A general framework for co-simulation-based optimization

Diego Alejandro Vega

UNIVERSITÉ DE LORRAINE
34 COURS LÉOPOLD — CS 25233
54052 NANCY CEDEX
TÉL : 03 54 50 54 00
CONTACT@UNIV-LORRAINE.FR
WWW.UNIV-LORRAINE.FR

UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

A general framework for co-simulation-based optimization

THÈSE

présentée et soutenue publiquement le 09 December 2025

pour l'obtention du

Doctorat de l'Université de Lorraine

(mention informatique)

par

Diego Alejandro Vega Vega

Composition du jury

Présidente :	Bernardetta ADDIS	Professeure, Université de Lorraine, Loria
Examinateurs :	Baya Lydia BOUDJELOUD-ASSALA	Professeure, Université de Lorraine, Loria
	Sébastien PICAULT	Chargé de recherche-HDR, INRAE
Rapporteurs :	Stéphane GALLAND	Professeur, Université de Technologie de Belfort-Montbéliard
	Paul-Antoine BISGAMBIGLIA	Maître de conférences-HDR, Université de Corse
Invité :	Elsy KADDOUM	Maîtresse de conférences-HDR, Université de Toulouse II
Directeur de thèse :	Vincent CHEVRIER	Professeur, Université de Lorraine, Loria

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503
Loria

Mis en page avec la classe thesul.

Acknowledgments

I would like to express my heartfelt gratitude to my supervisor, Vincent Chevrier, for his unwavering support and constant guidance throughout these four years of work together. His mentorship has been invaluable, helping me navigate the challenges of my PhD journey and building my research experience that I am sure will be a solid base for my future rigorous work in the research or industry field.

I also want to extend my sincere thanks to my research team for their warm welcome and the countless coffees shared. A special mention goes to Theo and Amaury, whose camaraderie made the day-to-day more enjoyable.

To my friends in both France and Colombia, your presence has been a cherished part of my experience. I particularly want to acknowledge two groups of friends I met in France. First, my international student group—William, Ana, Ilayda, Mash, Amaury, Argyris, and Arthur—who provided delightful memories with our evenings out in bars and restaurants, allowing us to unwind from our studies. Second, my closest friends, Ricardo de la Garza and Rodrigo Diaz-Caneja, who have become like brothers to me. Their friendship and support made my time in France truly enjoyable and helped me quickly feel at home.

I want to thank Sevra Çiçekli for her pivotal role in my research. Her encouragement has been a constant source of motivation, pushing me to strive for excellence in my work. Your insightful critiques challenged me to refine my ideas and strengthen my arguments, ultimately elevating the quality of my research. The constructive feedback you provided during our discussions was invaluable, helping me view my work from different perspectives. Your belief in my potential inspired me to overcome obstacles and remain focused on my goals. I am deeply grateful for both your academic insights and your personal support throughout this journey.

Lastly, I want to express my deep appreciation to my family in Colombia for their unwavering support. A special acknowledgment goes to my brother, David, whose guidance throughout my childhood was crucial to my success, and therefore, I consider this accomplishment to be a shared achievement for us. To my parents, I owe everything; this achievement is dedicated to them, as all I've and will accomplish in life is due to their love and support.

Dedico esta tesis a mis padres, a mi hermano y a Diego Felipe Becerra... descansa en paz mi hermano.

Contents

List of Figures

Chapter 1

Introduction

Contents

In this chapter, we introduce the thesis project with the motivation to study and improve complex systems, the definition of the research problem with the main objective of combining co-simulation and optimization in a framework, the contributions achieved during the thesis development, and the thesis manuscript structure.

1.1 Motivation

The growing presence of product complexity in the industry is expanding the need to understand and predict the behavior of systems while considering the internal complexity. The effective design of products and services requires a high-level analysis that involves the interactions between the environment and the internal mechanics of the system; the presence of heterogeneous mechanisms constitutes a complex system. A tool or method capable of studying a complex system can boost productivity in the design of products and services by providing increasingly comprehensive design test beds for research and development.

Co-simulation is presented as one way of studying complex systems throughout the virtual representation of heterogeneous entities interacting with each other at a local level, where, as a consequence, the global behavior emerges [7]. Furthermore, achieving an abstraction of the complex system offers the opportunity to interact with the model as a stakeholder; this means having the opportunity to test, validate, alter, or generally understand the model before a full system deployment [8], which could mean the optimization of some particular performance of the system. It is worth mentioning that the optimization we aim to consider is not related to the computation time efficiency, but the optimization of the parameters and structure of a co-simulation that represents a complex system. We refer to this combination as *Co-simulation-based optimization*, which consists of the use of co-simulation to construct a virtual model of a complex system [7], alongside optimization to identify configurations that either minimize or maximize a particular function [9]. The appeal of a virtual model that can simulate the behavior of the complex system is the testing possibilities at a low cost compared to real setup experiments.

From the time it became possible to analyze a random system using a computer program, scientists and engineers have always wanted to optimize systems using simulation models [10]. The constant progress in computational technologies pushes simulation models to improve and involve more components. In 1997, the most significant emerging technology within the simulation field was optimization [11, 12],

consequently, fields such as agriculture are profiting from the wide range of applicability of simulation-based optimization techniques in different domains and scales [13] as well as many widely different, albeit related, areas of operations research [10]. In the field of simulation, one of the ultimate goals is to find a combination of parameters that optimizes a key output performance [12]. Now, it is worth paying attention to the combination of the co-simulation and optimization technologies.

