An Automated Vehicle System Architecture Exploration, Evaluation and Uncertainty-Based Design Optimization Framework

L.H.L. Halsema

Delft University of Technology

TU Delft

An Automated Vehicle System Architecture Exploration, Evaluation and Uncertainty-Based Design Optimization Framework

by

L.H.L. Halsema

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Friday August 9, 2024

Student Number:	4659953	
Project Duration:	September, 2023 - August, 2024	
Thesis Committee(s):	Dr.ir. G. La Rocca	TU Delft, Supervisor
	S. Van Rijn	VDL Special Vehicles, Supervisor
	Dr.ir. MFM Hoogreef	TU Delft, Chair
	Dr.ir. R Merino-Martinez	TU Delft, Examiner

Faculty of Aerospace Engineering, Delft

TU Delft

Acknowledgement

This report marks the end of my studies at Delft University of Technology, where I completed both my Bachelor's and Master's degrees in Aerospace Engineering. During my Master's program, I extended my knowledge in the field of Flight Performance and concluded my academic career with a thesis project focused on Uncertainty-Based System Architecture Optimization for zero-emission drive trains.

I am deeply grateful to my thesis supervisors, Gianfranco La Rocca (TU Delft) and Sebastiaan van Rijn (VDL Special Vehicles), for their invaluable guidance, support, and feedback throughout this project. Their expertise and insights have been crucial in shaping this work.

I also want to thank the friends I made during my academic career. Our engaging conversations and shared passion for learning have greatly enriched my experience. It's important to continuously challenge oneself and stay proactive, especially in addressing current issues like climate change and societal challenges. Furthermore, I am thankful to my family for their support and encouragement, which allowed me to fully commit to my studies and achieve this engineering title.

Finally, I wish to express my sincere appreciation to Delft University of Technology. The education and experiences I gained here have been essential in shaping my academic and professional path.

L.H.L. Halsema
Delft, August 2024

"To be uncertain is to be uncomfortable, but to be certain is to be ridiculous."

Contents

Introduction

In response to increasing challenges posed by climate change, there is growing urgency within hard-to-change industries to adopt zero-emission solutions, driven by increasingly stringent regulations. Consequently, companies across various sectors (automotive, aerospace, maritime etc.) are turning to system integrators specializing in zero-emission drive trains to convert conventionally hydrocarbon-propelled vehicles into zero-emission alternatives. Projects focused on building zero-emission vehicles typically involve high levels of risk, due to limited knowledge about the behavior of emerging technologies, unknown effects of connecting various components, and the low technology readiness levels (TRL) of such zero-emission components. In the industry, this risk is managed by applying system engineering principles, such as building according to the standardized V-model.

One of the first steps in the V-model, is defining a system architecture. Due to the very large mixed-integer, hierarchical, combinatorial design space at this stage in the design cycle, system architects have trouble finding the optimal system architecture. Traditionally, this has been addressed through the introduction of expert judgement and safety factors to limit the design space and mitigate uncertainties, often resulting in sub-optimal system architectures. This approach, while effective for simpler systems in the past, proves inadequate for innovative products such as zero-emission drive trains due to their complexity, integration of advanced technologies, and stringent performance standards.

In this research, a different approach is implemented, which involves the implementation of uncertainty quantification and propagation into Multidisciplinary Design Analysis and Optimization (MDAO) practices focussed on System Architecture Optimization (SAO). This approach leads to the development of robust system architecture solutions that meet specified requirements, while also allowing for thorough exploration of the design space at reduced computational expense.

The current research aims at answering the following two research questions:

1. *How can uncertainty quantification and propagation be integrated into an automated system architecture optimization workflow for exploring and optimizing vehicle system architectures under uncertainty?*
2. *How can the integration of uncertainty quantification and propagation in system architecture optimization processes improve requirement risk management for innovative conceptual design studies of complex engineering systems?*

Lastly, the thesis report is composed of three parts. Firstly, in **Part I**, a scientific paper is presented highlighting the research as explained previously. **Part II** includes the relevant Literature which supports the current research. This has already been graded under the TU Delft master course 'AE4020'. Lastly, **Part III** provides additional supporting work, related to the research.

1

Scientific Paper

An Automated Vehicle System Architecture Exploration, Evaluation and Uncertainty-Based Design Optimization Framework

L.H.L. Halsema
Faculty of Aerospace Engineering, Delft University of Technology, Delft, The Netherlands

System integrators across industry adhere to the so called V-model to manage their product development process. The V-model starts with the conceptualization of a system architecture, guided by functional and non-functional requirements and ending with the verification and validation of the designed product. In the conceptual design stage, crucial decisions impacting the performance of the final product are made. However, due to limited knowledge about the behavior of emerging technologies, unknown effects of connecting various components, and the low technology readiness levels (TRL) of components, the architectural design space for innovative design projects becomes large and uncertain. This uncertainty is traditionally managed through engineering judgement and large safety factors, which restrict design space exploration and often lead to sub-optimal solutions.

To address this, an uncertainty-based system architecture design exploration and optimization framework is developed in which system architecture design inputs and models, commonly used in model-based systems engineering, are treated as stochastic. This is integrated into a multidisciplinary multi-objective surrogate-based optimization framework aimed at finding robust optimal vehicle system architectures.

The proposed framework has been successfully applied to the design of a heavy duty electric vehicle drive train. This resulted in increased system-specific knowledge of the vehicle's performance in real-world applications by finding the sensitivity of the optimal design solution to operational- and model-uncertainties, thereby facilitating better informed design decisions, enabling trade-offs and reducing project risks.

Nomenclature

α	=	Vehicle inclination angle
$\mathbf{B}$	=	The (selected) subset of feasible system architectures
$\mathbf{G}_{feasible}$	=	The set of feasible system architectures
$\mathbf{J}^*$	=	State-time performance matrix
$\mathbf{K}$	=	The set of reachable valid component states
$\mathbf{u}(t)$	=	Input vector extracted from the drive cycle
$\mathbf{x}_{control}$	=	Component control vector
$\mathbf{x}_{design}$	=	Component design vector
$\mathbf{x}_{state}$	=	Component state vector
$\mathcal{N}$	=	Normal (Gaussian) Distribution
$\mathcal{U}$	=	Uniform Distribution
θ	=	Optimization weighing factor
a	=	Vehicle acceleration
s, $s_{\Delta SoC}$	=	Quantified Uncertainty Measure (of ΔSoC)
V_{wind}	=	Wind speed

(D)(A)WG = (Directed) (Acyclic) Weighted Graph
(UB)SAO = (Uncertainty-Based) System Architecture Optimization
BSP = Boolean Satisfiability Problem
GA = Genetic Algorithm
MBSE = Model-Based Systems Engineering
MDAO = Multi-disciplinary Design Analysis and Optimization
PMP = Port-Matching Problem
SATG = System Architecture Topology Generator
SBO = Surrogate-Based Optimization
SoC = State of Charge
SVD = Singular Value Decomposition
TRL = Technology Readiness Level
XML = Extensible Markup Language

I. Introduction

The most commonly used design cycle used in industry is the V-model [1]. The V-model starts by defining mostly functional, and some non-functional, system requirements as specified by the customer. From this, a system architecture is to be defined. A system architecture is defined by Crawley et al. as [2]:

> *"The description of what components a system consists of, what functions they perform (i.e. why they are there), and how they are connected and related to each other."*

Due to the very large mixed-integer, hierarchical, combinatorial design space at this stage in the design cycle, *system architects* have trouble finding the optimal system architecture given the specified (non-) functional requirements, as an exhaustive search is impractical if not impossible [3] [4]. This increases project risks because it involves managing a vast number of potential configurations, each with its associated uncertainty and potential issues. This can lead to project delays and cost overruns as more time and resources are required to evaluate and validate each possible architecture [5].

Traditionally, these challenges have been addressed through the introduction of expert judgement and safety factors to limit the design space and mitigate uncertainties, often resulting in sub-optimal system architectures. This approach, while effective for simpler systems in the past, proves inadequate for innovative products such as zero-emission drive trains due to their complexity, integration of advanced technologies, and stringent performance standards.

In current literature, typically two new solution strategies have been proposed. The first approach, gaining traction due to advancements in computational power facilitated by modern CPU and GPU accelerated high-performance computing, focuses on increasing the fidelity of simulation tools early in the design cycle. By decreasing model errors, overall uncertainty in system architecture outcome is reduced. However, this solution strategy is constrained by the computational cost of function evaluations, limiting exploration of the entire design space.

A different approach, gaining recent interest in literature [6] [7], involves the implementation of uncertainty quantification and propagation into Multidisciplinary Design Analysis and Optimization (MDAO) practices. This solution strategy assumes inherent inaccuracies in models and variables. It integrates uncertainty quantification into the exploration and evaluation phases of system architecture development. This approach leads to the development of robust system architecture solutions that meet specified requirements, while also allowing for thorough exploration of the design space at reduced computational expense.

However, an automated framework that integrates architectural design space exploration, uncertainty quantification and propagation, and multidisciplinary design optimization is still lacking in the literature. Such a framework could be applied to innovative design studies, such as zero-emission drive trains.

This translates to the objective of this research:

> *This research aims to develop a generic methodological framework for automated exploration, evaluation, and uncertainty-based design optimization of vehicle system architectures, considering uncertainties in models and inputs. Additionally, the framework aims to be applied to the design of zero-emission heavy-duty vehicle drive trains, with a focus on minimizing performance uncertainty. The goal is to identify a robust optimal design that satisfies both qualitative and quantitative requirements.*

To achieve this objective, the research question that guides this study is twofold

1. *How can uncertainty quantification and propagation be integrated into an automated system architecture optimization workflow for exploring and optimizing vehicle system architectures under uncertainty?*
2. *How can the integration of uncertainty quantification and propagation in system architecture optimization processes improve requirement risk management for innovative conceptual design studies of complex engineering systems?*

II. Methodology

The proposed framework is divided into four main parts: Architectural Design Space Exploration (subsection II.A), Architectural Design Space Evaluation (MDA) (subsection II.B), Uncertainty propagation in coupled MDA (subsection II.C) and the integrated Uncertainty-Based System Architecture Exploration and Optimization framework applied to zero-emission vehicle drive trains (subsection II.D). Figure 1 highlights the process used in this research, including the appropriate section labels.

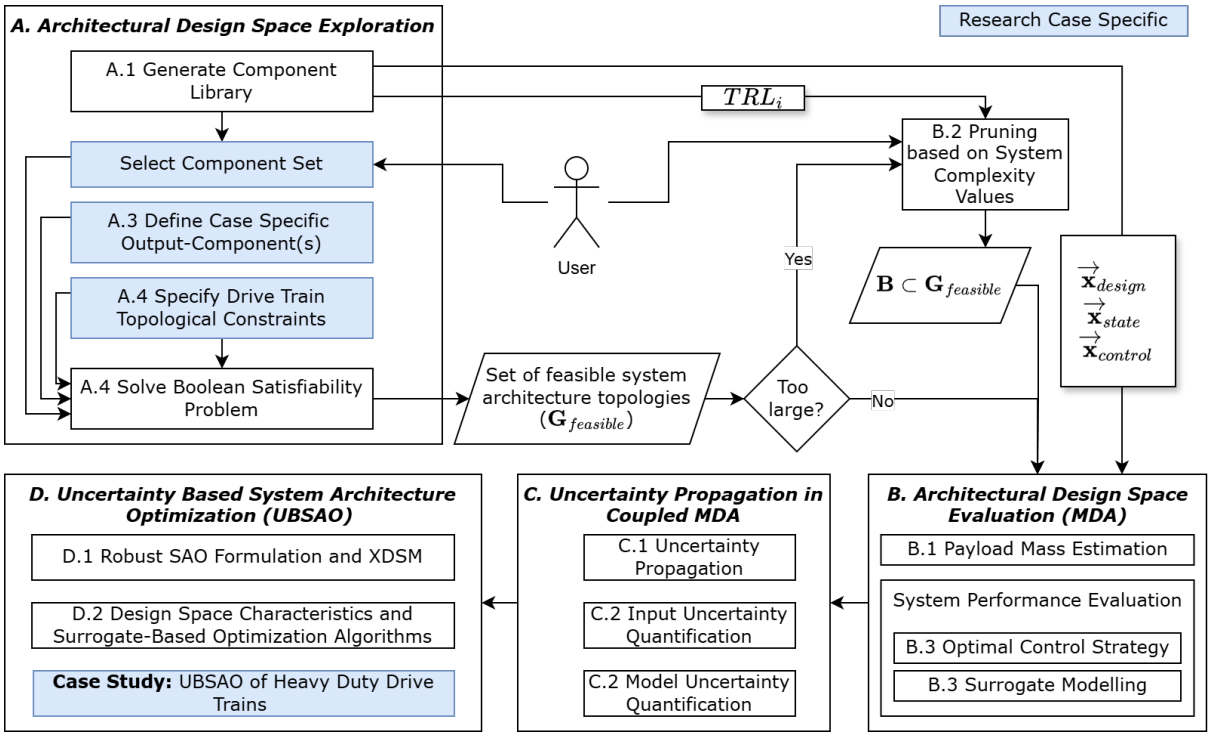


Figure 1. Flowchart depicting the steps in the methodology for developing the uncertainty-based system architecture exploration and optimization framework used in this research.

A. Architectural Design Space Exploration in Conceptual Design

In response to the increasing complexity of modern engineering projects and their associated challenges, such as project overruns and failures, the significance of system engineering principles has become more pronounced. As a result, most complex engineering projects use the V-model for their product or system development process.

One of the first steps in the V-model, as illustrated in Figure 2, is defining the system architecture given the specified functional and non-functional requirements. Unfortunately there is a high level of uncertainty regarding system performance at these early design stages, as a result of the lack of knowledge on component- and system behavior. However, selecting an appropriate system architecture is critical, as it forms the foundation of the overall product or system design. If a system architecture is chosen that fails to meet the specified requirements, resolving these issues becomes very costly, requiring multiple iterations of the V-cycle and resulting in substantial cost overruns. This is depicted in Figure 3.

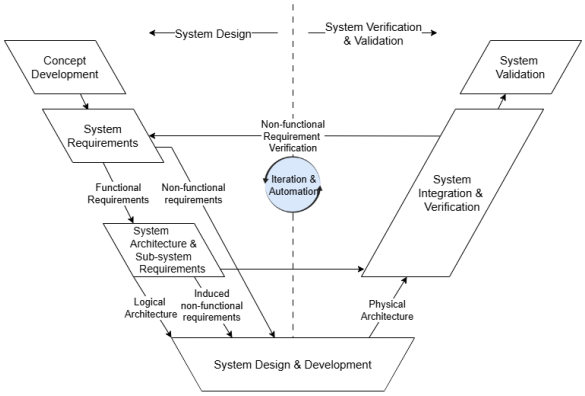


Figure 2. The V-Model commonly used in (hard/-soft)ware system development [5].

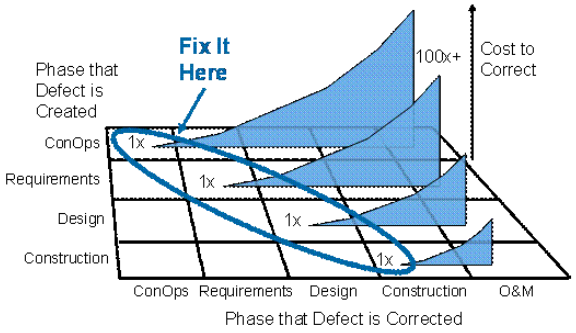


Figure 3. Visualization of the cost to correct issues generated in various stages of the design process [1].

The architectural design space in conceptual design phases is a highly mixed-integer, hierarchical, constraint and combinatorial design space [4]. To explore the large combinatorial system architecture design space, various methods have been proposed in literature. They can be categorized into two sections: a *bottom-up* and *top-down* approach.

- **Top-Down:** *Functional Decomposition:* The method of applying functional decomposition for system architecture optimization is commonly found in literature [8] [9] [10], as it allows for a top-down approach for generating system architectures. Functional decomposition starts by first defining a top-level requirement or function, for which then a form is found which fulfills that function. As this form usually requires its own functionalities, functional decomposition is then used to build a system architecture. Limitations of this method include inadequate consideration of spatial locations and component interrelationships, a rigid hierarchical structure, and a lack of consideration for components that may serve multiple functionalities within complex systems.
- **Bottom-Up:** *Morphological Matrices:* The method of using Morphological Matrices (MM) is first proposed by Weber & Condoor in 1998 [11] and is a very common initial design space exploration method used in conceptual design phases [12] [13] [14]. A MM is generated by first coming up with all viable systems or components which are able to fulfill a specified function. Here, *the forms of a function* are enumerated. By then allowing to mix-and-match the various components or systems from various functions, the design space can be explored. Several extensions to this method have been proposed to address system incompatibilities or incorporate system preference. A limitation to this method is the single-functionality principle, where morphological matrices only allow a single component to fulfill a single function, whereas in current complex systems, multi-functionality of components should be considered. Moreover, the evaluation of the morphological design space is extensive.

In this research, a bottom-up architecture design space exploration approach is implemented. This

approach is preferred by system integrators, and is driven by established relationships with a wide range of partners and suppliers, as well as a risk-averse approach. Consequently, most engineering projects begin by identifying existing systems and components that are familiar and meet the specified requirements. This approach limits the need for expensive engineering work on new systems and technologies and leverages existing, proven products to reduce project risk levels.

The process begins with a set of components from the component library and one or more output components. A Port-Matching Problem (PMP) is then solved to generate viable system architecture topologies. The steps involved, as illustrated in Figure 1, will be detailed further in the following sections.

1. Generating a component library

Typically, a system design process for system integrators starts by contacting strategic suppliers to find components capable of fulfilling a pre-specified function. This is done with all strategic partners to create a so called *component library*, where multiple specific components are found for each component type. This results in a clear overview of the number and types of components available to the integrator. In this thesis, the information for each component type is provided in the form of an XML file. The to-be-provided information per component type is explained below:

- **Name**: Unique component type name.
- **TRL**: Technology Readiness Level of the component type (Range[0,9]).
- **maxInstances:** The finite number of instances (integer) that the component type can be present in the design space.
- **typeInput, typeOutput:** An energy input/output type (see subsubsection II.A.2) that the component has. If the component does not have an input and/or output, this is kept empty.
- **minRangeInput**, **maxRangeInput**, **minRangeOutput**, **maxRangeOutput**: The minimum and maximum number of in/outputs per specified *typeInput* / *typeOutput*.
- **incompatibleInput**, **incompatibleOutput**: The component type which input/output cannot be connected to the output/input of this component.
- **designVariable**: The unique design variable naming which define a component. Design variables are used for component initialization.
- **designVariableInitialValue**: The design variable initial values. The datatype should be similar to designVariablesType.
- **designVariableLowerBound**, **designVariableUpperBound**: The design variable lower- and upper bound. The datatype should be similar to designVariablesType.
- **designVariablesType**: The design variable data type.
- **controlVariable**, **stateVariable**: The control- and state variables which determine the dynamic behavior of a component.
- **controlVariableLowerBound**, **controlVariableUpperBound**: Bounds for control variables.
- **stateVariableLowerBound**, **stateVariableUpperBound**: Bounds for state variables.
- **modellingPath**: The relative path of the modelling toolbox of the component.

By compiling a dataset for each component type, a comprehensive component library is generated. Each component within this library can be modeled as a block, which can take any of the following forms:

- **Source**: Only output ports, no input ports.
- **Sinks**: Only input ports, no output ports.
- **Converters**: Blocks which take similar input as output ports.
- **Transformers**: Blocks that take a number of input ports and convert them into different types of output ports.

Typical examples of source, sink, converter and transformer blocks in the context of mobility components are shown in Figure 4.

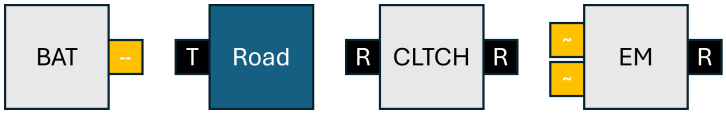


Figure 4. Source, Sink, Converter and Transformer block, shown from left to right.

2. Energy types

To identify viable system architectures for a given set of components and their respective input and output ports, specific energy types need to be considered (Figure 5). Defining energy types is crucial because it ensures that the interactions between components are physically meaningful and compatible. This helps in accurately modeling the flow of energy within the system.

The energy types illustrated in Figure 5 are those considered in this research, as the objective is to find a robust system architecture for a zero-emission vehicle. To extend the applicability of the System Architecture Topology Generator, additional specific energy types can be included. Examples of these are distinguishing between low- and high-voltage cabling, adding cooling water energy flow, or incorporating geometrical port types. It is important to note that this research does not consider the energy type Signal, also known as *Info* in literature [15].

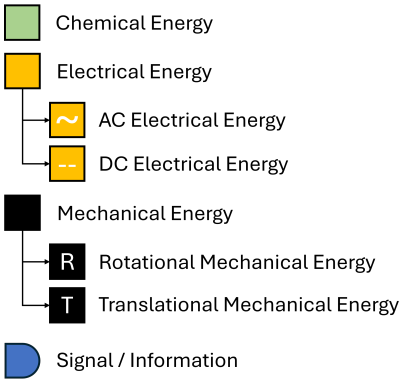


Figure 5. Typical examples of energy types related to zero-emission drive trains.

3. Define Case Specific Output Component(s)

Next, a system architecture output should be specified.
In this research, this output is referred to as the *output-component*, which follows a similar database pattern as the components in the component library. The output component is required to be of type *Sink*, only having input ports. The output component of the system can be explained as the (set of) component(s) that connects the internals of a system architecture to its environment. Multiple output-components, or sinks, are allowed, with a minimum of one. An example of a set of components placed within the system architecture environment and its connection to the (external) environment via an output-component is shown in Figure 6.

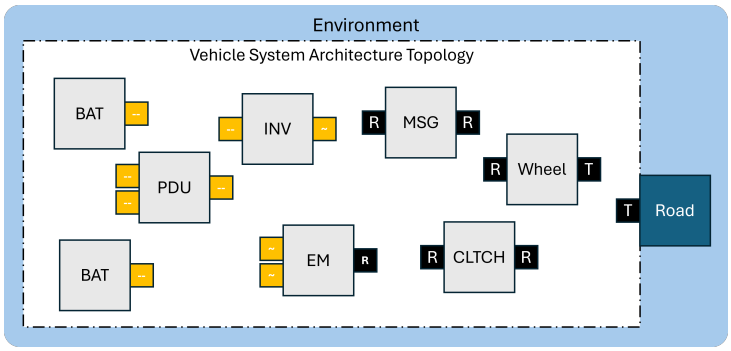


Figure 6. Visualization of selected components from the component library placed within the system architecture topology design space, and the system architecture connection to the environment through the output component (*Road*).

4. System Architecture Topology Generator

The port matching problem is now well-defined with a comprehensive component-library and defined output-component(s). The objective is to come up with a (set of) valid system architecture(s) based on specified component types and their respective input and output ports. For this, the System Architecture Topology Generator is developed, which is a tool which is able to generate a set of viable system architecture topologies from a specified component-library and output-components by solving a Port Matching Problem using Boolean Satisfiability Solvers. The subsequent sections will provide a detailed explanation of how the tool functions. First, some comprehensive requirements applicable to any valid system architecture are defined, compiled from discussions with experts within the industry.

- **Requirement 1:** all input and output ports of the various components in the system architecture are connected to the same input- and output port types (Port Matching).
- **Requirement 2:** Components cannot have self-loops, meaning an output port of a component cannot directly be connected to the component input port.
- **Requirement 3:** A single output port should be connected to a single similar input-port.
- **Requirement 4:** There should not be any open input or output ports in the final system architecture, except when explicitly stated (e.g. a Power Distribution Unit with a range of acceptable DC inputs).

An additional requirement has been included to prevent system loops. In the field of drive train design and optimization, system loops are uncommon and generally do not contribute to the functionality of the drive train. Therefore, system loops are prohibited:

- **Requirement 5:** System loops, where the output of a component feeds back into an input of a previously interconnected component, are not allowed within the system architecture.

In the literature, several port matching algorithms applicable to MDO applications have been established, one of which is KADMOS [16]. KADMOS, however, has trouble to identify multiple configurations or topologies applicable to drive train design due to the similarity in output types among multiple electro-mechanical components (e.g., torque) in a system architecture. Additionally, KADMOS is more specifically developed for automating the sequencing of analysis tools in MDAO applications, instead of generating system architectures. Other methods are also available in the literature. The method proposed by Herber et al. solely focuses on implementing a port matching algorithms applicable to Perfect Matching Graphs [17]. The method by Muenzer et al. lacks the possibilities to define port ranges (*Requirement 4*) [18] and the method by Helms and Shea is unable to find *all* unique solutions [19]. As a result, the SATG was developed to address these limitations in research.

As a first step, the Port Matching Problem (PMP) is converted into a Boolean Satisfiablity Problem (BSP) [18]. In this approach, a system architecture, with its respective input and output ports, is represented as an Adjacency Matrix (AM) or graph. The rows of the matrix denote the output ports of components, while the columns represent the input ports, resulting in a *n x n* boolean matrix for *n* components. Each index of the adjacency matrix is defined as a boolean variable using the open-source software *cpmpy*[1]. Next, the aforementioned requirements are integrated as constraints in the BSP, which restrict the feasible topology design space. An example of the self-loop constraint (**Requirement 2**) implemented in the BSP is given in Algorithm 1. For readability, boolean variables are converted to integer variables (0 = False, 1 = True) in the AM.

Algorithm 1 Port Matching Algorithm: Constraint 1 - Enforce No Self-Loops

Number of components in System Architecture Environment: n
$model \leftarrow cpmpy.Model()$
$X \leftarrow \text{IntVar}(\text{lb} = 0, \text{ub} = 1, \text{shape} = (n, n))$
for i in range(n) **do**
    $model \leftarrow model + (X[i, i] == 0)$
end for
return $model$

[1] https://cpmpy.readthedocs.io/en/latest/, accessed on 01/05/2024

Additional constraints are introduced to prevent isomorphic and disconnected graphs. Isomorphism refers to the phenomenon where two graphs have different edges but represent the same system architecture (similar functionality). Furthermore, it is crucial to ensure that all components are connected within the system architecture. This is enforced by ensuring only *weakly* connected graphs are a part of $\mathbf{G}_{feasible}$.

An example of the reduction in size of the topology design space, achieved by enforcing the aforementioned constraints for a set of 14 components, is shown in Table 1

Table 1. The reduction in topology design space size by enforcing constraints for a set of 14 components.

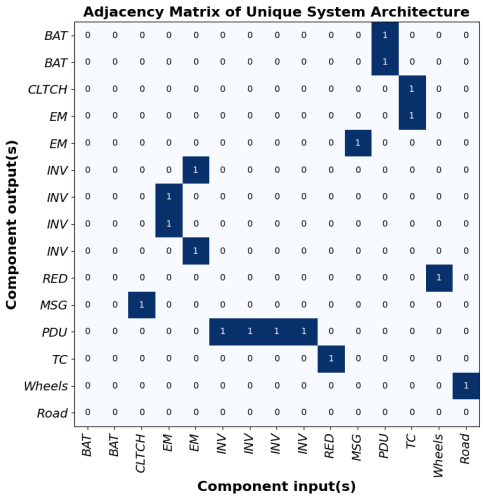
Description	**Value**
Total size of the system architecture topology design space	$2^{nxn} = 2^{14x14} = 1e59$
Number of weakly connected SA solution graphs	5,832
Number of weakly connected acyclic SA solution graphs	2,592
Number of unique, weakly connected, acyclic SA solution graphs	52

Utilizing the System Architecture Topology Generator reveals an important observation: the size of the feasible system architecture topology set, $O(Size(\mathbf{G}_{feasible}))$, scales proportionally with the product of the design space sizes associated with each energy type present within the system architecture. This observation provides a methodical approach to verifying the total number of feasible system architectures. By considering the design spaces for each energy type, the generator effectively generates and quantifies all potential configurations available within the system. This offers valuable insights into the scope and diversity of feasible system architectures.

$$O(Size(\mathbf{G}_{feasible})) = \prod_{i=0}^{n} O(Size(\mathbf{G}_{feasible,i})) \tag{1}$$

where n is the number of energy types considered in the analysis.

Eventually, the output of the System Architecture Topology Generator is a feasible set of directed, connected, acyclic, and unique system architecture topologies for the specified number and types of components within the system architecture. This is then stored as adjacency matrices and can easily be converted to graph networks. An example of a considered component set and output-component and found adjacency matrix (feasible system architecture topology) using the SATG is shown in Figure 7.



$$\begin{aligned}\mathbf{x}_{\text{components}} = [&\text{BAT}_0, \text{BAT}_1,\\ &\text{CLTCH}_0, \text{EM}_0,\\ &\text{EM}_1, \text{INV}_0,\\ &\text{INV}_1, \text{INV}_2,\\ &\text{INV}_3, \text{RED}_0,\\ &\text{MSG}_0, \text{PDU}_0,\\ &\text{TC}_0, \text{Wheels}_0]\\ \mathbf{x}_{\text{output}} = [&\text{Road}_0]\end{aligned} \tag{2}$$

Figure 7. Example of a SATG generated valid SA in the form of an adjacency matrix.

B. Architectural Design Space Evaluation in Conceptual Design

In order to compare and optimize a system architecture, it is essential to evaluate the generated architectures quantitatively. This requires evaluation tools which can produce coupling variables, objectives, or constraints used in the optimization process. The evaluation tools employed must be fast (low computational time) and stable to facilitate numerical optimization effectively. In this research, three analysis tools are considered: The Payload Mass Estimation Tool, The System Complexity Analysis Tool, and Optimal Control Solver. These tools are chosen because the goal of this research is to find a robust system architecture for a heavy-duty vehicle using the developed framework. The following sections will provide detailed insights into the workings of these tools.

The selected analysis tools align with the multidisciplinary nature of the optimization problem, while their outputs align with the multi-objective goal of vehicle drive train optimization, focusing on maximizing payload capacity and minimizing vehicle energy consumption (Wh / km).

1. The Payload Mass Estimation Tool

The typical starting point for the design of a heavy duty truck, is the maximum curb mass of the vehicle, determined by national highway operators. The maximum allowed curb mass for heavy-duty trucks in the Netherlands is set to be 44,000 kg [2]. The curb mass of the vehicle is defined as the sum of all individual masses, as given by Equation 3:

$$m_{curb} = m_{vehicle\,less\,drive\,train} + m_{drive\,train} + m_{payload} \tag{3}$$

In this equation, the mass of the vehicle less drive train, is assumed to be 22,500 kg (constant). These values are well-defined estimates. Given that the maximum curb mass is fixed, there is a clear trade-off between the drive train mass and the payload mass. One can either increase the drive train mass, thereby increasing the amount of propulsive energy on board and extending the vehicle's range at the cost of a lower payload mass, or prioritize payload mass, which would limit the range of the vehicle.

The drive train mass is computed using the following equation:

$$\begin{aligned} m_{drive\,train} &= m_{electric\,machine} + m_{energy\,storage} + m_{electrical\,converters} + \\ &\quad m_{mechanical\,converters} + m_{mechanical\,linkages} + m_{electrical\,linkages} \\ &= \sum_{i=0}^{n} m_{comp_i} \end{aligned} \tag{4}$$

Here, m_{comp_i} represents the mass of each individual component in the drive train.

2. The System Complexity Analysis Tool

Modern engineering projects are becoming increasingly complex, driven by increased demands on system performance (lower/no emissions, increased power). A review conducted by the US Government Accountability Office on 13 highly complex aerospace engineering projects since 2008 revealed an average increase of 55% in product development costs. According to Sinha and de Weck [20], these cost overruns can largely be attributed to the lack of effort in characterizing or quantifying associated complexity, which often results in underestimating project complexities and, consequently, setting inadequate project budgets. This shows the importance of quantifying and comparing system complexity in the early stages of the design process.

To quantify the complexity of a system architectures, various methods have been proposed in literature [20] [21]. In this research, the method of Sinha and De Weck is chosen due to its ability to capture both local and global aspects of system complexity. According to Sinha and de Weck, "System Complexity of

[2] https://uctransport.nl/diensten/maximum-gewichten-en-afmetingen/, accessed on 26/06/2024

technical systems depends on the heterogeneity and quantity of different elements and their connectivity patterns, and is a measurable system characteristic".

Sinha and de Weck quantify system complexity using the following formula:

$$\text{Structural Complexity:} \qquad C_{total} = C_1 + C_2 \gamma C_3 \tag{5}$$

Where,

- C_1: A measure of individual component complexity.
- C_2: A measure of the number and complexity of each pair-wise interaction. For this the various types of connections are defined as the types of energies as explained in subsubsection II.A.2.
- γ: A scaling parameter that adjusts the influence of the weighted interfaces on the overall complexity.
- C_3: A measure of the arrangement of interfaces.

Here it can be seen that C_1 and C_2 are local effects, whereas C_3 is a global effect, looking at the total system architecture as a whole.

C_1 is determined using Equation 6. Equation 6 demonstrates that a high Technology Readiness Level (TRL) corresponds to a low α_i, and consequently, a lower individual complexity value C_1.

$$C_1 = \sum_{i=1}^{n} \alpha_i, \qquad \text{where} \qquad \alpha_i = 5\left(\frac{TRL_{max} - TRL_i}{TRL_{max} - TRL_{min}}\right) \tag{6}$$

C_2 is computed using Equation 7:

$$C_2 = \sum_{i=1}^{n}\sum_{j=1}^{n} \beta_{ij} A_{ij} \qquad \text{where} \quad \beta_{ij} = f_{ij}\alpha_i\alpha_j \tag{7}$$

Table 2. Assumed energy type weighing factors (f_{ij}) used for computing C_2 value.

Energy type	$\mathbf{f}_{ij}$
Mechanical Rotation	0.5
Mechanical Translation	0.5
Electrical AC	0.7
Electrical DC	0.8
Chemical	0.2

This weighting value of each interface's complexity (β_{ij}) depends on the Technology Readiness Levels (TRLs) of the two connected components (α_i, α_j) and a specific weighting factor determined by the type of connection (edge). In this study, the connection weighing factor is dependent on the energy type. The assumed values of f_{ij} are detailed in Table 2.

Finally, the value of C_3 should be computed, which is a measure of topological complexity (global). C_3 is defined as the matrix energy of the adjacency matrix A, $E(A)$. Equation 8 is used to compute C_3.

$$C_3 = E(A) = \sum_{i=1}^{n} \sigma_i, \text{where } \sigma_i \text{ represents the } i^{th} \text{ singular value (found through SVD)} \tag{8}$$

This measure of topological complexity, C_3, increases when transitioning from centralized to more distributed architectures. Additionally, a trade-off can be made between topological complexity and component complexity (i.e., the first and third term in Equation 5).

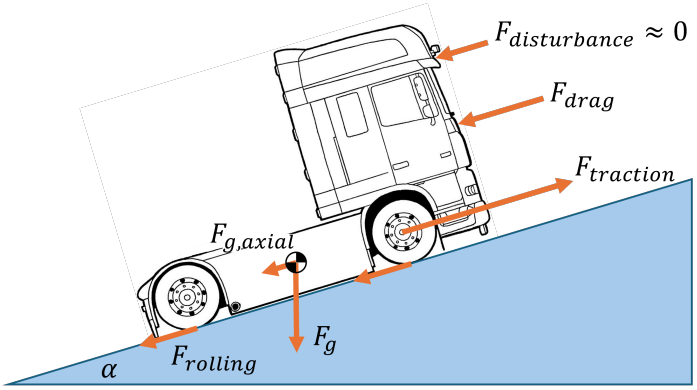
3. Optimal Control Solver

The final analysis tool used to quantitatively evaluate the performance of specific drive trains, is the Optimal Control Solver. This tool is crucial for system integrators to compare system architectures and

identify time-dependent optimal control strategies that yield the most favorable objective values. In this research, this tool is used to determine the optimal drive train control strategy for a heavy-duty truck, aiming to maximize driving range. Optimizing and vehicle driving range, and consequently vehicle efficiency, is crucial for system integrators to compete with current diesel-powered counterparts.

To quantitatively compare different system architectures, an automated analysis sequence is generated using the adjacency matrix, which represents the valid system architecture topology. For each component, the energy flow is computed based on the specified vehicle input conditions and operational state. This analysis begins with the output component (*Vehicle model*) and traces the energy flow through all components within the system architecture.

Vehicle model The vehicle model is used to determine the required torque and RPM levels at the wheels for a given set of input conditions. As mentioned in subsubsection II.A.3, the wheels serve as the connection between the drive train and the environment, making them the starting point of the analysis. The forces acting on the vehicle, along with their sign conventions, can be calculated using Equation 9.



$$
\begin{aligned}
F_{\text{traction}} &= \frac{T_{\text{wheels}}}{r_{\text{wheels}}} \\
&= gM_{\text{curb}}C_R\cos(\alpha) \quad + \\
&\quad gM_{\text{curb}}\sin(\alpha) \quad + \\
&\quad \frac{1}{2}\rho C_D(V - V_{\text{wind}})^2 A \quad + \\
&\quad \delta M_{\text{curb}} a
\end{aligned}
\tag{9}
$$

Figure 8. Considered forces acting on a truck (trailer not drawn here).

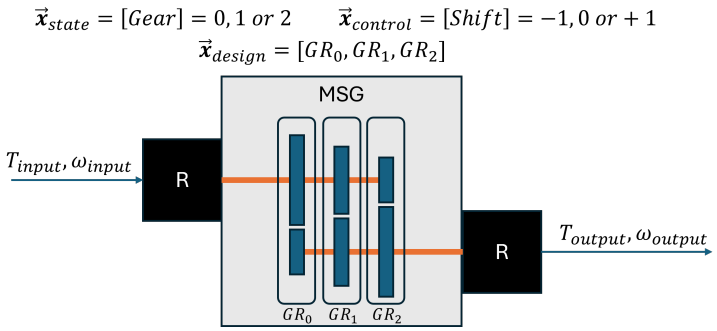
Component models The vehicle's driving range is determined by the performance of each individual component within the system architecture. To assess this, the states of individual components are computed by coupling various component analysis blocks according to the established adjacency matrix and specifying the component design and state variables ($\mathbf{x}_{design}$, $\mathbf{x}_{state}$). This approach enables the evaluation of each component configuration within any given topology based on the vehicle's operating conditions (V, α, V_{wind}, a). The method employs a backward calculation step to trace energy flows through all components, thereby eliminating the need for a convergence scheme.

To achieve an optimal system architecture design, the design- and control variables should also be optimized. An example illustrating the various design-, control- and state variables extracted from the component library for a multi-speed gearbox (MSG) is presented in Figure 9. An example of the low fidelity MSG model used in this research is shown inEquation 10. Low fidelity models were implemented due to their low computational cost (fast evaluation times) and sufficiently accurate results for conceptual design phases.

Multi-speed gearboxes have one input and one output shaft, with the ability to adjust gear engagement over time. Therefore, at any given moment, for a given input torque and RPM, there can be n output torque/RPM pairs for an n-speed gearbox. The implementation of gearboxes in zero-emission drive trains for heavy-duty trucks is becoming more common, as the required torque at low speed and relatively high top speed requires the RPM range of the propulsive electric machine to be adjusted over time [22].

Therefore, optimizing both $\mathbf{x}_{design}$ and $\mathbf{x}_{control}(t)$ is essential to determine the optimal efficiency of a drive train for specific operational profiles. This extra dimension of time is often overlooked in conceptual design studies but is crucial for accurately assessing drive train performance. By identifying

the optimal time-dependent control strategy ($\mathbf{x}^*_{control}(t)$) and resulting optimal drive train performance, system architecture performance can be fairly compared. This highlights the multi-dimensional nature of system architecture optimization in conceptual design, as also explained by Kabalan et al. [23].



$$T_i = \begin{cases} \frac{T_o}{GR_j \eta_{MSG}} & \text{if } T_o > 0.0, \\ \frac{T_o \eta_{MSG}}{GR_j} & \text{if } T_o < 0.0 \end{cases} \tag{10}$$

$$\omega_i = \omega_o GR_j \tag{11}$$

Figure 9. The design-, control-, and state variables of a 3-speed gearbox.

Topological Sorting To reduce the computational cost of evaluating the system architecture under specified operating conditions, a topological sorting algorithm was implemented. This algorithm rearranges the indices of the various components in the adjacency matrix to form an upper triangular matrix (UTM), allowing for a simple feed-forward of information without the need for an iterative scheme. The topological sorting algorithm, based on Manber's work [24], requires an acyclic directed graph (ADG) to function correctly. Since only ADGs are present in this research, the method is applicable.

By applying this sorting algorithm, the computational time for evaluating the total system architecture performance, given the input conditions and design- and state variables, is reduced by approximately 75%. The topologically sorted adjacency matrix is shown in Figure 10.

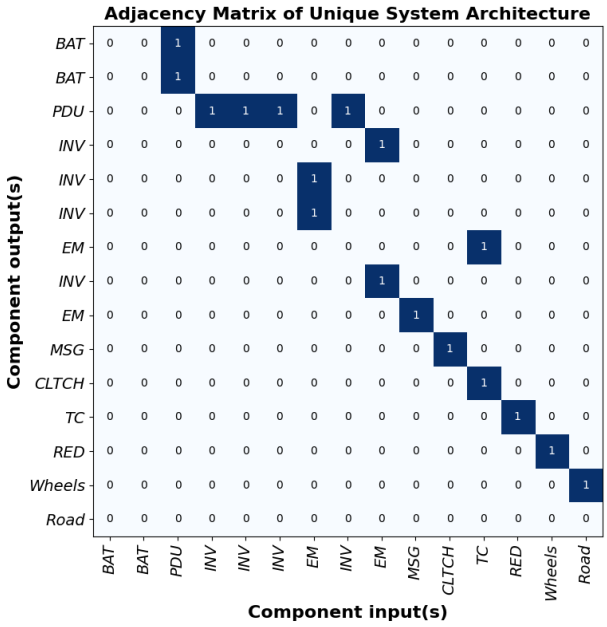


Figure 10. The sorted upper-triangular Adjacency Matrix (AM) generated from Figure 7.

Drive Cycle To quantify the performance of the drive train for a given topology, component configuration, and control strategy, a drive cycle must be considered. For this research, a modified version of the HHDDT Transient drive cycle is chosen [3] (see Figure 11). This drive cycle has been adjusted to allow for a maximum velocity of 90 km/h and acceleration rates within the typical range observed in heavy-duty trucks [25]. the HHDDT drive cycle was selected due to the significant transient responses across various velocity ranges, ensuring a robust and versatile solution. Additionally, it reflects the typical operating conditions of heavy-duty vehicles commonly found in the Netherlands ($\alpha = 0$ deg).

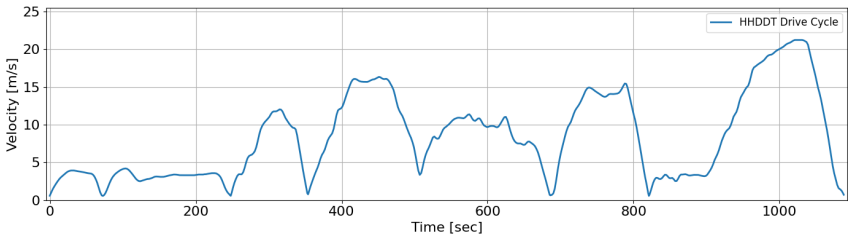


Figure 11. HHDDT Transient drive cycle.

Filling the state-time performance matrix The objective of the optimal control solver is to find the optimal time-dependent control strategy, for a given component design vector (x_{design}) and drive cycle, which minimizes ΔSoC. The *Optimal Control Solver* works by first computing the objective value (ΔSoC_i) for every time step ($t = i$) in the drive cycle and sampled $\mathbf{x}_{state}$. Here, a full-factorial sampling (or discretization) of the state variable space is performed. At each time step, the input conditions are aggregated into the input vector $\mathbf{u}(t = i)$. Consider the system architecture topology as illustrated in Figure 12. From the component library, two state variables are obtained: the Torque Split Ratio (*TSR*) and selected Gear (*gear*).

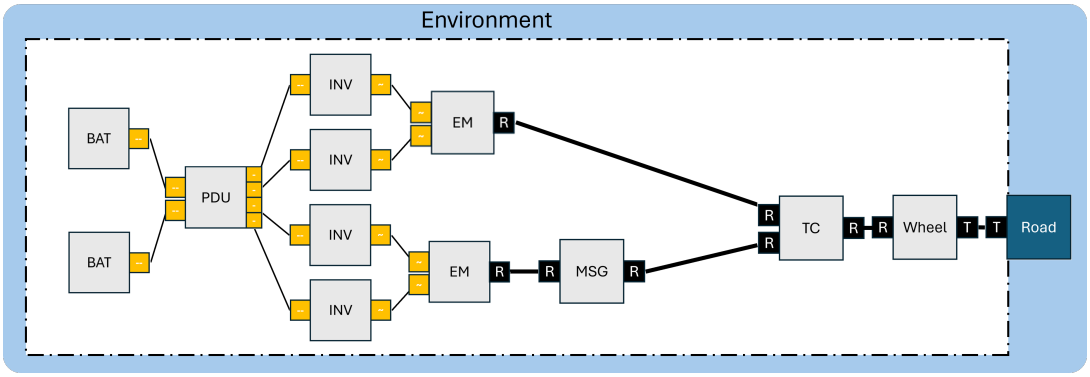


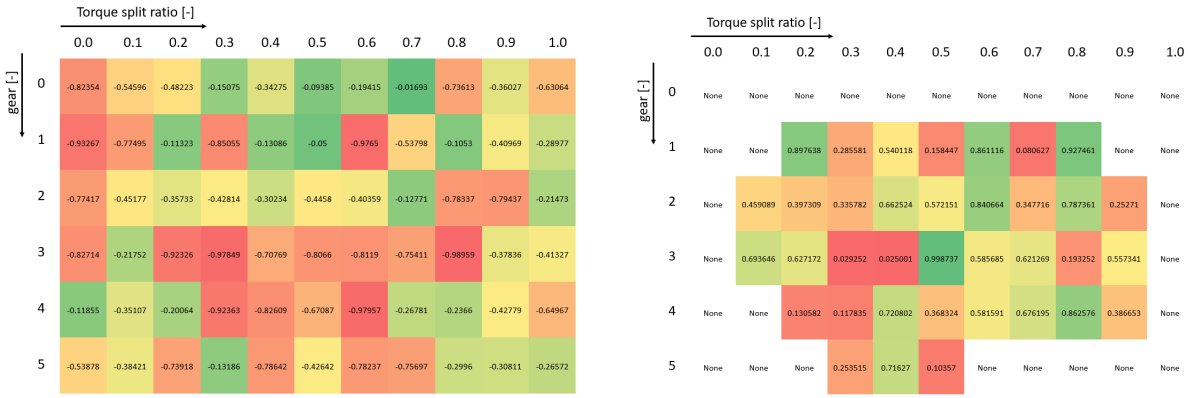
Figure 12. A system architecture topology with two state variables (Torque Split Ratio and Gear).

Table 3. The state- and control variable information (name, lower- and upper bound & variable type) for the system architecture topology as seen in Figure 12.

	Variable names	**Variable lower bound**	**Variable upper bound**	**Variable type**
$\overrightarrow{x}_{control}$	[shift, ΔTSR]	[-1, -1.0]	[1, 1.0]	[*int*, *float*]
$\overrightarrow{x}_{state}$	[gear, TSR]	[0, 0.0]	[5, 1.0]	[*int*, *float*]

The objective value of ΔSoC_i is computed for the given input boundary conditions ($\mathbf{u}(t = i)$) and sampled $\mathbf{x}_{state}$. The output of the analysis for two time steps in the HHDDT drive cycle is shown in Figure 13.

[3] https://www.transportpolicy.net/standard/us-heavy-duty-hhddt/, accessed on 26/06/2026



(a) $\mathbf{u}(t = 600) = [11.75, 1, 0, -0.01]$ (b) $\mathbf{u}(t = 384) = [18, 5, 0, 0]$

Figure 13. Values of ΔSoC_i for the discretized state variable space at various time steps (and $\mathbf{u}(t)$).

Figure 13b shows that the state space can be discontinuous for a given input vector $\mathbf{u}(t)$. This occurs when some components exceed their nominal operating ranges, rendering certain states of the drive train infeasible (*None* states).

By applying this procedure to a n-dimensional state variable vector, it is possible to compute the state-time performance matrix, from now on referred to as $\mathbf{J}^*$ matrix. For this, the n-dimensional state variable vector ($\mathbf{x}_{state}$) is discretized between its upper- and lower-bound given the state variable type, flattened and converted into a row vector.

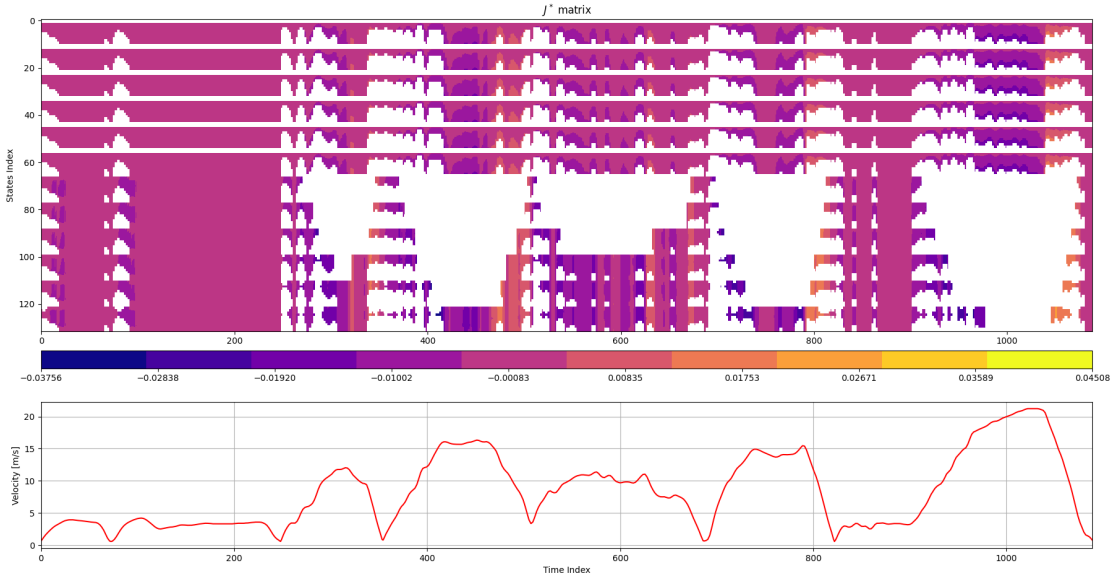


Figure 14. An example of a state-time performance matrix ($\mathbf{J}^*$). Invalid states are shown in white. Negative values of ΔSoC_i indicate battery discharge, whereas positive values signify regenerative braking (battery charging).

Finding the optimal control strategy To find the optimal time-dependent control strategy that minimizes the total ΔSoC over the drive cycle, a shortest path algorithm was implemented.

First, a Directed Weighted Graph (DWG (G)) was constructed using the $\mathbf{J}^*$ matrix. Here, all states are represented by nodes (vertices V), and edges (E) are created between states at $t = t_i$ and only "reachable" states at $t = t_{i+1}$. The total set of reachable states $\mathbf{K}$ is determined by $\mathbf{x}_{control}$. The weights of these edges correspond to the value of ΔSoC_{i+1} at those reachable states. By applying a shortest path algorithm to the generated DWG, the path that results in the minimal ΔSoC can be found. In this research, Dijkstra's

algorithm was selected for its computational efficiency and suitability to the use case. The total ΔSoC is computed using Equation 12.

$$\Delta SoC = \sum_{i=0}^{n} \Delta SoC_i \qquad \text{where n is the number of timesteps} \tag{12}$$

A mathematical formulation for the above explained generated Directed Weighted Graph (DWG) is provided in Equation 13:

$$\begin{aligned} &G = (V, E), \quad V = \bigcup_{i=0}^{n} V_i, \quad E = \bigcup_{i=0}^{n-1} E_i \\ &V_i = \{v \mid v \text{ is a node at time } t_i\} \\ &E_i = \{(u, v) \mid u \in V_i, v \in V_{i+1}, v \in K, \ \text{weight}(u, v) = \Delta\text{SoC}(v)\} \end{aligned} \tag{13}$$

In this research, both positive and negative edge weights (representing battery charging and discharging) are considered, making Dijkstra's algorithm unsuitable [26]. While the Bellman-Ford method can handle both types of weights, its much higher time complexity renders it impractical. Therefore, a method was chosen that shifts all negative weights by the value of the most negative edge weight. By applying Dijkstra's algorithm to the adjusted weights and then shifting all values back to their original values, the minimal ΔSoC and optimal control strategy could be determined. This method is valid because the graph is acyclic, and the number of edges is always constant (equal to the number of time steps).

Surrogate Models for Computing the State-Time Performance Matrix ($\mathbf{J}^*$) The *Optimal Control Solver* requires a filled $\mathbf{J}^*$ matrix, with dimensions m x n, where m is the number of possible states (dependent on $\mathbf{x}_{state}$) and n is the number of time steps in the drive cycle. For complex system architectures with an increased number of valid states, this matrix can grow significantly large, leading to high computational costs. To mitigate this issue, surrogate models are implemented to fill the $\mathbf{J}^*$ matrix. By using surrogate models, users can explore the design space with reduced computational cost at the cost of a slight decrease in accuracy. The application of surrogate models is common in system architecture optimization during conceptual design phases, where exhaustive exploration of the entire design space is impractical and unnecessary [27].

Due to the highly discontinuous nature of the state-time space (see Figure 14), where the distinction between valid and invalid states is determined by a penalty value assigned to invalid states, and valid states have very low order of magnitudes, training accurate surrogate models becomes challenging. Exploiting the fact that invalid states provide no useful information, a Random Forest (RF) classification surrogate model ($n_{estimators}$ = 500, $n_{training}$ = 1000) was trained to classify *valid* and *invalid* states with an average accuracy of 98.6%. The training set was sampled using Latin Hypercube Sampling (LHS).

