CNF8A', 'CNF9A1', 'CNF9A2']},
{'period': ['11:00', '12:59'], 'config': ['CNF8A', 'CNF8B2', 'CNF9A1', 'CNF9A2']},
{'period': ['13:00', '14:59'], 'config': ['CNF7A', 'CNF8A', 'CNF8B2', 'CNF9A2']},
{'period': ['15:00', '17:19'], 'config': ['CNF7A', 'CNF8A', 'CNF8B2', 'CNF9A2']},
 {'period': ['17:20', '20:29'], 'config': ['CNF7A', 'CNF8A', 'CNF8B2', 'CNF9A2']},
 {'period': ['20:30', '21:29'], 'config': ['CNF7A', 'CNF8A', 'CNF8B1', 'CNF8B2', 'CNF9A1', 'CNF9A2']},
 {'period': ['21:30', '22:29'], 'config': ['CNF4A', 'CNF4B', 'CNF4C', 'CNF5A']},
 {'period': ['22:30', '23:59'], 'config': ['CNF3B', 'CNF3C', 'CNF3D']}
]
```

From here, we can observe that there are more than 4 million (1 x *3* x 5 x *5* x 3 x *4* x 4 x *4* x 4 x *6* x 4 x 3 = 4,147,200) possible different opening schemes. Running all these combinations via actual simulation runs would be clearly unfeasible: simply considering 1 minute per single simulation run, testing all these opening schemes would take more than eight years.

A Neural Network (NN) was used as a metamodel and trained on a data set comprised of 2500 unique opening schemes, followed by the prediction of the configuration delay over the set of all 4 million possibilities. Figure 13 depicts a histogram of the resulting predicted delays.

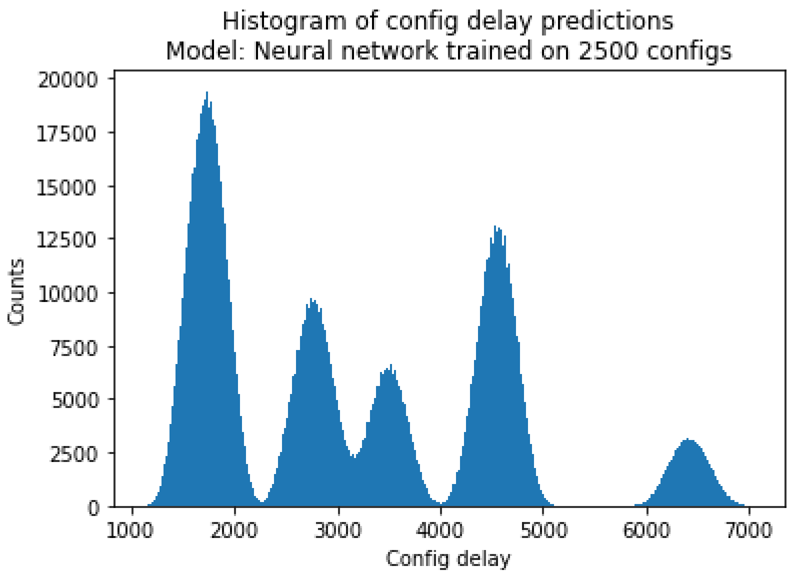


Figure 13. Histogram of the predicted configuration delays over approximately 4 million different opening schemes.

It is interesting to observe that the predicted delays appear to be clustered into five different Gaussian distributions. This allows us to classify each opening scheme into a specific delay profile, thereby simplifying the exploration process of the simulation output performance behaviour. For a given daily demand, this kind of metamodelling exploration may therefore help the ATM practitioner to find the most suitable opening scheme or set of opening schemes with respect to a particular system performance metric, without the burden of conducting all the different associated simulation runs. In this particular case, one may trivially aim to find the sector configurations that lead to the lowest values of flight delays.

Remember, however, the approximative nature of the obtained metamodels and the relative trade-off between computational speed and the ability to closely reproduce the real simulation results. Ultimately, our proposal is to always use the metamodels bundled together with the simulators themselves as an integrated modelling and exploration tool.

In the following set of experiments, we employed a GP to develop an integrated simulation active learning metamodelling strategy. Due to their nice capabilities for handling both data and predictive uncertainty, GPs are commonly used within active learning settings in which there is a direct cost (e.g. computational, such as in our case) of obtaining new labelled data. In such scenarios, modelling is often conducted with relatively small datasets and in an iterative manner.

As a Bayesian approach, the GPs generate predictions in form of well-defined Gaussian probability distributions, instead of single-point ones. Typically, each prediction comes associated with a pair of parameters, often called the predictive mean and the predictive variance, which unambiguously defines the latter mentioned distributions. Here, the variance itself can be viewed as a proxy for the modelling or prediction uncertainty, in the sense that high predictive variance regions potentially enclose more data variability and information (entropy). Therefore, it is generally wise to sample new simulation points from regions that potentially lead to more learning gains, while redundant simulation runs are avoided, resulting in a more computationally efficient and economical exploratory process.

In Figure 14, we present the evolution of Root Mean Square Error (RMSE) as the active learning process advances. In each iteration, the RMSE is computed by comparing the current metamodels' predictions against the real simulation output observations contained in a fixed data set generated a priori. From a computational perspective, the faster we reduce RMSE, i.e., with fewer simulation runs, the better.