1.2 Research problem and objective

The research problem of this thesis is the study of complex systems, which have a behavior conditioned by the interactions of two or more internal systems. This study is particularly focused on the improvement of the system in terms of a performance indicator. The improvement approach makes the topic mainly interesting for the design of products and services in the industry.

In the literature, we find co-simulation as a method to study systems that present complexity, and optimization as a tool to find improvements to the system. The presence of these two concepts in the literature provides evidence of their relevance; however, in the literature, we find only ad hoc or domain-specific applications, which means that the field is lacking a systematic and domain-independent integration method for co-simulation and optimization.

This thesis is focused on capitalizing on the promising combination of co-simulation with optimization. Starting from an analysis of the existing ad hoc or domain-specific applications, to understand the architectures required to apply co-simulation-based optimization. We identify the patterns in the method applications to propose a generalized architecture for the method to be reused in any complex system problem. The conceptual basis of the co-simulation-based optimization method has some expected compatibilities with the already-known simulation-based optimization method, as well as new challenges to be considered coming from the complexity [5]. In general, this thesis aims to explore the relationships between co-simulation and optimization to propose a global decision support approach. The path of this thesis research follows three guiding questions: how is the implementation of co-simulation and optimization done in the literature?, is there a general structure used for co-simulation-based optimization?, and what architectural features are important to include in any implementation of co-simulation-based optimization?. With these questions, we start the process of methodology analysis of co-simulation-based optimization to identify general approaches, architectural patterns, and the advantages of promoting reusability in the methodology structure.

1.3 Contributions

The first contribution of this thesis is the development of a state-of-the-art review on co-simulation-based optimization. This state-of-the-art was motivated by the lack of it and the abundant presence of ad hoc applications in the literature. With this literature review, we were able to analyse the existing applications and perform a classification of the 96 applications found in terms of the disciplines involved, the optimization methods used, and the implementation architectures. The literature compilation facilitates the comparison of results from one method when applied across different domains, enabling us to identify architectural patterns that emerge when combining co-simulation and optimization, including modularity, domain flexibility, extensibility, interface, and pre-processing.

In the course of developing the literature review, we did a contribution to the stand-alone co-simulation field, we proposed a new classification focused on the way the co-simulation is built, which relies on the deductive or inductive reasoning approach to build the co-simulation (Top-down or Bottom-up); this classification sheds light on some characteristics of the complex systems that make them suitable for a co-simulation implementation.

The main contribution of this thesis is a new general framework for co-simulation-based optimization, providing a global decision support system for implementing the method to study and improve complex systems. The framework is built with the patterns found in the literature review; it is intended to guide future researchers in the process of improving the performance of a complex system. The definition

of the framework is accompanied by an implementation case where we develop a software framework that implements the proposed framework architecture; additionally, we develop use cases applied in the implementation case to showcase the usability of the framework to study and improve complex systems.

The final contribution is the assessment of the framework application, where we confront the results of the use cases to assess the performance of the framework to study and improve them, we also confront the performance of the software framework implementation in combining the co-simulation and optimization use cases; finally, we assess the impact of the framework guidelines proposed (modularity, domain flexibility, extensibility, interface, and pre-processing) in the positive and negative results of the framework implementation.

1.4 Thesis layout

The organization of this thesis is the following: we start with the definition of the main concepts of the research context in chapter 2. The basic understanding of concepts such as modeling, simulation, co-simulation, and optimization is the key to diving into the co-simulation-based optimization problem properly.

Next, a state-of-the-art review of the co-simulation-based optimization method was performed in chapter 3, where we explore a database of publications on co-simulation-based optimization. This review showcases the different domains involved in the domain and the optimization strategies being used, as well as the theoretical and practical ways to combine co-simulation with optimization. The result of the literature review is the identification of the recurrent patterns found in the implementations, which define the key elements to be considered when describing the general framework.

In chapter 4, we propose a conceptual general co-simulation-based optimization framework, which is constructed using the patterns found in the literature review. We describe components and dynamics to be used as guidelines in a co-simulation-based optimization framework implementation. Due to the abstract nature of the general framework, we develop a software implementation of the framework to serve as an example, and a concrete product of the framework to be tested with use cases; the idea is to use this software framework implementation to assess its expected benefits.

Chapter 5 describes the assessment process of the conceptual framework proposed in chapter 4. This starts with the definition of three complex systems to be implemented in the co-simulation component of the framework (use cases), then the definition of three optimization algorithms to be implemented in the optimization component. Then we describe the usage of the interface component, which intends to simplify for the user the combination of the co-simulation and optimization components by generalizing the interactions and creating a plug-and-play structure for the creation of co-simulation-based optimization experiments. In general, the framework implementation intends to offer modularity, domain flexibility, extensibility, defined component interfaces, and standardized pre-processing information for the uniformity of experiment configuration.

Finally, the combination of the co-simulation case studies with the optimization methods using the framework produced insightful results in three different levels of development: application, implementation, and architectural level. The application level where we analyse the results of each experiment to conclude on the improvements to the co-simulation use cases. The implementation level, where we compare the experiment's results to conclude why some algorithm/co-simulation synergies are more effective than others. The architectural level where we can assess the presence and benefits of the modularity, domain flexibility, extensibility, interface, and pre-processing elements by confronting the usability of the implementation against the results of the applications. The conclusions of every level shed light on the effectiveness of the framework application (we conclude on the best configuration found for each use case in terms of the optimization objective function defined), the scope and limitations of the implementation (we conclude on the best optimization method for each co-simulation and the characteristics that make a co-simulation compatible with each algorithm), and the benefits of the architecture proposed (we conclude on the way that the co-simulation, optimization and interface components exploited the modularity, domain flexibility, extensibility, interface, and pre-processing during the implementation and application development).