By then fitting a RF regression surrogate model ($n_{estimators}$ = 500) on the remaining *valid* states, the $\mathbf{J}^*$ matrix can be accurately filled. The valid state-time performance space remains discontinuous per time step, making it unsuitable for Latin Hypercube Sampling (LHS). Instead, K-th sampling was applied, using a training set ratio of 6%. For each time step, density-based sampling was performed with a minimum of 4 training points. This approach resulted in an average error of -2.3%, which was deemed sufficient for this research.

Applying surrogate models to fill the state-time performance matrix ($\mathbf{J}^*$) resulted in a reduction of computational time by approximately 71%.

C. Uncertainty Propagation in Coupled MDA

The objective of this research is to integrate uncertainty-based design optimization into a system architecture exploration and optimization framework, aiming to find a robust multi-objective optimal design. This requires the quantification and propagation of uncertainty throughout the analysis.

Research in this area is limited, with existing studies mainly focusing on either simplistic first-order methods or computationally intensive approaches like Monte Carlo Simulation (MCS). Given the substantial computational burden of MCS, which is impractical for the already large multi-dimensional design space in conceptual design, this method will not be used. Instead, a simpler and more generic uncertainty-propagation method will be employed, derived from the method proposed by Gu et al. [28].

This method, known as the "Worst-Case Error Propagation" method, assumes a "deterministic" uncertainty measure or error and computes the worst-case error propagation. "Deterministic" in this context means that when the analysis is run multiple times, the error associated with that analysis at specific operating points remains constant, although it may vary across different operating conditions.

1. Uncertainty sources in conceptual design phases

Uncertainty can be categorized into aleatory and epistemic uncertainty [7]. Aleatory uncertainty can be explained as the uncertainty associated with the inherent variability of a physical system considered, which is irreducible and hence not considered in this research. Epistemic uncertainty arises from insufficient knowledge or data and can therefore be reduced as more information becomes available. In conceptual design phases for engineering systems, epistemic uncertainty arises from the variability or uncertainty in mission requirements (*model input uncertainty*) and the uncertainty associated with simulation-based approaches (*model uncertainty* and *model error*). Given that most models used are low-fidelity and produce deterministic outputs, the model error, as discussed by Yao et al. [7], is considered negligible (i.e., zero). Therefore, for this research, only model uncertainty remains relevant.

In a bottom-up approach, where system integrators acquire components from suppliers based on specific requirements, there is no uncertainty in the design variables ($s_{\mathbf{x_{design}}} = 0$) or the component mass. Consequently, the payload mass is deterministic. The uncertainties that are however considered are *model input uncertainty*, $s_{\mathbf{u(t)}}$, and *component model uncertainty* stemming from model simplifications and assumptions, s_{model_i}.

The assumptions from the research by Gu et al., which are relevant to this study, are outlined below:

- The bias error associated with a given simulation tool varies as a function of the tool's inputs
- Given the same input, the simulation tool will give the same output and consequently, the same bias error in the corresponding output (deterministic).
- If the tool input changes, in general, the bias error will change. However, if one is performing a sensitivity analysis, where the change in the inputs is very small then the bias error can assumed to be fixed.
- During a coupled system analysis, the tool used in each discipline remains the same.
- The resolution of each simulation tool is high enough so that its effect on the convergence of the coupled system analysis is negligible.
- The tool uncertainty information (i.e. bias error) is provided or assumed by the framework user.

2. Uncertainty Propagation

To quantify the uncertainty in the analysis output, which is of interest to the user and optimization framework, an uncertainty propagation method is necessary. Here, a widely accepted approach that relies on model sensitivities, as detailed by Ku et al. [29], will be used. This method, which assumes independent and uncorrelated variables, is based on a variance formula. The standard formula has been adjusted to incorporate model uncertainty, as illustrated in Equation 14.

$$s_{f,total}|_{(x_i,\ldots,x_n)} = \sqrt{s^2_{propagated}|_{(x_i,\ldots,x_n)} + s^2_{model,f}|_{(x_i,\ldots,x_n)}} \tag{14}$$

$$s_{f,total}|_{(x_i,\ldots,x_n)} = \sqrt{\left(\frac{\partial f}{\partial x_i}^2\right) s_{x_i}^2 + \left(\frac{\partial f}{\partial x_{i+1}}^2\right) s_{x_{i+1}}^2 + \ldots \left(\frac{\partial f}{\partial x_n}\right)^2 s_{x_n}^2 + s^2_{model,f}|_{(x_i,\ldots,x_n)}} \tag{15}$$

Here, the propagated uncertainty through function $f(x_i, ...x_n)$ is dependent on the first-order sensitivity of the function at that operating point, the variation (e.g. the standard deviation) of that variable at that point and the inherent model uncertainty at that operating point.

When coupling the various disciplines and propagating the input uncertainty $s_{\mathbf{u}(t)}$ through the various disciplines (m) and taking into account s_{model_m}, the total propagated output uncertainty, $s_{\Delta SoC}$, can be computed. As can be seen in Equation 16, the propagated uncertainty $\mathbf{s}_{\mathbf{y}_m}$ is in vector form, meaning that this method works for multiple inputs and outputs.

$$\mathbf{s}_{\mathbf{y}_m} = \sqrt{\left(\frac{\partial f_m}{\partial \mathbf{y}_{m-1}} \mathbf{s}_{\mathbf{y}_{m-1}}\right)^2 + s^2_{f_m(\mathbf{y}_{m-1})}} \tag{16}$$

3. Input- and Model-Uncertainty Quantification

The next step involves quantifying input- and model uncertainty. In this research, probability theory is used, which assumes stochastic or random continuous variables [7]. For continuous random variables, we can describe their behavior using a probability density function (PDF).

Model Input Uncertainty The model input uncertainty in this research stems from the uncertainty related to the operating conditions of the vehicle, hence the drive cycle. Due to a lack of quantitative uncertainty data, it was assumed that the uncertainty in the drive cycle velocity input, $s_{u_1} = s_V$, follows a normal distribution with mean zero and variance of 0.05, $\mathcal{N}(0, 0.05)$. As the vehicle acceleration is a dependent variable ($a = \frac{\delta v}{\delta t}$), $s_{u_4} = s_a = 0$. Figure 15 illustrates a visualization of the HHDDT drive cycle with the added uncertainty as noise.

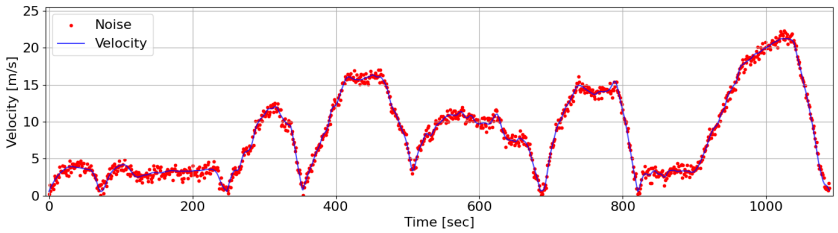


Figure 15. HHDDT drive cycle with added Gaussian noise $\mathcal{N}(0, 0.5)$.

Model Uncertainty Model uncertainty is related to the uncertainty of the underlying model to accurately quantify and predict real world physics. Due to the absence of detailed uncertainty quantification for each component type in this study, a simplifying assumption has been made: model uncertainty across all components follows a uniform distribution ranging between 0.175 and 0.225. The distributions of input- and model uncertainty are depicted in Figure 16.

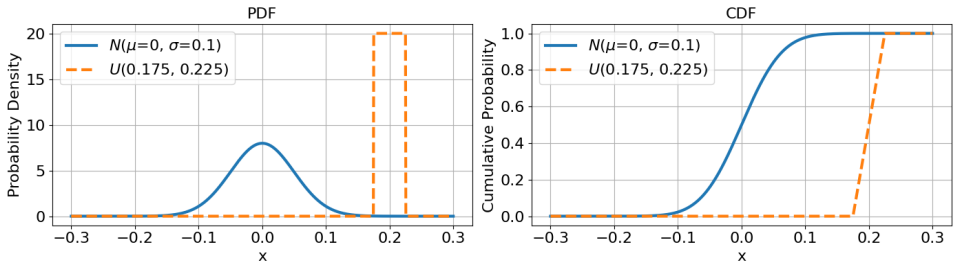


Figure 16. PDF and CDF of the assumed model input uncertainty $\mathcal{N}(0,0.05)$ and model uncertainty $\mathcal{U}(0.175, 0.225)$.

The cumulative propagated output uncertainty can be computed using Equation 17. This provides a quantitative output uncertainty level for the found optimal control strategy, given some system architecture topology and $\mathbf{x}_{design}$.

$$\text{Total } s^*_{\Delta SoC} = \sum_{j=1}^{n} (s^*_{\Delta SoC})_j \tag{17}$$

D. Uncertainty-Based System Architecture Optimization

Several methods proposed in literature implement this uncertainty quantification in an MDAO framework [6, 30–32]. Typically, two primary methodologies are utilized: Reliability-Based Design Optimization (RBDO) and Robust Design Optimization (RDO). RBDO emphasizes constraint satisfaction under uncertainty, while RDO focuses on identifying local or global optima that exhibit minimal sensitivity of the objective to input uncertainties. These approaches can be integrated together to form a Reliability-Based Robust Design Optimization (RBRDO) framework. For the case study described below, constraint satisfaction posed no issue due to the way the constraints were formulated (design variable constraints without uncertainty), hence an RDO framework was implemented.

1. Robust System Architecture Optimization and XDSM

The RBRDO problem formulation used in this research is presented in Equation 18. Here, the parameter θ is a weighting factor that determines the balance between optimizing the objective function and ensuring robust design (minimizing uncertainty).

$$\begin{aligned}
\text{find} \quad & \mathbf{x} = [x_1, x_2, ..., x_n]^T \\
\text{min} \quad & f_1^R(\mathbf{x}) = \theta \cdot f_1 + (1-\theta) \cdot \Delta f_1 \\
\text{max} \quad & f_2^R(\mathbf{x}) = M_{payload} \\
\text{s.t.} \quad & g_k^R(\mathbf{x}) = g_k \\
& h_l^R(\mathbf{x}) = h_l \\
& \mathbf{x}_i^L \leq \mathbf{x}_i \leq \mathbf{x}_i^U, \quad i = 1, 2, ..., n \\
\text{where} \quad & f_1 = f(\mathbf{x}) = \Delta SoC^* \quad \text{and} \quad g = g_k(\mathbf{x}) \quad \text{and} \quad h = h_l(\mathbf{x}) \\
& \Delta f_1 = \text{Total } s^*_{\Delta SoC} = \sum_j \left(s^*_{\Delta SoC}\right)_j \qquad \text{where j is the number of time steps} \\
& \text{where } s^*_{\Delta SoC} \text{ is the uncertainty measured at the output analysis m using:} \\
& \vec{s}_{y_i} = \sqrt{\left(\frac{\partial \vec{f}_i}{\partial \vec{y}_{i-1}} \vec{s}_{y_{i-1}}\right)^2 + s^2_{f_i(\vec{y}_{i-1})}} \qquad \text{propagated from i = 0 to i = m}
\end{aligned} \tag{18}$$

Alternatively, Δf_1 can be treated as a Quantity of Interest (QOI). In this method, the optimizer seeks a deterministic optimum, and the integrator subsequently determines the acceptable uncertainty level within predefined bounds. While this approach may lack rigorous mathematical substantiation, it offers flexibility by allowing users to decide on an acceptable risk level at a later stage.

This results in the following simplified XDSM:

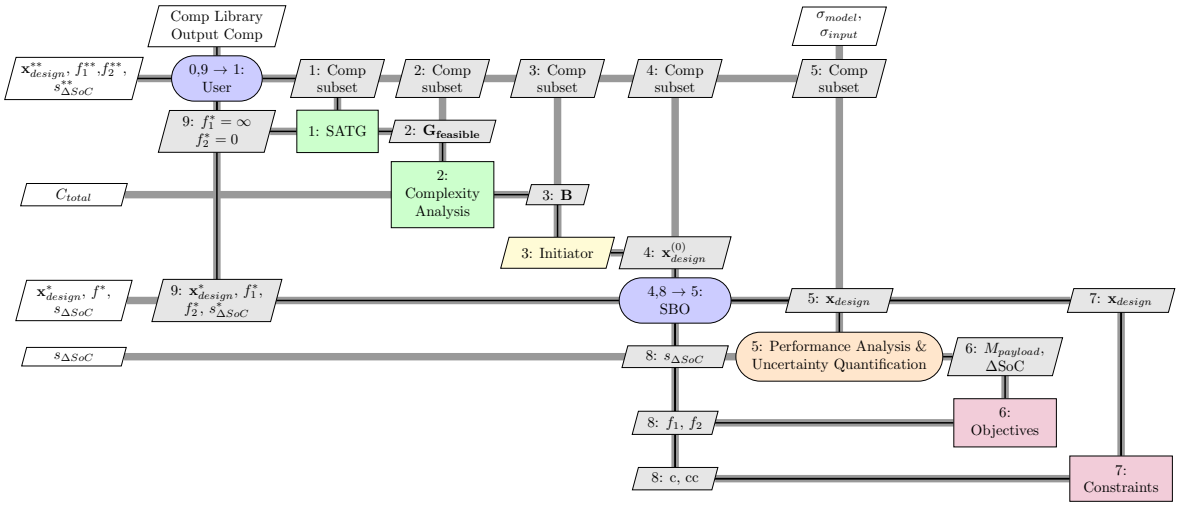


Figure 17. XDSM of the uncertainty-based system architecture optimization (UBSAO) framework using uncertainty as a QOI.

In this research, a pseudo-nested optimization strategy is implemented, where each level in the optimization process uses a different tailored optimization algorithm. Santiago et al. propose using Genetic Algorithms (GA) for the outer-loop design variable mutations in a nested strategy to explore the entire architectural design space [33]. However, this approach is not suitable for applications with expensive inner-loop optimization steps, as is the case in this study. Therefore, the outer-loop analyzer in this study is selected to be the *User*, who selects a subset of components. For this research, exploitation was preferred over exploration, for which nested optimization strategies work best according to the work done by Santiago et al. Future research should explore implementing such outer-loop algorithms or different optimization strategies (e.g., Global Optimization) to automate the entire system architecture optimization and increase both exploration and computational efficiency.

In the XDSM the following steps can be seen:

1. The framework begins by loading the component library and output component into the framework. The *User* then selects a subset of the component library, which is passed to the SATG.
2. The SATG solves a Boolean Satisfiability Problem (BSP) to generate a set of viable system architecture graphs, denoted as $\mathbf{G}_{feasible}$.
3. The resulting set of system architecture graphs can range from a few valid architectures to a large set, which is undesired from a computational cost perspective. Therefore, a subset of system architectures is chosen based on the system complexity value. In this research, this subset is chosen by picking the system architectures with the highest, lowest and average structural complexity values. User-selected SA's can also be added to subset. This subset is referred to as B.
4. A feasible starting position. $\mathbf{x}^{(0)}_{design}$, is found using an *Initiator*. This approach is chosen because the optimization function evaluations are computationally very expensive, making a feasible starting position more practical.
5. For each graph $\mathbf{B}_i$ in the subset $\mathbf{B}$, a mixed-integer non-linear optimization problem is formulated. A gradient-free optimizer, such as genetic-, evolutionary- or surrogate-based optimization algorithm, is implemented suitable for the mixed-integer nature of the optimization problem. The optimizer adjusts the design variables to *maximize* $M_{payload}$ and *minimize* ΔSoC, subject to the specified constraints and bounds. The uncertainty associated to the objective, $s^*_{\Delta SoC}$, is stored as a QOI.

2. Design Space Characteristics and Selected Optimization Algorithm

System Architecture Optimization (SAO) problems have a highly dimensional, mixed-integer, non-linear, discontinuous design space. This requires a gradient-free optimization algorithm, which is able to

efficiently explore (low number of function evaluations as computing $\mathbf{J}^*$ is computationally expensive) such design spaces.

For this purpose, a Surrogate-Based Optimization (SBO) algorithm, as implemented in SBArchOpt, was chosen [34] [35]. SBArchOpt utilizes Gaussian Process, or Kriging models [36], to map design variables to computed objectives. It then identifies points with the highest probability of improvement (Expected Improvement) through design space exploration. Surrogate-based optimization algorithms strike a balance between exploration and computational speed, typically requiring fewer function evaluations to find local minima, suitable for this research. Future work should investigate the use of surrogates that account for objective value uncertainty, such as Sparse Gaussian Processes [37].

III. Case Study Description

In order to evaluate the framework and compare the outcome to its intended purpose, a case study was performed applicable to the design and optimization of a heavy-duty vehicle drive train subject to input- and model uncertainty. Here, two component sets, extracted from the component library, are evaluated and optimized using the method as shown in Figure 17.

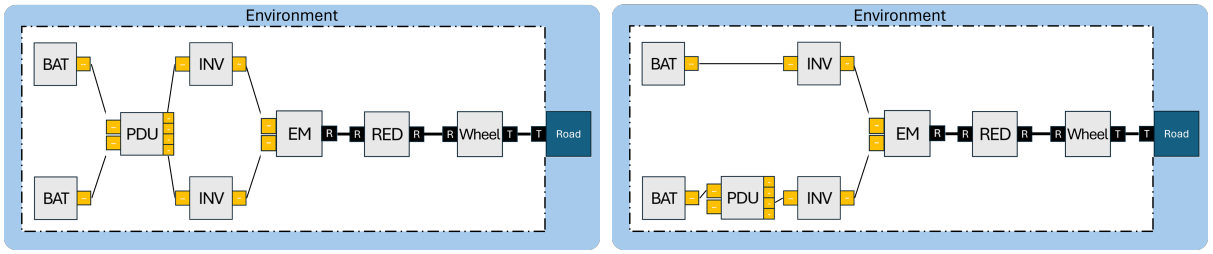
The two component sets chosen for this, are shown in Equation 19 and Equation 20. These are chosen in a way that they resemble both a simple and a more complex drive train [22]. This way, the effect of drive train complexity on its performance can be evaluated in terms of $M_{payload}$ and drive train efficiency (ΔSoC) for a given drive cycle.

$$\mathbf{x}_{components_1} = [BAT_0, BAT_1, EM_0, INV_0, INV_1, RED_0, PDU_0, Wheels_0] \tag{19}$$

$$\mathbf{x}_{components_2} = [BAT_0, BAT_1, CLTCH_0, EM_0, EM_1, INV_0, INV_1, INV_2, INV_3, RED_0, MSG_0, PDU_0, TC_0, Wheels_0] \tag{20}$$

A. Selecting A (Sub)set Of System Architecture Topologies For Optimization

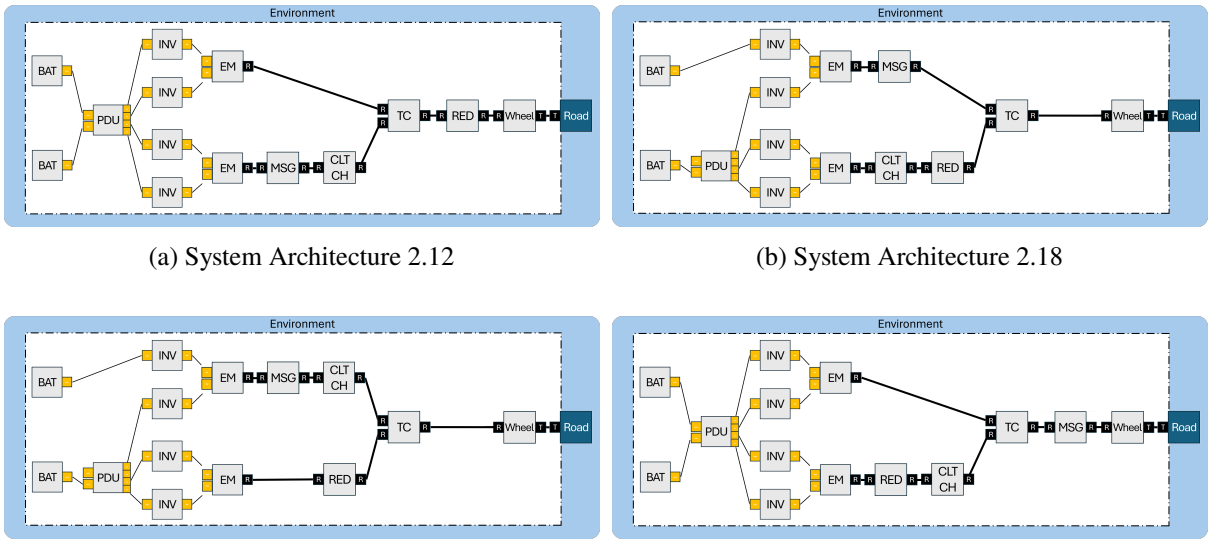
Using the proposed method as shown in Figure 17, for component set 1 (Equation 19), the SATG obtained two viable system architecture topologies. Therefore, no selection was required from a computational cost perspective. Figure 18 shows the found system architecture topologies. For component set 2 (Equation 20), the SATG obtained 52 viable system architecture topologies. Using the method as explained previously, the system architecture topologies as seen in Figure 19 were selected for further optimization. The structural complexity values of the identified system architecture topologies are compared in Figure 20.



(a) System Architecture 1.1

(b) System Architecture 1.2

Figure 18. SATG obtained viable system architectures for component set 1.



(a) System Architecture 2.12

(b) System Architecture 2.18

(c) System Architecture 2.19

(d) System Architecture 2.43

Figure 19. SATG obtained and selected viable system architectures for component set 2

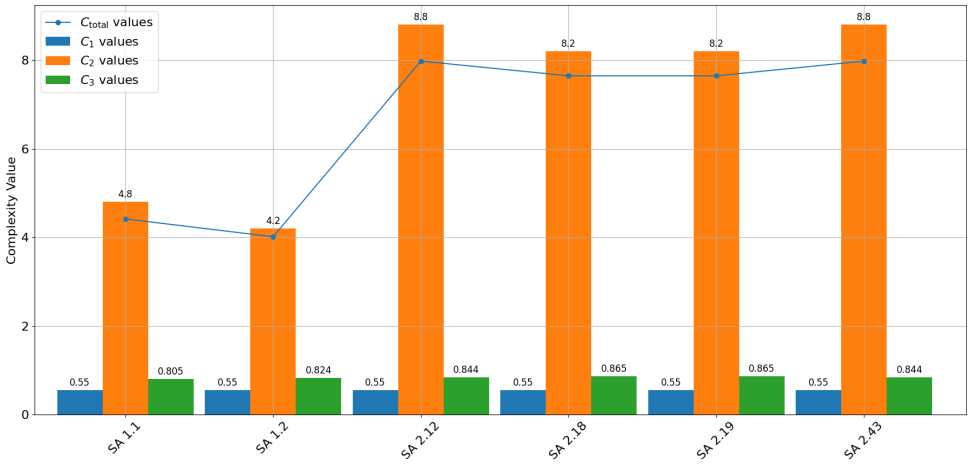


Figure 20. Structural complexity decomposition for the selected valid system architecture topologies generated by the SATG for $\mathbf{x}_{components_1}$ and $\mathbf{x}_{components_2}$.

Figure 20 shows that the overall system complexity values (C_{total}) of the topologies found for component set 1 are significantly lower than that for component set 2. This can be explained by the lower C_2 values, which are determined by the number and complexity of component connections (interfaces). It can also be seen that all found SA's have the same value of C_1. This can be explained by the fact that only the PDU has a TRL less than TRL_{max}.

Constraints (In)equality constraints can be added to the optimization problem to limit the feasible design space. These constraints should be formulated in the form where $h_i(x) = 0$ and $g_j(x) < 0$. The framework can easily extract constraints from a separate XML file, provided by the user. For $\mathbf{x}_{components_1}$, the implemented equality constraints are:

$$\mathbf{h_0}(\mathbf{x}) = InverterType_0 - InverterType_1 \tag{21}$$

For $\mathbf{x}_{components_2}$, more elaborate (in)equality constraints are implemented. They are shown in Equation 22 and Equation 23.

$$\mathbf{h}_i(\mathbf{x}) = InverterType_i - InverterType_{i+1} \tag{22}$$

$$\mathbf{g}_j(\mathbf{x}) = MSGgearRatio_j - MSGgearRatio_{j-1} \tag{23}$$

IV. Result & Discussion

This section will go into the obtained results from performing the previously explained case study focused on the uncertainty-based system architecture optimization of a simple and complex drive train. Alongside the optimization outcomes, we will discuss relevant aspects such as dynamic drive train behavior, uncertainty quantification, and other pertinent information. These findings will allow us to draw conclusions about the impact of incorporating uncertainty into system architecture optimization.

A. Pareto Front Of The Uncertainty-Based System Architecture Optimization

Using the surrogate-based optimizer, the Pareto fronts as seen in Figure 21 were obtained. Here, the optimization problem formulation as shown in Equation 18 was used ($\theta = 1$), combined with the previously specified (in)equality constraints. The objective value f_1, or ΔSoC, was converted to vehicle efficiency ($\eta_{vehicle}$ [Wh/km]), which is a measure independent of battery size. This allows for a comparison based solely on the combined mechanical- and electrical efficiency, and a more fair comparison between system architecture optimal design points.

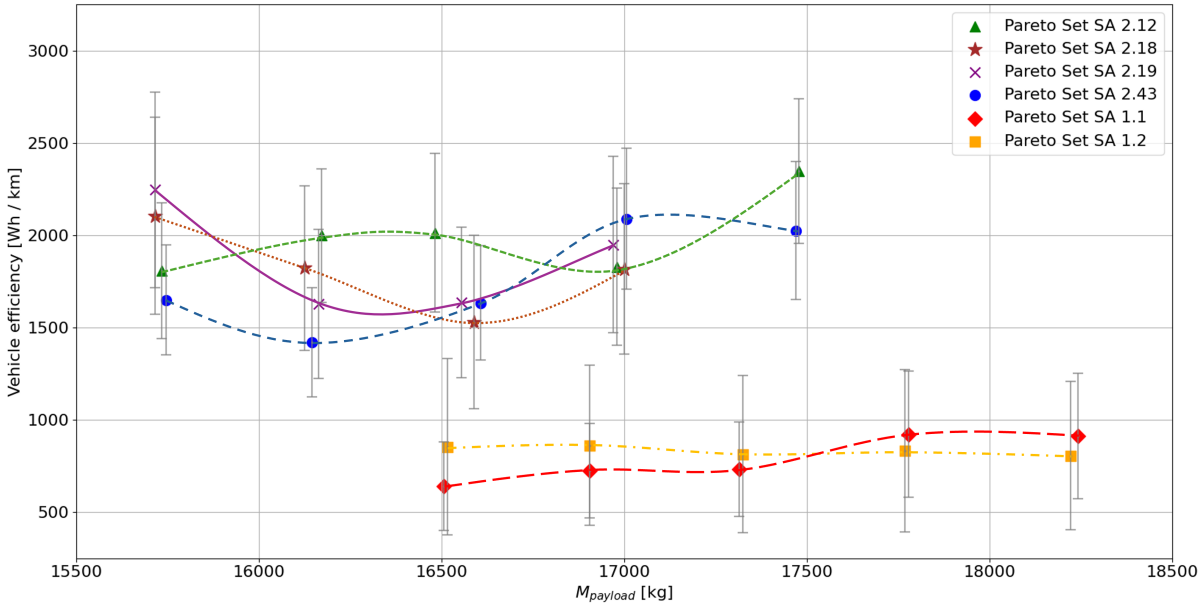


Figure 21. Pareto front for the selected system architecture topologies, subject to model- and input uncertainty ($s_{model} = \pm\mathcal{U}[0.1775, 0.225]$, $s_{input} = \mathcal{N}(0, 0.2)$).

From Figure 21 the following conclusions can be drawn:

- The obtained Pareto front is discrete on the x-axis because $f_2 = M_{payload}$ is computed by summing the discrete masses of individual components.
- The output of the optimization study reveals two distinct clusters looking at achievable payload masses and vehicle efficiencies. For component set 1 (SA 1.1 & SA 1.2), achievable $M_{payload}$ ranges from 16,500 to 18,250 kg with vehicle efficiencies between 620 and 900 Wh / km are obtained. The system architecture topologies for component set 2 achieve lower vehicle efficiencies (higher Wh / km) (range between 1450 and 2300 Wh /km) with an achievable $M_{payload}$ ranging between 15,750 and 17,500 kg. The uncertainty in vehicle efficiency is visualized in the figure using uncertainty bars, which represent worst- and best-case estimates.

- For the obtained system architecture topologies using component set 1, the found Pareto points have a significantly higher vehicle efficiency (lower Wh / km) compared to the selected system architecture topologies for component set 2. This can be explained by the fact that component set 2 consists of more components, each of which contributes to increased *inefficiency* in the vehicle drive train. The potential benefits of operating components at their optimal operating points are outweighed by the inefficiencies introduced by the additional components.
- Figure 21 shows that the Pareto points for the system architecture topologies from component set 1 have a significantly higher $M_{payload}$ compared to the Pareto points for the system architecture topologies of component set 2. This also can be explained by the increased number of components, which increases the weight of $M_{drive\,train}$, thereby decreasing $M_{payload}$.
- The found Pareto fronts are almost never strictly convex. This makes it difficult for system integrators to perform trade-offs.
- When comparing the computed uncertainty quantification for all Pareto points, it can be seen that per system architecture, the found Pareto fronts never have a constant uncertainty value associated with them. Also from system architecture to system architecture, the quantified uncertainty associated with every Pareto point is different. This allows for trade-offs, where indeed performance can be exchanged for a lower uncertainty level.

From Figure 21, it is clear that there is no single best solution, where one system architecture topology outperforms every other system architecture in terms of $M_{payload}$ and $\eta_{vehicle}$. This is in line with the comments made on the system architecture design space, which is very large and highly case specific. Therefore, system integrators must clearly formulate their requirements for $M_{payload}$ and Vehicle efficiency [Wh / km] to determine the optimal system architecture for their application.

The highly non-linear and mixed-integer design space is easily explored by the surrogate-based optimizer, as can be seen in Figure 22. This optimization process includes an initial exploration phase, using design of experiment, followed by exploitation steps aimed at minimizing ΔSoC for a specified $M_{payload}$. Figure 22 shows that some regions in the design space are more explored and optimized than others. For regions with low $M_{payload}$, which are less favorable for optimization due to the maximization of objective $f_2 = M_{payload}$, the optimizer shows less tendency for optimization.

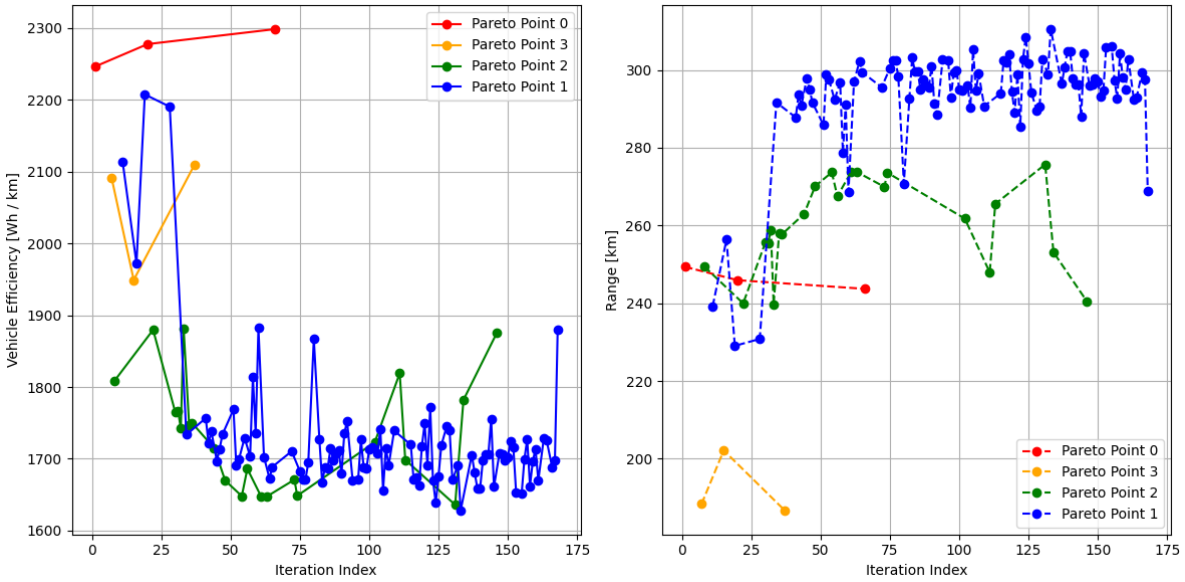


Figure 22. Optimizer Iterations per $M_{payload}$ Pareto point cluster for SA2.19.

A more in depth analysis into the found Pareto front is done for SA1.1 and SA1.2. Figure 23 shows the Pareto points for SA1.1 and SA1.2, where the vehicle efficiency metric is converted into achievable vehicle range by taking into account the capacity of the battery. Here it can be seen that the vehicle range

is inversely proportional to the vehicle's $M_{payload}$. It furthermore can be concluded that for almost all found Pareto points, SA1.1 achieves a higher achievable vehicle range, with a lower associated uncertainty. For system integrators this means that better performance in terms of range can be achieved, at a lower risk level (win-win).

However, when considering vehicle efficiency, Figure 21 shows that SA1.1 achieves higher vehicle efficiency (lower Wh/km) at lower $M_{payload}$, whereas at high $M_{payload}$, SA1.2 achieves better efficiency. From industry it is found that the vehicle efficiency for heavy duty vehicles is in the range of 900 to 1100 Wh/km. This is in good comparison with the found results, especially considering uncertainty.

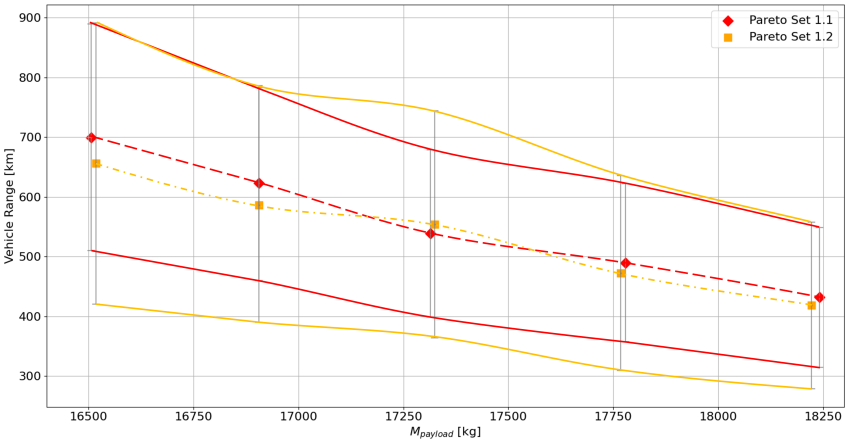
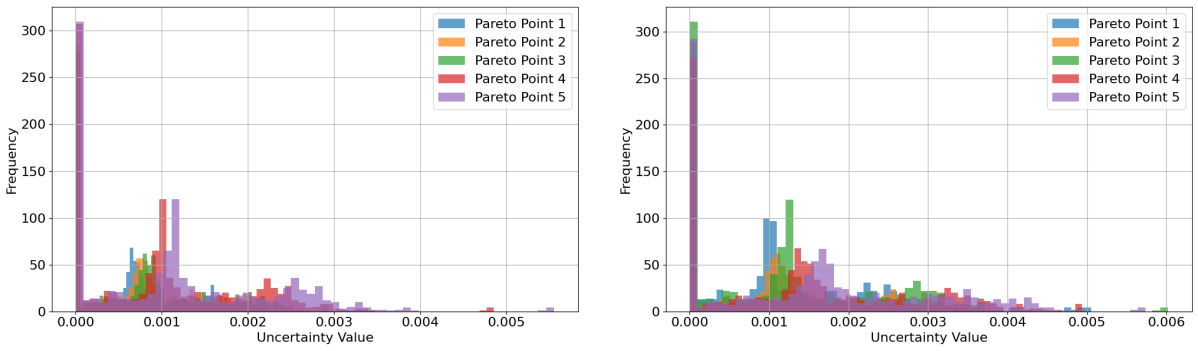


Figure 23. Pareto front with associated range uncertainty for SA1.1 and SA1.2.

B. Uncertainty Distribution For Pareto Optimal Design Points

An uncertainty distribution of the propagated output uncertainty per time step can be generated for every Pareto optimal design point. This is shown for SA1.1 and SA1.2 in Figure 24. Figure 24 highlights that no output uncertainty distribution shows a clear standard statistical distribution, despite assuming normal and uniformly distributed input and model uncertainties. It furthermore can be seen that between system architecture topologies, the maximum propagated uncertainty differs significantly. By analyzing the distribution of uncertainties, we can gain a clearer understanding of their magnitude and how they are spread. This analysis reveals that SA1.2 generally has a similar uncertainty distribution (shape), however with an increased variance compared to SA1.1. This is in line with the found results as shown in Figure 23, where SA1.2 has a larger output uncertainty value for all found Pareto points in comparison to SA1.1. The uncertainty value here is the $s_{\Delta SoC_j}$.



(a) SA1.1

(b) SA1.2

Figure 24. Output uncertainty distribution per system architecture and Pareto point over the HHDDT drive cycle subject to input- and model uncertainty ($s_{model} = \pm\mathcal{U}[0.1775, 0.225]$, $s_{input} = \mathcal{N}(0, 0.2)$).

C. Dynamic Behavior Of A Vehicle Drive Train Subject to Input- and Model Uncertainty

Gaining insights into the output uncertainty levels over a drive cycle is valuable for system integrators, as it allows for a comprehensive evaluation of how input and model uncertainties affect the performance of individual components and, consequently, the entire drive train over time. Therefore, for SA2.12, a detailed analysis of the drive train's dynamic behavior throughout the drive cycle was conducted.

Figure 25 illustrates the battery depletion over time, along with the associated uncertainty levels, for Pareto point 1 of SA2.12 for the found optimal control strategy. Pareto point 1 for SA2.12 includes two batteries with different capacities, resulting in differing ΔSoC curves. These discrepancies are undesirable because they lead to varying battery Open-Circuit Voltages (OCVs) in real-world applications, requiring additional DC-DC converters to maintain consistent battery voltage across energy sources.

Figure 25 furthermore shows the propagated output uncertainty at each time step for both batteries (green & yellow). This line shows a positive correlation between the required vehicle power and the output uncertainty level. This can primarily be explained by the non-linear behavior of the vehicle model.

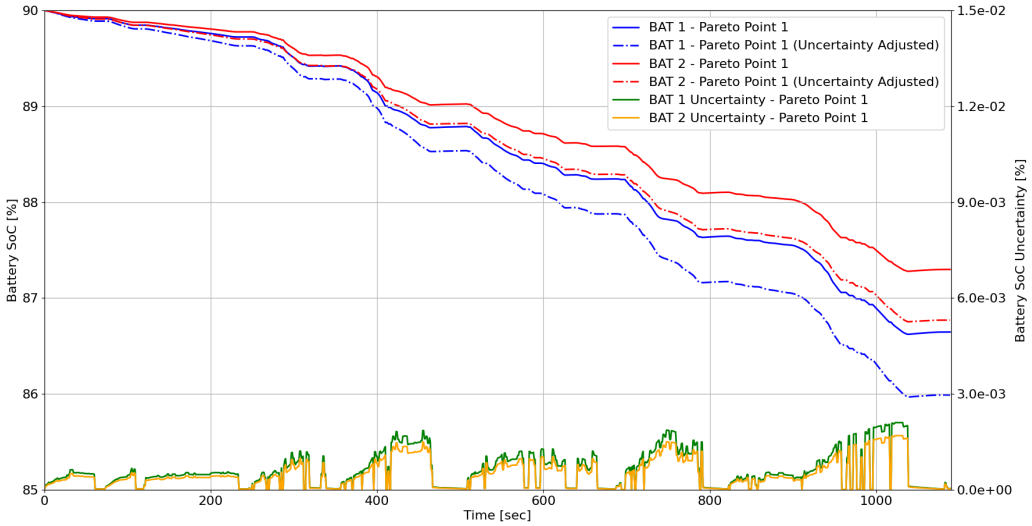


Figure 25. SA 2.12 Pareto point 1 battery depletion's over time with associated output uncertainty ($s_{\Delta SoC}$).

The found optimal control strategy for Pareto Point 1 of SA2.12, for all state variables ($\mathbf{x}_{state}$), is shown in Figure 26. This demonstrates the proper functioning of the *Optimal Control Solver*, as it successfully identifies implicit relationships, such as high gear selection at high vehicle velocities.

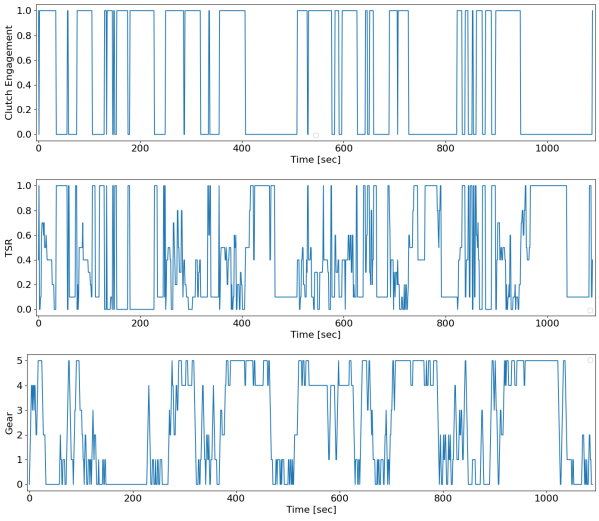


Figure 26. Optimal control strategy for all state variables of SA2.12 Pareto point 1.

V. Conclusion and Recommendations

This research aimed to develop a system architecture exploration, evaluation and uncertainty-based design optimization framework for use in conceptual design phases. By integrating model and input uncertainties — stemming from variability in mission requirements and the use of low-fidelity models typical of conceptual design — the system architecture optimization framework aims to identify a robust optimal system architecture. This approach reduces project risk for vehicle system integrators by increasing confidence in the optimal design's ability to meet predefined requirements.

This research aimed to address the following core research questions:

- *How can uncertainty quantification and propagation be integrated into an automated system architecture optimization workflow for exploring and optimizing vehicle system architectures under uncertainty?*
- *How can the integration of uncertainty quantification and propagation in system architecture optimization processes improve requirement risk management for innovative conceptual design studies of complex engineering systems?*

Designing complex systems is a multi-layered process that begins with defining the system architecture. From this, the optimal component specifications are determined by optimizing the component-specific design variables. As most electro-mechanical components in a vehicle system architecture have multiple operational states, the control of these components over time needs to be evaluated to fairly compare performance. This highlights that system architecture optimization is an interdependent process, where the optimal solution is highly case-specific and subject to significant uncertainty. To address this, an automated multi-layer approach was developed, focusing on architecture exploration, evaluation, and uncertainty-based design optimization.

In this framework, the system architecture exploration phase employs a *bottom-up* approach. This approach is preferred by system integrators, and is driven by established relationships with a wide range of partners and suppliers, as well as a risk-averse approach. Starting with a set of component types, a Boolean Satisfiability Problem (BSP) is solved to come up with a set of viable system architecture topologies. This boolean satisfiability problem is formulated using general system architecture constraints to ensure proper port matching of component ports, no self-loops and other connectivity rules. These architectures are represented as graphs or adjacency matrices, which are used to trace the energy flow across the various components in the system architecture. This results in a measure of vehicle efficiency, specifically battery depletion under different drive cycles (ΔSoC). It was found that the size of the set of viable system architectures scales exponentially with the number of components. Given the impracticality of evaluating all possible architectures from a computational cost perspective, a subset is selected based on high, mean and low structural complexity values.

Designing heavy-duty vehicle drive trains typically involves balancing conflicting objectives such as payload and range. In this research, tools are implemented to determine payload mass and the vehicle achievable range using an optimal control solver. This solver employs time-dependent performance assessments and uses Dijkstra's algorithm to find the control strategy that minimizes the change in battery state-of-charge (ΔSoC). To decrease the computational cost of the optimal control solver, a surrogate model using random forest classification & regression was implemented to achieve accurate results at low computational cost.

Using the above-formulated multi-disciplinary analysis (MDA), a surrogate-based optimization (SBO) algorithm was implemented to compute the deterministic optimal design point for a given system architecture topology by optimizing the component-specific design variables. To incorporate uncertainty quantification and propagation into the multi-disciplinary design optimization framework, model sensitivities were used to handle and propagate input and model uncertainties to the output ($s_{\Delta SoC}$). This approach provides system integrators with uncertainty measures for the identified Pareto points specific to each system architecture topology. With this increased knowledge, informed decisions can be made confidently, thereby reducing project risk.

A case study was conducted to optimize a heavy-duty vehicle drive train using the uncertainty-based

system architecture optimization framework, comparing two component sets representing a simple ($x_{components_1}$) and a complex ($x_{components_2}$) drive train. The results show that no clear optimal solutions ($max(M_{payload})$, $min(\Delta SoC)$) can be found when comparing the system architecture topology optimal design points.

For $x_{components_1}$, two feasible system architecture topologies are generated using the System Architecture Topology Generator. For the two topologies, Pareto fronts spanning from 16,500 to 18,250 kg $M_{payload}$ and vehicle efficiencies ranging from 750 to 930 Wh/km, with associated uncertainties between 200 and 500 Wh/km are found. These results align with real-world values for the heavy-duty trucking industry, typically around 900 to 1100 Wh/km. For these Pareto points, the computed uncertainty was non-constant and highly sensitive to the arrangement of components within the system architecture. Components with higher sensitivity to uncertainties, located closer to the wheels (i.e., with fewer intermediary components in the drive train), contributed more to the total uncertainty compared to components positioned further along the drive train. An in depth analysis into the uncertainty distribution of the drive train over the drive cycle, shows that no standard statistical form was observed, despite the use of normally distributed input-uncertainty and uniformly distributed model-uncertainty.

For $x_{components_2}$, a set of 52 feasible system architecture topologies were identified. Due to the high computational cost of optimizing all topologies, a subset was selected based on structural complexity values for further optimization. For the topologies, Pareto fronts spanning from 16,000 to 17,500 kg $M_{payload}$ and vehicle efficiencies ranging from 1450 to 2300 Wh / km, with associated uncertainties between 350 and 600 Wh / km are found. This shows that the found Pareto points have a significantly lower vehicle efficiency (higher Wh / km) compared to the system architecture topologies for $x_{components_1}$. This can be explained by the fact that component set 2 consists of more components, each of which contributes to increased inefficiency in the vehicle drive train. The potential benefits of operating components at their optimal operating points are outweighed by the inefficiencies introduced by the additional components. Furthermore, the added weight of the drive train components results in a lower $M_{payload}$ for $x_{components_2}$.

Future research should focus on several key areas to increase the effectiveness of the framework:

- **Advanced Uncertainty Quantification:** Improving uncertainty quantification methods is crucial for increasing the reliability of the Pareto fronts obtained from the framework. By investigating more precise uncertainty quantification techniques for estimating input and model uncertainties at various operating conditions, researchers increase confidence in the found optimization results. This will enable a more accurate assessment of the robustness of the optimal design points.
- **Implementation of Nested and Global Optimization Strategies:** Implementing and testing a fully nested optimization strategy versus a global optimization approach will provide valuable insights into the framework's performance in both system architecture exploration and exploitation, as proposed by Santiago et al. [33]. This will improve the framework's ability to identify the most effective single solution among many feasible options. By evaluating the strengths and limitations of each strategy, researchers can determine which approach best addresses the complexities of the system architecture design space when considering uncertainty.

The proposed framework contributes to better risk management by improving the robustness of optimal designs subject to uncertainties and providing *system integrators* with a clearer understanding of associated uncertainty levels in their projects. This is valuable for any system design process where risk mitigation is crucial. Although this research focuses on vehicle development, the underlying principles of uncertainty-based system architecture exploration and optimization have broad relevance across many applications.

Acknowledgments

This research was conducted in collaboration with VDL Special Vehicles, who made significant contributions to this study. the author would like to thank S. van Rijn and C. van Hoek for their invaluable guidance on design principles pertaining to zero-emission drive trains for heavy-duty applications.

References

[1] Krueger, M., Lewis, J., Jacoby, C., Rantowich, N., and Ice, M., *Systems Engineering Guidebook for Intelligent Transportation Systems*, 2009.

[2] Crawley, E., Cameron, B., and Selva, D., *System architecture: strategy and product development for complex systems*, Prentice Hall Press, 2015.

[3] Iacobucci, J. V., *Rapid Architecture Alternative Modeling (RAAM): a framework for capability-based analysis of system of systems architectures*, Georgia Institute of Technology, 2012.

[4] Bussemaker, J. H., Ciampa, P. D., and Nagel, B., "System architecture design space exploration: An approach to modeling and optimization," *AIAA Aviation 2020 Forum*, 2020, p. 3172.

[5] Bruggeman, A.-L., van Manen, B., van der Laan, T., van den Berg, T., and La Rocca, G., "An MBSE-based requirement verification framework to support the MDAO process," *AIAA Aviation 2022 Forum*, 2022, p. 3722.

[6] Babaei, A. R., Setayandeh, M. R., and Farrokhfal, H., "Aircraft robust multidisciplinary design optimization methodology based on fuzzy preference function," *Chinese Journal of Aeronautics*, Vol. 31, No. 12, 2018, pp. 2248–2259.

[7] Yao, W., Chen, X., Luo, W., Van Tooren, M., and Guo, J., "Review of uncertainty-based multidisciplinary design optimization methods for aerospace vehicles," *Progress in Aerospace Sciences*, Vol. 47, No. 6, 2011, pp. 450–479.

[8] Judt, D., and Lawson, C., "Methodology for automated aircraft systems architecture enumeration and analysis," *12th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference and 14th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 2012, p. 5648.

[9] Judt, D. M., and Lawson, C., "Development of an automated aircraft subsystem architecture generation and analysis tool," *Engineering Computations*, Vol. 33, No. 5, 2016, pp. 1327–1352.

[10] Frank, C., Pinon-Fischer, O. J., and Mavris, D. N., "A design space exploration methodology to support decisions under evolving requirements' uncertainty and its application to suborbital vehicles," *53rd AIAA Aerospace Sciences Meeting*, 2015, p. 1010.

[11] Matthews, P. C., "Challenges to Bayesian decision support using morphological matrices for design: empirical evidence," *Research in Engineering Design*, Vol. 22, No. 1, 2011, pp. 29–42.

[12] Chakrabarti, A., Shea, K., Stone, R., Cagan, J., Campbell, M., Hernandez, N. V., and Wood, K. L., "Computer-based design synthesis research: an overview," 2011.

[13] Ölvander, J., Lundén, B., and Gavel, H., "A computerized optimization framework for the morphological matrix applied to aircraft conceptual design," *Computer-Aided Design*, Vol. 41, No. 3, 2009, pp. 187–196.

[14] Simmons, W. L., "A framework for decision support in systems architecting," Ph.D. Thesis, Massachusetts Institute of Technology, 2008.

[15] Helms, B., "Object-oriented graph grammars for computational design synthesis," Ph.D. Thesis, Technische Universität München, 2013.

[16] van Gent, I., and La Rocca, G., "Formulation and integration of MDAO systems for collaborative design: A graph-based methodological approach," *Aerospace Science and Technology*, Vol. 90, 2019, pp. 410–433.

[17] Herber, D. R., Guo, T., and Allison, J. T., "Enumeration of architectures with perfect matchings," *Journal of Mechanical Design*, Vol. 139, No. 5, 2017, p. 051403.

[18] Münzer, C., Helms, B., and Shea, K., "Automatically transforming object-oriented graph-based representations into Boolean satisfiability problems for computational design synthesis," *Journal of Mechanical Design*, Vol. 135, No. 10, 2013, p. 101001.

[19] Helms, B., and Shea, K., "Computational synthesis of product architectures based on object-oriented graph grammars," 2012.

[20] Sinha, K., and de Weck, O. L., "Structural complexity quantification for engineered complex systems and implications on system architecture and design," *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, Vol. 55881, American Society of Mechanical Engineers, 2013, p. V03AT03A044.

[21] Efatmaneshnik, M., and Ryan, M. J., "A general framework for measuring system complexity," *Complexity*, Vol. 21, No. S1, 2016, pp. 533–546.

[22] Shangguan, J., Yue, M., Fang, C., and Qi, H., "Robust design optimization of component parameters for DMDEB powertrain system based on Taguchi method," *Journal of Dynamic Systems, Measurement, and Control*, Vol. 144, No. 9, 2022, p. 091003.

[23] Kabalan, B., Vinot, E., Trigui, R., and Dumand, C., "Systematic methodology for architecture generation and design optimization of hybrid powertrains," *IEEE Transactions on Vehicular Technology*, Vol. 69, No. 12,

2020, pp. 14846–14857.

[24] Manber, U., *Introduction to algorithms: a creative approach*, Addison-Wesley Longman Publishing Co., Inc., 1989.

[25] Proctor, C. L., Grimes, W. D., Fournier Jr, D. J., Rigol Jr, J., and Sunseri, M. G., "Analysis of acceleration in passenger cars and heavy trucks," *SAE transactions*, 1995, pp. 283–324.

[26] AbuSalim, S. W., Ibrahim, R., Saringat, M. Z., Jamel, S., and Wahab, J. A., "Comparative analysis between dijkstra and bellman-ford algorithms in shortest path optimization," *IOP Conference Series: Materials Science and Engineering*, Vol. 917, IOP Publishing, 2020, p. 012077.

[27] Wu, X., Liu, B., Ricks, N., and Ghorbaniasl, G., "Surrogate models for performance prediction of axial compressors using through-flow approach," *Energies*, Vol. 13, No. 1, 2019, p. 169.

[28] Gu, X., Renaud, J. E., Batill, S. M., Brach, R. M., and Budhiraja, A. S., "Worst case propagated uncertainty of multidisciplinary systems in robust design optimization," *Structural and Multidisciplinary Optimization*, Vol. 20, 2000, pp. 190–213.

[29] Ku, H. H., et al., "Notes on the use of propagation of error formulas," *Journal of Research of the National Bureau of Standards*, Vol. 70, No. 4, 1966.

[30] Cao, H., and Duan, B., "Uncertainty analysis for multidisciplinary systems based on convex models," *10th AIAA/ISSMO multidisciplinary analysis and optimization conference*, 2004, p. 4504.