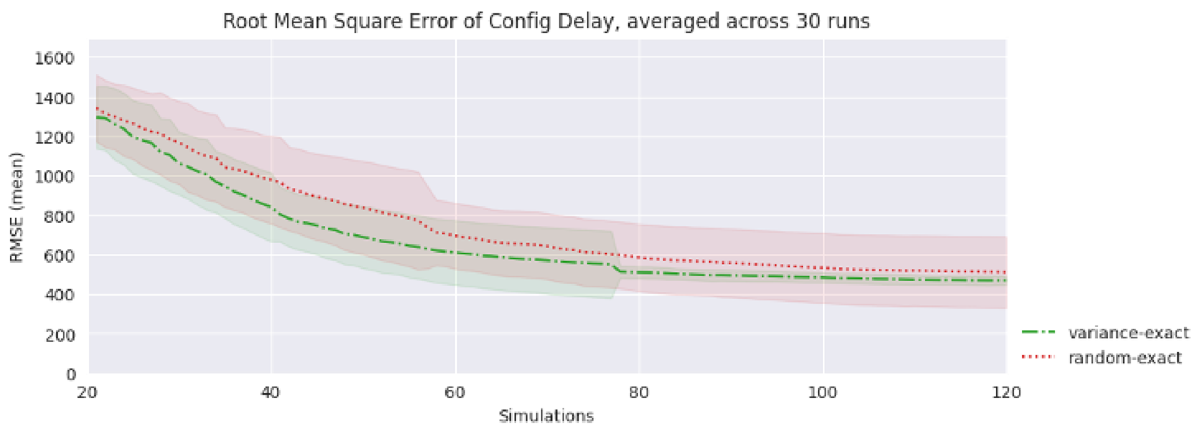


Figure 14. Active learning metamodelling performance for Flitan with DAC implementation. Each line corresponds to the average value across 30 independent algorithm runs and the shades to +/- standard deviation.

This active learning strategy is based on the sampling of high variance points and it is compared with a naive baseline with no query criterion (random selection). This criterion uses the predictive variance computed over the search input space or, in other words, the unexplored simulation region of interest, selecting the points yielding higher values to be posteriorly labelled by the simulator and eventually added to the expanding training set. From Figure 14, we see that the tested strategy outperforms, on average, the random selection of data points for the entire experimental setting of 120 simulation runs. Moreover, observe that the variability across experiments is consistently lower when active learning is employed, particularly after approximately 80 simulation requests.

In the limit, and in the absence of limited computational budgets, it is not difficult to conclude that all active learning strategies eventually lead the metamodels to converge to similar prediction performances or to reach their modelling plateau. The latter trivially holds even for the random querying. However, finite computational resources, amongst other limitations that constrain the timely exploration of a simulation model's output behaviour, call for the adoption of more efficient processes capable of extracting, to the greatest possible extent, more learning information with less computing demand.

PJ02-08 Solution

As mentioned earlier, we are still looking for historical data sets that can fit both Flitan's implementation of traffic optimisation on single/multiple runway airports and NOSTROMO's objectives. For this reason, we are unable to deliver any metamodelling results at the moment and refer further developments to the next iteration of the project.

4.2 Mercury

Unlike Flitan, most of Mercury's input variables do not require any kind of data encoding due to their already quantitative nature. Therefore, these inputs can be used directly by the metamodel, rendering the previously mentioned translation layer [9] mostly unnecessary. Some transformations may still be applied, such as data normalization, though not in terms of inherently encoding the variables' values.

For Mercury, and in order to assess the performance of solutions PJ01.01 and PJ07.02., a set of seven input variables and eight output variables were considered, summarized in Table 12 and Table 13, respectively. Except for the "Hotspot solver", all the remaining variables encompass quantitative (and continuous) values. For this reason, the former was encoded via one-hot-encoding.

4.2.1 Results

The main objective of this set of experiments was to explore and show the advantages of employing active learning to guide the exploration process of Mercury's input space. For each output variable, a different metamodel (in this case, a Gaussian Process) was independently used. Ultimately, we aim at approximating the simulator with a metamodel that should be able to predict the value of a single output within the predefined practical ranges given in Table 1 in the previous section. In other words, we seek not to find any specific or optimal set of inputs but to correctly predict the output of the simulator given that, e.g., if the fuel price is 1 or 5, or if the claim rate is 10% or 90%. Once again, one active learning strategy is presented and compared against the random selection of data points. Whereas the former is based on the data points with the highest predictive variance in each iteration, the latter has effectively no query criteria, thus representing both a simple and naive baseline.

Figures 14-21 present the results for the eight studied output variables of interest in the scope of solutions PJ01.01 and PJ07.02 using Mercury's scenario 9. In practice, the user should only run the active learning once, but in order to quantify the variation in the active learning strategies (mostly originated from the simulator's stochasticity and the first few randomly chosen simulations/samples), the two different strategies are run 30 times. Within each plot, the solid lines depict the average run and the shades are +/- one standard deviation. Here we use the negative log-likelihood to assess the fitting performance of the metamodel given both mentioned learning strategies. The likelihood function describes how likely is a certain model, along with its parameters, to generate the observed data, capturing both the error and the stochasticity/noise. Generally, given some data, the higher the likelihood of a model is, the better. In fact, it is common, especially within the Gaussian Process framework, to estimate the "best" models' parameters by maximizing the associated likelihood function, or equivalently, by minimizing the negative log-likelihood, given some training data. In this sense, the negative log-likelihood can be viewed as a kind of loss function which we trivially aim to minimize.