Finally, chapter 6 presents the concluding remarks from this thesis. We recall the context and research

problem, then we highlight the contributions of the thesis work, the importance of each element proposed, and the perspectives on the field regarding the future work to be done in the field to keep exploring the possibilities of the co-simulation-based optimization method.

Chapter 2

Context

Contents

In this chapter, we describe the context of the research. We start with a description of complex systems and the methods used to study and improve them (Section 2.1). For the study of complex systems (Section 2.2), we start with the definition of modeling and simulation as an established method to study systems; however, to cover the particularities of the complex systems, we resort to the co-simulation method, which is a method oriented to complex systems. Next, we describe the optimization method (Section 2.3) as a method to improve systems, starting from its general definition and classification, which leads us to the black-box optimization category, capable of proposing improvements to models with unknown internal behavior. Now, we discuss the co-simulation-based optimization method as the main topic of this thesis in Section 2.4, where we describe the combined methodology to the extend that is available in the literature, some details of the composition and dynamics of the co-simulation-based optimization method are not clear yet, we make explicit these gaps to be later filled with the results of the thesis.

2.1 Introduction

The idea of this chapter is to provide a proper definition of the key concepts that form the foundations of the thesis results; for this, we divide the research problem into 3 parts: the study of complex systems (where we include the formal definitions of the methods dedicated to this task, particularly the co-simulation method), the improvement of complex systems (where we describe the optimization concept and categories that are relevant for complex systems), and finally, the combination of co-simulation and

optimization (where we find that the conceptual structure of this combined method is not yet fully defined).

In the context of this thesis, we start with the definition of a system as:

Definition 1. *(System.) The notion of a system is directly linked to the objectives of the study being carried out. Indeed, studying a system means asking questions. The very fact of asking questions defines what a system is. A system is, therefore, the set of elements involved in questioning [14]. In the study context, it is usually referred to as the Target system or the Source system.*

We refer to a source or target system as:

Definition 2. *(Target or source system.) The system that is the object of the study.*

The interest of our research is a particular type of target system called complex systems, which is defined as:

Definition 3. *(Complex system.) Complexity may lie in the structure of a system, but it may also lie in the eye of a beholder of that system. Even when a system is "inherently" simple, i.e., describable, in principle, in simple terms, an observer may fail to discover that simple description, and may be able to characterize the system only in a very complex way. [15]. We can describe a complex system as a set composed of a large number of interacting entities, whose overall behavior emerges from the interaction of the system's component entities [14, 16].*

2.2 Study of Complex systems

The study of a system as described in Definition 1 can be performed using empirical methods like field experiments, lab experiments, surveys, case studies, forecasting, modeling & simulation; or interpretive methods such as descriptive interpretation, action research, subjective argumentation, and role-playing [17].

We draw our attention to the study approach called modeling & simulation, which uses only computational resources to study the behavior of a system. However, to comply with the requirements of a complex system as described in Definition 3, the methods need to consider the complexity and multidisciplinarity of the system, which requires a tool capable of representing the hierarchy of details present in complex systems [18]. At this point, we shed light on the co-simulation as an interesting method to be used to study complex systems. The co-simulation concept is a direct extension of the modeling and simulation concepts; therefore, we need to describe them in order.

2.2.1 Modeling and Simulation (M&S)

Modeling and simulation (M&S) is a method for the study of systems to generate an **abstract representation** of a system (artificial or natural). This abstraction means that the components are simplified to respond to a question (could be to understand how the system works or to predict its behavior) [19]. We define a model as

Definition 4. *(Model.) For an observer B, an object A* is a model of an object A to the extent that B can use A* to answer questions that interest him about A [20]*

The Modeling relation between the system and the model is defined by the inner constitution of the system (structure) and its outer manifestation (behavior). These definitions come in the form of a set of instructions, rules, equations, or constraints that generate the output behavior of the system. In other words, we describe a model with state transition and output generation mechanisms to accept input trajectories and generate output trajectories depending on its current state setting [21].

To compare the observed behavior of the source system with the one defined in the model, we require the simulation relation, which exists between the model and a simulator.

Definition 5. *(Simulator.) Computational device for generating behavior of the model [1].*

A simulation is typically specified at a high level since it is a system that we design intentionally to be synthesized from well-understood components that can be off-the-shelf [21]. The idea is to obtain a simulation of the model A* that behaves similarly to the entity A from Definition 4; this process is illustrated in Figure 2.1.

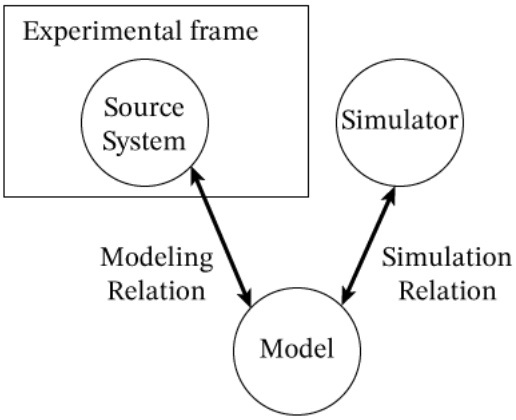


Figure 2.1: Basic entities and relationships in M&S (Taken from [1]).

The M&S process dynamic is iterative as seen in Figure 2.2, meaning it takes the shape of a loop. The method begins with the source or target system (Definition 2), where the modeling relation is used to create a model that will subsequently be simulated. Finally, an analysis stage consisting of comparing the values from the simulation to the observed values from the original target system; using this comparison, the necessary adjustments can be suggested until a validation is attained.