[31] Martins, J. R., and Lambe, A. B., "Multidisciplinary design optimization: a survey of architectures," *AIAA journal*, Vol. 51, No. 9, 2013, pp. 2049–2075.

[32] Park, S. H., *Robust design and analysis for quality engineering*, Chapman & Hall, 1996.

[33] Valencia Ibañez, S., "Optimization Strategies for System Architecting Problems," *TU Delft repository*, 2023.

[34] Bussemaker, J. H., Bartoli, N., Lefebvre, T., Ciampa, P. D., and Nagel, B., "Effectiveness of surrogate-based optimization algorithms for system architecture optimization," *AIAA Aviation 2021 forum*, 2021, p. 3095.

[35] Bussemaker, J. H., "SBArchOpt: Surrogate-based architecture optimization," *Journal of Open Source Software*, Vol. 8, No. 89, 2023, p. 5564.

[36] Bouhlel, M. A., Hwang, J. T., Bartoli, N., Lafage, R., Morlier, J., and Martins, J. R., "A Python surrogate modeling framework with derivatives," *Advances in Engineering Software*, Vol. 135, 2019, p. 102662.

[37] Bauer, M., Van der Wilk, M., and Rasmussen, C. E., "Understanding probabilistic sparse Gaussian process approximations," *Advances in neural information processing systems*, Vol. 29, 2016.

Appendix

XDSM of the UBSAO

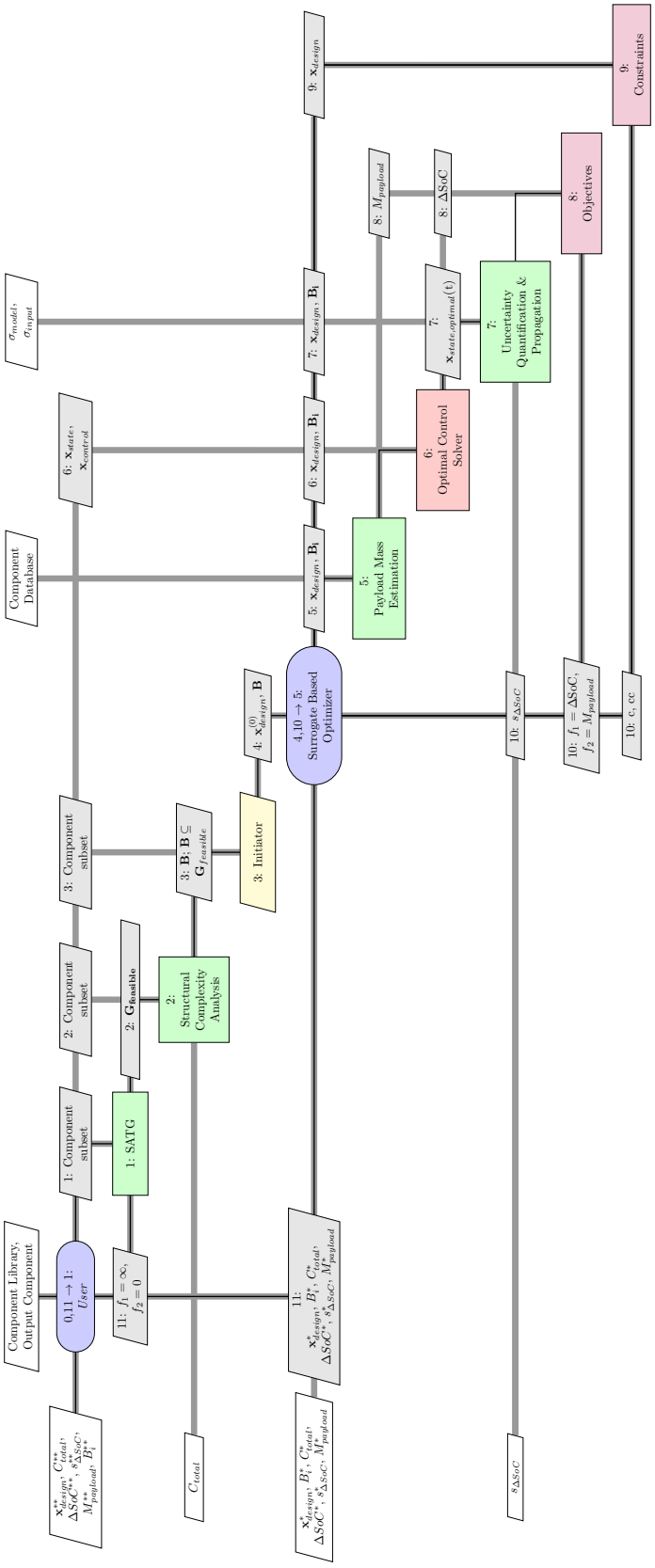


Figure 27. XDSM of the Uncertainty-Based System Architecture Optimization framework developed in this research.

UBSAO - Found Optimal Design Variables For Case Study

Below, the optimal design variables for every Pareto point found per system architecture are shown, together with their optimal objective values and associated uncertainty levels.

Found Pareto Points For SA1.1

Table 4. Optimal design variables for all Pareto points of SA1.1

Component	**Design Variable x**	$\mathbf{x_{lb}}$	$\mathbf{x_{ub}}$	**Type**	$\mathbf{x_1^*}$	$\mathbf{x_2^*}$	$\mathbf{x_3^*}$	$\mathbf{x_4^*}$	$\mathbf{x_5^*}$
BAT1	Battery Type 1	1	3	*Integer*	3	1	1	1	2
BAT2	Battery Type 2	1	3	*Integer*	3	3	1	2	2
INV1	Inverter Type 1	1	2	*Integer*	2	1	2	1	1
INV2	Inverter Type 2	1	2	*Integer*	2	1	2	1	1
EM1	Electric Machine Type 1	1	3	*Integer*	3	3	3	1	1
RED1	RED Gear ratio 1	13.0	15.0	*Continuous*	14.96	14.98	14.71	14.73	14.75
Wheels1	Wheel diameter	0.9	1.1	*Continuous*	0.999	0.917	0.92	0.90	0.9
ΔSoC [%]					2.10	2.35	2.72	2.99	3.40
$\mathbf{M_{payload}}$ [kg]					16506	16906	17314	17779	18241
$\mathbf{s_{\Delta SoC}}$ [%]					0.780	0.835	0.953	1.111	1.262
$\eta_{\mathbf{vehicle}}$ [Wh / km]					641	723	731	921	912
$\mathbf{s}_{\eta_{\mathbf{vehicle}}}$ [Wh / km]					238	256	256	341	339
Range [km]					699	623	539	489	431
$\mathbf{s_{range}}$ [km]					189	163	139	133	117

Found Pareto Points For SA1.2

Table 5. Optimal design variables for all Pareto points of SA1.2

Component	**Design Variable x**	$\mathbf{x_{lb}}$	$\mathbf{x_{ub}}$	**Type**	$\mathbf{x_1^*}$	$\mathbf{x_2^*}$	$\mathbf{x_3^*}$	$\mathbf{x_4^*}$	$\mathbf{x_5^*}$
BAT1	Battery Type 1	1	3	*Integer*	3	1	1	2	2
BAT2	Battery Type 2	1	3	*Integer*	3	3	1	1	2
INV1	Inverter Type 1	1	2	*Integer*	1	2	2	1	1
INV2	Inverter Type 2	1	2	*Integer*	1	2	2	1	1
EM1	Electric Machine Type 1	1	3	*Integer*	1	3	1	3	3
RED1	RED Gear ratio 1	13.0	15.0	*Continuous*	14.152	14.98	14.760	14.264	15.00
Wheels1	Wheel diameter	0.9	1.1	*Continuous*	0.933	0.970	0.9	0.953	0.9
ΔSoC [%]					2.235	2.501	2.647	3.103	3.505
$\mathbf{M_{payload}}$ [kg]					16517	16906	17325	17768	18222
$\mathbf{s_{\Delta SoC}}$ [%]					1.249	1.257	1.380	1.634	1.748
$\eta_{\mathbf{vehicle}}$ [Wh / km]					854	862	814	834	805
$\mathbf{s}_{\eta_{\mathbf{vehicle}}}$ [Wh / km]					478	433	425	439	402
Range [km]					656	586	554	473	418
$\mathbf{s_{range}}$ [km]					235	196	190	163	139

Found Pareto Points For SA2.12

Table 6. Optimal design variables for all Pareto points of SA2.12

Component	**Design Variable x**	$\mathbf{x_{lb}}$	$\mathbf{x_{ub}}$	**Type**	$\mathbf{x}_1^*$	$\mathbf{x}^*$	$\mathbf{x}_3^*$	$\mathbf{x}_4^*$	$\mathbf{x}_5^*$
BAT1	Battery Type 1	1	3	*Integer*	3	1	1	2	2
BAT2	Battery Type 2	1	3	*Integer*	3	1	1	1	2
INV1	Inverter Type 1	1	2	*Integer*	2	1	1	1	2
INV2	Inverter Type 2	1	2	*Integer*	2	1	2	1	2
INV3	Inverter Type 3	1	2	*Integer*	2	1	2	1	2
INV4	Inverter Type 4	1	2	*Integer*	2	1	2	1	2
EM1	Electric Machine Type 1	1	3	*Integer*	2	2	3	2	2
EM2	Electric Machine Type 2	1	3	*Integer*	3	2	3	3	1
MSG1	MSG Gear ratio 1	4.0	7.0	*Continuous*	5.36	6.03	5.91	6.60	5.70
	MSG Gear ratio 2	4.0	6.0	*Continuous*	4.64	4.89	4.38	4.22	5.20
	MSG Gear ratio 3	2.8	4.0	*Continuous*	3.40	3.70	3.06	3.27	3.49
	MSG Gear ratio 4	2.0	3.6	*Continuous*	2.82	2.47	2.31	2.85	2.49
	MSG Gear ratio 5	1.8	2.8	*Continuous*	1.96	2.47	2.12	2.80	2.08
	MSG Gear ratio 6	1.5	2.3	*Continuous*	1.73	1.91	2.08	2.17	1.54
TC1	TC Gear ratio 1	0.8	1.2	*Continuous*	0.98	0.96	1.12	1.15	0.94
	TC Gear ratio 2	0.8	1.2	*Continuous*	1.08	1.12	1.17	1.16	1.11
RED1	RED Gear ratio 1	4.0	8.0	*Continuous*	7.85	7.05	7.71	7.91	6.95
Wheels1	Wheel diameter	0.9	1.1	*Continuous*	0.988	0.919	1.045	0.963	0.936
ΔSoC [%]					5.242	6.531	6.539	6.806	10.215
$\mathbf{M_{payload}}$ [kg]					15734	16171	16481	16980	17477
sΔSoC [%]					1.065	1.189	1.396	1.579	1.707
$\eta_{\mathbf{vehicle}}$ [Wh / km]					1807	1997	2011	1828	2347
$\mathbf{s}_{\eta_{\mathbf{vehicle}}}$ [Wh / km]					367	363	429	424	392
Range [km]					279	224	224	215	143
$\mathbf{s_{range}}$ [km]					47	35	39	41	21

Found Pareto Points For SA2.18

Table 7. Optimal design variables for all Pareto points of SA2.18

Component	**Design Variable x**	$\mathbf{x_{lb}}$	$\mathbf{x_{ub}}$	**Type**	$\mathbf{x_1^*}$	$\mathbf{x_2^*}$	$\mathbf{x_3^*}$	$\mathbf{x_4^*}$
BAT1	Battery Type 1	1	3	*Integer*	3	1	2	1
BAT2	Battery Type 2	1	3	*Integer*	3	3	3	2
INV1	Inverter Type 1	1	2	*Integer*	2	2	1	1
INV2	Inverter Type 2	1	2	*Integer*	2	2	1	1
INV3	Inverter Type 3	1	2	*Integer*	2	2	1	1
INV4	Inverter Type 4	1	2	*Integer*	2	2	1	1
EM1	Electric Machine Type 1	1	3	*Integer*	3	3	2	2
EM2	Electric Machine Type 2	1	3	*Integer*	1	2	3	1
MSG1	MSG Gear ratio 1	4.0	7.0	*Continuous*	5.35	6.13	6.01	6.44
	MSG Gear ratio 2	4.0	6.0	*Continuous*	4.01	4.76	4.88	5.58
	MSG Gear ratio 3	2.8	4.0	*Continuous*	3.49	3.13	3.72	3.77
	MSG Gear ratio 4	2.0	3.6	*Continuous*	3.12	2.25	2.99	2.74
	MSG Gear ratio 5	1.8	2.8	*Continuous*	2.35	2.18	2.26	2.15
	MSG Gear ratio 6	1.5	2.3	*Continuous*	1.69	1.56	1.67	1.82
TC1	TC Gear ratio 1	0.8	1.2	*Continuous*	1.15	1.00	0.98	0.94
	TC Gear ratio 2	0.8	1.2	*Continuous*	1.08	0.98	1.05	1.08
RED1	RED Gear ratio 1	4.0	8.0	*Continuous*	7.79	6.82	5.27	6.70
Wheels1	Wheel diameter	0.9	1.1	*Continuous*	1.08	0.98	1.05	1.08
ΔSoC [%]					5.508	5.284	5.000	6.762
$\mathbf{M_{payload}}$ [kg]					15716	16126	16588	16999
$\mathbf{s_{\Delta SoC}}$ [%]					1.395	1.294	1.531	1.721
$\eta_{\mathbf{vehicle}}$ [Wh / km]					2104	1822	1530	1817
$\mathbf{s}_{\eta_{\mathbf{vehicle}}}$ [Wh / km]					266	277	293	217
Range [km]					533	446	468	462
$\mathbf{s_{range}}$ [km]					54	55	69	44

Found Pareto Points For SA2.19

Table 8. Optimal design variables for all Pareto points of SA2.19

Component	**Design Variable x**	$\mathbf{x_{lb}}$	$\mathbf{x_{ub}}$	**Type**	$\mathbf{x_1^*}$	$\mathbf{x_2^*}$	$\mathbf{x_3^*}$	$\mathbf{x_4^*}$
BAT1	Battery Type 1	1	3	*Integer*	3	1	1	1
BAT2	Battery Type 2	1	3	*Integer*	3	3	1	2
INV1	Inverter Type 1	1	2	*Integer*	2	1	1	2
INV2	Inverter Type 2	1	2	*Integer*	2	1	1	2
INV3	Inverter Type 3	1	2	*Integer*	2	1	1	2
INV4	Inverter Type 4	1	2	*Integer*	2	1	1	2
EM1	Electric Machine Type 1	1	3	*Integer*	3	2	2	3
EM2	Electric Machine Type 2	1	3	*Integer*	1	2	1	1
MSG1	MSG Gear ratio 1	4.0	7.0	*Continuous*	5.35	5.77	5.82	5.73
	MSG Gear ratio 2	4.0	6.0	*Continuous*	4.01	5.65	4.79	4.82
	MSG Gear ratio 3	2.8	4.0	*Continuous*	3.49	3.17	3.37	3.41
	MSG Gear ratio 4	2.0	3.6	*Continuous*	3.12	2.81	2.63	2.68
	MSG Gear ratio 5	1.8	2.8	*Continuous*	2.35	1.98	2.27	2.52
	MSG Gear ratio 6	1.5	2.3	*Continuous*	1.69	1.73	1.93	2.24
TC1	TC Gear ratio 1	0.8	1.2	*Continuous*	1.15	0.91	1.02	0.82
	TC Gear ratio 2	0.8	1.2	*Continuous*	1.16	1.07	1.14	1.07
RED1	RED Gear ratio 1	4.0	8.0	*Continuous*	7.79	6.40	4.86	6.22
Wheels1	Wheel diameter	0.9	1.1	*Continuous*	1.08	0.98	0.97	1.06
ΔSoC [%]					5.879	4.722	5.318	7.253
$\mathbf{M_{payload}}$ [kg]					15716	16163	16555	16970
$\mathbf{s_{\Delta SoC}}$ [%]					1.387	1.171	1.324	1.777
$\eta_{\mathbf{vehicle}}$ [Wh / km]					2246	1628	1636	1949
$\mathbf{s}_{\eta_{\mathbf{vehicle}}}$ [Wh / km]					249	311	276	202
Range [km]					530	311	276	202
$\mathbf{s_{range}}$ [km]					48	62	55	40

Found Pareto Points For SA2.43

Table 9. Optimal design variables for all Pareto points of SA2.43

Component	**Design Variable x**	$\mathbf{x_{lb}}$	$\mathbf{x_{ub}}$	**Type**	$\mathbf{x}_1^*$	$\mathbf{x}^*$	$\mathbf{x}_3^*$	$\mathbf{x}_4^*$	$\mathbf{x}_5^*$
BAT1	Battery Type 1	1	3	*Integer*	3	1	2	1	2
BAT2	Battery Type 2	1	3	*Integer*	3	3	3	2	2
INV1	Inverter Type 1	1	2	*Integer*	2	1	1	2	2
INV2	Inverter Type 2	1	2	*Integer*	2	1	1	2	2
INV3	Inverter Type 3	1	2	*Integer*	2	1	1	2	2
INV4	Inverter Type 4	1	2	*Integer*	2	1	1	2	2
EM1	Electric Machine Type 1	1	3	*Integer*	1	2	2	1	1
EM2	Electric Machine Type 2	1	3	*Integer*	1	2	2	2	1
MSG1	MSG Gear ratio 1	4.0	7.0	*Continuous*	5.96	5.10	5.11	5.69	5.99
	MSG Gear ratio 2	4.0	6.0	*Continuous*	5.40	4.97	5.02	4.19	5.40
	MSG Gear ratio 3	2.8	4.0	*Continuous*	3.20	3.58	3.84	3.28	3.57
	MSG Gear ratio 4	2.0	3.6	*Continuous*	2.79	2.81	2.83	3.22	3.08
	MSG Gear ratio 5	1.8	2.8	*Continuous*	2.31	2.06	1.94	2.35	2.10
	MSG Gear ratio 6	1.5	2.3	*Continuous*	1.71	1.62	1.93	1.67	1.69
TC1	TC Gear ratio 1	0.8	1.2	*Continuous*	1.06	0.98	1.04	1.03	1.01
	TC Gear ratio 2	0.8	1.2	*Continuous*	1.16	1.08	1.09	1.06	1.14
RED1	RED Gear ratio 1	4.0	8.0	*Continuous*	7.43	6.61	5.92	6.68	6.28
Wheels1	Wheel diameter	0.9	1.1	*Continuous*	1.08	1.02	1.02	1.00	1.01
ΔSoC [%]					5.389	5.284	7.103	7.768	8.813
$\mathbf{M_{payload}}$ [kg]					15745	16145	16607	17007	17469
$\mathbf{s_{\Delta SoC}}$ [%]					0.970	1.102	1.345	1.420	1.629
$\boldsymbol{\eta}_{\mathbf{vehicle}}$ [Wh / km]					1649	1420	1632	2087	2025
$\mathbf{s}_{\eta_{\mathbf{vehicle}}}$ [Wh / km]					272	277	206	189	166
Range [km]					297	296	309	382	374
$\mathbf{s_{range}}$ [km]					41	48	33	29	26

2

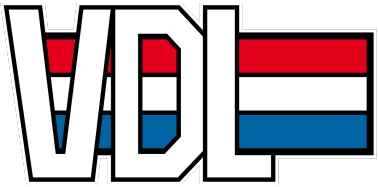
Literature Review

(PREVIOUSLY GRADED UNDER AE4020)

A Review of Vehicle System Architecture Exploration, Evaluation and Uncertainty-Based Design Optimization

Name	Student ID
Luuk Halsema	4659953

September 2023



Contents

Summary

The goal of this report is to provide a comprehensive review of all relevant state-of-the-art knowledge on vehicle system architecture design methods, including architecture exploration, evaluation and eventually optimization. It furthermore provides the reader with extensive knowledge on the quantification and propagation of uncertainty (both model and input) throughout a multidisciplinary analysis. A research gap was found between the two above mentioned topics, which could enhance the requirement risk management of vehicle drive train OEM's, such as VDL Special Vehicles, in conceptual design. This way, system engineers can, in early stages of the design process, increase their knowledge on the systems expected performance and optimize this. Then, by taking into account uncertainty, they can ensure that a determined system architecture will adhere to the specified requirements in later stages of the design process, while still being an optimal solution.

This literature review clearly highlights the absent knowledge of integrating uncertainty-based optimization into system architecture optimization. This absence of knowledge can be split-up into various facets, which should be addressed to give a scientifically substantiated answer to the research objective.

The first chapter elaborates on the topic of architectural design space exploration and evaluation implemented into the conceptual design phase of a standardized hardware V-model. It explains the potentially gains of increasing system specific knowledge in early stages of design process. One of the first steps in a hardware V-model is the determination of a system architecture. A preferred method of finding a system architecture which satisfies the specified requirements, is by performing an Architectural Design Space Exploration. By combining existing knowledge on the topic, and enumerating all viable design options, an Architectural Design Space Graph can be formulated. This is a discrete representation of all possible system architectures, showing all feasible connections between architectural components. A step which usually occurs prior to this, is the enumeration of all possible components in a system architecture. This is usually done by building a morphological matrix or by means of functional decomposition. A morphological matrix enumerates all *forms* which can fulfill a specific *function*. Functional decomposition works by finding the various functions a system should fulfill, after which a type of form is then assigned to that function. From this form, other functions can be deduced, which results in a tree build up of the overall system. Lastly, a more computational method was proposed which works by generating a constraint satisfactory problem using graph theory and a list of preferred or available components. By finding all combinatorial solutions of components, and adding constraints to limit the connectivity of components (incompatibility or preference), a set of feasible system architectures is found. Here, also the multi-domain nature of vehicle drive trains can be accounted for.

With the increased computational power of modern laptops, numerical simulation and evaluation of the relevant system architectures can take place using advanced, low- to mid-fidelity models. This in turn produces a better understanding of a systems' performance. In case a numerical model is to computationally expensive, surrogate models can be build which decrease the computational cost of a model at the expense of lower accuracy. Various surrogate modelling techniques are presented ranging from radial basis functions to kriging.

The next chapter is more geared towards obtaining a proper understanding of the modelling of the various systems and components in an electric vehicle. Here, as a starting point, the vehicle model which is used to determine the weight-, energy- and power budget of an electric vehicle is explained. Next, the relevant electrical/mechanical components are elaborated upon together with their most relevant (open-source) (surrogate) models. A widely applied low-fidelity method to dynamically model electrical components is through equivalent circuit models. Here, an electrical component is modelled as a series of resistances (R), capacitors (C) and inductors (L), where an input in voltage or current determines the output characteristics of the component. Equivalent circuit models are computationally cheap but still result in a relatively high level of accuracy. Mechanical components are modelled using simplified empirical or analytical relations.

Having determined how to build various system architectures using specified components and their respective constraints, and knowing how to model the components' performance using (surrogate) models, the next step of the research project can be elaborated upon, namely the optimization part. For this, first a clear introduction on the topic of Multidisciplinary Design Analysis and Optimization (MDAO) is given. Here, the various MDO architectures, convergence scheme's and optimization algorithms are explained. It was found that for system architecture optimization, where the design space is mixed-integer and hierarchical, specialized methods should be used, which are able to cope well with such design spaces. An algorithm well suited for this mixed-integer design space is the (AMI)EGO framework, which works by performing gradient-based optimization on the continuous design vector, and a gradient-free approach for the discrete design variables. By fitting a continuous kriging model on the discrete design space, a discrete optima can be found. To deal with the changing design vector as a result of the change in system components (and their continuous design variables) in system architecture optimization, a design vector *imputor* is proposed. This imputor determines which continuous design variables are (in)active as a result of the active discrete design variables chosen by the optimizer.

A big part of this thesis consists of quantifying input- and model-uncertainty and their propogation through a system architecture evaluation (MDA). Input uncertainty is the aleatory (random) or epistemic (lack of knowledge) uncertainty of the input vector of an analysis. This can either be uncertainty in the design variables ($\mathbf{x}_i$) or the coupling/state variables ($\mathbf{y}_i$) in a coupled MDA. Model uncertainty is the result of model assumptions (bias) when trying to mimic a system as true to nature. Uncertainty is quantified using probability theory, where design variables and models are stochastic, meaning that they have a probability distribution with mean and variance. By integrating this uncertainty into a MDAO framework, a system engineer can choose to design for a robust and/or reliable optimal design solution. Here Robust Design Optimization can be explained as a method trying to find a design solutions for which the objective function is less/in-sensitive to a variation in input parameters. Reliability-Based Design Optimization is more geared towards achieving constraints satisfaction in case of model- or input-uncertainty. By combining both methods, a reliable and robust design optima can be found, which is referred to as Reliability-Based Robust Design Optimization (RBRDO). In order to quantify this uncertainty and its propagation, both intrusive and non-intrusive methods are found. The most accurate, but computationally expensive method, is Monte Carlo Simulation, which assumes an input distribution and from that uses large amounts of function evaluations to determine the output distribution. More preferred and computationally cheap methods to account for uncertainty are first-order perturbation methods, which model uncertainty as a first-order Taylor expansion. Sensitivity Analysis (SA) can help to decrease the computational cost of uncertainty quantification by limiting the uncertainty analysis only to design variables which are affected by this model- or input uncertainty.

The second to last chapter focuses on presenting methods which can be used to dynamically build executable MDAO workflows. Here, a tool named KADMOS is proposed. KADMOS is able to automatically generate and adapt specific MDAO executable workflows given a set of tools and their respective in- and outputs. It does this by modelling an MDAO problem as a set of directed graphs to which smart sequencing algorithms are applied to come up with an executable MDAO workflow. KADMOS outputs a CMDOWS file, which is a type of centralized data scheme, which can be integrated into OpenMDAO.

Finally, the last chapter explains the thesis methodology including the research relevance, objective, (sub)questions and time line. By properly defining the research relevance, a research gap can be formulated. The formulation of a set of (sub)questions then helps to achieve the research objective. This research stems from VDL's need to increase the system specific knowledge in early stages of design without limiting the design space using engineering bias or safety factors. The goal of this research is therefore to build a tool which produces an optimal system architecture while taking into account model- and input uncertainty due to a lack of knowledge in conceptual design. This way, the system engineers can be more confident that the determined design will adhere to the specified customer requirements, all while still being an optimal system architecture.

Nomenclature

γ_i	Gear ratio i	[–]
$\hat{u}_i$	Coupling variable	[–]
$\mathcal{L}$	Lagrangian	[–]
μ	Mean	[–]
$\omega_{in,out}$	Rotational speed	$[rpm]$
Ψ_i	Orthogonal basis functions	[–]
ρ_a	Air density	$[kg/m^3]$
σ^2	Variance	[–]
$\mathbf{C}$	A finite set of constraints	[–]
$\mathbf{D}$	The set of variable specific domains	[–]
$\mathbf{E}$	A set of connection or edges in graph theory	[–]
$\mathbf{R}$	Reliability vector	[–]
$\mathbf{T}$	A graph set	[–]
$\mathbf{T}^i$	A set of infeasible topology graphs	[–]
$\mathbf{T}^p$	A set of all possible topology graphs	[–]
$\mathbf{T}^{fe}$	A set of feasible topology graphs	[–]
$\mathbf{V}$	A set of components in graph theory	[–]
$\mathbf{X}$	A finite set of variables	[–]
$\mathbf{x}$	Design variable vector	[–]
$\mathbf{x}_C$	Continuous design variable vector	[–]
$\mathbf{x}_I$	Integer design variable vector	[–]
A_f	Frontal area	$[m^2]$
C_d	Drag coefficient	[–]
c_m	Normalized ICE speed	[–]
C_r	Rolling friction coefficient	[–]
$Cov(X,Y)$	Covariance	[–]
f	Objective function	[–]
g_i	Inequality constraint	[–]
h_i	Equality constraint	[–]
$i_{fuelcell}$	Fuel cell generated current	$[A]$
J_f	Jacobian objective matrix	[–]
J_g	Jacobian inequality matrix	[–]

J_h	Jacobian equality matrix	$[-]$
J_i	Sub problem objective function	$[-]$
m_v	Vehicle mass	$[kg]$
P_e	Mechanical ICE power	$[W]$
p_{me}	Normalized ICE pressure	$[-]$
R_{ohm}	Electrical Resistance	$[\Omega]$
$T_{in,out}$	Torque	$[Nm]$
v	Vehicle velocity	$[m/s]$
$V_{fuelcell}$	Fuel cell generated voltage	$[V]$
w_i, β_i	Surrogate model weights	$[-]$
x_i	Design variable	$[-]$
x_i^*	Optimal design variable	$[-]$
x_i^L	Lower bound design variable	$[-]$
x_i^U	Lower bound design variable	$[-]$

List of abbreviations

MDAO Multidisciplinary Design Analysis and Optimization

INCOSE International Council on System Engineering

ADSE Architectural Design Space Exploration

TRL Technology Readiness Level

MBSE Model Based Systems Engineering

SAO System Architecture Optimization

ADSG Architectural Design Space Graph

(P)HEV (Plug-in) Hybrid Electric Vehicle

UDG Undirected Graph

CP Constraint-Programming

CSP Constraint Satisfaction Problem

SoI System of Interest

MDA Multidisciplinary Design Analysis

IP Intellectual Property

I/O Input/Output

RBFN Radial Basis Function Networks

SMT Surrogate Model Toolbox

RBF Radial Basis Function

IDW Inverse-Distance Weighing

APU Auxiliary Power Unit

ICE Internal Combutions Engine

EM Electrical Machine

ODE Ordinary Differential Equation

ECM Equivalent Circuit Model

PyBaMM Python Battery Mathematical Modelling

SEI Solid Electrolyte Interface

RPM Rotations Per Minute

MDO Multidisciplinary Design Optimization

IDF Individual Feasible Solution

MDF Multiple Feasible Solution

CO Collaborative Optimization

ATC Analytical Target Cascading

KKT Karush-Kuhn-Tucker

SLP Sequential Linear Programming

SQP Sequential Quadratic Programming

BEGA Binary-Encoded Genetic Algorithm

NSGA-II Non-dominated Sorting Genetic Algorithm

REGA Real-Encoded Genetic Algorithm

GA Genetic Algorithm

PSA Particle Swarm Algorithms

MINLP Mixed-Integer Non Linear Programming

AMIEGO A Mixed Integer Efficient Global Optimization

EGO Efficient Global Optimization

OpenMDAO Open-source Multi Disciplinary Analysis and Optimization

NASA National Aeronautics and Space Administration

UMDO Uncertainty-Based Design Optimization

RBRDO Reliability-Based Robust Design Optimization

RDO Robust Design Optimization

RBDO Reliability-Based Design Optimization

PDF Probability Density Function

CDF Cumulative Density Function

MCS Monte Carlo Simulation

LHS Latin Hypercube Sampling

FOPM First-Order Pertubation Method

SA Sensitivity Analysis

XDSM eXtended Design Structure Matrix

DSM Design Structure Matrix

KADMOS Knowledge- and graph-based Agile Design for Multidisciplinary Optimization System

RCG Repository Connectivity Graph

FPG Fundamental Problem Graph

MDG MDAO Data Graph

MPG MDAO Process Graph

PIDO Process Integration and Design Optimization

OpenLEGO Open-source Link between AGILE and OpenMDAO

CDS Centralized Data Scheme

List of Figures

List of Tables

1 Introduction

As a result of the increased emissions of greenhouse gasses in the last 100 years, man kind is required to come up with new, more sustainable ways of transportation. One of the solutions found to be promising is the implementation of electric or hybrid drive trains into mobility vehicles which in turn minimize emissions due to a lack of hydrocarbon combustion. A leader in the field of e-mobility for heavy duty applications is VDL Special Vehicles. VDL Special Vehicles is a subsidiary of the VDL Group focused on implementing these E-mobility solutions in heavy duty on-/off-road vehicles. Since the design of such electric or hybrid drive trains is highly complex and does not result in a "one-solution-fits-all" design, VDL Special Vehicles requires an automated system architecture evaluation and optimization tool, which takes into account model- and input uncertainty in conceptual design to come up with a feasible, optimal design for which they can be certain that in later stages of design, this design will adhere to specified constraints while still being an optimal design. Taking into account uncertainty is something which has not been widely applied in the design and optimization of vehicle drive trains but is required due to the lack of system specific knowledge in conceptual design stages. This is something that has not been investigated in literature to date, and will form the basis of this literature review.

This report aims to provide a review of all relevant knowledge regarding the architectural design space exploration phase in conceptual design of electric power trains, the integration of such MDAO problem formulation into a dynamic framework, and methods to integrate uncertainty propagation into such MDAO framework. As the goal of this thesis is to integrate the above mentioned methods into a properly set-up software tool for VDL, the acquired knowledge from this literature review will form the basis of this research.

This report will first go over the Architectural Design Space Exploration and Evaluation methods in conceptual design as described in section 2. This section presents various methods to come up with an Architectural Design Space Graph which forms the basis of an MDAO problem formulation. It furthermore specifies various methods to evaluate such architectural design space graphs using simplified numerical evaluations tools and surrogate models. section 3 will go over the various components present in an electric drive train and their function within the system. It will present simplified modelling techniques used to evaluate the performance of the various components, given a set of input parameters. section 4 will first present the reader the state-of-the art knowledge on what MDAO is, and how a proper MDAO problem formulation is set-up. Then the implementation of uncertainty (input and model) into a MDAO framework is elaborated upon, by highlighting the strengths and weaknesses of various uncertainty propagation techniques, ranging from computationally expensive methods such as Monte Carlo Simulation, to more simplified first-order methods. section 5 will present the reader with multiple frameworks which can be used to dynamically and without human interference, generate workflows which can be directly executed in various PIDO platforms. The last chapter will provide the reader with an outlining of the project plan for the forthcoming thesis work. This section furthermore explains the research objective and presents a comprehensive timeline, with the goal of obtaining a successful completion of the thesis. Here, also the research (sub)questions will be presented, which will form the guidelines for the forthcoming thesis work.

2 Architectural Design Space Exploration and Evaluation in Conceptual Design

This section will present the reader with all relevant state-of-the-art knowledge on architectural design space exploration and evaluation. It will first provide the reader with an introduction to systems engineering and the hardware V-model after which it will go into more detail on the exploration and evaluation phases of designing a system architecture.

2.1 An Introduction to Systems Engineering

As a response to the growing complexity of modern technological systems and the need to address the intricate challenges they presented, the principle of systems engineering was developed in the late 1950s. By integrating various engineering, scientific and management principles into the engineering design process, it become possible to create efficient, reliable and cost-efficient systems.

Before systems engineering was first introduced in the 1950's, complex engineering projects suffered from various problems:

- **Inefficiency**: Engineers and teams often worked in isolation, resulting in suboptimal system designs.
- **Incompatibility**: Component and subsystem designs were frequently incompatible, leading to integration difficulties and costly rework.
- **Cost Overruns**: Complex projects frequently exceeded budgetary constraints due to unforeseen complications and changes, which were required to obtain feasible and desired engineering solutions.
- **Project Delays**: Projects were often delayed, affecting schedules and customer expectations.
- **Lack of Accountability**: Who carried responsibility for various aspects of a project was unclear, leading to confusion and errors.

Therefore, the need for a structured and holistic approach to tackle these challenges led to the development of systems engineering. By providing a systematic way of thinking about complex systems and its design, it tried to solve the the following key objectives:

- **Integration**: Systems engineering promotes the integration of all relevant disciplines, ensuring that various components and subsystems work seamlessly together.
- **Efficiency**: By identifying and optimizing inter-dependencies early in the design process, systems engineering helps improve overall project efficiency.
- **Cost Control**: Through rigorous planning, risk assessment, and continuous monitoring, systems engineering aims to prevent budget overruns.
- **Time Delivery**: Systems engineering emphasizes well-defined processes and clear milestones to ensure projects are delivered on time.
- **Accountability**: It establishes clear roles and responsibilities for each team member, enhancing accountability throughout the project lifecycle.

Systems Engineering finds its applications in a wide range of industries such as Aerospace, Automotive, IT, Healthcare and Infrastructure. In order to fully understand what systems engineering is and what is pro's and cons are, first a clear definition of a system has to be found. According to the International Council on Systems Engineering (INCOSE), a system can be defined as follows [38]:

> *A combination of interacting elements organized to achieve one or more stated purposes.*

Furthermore, the INCOSE defines systems engineering as follows:

> *Systems Engineering is an interdisciplinary approach and means to enable the realization of successful systems. It focuses on defining customer needs and required functionality early in the development cycle, documenting requirements, then proceeding with design synthesis and system validation while considering the complete problem.*
>
> *Systems Engineering integrates all the disciplines and specialty groups into a team effort forming a structured development process that proceeds from concept to production to operation. Systems Engineering considers both the business and the technical needs of all customers with the goal of providing a quality product that meets the user needs.*

As most engineering projects are driven by cost, cost budget adherence is a good way of measuring whether or not a project can be named a success. Systems engineering tries to tackle the above mentioned problems such that the total cost (time and money) of projects is kept within budget.

As every engineering project is dynamic, meaning that changes within the system design happen continuously throughout the design phases, a mistake-free project development does not exist. It is found that changes made to the system design, even in early stages of the design, become more and more expensive, looking at cost and time, while the project progresses. This is clearly visualized in Figure 1.

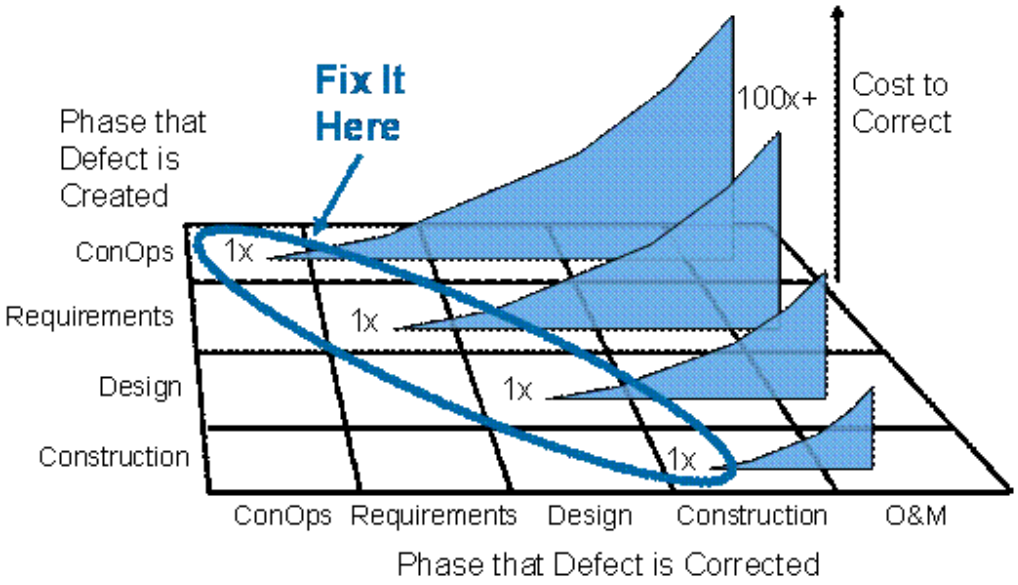


Figure 1: Late Changes Drive Project Costs [38]. Figure reproduced from Steve McConnell, Code Complete.

By solving all the above mentioned problems, the chance of budget adherence should increase. However from a study performed by the Standish Group in 1995 [13], it was found that still only 24 % of all projects met the criteria for success, meaning it was completed on time, on budget and with all the features originally specified. This study also showed that most of the system engineering related topics formed the basis of successful projects. Therefore, it can be concluded that even though applying proper and thorough systems engineering approaches to engineering projects might not result in guaranteed success of a project, it will help you to address issues early on in the project schedule. This will eventually result in improved changes of project success.

2.2 The V-Model in Hardware Systems Engineering

In hardware systems engineering, a common and widely used tool in the industry is the V-model, as first proposed by the US Department of Transport [38]. It emerges in industry as the de facto standard way to represent a systems engineering process.

The V-model segments every stage in the engineering design process based on the requirements that are set in that specific design phase, as well as the level of knowledge that is available at that current time. A revised version

of this V-model is presented by Bruggeman et. al. [7] which shows a clear distinction between system design and system verification/validation.

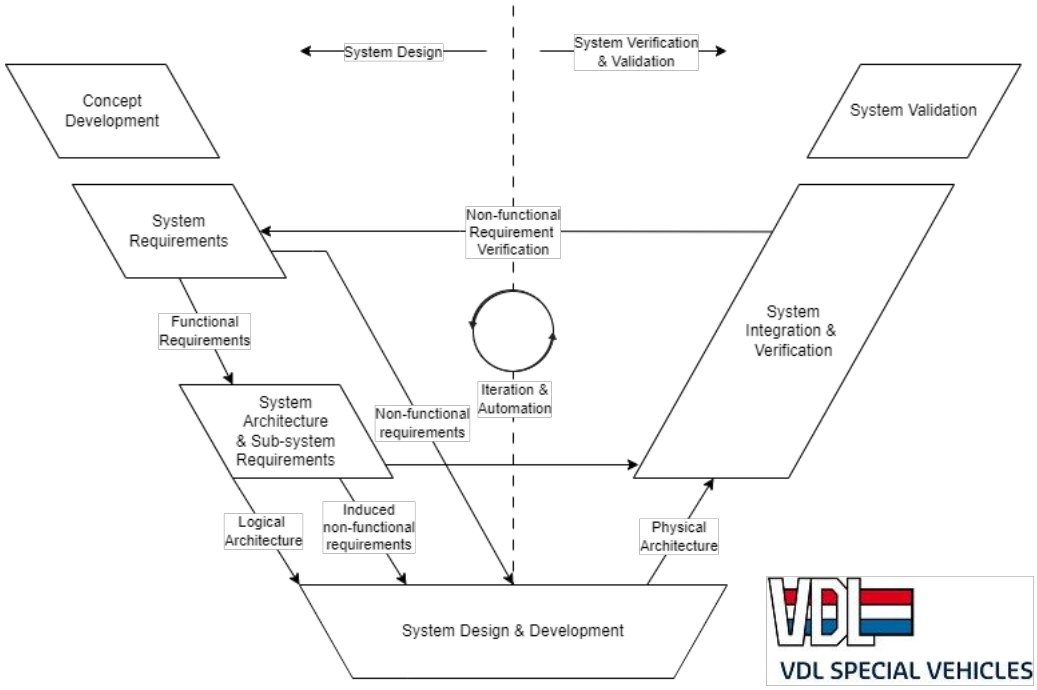


Figure 2: A modified version of the V-model as presented in [7], applicable to standard hardware development as commonly used within industry.

Here it is shown, that one of the first steps in a systems engineering process is the determination of a system architecture. A system architecture can be defined as a set of components or subsystems and the synergy (i.e. flow of information, mass or energy) between them, which combined can perform a set of functions (qualitative requirements). A method common in systems engineering used to find the best possible system architecture for a given set of functional requirements, is Architectural Design Space Exploration (ADSE). Architectural Design Space Exploration is a method used to explore all or several different system architectures with the goal of coming up with a list of (sub)systems or components, which combined, adhere to the set of functional requirements [9] [14]. As a vehicle architectural design space is highly multidimensional, finding an optimal design solution can become challenging if using conventional sequential design methods.

2.3 Architectural Design Space Exploration

Conventionally, within systems engineering, architectural design space exploration is conducted to evaluate, both quantitatively and qualitatively the various options for design. This is in most cases done by first obtaining a lot of data on what is currently commercially available, what is the standard from industry, what are the constraints that follow from choosing such options, and what is the synergy with other components that is induced from it and its effect on the overall performance of a system. Lastly, also the risks of certain design options that come from choosing such options are to be qualitatively evaluated. This way, engineering teams make sure that also cost and time budgets are adhered to.

In architectural design space exploration, it can become challenging to find all possible design options, as human interference is still required. This would mean that a system cannot think for itself and determine possible solutions to problems which are not yet fully quantified and for which constraints are not formally set. Furthermore, it is still highly unclear what the effect of certain design options are on the overall system performance and what requirements are induced choosing a certain solution. This is found to be extra challenging for the implementation of innovative technologies, as the lack of knowledge in for instance safety of systems and logistic constraints result

in a high level of uncertainty. In engineering design, therefore, a clear distinction can be made between derivative- and innovative design. Here derivative design focuses on the reuse of existing systems architectures, thereby maximizing reuse of concepts which results in a decrease in development cost, delays and risk. Here however, it can be said that there is limited design space exploration. Innovative design focuses mostly on the exploration of much larger design spaces, by abstracting previously designed systems or radical designs. The limitations on the design space are thereby based on the outcome of the objective and the preference based on given configurations, i.e. a mapping of the preference onto the design space [1]. This constraining of the design space is visualized in Figure 3

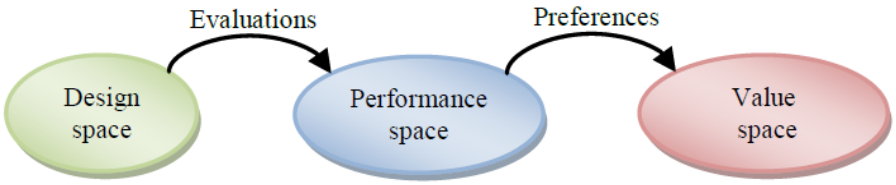


Figure 3: Mapping the preference of a design in the form of constraints to obtain the value space [1]

Architectural design space exploration is mostly focused on qualitatively comparing different architectures based on for instance Technology Readiness Levels (TRL's), (in)compatibility constraints etc. This is explained in Amadori et al. [2], as the data collection part of conceptual design, which in turn is highly subjective to expert bias and where limited traceability is present.

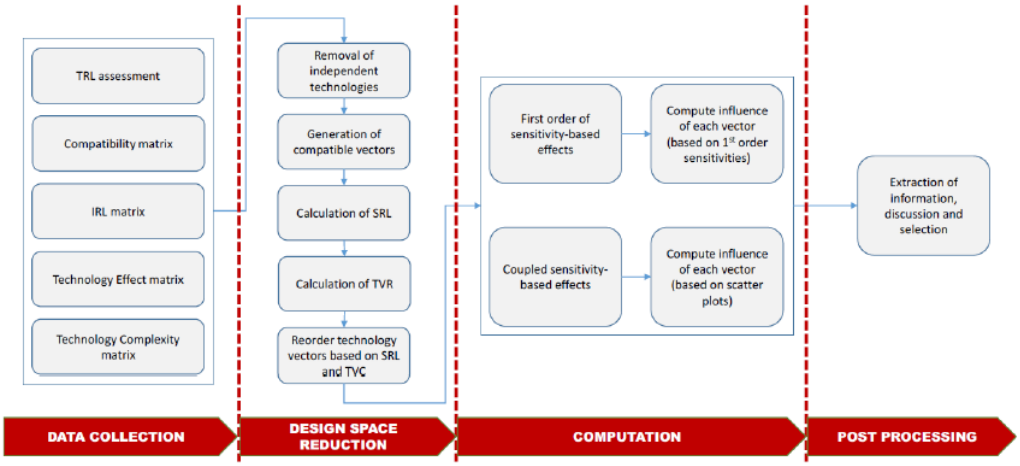


Figure 4: Methods of comparing qualitative data to come-up with feasible system architectures in conceptual design [2]

A method for finding possible architectural design options is called Morphological Matrix Enumeration and Reasoning. This method is still highly subjective to human interference, as it is limited by the system's engineers knowledge obtained during data collection.

2.3.1 Morphological Matrix Enumeration and Reasoning

The morphological matrix method is a highly used and popular method for conceptualizing and initiating the design space for a given problem. It was first introduced by Weber & Condoor in 1998 [72].

The method works by first listing all possible solutions, which are known to the system engineer, for the different functionalities in a system. In other words, the forms of a function are enumerated [9]. The function here specifies what a system should do, while the form is a way of choosing how a function is fulfilled. For this, first independent system functions should be identified, i.e. the functions a system should perform to meet design requirements.

Then, an enumeration is performed on all possible known solutions to fulfill each independent system function, which are added to a morphological matrix. An example of a morphological matrix for aircraft architectural design space exploration is given in Figure 5

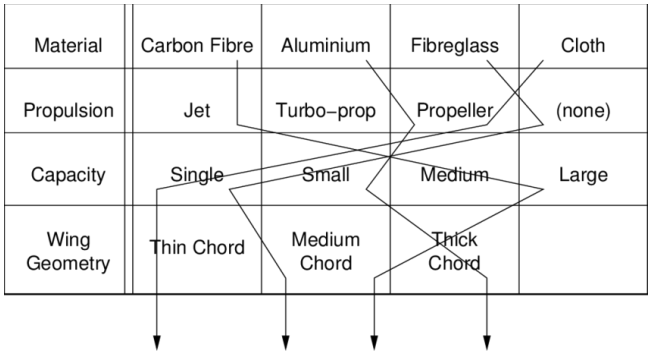


Figure 5: An example of a morphological matrix for aircraft architectural design space exploration [47]

Simmons [62] specifies the need for a mutually-exclusive solution strategy, where for each function, only one solution can be included in an architecture. They mention that if combinations of solutions are possible, these should be added as options on their own. What is limiting in this approach is that currently, with the introduction of more hybrid approaches, to combine strengths and minimize weaknesses of different solutions, multiple solutions can provide multiple functionalities. This multi-functionality approach is not well integrated in their proposed methodology, thereby already limiting the conceptual design space.

As mentioned above, currently, system engineering is becoming more and more focused on multi-functionality of systems, meaning that one solution carries more than one function. Therefore, the morphological matrix has been numerously extended to include ways to identify (in)compatibility and or synergism [9].

2.3.2 Architecture Generation using Functional Decomposition

In order to qualitatively build and construct different architectures, it can be useful to model the basic functionalities of a system and from that, determine what *form* could be able to resolve that functionality. Since a certain form would also require its' own functionality, certain functions are deducted from this. The build up of an architecture based on the functionalities that are derived from the decomposition of systems, is called functional decomposition. The method is widely used in literature as a way of finding alternatives to different functional requirements and thereby constructing various architectural options. There are many ways of decomposing a system, ranging from more generic to more domain-specific. Note that domain-specific methods of decomposition include to a certain level some solution- and experience bias, which is undesired. [9] Some benefits of using functional decomposition are [9]:

- It provides a breakdown generic to any architecture alternative
- It is free of solution-bias
- Functions suggest types of solutions rather than specific technologies.

A more generically applicable systematic approach for system architecting is the dynamic mapping of function to form. This method allows for easy integration with Model Based Systems Engineering (MBSE), which will be explained further in subsection 2.4. Furthermore it allows for proper tractability, be applicable to any kind of system architecture and be solution-bias free.

The mapping of function to form is implemented widely in literature. Here functions define what the required capabilities are that a system has to perform. Based on this, components are searched to find a suitable solution which fulfills these required capabilities. Based on a component's required inputs and outputs, and the direction with which this information flows between functional instances (components), the functional flow can be computed [1]. Based on the selected components, multiple functions can be induced from the instantiating of the component. This is called function induction. Other methods work using function trees, where solutions are iterative found by finding

function-solutions pairs, to obtain a system architecture [32]. This however, requires extensive human interference to set-up known function-solutions pairs, as well as highly consistent nomenclature to allow the algorithm to couple the various function-solution pairs. For example, a pump requires electrical energy to function. A gas turbine is able to provide power. However, coupling a gas turbine to provide power to a pump is highly unfeasible and undesired. Therefore, a clear distinction has to be made to for instance specify, whether it requires high power or low power. This in combination with clear constraint (preference) formulation would make the algorithm highly complex.

Also Bussemaker et al. [10], proposes a method based on an architecture function-based representation, combined with system modelling to allow for optimization within the architectural design space. They present a method to build up a graph which represents the total design space where the different elements in the graph represent:

- **Function-component mapping**: by assigning what component can fulfill a certain function, different components can be found and combined to alter system architectures
- **Component characterization**: by assigning attributes and for instance the number of instances, a specific instance of a component can be made.
- **Component connections**: Assign ports to the in- and outputs of components to model the functional flow of for instance signal, mass or energy.

This graph can then be altered by removing or fixing certain decision nodes, to obtain an instance of a system architecture. This can then be evaluated and eventually be optimized to find an optimal solution which satisfies the given set of requirements. An example of a so-called Architectural Design Space Graph (ADSG) is given in Figure 6. Note that such an ADSG can easily be converted in a MDAO problem formulation, given the set of decision-variables.

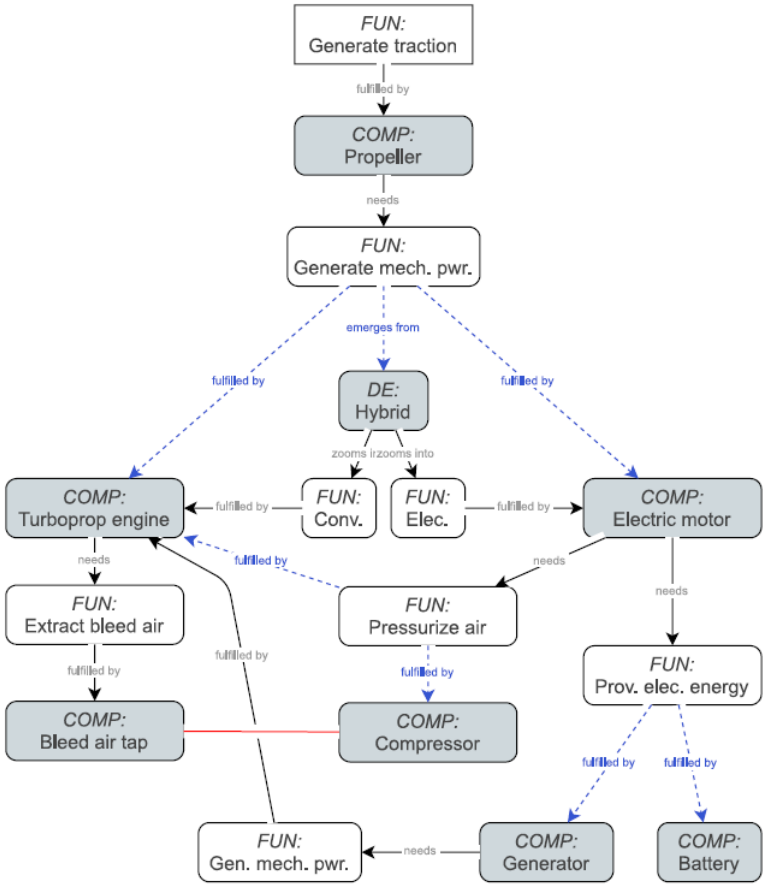


Figure 6: An example of a Architectural Design Space Graph (ADSG) of Hybrid Electric Aircraft [9]

2.3.3 Automated Architecture Generation using CSP

The above mentioned methodology as described by Bussemaker [9], still requires a set-up procedure in which human interference is required. This is due to the fact that every conceptual design problem is highly dependent on what is to be designed and to what requirements it needs to adhere. It furthermore is still difficult and unclear what the objective of the architecture evaluation is, or how architectures can be compared quantitatively / qualitatively. Another limitation to this method is the incompatible of certain components, which can not be derived from the simple boolean design vector of components.