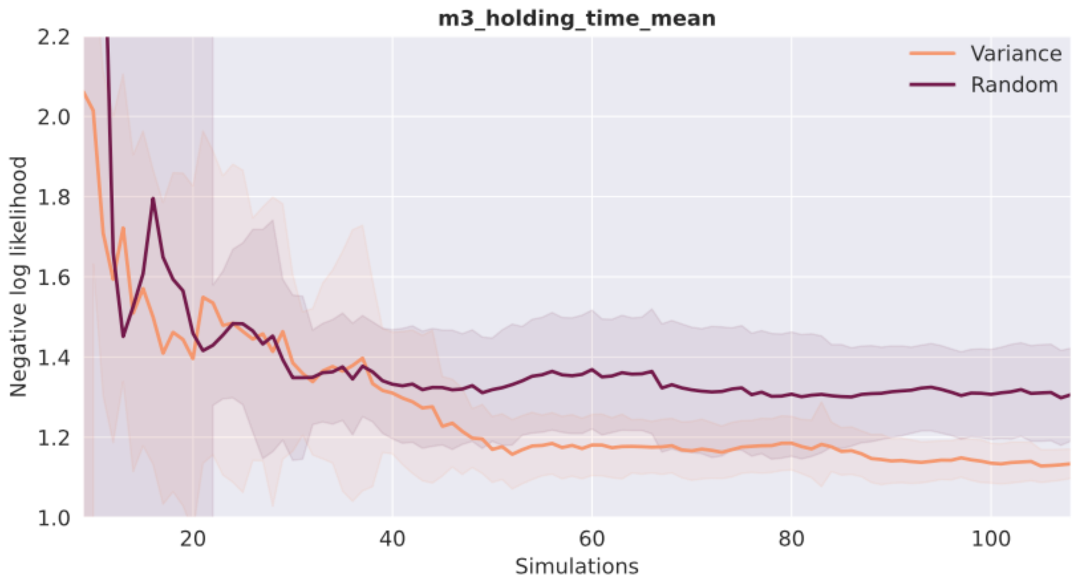


Figure 15. Active learning metamodelling performance for holding time.

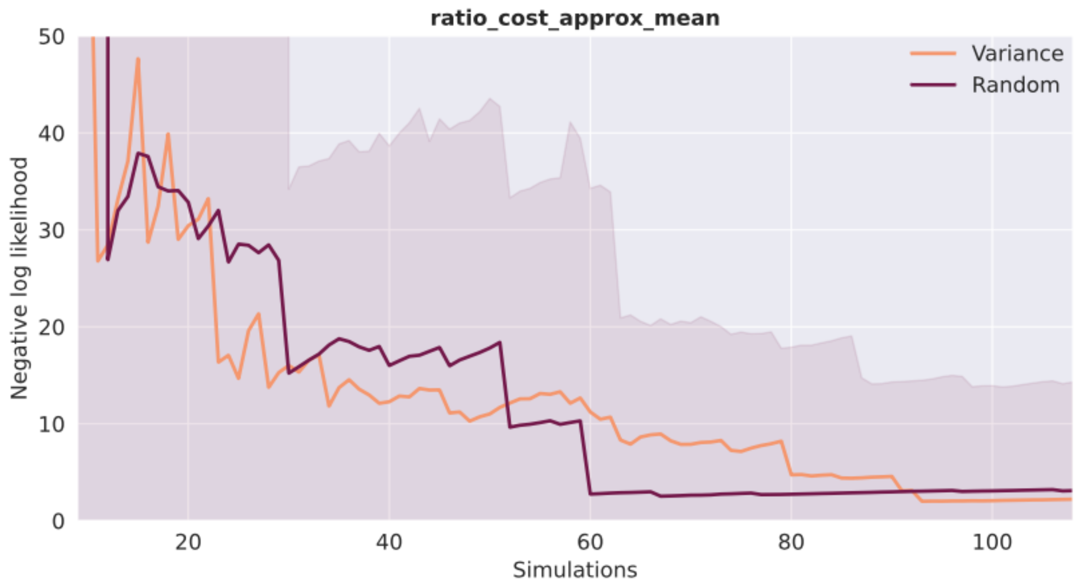


Figure 16. Active learning metamodelling performance for the saved declared cost in regulation w.r.t. FPFS allocation.

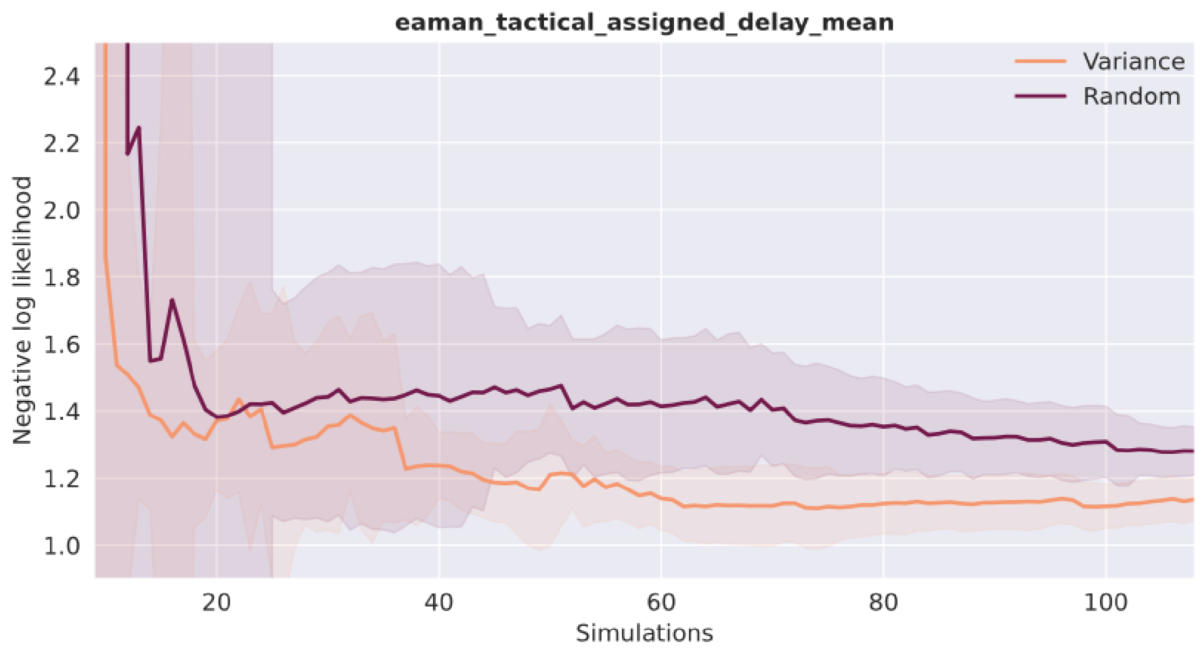


Figure 17. Active learning metamodelling performance for the assigned delay at tactical horizon.