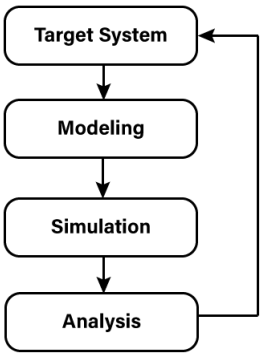


Figure 2.2: Simulation process scheme (Adapted from [1]).

To show the relevance of the objective of the study of a system, we propose examples 1 and 2.

Example 1. *(Study of a vehicle's engine.) To study a vehicle as the target system to understand the behavior of the engine, the M&S process should focus on elements such as the fuel characteristics, the motor specifications, the weight of the vehicle, and so on.*

Example 2. *(Study of a vehicle's external body.) However, if in the vehicle study we are interested in studying the aerodynamic performance of the vehicle's external body, we would focus the M&S on elements such as the shape of the external layer of the vehicle and how it is affecting the velocity when interacting with the wind at different velocities.*

Examples 1 and 2 show the simulation approach for the study of a system; however, sometimes the interest of the study is the emergent behavior from the interaction of two different domains; that is, if we want to study the impact of different vehicle engine configurations on the external body performance;

this kind of analysis calls for considering the system as a complex one. The study of complex systems by means of the M&S method sometimes is not enough to consider the emergent interactions between sub-system domains. The co-simulation method takes the M&S concept and extends it to include the complexity of the studied system.

2.2.2 Co-simulation

The Modeling and Simulation relations seen in 2.1 can be extended to fit in the context of complex systems. This extension considers the decomposition and coupling of sub-systems, which are shown in Figure 2.3. First, the modeling relation produces coupled models that we define as:

Definition 6. *(Multi-model or coupled models.) A multi-model is a modular model that subsumes multiple submodels that together represent the behavior of the model [22].*

The multi-model is then affected by the simulation relation, which generates the coupled simulations known under names like co-simulation, cosimulation, or coupled simulation.

Definition 7. *(Co-simulation.) The theory and techniques to enable global simulation of a coupled system/models via the composition of simulators [7].*

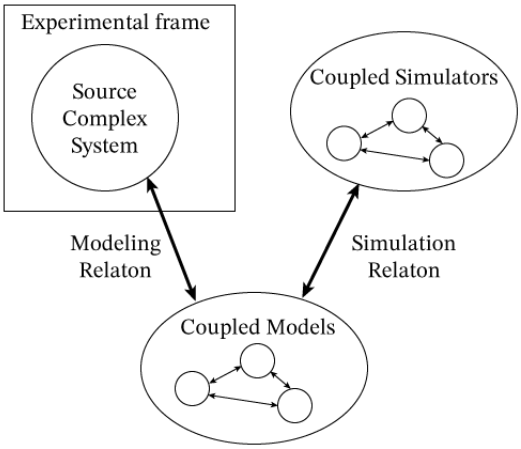


Figure 2.3: Basic entities and relationships in M&S of a complex system (extension of the structure in [1]).

This definition presents co-simulation as an extension of the simulation method that divides the target system into coupled sub-systems. It is a tool intended for abstracting complex systems into several interconnected sub-models.

An example of the process explained in Figure 2.3, where we can retake examples 1 and 2 that used the M&S method to study two isolated sections of a vehicle, we can consider a complex system example study using co-simulation as follows:

Example 3. *Let's consider the study of an electric vehicle based on the design of the car body and engine to ensure the best performance at high velocities. We assume that the size of the engine will directly affect the shape of the body, and the shape will affect the performance of the engine at high velocities. In these cases, we would require the M&S of both sub-systems to be correctly coupled to obtain the coupled simulations that represent the complex system under study. The resulting multi-model is capable of simulating the behavior of both systems and the way that modifications in one of them impact the other one.*

To illustrate the co-simulation method dynamics, we propose the scheme seen in Figure 2.4 as a general abstraction of the co-simulation process. A co-simulation is based on a complex target system

that is divided into sub-systems, each sub-system is a simulation that follows the same modeling and simulation conception described in Figure 2.2, then the simulations are coupled and co-simulated in a shared environment, where the interactions between the different models are described in terms of exchanges and synchronization data between models [7, 23], and finally, a global analysis to determine if the representation obtained is reliable; otherwise, the process can restart for adjustments.

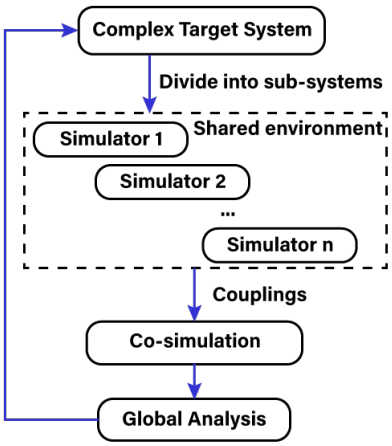


Figure 2.4: Proposed Co-simulation process scheme as an extension of 2.2, including the division of the complex system into sub-models that are later coupled and co-simulated.