A more holistic approach to architectural design space exploration applied to hybrid vehicle drive train evaluation and optimization, is proposed by Kabalan et al. This methods combines architecture generation with architecture evaluation based on simple surrogate models [33]. They apply their methodology to the design of hybrid power-trains, which are highly intricate systems involving a large number of design variables, at multiple design levels. They split these design choice levels accordingly:

- **System Architecture**

  Series Hybrid, Parallel Hybrid, Series-Parallel Hybrid, Plug-in Hybrid

- **Component Technologies**

  LTO, LFP, NCA, NMC Battery Technology

- **Component Sizing**

  Size of battery pack, Number of cells in series/parallel, Number of modules

- **Control System**

  On-Off Control, Constant Charge

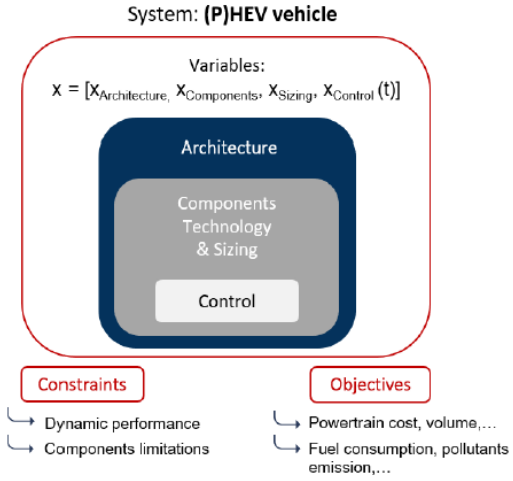


Figure 7: Design problem for the determination of a conceptual PHEV drive-train on all four levels. Figure from [33].

Based on Figure 7, it can be concluded that the design of a PHEV is a multi-objective optimization problem, which takes into account the high dimensionality of the design problem (multiple levels) [33].

In order to generate multiple different system architectures, Kabalan [33] starts by first defining the list of components which can be used, and the vehicle specifications (system requirements (both functional and non-functional)). By modelling a power train as an undirected graph (UDG), based on components (nodes), and connectors (edges), a simple but systematic methodology for finding different configurations is derived. Here the components library can be split up into various types of components, ranging from simple power train components (limited to one connection), to more general transmission components (which allowed to have more connections). In order to reduce the level of complexity of a gearbox, only a 2-speed gearbox was taken into account. An example of a simple series-parallel architecture is given in Figure 8 More complex methods which integrate multi-speed gearboxes using graph theory are assessed by Masfaraud et al. [46].

Furthermore, other extensions have been added to this method to account also for automated constraint generation through a user-provided knowledge base and the generation of multi-domain (electrical, mechanical, hydraulic or signal) discrete system topologies [37]. Here the connecting edges of the various components (nodes) have attributes, which allow it to only be connected to edges with similar attributes. These attributes are specified in *edge-domains*.

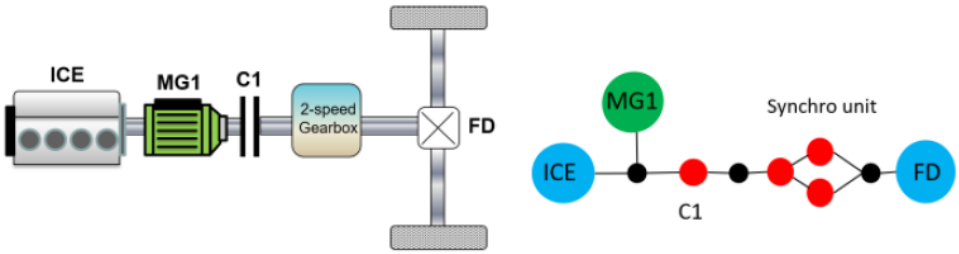


Figure 8: An example of a parallel architecture with a 2 speed gearbox [33]

In order to find all possible design options, the undirected graph can be described as a set of components (**V**) and connections/edges (**E**), which make up the graph **T**.

$$\begin{aligned} & \mathbf{T} = (\mathbf{V}, \mathbf{E}) \\ \text{with} \quad & \mathbf{V} = \{V_{11}, V_{61}, V_{31}, V_{51}\} \\ & \mathbf{E} = \{\{V_{11}, V_{61}\}, \{V_{61}, V_{31}\}, \{V_{31}, V_{51}\}\} \end{aligned} \tag{2.1}$$

Where the set of all possible topologies is denoted as $\mathbf{T}^p$, the feasible topologies as $\mathbf{T}^{fe}$ and the infeasible topologies as $\mathbf{T}^i$ [61].

Then, a Constraint-Programming (CP) algorithm is used to compute all the possible system architectures (components & connections). Constraint programming is a method used to solve complex combinatorial problems. By allowing the algorithm to define all possible solutions, and adding constraints to the problem definition, a set of solutions which satisfy these constraints is found. Such algorithms rely heavily on systematic search algorithms to explore the broad range of possible solutions. The constraints are modelled as either functional constraints or cost constraints (mapping of preference onto the design space [1]):

$$\begin{aligned} \text{Find all} \quad & \mathbf{T}^{fe} \subseteq \mathbf{T}^p \\ s.t. \quad & \mathbf{c}^f_{1,\cdots,l} \subseteq \mathbf{C} \\ & \mathbf{c}^c_{l+1,\cdots,z} \subseteq \mathbf{C} \\ where \quad & \mathbf{C} = \mathbf{c}^f_{1,\cdots,l} \cup \mathbf{c}^c_{l+1,\cdots,z} \end{aligned} \tag{2.2}$$

Here, **C** represents the complete set of constraints, consisting of $\mathbf{c}^f_{1,\cdots,l}$, representing the functionality related constraints, and $\mathbf{c}^c_{l+1,\cdots,z}$, representing the cost related constraints.

The formalization of constraints can be explained by first modelling a Constraint-Satisfactory Problem (CSP) as a set $< \mathbf{X}, \mathbf{D}, \mathbf{C} >$, where **X** is a finite set of variables, **D** is the set the variable specific domains and **C** is the set of constraints, which is real and finite. A constraint C_{ijk} between variables $X_i, X_j, \cdots, X_k$ is any subset of possible combinations of values of $X_i, X_j, \cdots, X_k$ [61]. In other words,

$$C_{ijk....} \subseteq D_i \times D_j \times D_k \times \cdots \tag{2.3}$$

To use this methodology more geared towards system architecture generation, the variables and domains can be specified as follows:

$$\mathbf{X} = \mathbf{V} \cup \mathbf{E} \qquad \mathbf{D} = \{0, 1\}^{|\mathbf{V} \cup \mathbf{E}|} \tag{2.4}$$

Functional constraints are integrated into the CSP by either managing graph consistency, power-train hybridization and components and subsystems correct functionality [61]. Cost constraints are then added to further decrease $\mathbf{T}^{fe}$ by removing redundant design solutions or undesired topologies. As a last check, isomorphism has to be detected to remove similar or equivalent system architectures from the set of feasible topologies. As this can result in similar, but for instance mirrored, system architectures. As an example, 4 components are connected to 2 virtual nodes, which are simply connecting points which can in-/output multiple edges. Due to isomorphism this results in *4!* similar solutions, all representing the same system architecture. This shows that the check for isomorphism in graph theory is very important to achieve a small enough set of possible, but different, topologies or system architectures.

The final set of functional and cost effective solutions can then be further evaluated to come up with the most feasible design solution. Methods to do this will be explained in subsection 2.4.

2.4 Architectural Design Space Evaluation

The set-up and evaluation of an architectural design space exploration in conceptual design, is highly dependent on expert bias, subjectivity and conservatism, as mentioned by Roelofs & Vos [2]. Reasons for this is the lack of clear quantification of performance metrics of a System of Interest (SoI) in conceptual design stages. Furthermore other, limitations to the conventional conceptual design space exploration and evaluation are:

- Non-performance metrics on non-quantifiable system traits
- Technology descriptions are not yet meaningful and cannot traverse from detailed to high-level descriptions, nor capture quantifiable effects and enabling fair comparison between technologies
- Finding the best technology portfolio without enumerating all possible combinations cannot yet be done objectively.
- The assessment of dependencies between technologies is too subjective
- When uncertainty is quantified, decision making is impaired in case of wide uncertainty bands

Since it can become challenging, based on the set of non-functional requirements already set in the first step of the V-model, to come up with an optimal solution, usually basic numerical models / empirical models are used to quantitatively evaluate different design options. These are usually back-of-the-envelope calculations and result in a highly basic, non-optimal sizing and quantification of the performance characteristics of a complex system. A current trend in Multi Disciplinary Analysis (MDA) is the integration of mid- to sometimes even high-fidelity models into conceptual design stages, to generate higher confidence levels on the performance of a certain system [34] [36] [41]. This way, already in early design stages, a clear understanding on the behavior of a system can be found. Furthermore, with the computational capabilities of current computers, even these high-fidelity systems can easily be evaluated.

As the architectural decisions made in early design stages, such as conceptual design, highly effect the performance of the final design, the reason for applying a broad and extensive architectural design space exploration is clear. By allowing the exploration to investigate all possible system architectures, and evaluating them using simple evaluation methods, a conclusion can be drawn on what type of system architecture is most desirable for a given set of system requirements. A downside to this methodology is that in early design stages, very little is known about behavior of a certain component or its synergy in combination with other components. While progressing through the design stages, the knowledge about the component- and system behavior becomes more advanced, but changing system architecture becomes challenging or even impossible based on time or cost restrictions. This is referred to as the knowledge paradox, as first proposed by La Rocca [39]. The knowledge paradox, is shown in Figure 9

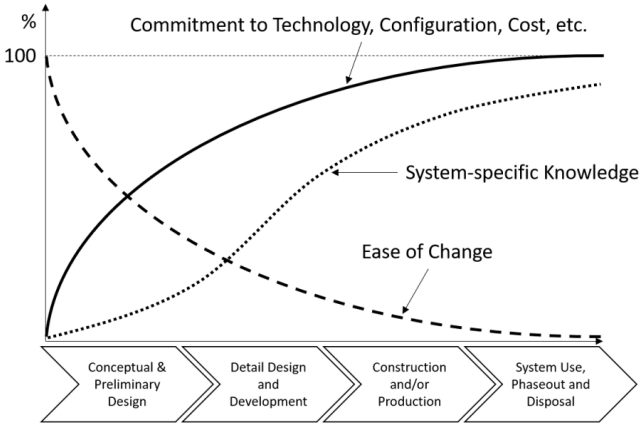


Figure 9: The knowledge paradox [39]

Currently, limitations in knowledge in early stages of the design process are solved in industry by applying expert judgement and experience to solve for decision uncertainty. As a result, these decisions suffer from expert bias, subjectivity, conservatism or overconfidence [55]. By allowing this expert bias, conservatism and subjectivity to determine the system architecture, a real understanding and exploration of the entire design space and its opportunities is missed. By allowing the utilisation of computer aided computation to thoroughly explore the design space, without bias or subjectivity, but based on simple quantification of objectives, a real true optimal solution based on a set of requirements and needs can be found.

In order to properly evaluate all possible system architecture candidates, which are generated using algorithms described in subsection 2.3, quantitative models should be integrated to define system performance. In literature, this method of quantitative evaluation of systems or components is referred to as Model Based Systems Engineering (MBSE). Here, every component or (sub)system is modelled using simple analytical models, surrogate models or black boxes. To an optimizer, the only information required is the flow of information at the in- and output ports of the model, and some additional values which build up the model characteristics [28]. Model Based Systems Engineering has proven itself in recent years to be highly effective in determining basic characteristics of components and architectures at early stages of design. Especially the integration of MBSE in control system design is something which is investigated extensively. With the possibility to access closed-loop system's behavior in the beginning of the design cycle, combined optimization of an architecture and it's control system can occur.

2.4.1 Architecture Modelling and Evaluation using SysML

Methods to model different types of architectures are usually in MBSE based on the SysML language [29]. SysML was first proposed by Holt & Perry [29], and is a modelling framework which shows dependencies between various components within a system architecture. The SysML modelling language models the synergy of different components through coupling of the flow of information (energy, power, mechanical, mass, information etc.) in a system architecture. Within the standardized SysML framework, a user is able to implement certain parametric modelling features, which allows for simple requirement validation.

A downside of the SysML modelling language is the lack of advanced system simulation and evaluation capabilities [9]. Therefore, a request was made to integrate modelling capabilities into the SysML modelling framework. Several attempts were made to convert the systematic model of an architecture, with its attributes, to an executable system evaluation. One example of this is the SysML4Modelica toolbox [50], which allows a SysML model to be converted to a Modelica model, which can be quantitatively be examined. The integration between SysML, SysML4Modelica and Modelica is given below:

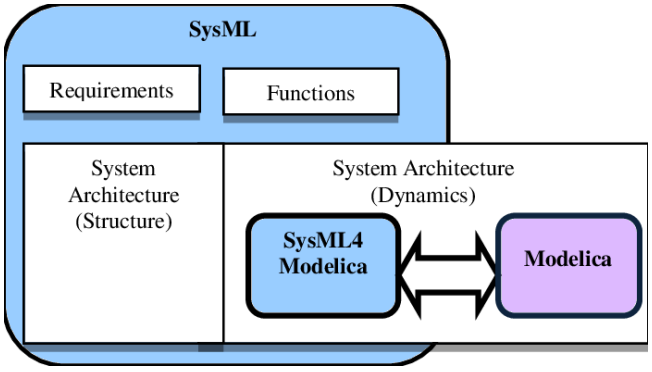


Figure 10: A visual representation of the connection between SysML, SysML4Modelica and Modelica [50].

An example in which the SysML4Modelica toolbox has been applied to conceptual aircraft design, is proposed by Guenov et al. [25] [26]. Here, a simplified Design of Experiments can be applied to a range of design variables, for which simple mathematical models are present. The AirCADia uses visual inspection to determine the feasibility of the design space. By then allowing further design space constriction to occur, through constraints, an overall system optima was found. AirCADia is a simplistic framework which shows the potential of using SysML4Modelica for conceptual aircraft design space exploration.

Other, commonly applied methods for qualitatively evaluating system architectures is by means of comparing energy efficiencies or by performing a complexity analysis based on the number of components [33]. Downsides to these methods are that they only partially model the behavior and expected performance of a system architecture.

2.4.2 Surrogate Modelling

More recently, the implementation of surrogate models to find model characteristics without having to run computationally expensive numerical models, was introduced. Surrogate models, also known as metamodels or approximation models, are powerful mathematical or computational representations that emulate the behavior of complex real world systems. The fast and computationally light evaluation of a surrogate model allows for efficient integration with optimization algorithms, as repetition of these numerical simulations can otherwise be time-consuming or costly.

A surrogate models works by approximating the input-output (I/O) relationship of underlying complex systems, thereby making it possible to predict (with some accuracy) the system output for new inputs. There are various forms of surrogate models:

- Regression models
- Neural Networks
- Support Vector Machines
- Gaussian Processes
- Decision Trees
- Basis functions

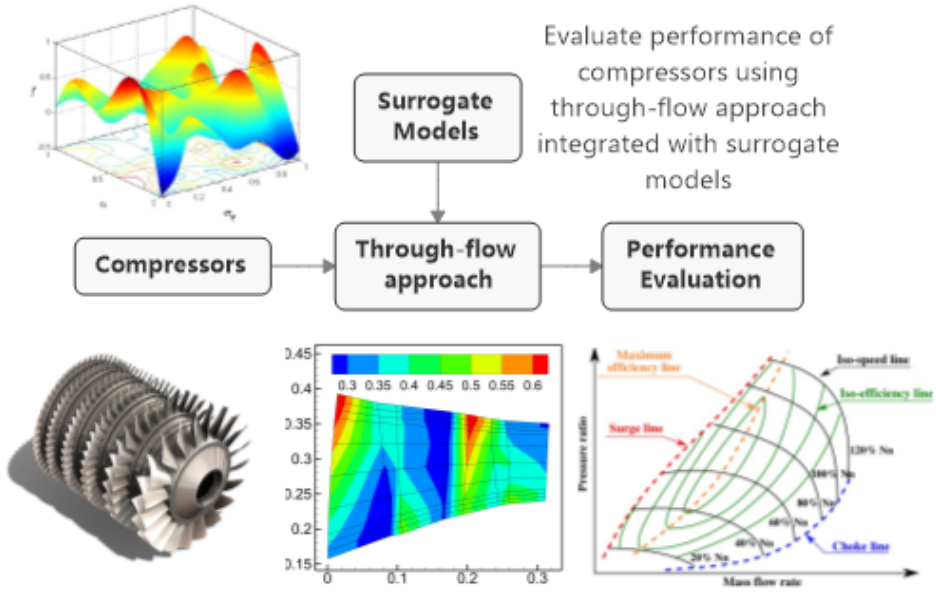


Figure 11: A visualization of a surrogate model applicable to determining the performance of axial compressors. Figure reproduced from [74].

The level of accuracy of a surrogate model is a direct result of its training process. As surrogate models use sampling of the training data, for which the output is known, the accuracy of the in- and output data determines how well a surrogate model will perform. Furthermore, it is estimated that an increase in training data will result in a higher level of accuracy of the model, thereby modelling the relation between input and generated output more accurately and true to nature. This can be explained by the Central Limit Theory and the law of large numbers. Some downsides to surrogate modelling however are:

+ **Approximation Errors**: Surrogate models are, by definition, approximations of the original system. They may not capture all the complexities and nuances of the real-world process, leading to approximation errors.
+ **Over-fitting**: Surrogate models can suffer from over-fitting, where they fit the noise in the training data rather than the underlying system behavior.
+ **Data-dependency**: As the quality of the model can only be represented by the quality of the data that is extracted from it, it might become troublesome whenever limited, biased or unrepresentative data is present to train the surrogate model.
+ **Limited Interpolative Range**: Some surrogate models which are dependent on regression models, have a limited interpolative range meaning they may not perform well for inputs that are significantly outside the range of the training data (Outliers).
+ **Model Selection**: It can become cumbersome to choose the right model and hyperparameters to mirror the behavior of the system as accurately as possible. Especially in the case when limited information on the physics underlying the problem are scarce (Black-box).
+ **Uncertainty Estimation**: While some surrogate models can provide point predictions, estimating the uncertainty or confidence intervals in their predictions can be challenging, limiting their utility in decision-making under uncertainty.

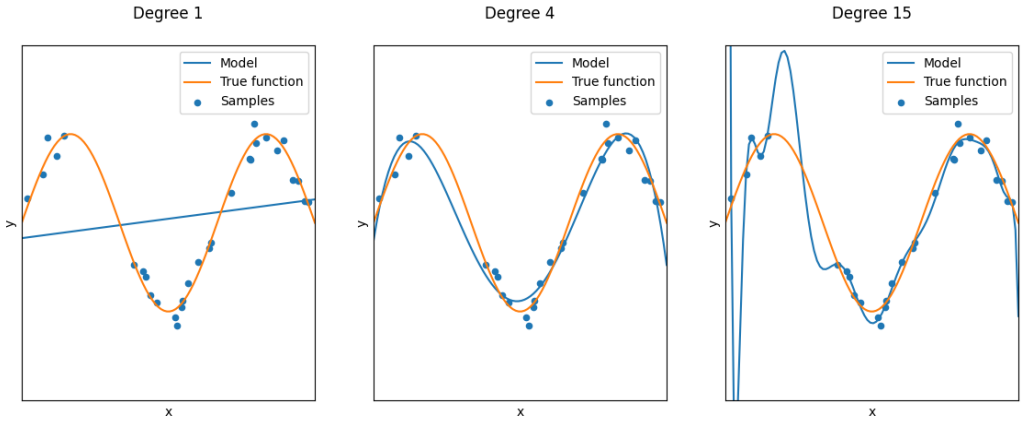
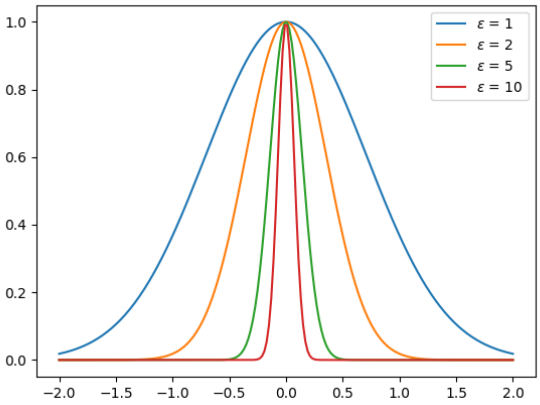


Figure 12: A visualization of the principle of over-fitting, where the system starts modelling the noise of data, instead of the trend [74].

There are various methods to generate surrogate models based on training data that is present. Below, the most used methods are in detail described to show their strengths and weaknesses, based on the available data and use case.

Radial Basis Function Networks (RBFN's) Radial Basis Function Networks is a method used for obtaining an unknown function value by approximating the value as a linear combination of radial basis functions and weights (Equation 2.5). Based on the set of input data, which are called origins or center points , RBFN will approximate a function value based on the distance between an input point and the center points, for a given Radial Basis Function (RBF). The most commonly applied distance here is the Euclidean distance. In most SMT toolboxes, the Gaussian Infinitely Smooth RBF is implemented [6],



$$y = \sum_{i=1}^{N} w_i \gamma(||\mathbf{x} - \mathbf{x}_i||) \tag{2.5}$$

$$\gamma(||\mathbf{x} - \mathbf{x}_i||) = e^{-\epsilon||\mathrm{x}-\mathrm{x}_i||} \tag{2.6}$$

Figure 13: The Gaussian Radial Basis Function for a range of epsilon values

Inverse-Distance Weighing Inverse-Distance Weighing (IDW) is a somewhat simpler type of multivariate interpolation, where the to be determined function value is calculated through weighted averaging of the values available at the known points.

The IDW-interpolation function can be give described using the following equations:

$$u(\mathbf{x}) = \begin{cases} \frac{\sum_{i=1}^{N} w_i(\mathrm{x}) u_i}{\sum_{i=1}^{N} w_i(\mathrm{x})}, & \text{if } d(\mathbf{x}, \mathbf{x}_i) \neq 0 \text{ for all } i \\ u_i, & \text{if } d(\mathbf{x}, \mathbf{x}_i) = 0 \text{ for some } i \end{cases} \tag{2.7}$$

Here u_i is the known value for the interpolation function, as the location is similar to a location specified in the training data, hence the distance from the to-be-evaluated point to that known point is zero. For all other points, a weighted average of the known points is used, where the weights are assigned based on the distance from a to-be-evaluated point to a known point. As the weight of the point decreases with increasing distance, this is called inverse-distance weighting [60]. The equation to calculate the given weights is shown below, where p is the power parameter which is up to the user to determine:

$$w_i(\mathbf{x}) = \frac{1}{d(\mathbf{x}, \mathbf{x}_j)^p} \tag{2.8}$$

Kriging Kriging is a method of spatial interpolation which finds it origin in the field of mining geology, first developed by mining engineer Danie Krige.

This approach leverages a limited dataset of sampled data points for estimating variable values within a continuous spatial domain. By exploiting the spatial relationships among these sampled points, interpolation is used to derive values across the entire spatial field. Additionally, it provides uncertainty assessments, a valuable feature considering that surrogate models are inherently approximations and non-deterministic [53]. Kriging is found to be most effective whenever there is considerable spatial autocorrelation.

The methods works by assigning weights, so-called kriging weights, to all data points present within the data set. The magnitude of these kriging weights is dependent on the location and distance of the sample point and the location of interest. This means that nearby points have a higher weight than those farther away. The kriging method also takes into account clustering of points, resulting in lower overall weights which are clustered together, as the sparsity and randomness of sampled data points should not affect the overall accuracy of estimation. The to be determined interpolation, based on the assigned weights, is then evaluated at the sampled data locations, to determine the prediction error at that point. Obtaining these interpolation is based on a two-step process:

1. Initially, it involves establishing the spatial covariance structure of the sampled points through variogram fitting. A variogram is a visual depiction of the covariance between a pair of sampled points plotted versus distance.

2. Subsequently, this covariance-derived information is used to assign weights for the purpose of interpolating values at unsampled locations or within spatial blocks across the field.

Kriging can in mathematical terms be explained as a linear combination of a known function $f_i(\mathbf{x})$, the kriging weights β_i and some stochastic process $Z(\mathbf{x})$, as shown in Equation 2.9 [58]:

$$\hat{y} = \sum_{i=1}^{k} \beta_i f_i(\mathbf{x}) + Z(\mathbf{x}) \tag{2.9}$$

The stochastic process $Z(\mathbf{x})$ has a zero mean with spatial covariance function given below, where σ^2 is the process variance and R is the correlation function.

$$Cov\left[Z\left(\mathbf{x}^{(i)}\right), Z\left(\mathbf{x}^{(j)}\right)\right] = \sigma^2 R\left(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}\right) \tag{2.10}$$

Based on the covariance, a correlation function can be defined. A number of correlation functions are present in standardized surrogate model toolboxes [6].

An illustration of a 1D data interpolation using kriging is given in Figure 14. Here, also the credible confidence intervals are visualized based on the standard deviation selected. The kriging interpolation, depicted by the red line, aligns with the means of the normally distributed standard deviation intervals.

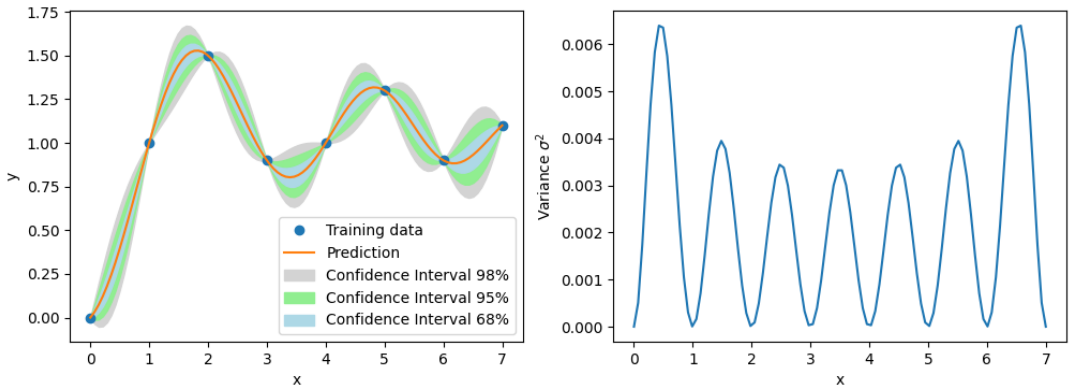


Figure 14: A 1D interpolation problem solved using kriging [6]. Also the the variance is plotted.

2.5 Key Takeaways

1. Methods to allow for Architectural Design Space Exploration have been extensively studied in past literature, resulting in various techniques, ranging from more simplistic to more computational extensive methods. A first step usually revolves around finding all relevant options and combing them, if compatible, in a Morphological Matrix. A more generic method explained as functional decomposition can also be used which goal is to find which form fulfills a certain function, and from that specify the various system architectures.

2. Functional Decomposition can be used to find the most generic function a component has to fulfill, thereby not requiring deep understanding of the synergy of components. The multi-functionality of components can be difficult to find using functional decomposition, but is required in more complex engineering systems.

3. A more computational approach to implement a Morphological Matrix is by solving a Constraint Satisfactory Problem (CSP) to find all compatible components. From this, filtering can be applied using for instance proxi metrics, to find the most feasible set of system architectures. This filtering can be explained as a sort of mapping of compatibility and preference onto the design space.

4. A method to automatically generate multi-level system architectures for a given set of components, and its I/Os is presented by Kabalan et al. [33] and Kort et al. [37]. Here a unidirected graph is formulated between all specified components, to which constraints are added (both functional and cost or preference). Then Constraint Programming is used to find the constraint satisfied system solutions. By automating this process, a lot of feasible system architectures can be found with limited computational time and cost.

5. In order to evaluate the various architectural design space options, various techniques have been successfully implemented in past literature, for instance SysML4Modelica and surrogate models.

6. Surrogate models can be used to quantitatively evaluate the behavior of a complex system at low cost, in exchange for a marginal reduction in accuracy. By sampling and training a surrogate model based on a complex model, a simplistic computational approximation can be generated using for instance the method of Kriging, Radial Basis Functions (RBF) or Inverse-Distance Weighing (IDW).

3 An Overview of the Design of Vehicle Power Trains

This section will present the reader with all relevant state-of-the-art knowledge on the design and modelling of the various components and (sub)systems within a sustainable vehicle drive train. It does this by first presenting the various system architectures of hybrid or electric vehicles, after which more detail will be given on the modelling tools used to predict the performance of such power train components.

3.1 VDL Special Vehicles

VDL Special Vehicles is a specialist in the field of battery electric and hydrogen solutions for on- and off-road vehicles. As a contract manufacturer, it provides production capacity for small to medium-sized series of zero-emission vehicles. In addition to that, VDL Special Vehicles can provide support in areas such as product development, prototyping and certification of zero-emission drive lines, up to the extent of fully developing an alternative drive line for a specific application. VDL does this with a strong basis in the development and production of chassis for busses and coaches in a variety of applications. Since VDL Special Vehicles builds either using Build2Print or Build2Spec, it integrates innovative technologies and solutions into existing third-party vehicles.

Since this thesis work will only focus on improving the requirement risk management of VDL's Build2Spec vehicles, where a customer comes in with a set of requirements and needs, and VDL converts this into a feasible design solution, no further detail will be given on Build2Print applications.

In architectural design space exploration, which mostly occurs in conceptual design stages (see Figure 2), proper modelling of the various drive trains is required to evaluate options and perform trade-off studies. Based on this, from a performance perspective, the system integrator is able to decide on what type of drive-train solution is preferred given the set of requirements as previously specified. As previously explained, requirements can be divided into functional and non-functional requirements. These non-functional requirements quantitatively specify some performance metric of a (sub)system. Conventionally, back-of-the-envelope calculations are done to get an simplistic overview of what is required to pass these non-functional requirement checks. Recent trends focus on implementing more advanced calculation methods and simulation tools to better grasp the physics underlying these drive train components. Combining that we the increased computational power and decreased computational cost of current standard computers, these simulation tools can already be afforded at an early stage in the design process.

As most literature on vehicle system architectures focuses on evaluating or optimising a specific type of drive train [42], or only on the mechanical aspect of these drive trains [46], a more general approach taking into account also electric components in hybrid or electric drive trains is missing. These elements will be further discussed in the subsequent subsections.

3.2 Hybrid Drive Train Overview

Below an overview is given on the various drive train options which are most common in industry. The system architecture of a hybrid electric vehicle is not new nor state-of-the-art, as a lot of vehicles have been designed using this philosophy. However, various combinations have been made to utilize the strengths and limit the weaknesses of each hybrid power train type. Please note, that the below given types of hybrid drive trains do not specify what type of electric energy storage is used. subsubsection 3.2.4 will go into a comparison of the different drive trains.

3.2.1 Series Hybrid Electric Vehicles

Series hybrid vehicle architectures work by means of an auxiliary power unit (APU) providing excess energy to the main power source to extend it's driving range. Here, usually, the main power source is a battery pack. A schematic of a series hybrid electric drive train with brake resistor is given in Figure 15. Please note that the brake resistor is added, as this is required for regulatory compliance in VDL's trucks.

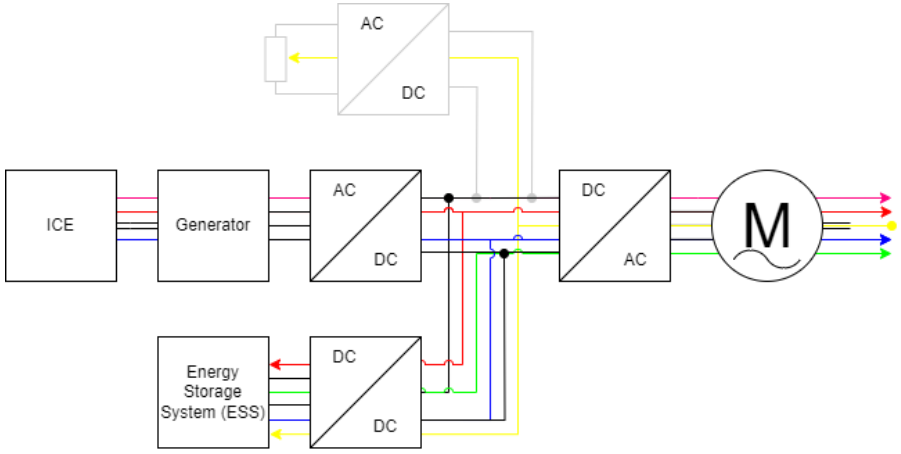


Figure 15: Series hybrid drive train with indicated driving modes; Electric-only, Battery recharge, Boost, Regenerative braking, ICE-only.

3.2.2 Parallel Hybrid Electric Vehicles

Parallel hybrid vehicle architectures work by mechanically coupling the Internal Combustion Engine (ICE) (ω_{ICE}, T_{ICE}) with the Electrical Machine (EM) (ω_{EM}, T_{EM}). These power flows are combined in a so called torque-coupler. Dependent on the drive mode, typical power ratio's $P_{ICE} : P_{EM}$ can be configured. A schematic of a parallel hybrid electric drive train with brake resistor is given in Figure 16.

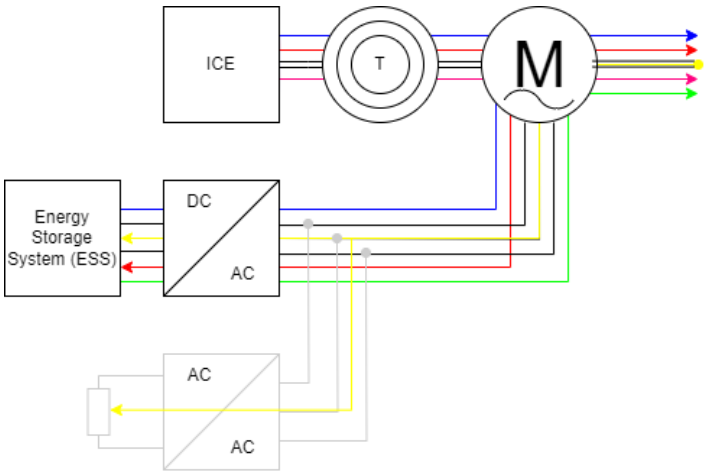


Figure 16: Parallel hybrid drive train with indicated driving modes; Electric-only, Battery recharge, Boost, Regenerative braking, ICE-only.

3.2.3 Series-Parallel Hybrid Electric Vehicles

Lastly, a more complex hybrid drive train configuration will be discussed, namely a series-parallel hybrid configuration. This configuration was first developed by Toyota, for it's Prius model, which turned out to be one of the most energy efficient and most-successful hybrids ever sold by Toyota [8]. A series-parallel drive train works by combining the strengths of both series and parallel configurations, while minimizing the weaknesses of each type of drive train. It works by integrating both torque- and rpm-couplers through a planetary gear set.

A schematic of a parallel hybrid electric drive train with brake resistor is given in Figure 17.

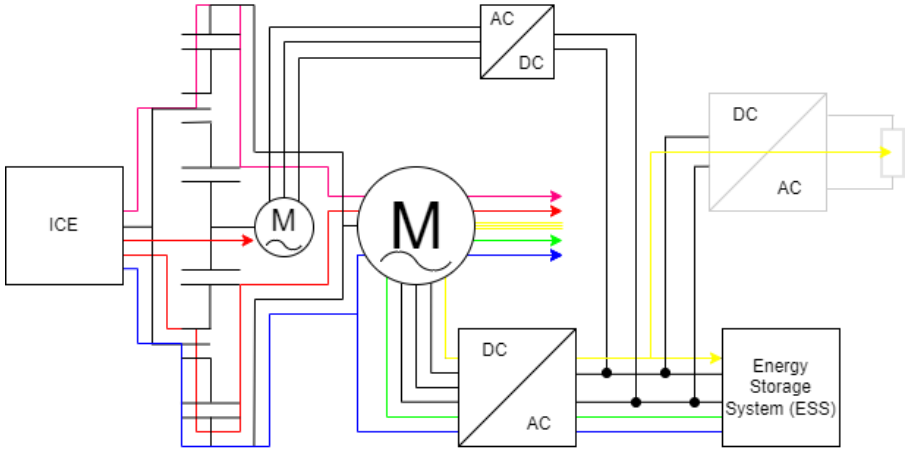


Figure 17: Series-Parallel hybrid drive train with indicated driving modes; Electric-only, Battery recharge, Boost, Regenerative braking, ICE-only.

3.2.4 Comparison Between Hybrid Drive Trains

Next a comparison between the various types of hybrid drive trains will be presented in Table 1. This shows the various pro's and con's of the different hybrid drive train configurations.

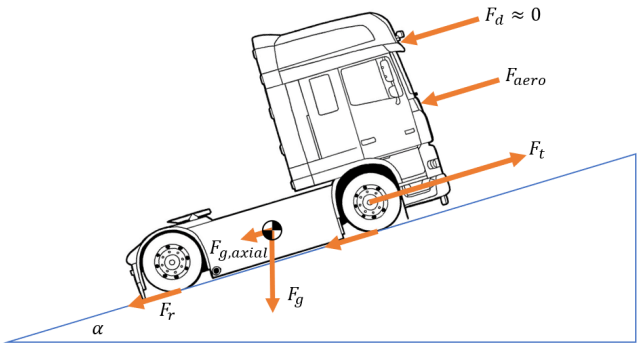
Table 1: Comparison between the various hybrid drive train configurations

Series Hybrid	Parallel Hybrid	Series-Parallel Hybrid
Wheels and ICE are not mechanically coupled. ICE always operates at optimal efficiency	Wheels and ICE are mechanically coupled.ICE not always operating on optimal point.	Wheels and ICE are mechanically coupled. ICE not alway operating on optimal point
Increased energy recuperation through regenerative braking as P_{EM} is larger.	Extra gearbox required for RPM or torque matching at ICE.	High efficiency, due to efficient power distribution
Lower efficiency, due to many power conversion steps.	A mechanical coupler is requried, which decreases efficiency.	Highly complex system

This table shows that no one-solution-fits-all can be found and that the selection of a hybrid drive train is highly dependent on the use case and the requirements specified by the customer. This way, by integrating optimization into the vehicle system architecture design phase, a proper case specific design can be found at relatively low cost. Note that a lot of variations are made on the above mentioned drive train types, ranging from more electric machines on various axles to the implementation of multi speed gearboxes. Furthermore, no explanation will be given on the topic of electric or fuel cell drive trains, as the mechanical system architecture is not very different from the above mentioned ones, but only the way of generating electrical power is different.

3.3 Vehicle Modelling

A starting point for the design of a vehicle system architecture is the energy and power budget of such vehicle for a given drive cycle. This energy and power budget is dependent on the type of vehicle and the forces acting on it. A longitudinal vehicle model describing the forces acting on the vehicle in time is given in Equation 3.1. This model is based on the free body diagram of Figure 18:



$$m_v \frac{d}{dt} v(t) = F_t(t) - [F_{aero}(t) + F_r(t) + F_g(t) + F_d(t)] \quad (3.1)$$

Figure 18: Free body diagram of a vehicle driving uphill.

Here, the traction force (F_t) should at least counteract the resistive forces such as rolling resistance, aerodynamic resistance, gravitation resistance due to inclination (α) and other disturbance forces, to achieve constant speed or acceleration. Some assumptions made for the estimation of the various resistive forces are:

$\mathbf{F}_{\text{aero}}$: For road vehicles, it is assumed that the compressibility effects are not present, due to the low velocity and high dimensions of the vehicle. This results in a constant value of drag coefficient.

$\mathbf{F}_{\text{aero}}$: It is assumed that the acceleration of the vehicle does not significantly affect the pitch angle of the vehicle, meaning that the drag coefficient is not dependent on pitch angle, hence is constant.

$\mathbf{F}_{\text{aero}}$: F_{aero} can be modeled using the following equation:

$$\mathbf{F}_{\text{aero}}(v) = \frac{1}{2} \cdot \rho_a \cdot A_f \cdot C_d \cdot v^2 \quad (3.2)$$

$\mathbf{F}_{\text{r}}$: The rolling resistance is a function of vehicle speed, tire pressure, and road conditions. Typically, the tire pressure has an inverse square-root contribution to the rolling resistance. It is assumed that vehicle velocity has no significant impact on the rolling resistance, due to the low operating velocity, hence $\mathbf{C}_{\text{r}}$ is assumed constant over the entire speed range for a given pressure and road condition.

$\mathbf{F}_{\text{r}}$: F_r can be modeled using the following equation:

$$\mathbf{F}_{\text{r}}(p, \text{road conditions}, \alpha) = C_r(p, \text{road conditions}) \cdot m_v \cdot g \cdot \cos(\alpha) \quad (3.3)$$

$\mathbf{F}_{\text{g,axial}}$: $F_{g,axial}$ is only dependent on the angle of inclination, not on vehicle velocity, and can be modeled using the following equation:

$$\mathbf{F}_{\text{g,axial}}(\alpha) = m_v \cdot g \cdot \sin(\alpha) \quad (3.4)$$

$\mathbf{F}_{\text{d}}$: Disturbance forces such as inertial forces due to the rotation of gearboxes and engines are not taken into account, therefore:

$$\mathbf{F}_{\text{d}}(v) = 0 \quad (3.5)$$

3.4 Internal Combustion Engine Modelling

The behavior in terms of performance and efficiency of Internal Combustion Engines (ICE's) can usually be quantified as a function of engine speed (n_e) and engine torque (T_e). However, problems arise whenever various size engines are compared which are similar in the way they operate, but highly different in the magnitude of torque or rpm. In order to accurately model Internal Combustion Engines (ICE's) for a large range of engine sizes, the normalized engine variables of c_m and p_{me} is proposed by Guzzella and Sciarretta [27]. This means that for engines of similar type the speed boundaries $c_{m_{lower}}/c_{m_{upper}}$ vary less than the torque limits, hence more accurate comparison can be done.

$$c_m = \frac{\omega_e \cdot S}{\pi} \tag{3.6}$$

$$p_{me} = \frac{N \cdot \pi \cdot T_e}{V_d} \tag{3.7}$$

From this, the mechanical power produced by the ICE for a given:

$$P_e = z \cdot \frac{\pi}{16} \cdot B^2 \cdot p_{me} \cdot c_m \tag{3.8}$$

3.5 Battery Modelling

Battery modelling is most commonly modelled using the method of equivalent circuit modelling (ECM), for which the cells are modelled as a theoretical electrical circuit given a set of characteristics such as voltage (V), current (I), capacitance (C), inductance (L) and resistances (R) in series or parallel. The type of cell and cell chemistry determines all parameters of the equivalent circuit used for analysing the cell. Using experimental data, model fitting can be applied to fit the cell electrical properties to an equivalent circuit model. An example of a Battery Equivalent Circuit is given in Figure 19:

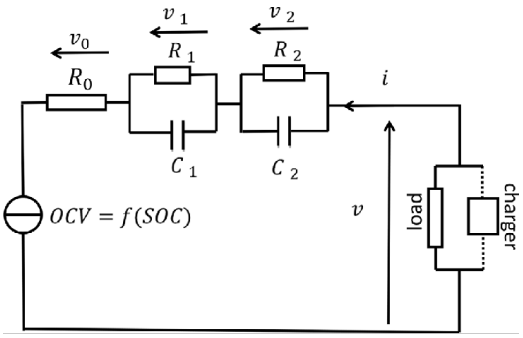


Figure 19: Equivalent Circuit Model of standard battery cells [77].

Various battery modelling methods are proposed in literature [76], ranging from basic sets of equations, to dynamic mid- or high-fidelity modelling codes. A common used Python module for battery pack modelling is the Python Battery Mathematical Modelling (PyBaMM [1]) package, developed by Sulzer et al. [63]. This module uses non-linear battery models to estimate cell output properties in time. For this, a battery load profile can be added to estimate it's performance properties. An extension on this is made in the Liionpacks [2] package by Tranter et al. [67]. Here, entire battery packs can be evaluated, ranging from thermal modelling to the addition of battery degradation using Solid Electrolyte Interface (SEI) quantification.

3.6 Electric Machine Modelling

In an electric vehicle, one of the most important components is the electric machine. The electric machine takes either electrical power as input, and converters this through a magnetic field, into rotational motion of the outgoing

[1] https://docs.pybamm.org/en/latest/, accessed on 24/10/2023
[2] https://liionpack.readthedocs.io/en/latest/, accessed on 24/10/2023

shaft, or works as a generator and converts rotational motion to electrical power. As there have been designed various types of electric machines, which operate using different working principles, a clear distinction has to be made between Direct Current (DC) Motors and Alternating Current (AC) Motors.

3.6.1 DC Motors

DC motors consist of an armature, which is a coil of wire that can rotate within the magnetic field (rotor) and a electromagnetic stator. By applying a direct current to the armature, a magnetic field is created. This interacts with the magnetic field of the stator, resulting in a force, according to Maxwell's equations. This force is used to rotate the armature. By using a commutator (connected to the armature), which is basically a physical rotary switch, the current direction is reversed at the right moments. This change in current direction, results in changing magnetic fields, which results in a constant push of the armature in the proper direction. A simplified equivalent circuit of a separately excited DC machine, together with a figure representing the working principles behind DC machines is given in Figure 20 and Figure 21.

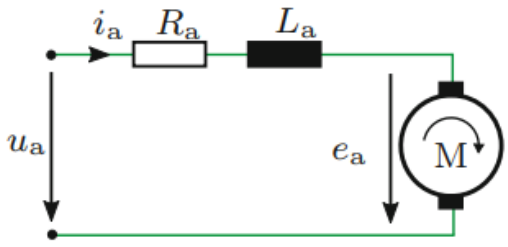


Figure 20: Equivalent Circuit of a DC motor according to De Doncker [15]

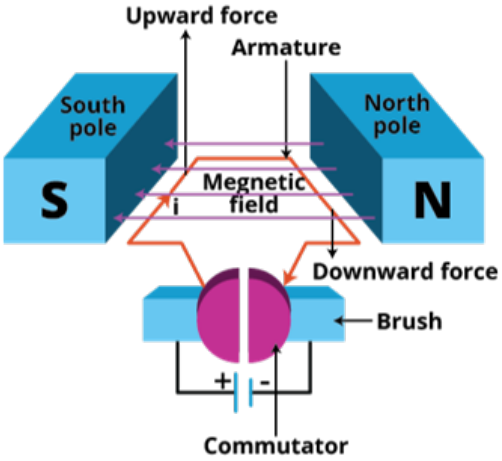


Figure 21: Simple representation of the working principle of a DC motor through the generation of an armature magnetic field and a constant stator field.

3.6.2 AC Motors

A more widely used type of electric machines for the mechanical power generation of e-mobility solutions are AC motors. This is due to the fact that DC motors are not very power dense, require commutator brushes which are not very heavy duty (reliability becomes an issue) and have very little starting torque. Although many variations on AC motors have been designed, the most widely used type of AC motors are asynchronous and synchronous AC machines. Their working principles are very different. Where synchronous AC machines work by matching (synching) the rotational speed of the stator's magnetic field and that of the magnetic field of the rotor, asynchronous AC machines have a certain *lag* between the rotational speed of the stator's magnetic field and the rotor. AC asynchronous machines (also referred to as induction machines), work using the principle of induction, where only the stator is excited, and an induced magnetic field is generated in the rotor in accordance to Maxwell's equation on varying magnetic field strength. The differences between synchronous machines and asynchronous or induction machines are given Table 2. Figure 24 furthermore shows the difference in equivalent circuit model between an asynchronous and a synchronous AC machine.

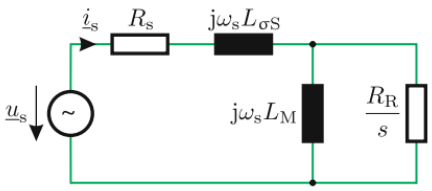


Figure 22: Equivalent Circuit of an induction motor according to De Doncker [15]

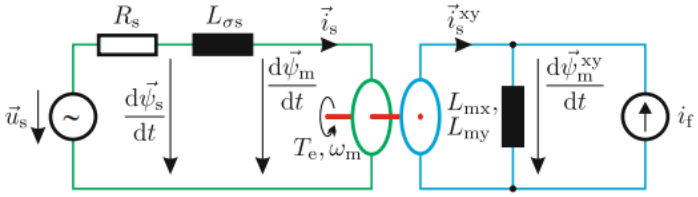


Figure 23: Equivalent Circuit of a permanent magnet synchronous motor with salient rotor according to De Doncker [15]

Figure 24: Equivalent circuit comparison between the two distinct AC machines.

Table 2: Comparison between synchronous and asynchronous (induction) machines

Synchronous Machine	Asynchronous Machine
The rotor of a synchronous machine rotates at a constant speed that is synchronized with the AC supply frequency	The rotor of an induction machine rotates at a speed slightly less of the AC supply frequency due to a certain "slip" ratio (s)
Synchronous machines are often used in applications which require high level of precise control.	Asynchronous mahcines are the working horse of the industry due to their robustness and low-maintenance.
Synchronous machines are more energy efficient	Asynchronous machines are less energy efficient.
Synchronous machines are able to run at constant speed for long durations of time.	Asynchronous machines are more suitable for variable speed application where precise control is not required.
Synchronous machines are more environmental demanding due to the use of permanent magnets (rare earth materials)	Asynchronous machines require very little / no rare earth materials thereby making them cheap to manufacture.

In order to properly model the performance (T_{out}, ω_{out} & P_{out}) output of a given electric machines given a set of input voltage / current and motor parameters (number of coils, gap length, materials used, dimensions), various techniques are used, ranging from simplified empirical data, to equivalent circuit methods or more advanced multi-physics toolboxes such as Pyleecan [3]. The pyleecan model is interesting as it allows for Object-Oriented Programming with a computationally efficient basis written in C. It models the electrical machines as equivalent circuits using the methods as described by De Doncker [15]. Pyleecan discretizes a 2D cross-section of an electrical machine to come up a time-dependent solution of the performance output of such a machine.

3.7 Fuel Cell Modelling

One of the main power generation components that is being investigated heavily within VDL's R&D department is a hydrogen fuel cell. A hydrogen fuel cell is a electrochemical power conversion device which allows hydrogen to react with oxygen in the fuel cell stack which results in the flow of electrons from anode to cathode. This way, voltage is generated which in turn can provide power to for instance an electrical motor or similar. The electrochemical reaction which occurs on a fuel cell's anode and cathode is given below:

[3] https://www.pyleecan.org/, accessed on 13/11/2023

$$\text{Anode:} \quad 2H_2 + 4OH^- \longrightarrow 4H_2O + 4e^-$$
$$\text{Cathode:} \quad O_2 + 4H^+ + 4e^- \longrightarrow 2H_2O$$

Various methods to model the performance of such fuel cells have been presented in literature [59] [68]. Especially, the paper presented by Tremblay et al. provides the reader with a clear methodology to determine the fuel cell's static and dynamic (transient) behavior with a relatively small error ($\pm 1\%$) for a given specification sheet provided by the fuel cell manufacturer. The method relies on a simplified model of a fuel cell stack which models the fuel cell stack as a controllable voltage source in series with a constant resistance. This way, the controlled voltage can be computed using Equation 3.9 for static conditions. A more detailed model is then presented which shows the fuel cell performance as a function of varying inflow conditions of fuel and air (time dependent) (Figure 25). Please note, that only Low Temperature Proton Exchange Membrane (LT-PEM) fuel cells are considered, as these are the only commercially available solutions at the moment and prove to be most energy- and power efficient.

$$E = E_{\text{opencircuit}} - NA \ln\left(\frac{i_{\text{fuelcell}}}{i_0}\right) \cdot \frac{1}{sT_d/3+1}$$
$$V_{\text{fuelcell}} = E - R_{\text{ohm}} \cdot i_{\text{fuelcell}} \tag{3.9}$$

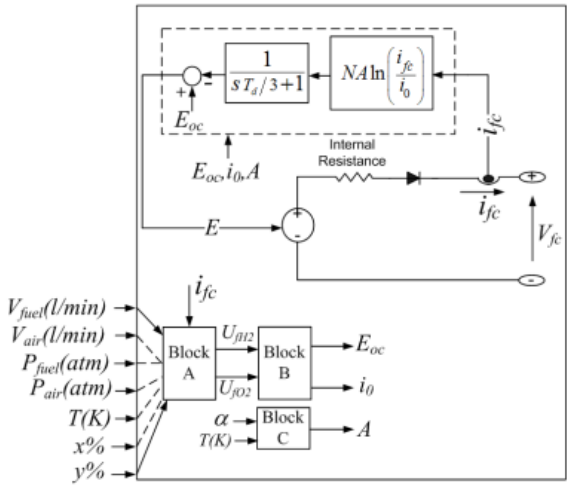


Figure 25: Equivalent Circuit of a fuel cell according to Tremblay et al. [68]

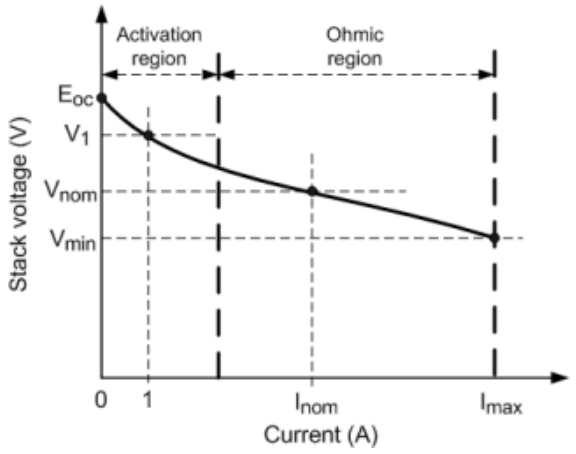


Figure 26: The required polarization curve of a fuel cell needed for the generic model parameter determination [68]

3.8 Transmission Modelling

The main goal of a vehicle transmission is to provide the best performance (torque) range for a given range of rotational speeds. As the torque/speed curve of internal combustion engines is very different from that of an electric machine, electric drive trains requires far less gears to achieve similar performance. This difference in torque-speed curve is given in Figure 27

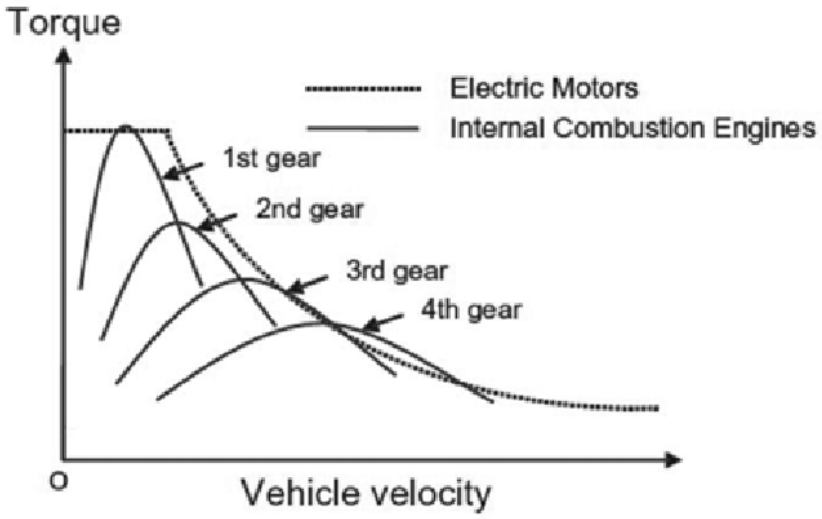


Figure 27: Torque-speed curve comparison between a single-speed EV and an ICE combined with multi-speed gearbox

Gearboxes can be defined as a transformer block, which takes as input two independent variables, namely rotational speed, typically in RPM, and torque, i.e. force at a distance r from the origin, and exchange the two, to obtain two different torque and rpm levels given a gear ratio. By taking into account the efficiency of such gearbox, the following two equations can be found:

$$\omega_1 = \gamma_i \cdot \omega_2 \cdot \eta_{gear,i} \tag{3.10}$$

$$T_2 = \gamma_i \cdot T_1 \cdot \eta_{gear,i} \tag{3.11}$$

There are various types and configurations of gearboxes present in industry. However, the most commonly found gearboxes are either manual gearboxes, where the vector of all gear ratio's is real and finite (typically order of 6 or 7) $\gamma = [\gamma_1, ..., \gamma_n]$, automatic transmissions, or continuous variable transmissions, which realize an unlimited amount of gear ratios, hence $\gamma = \infty$.