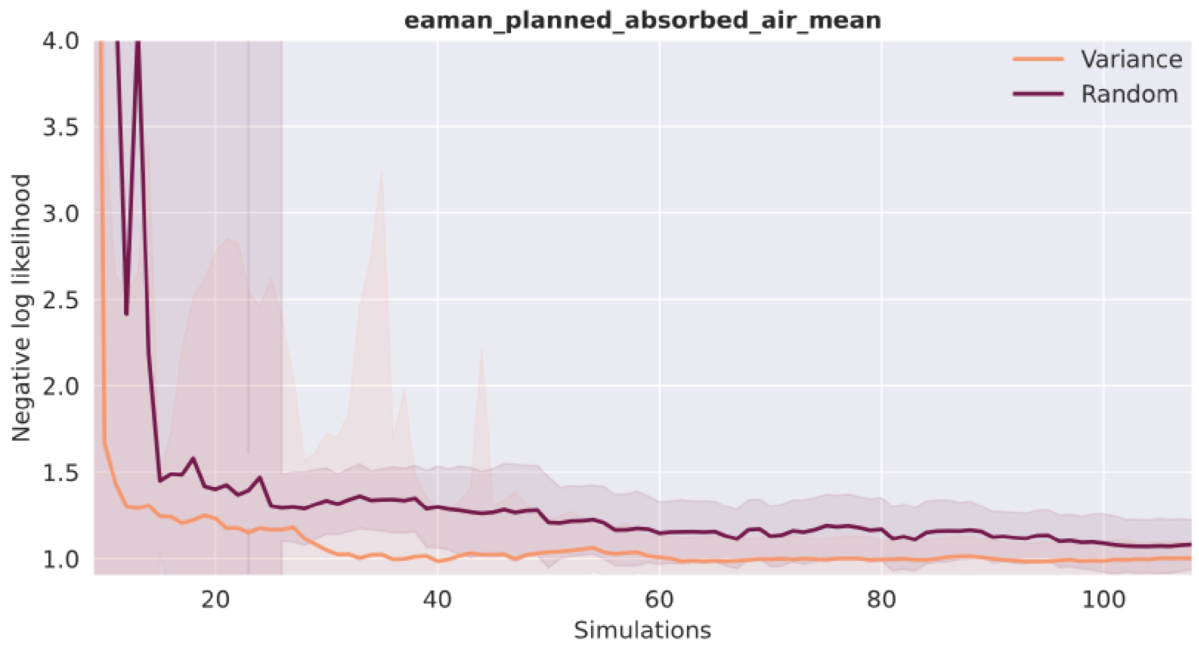


Figure 18. Active learning metamodelling performance for the planned absorbed delay.

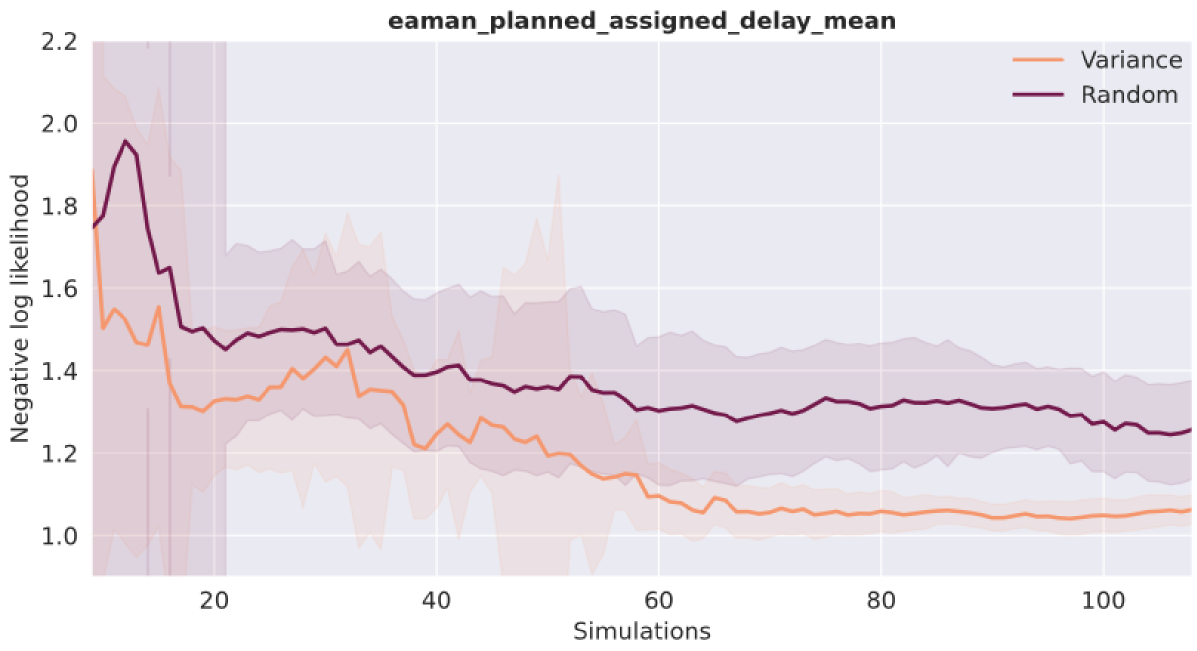


Figure 19. Active learning metamodelling performance for the assigned delay at planning horizon.