Example 4. *Similar to Example 3, where we consider the study of a vehicle as a complex system, where the configuration of the engine directly affects the shape of the vehicle body. In this case, we can conduct a study on how the engine performance is affected by the vehicle's body shape. For this, we would develop individual simulations for the body dynamics and the engine mechanism that are coupled. The interesting feature of co-simulation is that we can develop the simulations separately in domain-specific software to be later coupled. The body shape dynamics can be modeled in a Fluid dynamics simulation software (treating the wind as a fluid that interacts with the vehicle) and the engine in a Mechanical physics simulator, which can be later coupled and run in a shared environment.*

In Definition 1, it was mentioned that a system is defined by the objectives of the study. In the case of the example 4, we established the objective as understanding the behavior of the engine's performance based on the dimensions of the engine and the aerodynamics of the vehicle's body. We acknowledge that a vehicle model contains additional sub-systems (such as the wheels, the driver, the path, etc.); however, they are not part of the study objective in the example and therefore could be ignored.

2.2.3 System Specification formalism

The description of a system is a key process in the creation of a model; there are different specification languages used to characterize systems.

Definition 8. *(Specification formalism.)The way a specification is done is called the system specification formalism, which in system theory distinguishes between the system structure and behavior [1].*

We describe the key elements of three formalisms that are particularly interesting for this research from their presence in simulation and optimization technologies, which are: DEVS, Coupled model DEVS, and the black-box approach. These formalisms are used later in the thesis in the implementation process, where we intend to exploit the ability of the formalism to provide a strong link between the conceptual rigour embedded in the formalism and a software architecture implementation.

Discrete Event System Specification (DEVS)

The Discrete Event System Specification (DEVS) is an approach to capture the structure of a system from both functional and physical points of view [24].

Definition 9. *(Atomic DEVS model.) The DEVS formalism represents the behavior of systems using a model M_i [1] with the following structure:*

$$M_i = \langle X_i, Y_i, S, \delta_{ext}, \delta_{int}, \lambda, ta \rangle, \tag{2.1}$$

where:

- $X_i = \{(p, v) | p \in IPorts_i, v \in V_{X_i}\}$ *is the set of input ports ($IPorts_i$), and its acceptable values (V_{X_i}).*
- $Y_i = \{(p, v) | p \in OPorts_i, v \in V_{Y_i}\}$ *is the set of output ports ($OPorts_i$), and its acceptable values (V_{Y_i}).*
- S *is the set of states of the model.*
- $\delta_{ext} : Q \times X_i \to S$ *is the external state transition function.*
  - $Q = \{(s, e) | s \in S, 0 \leq e \leq ta(s)\}$ *is the set of total states (a state from the set S associated to time e spent in that state).*
  - e *corresponds to the time past since the last state transition.*
- $\delta_{int} : S \to S$ *is the internal state transition function.*
- $\lambda : S \to Y_i$ *is the output function.*
- $ta : S \to \Re_0^+ \cup \infty$ *is the time advance function.*

In definition 9, we describe the general structure of a model under the DEVS formalism, where we use the term **atomic model** to describe the basic form of a model. This formalism is used for the implementation of the modeling and simulation concepts from subsection 2.2.1.

DEVS for Coupled models

The DEVS formalism also counts with an extension that covers the co-simulation concept from subsection 2.2.2, that is, the coupled DEVS models formalism defined as:

Definition 10. *(DEVS Coupled model.) In [25, 26, 27], the DEVS formalism is used to represent the structure of a coupled model N as:*

$$N = \langle X, Y, D, \{M_d | d \in D\}, EIC, EOC, IC, Select \rangle, \tag{2.2}$$

where:

- $X = \{(p, v) || p \in IPorts, v \in X_p\}$ *is the set of inputs, i.e., the couple (port, input value).*
- $Y = \{(p, v) || p \in OPorts, v \in Y_p\}$ *is the set of outputs, i.e., the couple (port, output value).*
- $\{M_d | d \in D\}$ *is the set of DEVS sub-models (which can be an atomic model as defined in 9 or another DEVS coupled model).*
- $EIC = \{((N, ip_N), (d, ip_d)) || ip_N \in IPorts, d \in D, ip_d \in IPorts_d\}$ *(External Input Coupling) is the set of connections between the coupled model inputs and the sub-model inputs.*
- $EOC = \{((d, op_d), (N, op_d)) || op_N \in OPorts, d \in D, op_d \in OPorts_d\}$ *(External Output Coupling) is the set of connections between the coupled model outputs and the sub-model outputs.*

- $IC = \{((a, op_a), (b, ip_b)) | a, b \in D, op_a \in OPorts_a, ip_b \in IPorts_b\}$ *(Internal Couplings) is the set of internal connections between the sub-model.*
- $Select : 2^D \rightarrow D$ *is a function to solve conflicts where two models want to execute their internal event at the same time; this function dictates the order of execution.*

The interest of DEVS lies in the advantages that this approach brings to the M&S process, such as scalability, because a DEVS model can be composed either of an atomic model or a coupled model. The combination of DEVS with object-oriented architectures provides a sound M&S foundation upon which to build a comprehensive and collaborative DEVS modeling and simulation environment [24].

For this thesis, the DEVS and Coupled models DEVS formalisms are key elements for the implementation process; many simulation software are based on programming languages with object-oriented structures that implement the DEVS structure.

Black-box approach

The external behavior of a system is the relationship it imposes between the input and output information. The internal structure of a system includes its state, state transition mechanism, and the state-to-output mapping (following the DEVS formalism). The process of inferring the structure of a system based on its behavior is not univalent; indeed, discovering a valid representation of a reference behavior is one of the key contents of the M&S enterprise [1], however, when researchers want to test software development of others, without knowing the program code or being able to peek inside [28], it is necessary to assess the code or model based only on its external manifestation, disregarding the internal mechanisms.