Other components which are commonly found in gearboxes are clutches and torque converters. The components decouple the prime mover from the vehicle propulsion system [27], thereby allowing the vehicle to be more energy efficient and increase its performance.

3.9 Key Takeaways

1. As we learn more about the negative effects caused by the combustion of hydrocarbon fuels, more research is done on investigating alternative ways of transportation, with limited environmental effects. E-mobility solutions are increasingly gaining significance in this context, due to their minimal emission of green house gasses. VDL Special Vehicles has positioned itself at the forefront of this movement by dedicating efforts to research, development, and construction of electric mobility solutions tailored to both on- and off-road vehicles.

2. As the architectural design space for E-mobility vehicles is highly dimensional, with both integer and continuous design variables, it proves difficult to converge to a single "best-solution-fits-all" system architect. It furthermore is found that the most-suitable, or optimal, system architecture is highly dependent on the use case, and hence the constraints to which the design has to adhere.

3. In order to properly evaluate the various system architectures and the synergy between various multi-domain components, various modelling tools have to be found, ranging from low-fidelity analytical tools to mid-fidelity equivalent circuit models. These have been presented above.

4. The starting point of a system architecture evaluation method is the vehicle model, which takes into account the driving characteristics and forces applied to the vehicle to compute the required energy- and power requirements for the electric drive-train system architecture.

5. Various electrical power generation components and their modelling tools have been presented ranging from batteries, fuel cells and generators. Only these components will be required to be modelled as this is the main focus point of VDL, as well as the fact that these solutions are found to be most promising and commercially available.

4 Reliability Based Robust Design Optimization

This section will present the reader with all relevant state-of-the-art knowledge on applying and integrating Reliability Based Robust Design Optimization into an MDAO problem formulation for the design of vehicle system architectures. First, a basic introduction on the various subjects of MDAO are explained, after which the reader will go into more depth on the methods found in literature to properly implement and take into account uncertainty in such MDAO problem formulation. Also a basic introduction on the numerous gradient-based and gradient-free optimization algorithms will be given. Lastly, more information will be provided on the topic of mixed-integer optimization and decision hierarchy which are relevant topics in vehicle system architecture optimization.

4.1 A Small Introduction to MDAO

Multi-disciplinary Design Optimization (MDO) is a powerful methodology which is used to increase system performance while at the same time reducing design time, costs, and uncertainties, using the power of fast computing. Conventional, or classic system engineering approaches to designing components of a system, for instance, chassis, electric machines, gearboxes, relied on sequential non-coupled methods. Here an individual component was optimized and later checked for system performance within the overall system architecture. This can of course yield benefits, however widening the optimization scope will ultimately always increase system performance, on either component or system level. [44]. This effect becomes more evident when we acknowledge the fact that various disciplines, such as structures and aerodynamics, are dependent on each other. This means that in most engineering cases, optimizing for one discipline alone might lead to sub optimal design choices for other disciplines.

With the increased computational capabilities that come with these new high performance computers, it is possible to find near optimal design solution through numerical simulations. In the case that the problem formulation is set up correctly, this does not require any human or engineering interference, which will save significant amount of time and cost. Therefore, these MDO problem formulations and solution frameworks can be described as:

Definition 1 MDO *A numerical simulation framework, including coupled disciplines, which is used to optimize a system considering all disciplines present within a system and its design variables simultaneously, with the goal of minimizing an objective and satisfying constraints.*

From literature it is found that Multi Disciplinary Analysis and Optimization (MDAO) is already regularly applied to architectural design space exploration [3] [9] [42] [48]. Since the architectural design space optimization problem is a highly multi-variate, hierarchical, non-convex, mixed-integer, constraint, multi-objective design optimization problem [9], it can become challenging to find optimal design solutions. A lot of research has gone into choosing the appropriate gradient-based, or gradient-free optimization strategies to find global optima, as well as choosing the appropriate MDO framework to solve the optimization problem [44] [48] [52]. It is found that each optimization problem focused on architectural design space exploration and optimization is very problem specific, where no one-solution-fits-all approach can be applied. The best choice for your optimization framework depends on factors such as the complexity of your problem, computational resources, the level of coupling between disciplines, and the importance of maintaining inter-disciplinary feasibility. Therefore, the goal of this section is to go over the various optimization frameworks and elaborate on its strengths, weaknesses and use cases. A MDAO framework can be divided up into: MDA Convergence Solvers, Disciplinary Coupling of Variables and (Multi)-Objective Optimization Algorithms.

4.2 MDA Convergence Scheme's

In order to obtain a consistent set of design variables for all disciplines, an Multi-Disciplinary Analysis block requires a convergence scheme, whenever coupling is present. The mostly used solvers for this are: *Nonlinear Block Jacobi*, *Nonlinear Block Gauss-Seidel* and *Newton's Method*. Their strengths and weaknesses are discussed below:

4.2.1 Nonlinear Block Jacobi

A nonlinear block Jacobi convergence scheme in MDA is a straightforward method where first an initial guess of the coupling variables is required. These coupling variables are defined as follows:

Definition 2 Coupling Variables *are variables from which multiple disciplinary tools are dependent of in order to be solved.*

The MDA block converges whenever the output of the disciplines based on the provided initial guess of coupling variables, converges to a given solution. The main advantage of a Nonlinear Jacobi Convergence Scheme is the fact that multiple solvers can run in parallel, hence, no feed-forward / feed-back of information is required. This decreases the computational time to convergence, however does result in more disciplinary infeasible solutions.

4.2.2 Nonlinear Block Gauss-Seidel

A nonlinear block Gauss-Seidel convergence scheme in MDA requires, in contrast to the Nonlinear Block Jacobi convergence scheme, the flow of information between various disciplines. The Gauss-Seidel convergence scheme makes use of direct coupling between the disciplines by allowing the outcome of disciplines to be directly passed to the next discipline, within the same iteration. One of the main advantages of this, is that convergence can be achieved much faster, however this is highly dependent on the correct sequencing of disciplines. By wrongly ordering the disciplines, convergence can take a very long time or not happen at all.

4.2.3 Newton's Method

Newton's method, also referred to as Newton-Raphson method, is an iterative numerical technique used to approximate solutions of a real-valued function. It specifically searches for the root, i.e. $f(x) = 0$. By calculating the tangent line to a curve, for a given starting point, the iterative solver is able to find the root, within a relative low number of iterations, by constantly evaluating the tangent at a given point. The biggest reason why Newton-Raphson is used extensively in engineering, is that the solution scheme converges quadratically near the root. A drawback lies in the fact that whenever an initial guess is too far from the solution, the solver might not converge at all. Furthermore, the Newton-Raphson solver has trouble finding global roots, whenever a function is multi-modal. In MDO applications this is usually solved by using an existing design (and hence its design variables) as the initial starting position, which is assumed to be close to optimal.

4.2.4 Comparison of coupled system solvers

In Figure 28, we illustrate the convergence behavior of the methods previously mentioned. Among the mentioned methods, the Jacobi method exhibits the least favorable performance, characterized by significant oscillations. In contrast, the Newton method demonstrates the most favorable behavior due to its quadratic convergence near the root. However, it is important to note that the Newton solver's convergence is not guaranteed, and this potential lack of convergence could propose a significant challenge during the optimization.

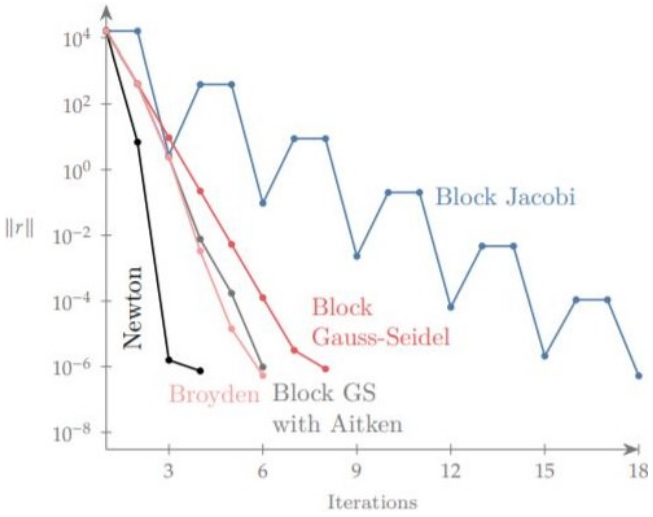


Figure 28: Comparison of different MDA solvers[45]

4.3 Monolithic MDO Architectures

Most engineering problems are formulated as a monolotic MDO problem, where only one objective function is to be minimized. Most MDO architectures are specifically designed to perform well on monolitic problem formulations, to ensure fast, and at least locally optimal, feasible, design solutions. Below, the various monolitic MDO architectures which are most found in literature are explained.

4.3.1 IDF

An IDF architecture makes use of the parallel computation of the various disciplines within a MDO problem to find an Individual Feasible Solution. This MDO architecture therefore computes various design solutions, which might not be feasbile throughout the various disciplines. It does this by adding independent copies of the coupling variables, to allow independent solvers to run in parallel [45]. These coupling variables are called target variables in IDF architectures, and are independent from the real coupling variables. To enable the optimizer to come up with a feasible interdisciplinary design, extra consistency or equality constraints are added which make sure the target variables are equal to the real coupling variables.

This can be formulated as follows:

$$\begin{aligned} \min \quad & f(x;\hat{u}^*) \\ \text{by varying} \quad & x, \hat{u}^t \\ \text{s.t.} \quad & g(x,\hat{u}) \leq 0 \\ \text{s.t.} \quad & h_i^c = \hat{u}_i^t - \hat{u}_i = 0 \qquad i = 1,...,m \\ \text{while solving} \quad & r(\hat{u}_i; x, \hat{u}_{j\neq i}^t) = 0 \qquad i = 1,...,m \\ \text{for} \quad & \hat{u} \end{aligned} \tag{4.1}$$

Unlike MDF, IDF does not guarantee multidisciplinary feasible at every step of the optimization iteration. Hence, this architecture is most likely to only produce a feasible design after the optimizers is stopped. Furthermore, the number of design variables is increased by the number of coupling variables (target variables). In an optimization problem where the design vectors is already large, this might result in a problem formulation which is unable to be solved efficiently. Lastly, due to the fact that the various components, or disciplines, are evaluated individually in parallel, gradient-based optimizers are more robust and result in better convergence rates [45].

4.3.2 MDF

An MDF architecture is based on solving a coupled system of disciplines, for which an interdisciplinary feasible design is found through every iteration. The objective function and constraint function can from that be evaluated, such that optimization can occur through an optimization algorithm. An MDF approach is also called a reduced-space approach, since the optimizer has no effect during the optimization on the state and coupling variables [45].

This can be formulated as follows:

$$\begin{aligned} \min \quad & f(x;\hat{u}^*) \\ \text{by varying} \quad & x \\ \text{s.t.} \quad & g(x,\hat{u}^*) \leq 0 \\ \text{s.t.} \quad & h(x,\hat{u}^*) = 0 \\ \text{while solving} \quad & \hat{r}(\hat{u}^*; x) = 0 \\ \text{for} \quad & \hat{u} \end{aligned} \tag{4.2}$$

Here, $\hat{u}^*$ refers to the multidisciplinary feasible point found trough the MDA solver, and $\hat{u}$ refers to the set of coupling variables found through MDA. The residual function $\hat{r}$ hence needs to solve for a root in MDA convergence scheme, based on a given set of coupling- and design variables.

An advantage of implementing an MDF formulation is that for each iteration of the optimization algorithm, based on consistent variable values across the disciplines, all disciplines will produce a feasible design [52]. This can be beneficial in case that time resources are limited, and the need for an optimal design is not desired. It should

be noted, that constraints at early termination of the optimization might not be satisfied. Lastly, this shows that MDF requires an convergence scheme (MDA converger) evaluated at every optimization iteration. This can be cumbersome and time consuming.

4.4 Distributed MDO Architectures

In contrast to monolitic MDO architectures, as discussed in subsection 4.3, which are based on a single objective function which is to be minimized, distributed MDO architectures split up the optimization problem into various, smaller optimization problems which are to be solved. This means that it can be applied to various multi-objective, hierarchical optimization problems.

4.4.1 Collaborative Optimization

One type of distributed MDO architectures is Collaborative Optimization (CO) Architecture. This is a type of distributed IDF framework, where all optimization problems are independent from each other by use of target variables. In order to specify the minimization of residuals between target variables and real coupling variables, constraints are used. Here this minimization of the residuals in every sub optimization problem can be defined as the objective function of that sub problem:

$$\begin{aligned} \text{min} \quad & J_i(x_{oi}^t, x_i; \hat{u}_i) \\ \text{by varying} \quad & x_{0i}^t, x_i \\ \text{s.t.} \quad & g_i(x_{oi}^t, x_i; \hat{u}_) \leq 0 \\ \text{while solving} \quad & \hat{r}_i(\hat{u}_i; x_{oi}^t, x_i, \hat{u}_{j\neq i}^t) = 0 \\ \text{for} \quad & \hat{u}_i \end{aligned} \tag{4.3}$$

Now the overall problem formulation can then be specified as follows, where the sub problem objective function is added as a consistency constraint.

$$\begin{aligned} \text{min} \quad & f\left(x_0, x_1^t, \ldots, x_m^t, \hat{u}^t\right) \\ \text{by varying} \quad & x_0, x_1^t, \ldots, x_m^t, \hat{u}^t \\ \text{s.t.} \quad & g_0\left(x_0, x_1^t, \ldots, x_m^t, \hat{u}^t\right) \leq 0 \\ & J_i^* = \left\|x_{0i}^t - x_0\right\|_2^2 + \left\|x_i^t - x_i\right\|^2 \\ & + \left\|\hat{u}_i^t - \hat{u}_i\left(x_{0i}^t, x_i, \hat{u}_{j\neq i}^t\right)\right\|^2 = 0 \quad \text{for } i = 1, \ldots, m, \end{aligned} \tag{4.4}$$

4.4.2 Analytic Target Cascading

Another method of solving multi-level optimization problems for sizing of architectural design choices and finding optimal solutions is presented by Beernaert et al. [5]. Here a system architecture and its component are modelled in a way such that it can directly be converted into a Analytic Target Cascading MDO formulation. This way of applying ATC in multi-level MDO problems was first developed by [35], and is based on minimizing first all sub-problem MDO formulations, as given in Equation 4.5, to eventually minimize the final objective function. The overall objective function is a summation of the overall objective function in combination with a penalty function, which is dependent on the objective functions of the various sub-problems. ATC is a type of IDF distributed MDO architectures.

$$\begin{aligned} \min_{\bar{\mathbf{x}}_{ij}} \quad & f_{ij}(\bar{\mathbf{x}}_{ij}) + \phi(c_{ij}) \\ \text{s.t.} \quad & \mathbf{g}_{ij}(\bar{\mathbf{x}}_{ij}) \leq 0, \quad \mathbf{h}_{ij}(\bar{\mathbf{x}}_{ij}) = 0 \\ \text{where} \quad & \bar{\mathbf{x}}_{ij} = \left[\mathbf{x}_{lij}^T, \mathbf{t}_{(i+1)k_1}^T, \cdots, \mathbf{t}_{(i+1)k_{nc_{ij}}}^T, \mathbf{r}_{ij}^T\right]^T \\ \text{and} \quad & \bar{\mathbf{c}}_{ij} = \left[(\mathbf{t}_{ij} - \mathbf{r}_{ij})^T, (\mathbf{t}_{(i+1)k_1} - \mathbf{r}_{(i+1)k_1})^T, \cdots, (\mathbf{t}_{(i+1)k_{nc_{ij}}} - \mathbf{r}_{(i+1)k_{nc_{ij}}})^T\right]^T \end{aligned} \tag{4.5}$$

Here, constraints are assigned to all single sub-problems, while variables are shared among sub-problems. This way, each sub-problem has a local objective function f_{ij} and local (in)equality constraints $\mathbf{g}_{ij}, \mathbf{h}_{ij}$. This way each parent sub problem sends target values to their children sub-problem, which in return send response values back in an effort to reach their acquired targets [5]. The goal is for the sub optimization problem to reach an optimum close to the target values as specified by the parents sub-problems. Finally, the sum of all local objective functions build up the overall optimization problem. A visualization is given in Figure 29:

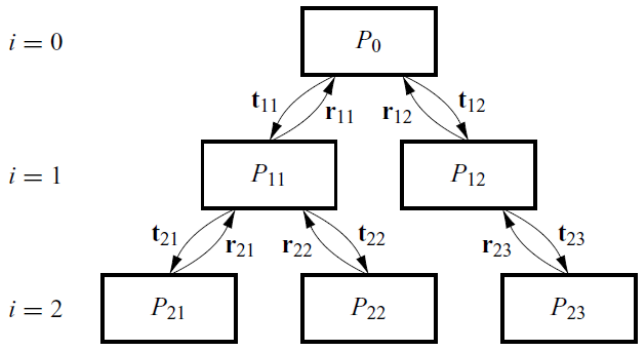


Figure 29: A multi-level optimization, as used in ATC optimization problem. Here $\mathbf{r}_{ij}$ and $\mathbf{t}_{ij}$ are the response and target values as send from parent to children sub-problem. Figure reproduced from [5].

4.5 Optimization Algorithms

The last distinctive part of obtaining a complete and correct MDAO problem formulation is the selection of the type of optimization algorithm. The type of problem, objective function and design space characteristics determine what optimization algorithm is best suited, to solve the optimization problem. There is a clear distinction between gradient-based and gradient-free optimization algorithms which is explained below.

4.5.1 Gradient-based

Gradient-based optimization algorithms are most used in engineering purposes, as it achieves fast convergence to an optimum in a convex, continuous design space. Gradient-based algorithms use, as the name suggests, gradients to solve fast for a local optima. to make sure that the optimizer finds a real (local) optima, the solution should satisfy some optimality conditions. These are in gradient-based optimization referred to as the Karush-Kuhn-Tucker(KKT) optimality conditions for constraint optimization, as presented below:

First-Order Optimality Conditions:

- $\nabla f + J_h^T \lambda + J_g^T \sigma$
- $h(x) = 0$
- $g + s \odot s = 0$
- $\sigma \odot s = 0$
- $\sigma \geq 0$

Second-Order Optimality Conditions

- $p^T H_{\mathcal{L}} p > 0$ for all p such that:
- $J_h p = 0$
- $J_g p \leq 0$ for the active constraints

For every objective and constraint, the Lagrangian multipliers are computed to solve for the fastest "route" to an optimal solution. Hence, gradient-based algorithms require the function to be continuous in value (C^0), gradient (C^1) and Hessian (C^2) in at least a small neighborhood of the optimum [45].

$$J_h = \frac{\partial h}{\partial x} = \underbrace{\begin{bmatrix} \frac{\partial h_1}{\partial x_1} & \cdots & \frac{\partial h_1}{\partial x_{n_x}} \\ \vdots & \ddots & \vdots \\ \frac{\partial h_{n_h}}{\partial x_1} & \cdots & \frac{\partial h_{n_h}}{\partial x_{n_x}} \end{bmatrix}}_{(n_h \times n_x)} \tag{4.6}$$

$$J_g = \frac{\partial g}{\partial x} = \underbrace{\begin{bmatrix} \frac{\partial g_1}{\partial x_1} & \cdots & \frac{\partial g_1}{\partial x_{n_x}} \\ \vdots & \ddots & \vdots \\ \frac{\partial g_{n_h}}{\partial x_1} & \cdots & \frac{\partial g_{n_g}}{\partial x_{n_x}} \end{bmatrix}}_{(n_g \times n_x)} \tag{4.7}$$

$$J_f = \frac{\partial f}{\partial x} = \underbrace{\begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_{n_x}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_{n_f}}{\partial x_1} & \cdots & \frac{\partial f_{n_f}}{\partial x_{n_x}} \end{bmatrix}}_{(n_f \times n_x)} \tag{4.8}$$

Here, the derivatives can be computed analytically or using finite difference methods.

Sequential Linear Programming Algorithm The most effective and widely-used gradient-based solver is the Sequential Linear Programming algorithm, as first proposed by Wilson [73]. Sequential Quadratic Programming is a conceptual method in programming where the algorithm tries to numerically solve the KKT-conditions for which sequential (every iteration) quadratic problems are solved, assuming quadratic objective functions and constraints. As quadratic programming problems are relatively easy to solve, computational wise, a sequential quadratic programming algorithm proves most efficient when dealing with optimization problems for which gradients can be computed. From the KKT-conditions, it is also required to compute the Hessian. In cases where this is not possible, quasi-Newton methods can be used to approximate the Hessian. For a SQP algorithm with only equality constraints, the residual of the KKT-conditions is given as Equation 4.9, which is to be solved using convergence methods for λ and x as explained in subsection 4.2:

$$r = \begin{bmatrix} \nabla_x \mathcal{L}(x, \lambda) \\ \nabla_\lambda \mathcal{L}(x, \lambda) \end{bmatrix} = \begin{bmatrix} \nabla f(x) + J_h^\top \lambda \\ h(x) \end{bmatrix} = 0 \tag{4.9}$$

4.5.2 Gradient-free

While gradient-based algorithms can be highly efficient in finding (local)optimal design solutions, in a short time, it does require the computation of gradients, which has its limitations in some cases. It furthermore requires the function to be continuous, which sometimes is not the case or is unknown. In those situations, the use of gradient-free optimization algorithms proves to be the solution. A list of situations, where gradient-free algorithms are most useful, is provided below:

- Cases where the is a lack of gradient information (i.e. when dealing with black boxes)
- Cases where the design space is multimodal.
- Cases where the problem formulation is a multi-objective problem formulation
- Cases where certainty of a global minima is required, instead of local minima.
- Cases with discrete design variables (derivatives w.r.t. discrete variables are invalid).

Genetic Algorithms Genetic Algorithms are one of the most widely known evolutionary algorithms used to find optimal design solutions without relying on the computation of gradients. The optimizer starts by computing sets of design variables which minimize/maximize the design objective. By updating the set of design variables at every iteration, through selection, crossover and mutation, the optimizers selects the set of most promising design variables which have the highest change of minimizing/maximizing the objective function. Genetic algorithms have various ways of representing design variables, ranging from real-encoded to binary-encoded variables. A widely used type of Binary-Encoded Genetic Algorithm (BEGA) in optimization is the NSGA-II algorithm [17]. An overview view of how a simplified genetic algorithm works is presented in Algorithm 1.

A downside of the BEGAs is that the precision of the final design variable vector is dependent on the required number of bits and the bounds of the design variable x_i. This problem is not present in Real-Encoded Genetic Algorithms (REGA). Another problem with BEGA's is referred to in literature as the "Hamming Cliff", which is a result of the large change required in bits to move to adjacent real numbers [45]. Genetic algorithms can struggle

with this, as crossover and mutation have less effect on this outcome (e.g. 0111 to 1000). A benefit of BEGAs over REGAs is the fact that BEGAs are able to handle integer or discrete variables, whereas for REGAs to use discrete variables, rounding functions are required which are inefficient.

Algorithm 1 Genetic algorithm

[H]

$k = 0$

$P_k = \{x_1, x_2, ..., x_{n_p}\}$

while $k < k_{max}$ **do**

1) Compute f(x) for all $x \in P_k$ *Evaluate objective function*

2) Select $n_p/2$ parent pairs from P_k for crossover *selection*

3) Generate a new population of n_p offspring (P_{k+1} *crossover*

4) Randomly mutate some points in the population *mutation*

5) $k = k + 1$

Particle Swarm Algorithms Particle Swarm Algorithms (PSAs) are similar to GAs in a way that they both consist of a population of design vectors which are used to determine the local or global optima for a given set of constraints or bounds. Here the overall behavior of all particles (individual design vector) is determined by the local behavior of an individual particle and its interaction with the environment. Combining this information from all particles in a swarm, decisions can be made on the most optimal direction to converge to a local or global optima [19]. This shows that by properly connecting a set of highly simplistic particles into a swarm, and combining their knowledge about the design space based on their current and previous design point, can result in finding a local or global optima.

A PSA determines it's next design point based it's previous design point and it's velocity, which can be computed as follows:

$$x_{k+1}^{(i)} = x_k^{(i)} + v_{k+1}^{(i)} \Delta t \tag{4.10}$$

Where the velocity of the particle is updated as follows:

$$v_{k+1}^{(i)} = \alpha v_k^{(i)} + \beta \frac{x_{best}^{(i)} - x_l^{(i)}}{\Delta t} + \gamma \frac{x_{best} - x_k^{(i)}}{\Delta t} \tag{4.11}$$

This shows that the particle position is determined by either the particles inertia, which is a measure to how similar the velocity component is to that of previous iteration, the particles memory, which is a direction showing the location of the best position particle in all iterations for that specific particle, and the social influence, which is determined by the distance to the global (population) best solution. A downside of PSAs over standard GAs is the fact that PSAs tend to convergence faster to a local optima (instead of global), by taking into account particle location history. It furthermore requires manual tuning of the PSAs paramaters as explained above. A clear visualization of the updating step in design point for PSAs is visualized in Figure 30.

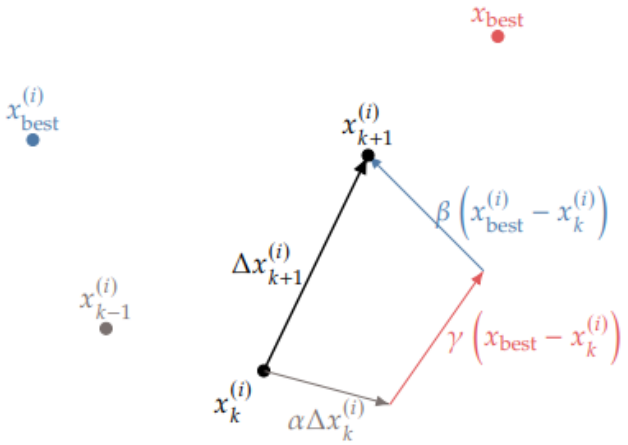


Figure 30: The procedure of updating the design point based on the particles inertia (α), memory (β) and social influence (γ). Figure reproduced from Martins [45].

4.5.3 Mixed-Integer Optimization

Since architectural design space exploration and optimization is a mixed-integer design space, an algorithm which is able to cope with this mixed-integer design space is required. A mixed-integer design space or optimization problem can be explained as:

Definition 3 Mixed-Integer Optimization Problem *is a type of optimization problem formulation where the design variables vector ($\mathbf{x}$) is a set of both continuous, as well as discrete variables. Hence $\mathbf{x} = [\mathbf{x}_C, \mathbf{x}_I]$ where $\mathbf{x}_C \in \mathbb{R}^C$ and $\mathbf{x}_I \in \mathbb{Z}^+$. Great care should be taken when choosing the appropriate algorithms that cope well with handling these different types of variables.*

There are multiple methods implemented in literature which are able to cope with this type of mixed-integer optimization. The discrete design variables can either be binary (0 or 1), integer or discrete variables. Here the integer variables are for instance the number of electric machines or the number of battery modules in series and parallel. Discrete variables refer for instance to the type of battery cell chemistry that is applied, i.e. LFP or NiCd, and usually represent a string in programming languages. In literature, discrete variables such as the type of battery cell chemistry as mentioned above, are usually represented as an integer variable, where there is a mapping from type to integer value. The simplest form of dealing with mixed-integer design spaces is through exhaustive search. This however can become computationally heavy if the number of discrete design variables is moderate to large, as now the number of design options scales (n-1)!.

Other approaches, which are designed to handle a mixed-integer design space well uses the method of rounding, where a continuous design variable is rounded to the nearest integer for evaluation. This however, also is computationally heavy and does not always result in a feasible or optimal result. Lastly, a method which was developed specially for integer programming problems, is branch and bound. Branch and bound works on linear convex optimization problems, where it assumes linearity in constraints and objectives. It is found to be highly robust and well suited for integer programming problems.

However, most complex multidisciplinary optimization problems are non-linear and might not be convex. Therefore, more advanced Mixed-Integer Non-Linear Programming (MINLP) algorithms were developed by Roy et al. [56] [57]. These methods work by splitting the design variable vector up into a set of continuous and discrete design variables, each handled differently, by different algorithms. This way, the two distinct design spaces are handled most efficiently by algorithms specifically suited for those types of design spaces. The algorithm first proposed by Roy et al. [56], named AMIEGO (A Mixed-Integer Efficient Global Optimization), works by combining branch and bound, efficient global optimization, surrogate model optimization and gradient-based algorithms to solve such MINLP problems. By performing expensive optimization on the set of continuous design variables for a given set of discrete or integer design problems ($\mathbf{x}_{integer} = constant$), a kriging or surrogate model can be created on the discrete design variable space to eventually obtain global optima in a mixed-integer design space. A lot

of these specific algorithms are based on the EGO (Efficient Global Optimization) framework [31]. The biggest strengths of applying such a MINLP algorithms such as AMIEGO is it's ability to explore the entire integer design space cost-effectively combined with the efficiency of gradient-based optimization algorithms. Furthermore it is implemented into the OpenMDAO framework [23], which allows for even faster convergence due to the integrated parallel computing capability. A flow diagram of the AMIEGO framework is specified below, where the red blocks are derived from the EGO framework for the exploration of the integer design space, and the blue blocks represent the gradient-based exploration for the continuous design space

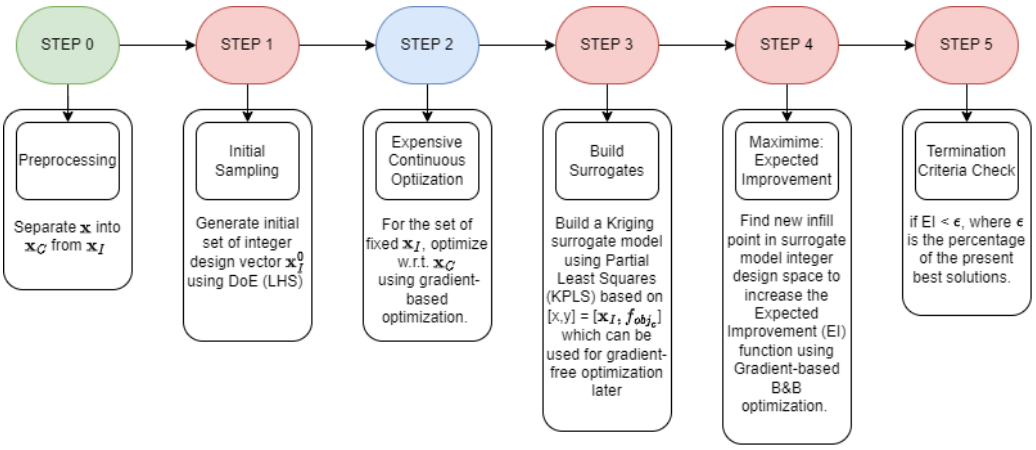


Figure 31: An overview of the AMIEGO framework for MINLP problems. Figure reproduced from Roy et al. [57]

Decision Hierarchy in System Architecture Optimization One of the most troublesome attributes of System Architecture Optimization (SAO) is the fact that the design space is highly discontinuous due to the incompatibility of certain discrete design variables. This is due to the fact that for a system architecture, some discrete design variables are active (1), while others are inactive (0). As a result of this, some continuous design variables do also become active or inactive. This is a phenomena described in optimization as decision hierarchy [10]. Methods to deal with this phenomena have been described numerously in literature [11] [16] [69].

Due to the fact that gradient-based optimization algorithms have trouble dealing with such discontinuous design spaces, a gradient-free approach is normally used. To explicitly deal with the hierarchical connection between some discrete and continuous design variables, an intermediate step should be added between the optimizer and the MDA block. This steps consists of an architecture generation module, which makes sure that only valid and feasible system architectures are to be evaluated in the continuous optimization loop. This separates the feasible architecture design space from the apparent architecture design space as a result of the (in)compatibility constraints [11]. This is done through the process of imputation, which modifies the design vector to only deal with feasible design variables as a result of hierarchy. This method is presented below:

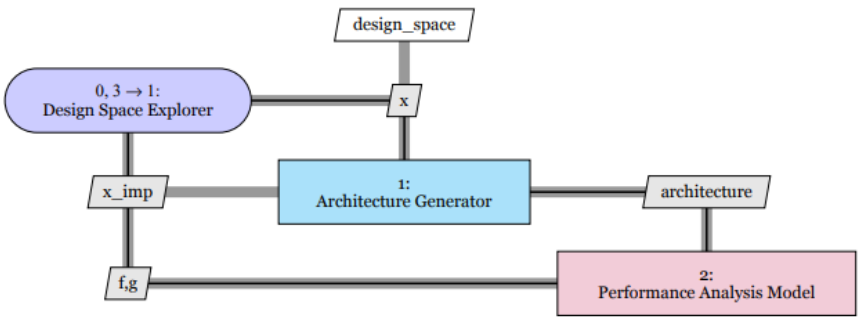


Figure 32: An XDSM representation of the added *imputer* step which deals with the hierarchical nature of the design vector.

Here, the imputer tries to solve the problem where two design vector's might look very different, but in turn result in similar, equally (in)feasible, system architectures. An example here might be that a battery electric vehicle with one diesel generator is a similar system architecture as a battery electric vehicle with three diesel generators, as the discrete decision variables of the diesel generator are inactive, however their design vectors look very different. As both design vectors (system architectures) are infeasible/incompatible, they should both be removed from the design space to increase the performance of the optimization algorithm. Three methods, including the method of imputation, of dealing with this decision hierarchy within the design vector are further explained below, as explained by De Smedt [16]:

1. **Naive**: No methods to deal with decision hierarchy are implemented. Hence all design variables are every iteration evaluated, leading to large number of unnecessary evaluations.
2. **Imputation**: Integrate an *imputer* between the optimizer and evaluation steps to impute the design vector to only deal with active variables. This results in only necessary function evaluations, but this method might not be easily integrated in existing optimization algorithms.
3. **Explicit Consideration**: By mathematically integrating the (in)compatibility of certain design variables in the design space definition, more accurate models can be applied. This however requires intrusive approaches which alter the design space a priori.

4.5.4 Comparing gradient-based and gradient-free optimization algorithms

Since the selection of the type of optimization algorithm is highly dependent on the type of problem formulation, the reader should first gain proper understanding of the requirements that flow from the problem formulation. Problem specific traits which effect the optimizers performance might be the cost of model evaluation, the convexity of the design space and whether it is multi- or single objective etc.

Lyu et al. [43] performed a comparison study on the various optimization algorithms applied to the Rosenbrock function (gradient-based and gradient-free), to compare the number of function evaluations and the degree of design variables. Here, both analytical and finite-difference methods were used to compute the derivatives for the gradient-based algorithms. The results are summarised in Figure 33.

Some key takeaways from this benchmarking are:

- The gradient-free optimization algorithms require already significant amount more function evaluations at relatively low number of design variables. Here the NSGA-II algorithm and ALPSO algorithm are compared, which are state-of-the-art genetic algorithms and particle swarm optimizers.
- The gradient-based optimization algorithms require far-less function evaluations as compared to gradient-free algorithm. Here, finite-difference methods requiring more function evaluations than analytical derivative calculations, which is expected.
- The increase in function evaluations with an increase in dimensionality of the problem is far larger for gradient-free algorithms as compared to gradient-based algorithms.

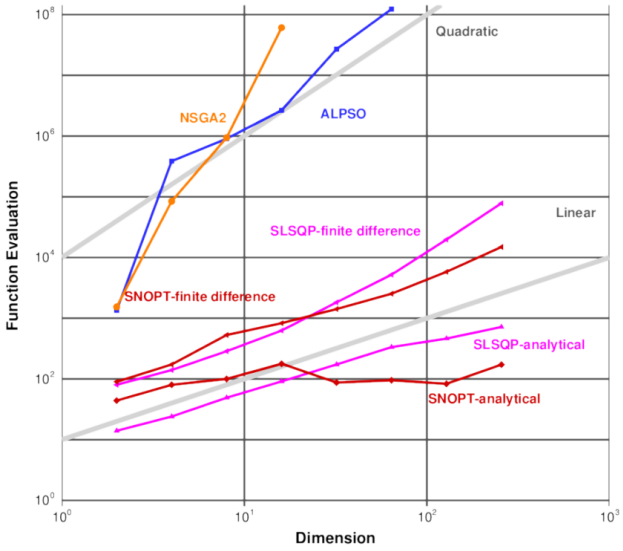


Figure 33: Benchmarking test performed for comparison of different optimisation algorithms[43]

4.6 Uncertainty-Based Multidisciplinary Design Optimization

Designing any project or system requires proper understanding of the knowledge at hand. However, since in the beginning of any design project, extensive knowledge is still missing, it is important to acknowledge and account for uncertainty throughout this design process. This uncertainty can originate from various sources, such as changing requirements, unforeseen technological challenges or even fluctuations in available resources. Quantifying uncertainty has been proven difficult, resulting in exceedance of project cost, time and performance budgets. This is visualized in Table 3, for some NASA projects in the late 1990s / early 2000s.

Table 3: Design margins for several NASA projects in the late 1990s / early 2000s [65]

	Value		Margin		
	Predicted	Actual	Allocated [%]	Actual [%]	Difference [%]
Mars Pathfinder					
Entry Mass	390 kg	580 kg	28.2	48.7	+20.5
Cost	$100M	$171M	50	71.0	+21.0
Clark					
Schedule	1.8 years	3.6 years	11.1	100.0	+88.9
Cost	$44M	$55M	11.3	25.0	+13.7
Deep Space 1					
Schedule	2.3 years	3.1 years	17.3	34.7	+17.4
Cost	$128M	$152.3M	10.1	19.0	+8.9
Mars Exploration Rover					
Mass	918 kg	1062 kg	8.2	15.7	+7.5
Cost	$630M	$820M	20.0	30.0	+10.0

4.6.1 Safety Factors are Stupid

In today's rapidly evolving technological landscape, industries across the board trying to integrate cutting-edge innovations into their products and services. However, this pursuit of innovation is not without its challenges. One of the biggest hurdles that industry is facing is the inherent uncertainty that comes with these groundbreaking technologies. There is often very little pre-existing knowledge available about these innovative technologies. This lack of prior experience and data significantly amplifies the level of unpredictability surrounding these technologies.

In traditional design processes, uncertainty was only taken into account in the conceptual design phase, by reformulating constraints using empirical relations or predefined factors [75]. This meant that in order to take into account this uncertainty as a result of lack of knowledge, safety factors were applied. Safety factors are based on past experience and prior knowledge about the system, but limit the design space thereby rarely resulting in an optimal design. As a result, traditional designs usually turn out to be too conservative or over-redundant. Therefore, a request for a more sophisticated way of taking into account uncertainty in the conceptual design phase is required.

4.6.2 Integrating UMDO in ADSE

Uncertainty-Based Multidisciplinary Design Optimization (UMDO) is a specific field within multidisciplinary optimization which is gaining traction within industry. Optimization under uncertainty takes into account the fact that optimal, real-world, design is highly uncertain, as a result of the lack of knowledge on how the system will behave under real-world conditions.

To account for this lack of knowledge, Uncertainty-Based Multidisciplinary Design Optimization models' its design variables and models as stochastic, instead of deterministic, which is most commonly used in MDO applications. This means that instead of assuming that all models and systems have a deterministic outcome, it models' them as a mean with some variance. By actively taking into account this uncertainty in the optimization loop, the system engineer can say with more confidence that the optimal design will indeed result in a real-world optimal design.

UMDO, is more generally referred to in literature as Reliability-Based Robust Design Optimization (RBRDO), as it takes into account robustness and reliability in the optimization, either as objective or constraint. This is a combination of two separate uncertainty-based optimization fields, namely Robust Design Optimization (RDO) and Reliability-Based Design Optimization (RBDO) [75].

The integration of RBRDO in architectural design space exploration and optimization is something which, to the author's knowledge, has not been implemented or tested. The use case and potential gains of applying this optimization under uncertainty in architectural design space exploration is however very clear. Since architectural design space exploration is performed in conceptual design phases, when there is very little knowledge about what system behavior is required, taking into account this uncertainty to come up with a robust and reliable optimal design is highly beneficial.

This is due to the fact that the decisions already made in the conceptual design phases, highly effect the outcome and performance of the final overall design [39]. By allowing the optimizer to evaluate all possible design options, all while assuming stochastic design variables and model disciplines, the system engineer can say with more confidence that the chosen optimal design will indeed result in a real-world optimal design which will satisfy the set constraints.

4.6.3 UMDO Theory

It is important to understand the various concepts applied within Uncertainty-Based Design Optimization. Therefore, first some concepts are explained by Yao et al. [75].

Definition 4 Uncertainty *is the incompleteness in knowledge and the inherent variability of the system and its environment.*

Definition 5 Robustness *is the degree of tolerance of the system to be insensitive to variations in both the system itself and the environment.*

Definition 6 Reliability *is the likelihood that a component (or a system) will perform its intended function without failure for a specified period of time under stated operating conditions.*

In mathematics there are various theories to model these uncertainties, ranging from probability in probability theory to belief and plausibility in evidence theory. In this review, only probability will be taken into account, as its most applicable to engineering design processes.

Uncertainty can be classified in two distinct sections, which was first proposed in risk assessment, namely aleatory uncertainty and epistemic uncertainty.

Definition 7 Aleatory Uncertainty *describes the inherent variability of the physical system or the environment under consideration. This is a type of irreducible uncertainty, meaning it can not be eliminated by simply increasing the amount of data or knowledge about the system.*

Definition 8 Epistemic Uncertainty *describes the variation of the physical system or the environment due to a lack of knowledge about the behavior of the system. This leads to inaccuracy but can be solved for my gathering more useful data or increasing the knowledge about the system.*

Thunnissen [66] [65], proposes a method to incorporate also the classification terms of ambiguity and interaction, where ambiguity represents the little precision in general communication, and interaction uncertainty stems from unforeseen interactions between events or fields. The definition of uncertainty is further explained by modelling simulation prediction uncertainty as internal and external uncertainty [4]. Internal uncertainty refers to the inherent variability and uncertainty within the model itself, which can be split up into model structure uncertainty (uncertainty as a result of underlying model assumptions) and model parameter uncertainty (uncertainty in defining proper model parameters for the use case) [75]. External uncertainty is the model input uncertainty referring to the variability of input to the simulation model. Lastly, since simulation models are used in conceptual design phases and architectural design space exploration, model uncertainty in the form of model error is present, which is a result of discretization- or round-off errors.

4.6.4 Robust Design Optimization

Robust Design Optimization (RDO) was first introduced by the Japanse engineer Genichi Taguchi, then named the Taguchi method, to improve the quality of manufacturing goods. As a result, the manufactured product is performance insensitive to variation beyond the control of manufacturers [75] [64] [51]. The Taguchi method works by maximizing so-called signal-to-noise ratio's, where the signal is the desired performance of the system, and the noise is the variability of the system. By increasing the signal-to-noise ratio of a system, robustness is increased. It evaluates these signal-to-noise ratio's by modelling a system as either control factors or noise factors. Here control factors are within an engineers field of influence and hence can be controlled, whereas noise factors are external factors which can influence the process but cannot be controlled by the system engineer. The goal is to isolate and minimize these noise factors, or at least limit the systems sensitivity with respect to these noise factors.

The RDO problem formulation is given by Equation 4.12

$$\begin{aligned} \text{find} \quad & \mathbf{x} \\ \min \quad & f(\mathbf{x},\, \mathbf{p}) = F(\mu_f(\mathbf{x},\mathbf{p}), \sigma_f(\mathbf{x},\mathbf{p})) \\ \text{s.t.} \quad & \mathbf{g}(\mathbf{x},\mathbf{p}) \leq 0 \\ & \mathbf{x}^L \leq \mathbf{x} \leq \mathbf{x}^U \end{aligned} \tag{4.12}$$

The goal of an RDO problem is to minimize the variance in the objective function at an optimal design point. In other words, minimize the system sensitivity to uncertainties in the design. The most simplified function for $F(\mu_f(\mathbf{x},\mathbf{p}), \sigma_f(\mathbf{x},\mathbf{p}))$ is the weighted sum of the mean and standard deviation [75].

A visualization of the difference between an global optimum, and a robust optimum (hence minimization of the variability of objective function with variance in design vector), is given in Figure 34

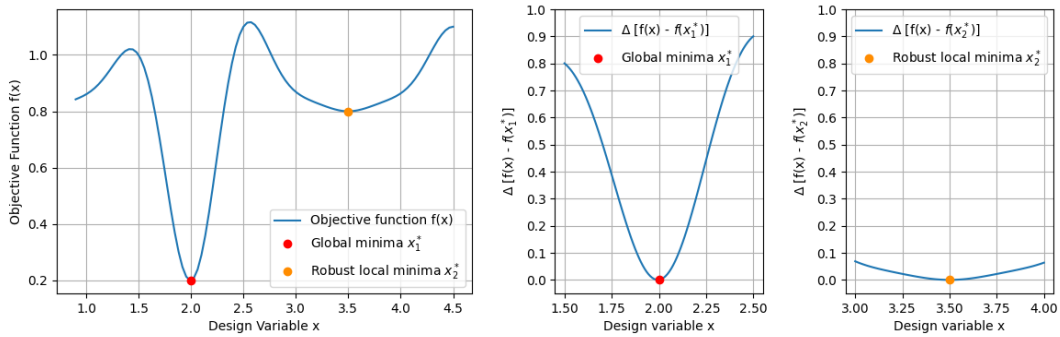


Figure 34: Visualization of a global optima in comparison to a robust local optima. The two right figures display the change in objective function for the same range of variance around the minima (± 0.5)

4.6.5 Reliability-Based Design Optimization

Reliability-Based Design Optimization (RBDO) is a type of stochastic optimization which is more focused on satisfying constraints under uncertainty than minimizing the objective function. The mathematical formulation of an RBDO problem using probability theory is given in Equation 4.13 [75]:

$$\begin{aligned} \text{find} \quad & \mathbf{x} \\ \min \quad & f(\mathbf{x},\ \mathbf{p}) = \mu_f(\mathbf{x},\mathbf{p}) \\ \text{s.t.} \quad & P(\mathbf{g}(\mathbf{x},\mathbf{p}) \leq 0) \geq \mathbf{R} \\ & \mathbf{x}^L \leq \mathbf{x} \leq \mathbf{x}^U \end{aligned} \tag{4.13}$$

Equation 4.13 is focused on finding a minimal mean objective function given the fact that the probability (P) of constraint satisfaction is larger than some specified reliability vector $\mathbf{R}$. This is clearly visualized in Figure 35 for the "Bean contour" problem. Here it can be seen that after finding a deterministic local optima (red dot), given the infeasible design space due to constraints (light red), the area of uncertainty may lead to a non-feasible design. Therefore, taking into account the probability that constraints are satisfied within some specified range of reliability, specified in gray, the reliable local optima is found (orange dot).

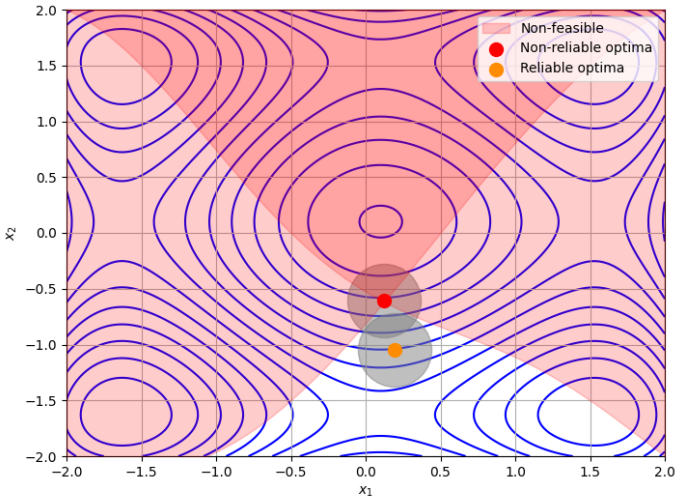


Figure 35: Visualization of a found deterministic local optima in comparison to a reliable based local optima. Here the range of uncertainty is specified in gray, showing that for the reliable design optima, no constraints are violated.

4.6.6 Uncertainty Modelling, Propagation and Analysis

In order to implement uncertainty of design- and model variables into the optimization process, a clear understanding of uncertainty quantification is required. This is highly dependent on the underlying basis of statistics. The uncertainty in model output or objective function is a result of the variability in model inputs and the propagation of that uncertainty through the analysis. Uncertainty quantification approaches work through either intrusive or non-intrusive approaches. An overview of the most used approaches in literature is given in below.

Intrusive: Polynomial Chaos is a method which reformulates the governing equations of an MDAO problem formulation such that uncertainty quantification can directly be incorporated into the system. Polynomial Chaos assumes smoothness across its function evaluations, objective and constraints, as it approximates outputs using polynomial approximations (Gaussian, Simpson, Lagrange). Polynomial chaos approximates a function as the linearized sum of basis functions Ψ_i and weights w_i, as given by Martins [45] as:

$$f(x) \approx \sum_{i=1}^{n} \Psi_i(x) w_i \tag{4.14}$$

Here, the weights w_i are computed to fit to a desired orthogonal basis function. From this, for instance, statistical moments and other interesting information (PDF) can be derived. Due to the fact that this method makes use of orthogonal basis functions, the following statement should always hold:

$$\langle \Psi_i, \Psi_j \rangle = 0 \quad \text{if} \quad i \neq j \tag{4.15}$$

By allowing sampling methods to be used (e.g. MCS), the statistical moments can easily be computed. This requires approximately, as a rule of thumb, twice the number of unknowns $(n+1)$ as samples (m).

Non-intrusive: Monte Carlo Simulation is one of the most used methods to quantify model output uncertainty as a function of model input uncertainty and uncertainty propagation. Monte Carlo Simulation (MCS) is a sampling-based method, where recurrent sampling and simulation is performed to compute the statistics of the model output. MCS works by running significant amount of simulations, all randomly sampled, such that from that, the output distribution can be approximated. By increasing the number of simulations, the output distribution can be approximated with higher accuracy, at the expensive of increased computational cost.

Monte Carlo Simulation makes use of the law of large number which states that if enough samples are evaluated, our output statistics converge to the actual values [45]. The first and second statistical moments can be computed as follows:

$$\mu_f = \frac{1}{n_s} \sum_{i=1}^{n_s} f_i \tag{4.16}$$

$$\sigma_f^2 = \frac{1}{n_s - 1} \left(\sum_{i=1}^{n_s} (f_i)^2 - n_s \mu_f^2 \right) \tag{4.17}$$

An example which clearly shows the power of MCS for a simple 1D function, is given in Figure 36.

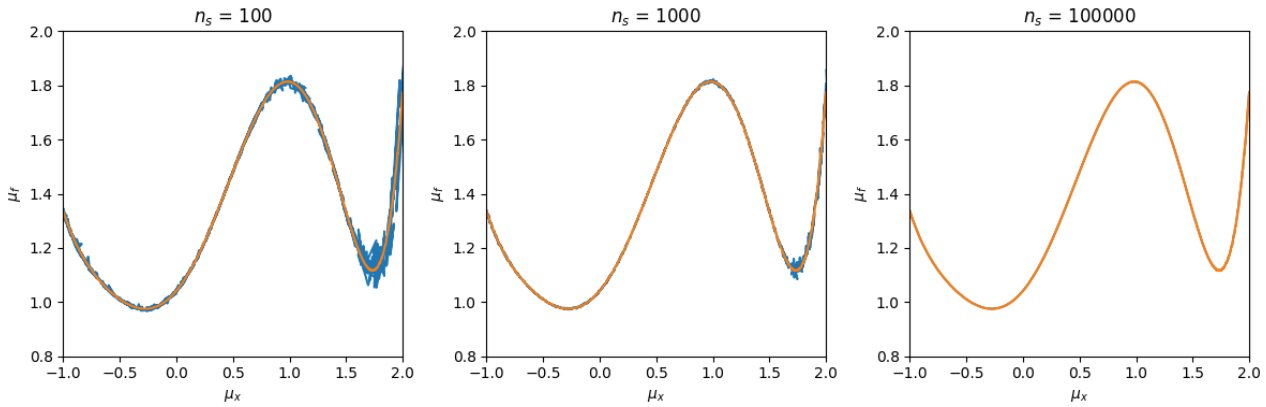


Figure 36: Visualization of applying MCS to a simple 1D problem to outline the behavior of the objective function's mean as a function of the mean of design variable x and the number of sampling points. Figure reproduced from Martins [45].