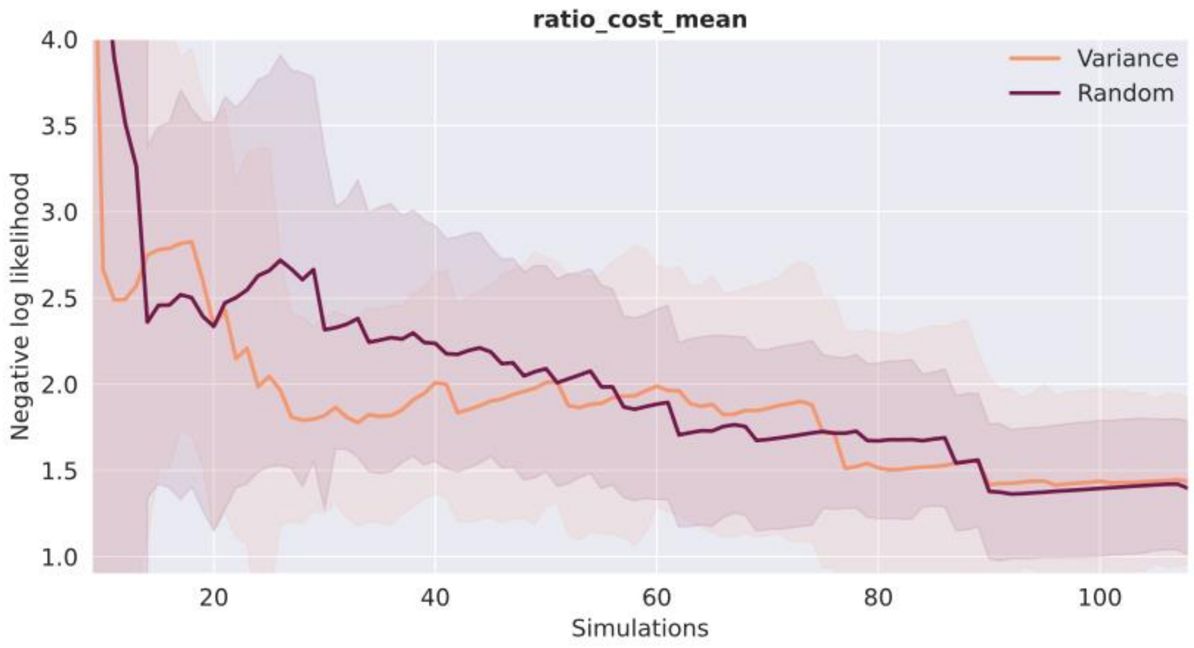


Figure 20. Active learning metamodelling performance for the saved real cost in regulation w.r.t. FPFS allocation.

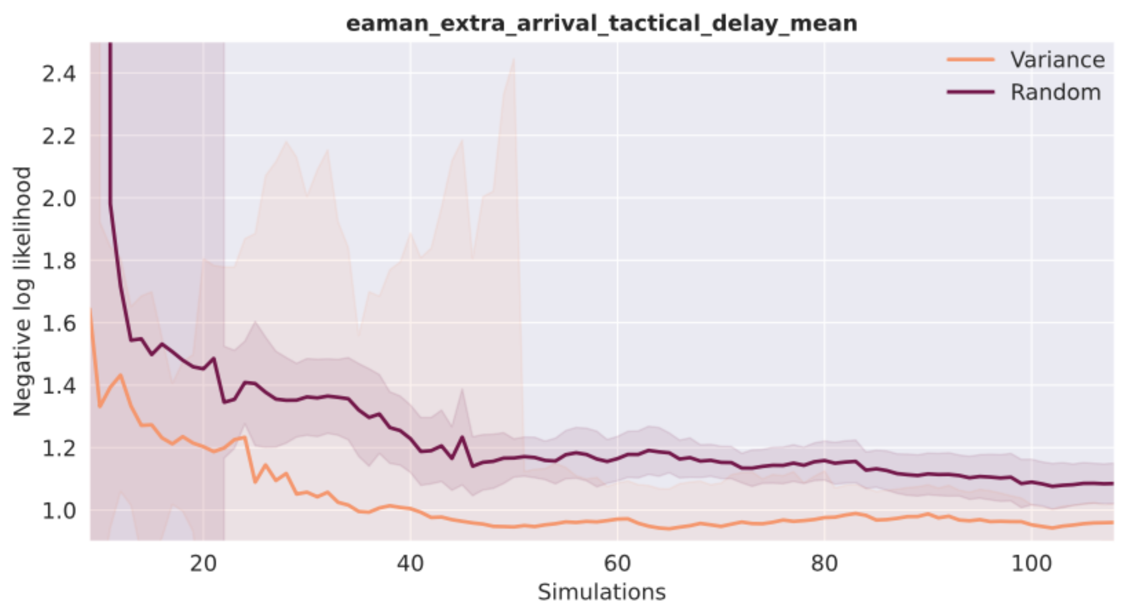


Figure 21. Active learning metamodelling performance for the extra tactical delay.