The concept of the black box is defined as

Definition 11. *(Black-box approach.) Hardware and software components that show an interface so that programming can be conducted based on knowledge of the operation of the interface, but not the functioning of the components as such [28].*

The idea is to have a module or "box" with internal unknown mechanisms and states, and the only way to interact with this box is by providing information (input), asking the module to process that information, and getting the results (output). The black-box approach is of great interest in the development of this thesis, as it offers a decomposing and coupling perspective to the study of a system; this means that we can encapsulate in black-boxes the different components of the method in a shared environment.

We introduce the graphic notation that we will follow for the representation of systems (or models) using the DEVS formalism (Definition 9) from a black-box perspective as seen in Figure 2.5. The idea is that from the elements of the DEVS model, we only focus on X_i and Y_i.

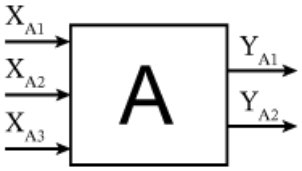


Figure 2.5: Black-box/DEVS Atomic model.

Additionally, the representation of a N coupled model can remain from a black-box perspective (i.e., only visibility of the X_i, Y_i elements), as seen in Figure 2.6, or we can see the internal structure of the same N coupled model as shown in Figure 2.7, where we include the elements X_i, Y_i, EIC (in blue), EOC (in red), and IC (in green).

2.2.4 Perspectives

The modeling and simulation method, alongside co-simulation, are concepts capable of ensuring the proper study of a complex system from a theoretical point of view. From the practical point of view,

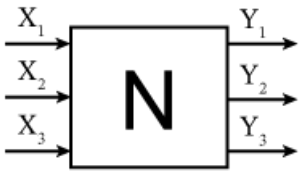


Figure 2.6: Black-box/DEVS Coupled model.

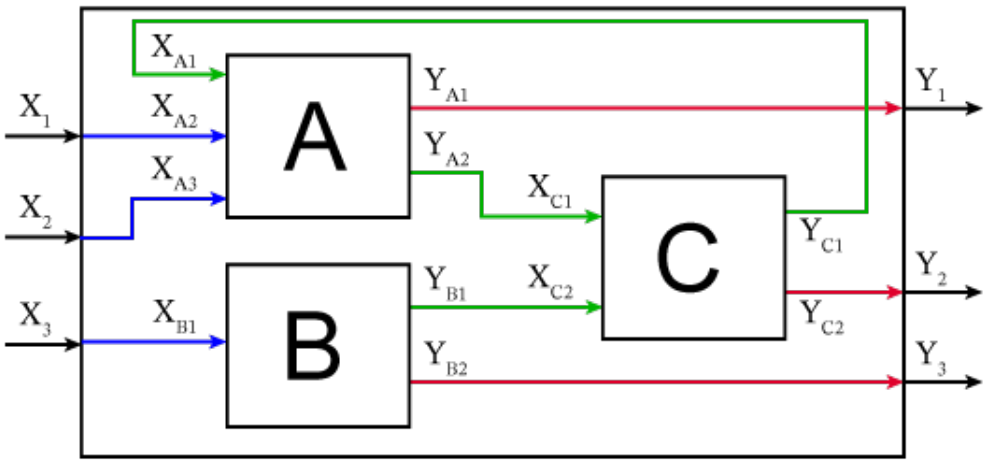


Figure 2.7: Internal structure of a DEVS Coupled model (includes sub-model *A* from Figure 2.5).

the specification formalism serves as a powerful tool to implement the concepts without compromising on the theoretical features. With an appropriate system description, we can now focus on the methods to improve the system performance.

2.3 Improvement of complex systems

With a validated co-simulation of the system under study, we can test configurations by tuning parameters or coupling patterns, which means we can find a combination of parameters that optimizes a key output performance. This makes the optimization concept an interesting next step for the improvement of complex systems.

2.3.1 Optimization

Definition 12. *(Optimization.) Consists of finding a value that minimizes or maximizes some specific function among all possible values (in a defined domain) that satisfy some conditions or constraints [9].*

The general optimization process is shown in Figure 2.8. To begin, a target system is modeled, which for the optimization context means defining constraints, objective function, and the decision variables [2]. Then, the model output is analyzed using an optimization method to determine the ideal model configuration. Lastly, an analysis is conducted on the data to determine if the outcomes align with the model's expectations; if not, the process is repeated.

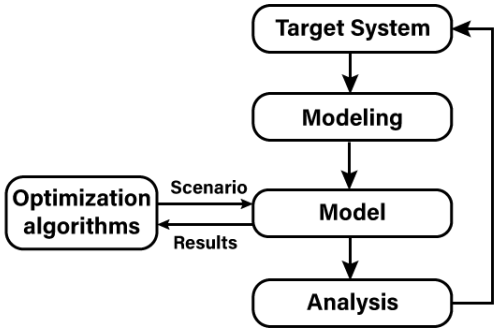


Figure 2.8: Optimization process scheme. Adapted from [2]

2.3.2 Types of optimization

There are two types of optimization problems: parametric and control [10]. Parametric optimization refers to problems that can be represented with the general Equation

$$\begin{aligned} &\text{min or max} \quad f(\vec{x}) \\ &\text{subject to:} \\ &g_i(\vec{x}) \leq 0, \quad i = 1, ..., n \\ &h_j(\vec{x}) = 0, \quad j = 1, ..., m. \end{aligned} \tag{2.3}$$

Where we are minimizing or maximizing an objective function $f(\vec{x})$, the vector $\vec{x}$ represents all the parameters or decision variables of the problem, the problem also has $n + m$ constraints represented by $g_i(\vec{x})$ denoting the inequality constraint functions and $h_j(\vec{x})$ denoting the equality constraint functions [2, 10].