Some benefits ([+]) and drawbacks ([-]) of using MCS:

+ The convergence rate is independent on the number of inputs, due to the randomization of input variables for each sample. MCS therefore does not suffer from the curse of dimensionality, and is more efficient at high-dimensional problems as compared to other non-intrusive uncertainty propagation techniques.
+ The MCS method can be run in parallel, as all function evaluations are independent.
+ MCS provides more information regarding the probability density function, instead of only providing mean and variance (summary statistics)
- MCS requires a high sampling number to achieve reliable results, i.e. convergence to the real solution is slow $\mathcal{O}(1/\sqrt{n_s})$.

To increase the convergence rate, smart sampling techniques were developed which still resulted in the required accuracy. These methods are the Latin Hypercube Sampling (LHS) and low-discrepancy sequences (i.e. quasi-Monte Carlo). These methods increase the convergence rate to $\mathcal{O}(1/n_s)$, which is still only one order of magnitude larger than standard randomly sampled MCS. The difference in convergence between the various sampling methods is visualized in Figure 37 [45].

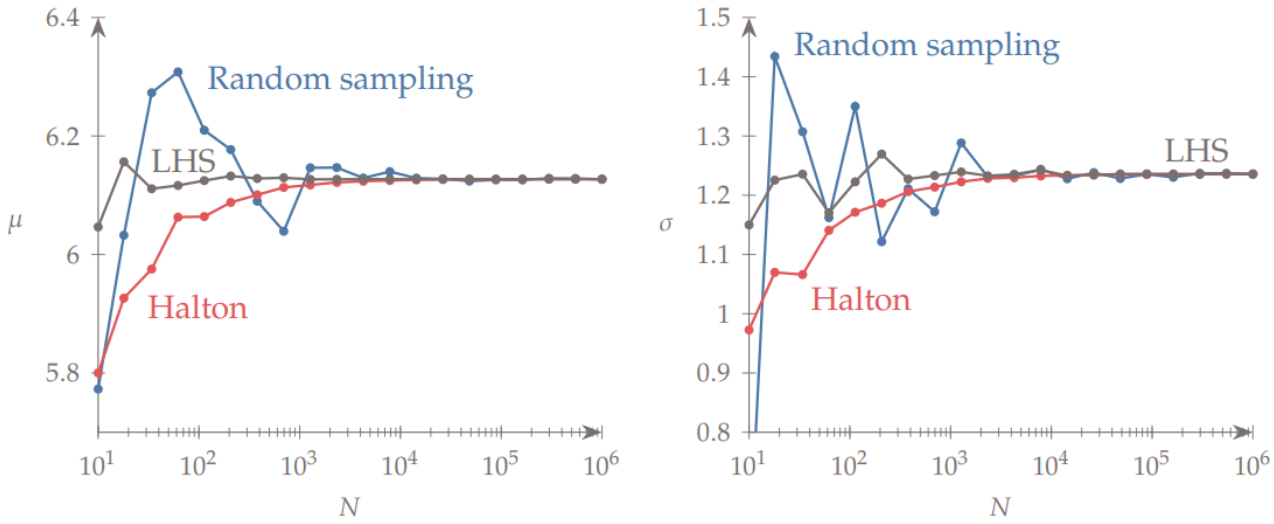


Figure 37: Convergence behavior for the various sampling methods used in MCS for a standardized optimization under uncertainty problem. [45].

Non-intrusive: First-order perturbation method relies on Taylor-series approximations to determine statistical moments, by taking into account the partial derivatives of the objective function f with respect to the elements of a random design variable vector $\mathbf{x}$. Using uncertainty propagation of the random design variable $\mathbf{x}$, the mean and standard deviation can be computed as follows:

$$f(\mathbf{x}) \approx f(\mu_x) + \sum_{i=1}^{n} \frac{\partial f}{\partial x_i}(x_i - \mu_{x_i}) \tag{4.18}$$

$$\begin{aligned} \mu_f &= \mathbb{E}(f(\mathbf{x})) \\ &\approx \mathbb{E}(f(\mu_x)) + \sum_i \mathbb{E}\left(\frac{\partial f}{\partial x_i}(x_i - \mu_{x_i})\right) \\ &= f(\mu_x) + \sum_i \frac{\partial f}{\partial x_i}(\mathbb{E}(x_i) - \mu_{x_i}) \\ &= f(\mu_x) + \sum_i \frac{\partial f}{\partial x_i}(\mu_{x_i} - \mu_{x_i}) \\ &= f(\mu_x) \end{aligned} \tag{4.19}$$

$$\begin{aligned} \sigma_f^2 &= \mathbb{E}(f(\mathbf{x})^2) - (\mathbb{E}(f(x)))^2 \\ &\approx \mathbb{E}\left[f(\mu_x)^2 + 2f(\mu_x)\sum_i \frac{\partial f}{\partial x_i}(x_i - \mu_{x_i}) + \sum_i \sum_j \frac{\partial f}{\partial x_i}\frac{\partial f}{\partial x_j}(x_i - \mu_{x_i})(x_j - \mu_{x_j})\right] - f(\mu_x)^2 \\ &\sum_i \sum_j \frac{\partial f}{\partial x_i}\frac{\partial f}{\partial x_j}\mathbb{E}\left[(x_i - \mu_{x_i})(x_j - \mu_{x_j})\right] \end{aligned} \tag{4.20}$$

Here, the last expectation term can be explained as the covariance matrix, which shows the covariance between the components of the input vectors. By assuming mutually independent random input variables, this expression can be rewritten to:

$$\sigma_f^2 = \sum_{i=1}^{n} \left(\frac{\partial f}{\partial x_i}\sigma_{x_i}\right)^2 \tag{4.21}$$

Some benefits([+]) and drawbacks ([-]) of using first-order perturbation methods based on Taylor expansion are:

+ Highly simplistic, thereby easy to integrate into an MDAO problem formulation.
- Lack of accuracy due to linearization and the assumption of uncorrelation of uncertain parameters.
- It assumes symmetry in the input distributions, as only the first and second statistical moment are used.
- It assumes a continuously differentiable probability distribution, which might not be the case.
- It becomes computationally expensive to determine the statistical variance as given in Equation 4.21, due to the required computation of derivatives, which might not be available.

Usually an assumption on the probability distribution is needed for the constraints and required reliability level (RBDO). As this methods assumes uncorrelated active constraints, Equation 4.21 can be used. According to Martins [45], this methods is easy to use and results in a certain magnitude of error, which is appropriate for the conceptual design phase.

An extension to this method is presented by Gu et al [24], Cao and Duan [12], and Du and Chen [18]. They propose a method to generate a worst case estimate of the propagated uncertainty in the systems output from a multidisciplinary system analysis. Here, the MDAO problem formulation is subject to simulation tool bias and input uncertainty (or input precision error). They are able relate the change in output function of the various disciplines to the uncertainty in input design vector and the simulation bias error accordingly:

$$\begin{Bmatrix} \Delta y_1 \\ \Delta y_2 \\ \vdots \\ \Delta y_n \end{Bmatrix} = \begin{Bmatrix} \frac{dy_1}{dx} \\ \frac{dy_2}{dx} \\ \vdots \\ \frac{dy_n}{dx} \end{Bmatrix} \cdot \Delta \mathbf{x} + \begin{bmatrix} I_1 & -\frac{\partial T_1}{\partial y_2} & \cdots & -\frac{\partial T_1}{\partial y_n} \\ -\frac{\partial T_2}{\partial y_1} & I_2 & & \\ \vdots & & \ddots & \vdots \\ -\frac{\partial T_n}{\partial y_1} & & \cdots & I_n \end{bmatrix}^{-1} \cdot \begin{Bmatrix} \Delta T_1(\mathbf{x}, \mathbf{y}_2, ..., \mathbf{y}_n) \\ \Delta T_2(\mathbf{x}, \mathbf{y}_1, \mathbf{y}_3, ..., \mathbf{y}_n) \\ \vdots \\ \Delta T_n(\mathbf{x}, \mathbf{y}_1, ..., \mathbf{y}_{n-1}) \end{Bmatrix} \tag{4.22}$$

Using this, the worst case estimation occurs when all variables have the same or equal sign. This can be achieved by taking the absolute values of all parts, i.e.:

$$\begin{Bmatrix} \Delta y_1 \\ \Delta y_2 \\ \vdots \\ \Delta y_n \end{Bmatrix} = \begin{Bmatrix} |\frac{dy_1}{dx}| \\ |\frac{dy_2}{dx}| \\ \vdots \\ |\frac{dy_n}{dx}| \end{Bmatrix} \cdot |\Delta \mathbf{x}| + \begin{bmatrix} |I_1| & |-\frac{\partial T_1}{\partial y_2}| & \cdots & |-\frac{\partial T_1}{\partial y_n}| \\ |-\frac{\partial T_2}{\partial y_1}| & |I_2| & & \\ \vdots & & \ddots & \vdots \\ |-\frac{\partial T_n}{\partial y_1}| & & \cdots & |I_n| \end{bmatrix}^{-1} \cdot \begin{Bmatrix} |\Delta T_1(\mathbf{x}, \mathbf{y}_2, ..., \mathbf{y}_n)| \\ |\Delta T_2(\mathbf{x}, \mathbf{y}_1, \mathbf{y}_3, ..., \mathbf{y}_n)| \\ \vdots \\ |\Delta T_n(\mathbf{x}, \mathbf{y}_1, ..., \mathbf{y}_{n-1})| \end{Bmatrix} \tag{4.23}$$

This method of using worst-case uncertainty estimation in coupled MDAO problems is applied by Gu et al. [24] to a Robust Design Optimization problem, where both the variation in objective function and the variation in constraints is taken into account. Here, the variation in objective function and constraint is given by:

$$\Delta f = \sum_i \left| \frac{\partial \bar{f}(\mathbf{x}, \mathbf{y})}{\partial x_i} \Delta x_i \right| + \left| \frac{\partial \bar{f}(\mathbf{x}, \mathbf{y})}{\partial y_k} \Delta y_k \right| \tag{4.24}$$

$$\Delta g_j = \sum_i \left| \frac{\partial \bar{g}(\mathbf{x}, \mathbf{y})}{\partial x_i} \Delta x_i \right| + \left| \frac{\partial \bar{g}(\mathbf{x}, \mathbf{y})}{\partial y_k} \Delta y_k \right| \tag{4.25}$$

Here, Δx_i is the worst case estimation of variability in the i-th design variable x_i and Δy_k is the worst case estimation of uncertainty in the k-th system state (output k-th discipline) using Equation 4.23. Similar for the j-th constraint g_j, Δg_j is the estimated variation of the j-th constraint g_j. Please note, since linearization is applied, these assumptions for constraints and objectives are only valid if variability is small, higher-order terms are neglected.

The robust and reliable MDAO problem formulation can then be rewritten, with relaxed equality constraints:

$$\begin{aligned}
\text{find} \quad & \mathbf{x} = [x_1, x_2, ..., x_n]^T \\
\text{min} \quad & f^R(\mathbf{x}) = \alpha \cdot f + (1 - \alpha) \cdot \Delta f \\
\text{s.t.} \quad & g_j^R(\mathbf{x}) = g_j - \Delta g_j \geq 0 \quad j = 1, 2, ..., J \\
& \mathbf{x}_i^{L^R} \leq \mathbf{x}_i \leq \mathbf{x}_i^{U^R}, \quad i = 1, 2, ..., n \\
\text{where} \quad & \mathbf{x}_i^{L^R} = x_i^L + \Delta x_i \quad \text{and} \quad \mathbf{x}_i^{U^R} = x_i^U - \Delta x_i \\
& f = f(\mathbf{x}) = \bar{f}(\mathbf{x},\mathbf{y}) \quad \text{and} \quad g = g_j(\mathbf{x}) = \bar{g}_j(\mathbf{x},\mathbf{y}) \\
& \Delta f = \sum_i \left| \frac{\partial \bar{f}(\mathbf{x}, \mathbf{y})}{\partial x_i} \Delta x_i \right| + \left| \frac{\partial \bar{f}(\mathbf{x}, \mathbf{y})}{\partial y_k} \Delta y_k \right| \\
& \Delta g_j = \sum_i \left| \frac{\partial \bar{g}(\mathbf{x}, \mathbf{y})}{\partial x_i} \Delta x_i \right| + \left| \frac{\partial \bar{g}(\mathbf{x}, \mathbf{y})}{\partial y_k} \Delta y_k \right| \\
\text{and} \quad & 0 \leq \alpha \leq 1
\end{aligned} \tag{4.26}$$

Here, the value of α can be explained as the weighting factor to determine the significance of both optimal design and robust and reliable design. If $\alpha > 0.5$, then more significance is put on the minimization of the objective function, whereas if $\alpha < 0.5$, more focus is put on the robustness of the design. Hence this can be explained as a bi-objective optimization problem.

4.6.7 Sensitivity Analysis

Sensitivity Analysis is a method widely used in UMDO to simplify the problem formulation [30] [70] [49]. This is done, as most UMDO problem formulation and solution methods rely on computationally heavy algorithms, such

as MCS (section 4.6.6) or Polynomial Chaos (section 4.6.6), which is undesired. To limit the freedom of the problem, i.e. the dimensionality of the problem, sensitivity analysis is applied. Sensitivity Analysis (SA) studies the variability of the model output (objective) as a function of it's design inputs (design variables). From this, certain conclusions can be drawn on how sensitive an output is to a change in input. By eliminating the insensitive relations, meaning that uncertainty modelling and propagation on insensitive design variables will not be done, the problem of dimensionality in UMDO can be decreased.

4.7 Key Takeaways

1. Mixed-Integer Non Linear Programming (MINLP) optimizers such as AMIEGO are most suited for ADSE due to the inherent mix of both discrete and continuous design variables. By allowing for efficient handling of the discrete design space using surrogate models (Kriging), a global optima can be found with relatively low computational effort. It furthermore combines the most efficient optimization algorithms for the two distinct design spaces (gradient-free for $\mathbf{x}_I$ and gradient-based for $\mathbf{x}_C$)

2. Uncertainty can be modelled as either aleatory or epistemic. Aleatory uncertainty is the inherent variability of the physical system (non-deductible), whereas epistemic is the variability of the physical system as a result of lack of knowledge.

3. Robust Optimization is focused on minimizing a weighted average of the mean optimal objective function and variance at that design point.

4. Reliability-Based Optimization focuses on minimizing an objective for which constraint satisfaction taking into account stochastic design variables is achieved.

5. A Monte Carlo Simulation is too computationally expensive to incorporate directly into the multidisciplinary optimization of system architectures. Therefore the use of First-Order Perturbation Methods, which rely on the Taylor expansion of objectives and constraints, prove to be most suitable to quantify the propagation of input and model bias uncertainty. This method was first proposed by Gu et al. [24]

6. The method of worst-case estimation proves to be the most suitable tool for uncertainty quantification and propagation in system architecture optimization in conceptual design. This is due to the relatively low computational burden on the optimization, while still achieving a high level of accuracy.

7. Sensitivity Analysis can be used to find what variables have a large influence on the outcome of the objective near its optimal design point. For variables which do not have a large sensitivity on the objective function, taking into account uncertainty is not necessary / will not increase the robustness or reliability of the optimal design. Hence, these shall be removed without significant loss of accuracy.

8. In ADSE, it can be interesting to determine a small set-of feasible optimal designs (from for instance a Pareto-front), to which RBRDO using MCS will be applied. This alleviates the computational burden of robust optimization. Other sampling methods can be used to increase convergence rates (LHS & Halton)

5 Dynamic MDAO workflow

In order to properly define an executable MDAO workflow, various methods and toolboxes have been developed by profound institutes, such as the TU Delft and the MDO Lab from the University of Michigan, to increase the productivity and enhance performance of solving MDAO problems. An MDAO problem formulation is usually visualized using either an N2 chart or an eXtended Design Structure Matrix (XDSM), an extended version of the Design Structure Matrix (DSM) first proposed by Lambe and Martins [40]. Here, all aspects that define an MDAO problem formulation are visualized, including the flow of (coupled) design variables, the type of convergence scheme in MDA and the chosen gradient-based/free optimizer. An example of an XDSM diagram with a simple MDF set-up with MDA convergence scheme is shown in Figure 38 [45].

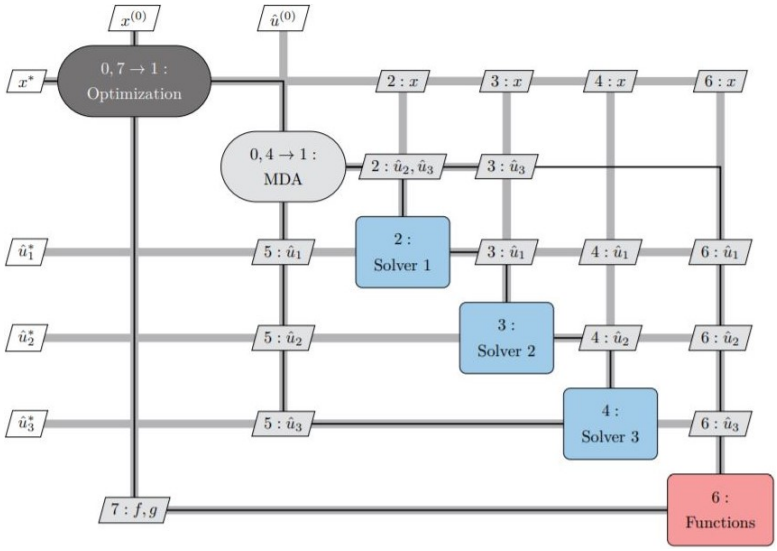


Figure 38: MDF architecture[45]

5.1 What are dynamic MDAO workflows

In architectural design space exploration, due to the high level of dimensionality and the fact that multiple component types can be combined to compose a large number of system configurations, the MDAO problem formulation is not constant, but can vary between iterations. An example might be in the design of a system architecture of an eVTOL aircraft, where the power generation source might change from batteries, to turbo-generator or fuel cells. This results in the fact that various analysis tools might be necessary to find an optimal solution. This variation of analysis tools, constraints and design variables is something that should be accounted for in a systematic way to ensure successful use of MDAO. A clear definition of dynamic MDAO workflows therefore is:

Definition 9 Dynamic MDAO workflows *is the dynamic (between iterations) alteration of the configured MDAO computational system, including its design variables, constraints and MDA disciplines, to ensure that optimization can succesfully be executed.*

Since most MDAO problem formulations (XDSM) that were presented in section 4 remain fixed (static) between iterations, it can become challenging to apply such methods in a dynamic way. Especially the correct sequencing of design disciplines is important here, as it should allow for proper flow of information (feed-forward/backward) to ensure fast convergence. Various solutions have been developed to solve and automate this process, which will be discussed in the sections below.

5.2 KADMOS

KADMOS is a novel methodology and software package first presented by van Gent en La Rocca [21] to increase the agility of design teams in MDAO, by allowing for easy assembly, adjustment and reconfiguration of MDAO

problem formulations. It does this through a graph-based methodological approach where an executable MDAO problem is formulated by splitting up a problem formulation and it's design variables, objectives and constraints in four distinct graphs:

- Repository Connectivity Graph (RCG) (subsubsection 5.2.1)
- Fundamental Problem Graph (FPG) (subsubsection 5.2.2)
- MDAO Data Graph (MDG) (subsubsection 5.2.3)
- MDAO Process Graph (MPG) (subsubsection 5.2.4)

This MDAO problem formulation can thereafter be converted into a MDAO executable workflow through CMDOWS data schemes, as first proposed by van Gent, La Rocca and Hoogreef [22]. These data schemes can then be imported into PIDO applications such as RCE, Optimus or OpenMDAO through parsers (OpenLEGO for OpenMDAO) [71] [4]. It has been estimated that KADMOS is able to resolve halve the time typically required to set up, formulate, adjust and reconfigure a complex MDAO system. It does this while making sure that MDAO experts maintain oversight and control of the overall MDAO computational system. It furthermore allows for distributed and heterogeneous design teams to work efficiently in an MDAO setting, eventually reducing the "implementation gap", as first explained by van Gent and La Rocca [21], even further.

5.2.1 Repository Connectivity Graph

The Repository Connectivity Graph (RCG) is a graph which is build up of all the different analysis tools provided by the user to KADMOS including its in- and outputs. Based on the given set of disciplinary analysis tool and an XML file including its inputs and outputs, a connectivity graph is produced which links all the relevant disciplines. Here, it should be noted that the irrelevant analysis tools, which have no contribution to the MDA block, are removed, resulting in a directed graph showing the flow of information between various tools. An example of this is presented below in Figure 39:

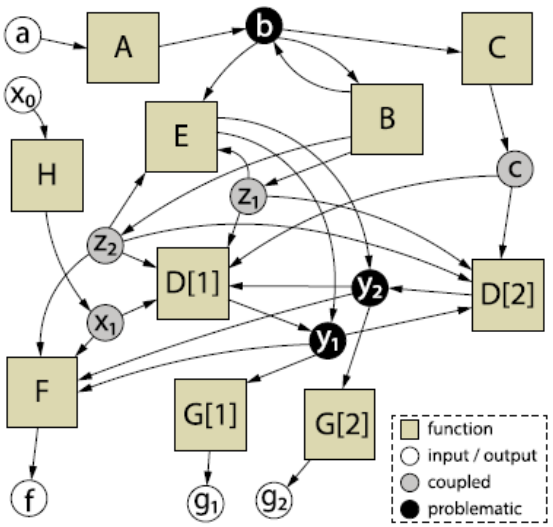


Figure 39: Repository Connectivity Graph for the Sellar problem

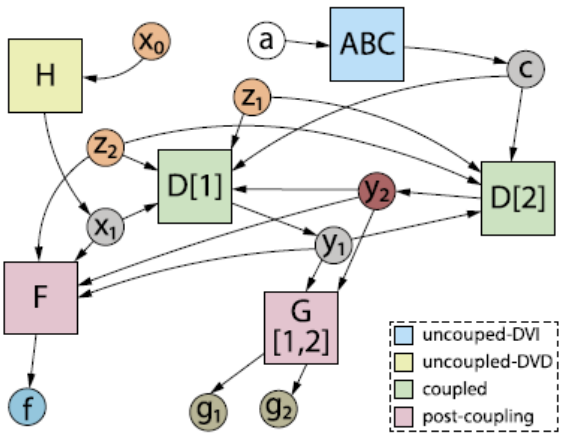


Figure 40: Fundamental Problem Graph for the Sellar problem

[4] https://github.com/daniel-de-vries/OpenLEGO, accessed on 26/10/2023

5.2.2 Fundamental Problem Graph

A Fundamental Problem Graph (FPG) is a more advanced graph produced by KADMOS which is derived from the RCG, but here, graph manipulations have been performed to come up with a feasible problem graph. As mentioned, disciplines and its I/O which have no contribution will be removed, as well as circular dependencies, which are a limitation to KADMOS [54]. This is the first instance where an MDAO architecture can be imposed, such as MDF, IDF or DoE. In the FPG, also sequencing and merging algorithms are applied to compile the workflow in a logical order, based on the I/O of every discipline. A problem which could occur in KADMOS is whenever the evaluation of certain disciplines is not instant, resulting in unknown values / sparsity in the Centralized Data Scheme (CDS). This is solved by implementing a sleep function in Python between the execution of various disciplines, thereby making sure that all relevant information is added to the CDS.

5.2.3 MDAO Data Graph

The MDAO Data Graph (MDG) is a graph used to store the exchanged data for the various blocks (sequenced I/O) as well as the disciplines itself stored in a CDS, for a given MDAO solving architecture. An example for the Sellar problem obtained from van Gent et al [21] is given in Figure 41

5.2.4 MDAO Process Graph

The MDAO Process Graph (MPG) is a simplified graph of the MDG where only the executable blocks and its sequencing is stored. All of this is then converted into an executable workflow using for instance CMDOWS [22] files and parsers such as OpenLEGO to execute the optimization in various PIDO programs (Optimus, OpenMDAO, RCE). The MPG for a simplied Sellar problem is given in Figure 42

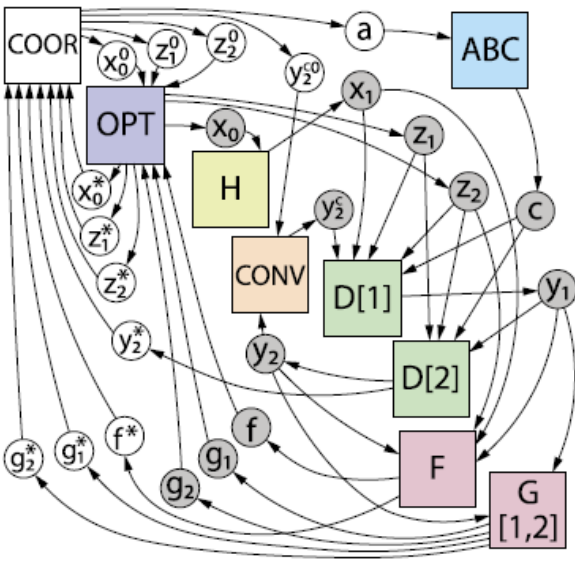


Figure 41: MDAO Data Graph for the Sellar problem

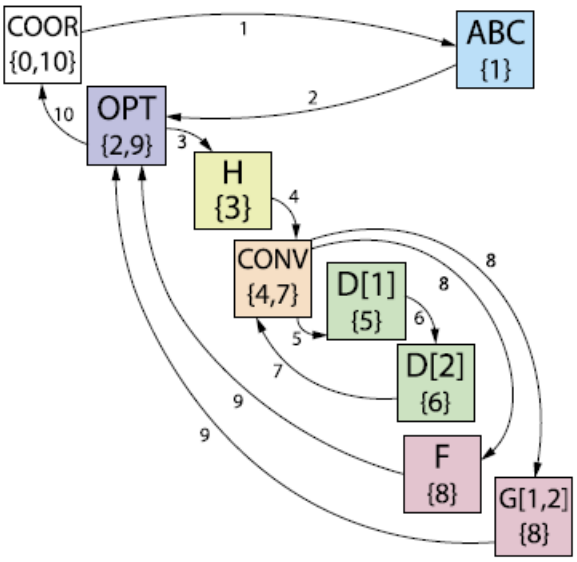


Figure 42: MDAO Process Graph for the Sellar problem

5.3 MDAx

MDAx is a similar framework as KADMOS, but developed by the the German Aerospace Centre (DLR). MDAx allows for more effective collaboration between various heterogeneous design teams, by providing oversight and increased transparency. Current limitations of for instance KADMOS as compared to MDAx are the lack of process inflexibility's, difficult customization of workflows and the lack of user-friendliness [54]. Here, process inflexibility refers to the fact that the process in which these MDAO workflows are set-up is not robust and can not be altered. Other changes which have been implemented into MDAx as compared to KADMOS are:

- MDAx has a distinct, well visualized user interface which was lacking with KADMOS. Over time though this was added to KADMOS through an external user interface named VISTOMS [5].
- KADMOS is written in an elaborate graph based syntax, which for an everyday engineer with no further knowledge on the software, can be difficult to understand. MDAx is build more intuitively.
- KADMOS does not allow for flexible workflow modelling processes through its strict methodologies. More flexibility is added by MDAx.
- KADMOS relies on very significant assumptions, which consequently result in limiting capabilities. In KADMOS, self-loops are simulation tools without I/O are not feasible, which are possible in MDAx.
- Lastly, KADMOS is from a software perspective, not well written as an Object-Oriented Programming software, due to a lack of unit testing. This can result in errors when trying to implement extensions or modifications to the software.

Since the MDAx software is not open-source or commercially available, and lacks a proper scripted interface with Python, KADMOS is the only valid option for dynamic MDAO workflow generation in this thesis.

5.4 Key Takeaways

1. By utilizing software which can efficiently create executable MDAO workflows for a given design problem with a specified MDAO architecture, the implementation gap of MDAO problems can be decreased, thereby increasing the productivity of heterogeneous MDAO design teams.
2. It is estimated that the use of KADMOS to automate the dynamic generation and alteration of executable MDAO workflows, results in a 50% decrease in set-up time of such MDAO workflow. It furthermore provides more overview of the process thereby increasing the chances of successful and industry wide implementation of MDAO even further.
3. MDAx is not applicable to this master thesis, therefore only KADMOS will be used. KADMOS furthermore allows for proper interface with existing Python modules due to it's scripted set-up.

[5] https://www.agile-project.eu/files/VISTOMS_SellarProblem/, accessedon11/1/2023

6 Research Methodology

This chapter aims to outline the next steps of the thesis, by explaining the relevance, research gap and objective, methodology and project planning. Achieving the research objective will be through answering of the various research questions. First the relevance of the research project will be explained, thereby highlighting the knowledge or research gap. By knowing the research gap, the research objective can then be formulated, which is followed by the various research questions which will contribute to obtaining the research objective. Next, a research plan in the form of a flow chart will be showcased, providing a clear guideline on how to achieve the research objective. Lastly a Ggant chart will highlight the project timeline including relevant milestones.

6.1 Relevance of Research Project

Man kind is required to come up with new, more sustainable ways of transportation as a consequence of the pressing climate crisis. These more sustainable ways of transportation should replace the conventional fossil fuels as an energy carrier in mobility, by more advanced, electric drive trains utilizing clean energy from wind or solar to transport people between places. Electric drive-trains have a significantly higher well-to-wheel efficiency, combined with the fact that electrical energy can be produced green, making them an ideal solution for future modes of transportation. Such innovative ways of transportation are possible due to a combination of complex drive systems, which require a clear understanding of all components and their effect on the overall system's performance to assess and optimize such complex systems. However, as electric drive train components suffer from a high level of dimensionality in conceptual design, conventional design methods prove to be insufficient. This is where the implementation of MDAO becomes interesting. MDAO uses computer-aided optimization techniques to efficiently explore the design space and come up with a feasible and optimal design.

In conceptual design however, a lot of information is still unknown, as can be explained by the Knowledge Paradox first presented by La Rocca et al. [9]). Implementing MDAO in conceptual design will output deterministic solutions which might not be true to reality. Therefore, a more sophisticated method, which takes into account input- and model-uncertainty in the multidisciplinary design and optimization of electric drive trains is required, but lacking from literature. An explanation for this could be that it is difficult to model these input- and model-uncertainty at such early stages, as well as the mixed-integer design space a vehicle drive train design. By assuming, based on experimental data, prior- or engineering knowledge, these uncertainty factors, and calculating their effect on the objective functions, constraints or qualities of interest, a more robust and reliable optimal system architecture can be found. This can then lead to a lower risk of not satisfying requirements in conceptual design, which in turn results in faster design cycles.

The knowledge gap of this thesis can therefore be explained as:

To date, a combined automatic vehicle drive-train architectural exploration and uncertainty-based design optimization framework is missing due to a lack of integration methods which take into account the mixed-integer discontinuous hierarchical design space of vehicle system architectures and uncertainty-based optimization.

This project aims to implement uncertainty-based MDAO into the system architecture exploration and optimization phase in conceptual design. It does this by building a software tool which allows the user to specify a number of useful components and their models, as well as a set of constraints, and from that generate a feasible design for which the system engineers can be with, for instance, 95% certain that it will adhere to the requirements specified while still being a robust optima.

6.2 Research Objective

The research objective is stated as:

How can uncertainty-based optimization be integrated into a dynamic MDAO workflow for exploring and optimizing vehicle system architectures to ultimately improve the requirement risk management for innovative conceptual design studies of complex engineering systems?

Which will be achieved by:

Integrating an automated architectural design space exploration, evaluation and optimization framework into a software package, which based on a set of known surrogate component models, will perform Reliability-Based Robust Design Optimization (RBRDO), to find the most optimal design given a set of qualitative and quantitative requirements.

6.3 Research Questions

The top level research question is defined as:

How can uncertainty-based optimization be integrated into a dynamic MDAO workflow for exploring and optimizing vehicle system architectures to ultimately improve the requirement risk management for innovative conceptual design studies of complex engineering systems?

This top level question can be broken down into a number of sub-questions that are detailed below

1. How can a set of consistent system architectures be generated from a set of known multi-domain components and their I/Os?
2. What filtering methods are most effective when applied to a set of feasible system architectures to restrict the design space?
3. How can the mixed-integer design space most efficiently be explored to find an optima, where especially the integer design variable space is highly discontinuous, without the use of Design of Experiment?
4. How can the consideration of uncertainty in optimization lead to more comprehensive exploration of the design space and hence a more optimal design looking at requirement satisfaction, in comparison to deterministic optimization.
5. How can uncertainty be quantified and integrated into the various stages of a dynamic MDAO workflow for vehicle system architecture optimization?
6. What are the most suitable optimization algorithms and techniques for an uncertainty-based MDO problem formulation in the context of vehicle system architectural design?
7. What are the effects on the optimal design solution by taking into account these model- and variable uncertainties?
   (a) What is the difference in performance output (objective) and optimal design vector between a deterministic framework and a stochastic framework?
   (b) What are the differences in optimal system architecture between a robust and reliable design optima? What aspects have the highest sensitivity to the change in uncertainty-based optimization framework (robust vs reliable)?
8. How does the inclusion of uncertainty quantification impact the overall computational cost and time required for vehicle system architecture design optimization?
9. Can the implementation of uncertainty-aware optimization in MDAO lead to improvements in risk mitigation and requirement management in vehicle design projects?
   (a) What metrics and criteria can be used to evaluate the effectiveness of integrating uncertainty-aware optimization into the MDAO workflow?
10. What are the follow-up steps to further decrease the uncertainty at conceptual design stages of design, after optimization under uncertainty is performed and a reliable and robust design optimum was found?
11. Would it be possible to allocate the highest level of uncertainty within a robust and reliable optimal architectural design and find methods to resolve this uncertainty for that specific design choice.

6.4 Research Strategy

A schematic of the proposed research strategy is shown in Appendix A. Here, the project is split up into three distinct parts, each being executed separately.

- **Part 1**: Part 1 consists of constructing an automated architectural design space exploration tool provided the user specified a set of feasible components. From this a graph constraint satisfactory problem is formulated which takes into account the multi-domain nature of electric vehicle system architecture optimization. Next, multiple filtering steps are applied, based on user preference or isomorphism to come up with a much smaller set of feasible and desired system architectures. Here, also the modelling toolboxes which are used to model component and/or system performance are properly implemented to allow for optimization. Since optimization requires large amounts of iterative steps, a computationally inexpensive solution for all modelling toolboxes is desired.
- **Part 2**: Part 2 will focus on building the optimization framework for the continuous design variables derived from the set of used components within a specified solution of the set of feasible integer system architectures. Here, KADMOS will be used to dynamically build a correct executable optimization workflow. Since KADMOS is already integrated through OpenLEGO with OpenMDAO, no extensive research and effort has to be put into making this connection.
- **Part 3**: Part 3 will be on implementing a first-order pertubation method to compute the variation in objective function and constraint due to input- and model-uncertainty. For this, uncertainty is assumed to be normally distributed or if experimental data is known, numerically computed through statistical analysis.

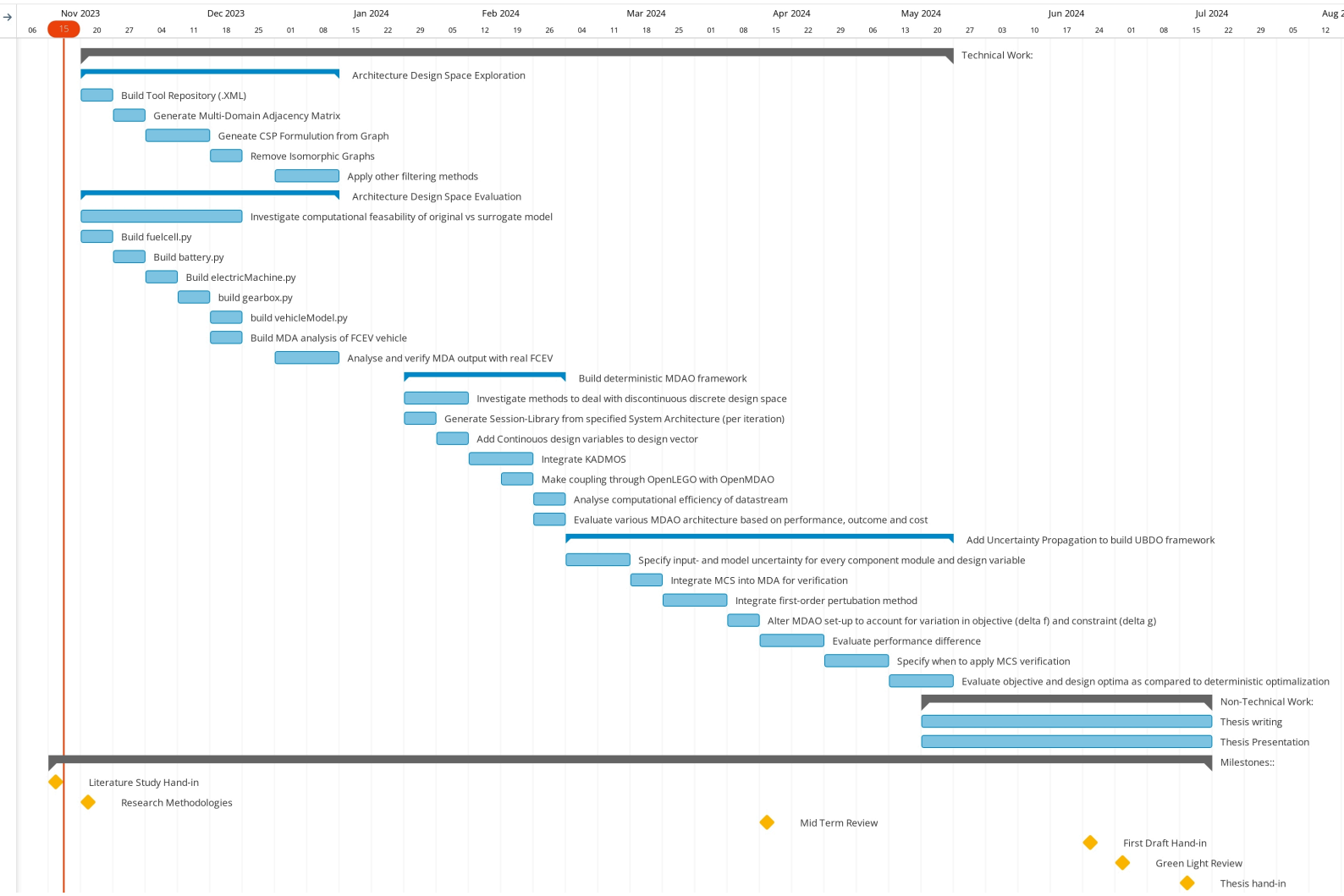
These various steps are further explained in detail in Appendix A. What is less visible from Appendix A, is that before optimization can be performed, first model building and analysis will be performed to see if the desired outcome is generated (i.e. model verification). Therefore, the project will actually be split up into the development, analysis and lastly the optimization phase.

6.5 Gantt Chart

A Gantt chart detailing the project outline is given at the end of this section.

GanttChart_Thesis

Read-only view, generated on 15 Nov 2023



7 Conclusion

For the past decades, human's have travelled the world by means of converting chemical energy into propulsive power through the combustion of hydrocarbons. Such fossil fuels, produce a lot of power through combustion, but as a downside emit a lot of greenhouse gasses such as carbon dioxide (CO_2), nitrous oxide (NOx), hydrofluorocarbons (HFCs) and ozone, which have a negative impact on the greenhouse effect. The mobility sector is responsible for a significant portion of these emissions, which forced large regulatory organizations to impose restrictions on the allowable emissions produced per sector. In an era marked by these increasing concerns over climate change, the need has emerged to reassess our modes of transportation. Pivotal to assessing these new modes of transportation are the OEM's, which are researching, developing and producing these new, low-emission, regulatory compliant mobility solutions. For the past decade, the development of electric or low-emission solutions has changed from a trend in research to an essential trajectory towards a future where vehicle mobility is not at the expense of ecological preservation.

An emerging trend in the engineering and optimization of innovative complex systems is the implementation of Multidisciplinary Design Analysis and Optimization (MDAO). It is estimated that MDAO can offer significant benefits in the design of System of Systems (SoS), estimated to range from 8-10% for innovative derivative design and even larger (40-50%) for the design of radically new concepts [21]. Despite these expected gains, MDAO is not as widely used in the industry as one would anticipate. This hampering is a result of both technical and non-technical barriers present in industry at the moment. The gains, which become especially clear when looking at the number of design iterations in a specified time frame, are visualized in Figure 43, where a comparison is made with Boeing's conventional Legacy design method based on sequential design.

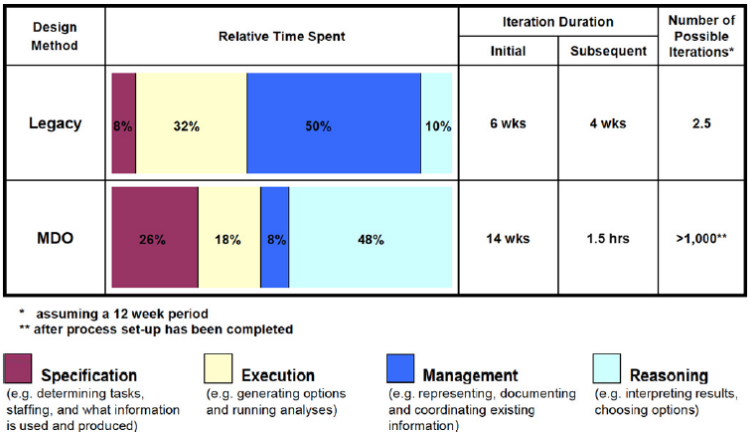


Figure 43: Comparison between the Legacy design method (Boeing) and MDO design methods[20]

Especially the integration of MDAO in early stages of design are expected to bring significant performance and cost gains as a result of the increased knowledge on the system behavior, its components and their synergy. By increasing the system specific knowledge early on, where the cost of change is relatively low, the success-rate of these innovative projects can be increased, looking at time, budget and requirement adherence.

This literature review will focus on presenting ways to perform system architecture optimization in early stages of design (conceptual) by utilizing methods which work well with such mixed-integer hierarchical design space and integrating uncertainty quantification and propagation to account for the lack of knowledge. An important methodology presented from literature which describes this process is Architectural Design Space Exploration, Evaluation and Optimization (ADSE). ADSE is a method used for finding various options of a system architectures based on a predefined set of components and a knowledge base explaining their interactions. Numerous methods of manually finding various system architectures have been presented in literature, ranging from Morphological Matrices to Functional Decomposition. A more computationally efficient way of finding all possible feasible system architectures in the ADSE phase, is by means of automated topology generation through constraint programming and graph theory. Here a set of components (nodes) are connected through edges. By assigning values to these nodes, and formulating constraints on (in)feasible graph formulations, a set of feasible system architectures can

be found. This set of feasible system architectures can then be evaluated in the Architectural Design Space Evaluation phase. In order to implement more advanced, mid- to high-fidelity models in the optimization of system architectures, utilizing surrogate models to find the performance characteristics of every system architecture within the feasible set is used. The above mentioned constraint satisfactory problem using graph theory can be further extended to multi-domain discrete system architecture optimization, as not only mechanical components and connections are feasible, but also electrical, hydraulic and signal connections are feasible. This way the knowledge on the system's overall performance is increased.

A downside however of integrating MDAO in early stages of the design process, is that still a lot of information is unknown about the various systems and their interaction. MDAO makes use of the fact that all disciplines and design variables are deterministic, which assumes that all variables are 100% accurate and known. This is of course not true. To solve this problem of uncertainty and finding an optimal solution under model and input uncertainty, a special type of MDAO is investigated in this literature review, namely Uncertainty-Based Design Optimization (UBDO). A special type of UBDO is Reliability-Based Robust Design Optimization. This type of MDAO problem formulation relies on combining methods to find both Reliable and Robust design optima. Reliability-Based Design Optimization (RBDO) is a method which tries to find a local optima which in case of model or input uncertainty, does not result in an infeasible design through constraint violation. Hence the method mostly is geared towards the MDAO constraints and bounds specified. Robust Design Optimization (RDO) is a method which tries to find a local optima which is less/insensitive to model or input uncertainty, and hence does not result in a significant loss in objective function performance. Hence, it is more focused on minimizing a weighted sum of the mean and variation in objective function.

Since the ADSE phase consists of evaluating and optimizing various system architectures with different modelling tools (disciplines) and continuous design variables, a dynamic MDAO problem formulation framework needs to be implemented which between iterations of discrete system architectures, formulates a coherent executable MDAO workflow which can be executed in various PIDO platforms such as OpenMDAO. For this, a solution based on KADMOS was chosen. KADMOS is an agile dynamic workflow framework which can very easily segment, sequence and build executable workflows without human interference, given a specified knowledge base.

In literature, no previous work was found which focused on combining automated system architecture generation through constraint satisfaction programming with the use of uncertainty-based multidisciplinary design optimization. The above mentioned examples however showed that by taking into account uncertainty (model and input) in the architectural design space exploration and optimization phase of conceptual design, can result in significantly more confidence in a found optimal system architectural design solution.

A thesis project plan, including research relevance, objective, questions and project planning is presented in section 6. This research stems from VDL's need to increase the system specific knowledge in early stages of design without limiting the design space using engineering bias or safety factors. The goal of this research is therefore to build a tool which produces an optimal system architecture while taking into account model- and input uncertainty due to a lack of knowledge in conceptual design. This way, the system engineers can be more confident that the determined design will adhere to the specified customer requirements, all while still being an optimal system architecture.

The research objective for this thesis work therefore is:

How can uncertainty-based optimization be integrated into a dynamic MDAO workflow for exploring and optimizing vehicle system architectures to ultimately improve the requirement risk management for innovative conceptual design studies of complex engineering systems?

Which will be achieved by:

Integrating an automated architectural design space exploration, evaluation and optimization framework into a software package, which based on a set of known surrogate component models, will perform Reliability-Based Robust Design Optimization (RBRDO), to find the most optimal design given a set of qualitative and quantitative requirements.

Other interesting research sub-questions are also presented, which will provide more in depth knowledge on the research topic. Next, the research plan in presented through a simplified flowchart, which divides the research plan into three sections; constraint satisfactory programming through graph theory, MDAO problem formulation & uncertainty quantification and propagation. Lastly, also a project planning is presented in subsection 6.5.

References

[1] Nicolas Albarello, Jean-Baptiste Welcomme, and Claude Reyterou. "A formal design synthesis and optimization method for systems architectures". In: (June 2012).

[2] Kristian Amadori, Erik Bäckström, and Christopher Jouannet. "Future Technologies Prioritization for Aircraft Conceptual Design". In: *2018 AIAA Aerospace Sciences Meeting*. DOI: 10.2514/6.2018-1746. eprint: https://arc.aiaa.org/doi/pdf/10.2514/6.2018-1746. URL: https://arc.aiaa.org/doi/abs/10.2514/6.2018-1746.

[3] Ali-Reza Babaei, Mohammad Setayandeh, and Hamid Farrokhfal. "Aircraft robust multidisciplinary design optimization methodology based on fuzzy preference function". In: *Chinese Journal of Aeronautics* 31 (May 2018). DOI: 10.1016/j.cja.2018.04.018.

[4] S.M. Batill, John Renaud, and Xiaoyu Gu. "Modeling And Simulation Uncertainty In Multidisciplinary Design Optimization". In: 4803 (Nov. 2000). DOI: 10.2514/6.2000-4803.

[5] T. F. Beernaert and L. F. P. Etman. "Multi-level Decomposed Systems Design: Converting a Requirement Specification into an Optimization Problem". In: *Proceedings of the Design Society: International Conference on Engineering Design* 1.1 (2019), pp. 3691–3700. DOI: 10.1017/dsi.2019.376.

[6] Mohamed Amine Bouhlel et al. "A Python surrogate modeling framework with derivatives". In: *Advances in Engineering Software* (2019), p. 102662. ISSN: 0965-9978. DOI: https://doi.org/10.1016/j.advengsoft.2019.03.005.

[7] A.M.R.M Bruggeman et al. "An MBSE-Based Requirement Verification Framework to support the MDAO process". In: DOI: 10.2514/6.2022-3722. URL: https://repository.tudelft.nl/islandora/object/uuid:fe5d5252-ecc0-4e63-875d-549a4a997192?collection=research.

[8] Timothy A Burress et al. *Evaluation of the 2010 Toyota Prius hybrid synergy drive system*. Tech. rep. Oak Ridge National Lab.(ORNL), Oak Ridge, TN (United States). Power . . ., 2011.

[9] Jasper Bussemaker and Pier Davide Ciampa. "MBSE in Architecture Design Space Exploration". In: Apr. 2022. DOI: 10.1007/978-3-030-27486-3_36-1.

[10] Jasper Bussemaker, Pier Davide Ciampa, and Björn Nagel. "System Architecture Design Space Exploration: An Approach to Modeling and Optimization". In: June 2020. DOI: 10.2514/6.2020-3172.

[11] Jasper H Bussemaker et al. "System architecture optimization: An open source multidisciplinary aircraft jet engine architecting problem". In: *AIAA Aviation 2021 Forum*. 2021, p. 3078.

[12] Hongjun Cao and Baoyan Duan. "Uncertainty analysis for multidisciplinary systems based on convex models". In: *10th AIAA/ISSMO multidisciplinary analysis and optimization conference*. 2004, p. 4504.

[13] Tom Clancy. "The chaos report". In: *The Standish Group* (1995).

[14] Edward F. Crawley, Bruce G. Cameron, and Daniel Selva. "System Architecture: Strategy and Product Development for Complex Systems". In: 2015. URL: https://api.semanticscholar.org/CorpusID:113252225.

[15] Rik W De Doncker, Duco WJ Pulle, and André Veltman. *Advanced electrical drives: analysis, modeling, control*. Springer Nature, 2020.

[16] Thibault De Smedt. "Aircraft Jet Engine Architecture Modeling: Creating a Benchmark Problem Using a System Architecting Approach with Mixed-Discrete Multi-Objective Capabilities". In: *Master's Thesis*. Delft University of Technology. 2021.

[17] K. Deb et al. "A fast and elitist multiobjective genetic algorithm: NSGA-II". In: *IEEE Transactions on Evolutionary Computation* 6.2 (2002), pp. 182–197. DOI: 10.1109/4235.996017.

[18] Xiaoping Du and Wei Chen. "An efficient approach to probabilistic uncertainty analysis in simulation-based multidisciplinary design". In: *38th Aerospace Sciences Meeting and Exhibit*. 2000, p. 423.

[19] R. Eberhart and J. Kennedy. "A new optimizer using particle swarm theory". In: *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*. 1995, pp. 39–43. DOI: 10.1109/MHS.1995.494215.

[20] Forest Flager and John Haymaker. "A comparison of multidisciplinary design, analysis and optimization processes in the building construction and aerospace industries". In: *24th international conference on information technology in construction*. Maribor Slovenia. 2007, pp. 625–630.

[21] Imco van Gent and Gianfranco La Rocca. "Formulation and integration of MDAO systems for collaborative design: A graph-based methodological approach". In: *Aerospace Science and Technology* 90 (2019), pp. 410–433.

[22] Imco van Gent, Gianfranco La Rocca, and Maurice Hoogreef. "CMDOWS: A Proposed New Standard to Store and Exchange MDO Systems". In: Oct. 2017.

[23] Justin S. Gray et al. "OpenMDAO: An Open-Source Framework for Multidisciplinary Design, Analysis, and Optimization". In: *Structural and Multidisciplinary Optimization* 59 (4 2019), pp. 1075–1104. DOI: 10.1007/s00158-019-02211-z.

[24] X. Gu et al. "Worst case propagated uncertainty of multidisciplinary systems in robust design optimization". In: *Structural and Multidisciplinary Optimization* 20 (Jan. 2000), pp. 190–213. DOI: 10.1007/s001580050148.

[25] Marin Guenov et al. "Aircadia -an interactive tool for the composition and exploration of aircraft computational studies at early design stage". In: (Jan. 2014).

[26] Marin Guenov et al. "Aircraft Systems Architecting: Logical-Computational Domains Interface". In: Sept. 2018.

[27] Lino Guzzella and Antonio Sciarretta. *Vehicle Propulsion Systems: Introduction to Modeling and Optimization*. Jan. 2007. ISBN: 978-3-642-35912-5. DOI: 10.1007/978-3-642-35913-2.

[28] Sandra Hamze. "Model-based System Engineering for Powertrain Systems Optimization". PhD thesis. Nov. 2019.

[29] Jon Holt and S. Perry. *SysML for Systems Engineering, 2nd Edition: A Model-Based Approach*. Nov. 2013, pp. 1–935. ISBN: 9781849196512. DOI: 10.1049/PBPC010E.

[30] L Jaeger et al. "Aircraft multidisciplinary design optimization under both model and design variables uncertainty". In: *Journal of Aircraft* 50.2 (2013), pp. 528–538.

[31] Donald R Jones, Matthias Schonlau, and William J Welch. "Efficient global optimization of expensive black-box functions". In: *Journal of Global optimization* 13 (1998), pp. 455–492.

[32] David Judt and Craig Lawson. "Development of an automated aircraft subsystem architecture generation and analysis tool". In: *Engineering Computations* 33 (July 2016), pp. 1327–1352. DOI: 10.1108/EC-02-2014-0033.

[33] Bilal Kabalan et al. "Systematic Methodology for Architecture Generation and Design Optimization of Hybrid Powertrains". In: *IEEE Transactions on Vehicular Technology* PP (Dec. 2020), pp. 1–1. DOI: 10.1109/TVT.2020.3041501.