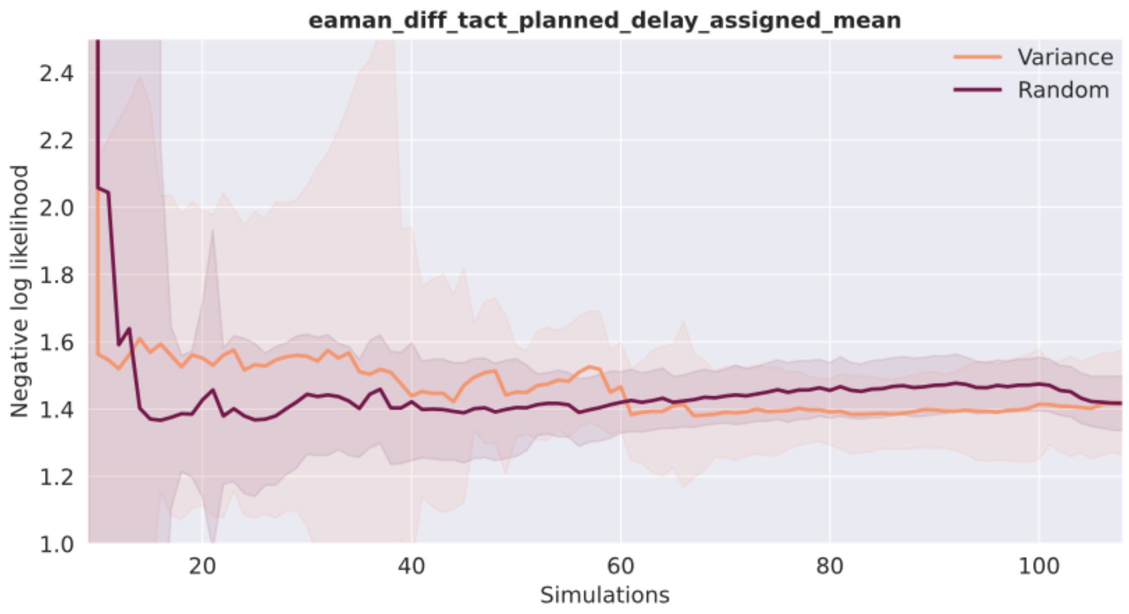


Figure 22. Active learning metamodelling performance for the difference between the tactical and the assigned delay.

Remember that active learning aims to use as few data points as possible. Starting from a rather small training data set, the process evolves sequentially as new simulation results are added to the latter, thereby expanding it. At each active learning step, the parameters of the metamodel are re-estimated with the additional new labelled data (i.e., new simulation result) originating from the simulator through query requests. While initially high, when just a few points are used to fit the metamodel, it is expected that the likelihood tends to decrease towards the model's modelling performance threshold as the process advances. We can clearly observe this behaviour in Figures 14-21. Across all the studied outputs, with a few exceptions, we generally see that at least after circa 40-60 simulation runs, both metamodels begin to stabilize with respect to the attained negative log-likelihood. Naturally, if the number of simulation runs is large enough, then any kind of query strategy will eventually lead the metamodel to achieve its modelling plateau. However, our objective with active learning is to achieve the lowest values as soon as possible or, in other words, with the least number of simulation runs, assuming a certain computational budget or minimum accuracy requirements.

Except for the "eaman_diff_tact_planned_delay_assigned" output (see Figure 22), the active learning strategy shows, on average, a superior performance against the random approach, generally reaching lower negative log-likelihood values in earlier modelling stages.

It is easy to observe that the performance of the metamodel varies as a function of the output being considered. A different GP was employed to model the outputs individually. Since each of the latter ultimately describes a different input-output mapping, it is not a surprise that the individual GPs perform differently in both prediction and active learning performances. For this reason, more often than not, it can be tricky to choose a clear cut-off point from which we decide that the metamodel is performing well enough concerning the modelling needs and objectives at hand. In the present case, we can arguably assume that, with approximately 50 simulation runs or less, the metamodel attains a reasonably good performance across all the output metrics, additionally nearing its modelling threshold at the same time.

In Figure 23, we present two correlation matrices computed across the output variables of the metamodel predictions (left) and real simulation results (centre). Additionally, for enhanced comparison, we also computed the entry-wise difference between the latter and the former (right). The metamodel in question was trained with 50 simulation results chosen by active learning guided by the predictive uncertainty in "Assigned delay at planning horizon". The predictions were computed over a random set comprised of 1k input data points (simulation results). Here we can observe that the metamodel captures the simulator's output behaviour generally well, attaining a reasonably good prediction across the board. Recall the approximate nature of the metamodel, which is also clearly displayed in these results.

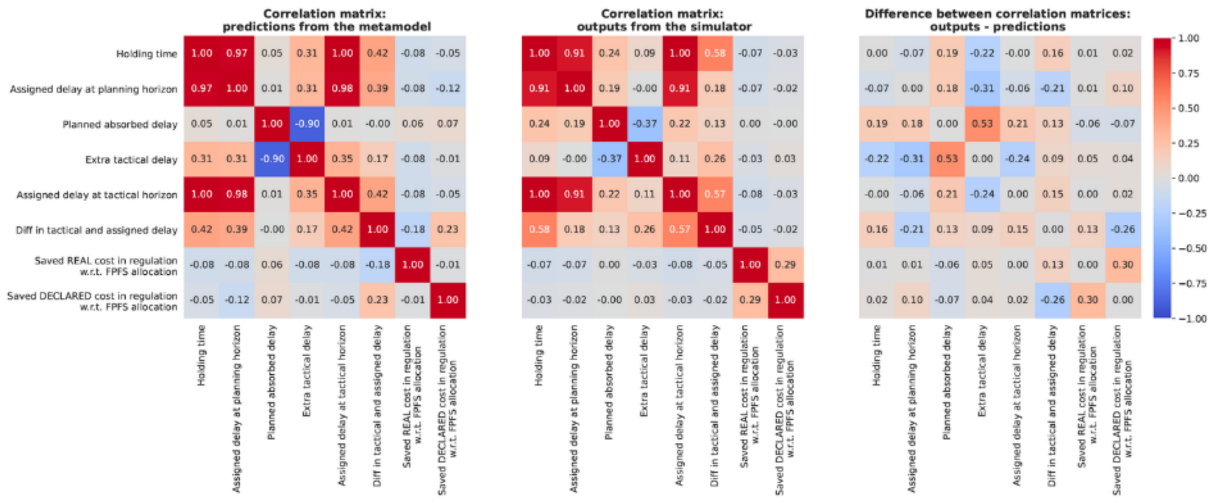


Figure 23. Correlation matrices computed across the output variables of the metamodel predictions (left) and real simulation results (centre).