Example 5. *Following Figure 2.8, we consider the waiting line at a bank as the target system, and using mathematical modeling, we come up with the mathematical model of the customer waiting time in line. Let's consider the optimization objective of minimizing the waiting time by adding the number of bank tellers available during the day. Using this information, a parametric optimization algorithm should be able to come up with the optimal number of tellers necessary to serve the bank's customers in the fastest way.*

In general, parametric optimization problems are known as static systems [29, 30], whereas control optimization problems handle dynamic systems as they change in some way from time to time; the objective is to choose a set of actions (also called policies) to move the system along a desirable path. The general Equation

$$\text{min or max} \quad f(\mu(1), \mu(2), ..., \mu(|\mathcal{S}|)) \tag{2.4}$$

is defined in [10] where $\mathcal{S}$ denotes the states of the system and $|\mathcal{S}|$ the number of states in the system, $\mu(i)$ refers to the action taken in state i, and finally the function f denotes the objective function to be optimized.

Example 6. *Let's consider the same bank line of example 5, however, we use a modeling process to create a control optimization model, which means that the model has states (number of people waiting in line), and different policies can be tested where, depending on the current state, a certain number of tellers will be open. For example, a policy could be from 1-4 clients in line, we open 1 teller, from 2-6 we open 2 tellers, and from 6-∞, we open 3 tellers [10].*

Control and parametric optimization is the main classification criterion; however, there are more ways to classify optimization methods, such as stochastic/non-stochastic, continuous/discrete, linear/non-linear, and heuristic/meta-heuristic [10]. Each optimization algorithm can fit into several categories, including the possibility of hybrid methods. This widespread categorization results from the large number of current optimization methods.

2.3.3 Black-box optimization

Definition 13. *(Black-box Optimization.) Black-box Optimization considers the design and analysis of algorithms for problems where the structure of an objective function* $f : \Omega \to \mathbb{R}$ *(with* $\mathbb{R} = \mathbb{R} \cup +\infty$*) and/or the constraints defining the set* Ω *is unknown, unexploitable, or non-existent [31].*

The black-box optimization method is used when the evaluation of the objective and/or constraint function involves an external and unknown function or process (usually used in problems where the evaluation depends on the execution of a computer code or simulation [31]). The general structure of the black-box optimization is shown in Figure 2.9, where we see that the optimization method (usually referred to as the algorithm) interacts with the unknown function in an iterative loop.

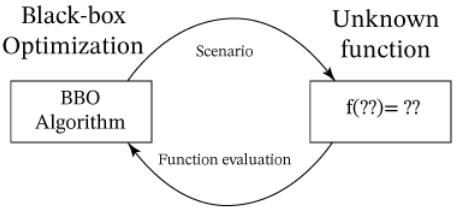


Figure 2.9: Illustration of the black-box optimization structure.

The black-box optimization approach is particularly interesting for simulation-based optimization [10]. For this application, an additional consideration is the possibility that the evaluation of the optimization model produces an invalid output, which is usually managed by flagging the result as $+\infty$ for minimization or $-\infty$ for maximization cases.

For the context of the study and improvement of complex systems, black-box optimization is appealing from its ability to blindly handle the improvement of a system; this is helpful to tackle the added complexity in the optimization process; however, up to this point, the impact of the complexity in the black-box optimization algorithm is not evident. We intend to explore the particularities of this interaction, which can lead to limitations to the optimization strategies and/or extend the reach of the optimization exploration process.

2.4 Combining co-simulation with optimization

We explore the conceptual combination of the co-simulation and optimization methods, which follows the same order as the definition of co-simulation in subsection 2.2.2, where we started with simulation to build the basis for the co-simulation description. In the same way, we start with the simulation-based optimization method to make the transition to the co-simulation-based optimization method.

2.4.1 Simulation-based optimization

The schemes presented in Figures 2.2 and 2.8 present a general methodology similarity since they study a target system by proposing a model to understand it or predict it, and then analyze and compare the results of the model with the target system to provide feedback and adjustments. This suggests a compatibility between simulation and optimization, which is a well-established method of collaboration known as simulation-based optimization. This method was considered the most important new simulation technology between 1997 and 2002 [32].

The black-box optimization was already defined in 13, and now we propose a definition of simulation as a black-box.

Definition 14. *(Simulation as a Black-box.) Following the definition of the black-box approach in 11, a simulation is described as a software element with an unknown internal mechanism, where the only way to interact is by using a scenario as inputs (parameters or policies) and an output performance (based on the optimization objective) [10].*

We incorporate both concepts in one structure using the black-box approach to describe and couple both components as shown in Figure 2.10. The typical areas in which simulation-based optimization is applied are large-scale, stochastic, and high-complexity problems [10]. However, considering the complexity of systems in simulations can now present a challenge for simulation-based optimization methods. This challenge is the result of going deeper into the description of a system, which means adding complexity to the modeling, simulation, and optimization processes, so the use of simulation-based optimization to study and improve the complex system is sometimes not sufficient. We find strategies to handle the complexity of a system, such as the inclusion of auto-adaption in multi-agent simulation systems [33, 34], the inclusion of an environment model in a multi-agent-based Simulation [35]; however, we follow another approach that handles the complexity by expanding the simulation reach to several coupled simulations.

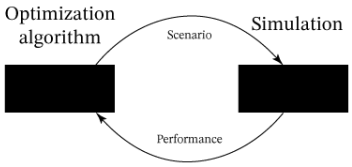


Figure 2.10: Illustration of the simulation-based optimization structure using the black-box approach.