[34] Gaetan KW Kenway and Joaquim RRA Martins. "Multipoint high-fidelity aerostructural optimization of a transport aircraft configuration". In: *Journal of Aircraft* 51.1 (2014), pp. 144–160.

[35] Hyung Min Kim et al. "Target Cascading in Optimal System Design ". In: *Journal of Mechanical Design* 125.3 (Sept. 2003), pp. 474–480. ISSN: 1050-0472. DOI: 10.1115/1.1582501. eprint: https://asmedigitalcollection.asme.org/mechanicaldesign/article-pdf/125/3/474/5799301/474_1.pdf. URL: https://doi.org/10.1115/1.1582501.

[36] Thomas Klimmek et al. "cpacs-MONA–An independent and in high fidelity based MDO tasks integrated process for the structural and aeroelastic design for aircraft configurations". In: *International Forum on Aeroelasticity and Structural Dynamics 2019, IFASD 2019*. 2019.

[37] Aart-Jan Kort et al. "Automated Multi-Level Dynamic System Topology Design Synthesis". In: *Vehicles* 2.4 (2020), pp. 603–624.

[38] Michael Krueger et al. *Systems Engineering Guidebook for Intelligent Transportation Systems*. Nov. 2009.

[39] Gianfranco La Rocca and Michael Van tooren. "Knowledge-based engineering to support aircraft multidisciplinary design and optimization". In: *Proceedings of The Institution of Mechanical Engineers Part G-journal of Aerospace Engineering - PROC INST MECH ENG G-J A E* 224 (Sept. 2010), pp. 1041–1055. DOI: 10.1243/09544100JAERO592.

[40] Andrew B Lambe and Joaquim RRA Martins. "Extensions to the design structure matrix for the description of multidisciplinary design, analysis, and optimization processes". In: *Structural and Multidisciplinary Optimization* 46 (2012), pp. 273–284.

[41] Thierry Lefebvre et al. "Aircraft conceptual design in a multi-level, multi-fidelity, multi-disciplinary optimization process". In: *Proceedings of the 28th International Congress of The Aeronautical Sciences, Brisbane, Australia*. 2012, pp. 23–28.

[42] Fang Lincun et al. "Simultaneous Optimization for Hybrid Electric Vehicle Parameters Based on Multi-Objective Genetic Algorithms". In: *Energies* 4 (Dec. 2011). DOI: 10.3390/en4030532.

[43] Zhoujie Lyu, Zelu Xu, and Joaquim R.R.A. Martins. "Benchmarking Optimization Algorithms for Wing Aerodynamic Design Optimization". In: URL: https://api.semanticscholar.org/CorpusID:16276466.

[44] Joaquim Martins and Andrew Lambe. "Multidisciplinary Design Optimization: A Survey of Architectures". In: *AIAA Journal* 51 (Sept. 2013), pp. 2049–2075. DOI: 10.2514/1.J051895.

[45] Joaquim R. R. A. Martins and Andrew Ning. *Engineering Design Optimization*. Cambridge University Press, 2021. DOI: 10.1017/9781108980647.

[46] Steven Masfaraud et al. "Automatized gearbox architecture design exploration by exhaustive graph generation". In: July 2016.

[47] Peter Matthews. "Challenges to Bayesian decision support using morphological matrices for design: Empirical evidence". In: *Research in Engineering Design* 22 (Jan. 2010), pp. 29–42. DOI: 10.1007/s00163-010-0094-1.

[48] Daniel Neufeld, Joon Chung, and Kamaran Behdinan. "Development of a flexible MDO architecture for aircraft conceptual design". In: *Proceedings of the 2008 EngOpt conference. Rio de Jenario, Brazil*. Citeseer. 2008.

[49] NV Nguyen et al. "Possibility-based multidisciplinary optimisation for electric-powered unmanned aerial vehicle design". In: *The Aeronautical Journal* 119.1221 (2015), pp. 1397–1414.

[50] Christiaan Paredis et al. "An Overview of the SysML-Modelica Transformation Specification". In: vol. 20. July 2010. DOI: 10.1002/j.2334-5837.2010.tb01099.x.

[51] Sung H Park. "Robust design and analysis for quality engineering". In: *(No Title)* (1996).

[52] Ruben Perez, Hugh Liu, and Kamran Behdinan. "Evaluation of Multidisciplinary Optimization Approaches for Aircraft Conceptual Design". In: *Collection of Technical Papers - 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference* 4 (Aug. 2004). DOI: 10.2514/6.2004-4537.

[53] Mailman School of Public Health. "Kriging Interpolation Explanation". In: (). URL: https://www.publichealth.columbia.edu/research/population-health-methods/kriging-interpolation#.

[54] Andreas Risueño et al. "MDAx: Agile Generation of Collaborative MDAO Workflows for Complex Systems". In: June 2020. DOI: 10.2514/6.2020-3133.

[55] Martijn Roelofs and Roelof Vos. "Uncertainty-Based Design Optimization and Technology Evaluation: A Review". In: Jan. 2018. DOI: 10.2514/6.2018-2029.

[56] Satadru Roy et al. "A mixed integer efficient global optimization algorithm for the simultaneous aircraft allocation-mission-design problem". In: *58th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*. 2017, p. 1305.

[57] Satadru Roy et al. "A mixed integer efficient global optimization algorithm with multiple infill strategy-applied to a wing topology optimization problem". In: *AIAA Scitech 2019 Forum*. 2019, p. 2356.

[58] Jerome Sacks, Susannah B. Schiller, and William J. Welch. "Designs for Computer Experiments". In: *Technometrics* 31.1 (1989), pp. 41–47. ISSN: 00401706. URL: http://www.jstor.org/stable/1270363 (visited on 10/12/2023).

[59] Akeel A Shah et al. "Recent trends and developments in polymer electrolyte membrane fuel cell modelling". In: *Electrochimica Acta* 56.11 (2011), pp. 3731–3757.

[60] Donald Shepard. "A Two-Dimensional Interpolation Function for Irregularly-Spaced Data". In: *Proceedings of the 1968 23rd ACM National Conference*. ACM '68. New York, NY, USA: Association for Computing Machinery, 1968, pp. 517–524. ISBN: 9781450374866. DOI: 10.1145/800186.810616. URL: https://doi.org/10.1145/800186.810616.

[61] E. Silvas et al. "Functional and cost-based automatic generator for hybrid vehicles topologies". English. In: *IEEE/ASME Transactions on Mechatronics* 20.4 (Aug. 2015), pp. 1561–1572. ISSN: 1083-4435. DOI: 10.1109/TMECH.2015.2405473.

[62] Willard Simmons. "A framework for decision support in systems architecting". In: (Nov. 2008).

[63] Valentin Sulzer et al. "Python battery mathematical modelling (PyBaMM)". In: *Journal of Open Research Software* 9.1 (2021).

[64] Genichi Taguchi, Elsayed A Elsayed, and Thomas C Hsiang. *Quality engineering in production systems*. McGraw-Hill Companies, 1988.

[65] Daniel Thunnissen. "Propagating and mitigating uncertainty in the design of complex multidisciplinary systems". In: (Dec. 2005).

[66] Daniel P. Thunnissen. "Uncertainty Classification for the Design and Development of Complex Systems". In: 2003. URL: https://api.semanticscholar.org/CorpusID:15676167.

[67] Thomas Tranter et al. "liionpack: A Python package for simulating packs of batteries with PyBaMM". In: *Journal of Open Source Software* 7.70 (2022).

[68] Olivier Tremblay, Louis-A Dessaint, et al. "A generic fuel cell model for the simulation of fuel cell vehicles". In: *2009 IEEE vehicle power and propulsion conference*. IEEE. 2009, pp. 1722–1729.

[69] Santiago Valencia Ibanez. "Optimization Strategies for System Architecting Problems". In: *Master's Thesis*. Delft University of Technology. 2023.

[70] Nhu Van Nguyen et al. "A multidisciplinary robust optimisation framework for UAV conceptual design". In: *The Aeronautical Journal* 118.1200 (2014), pp. 123–142.

[71] Daniël de Vries. "Towards the Industrialization of MDAO: Evolving 1st Generation MDAO to an Industry Ready Level of Maturity". In: (2017).

[72] Richard G. Weber, Sridhar, and Condoor. "Conceptual design using a synergistically compatible morphological matrix". In: *FIE '98. 28th Annual Frontiers in Education Conference. Moving from 'Teacher-Centered' to 'Learner-Centered' Education. Conference Proceedings (Cat. No.98CH36214)* 1 (1998), 171–176 vol.1. URL: https://api.semanticscholar.org/CorpusID:13572925.

[73] Robert B Wilson. "A simplicial algorithm for concave programming". In: *Ph. D. Dissertation, Graduate School of Bussiness Administration* (1963).

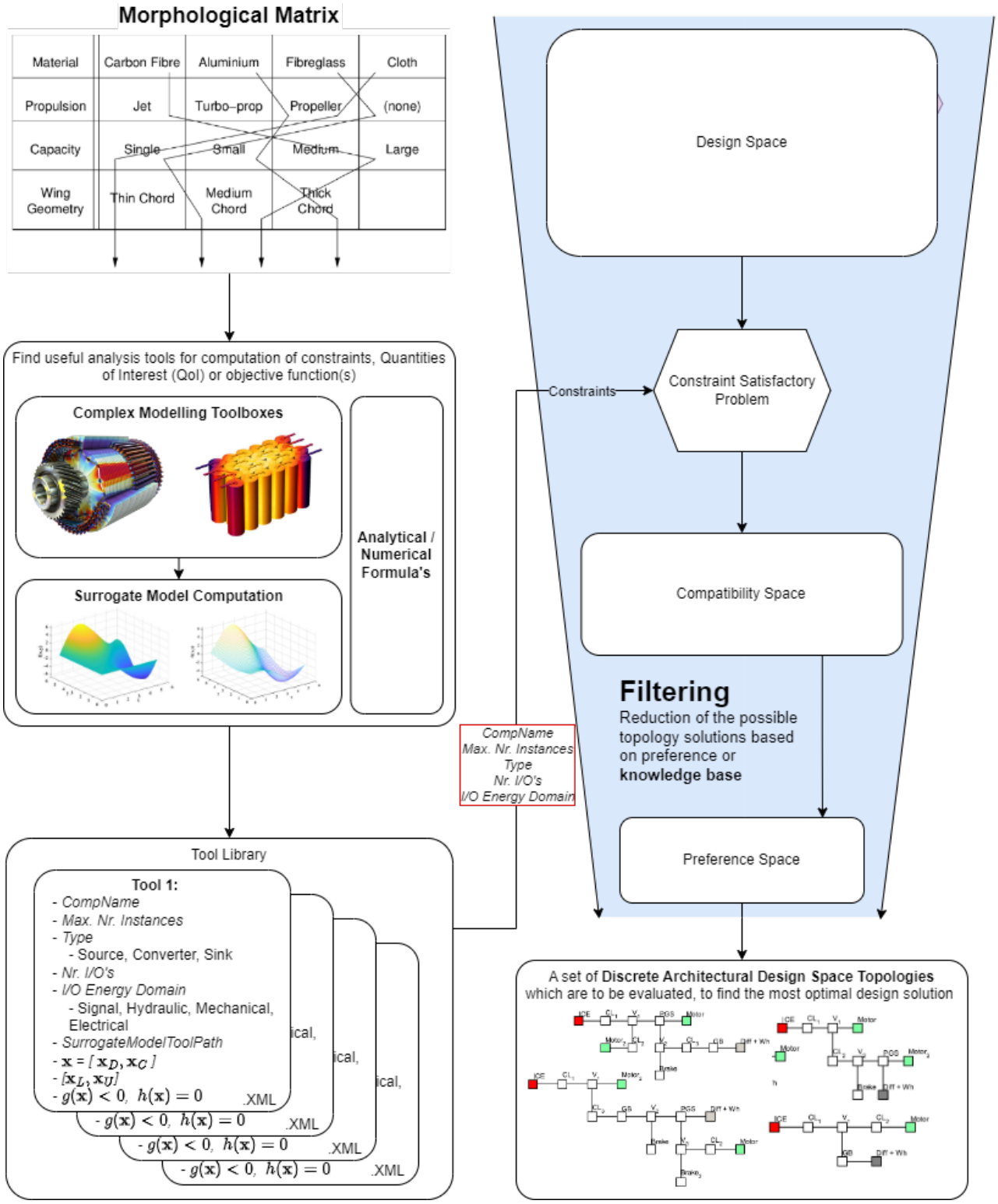
[74] Xiaoxiong Wu et al. "Surrogate Models for Performance Prediction of Axial Compressors Using through-Flow Approach". In: *Energies* 13.1 (2020). ISSN: 1996-1073. DOI: 10.3390/en13010169. URL: https://www.mdpi.com/1996-1073/13/1/169.

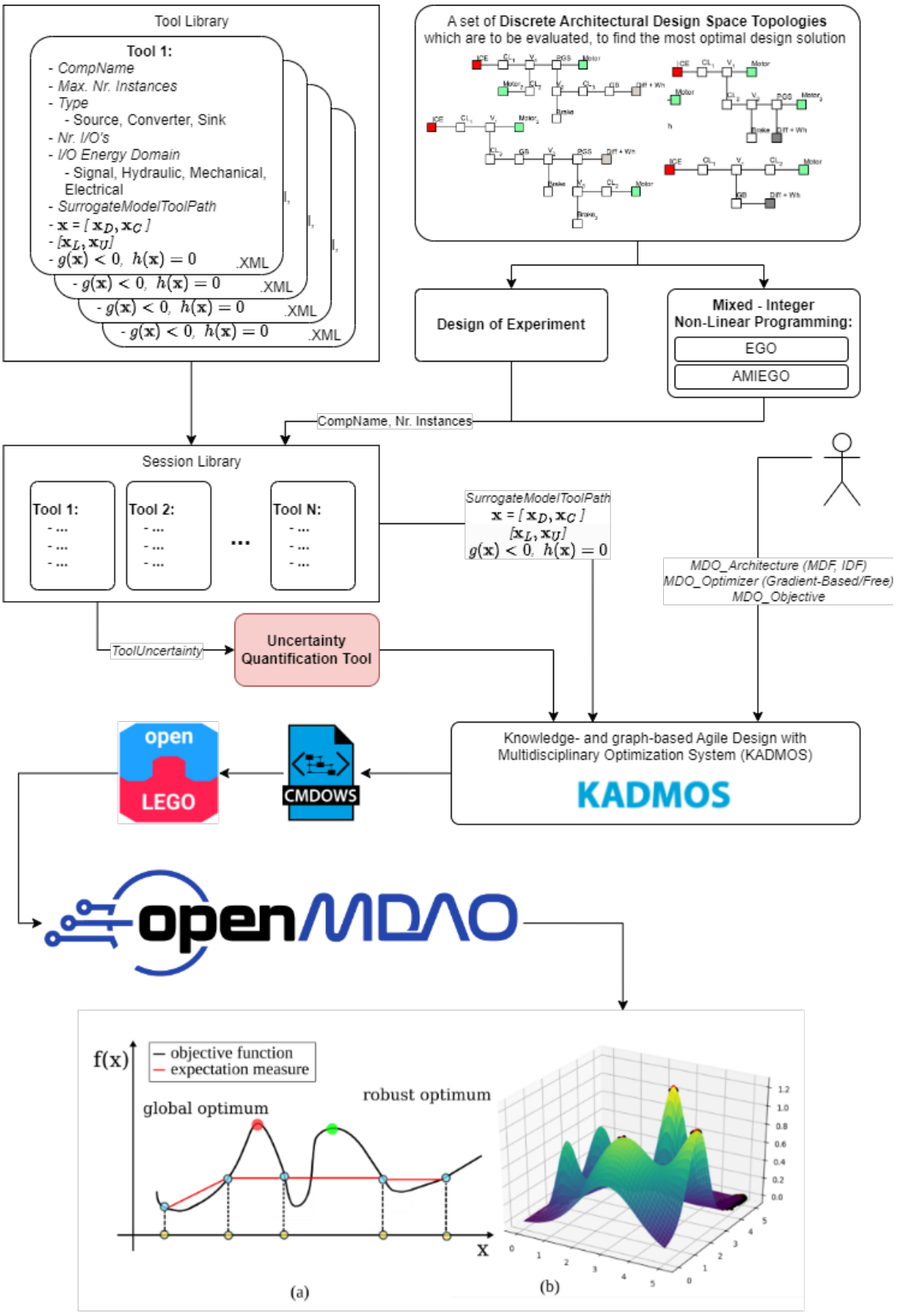
[75] Wen Yao et al. "Review of uncertainty-based multidisciplinary design optimization methods for aerospace vehicles". In: *Progress in Aerospace Sciences* 47 (Aug. 2011), pp. 450–479. DOI: 10.1016/j.paerosci.2011.05.001.

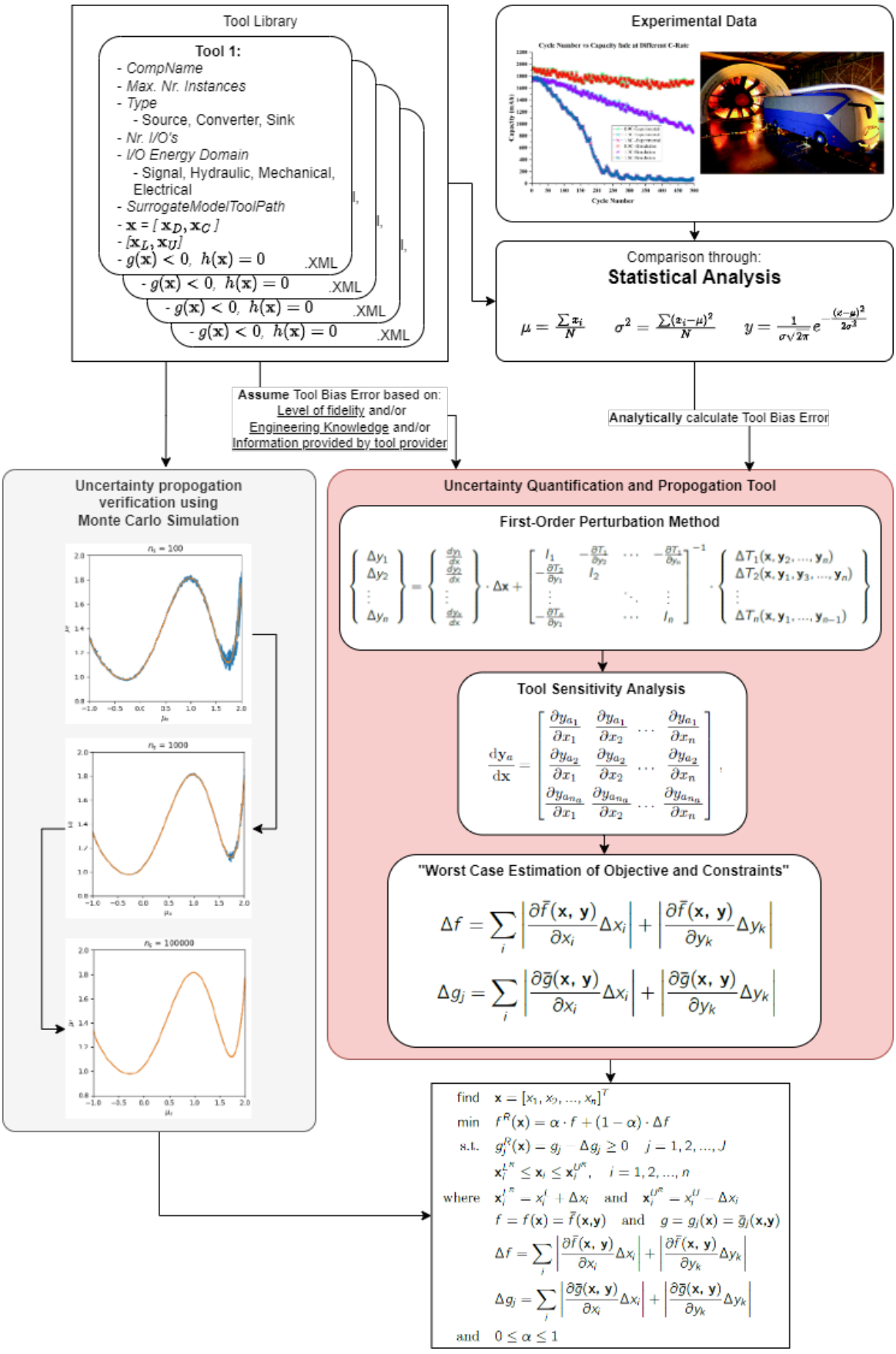
[76] Cheng Zhang et al. "Battery modelling methods for electric vehicles-A review". In: *2014 European Control Conference (ECC)*. IEEE. 2014, pp. 2673–2678.

[77] Cheng Zhang et al. "Online estimation of battery equivalent circuit model parameters and state of charge using decoupled least squares technique". In: *Energy* 142 (Oct. 2017). DOI: 10.1016/j.energy.2017.10.043.

A Appendix A


Morphological Matrix
Material
Carbon Fibre
Aluminium
Fibreglass
Cloth
Propulsion
Jet
Turbo–prop
Propeller
(none)
Capacity
Single
Small
Medium
Large
Wing Geometry
Thin Chord
Medium Chord
Thick Chord
Find useful analysis tools for computation of constraints, Quantities of Interest (QoI) or objective function(s)
Complex Modelling Toolboxes
Surrogate Model Computation
Analytical / Numerical Formula's
Tool Library
Tool 1:
- CompName
- Max. Nr. Instances
- Type
- Source, Converter, Sink
- Nr. I/O's
- I/O Energy Domain
- Signal, Hydraulic, Mechanical, Electrical
- SurrogateModelToolPath
- x = [x_D, x_C]
- [x_L, x_U]
- g(x) < 0, h(x) = 0
.XML
CompName
Max. Nr. Instances
Type
Nr. I/O's
I/O Energy Domain
Design Space
Constraints
Constraint Satisfactory Problem
Compatibility Space
Filtering
Reduction of the possible topology solutions based on preference or knowledge base
Preference Space
A set of Discrete Architectural Design Space Topologies which are to be evaluated, to find the most optimal design solution


Tool Library
Tool 1:
- CompName
- Max. Nr. Instances
- Type
- Source, Converter, Sink
- Nr. I/O's
- I/O Energy Domain
- Signal, Hydraulic, Mechanical, Electrical
- SurrogateModelToolPath
- x = [x_D, x_C]
- [x_L, x_U]
- g(x) < 0, h(x) = 0
.XML
A set of Discrete Architectural Design Space Topologies which are to be evaluated, to find the most optimal design solution
Design of Experiment
Mixed - Integer Non-Linear Programming:
EGO
AMIEGO
CompName, Nr. Instances
Session Library
Tool 1:
Tool 2:
Tool N:
SurrogateModelToolPath
x = [x_D, x_C]
[x_L, x_U]
g(x) < 0, h(x) = 0
MDO_Architecture (MDF, IDF)
MDO_Optimizer (Gradient-Based/Free)
MDO_Objective
ToolUncertainty
Uncertainty Quantification Tool
open LEGO
CMDOWS
Knowledge- and graph-based Agile Design with Multidisciplinary Optimization System (KADMOS)
KADMOS
openMDAO
f(x)
objective function
expectation measure
global optimum
robust optimum
x
(a)
(b)


Tool Library
Tool 1:
- CompName
- Max. Nr. Instances
- Type
- Source, Converter, Sink
- Nr. I/O's
- I/O Energy Domain
- Signal, Hydraulic, Mechanical, Electrical
- SurrogateModelToolPath
- $\mathbf{x} = [\mathbf{x}_D, \mathbf{x}_C]$
- $[\mathbf{x}_L, \mathbf{x}_U]$
- $g(\mathbf{x}) < 0,\ h(\mathbf{x}) = 0$ .XML
- $g(\mathbf{x}) < 0,\ h(\mathbf{x}) = 0$ .XML
- $g(\mathbf{x}) < 0,\ h(\mathbf{x}) = 0$ .XML
- $g(\mathbf{x}) < 0,\ h(\mathbf{x}) = 0$ .XML
Experimental Data
Comparison through:
Statistical Analysis
$\mu = \frac{\sum x_i}{N}$ $\sigma^2 = \frac{\sum (x_i - \mu)^2}{N}$ $y = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$
Assume Tool Bias Error based on:
Level of fidelity and/or
Engineering Knowledge and/or
Information provided by tool provider
Analytically calculate Tool Bias Error
Uncertainty propogation verification using Monte Carlo Simulation
Uncertainty Quantification and Propogation Tool
First-Order Perturbation Method
Tool Sensitivity Analysis
"Worst Case Estimation of Objective and Constraints"
$\Delta f = \sum_i \left| \frac{\partial \bar{f}(\mathbf{x}, \mathbf{y})}{\partial x_i} \Delta x_i \right| + \left| \frac{\partial \bar{f}(\mathbf{x}, \mathbf{y})}{\partial y_k} \Delta y_k \right|$
$\Delta g_j = \sum_i \left| \frac{\partial \bar{g}(\mathbf{x}, \mathbf{y})}{\partial x_i} \Delta x_i \right| + \left| \frac{\partial \bar{g}(\mathbf{x}, \mathbf{y})}{\partial y_k} \Delta y_k \right|$
find $\mathbf{x} = [x_1, x_2, ..., x_n]^T$
min $f^R(\mathbf{x}) = \alpha \cdot f + (1 - \alpha) \cdot \Delta f$
s.t. $g_j^R(\mathbf{x}) = g_j - \Delta g_j \geq 0 \quad j = 1, 2, ..., J$
$\mathbf{x}_i^{L^R} \leq \mathbf{x}_i \leq \mathbf{x}_i^{U^R}, \quad i = 1, 2, ..., n$
where $\mathbf{x}_i^{L^R} = x_i^L + \Delta x_i$ and $\mathbf{x}_i^{U^R} = x_i^U - \Delta x_i$
$f = f(\mathbf{x}) = \bar{f}(\mathbf{x}, \mathbf{y})$ and $g = g_j(\mathbf{x}) = \bar{g}_j(\mathbf{x}, \mathbf{y})$
$\Delta f = \sum_i \left| \frac{\partial \bar{f}(\mathbf{x}, \mathbf{y})}{\partial x_i} \Delta x_i \right| + \left| \frac{\partial \bar{f}(\mathbf{x}, \mathbf{y})}{\partial y_k} \Delta y_k \right|$
$\Delta g_j = \sum_i \left| \frac{\partial \bar{g}(\mathbf{x}, \mathbf{y})}{\partial x_i} \Delta x_i \right| + \left| \frac{\partial \bar{g}(\mathbf{x}, \mathbf{y})}{\partial y_k} \Delta y_k \right|$
and $0 \leq \alpha \leq 1$

3

Supporting Work

3.1. Port Matching Constraints

Algorithm 1 Port Matching Algorithm: Constraint 2 - Enforce Port Matching

Number of components in System Architecture Environment: n
$model \leftarrow cpmpy.Model()$
$X \leftarrow$ IntVar(lb $= 0$, ub $= 1$, shape $= (n, n)$)
for $comp_i$ in range(n) **do**
  for $comp_j$ in range(n) **do**
    if $comp_i \neq comp_j$ **then**
      for $output$ in $comp_i$[outputs] **do**
        if $output$ not in $comp_j$[inputs] **then**
          $model \leftarrow model + (X[comp_i, comp_j] == 0)$
        end if
      end for
    end if
  end for
end for
return $model$

Algorithm 2 Port Matching Algorithm: Constraint 3 - Enforce Proper Number of Port Matchings

Number of components in System Architecture Environment: n
$model \leftarrow cpmpy.Model()$
$X \leftarrow$ IntVar(lb $= 0$, ub $= 1$, shape $= (n, n)$)
for $comp_i$ in range(n) **do**
  if $comp_i$[inputs] $= \emptyset$ **then**
    $model \leftarrow model + (\sum X[:, comp_i] == 0)$
  end if
  if $comp_i$[outputs] $= \emptyset$ **then**
    $model \leftarrow model + (\sum X[comp_i, :] == 0)$
  end if
end for
for $input_i$ in Count$(comp_i$[inputs]$)$ **do**
  $model \leftarrow model + (\text{min}(\text{Count}(input_i)) \leq \sum X[:, comp_i])$
  $model \leftarrow model + (\sum X[:, comp_i] \leq \text{max}(\text{Count}(input_i)))$
end for
for $output_i$ in Count$(comp_i$[outputs]$)$ **do**
  $model \leftarrow model + (\text{min}(\text{Count}(output_i)) \leq \sum X[comp_i, :])$
  $model \leftarrow model + (\sum X[comp_i, :] \leq \text{max}(\text{Count}(output_i)))$
end for
return $model$

3.2. Component Library Case Study

battery.xml

```
<component>
    <name>BAT</name>
    <TRL>9</TRL>
    <maxInstances>2</maxInstances>
    <typeInput></typeInput>
    <minRangeInput></minRangeInput>
    <maxRangeInput></maxRangeInput>
    <incompatibleInput></incompatibleInput>
    <typeOutput>Electrical DC, Signal</typeOutput>
    <minRangeOutput>1,1</minRangeOutput>
    <maxRangeOutput>1,1</maxRangeOutput>
    <incompatibleOutput></incompatibleOutput>
    <designVariables>BatteryType</designVariables>
    <initialValue>1</initialValue>
    <lowerBound>1</lowerBound>
    <upperBound>3</upperBound>
    <designVariablesType>Int</designVariablesType>
    <controlVariables></controlVariables>
    <lowerBoundControl></lowerBoundControl>
    <upperBoundControl></upperBoundControl>
    <controlVariablesType></controlVariablesType>
    <stateVariable></stateVariable>
    <initialValueState></initialValueState>
    <lowerBoundState></lowerBoundState>
    <upperBoundState></upperBoundState>
    <stateVariableType></stateVariableType>
    <modellingPath>modelling\battery\battery_categorical.py</modellingPath>
</component>
```

clutch.xml

```
<component>
    <name>CLTCH</name>
    <TRL>9</TRL>
    <maxInstances>3</maxInstances>
    <typeInput>Mechanical Rot, Signal</typeInput>
    <minRangeInput>1,1</minRangeInput>
    <maxRangeInput>1,1</maxRangeInput>
    <incompatibleInput>CLTCH</incompatibleInput>
    <typeOutput>Mechanical Rot</typeOutput>
    <minRangeOutput>1</minRangeOutput>
    <maxRangeOutput>1</maxRangeOutput>
    <incompatibleOutput>CLTCH</incompatibleOutput>
    <designVariables></designVariables>
    <initialValue></initialValue>
    <lowerBound></lowerBound>
    <upperBound></upperBound>
    <designVariablesType></designVariablesType>
    <controlVariables>clutchCommand</controlVariables>
    <lowerBoundControl>-1</lowerBoundControl>
    <upperBoundControl>1</upperBoundControl>
    <controlVariablesType>Int</controlVariablesType>
    <stateVariable>clutchEngaged</stateVariable>
    <initialValueState>1</initialValueState>
    <lowerBoundState>0</lowerBoundState>
    <upperBoundState>1</upperBoundState>
    <stateVariableType>Int</stateVariableType>
    <modellingPath>modelling\powertrain\clutch.py</modellingPath>
</component>
```

EM.xml

```
<component>
    <name>EM</name>
    <TRL>9</TRL>
    <maxInstances>2</maxInstances>
    <typeInput>Electrical AC, Signal</typeInput>
    <minRangeInput>1,1</minRangeInput>
    <maxRangeInput>2,1</maxRangeInput>
    <incompatibleInput></incompatibleInput>
    <typeOutput>Mechanical Rot</typeOutput>
    <minRangeOutput>1</minRangeOutput>
    <maxRangeOutput>1</maxRangeOutput>
    <incompatibleOutput></incompatibleOutput>
    <designVariables>ElectricMachineType</designVariables>
    <initialValue>1</initialValue>
    <lowerBound>1</lowerBound>
    <upperBound>3</upperBound>
    <designVariablesType>Int</designVariablesType>
    <controlVariables></controlVariables>
    <lowerBoundControl></lowerBoundControl>
    <upperBoundControl></upperBoundControl>
    <controlVariablesType></controlVariablesType>
    <stateVariable></stateVariable>
    <initialValueState></initialValueState>
    <lowerBoundState></lowerBoundState>
    <upperBoundState></upperBoundState>
    <stateVariableType></stateVariableType>
    <modellingPath>modelling\electricmachine\electricmachine_categorical.py</modellingPath>
</component>
```

fuelcell.xml

```
<component>
    <name>FC</name>
    <TRL>7</TRL>
    <maxInstances>5</maxInstances>
    <typeInput>Chemical, Signal</typeInput>
    <minRangeInput>1,1</minRangeInput>
    <maxRangeInput>1,1</maxRangeInput>
    <incompatibleInput></incompatibleInput>
    <typeOutput>Electrical DC</typeOutput>
    <minRangeOutput>1</minRangeOutput>
    <maxRangeOutput>1</maxRangeOutput>
    <incompatibleOutput></incompatibleOutput>
    <designVariables>FuelCellType</designVariables>
    <initialValue>1</initialValue>
    <lowerBound>1</lowerBound>
    <upperBound>4</upperBound>
    <designVariablesType>Int</designVariablesType>
    <controlVariables>deltapowerFractionFC</controlVariables>
    <lowerBoundControl>0.1</lowerBoundControl>
    <upperBoundControl>-0.1</upperBoundControl>
    <controlVariablesType>Float</controlVariablesType>
    <stateVariable>powerFraction</stateVariable>
    <initialValueState>0.2</initialValueState>
    <lowerBoundState>0.2</lowerBoundState>
    <upperBoundState>1</upperBoundState>
    <stateVariableType>Float</stateVariableType>
    <modellingPath>CODE\modelling\fuelcell\fuelcell.py</modellingPath>
</component>
```

mainreducer.xml

```
<component>
    <name>RED</name>
    <TRL>9</TRL>
    <maxInstances>1</maxInstances>
    <typeInput>Mechanical Rot, Signal</typeInput>
    <minRangeInput>1,1</minRangeInput>
    <maxRangeInput>1,1</maxRangeInput>
    <incompatibleInput>RED</incompatibleInput>
    <typeOutput>Mechanical Rot</typeOutput>
    <minRangeOutput>1</minRangeOutput>
    <maxRangeOutput>1</maxRangeOutput>
    <incompatibleOutput>MSG</incompatibleOutput>
    <designVariables>RED_gearRatio1</designVariables>
    <initialValue>7</initialValue>
    <lowerBound>4</lowerBound>
    <upperBound>8</upperBound>
    <designVariablesType>Float</designVariablesType>
    <controlVariables></controlVariables>
    <lowerBoundControl></lowerBoundControl>
    <upperBoundControl></upperBoundControl>
    <controlVariablesType></controlVariablesType>
    <stateVariable></stateVariable>
    <initialValueState></initialValueState>
    <lowerBoundState></lowerBoundState>
    <upperBoundState></upperBoundState>
    <stateVariableType></stateVariableType>
    <modellingPath>modelling\powertrain\mainreducer.py</modellingPath>
</component>
```

msgearbox.xml

```
<component>
    <name>MSG</name>
    <TRL>9</TRL>
    <maxInstances>2</maxInstances>
    <typeInput>Mechanical Rot, Signal</typeInput>
    <minRangeInput>1,1</minRangeInput>
    <maxRangeInput>1,1</maxRangeInput>
    <incompatibleInput>MSG</incompatibleInput>
    <typeOutput>Mechanical Rot</typeOutput>
    <minRangeOutput>1</minRangeOutput>
    <maxRangeOutput>1</maxRangeOutput>
    <incompatibleOutput>MSG</incompatibleOutput>
    <designVariables>MSG_gearRatio1, MSG_gearRatio2, MSG_gearRatio3, MSG_gearRatio4,
        MSG_gearRatio5, MSG_gearRatio6</designVariables>
    <initialValue>6,4.9,3.6,2.8,2.4,1.75</initialValue>
    <lowerBound>4,4,2.8,2,1.8,1.5</lowerBound>
    <upperBound>7,6,4,3.6,2.8,2.3</upperBound>
    <designVariablesType>Float, Float, Float, Float, Float, Float</designVariablesType>
    <controlVariables>shift</controlVariables>
    <lowerBoundControl>-1</lowerBoundControl>
    <upperBoundControl>1</upperBoundControl>
    <controlVariablesType>Int</controlVariablesType>
    <stateVariable>gear</stateVariable>
    <initialValueState>0</initialValueState>
    <lowerBoundState>0</lowerBoundState>
    <upperBoundState>5</upperBoundState>
    <stateVariableType>Int</stateVariableType>
    <modellingPath>modelling\powertrain\transmission.py</modellingPath>
</component>
```

pdu.xml

```
<component>
    <name>PDU</name>
    <TRL>8</TRL>
    <maxInstances>2</maxInstances>
    <typeInput>Electrical DC, Signal</typeInput>
    <minRangeInput>1,1</minRangeInput>
    <maxRangeInput>6,1</maxRangeInput>
    <incompatibleInput>PDU</incompatibleInput>
    <typeOutput>Electrical DC</typeOutput>
    <minRangeOutput>1</minRangeOutput>
    <maxRangeOutput>4</maxRangeOutput>
    <incompatibleOutput>PDU</incompatibleOutput>
    <designVariables></designVariables>
    <initialValue></initialValue>
    <lowerBound></lowerBound>
    <upperBound></upperBound>
    <designVariablesType></designVariablesType>
    <controlVariables></controlVariables>
    <lowerBoundControl></lowerBoundControl>
    <upperBoundControl></upperBoundControl>
    <controlVariablesType></controlVariablesType>
    <stateVariable></stateVariable>
    <initialValueState></initialValueState>
    <lowerBoundState></lowerBoundState>
    <upperBoundState></upperBoundState>
    <stateVariableType></stateVariableType>
    <modellingPath>modelling\powertrain\pdu.py</modellingPath>
</component>
```

torquecoupler.xml

```
<component>
    <name>TC</name>
    <TRL>9</TRL>
    <maxInstances>2</maxInstances>
    <typeInput>Mechanical Rot</typeInput>
    <minRangeInput>1</minRangeInput>
    <maxRangeInput>2</maxRangeInput>
    <incompatibleInput>TC</incompatibleInput>
    <typeOutput>Mechanical Rot</typeOutput>
    <minRangeOutput>1</minRangeOutput>
    <maxRangeOutput>1</maxRangeOutput>
    <incompatibleOutput>TC</incompatibleOutput>
    <designVariables>TC_gearRatio1, TC_gearRatio2</designVariables>
    <initialValue>1,1</initialValue>
    <lowerBound>0.8,1</lowerBound>
    <upperBound>1.2,1.2</upperBound>
    <designVariablesType>Float, Float</designVariablesType>
    <controlVariables>deltasplitRatioTC</controlVariables>
    <lowerBoundControl>-1</lowerBoundControl>
    <upperBoundControl>1</upperBoundControl>
    <controlVariablesType>Float</controlVariablesType>
    <stateVariable>splitRatioTC</stateVariable>
    <initialValueState>0.5</initialValueState>
    <lowerBoundState>0</lowerBoundState>
    <upperBoundState>1</upperBoundState>
    <stateVariableType>Float</stateVariableType>
    <modellingPath>modelling\powertrain\torquecoupler.py</modellingPath>
</component>
```

wheels.xml

```
<component>
    <name>Wheels</name>
    <TRL>9</TRL>
    <maxInstances>1</maxInstances>
    <typeInput>Mechanical Rot</typeInput>
    <minRangeInput>1</minRangeInput>
    <maxRangeInput>1</maxRangeInput>
    <incompatibleInput></incompatibleInput>
    <typeOutput>Mechanical Trans</typeOutput>
    <minRangeOutput>1</minRangeOutput>
    <maxRangeOutput>1</maxRangeOutput>
    <incompatibleOutput></incompatibleOutput>
    <designVariables>wheelDiameter</designVariables>
    <initialValue>0.98</initialValue>
    <lowerBound>0.9</lowerBound>
    <upperBound>1.1</upperBound>
    <designVariablesType>Float</designVariablesType>
    <controlVariables></controlVariables>
    <lowerBoundControl></lowerBoundControl>
    <upperBoundControl></upperBoundControl>
    <controlVariablesType></controlVariablesType>
    <stateVariable></stateVariable>
    <initialValueState></initialValueState>
    <lowerBoundState></lowerBoundState>
    <upperBoundState></upperBoundState>
    <stateVariableType></stateVariableType>
    <modellingPath>modelling\vehicle\vehicle.py</modellingPath>
</component>
```

3.3. Selected Output Component For Case Study

problemlibrary.xml

```
<library><component>
    <name>Road</name>
    <TRL>9</TRL>
    <maxInstances>1</maxInstances>
    <typeInput>Mechanical Trans</typeInput>
    <minRangeInput>1</minRangeInput>
    <maxRangeInput>4</maxRangeInput>
    <incompatibleInput />
    <typeOutput />
    <minRangeOutput />
    <maxRangeOutput />
    <incompatibleOutput />
    <designVariables />
    <initialValue />
    <lowerBound />
    <upperBound />
    <designVariablesType />
    <controlVariables />
    <lowerBoundControl />
    <upperBoundControl />
    <controlVariablesType />
    <stateVariable />
    <initialValueState />
    <lowerBoundState />
    <upperBoundState />
    <stateVariableType />
    <modellingPath />
</component></library>
```

3.4. Topological Sorting Algorithm

Algorithm 3 Topological Sorting

Require: $G = (V, E)$ to be a DAG
Let L be an empty list that will contain the sorted elements
Let S be a set of all nodes with no incoming edges ($S(V.indegree = 0)$)
while S is not empty **do**
    Remove a node n from S
    Add n to L
    for each node m with an edge e from n to m **do**
        Remove edge e from the graph
        if m has no other incoming edges **then**
            Insert m into S
        end if
    end for
end while
if the graph has edges **then**
    error (the graph has at least one cycle)
else
    return L (a topologically sorted order)
end if

The algorithm returns a sequence or array of nodes where each node appears before all the nodes that it points to. Note that this algorithm does not produce unique solutions, meaning that multiple solutions can be found.

3.5. Optimal Control Solver Using Dijkstra's Algorithm

Below, an example is shown for a drive train with three state variables, where the state variable space is discretized and full-factorial sampled, returning a total of 130 possible operating states per timestep. For all states at $t = t_i$, the set of reachable states at time step $t = t_{i+1}$ ($\mathbf{L} \subset \mathbf{K}$) is shown in a Control Viable Edge Matrix. Here, all reachable states (*Target states*) from the *Source states* are shown in black.

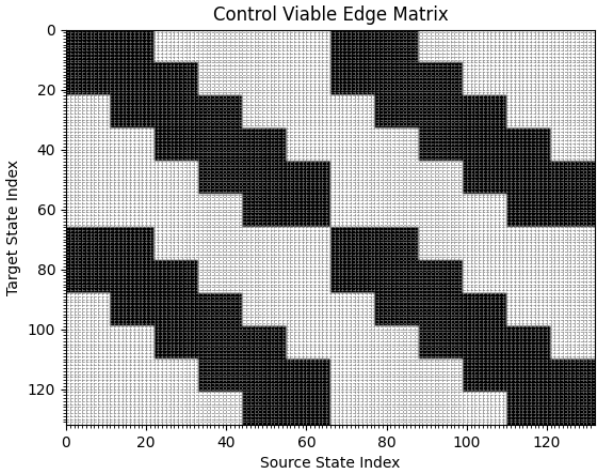


Figure 3.1: Viable edge matrix showing the reachable states given the control constraints

An example showing all reachable states ($\mathbf{L} \subset \mathbf{K}$) from state 40 is shown in Figure 3.2, together with the associated weights.

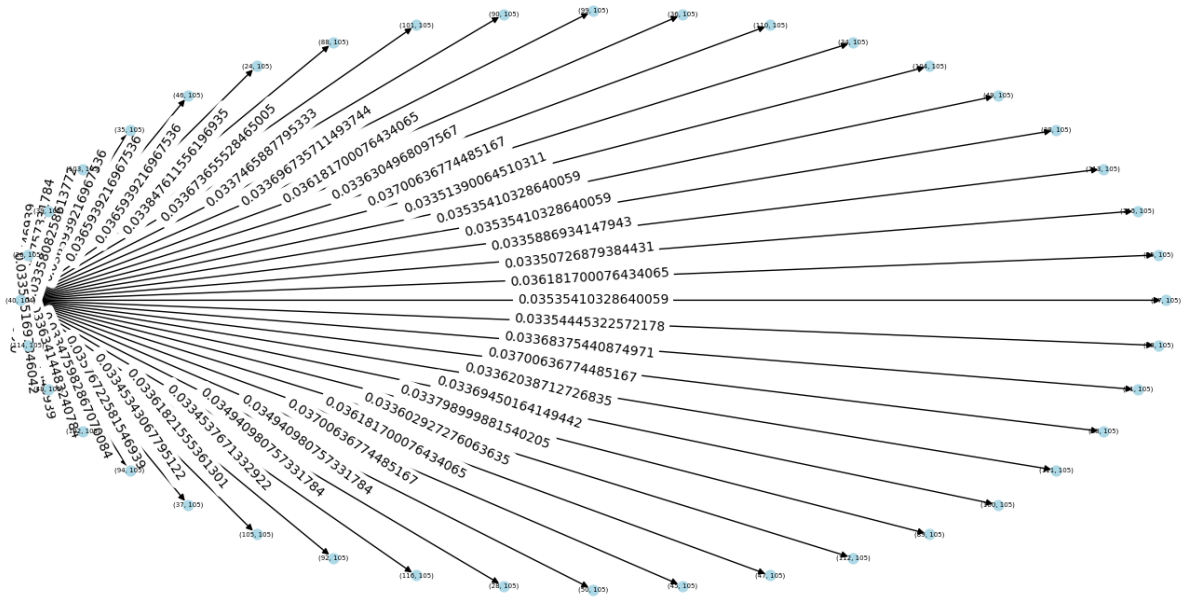


Figure 3.2: A Graph Representation of a single state (40) DWG at time step 384

In this research, Dijkstra's algorithm was selected for its computational efficiency and suitability to the use case. A comparison with another shortest path algorithm, namely Bellman-Ford, is presented in Table 3.1.

Table 3.1: Comparing how different shortest path algorithms cope with graph features. A comparison is made with this research.

	Dijkstra	Bellman-Ford	This research
Non-Negative Weights	Works correctly for directed and undirected graphs	Works correctly for directed and undirected graphs	✓
Negative Weights	Fails	Works correctly with directed graphs only	✓
Negative cycles	Fails	Can detect negative cycles in directed graphs	×
Time Complexity	O(V + E $\cdot log(V)$)	O(V $\cdot E$)	

Bellman-Ford offers several advantages over Dijkstra, particularly in handling graphs with both positive and negative edge weights. Given that this research considers both types of weights (battery-discharge and charge), Bellman-Ford initially seems the only suitable option. However, Table 3.1 shows that Bellman-Ford's time complexity is significantly higher. Here the number of nodes equals the product of valid states per time step and the number of time steps, and the number of edges equals the size of $\mathbf{L}$ per time step per state. This computational demand makes Bellman-Ford impractical for implementation in this study. Instead, a workaround involves adjusting the Directed Acyclic Weighted Graph (DAWG) to ensure all weights are non-negative. This is achieved by shifting all weights by the minimum weight found in the graph. After finding the shortest path, these weights are adjusted back to their original values by subtracting the minimum value.

Some may question whether adjusting weights to be positive yields valid results. This concern is addressed in the following figure:

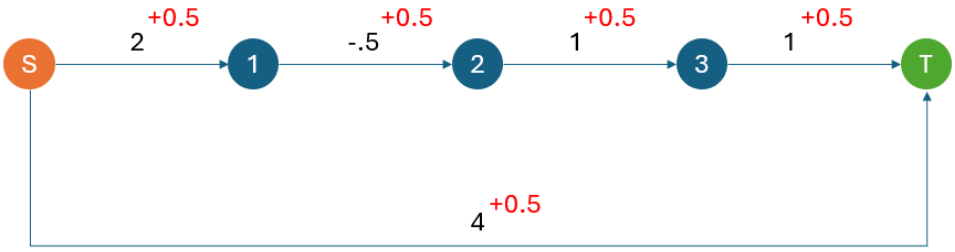


Figure 3.3: Example showing why graph weight shifting typically does not produce valid results

In Figure 3.3, the sum of weights along the top path is 3.5 (shown in black), whereas the sum along the bottom path is 4. Therefore, the shortest path is found to be the top path. However, upon shifting the weights by the most negative value (here, $w_{(1,2)} = -0.5$), the sum along the top path becomes 5.5, and along the bottom path it becomes 4.5. Consequently, the bottom path is now the shortest path. This phenomenon occurs because the bottom path has fewer edges, resulting in a lesser overall impact of the weight shift on the total weight sum.

In this study, the number of edges between consecutive time steps is fixed, meaning the drive train must progress from a state at $t = t_i$ to a state at $t = t_{i+1}$ without skipping time steps (e.g., jumping from $t = t_i$ to $t = t_{i+10}$). Thus, any path from a state at $t = 0$ to a state at $t = t_{\text{end}}$ consists of a sequence of edges equal to the number of time steps. This characteristic is illustrated in Figure 3.4.

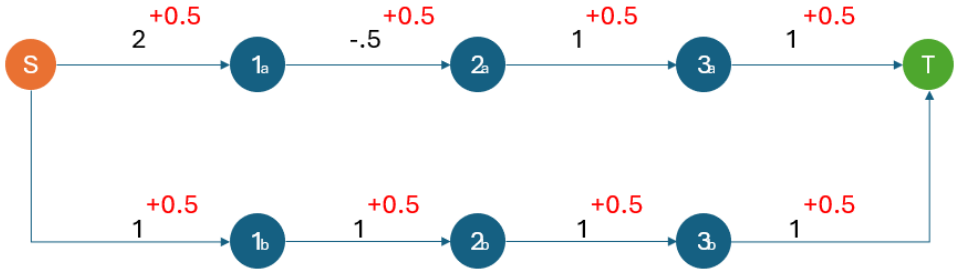


Figure 3.4: Example showing why graph weight shifting works for this research and produces valid results

3.6. Surrogate Models For Computing The J^* Matrix

Surrogate models are typically used when the computational expense of standard function evaluations becomes prohibitively high or when the trade-off between accuracy and computational cost is acceptable. By using surrogate models, users can explore the design space with reduced computational cost at the cost of a slight decrease in accuracy. The application of surrogate models is common in system architecture optimization during conceptual design phases, where exhaustive exploration of the entire design space is impractical and unnecessary [9].

Surrogate models work by sampling the design space, where the sampling set size is much smaller than the size of the design space, $\mathcal{O}(Size(\text{sampling set})) \ll \mathcal{O}(Size(\text{design space}))$. The computationally expensive function evaluations are performed on this sampled set to create a training dataset. Subsequently, a surrogate model is trained on this data, which can range from simple regression models to more complex models like neural networks. When new inputs need evaluation, the surrogate model provides predictions at a significantly lower computational cost. However, there may be slight discrepancies between the outputs of the surrogate model and those of the original function, as illustrated in Figure 3.5.

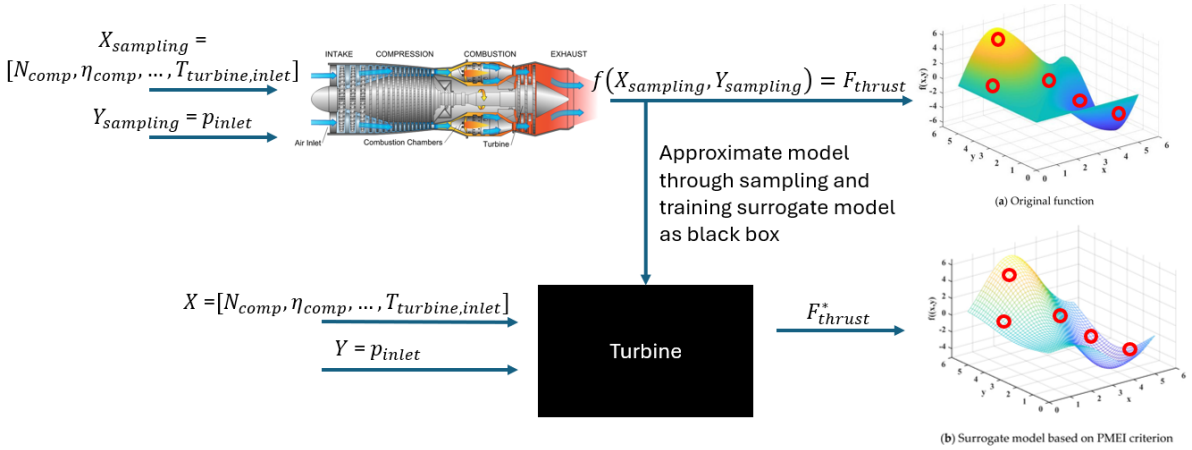


Figure 3.5: Example of the procedure of using a surrogate model

Due to the highly discontinuous nature of the state-time space, where the distinction between valid and invalid states is determined by a penalty value assigned to invalid states, and valid states have very low order of magnitudes, training accurate surrogate models becomes challenging. Various approaches have been explored in the literature to address this issue, including the use of local surrogates to differentiate between valid and invalid state-time regions, employing mixture of experts [1], Gaussian Processes (Global Kriging), or simply increasing the size of the training set. In this research, multiple methods were investigated, but struggled to capture the state-space trends with sufficient accuracy. This difficulty arises because valid states typically exhibit low values of ΔSoC, whereas invalid states are penalized with higher ΔSoC values (specifically, a penalty of 20 in this study), and reducing this penalty did not resolve the issue.

This raises the question of whether we can exploit the fact that these invalid indices provide no useful information and therefore do not require accurate estimation. From this perspective, an approach was developed in this study that utilizes random forest classification trees to accurately classify state-time indices as either invalid or valid. Subsequently, a random forest regression tree, is trained exclusively on the valid states to construct a combined surrogate model. This combined model aims to accurately predict the entire state-time performance matrix at a reduced computational cost. A flowchart depicting this methodology is presented in Figure 3.6 and Figure 3.7.