4.3 Computational analysis preview

We now provide a brief overview of the computational workloads and experimental settings involved in the reported experiments.

All the experiments were conducted on a shared 24-core 3.8 GHz CPU with 128 GB RAM under Ubuntu 20.04 LTS. Depending on its use, a single simulation run of Flitan can approximately take up to two minutes, whereas Mercury can reach up to four. As of now, in both cases, we are using simplified scenarios to facilitate the exploratory nature of the present work and more easily gain new insights and develop guidelines to readily embrace the modelling challenges of the upcoming steps. Nonetheless, in the next iteration of the project, we aim at exploring more realistic simulation settings which will inevitably lead to increased computer running times. In any case, the results reported herein show the potential of the presented methodology in minimizing the computational burden of systematic computer experiments within the ATM research field and its simulation modelling applications.

As mentioned before, the reported Flitan's DAC results refer only to a single day of traffic data and air traffic controller. In practice, it takes the day D of interest, plus days D-1 and D+1. We can reasonably assume its time complexity to be mostly a linear function of the number of days being processed and the number of daily flights. For example, considering the current one-day data from Madrid ATC, Flitan's runtimes for each different opening scheme are expected to easily increase to approximately one hour if we consider a month of daily traffic data.

In its turn, Mercury's time complexity seems to be generally related to the number of flights and passengers, and layovers, as well as to the type of optimizing procedures (e.g. integer-programming solvers) underlying the simulation model's implementation itself. Currently, a single run for the scenario loaded with CDG airport data and encompassing just 1423 flights, takes roughly 3 to 4 minutes to conclude. Similar run times are observed with another scenario with lower levels of delays. On the other hand, similar scenarios with only departure flights at CDG both contain 711 flights, yielding about 2 minutes. Moreover, from previous experiments with a more complete 27k flights scenario [18], we obtain times in the order of 20+ minutes. Hence, the number of flights, and consequently their intrinsic interactions, have an expected trivial impact on the simulator's time complexity, seemingly a non-linear one.

Hereupon, the ultimate aim of the employed methodology is to reduce the computational burden underlying systematic simulation-based analysis while, at the same time, contributing to an efficient way of guiding the exploration process of the simulators' input-output behaviour. Note, however, that is it not our intent to entirely replace the simulation in question with the corresponding metamodel, as the simulator itself constitutes a crucial provider of the ground truth data. In this sense, this approach aims at deploying both the metamodel and the simulator in a bundled solution, where the performance and trade-offs are constantly monitored, potentially allowing for interactive supervision by ATM experts throughout the entire process.

Given an arbitrary case study, guidelines for sampling computer observations from the underlying simulators, may not be entirely clear or, for that matter, unanimously established across the entire ATM field. For this reason, it is generally difficult to adopt a baseline experimental design to which we compare our results and finally assess how much computational burden are we saving in practice. Regardless of the rather generalized or ad-hoc way of conducting such computer experiments, the required number of simulation runs is by and large conditional on the output stochasticity, dimensionally of both input and output spaces, specificities of the case study, and, naturally, on the simulation model itself. In any case, a reference of 100 model (simulation) runs, suggested by [8] within a particular application of network-wise assessment using Mercury, is adopted in the following, generally representing what one would do in the absence of any active learning metamodelling strategy.

Previously, we argued that 80 simulation runs seem to be a candidate cut-off point in which we obtain a reasonably good metamodel for Flitan addressing the DAC solution. In turn, this decision would lead to savings of approximately 40 minutes. With respect to Mercury, we argued that 50 model runs would suffice to train the metamodel, therefore saving up to 200 minutes in this case. Although these processing runtimes refer to rather simplified scenarios, they provide us with a glimpse of the potential savings with more realistic ones.

On the other hand, we also must take into account the underlying workload associated with the metamodelling process itself. The GP framework used as a metamodel has cubic complexity, N^3, meaning that the computing time is proportional to the cube of the number N of training points. As pointed out in [19], it can become a real computational issue in settings with more than 10k data points. There are several solutions that aim at reducing the complexity of the GPs, such as, via the introduction of inducing points [20], [21], or spectral representation [21]. However, this is not the typical problem we are addressing in this project. On the contrary, we are interested in maximizing the insights with respect to the simulator of interest, while minimizing the number of simulation observations (or runs) required to do so.

In this context of a relatively low number of data points, the complexity of the GPs is unlikely to pose a significant hindrance to the overall active learning metamodelling process. Finally, it is worth mentioning that, as with most machine learning techniques, after a GP is fitted to the training data, its prediction time is generally fast and oftentimes negligible in practical terms. Within the current setup, a batch of 50 iteration steps takes less than 3 minutes to process with respect to the metamodel's sequential refitting, regardless of the simulator used. Note that the complexity of GPs is a function of the number of data points and not so much on the dimensionality of the input space.

5 Conclusions and next steps

The two simulators, Flitan and Mercury, have different strengths and weaknesses when it comes to modelling. For this reason, we selected different solutions, the concepts of which could be implemented in the simulators.

The development in Flitan first focused on airspace dynamic configurations. By implementing really detailed rules for merging and splitting sectors, the model is able to assess various sector configurations opening schemes. The development in Flitan then focused on runway traffic optimisation. By implementing specific rules for runway capacity, mirroring the actual capacity variations due to different configurations, Flitan is now able to assess different possible runway configurations and their associated activation plans.

Mercury development was focused first on the UDPP solution, allowing airlines to swap their flights delayed by an ATFM regulation. By implementing several algorithms for solving hotspots, Mercury is now able to assess how efficient UDDP and these other algorithms are, and what other impact they may have on the air transportation system. Further, a simple extension of the EAMAN concept was implemented in Mercury, where a better proxy for the airlines' costs are taken into account when assigning speed changes and holding delays. This allows to estimate not only the amount of delay reductions, but also the related costs, improving the overall performance assessment of the solution implementation.