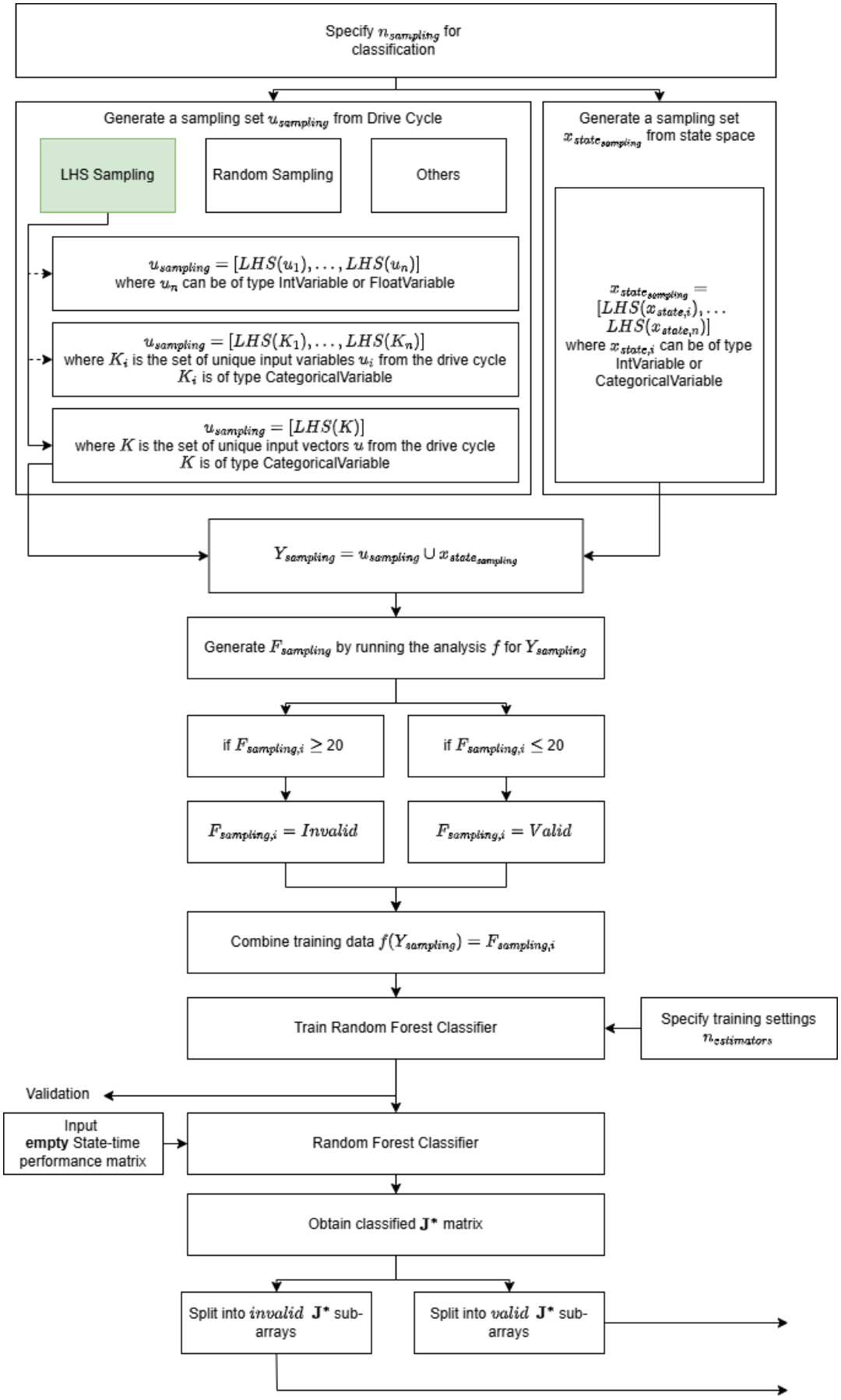


Figure 3.6: Flow Chart of Classification Surrogate Model

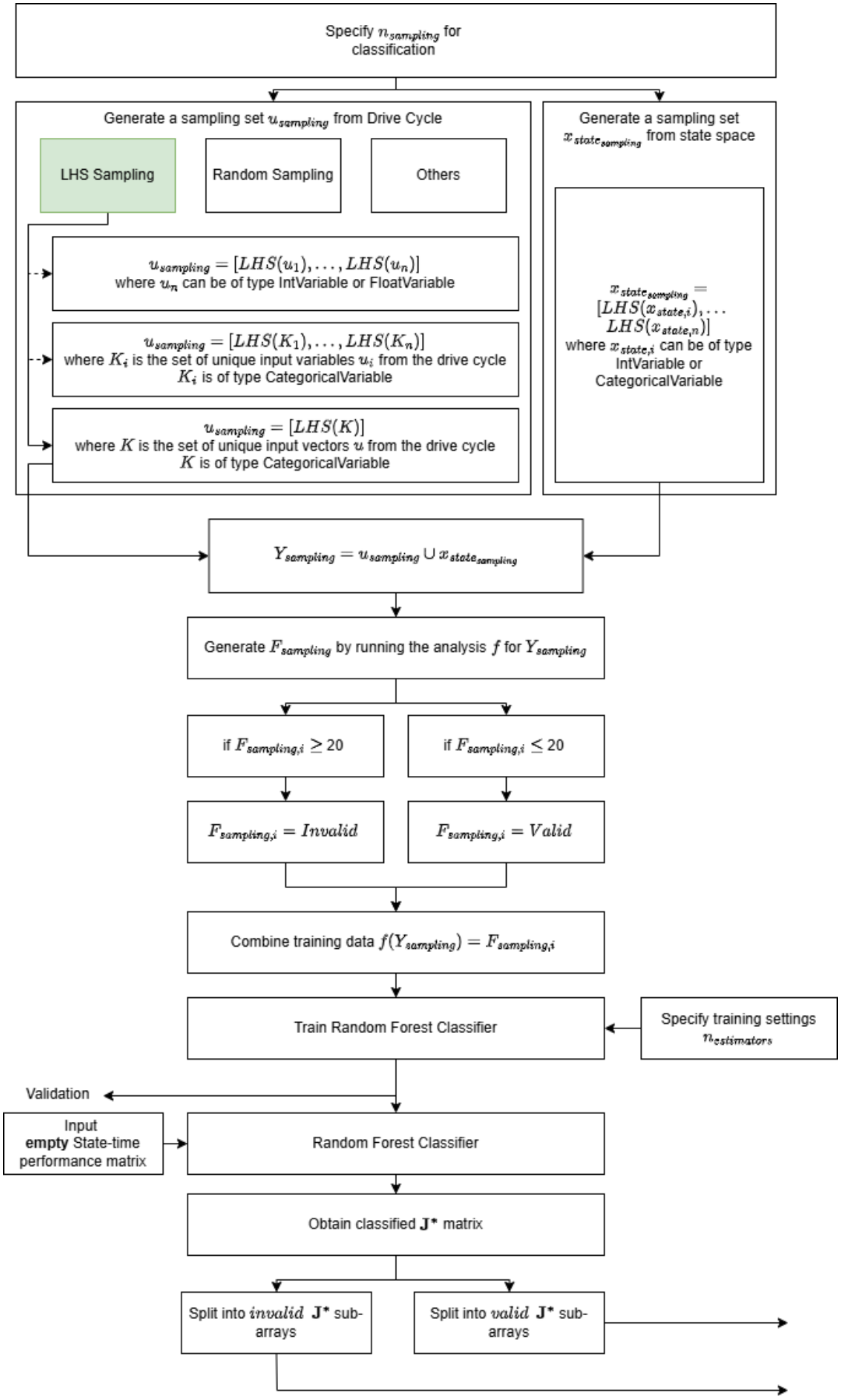


Figure 3.7: Flow Chart of Regression Surrogate Model

Employing the random forest classifier surrogate model achieved an impressive accuracy of 98.6% in classifying valid and invalid states throughout the drive cycle. The confusion matrix depicting the true-positive and false-negative estimations is illustrated inFigure 3.8. This high accuracy demonstrates that even with a small sampling set of 1000 data points (0.7% of the total), the classifier effectively captures the system architecture trends looking at (in)valid states.

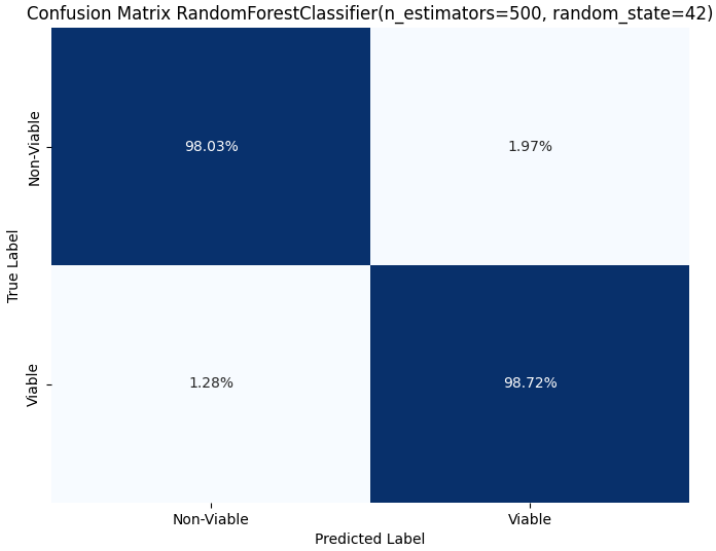


Figure 3.8: Confusion Matrix for the Random Forest Classifier with $n_{estimators} = 500$ with a training set size of 1000.

As seen in Figure 3.7, next, K-th sampling was implemented on the set of feasible states Y_i per time index t_i. Here, the size of Y_i, determines the number of sampling points. This approach yields a higher impact of these states on the surrogate model when the size of Y_i is large. A minimum of 4 sampling points per time index was taken. This method achieved an impressive 96% accuracy compared to other techniques, which was deemed acceptable for this research. Validation across various system architectures involved testing multiple sampling strategies, ranging from random to K-th sampling, as depicted in Figure 3.9. A statistical analysis shows that indeed the method proposed using K-th sampling returns the lowest mean error when compared to a non-surrogate, or analytical model. Here, a density based approach with a 6% training set size returns the most optimal results.

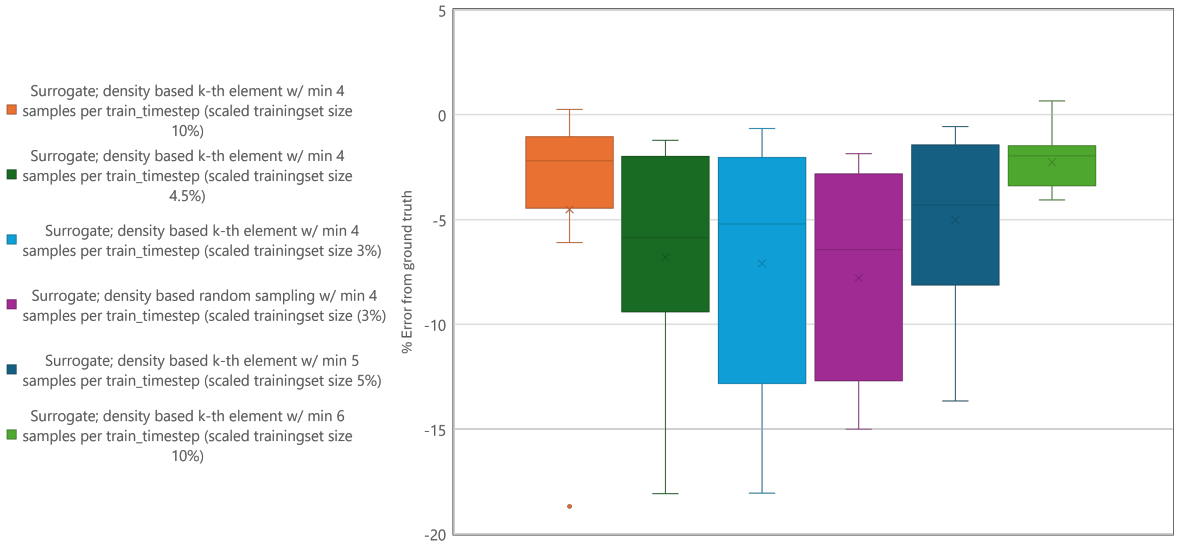


Figure 3.9: Statistical analysis showing the error distribution in the form of a box-plot of the surrogate model with different sampling techniques compared to the analytical model (ground truth). This was done for various system architectures.

3.7. Component Models

This section will go into the various component models used in the optimal control solver to model the dynamic behavior of the drive train. Please note that subscript of i, input is used for input shafts and o, output is used for output shafts.

3.7.1. Clutch

A clutch is an active mechanical coupler which allows for the dynamic decoupling of the in- and output shaft. Clutches are commonly used in internal combustion engine cars to decouple the engine from the gearbox during shifting.

$$(T_i, \omega_i) = \begin{cases} (T_o, \omega_o) & \text{if } \textit{clutchEngaged} = 1.0 \\ (0, 0) & \text{if } \textit{clutchEngaged} = 0.0 \end{cases} \tag{3.1}$$

3.7.2. Main Reducer

A main reducer is a of single speed gearbox which allows to convert the high torque, low RPM at the output to a more manageable low torque, high RPM range at the input for high gear ratio's. Main reducer are commonly used in drive trains to alter the rpm range to be within the operational range of the mechanical rotational energy supplier, here the electric machine. A main reducer is a passive system, which performance can be modelled using simple equations as can be seen below.

$$T_i = \begin{cases} \frac{T_o}{k_{red}\,\eta_{reducer}} & \text{if } T_o > 0.0 \\ \frac{T_o\,\eta_{reducer}}{k_{red}} & \text{if } T_o < 0.0 \end{cases} \qquad \omega_i = \omega_o k_{red} \tag{3.2}$$

3.7.3. Planetary Gearbox

A planetary gearbox is used in drive trains to alter the working incoming torque and rpm ranges to a more manageable range. Planetary gearboxes are a multi input/output mechanical component, where three gears are present, namely the sun-, planet (yoke)- and ring gear. Planetary gearboxes work, opposite to torque couplers, by speed coupling the various input- and output shafts. Here the torque is kept constant, but the RPM from gear-to-gear differs. Planetary Gearboxes are actively controlled mechanical components, which dynamically can alter the rotational energy of the various shafts by locking various gears.

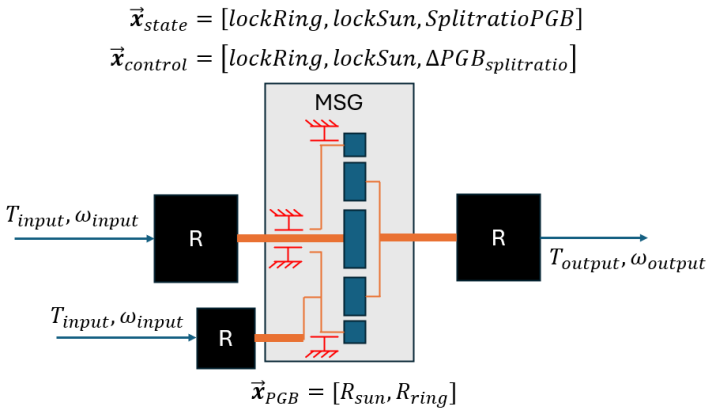


Figure 3.10: Schematic of a planetary gearbox with its component specifications and control- and state variables.

if: lockRing = 0 , lockSun = 0, and 0 < splitRatioPGB < 1

Case: Output connected to Sun Gear

$$\omega_{\text{yoke}} = \frac{1}{k_{ys}}\omega_{\text{sun}} \times \text{splitRatioPGB}$$
$$T_{\text{yoke}} = -k_{ys}T_{\text{sun}}$$
$$\omega_{\text{ring}} = \omega_{\text{sun}} \times \frac{k_{yr}}{k_{ys}} \times (-\text{splitRatioPGB} - 1)$$
$$T_{\text{ring}} = T_{\text{sun}} \times \frac{-k_{ys}}{-k_{yr}}$$

Case: Output shaft connected to Yoke Gear

$$\omega_{\text{ring}} = k_{yr} \times (1 + \text{splitRatioPGB}) \times \omega_{\text{yoke}}$$
$$T_{\text{ring}} = \frac{T_{\text{yoke}}}{-k_{yr}}$$
$$\omega_{\text{sun}} = \text{splitRatioPGB} \times (-k_{ys}) \times \omega_{\text{yoke}}$$
$$T_{\text{sun}} = \frac{T_{\text{yoke}}}{-k_{ys}}$$

Case: Output shaft connected to Ring Gear

$$\omega_{\text{sun}} = \text{splitRatioPGB} \times \frac{k_{ys}}{k_{yr}} \times \omega_{\text{ring}}$$
$$T_{\text{sun}} = \frac{-k_{yr}T_{\text{ring}}}{-k_{ys}}$$
$$\omega_{\text{yoke}} = \frac{\omega_{\text{ring}} \times (\text{splitRatioPGB} + 1)}{k_{yr}}$$
$$T_{\text{yoke}} = -k_{yr}T_{\text{ring}}$$

if: lockRing = 0 and lockSun = 1

Case: Output shaft connected to Ring Gear

$$\omega_{\text{yoke}} = \frac{1}{k_{yr}}\omega_{\text{ring}}$$
$$T_{\text{yoke}} = -k_{yr}T_{\text{ring}}$$
$$T_{\text{sun}} = \frac{-k_{yr}T_{\text{ring}}}{-k_{ys}}$$

Case: Output shaft connected to Yoke Gear

$$\omega_{\text{ring}} = \omega_{\text{yoke}} \times k_{yr}$$
$$T_{\text{ring}} = \frac{T_{\text{yoke}}}{-k_{yr}}$$
$$T_{\text{sun}} = \frac{T_{\text{yoke}}}{-k_{ys}}$$

if: lockRing = 1 and lockSun= 0

Case: Output shaft connected to Sun Gear

$$\omega_{\text{yoke}} = \frac{1}{k_{ys}}\omega_{\text{sun}}$$
$$T_{\text{yoke}} = -k_{ys}T_{\text{sun}}$$
$$T_{\text{ring}} = \frac{-k_{ys}T_{\text{sun}}}{-k_{yr}}$$

Case: Output shaft connected to Yoke Gear

$$\omega_{\text{sun}} = \omega_{\text{yoke}} \times k_{ys}$$
$$T_{\text{sun}} = \frac{T_{\text{yoke}}}{-k_{ys}}$$
$$T_{\text{ring}} = \frac{T_{\text{yoke}}}{-k_{yr}}$$

3.7.4. Torque Coupler

A torque coupler is a type of mechanical coupler also known as a Power Take-Off (PTO). Torque couplers are used to redistribute torque among multiple outputs. Unlike planetary gearboxes, which primarily adjust speed while maintaining constant torque across outputs, torque couplers maintain constant RPM across outputs while enabling independent torque control per shaft [[4], [3]]. The operational principle of torque couplers contrasts with that of planetary gearboxes, which are designed as speed couplers allowing for independent speed control of output shafts while maintaining constant torque. Torque couplers, on the other hand, distribute torque according to the following relationships:

$$T_{i,1}\begin{cases}\frac{T_o\,TSR}{\eta_{TC}} & \text{if } T_o > 0.0\\ T_o\,TSR\,\eta_{TC} & \text{if } T_o < 0.0\end{cases} \qquad T_{i,2}\begin{cases}\frac{(1-TSR)\,k_1\,T_o}{k_2\,\eta_{TC}} & \text{if } T_o > 0.0\\ \frac{(1-TSR)\,k_1\,T_o\,\eta_{TC}}{k_2} & \text{if } T_o < 0.0\end{cases}$$

$$\omega_{i,1}\begin{cases}\omega_o\,k_1 & \text{if } T_o > 0.0\\ \omega_o\,k_1 & \text{if } T_o < 0.0\end{cases} \qquad \omega_{i,2}\begin{cases}\omega_o\,k_2 & \text{if } T_o > 0.0\\ \omega_o\,k_2 & \text{if } T_o < 0.0\end{cases}$$

3.7.5. Multi-Speed Gearboxes

A multi-speed gearbox, also referred to as a multi-gear transmission, plays a crucial role in mechanical systems by enabling varied torque and speed combinations across its outputs. Unlike single-speed gearboxes that provide fixed speed and torque ratios, multi-speed gearboxes offer multiple gear ratios selectable by the operator or automatically adjusted based on operating conditions. This capability allows for optimizing vehicle performance across different speeds and loads.

In contrast to torque couplers, which maintain constant RPM across outputs while adjusting torque independently, multi-speed gearboxes primarily focus on varying both speed and torque simultaneously across different outputs. This versatility makes them essential in applications requiring efficient power transmission, such as automotive transmissions and industrial machinery [6].

The operational principle of multi-speed gearboxes involves gear sets with different ratios, typically achieved through gear meshing and synchronization mechanisms. The multi-speed gearbox is modelled using the same set of equations as the (single-speed) main reducer, Equation 3.2.

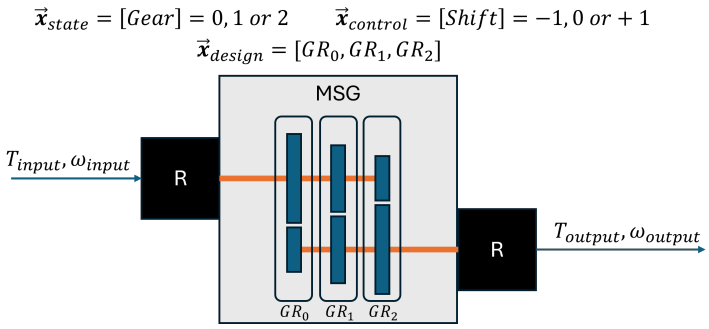


Figure 3.11: Schematic of a three-speed gearbox with its component specifications and control- and state variables

3.7.6. Electric Machine

The electro-mechanical component explored in this study is an electric machine, functioning as a transformer block that converts electrical energy into rotational energy. This conversion process involves transforming electrical energy into a rotating magnetic field, which subsequently drives the motion of the output shaft.

Considerable research has been devoted to developing accurate low- to medium-fidelity modeling tools [2], typically employed in the conceptual design phase, to simulate and predict the performance of electric machines under varying operational conditions. To obtain more practical performance data for electric machines, a generic method is proposed in this research. This method utilizes the torque/RPM/efficiency characteristics of specific electric machines as inputs, employing 2D grid interpolation techniques to derive the electric power required for a given torque and RPM.

This study focuses exclusively on alternating current (AC) electric machines, chosen for their high efficiency and widespread use across various industries. AC electric machines necessitate a three-phase AC input to initiate rotation. The equations utilized to model the electric machine are detailed below:

$$V_{AC_{rms,\Delta}} = 500 \tag{3.3}$$

$$A_{AC_{rms,\Delta}} = \frac{T_o \cdot \omega_o}{\eta_{EM}(T_o, \omega_o) \cdot V_{AC} \cdot PF \cdot \sqrt{3}} \tag{3.4}$$

$$V_{AC_{rms,Y}} = \frac{500}{\sqrt{3}} \tag{3.5}$$

$$A_{AC_{rms,Y}} = \frac{T_o \cdot \omega_o}{\eta_{EM}(T_o, \omega_o) \cdot V_{AC} \cdot PF} \tag{3.6}$$

$$\eta_{EM}(T_o, \omega_o) = GridInterpolate\left([T_i, T_j], [\omega_i, \omega_j]\right)(T_o, \omega_o) \tag{3.7}$$

where $T_i < T_o < T_j$ and $\omega_i < \omega_o < \omega_j$

3.7.7. Inverter

As electric machines typically require three-phase alternating current while batteries supply direct current, an additional transformer block is necessary to convert DC power into three-phase AC power. This conversion is typically achieved using an inverter. In mobility applications, three-phase inverters switch DC current using transistors at high frequencies, typically ranging from a few kHz to tens of kHz. This switching generates Pulse Width Modulation (PWM) signals, approximating sinusoidal phase currents and voltages. Inverter efficiency depends significantly on the switching frequency; higher frequencies reduce the size of required inductors, lowering resistance values and thereby increasing efficiency. An electrical schematic of a typical three-phase inverter used in automotive applications is illustrated in Figure 3.13. Also the equations used to compute the inverter supply (DC) voltage and current are shown.

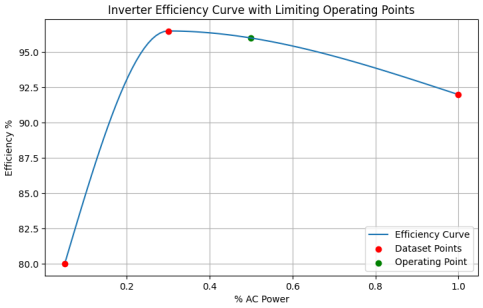


Figure 3.12: Typical η_{inv} curve

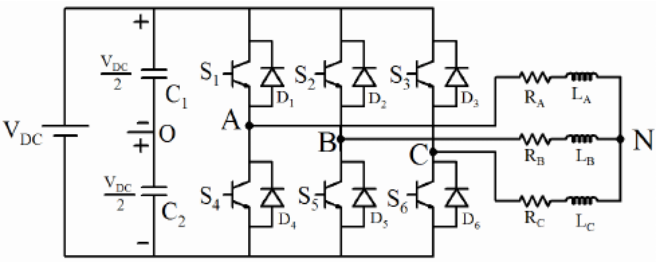


Figure 3.13: Inverter schematic as produced by Erfidan et al. [**inverter**]

$$V_{DC} = 600 \tag{3.8}$$

$$A_{DC} = \frac{A_{AC_{rms}} V_{AC_{rms}}}{\eta_{inv}(A_{AC_{rms}} V_{AC_{rms}})} \tag{3.9}$$

$$\eta_{inv}(A_{AC_{rms}} V_{AC_{rms}}) = 1DInterpolate\, f(P_{min}, \eta_{P_{min}}, P_{\eta_{max}}, \eta_{max}, P_{max}, \eta_{P_{max}}) \tag{3.10}$$

The η_{inv} value is computed by cubic interpolation of the efficiency curve given the component specifications on $P_{min}, \eta_{P_{min}}, P_{\eta_{max}}, \eta_{max}, P_{max}$ and $\eta_{P_{max}}$.

3.7.8. Power Distribution Unit

For heavy-duty vehicle applications, the effective distribution of power is crucial for ensuring reliable operation of diverse electrical components and systems. This necessitates the use of a Power Distribution Unit (PDU), a pivotal component designed to manage multiple power sources and maintain a stable supply to critical systems throughout the vehicle.

The PDU serves as a centralized hub where various power inputs, often from batteries and other sources, are consolidated onto a constant voltage busbar. This configuration allows for seamless distribution of direct current (DC) power across the vehicle's electrical infrastructure, catering to the demanding requirements of electric machines, auxiliary systems, and onboard electronics.

Redundancy in power supply is a key feature of modern PDUs in heavy-duty vehicles. By integrating redundant pathways and components, PDUs ensure uninterrupted power delivery even in the event of a component failure or operational anomaly. This design approach enhances system reliability and uptime, crucial factors in the operation of commercial and industrial vehicles where downtime can lead to significant operational and economic losses.

A Power Distribution Unit is modelled as a passive electrical component in this research, where there should be an energy balance, meaning that all input power should be equally distributed over the output power paths. For this, Equation 3.12 is used.

$$V_{DC} = 600 \tag{3.11}$$

$$A_{DC_i} = \begin{cases} \frac{\sum A_{DC_o}}{\eta_{PDU}\, nr_i} & \text{if } \sum A_{DC_o} > 0.0 \\ \frac{\sum A_{DC_o}\, \eta_{PDU}}{nr_i} & \text{if } \sum A_{DC_o} \leq 0.0 \end{cases} \tag{3.12}$$

3.7.9. Battery

Lastly, we model the electrical energy provider, or source block, known as the battery. In current mobility applications, lithium-ion batteries are predominantly used due to their high power and energy density, which are essential for these applications. For heavy-duty applications, lithium-ion battery

packs typically have capacities in the order of hundreds of kilowatt-hours (kWh) to meet the high power demands. Additionally, lithium-ion battery packs are generally limited to a 1C charge rate to ensure long cycle life.

Extensive research has been conducted on modeling the behavior of lithium-ion batteries under load, utilizing low-, mid-, and high-fidelity codes [8] [10]. Due to the relatively flat voltage/SoC curve of standard lithium-ion battery cells under a 1C load, the battery voltage is assumed to remain constant within the 10% to 90% SoC range, which is the typical operating range. Consequently, battery depletion is simply a function of the current drawn from the battery [7]. In this study, the change in internal resistance, which typically depends on the Open-Circuit Voltage (OCV) and the (dis)charge rate (C-rating), is assumed to remain constant.

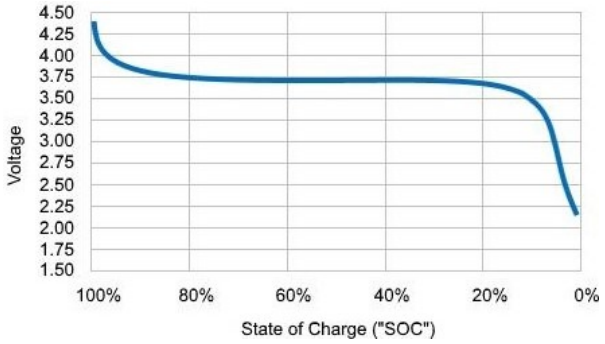


Figure 3.14: Typical Lithium-ion Discharge Curve

$$V = V_{OCV} - R_{int} * I_o \tag{3.13}$$

$$P_o = V\, I_o = V_{OCV} I_o - R_{int} I_o^2 \tag{3.14}$$

$$0 = I_o^2 R_{int} - V_{OCV} I_o + P_o \tag{3.15}$$

$$I_o = \frac{V_{OCV} \pm \sqrt{V_{OCV}^2 - 4\, R_{int}\, P_o}}{2\, R_{int}} \tag{3.16}$$

$$\dot{SoC} = -\frac{I_o}{Q} \tag{3.17}$$

$$= -\frac{V_{OCV} - \sqrt{V_{OCV}^2 - 4\, R_{int}\, P_o}}{2\, R_{int} Q} \tag{3.18}$$

For this low-fidelity code, the Rint battery model is taken, where a battery is modelled as voltage source in series with a resistance R_{int} [5].

3.7.10. Component Model Constants

In this section, we provide the assumed (and constant) variables used in the component models. These values were derived from engineering estimates, discussions with heavy-duty system integrators, and relevant literature (citations provided).

Table 3.2: List of Constants Assumed in This Research

Component	Constant	Value
Vehicle	M_{curb}	44000 kg
	$M_{vehicle\,less\,drive\,train}$	22500 kg
	$A_{vehicle}$	9.6 m^2
	g	9.81 m/s^2
	ρ	1.225 kg/m^3
	C_r	0.008
	C_d	0.646
Clutch	η_{CLTCH}	0.95
	M_{CLTCH}	25 kg [4]
Reducer	η_{RED}	0.92
	M_{RED}	185 kg [5]
Planetary Gearbox	η_{PGB}	1
	M_{PGB}	100 kg
Torque Coupler	η_{TC}	0.95
	M_{TC}	183 kg [6]
Multi-speed Gearbox	η_{MSG}	0.95
	M_{MSG}	250 kg
Inverter	$V_{AC_{rms}}$	500 V
Battery	V_{DC}	600 V
	R_{int}	0.05 Ω
	Depth of Discharge	80%
	SoC_{min}	10%
	SoC_{max}	90%

[4] `https://www.eaton.com/us/en-us/products/clutches-brakes/commercial-vehicle-clutches.html`, accessed on 26/06/2024

[5] `https://nadascientific.com/default/heavy-duty-truck-differential-with-double-reducer-unit.html`, accessed on 26/06/2024

[6] `https://www.optimadrives.de/c_split-shaft-pto-ssu-splitgearbox_58_en.html`, accessed on 26/06/2024

3.8. Framework Verification

In engineering research, verifying the applied methods and models is essential. By employing various verification techniques on these methods and models, confidence in the systems output can be increased. While it's impractical to detail every verification step performed, a select number of verification steps are highlighted here. However, every component of this framework has undergone verification at some stage.

The primary focus of verification in this research is on the component models used within the optimal control solver, as well as the solver itself. The mass estimation tool, which simply aggregates component weights, requires no additional verification. System complexity analysis tools have been rigorously tested on simple cases, to validate the number of valid system architectures. In this research, verification of various analysis tools has been conducted step by step, establishing a solid basis for increasing the confidence in the optimization outputs of the system architecture.

3.8.1. Component Model Verification

The fundamental building blocks of the optimal control solver are the various electrical or mechanical component models within the system architecture. Ensuring the proper functioning of these models is crucial to validate the final output. This section focuses on verifying selected component models to demonstrate their correct implementation.

Vehicle Model Verification

To verify the correct implementation of the vehicle model, a vehicle velocity and α-sweep analysis was conducted. This analysis resulted in Figure 3.15, illustrating the rolling resistance force and aerodynamic resistance force.

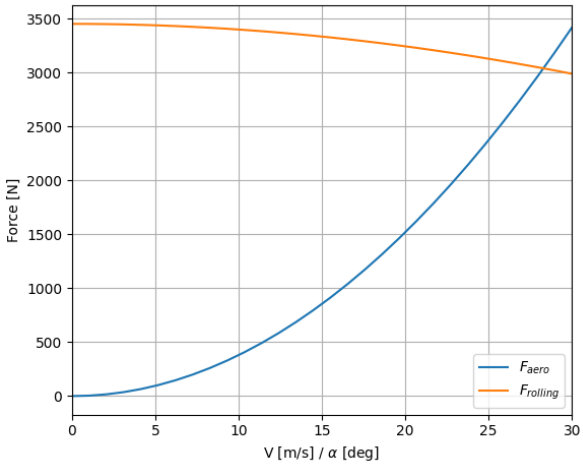


Figure 3.15: Aerodynamic and rolling resistive forces acting on the vehicle

This clearly illustrates the quadratic nature of the aerodynamic force and the nonlinear behavior of the rolling resistance. These forces are proportional to the square of the vehicle's velocity and the cosine of the inclination angle, as described in the Vehicle model.

Torque Coupler Model Verification

A torque coupler is a type of mechanical coupler also known as a Power Take-Off (PTO). Torque couplers are used to redistribute torque among multiple outputs. Unlike planetary gearboxes, which primarily adjust speed while maintaining constant torque across outputs, torque couplers maintain constant RPM across outputs while enabling independent torque control per shaft [4, 3]. The operational principle of torque couplers contrasts with that of planetary gearboxes, which are designed as speed couplers allowing for independent speed control of output shafts while maintaining constant torque. Torque couplers, on the other hand, distribute torque according to the following relationships:

$$T_3 = k_1 T_1 + k_2 T_2 \tag{3.19}$$

$$\omega_3 = \frac{\omega_1}{k_1} = \frac{\omega_2}{k_2} \tag{3.20}$$

Here, T_1 and T_2 represent the input torques, T_3 is the resultant output torque, ω_1 and ω_2 are the input angular velocities, ω_3 is the output angular velocity, and k_1 and k_2 are the gear ratios determining the final torque distribution.

When $k_1 = k_2 = 1$, the torque coupler simply sums the input torques T_1 and T_2 to produce the output torque T_3. Torque couplers operate as passive systems and lack the ability to selectively fix shafts as planetary gearboxes can [3].

3.8.2. SATG Verification

Next, a verification study was conducted to validate the functionality of the System Architecture Topology Generator. This involved evaluating a range of discrete design vectors to ensure the generation of not only the correct valid system architectures but also the accurate count of identified valid system architectures. Initially, the verification focused on a specific design vector, detailed in Equation 3.21, to assess both the quantity and specific configuration of valid system architectures.

$$\begin{aligned} \mathbf{x}_{discrete} &= [1,0,0,0,0,1,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1] \\ &= [BAT, EM, Fuel, INV, Wheels] \end{aligned} \tag{3.21}$$

For this discrete design vector, the SATG fails to find a valid system architecture because of the *Fuel* component, which acts as a source with only a chemical energy output port. This output port cannot connect to any chemical energy input port, resulting in no valid system architecture being identified.

Upon deeper investigation, it was found that the assertions made regarding the size of the valid design space, held true. The total size of the set of valid system architectures is determined by the product of the sizes of individual energy type system architectures (Equation 3.22).

$$\mathcal{O}(\textit{Size}(\mathbf{G}_{feasible})) = \prod_{i=0}^{n} \mathcal{O}(\textit{Size}(\mathbf{G}_{feasible,i})) \tag{3.22}$$

where n is the number of energy types considered in the analysis.

Applying this principle to the component set as seen in Figure 3.16, it was observed that:

- The valid system architecture set for *Electrical DC* was of size 2
  - $\mathcal{O}(\textit{Set}(\mathbf{G}_{feasible,ElectricalDC})$ = 2
- The valid system architecture set for *Mechanical Rot* was of size 26
  - $\mathcal{O}(\textit{Set}(\mathbf{G}_{feasible,MechanicalRot})$ = 26
- The valid system architecture set for *Mechanical Trans* was of size 1
  - $\mathcal{O}(\textit{Set}(\mathbf{G}_{feasible,MechanicalTrans})$ = 1

Using Equation 3.22, the total number of valid system architectures for the component set as depicted in Figure 3.16 was calculated to be 52, which was consistent with the results obtained from the SATG.

3.8.3. Optimal Control Solver Verification

To verify the objective value of the optimal control solver, specifically $\Delta\text{SoC}_{optimal}$, a bottom-up approach is adopted. Now that the individual component models are verified, the aggregated performance of the entire drive train under a specified drive cycle needs verification. Finally, the functionality and accuracy of the optimal control solver, employing Dijkstra's algorithms, are thoroughly verified.

A verification study was conducted for the system architecture depicted in Figure 3.16. An initial drive train configuration was established with the following component design variables:

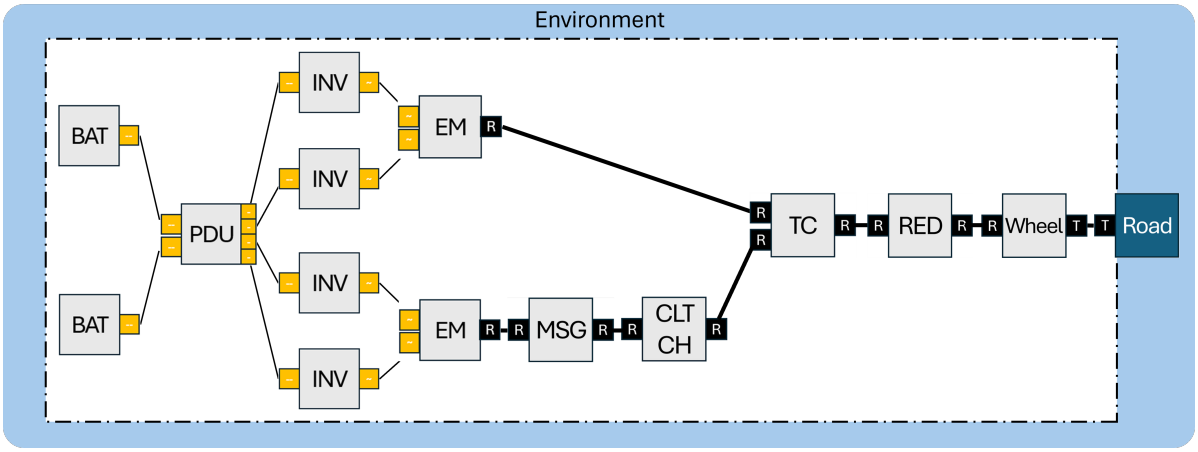


Figure 3.16: Valid system architecture used for verification

$$
\begin{aligned}
\mathbf{x}_{\text{component}} &= [\mathbf{x}_{\text{BAT1}}, \mathbf{x}_{\text{BAT2}}, \mathbf{x}_{\text{INV1}}, \mathbf{x}_{\text{INV2}}, \mathbf{x}_{\text{INV3}}, \mathbf{x}_{\text{INV4}}, \mathbf{x}_{\text{EM1}}, \\
&\quad \mathbf{x}_{\text{EM2}}, \mathbf{x}_{\text{MSG}}, \mathbf{x}_{\text{TC}}, \mathbf{x}_{\text{RED}}, \mathbf{x}_{\text{Wheels}}] \\
&= [[x_{\text{Battery Type 1}}], [x_{\text{Battery Type 2}}], [x_{\text{Inverter Type 1}}], [x_{\text{Inverter Type 2}}], [x_{\text{Inverter Type 3}}], \\
&\quad [x_{\text{Inverter Type 4}}], [x_{\text{Electric MachineType 1}}], [x_{\text{Electric Machine Type 2}}], [x_{\text{MSG gearRatio 1}}, \\
&\quad x_{\text{MSG gearRatio 2}}, x_{\text{MSG gearRatio 3}}, x_{\text{MSG gearRatio 4}}, x_{\text{MSG gearRatio 5}}, \\
&\quad x_{\text{MSG gearRatio 6}}], [x_{\text{TC gearRatio 1}}, x_{\text{TC gearRatio 2}}], [x_{\text{RED gearRatio 1}}], \\
&\quad [x_{\text{Wheel Diameter 1}}]] \\
&= [[1], [1], [1], [1], [1], [1], [1], [1], [6.0, 4.9, 3.6, 2.8, 2.4, 1.75], \\
&\quad [1.0, 1.0], [7.0], [0.98]]
\end{aligned}
\tag{3.23}
$$

First, the system behavior under static conditions is verified by assessing the energy flow across various components, using a specified input vector $\mathbf{u} = [V, \alpha, V_{wind}, a]$ and state vector $\mathbf{x}_{state} =$ [clutchEngaged, gear, TSR]. Initially, the behavior is evaluated for an input vector $\mathbf{u} = [0, 0, 0, 0]$. Based on engineering principles, it is anticipated that no current will be drawn from the battery as the amount of translational energy is zero. The total (filled) adjacency matrix corresponding to this input vector u is shown in Table 3.3

Table 3.3: Filled adjacency matrix for $\mathbf{u} = [0, 0, 0, 0]$ and $\mathbf{x}_{state} = [1, 5, 0.55]$

BAT	0	0	[600, 0.0]	0	0	0	0	0	0	0	0	0	0	0	0
BAT	0	0	[600, 0.0]	0	0	0	0	0	0	0	0	0	0	0	0
PDU	0	0	0	[600, 0.0]	[600, 0.0]	[600, 0.0]	0	[600, 0.0]	0	0	0	0	0	0	0
INV	0	0	0	0	0	0	0	0	[500, 0.0]	0	0	0	0	0	0
INV	0	0	0	0	0	0	[500, 0.0]	0	0	0	0	0	0	0	0
INV	0	0	0	0	0	0	[500, 0.0]	0	0	0	0	0	0	0	0
EM	0	0	0	0	0	0	0	0	0	0	0	[152.11, 0.0]	0	0	0
INV	0	0	0	0	0	0	0	0	[500, 0.0]	0	0	0	0	0	0
EM	0	0	0	0	0	0	0	0	0	[74.85, 0.0]	0	0	0	0	0
CLTCH	0	0	0	0	0	0	0	0	0	0	[74.85, 0.0]	0	0	0	0
MSG	0	0	0	0	0	0	0	0	0	0	0	[124.45, 0.0]	0	0	0
TC	0	0	0	0	0	0	0	0	0	0	0	0	[262.73, 0.0]	0	0
RED	0	0	0	0	0	0	0	0	0	0	0	0	0	[1692.02, 0.0]	0
Wheels	0	0	0	0	0	0	0	0	0	0	0	0	0	0	[0, 0, 0, 0]
Road	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	BAT	**BAT**	**PDU**	**INV**	**INV**	**INV**	**EM**	**INV**	**EM**	**CLTCH**	**MSG**	**TC**	**RED**	**Wheels**	**Road**

This scenario illustrates the expected system behavior: although the torque required at the wheels is non-zero due to rolling resistance, no mechanical power is needed ($P \propto T[Nm] \cdot \omega[RPM]$). Consequently, the electric machines do not draw any current, indicated by zero ampere values. The analysis assumes constant AC and DC voltage, hence the non-zero values at the respective electrical components (see section 3.7).

Further investigation reveals that when applying a tractive input vector, $\mathbf{u} = [+V, 0, 0, +a]$, a negative ΔSoC is returned. Conversely, inputting a sufficiently high deceleration, $\mathbf{u} = [+V, 0, 0, –a]$, yields positive ΔSoC. Additionally, it was confirmed that in regenerative mode, where V remains constant and a is equal but opposite in sign, the absolute output of the analysis is lower due to inefficiencies and losses incurred during regeneration. Other cases for u and x_{state} and their respective output are shown in Table 3.4.

Table 3.4: Verification of the coupling of the various disciplines

State Vector	Input vector	ΔSoC$_i$	Verified
$\mathbf{x}_{state} = [1.0, 1, 0.5]$	$\mathbf{u}_1$ = [0,0,0,0]	[[0],[0]]	✓
	$\mathbf{u}_2$ = [5, 0, 0, -0.1]	[[0.000193, [0.000193]]	✓
	$\mathbf{u}_3$ = [5, 0, 0, 0.1]	[[-0.00377], [-0.00377]]	✓
$\mathbf{x}_{state} = [1.0, 5, 0.5]$	$\mathbf{u}_4$ = [14, 0, 0, -0.8]	[[None], [None]]	✓
	$\mathbf{u}_5$ = [14, 0, 0, -0.1]	[[0.000226], [0.000226]]	✓
	$\mathbf{u}_6$ = [14, 0, 0, 0.1]	[[-0.0084], [-0.0084]]	✓
$\mathbf{x}_{state} = [0.0, 5, 0.5]$	$\mathbf{u}_7$ = [14, 0, 0, -0.1]	[[None],[None]]	✓

Table 3.4 presents the analysis outputs for a range of state vectors $\mathbf{x}_{state}$ and input vectors $\mathbf{u}$. It highlights instances where, for example, equal but opposite acceleration input vectors ($\mathbf{u_2}/\mathbf{u_3}, \mathbf{u_5}/\mathbf{u_6}$), the drive train is not able to recuperate as much battery capacity in regenerative mode, due to system losses. The table also demonstrates cases where the electric machine cannot supply sufficient mechanical power in cases where input demans are too high, resulting in a *None* value for the ΔSoC$_i$. Furthermore, the impact of state variables is examined. For $\mathbf{u_7}$, it is visible that the clutch is disengaged ($x_{state,CLTCH} = 0.0$). This conflicts with the Torque Split Ratio (TSR), which distributes torque evenly between axles (in case $x_{state,TSR} = 0.5$). In the case of a disengaged clutch, the TSR is not 0.5 as expected but 1.0, leading to another *None* value being returned.

Now that the coupled analysis of various components has been confirmed to function correctly, the next step involves verifying the optimal control solver and Dijkstra's algorithm. Here, the drive train specified in Equation 3.23 is evaluated. The filled state-time performance matrix (J^*) can be seen below.

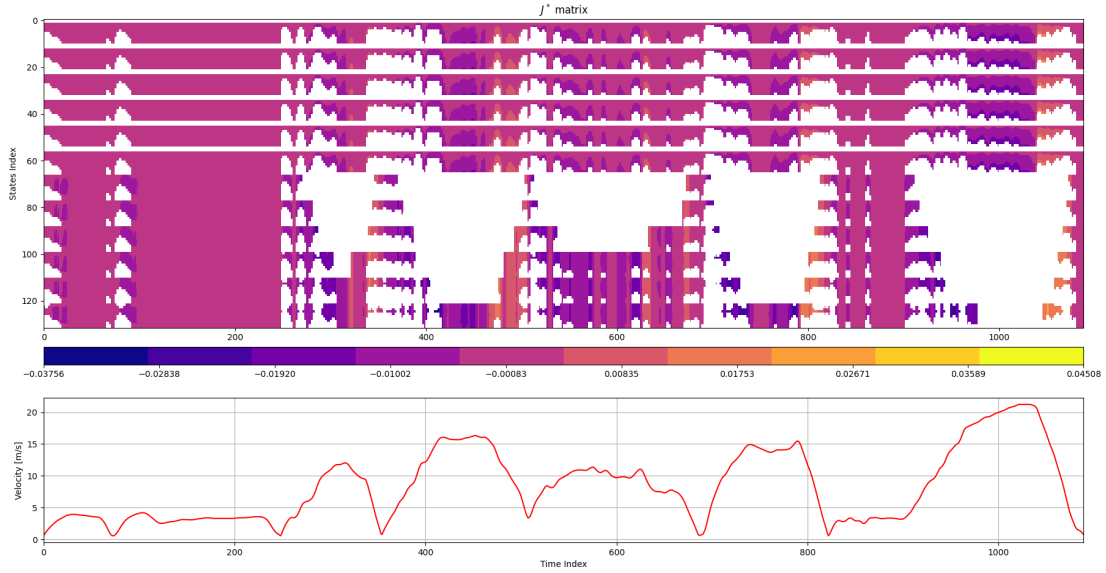


Figure 3.17: An example of a state-time performance matrix (**J***). Invalid states are shown in white. Negative values of ΔSoC$_i$ indicate battery discharge, whereas positive values signify regenerative braking (battery charging).

This matrix reveals the presence of invalid states, primarily occurring during periods of high torque demand for acceleration or recuperative torque during deceleration. These conditions exceed the power

handling capabilities at various RPM ranges, resulting in invalid states. Additionally, Figure 3.17 illustrates that the vehicle can efficiently recuperate energy during high deceleration phases. The optimal control solver subsequently determines the optimal control path for the various component states, achieving the most efficient drive train control. This optimal path is visualized in Figure 3.18. Based on these findings, it is concluded that the optimal control solver performs as intended and is successfully verified as:

- The optimal control solver identifies a path for which no **invalid** states are passed.
- The optimal control solver reveals implicit relationships between various operational states and the input vector $\mathbf{u}_i$. This capability allows it to discern logical trends such as higher gear selection at higher speeds and lower gear selection during periods of high acceleration and torque demand.
- The optimal control solver finds a global control strategy for the drive train across different initial states. This capability aligns with the expected operational principles of complex electromechanical drive trains.
- As expected, the battery depletion for both batteries is similar, due to their parallel connection to the Power Distribution Unit, where all power is equally redistributed.

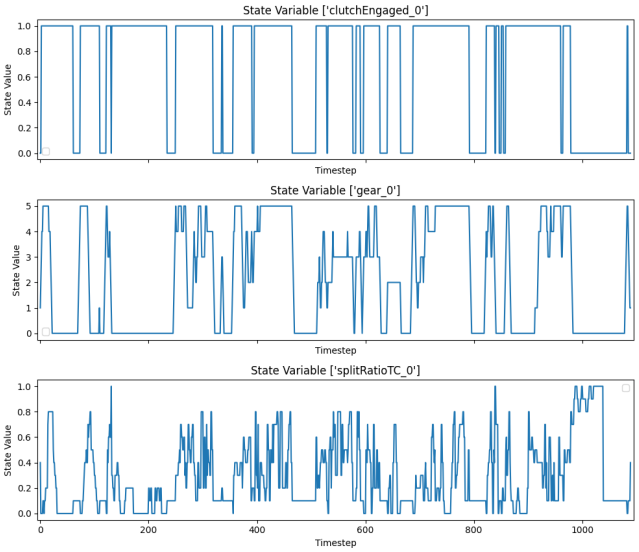


Figure 3.18: Optimal Control Strategy Per State Variable For Minimal ΔSoC

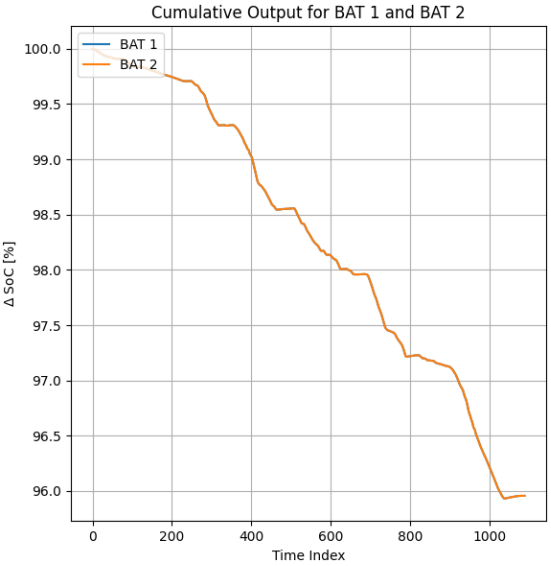


Figure 3.19: The Batteries SoC behavior for the optimal control strategy

Bibliography

[1] Dimitri Bettebghor et al. “Surrogate modeling approximation using a mixture of experts based on EM joint estimation”. In: *Structural and multidisciplinary optimization* 43 (2011), pp. 243–259.

[2] Pierre Bonneel et al. “Pyleecan: an open-source Python object-oriented software for the multiphysic design optimization of electrical machines”. In: *2018 XIII International Conference on Electrical Machines (ICEM)*. IEEE. 2018, pp. 948–954.

[3] Mehrdad Ehsani et al. *Modern electric, hybrid electric, and fuel cell vehicles*. CRC press, 2018.

[4] John G Hayes and G Abas Goodarzi. *Electric powertrain: energy systems, power electronics and drives for hybrid, electric and fuel cell vehicles*. John Wiley & Sons, 2018.

[5] V.H. Johnson. “Battery performance models in ADVISOR”. In: *Journal of Power Sources* 110.2 (2002), pp. 321–329. ISSN: 0378-7753.

[6] Steven Masfaraud et al. “Automatized gearbox architecture design exploration by exhaustive graph generation”. In: (July 2016).

[7] Jinyong Shangguan et al. “Robust design optimization of component parameters for DMDEB powertrain system based on Taguchi method”. In: *Journal of Dynamic Systems, Measurement, and Control* 144.9 (2022), p. 091003.

[8] Valentin Sulzer et al. “Python battery mathematical modelling (PyBaMM)”. In: *Journal of Open Research Software* 9.1 (2021).

[9] Xiaoxiong Wu et al. “Surrogate models for performance prediction of axial compressors using through-flow approach”. In: *Energies* 13.1 (2019), p. 169.

[10] Cheng Zhang et al. “Online estimation of battery equivalent circuit model parameters and state of charge using decoupled least squares technique”. In: *Energy* 142 (Oct. 2017).