The results of the metamodel show that the procedure of active learning is very efficient in most cases, as opposed to random estimations of the micromodels. For Flitan, after 80 simulation points, the results from the model are particularly satisfying, especially from the variance point of view. Furthermore, the experiments with this simulator also highlighted the importance of “translation layers” that transform categorical or string-based data in quantitative values so that it can be handled by most machine-learning techniques, particularly the GPs used in this work. For Mercury, independent GPs used on different output variables show that the active learning procedure is reasonably better than random sampling and that 50 runs of the micro-simulator allow attaining a good approximation already. Correlation matrices – which only capture linear relationships between variables – show that the metamodel captures well some of the relationships between variables, but not so much for others. It might be the case that the metamodel is capturing non-linearities on the latter variables, which cannot be reported via the correlation coefficient. Nonetheless, alternative performance metrics should be used and further investigation will be conducted in the near future.

It is important to always bear in mind the approximate nature of the metamodel. It is not our expectation that it can fully and perfectly capture the functional form mapping the simulator’s inputs into the outputs. Instead, we acknowledge that the metamodel is a de facto approximator of the underlying simulation model and should be used as an auxiliary exploratory tool to enhance simulation-based studies. As such, the critical trade-off between accuracy and computational speed should be constantly supervised and adjusted, whenever possible and needed, following the metamodelling’s initial objectives.

Although active learning provides a rather automatic approach to the training of the metamodel, it should not waive frequent monitoring and assessment from a domain expert throughout the entire process. Fine-tuning and other adjustments might be required in more complex settings. To this end, the metamodel should be deployed in a bundle together with the simulator itself and allow for a certain degree of interactivity with the user. Whereas the metamodel helps reduce the redundancy of simulation-based processes, providing directions on how to explore the simulation input space more efficiently, the simulation model guarantees that this exploration is supported on solid ground truth data (simulation results). The metamodel can be particularly relevant in suggesting potential trends in the simulator's behaviour and uncovering associations between the involved variables or ultimately serving as a confirmatory tool of domain knowledge premisses.

In summary, the results from the metamodel are satisfactory, despite requiring more work to understand why some of the variables are less well modelled than others. The next steps for the project will thus be:

- to analyse in more depth, particularly from the domain knowledge point of view, the results obtained with both the micromodels and metamodel, which will then be assessed in WP7.
- to investigate how well the metamodel reproduces the micromodels, in particular with respect to training times and execution time, and its limitations.
- to expand the concepts presented herein for the third iteration of the micromodels, using more realistic scenarios, advanced algorithms and combining them whenever possible.

6 References

[1] NOSTROMO , "D4.1 Preliminary Specification of Case Studies," 2021.

[2] NOSTROMO, "D3.2 Metamodels Requirements Specification".

[3] EUROCONTROL, *DDR2 reference manual for generic users. Technical Report V. 2.1.2,* EUROCONTROL, 2015a.

[4] A. Cook and G. Tanner, " European airline delay cost reference values, updated and extended values. Technical report,," University of Westminster, 2015.

[5] Cassiopeia project, "DCI-4HD2D - D3.2 Final technical report. Technical report," 2016.

[6] ComplexityCosts project, "D4.5 – Final Technical Report. Technical report," SESAR JU, 2016.

[7] EUROCONTROL, " CODA Digest: All-Causes Delay and Cancellations to Air Transport in Europe – 2014. Technical report," EUROCONTROL, 2015b.

[8] 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.

[9] EUROCONTROL, "ATFM Modelling capability - AMOC," EUROCONTROL, 1997.

[10] BEACON consortium, "D3.1 High-level modelling requirements," 2021.

[11] L. Friedman, the simulation metamodel, Springer Science & Business Media, 2012.

[12] J. P. Kleijnen and R. G. Sargent, "A methodology for fitting and validating metamodels in simulation," *European Journal of Operational Research,* vol. 120, no. 1, pp. 14-29, 200.

[13] R. Gramacy, Surrogates: Gaussian process modeling, design, and optimization for the applied sciences., Chapman and Hall/CRC, 2020.

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

[15] X. Wang and J. Zhai, Learning with Uncertainty, CRC Press, 2016.

[16] NOSTROMO, "D3.1 Preliminary Metamodeling Methodology," 2021.

[17] F. Viana, "A tutorial on Latin hypercube design of experiments.," *Quality and reliability engineering international,* vol. 32, no. 5, pp. 1975-1985, 2016.

[18] NOSTROMO, “D3.3 NOSTROMO Framework API + Associated documentation,,” NOSTROMO, 2020.

[19] G. Gurtner, L. Delgado and D. Valput, “An agent-based model for air transportation to capture network effects in assessing delay management mechanisms,” *Transportation Research Part C: Emerging Technologies,* vol. 133, p. 103358, 2021.

[20] C. Williams and C. Rasmussen, Gaussian Processes for Machine Learning, Cambridge, MA: MIT Press, 2006.

[21] J. Quinonero-Candela, C. E. Rasmussen and C. K. Williams, “Approximation methods for Gaussian process regression,” in *Large-scale kernel machines*, MIT Press, 2007, p. 203223.

[22] H. Bijl, J. W. van Wingerden, T. B. Schön and M. Verhaegen, “ Online sparse Gaussian process regression using FITC and PITC approximations.,” *IFAC-PapersOnLine,* vol. 48, no. 28, pp. 703-708, 2015.

[23] M. Lázaro-Gredilla, J. Quinonero-Candela, C. E. Rasmussen and A. R. Figueiras-Vidal, “Sparse spectrum Gaussian process regression.,” *The Journal of Machine Learning Research,,* vol. 11, pp. 1865-1881, 2010.