The interest of this method is to use optimization to improve a system, which is achieved by using the M&S principles to create a simulation that can interact with an algorithm capable of finding a set of parameters or policies that optimize an objective function. We want to push forward the simulation-based optimization method into the complex systems research field.

2.4.2 Co-simulation-based optimization

Definition 15. *(Black-box co-simulation.)The black-box approach discussed in 11 can be used to describe the multi-model of a co-simulation using boxes and arrows. The boxes represent the sub-system simulations (using Definition 14), and the arrows are the couplings.*

In Definition 14, we already defined the interaction characteristics of a black-box simulation as the input scenario and the output performance indicator; however, in co-simulation, we have the opportunity to propose the definition of a scenario as:

Definition 16. *(Scenario.) In the context of a co-simulation that uses the black-box approach to describe its multi-model, a scenario for the co-simulation consists of a set of parameters or policies and the models to be used in the case of sub-model variations.*

We propose a general co-simulation black-box representation in Figure 2.11 that shows how several coupled sub-models create a bigger box representing the multi-model, and one of the models has an exchangeable variant model. The model exchangeability in co-simulation is the main feature that makes co-simulation-based optimization a novelty when compared with simulation-based optimization. This feature can be used by the black-box optimization method in the exploration of the search space.

Example 7. *Let's study the complex system of a vehicle design, where we want to configure the engine and the wheels to compare their performance with different types of pavement. In this case, we can modify key parameters from the engine, such as the number of cylinders (C) and maximum power (P). We also possess two simulations of wheels that come from two different providers ($Wheels_a, Wheels_b$). The idea is to test the co-simulation by trying different scenarios, which are defined by the parameters C, P, and the model $Wheels_?$.*

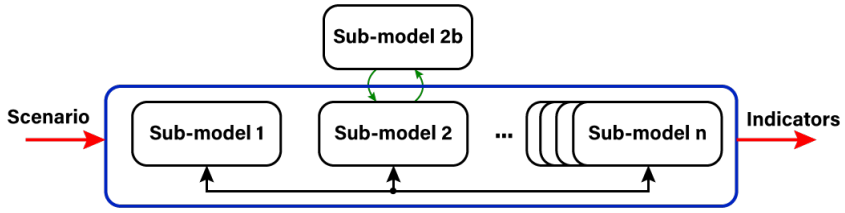


Figure 2.11: Proposed co-simulation black-box general representation.

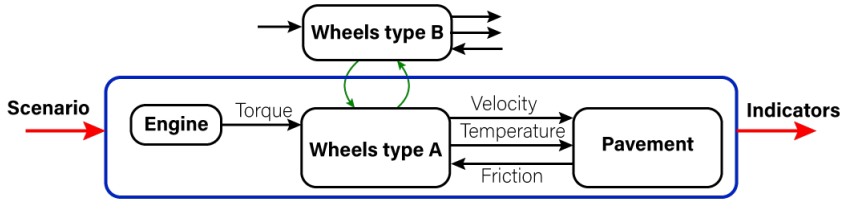


Figure 2.12: Multi-model of the Example 7, which illustrates the sub-model variation feature of the black-box co-simulation.

We recall Equation 2.2.3, where we discussed the DEVS Coupled model formalism as a representation tool for a multi-model. We clarify that for the model variation feature to work, the input/output structure should be the same between the varying models. To be more precise, following Definition 9, for two models M_1 and M_2, it is required that $X_1 = X_2$ and $Y_1 = Y_2$, that is, the set of input and output definitions is the same. In Figure 2.12, we represent the input/output structure with the four arrows connected to the Wheels models. The objective is to exchange the sub-models with no need for further intervention into the co-simulation, which simplifies the testing process done with the scenario method.

We propose a preliminary generic representation of the cooperation between co-simulation and optimization using the black-box approach in Figure 2.13. This representation combines the multi-model generalization from Figure 2.11 and the optimization implementation from Figure 2.10. Though the interactions between the components are similar to those used in simulation-based optimization, we consider that additional considerations have to be included to handle the complexity of the co-simulation. The bond between co-simulation and optimization is still not formally defined. This undefined intermediate component is responsible for translating the inputs and outputs of the co-simulation towards the algorithm and vice versa.

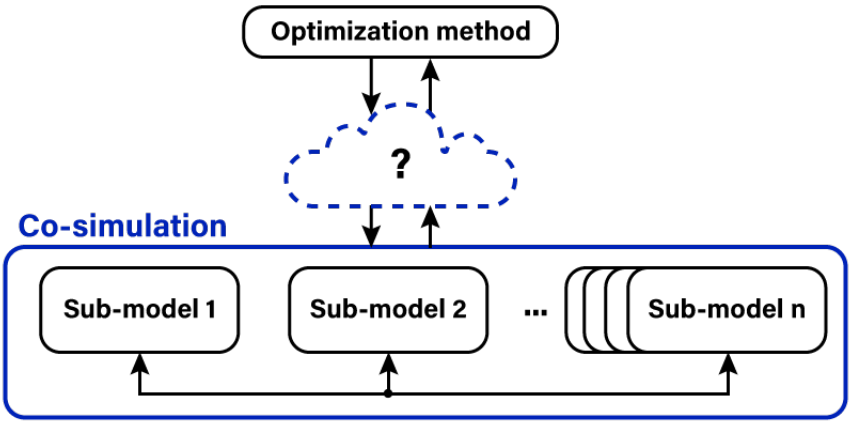


Figure 2.13: Proposed Co-simulation and optimization scheme, showing a preliminary proposal of the structure.

2.5 Challenges and conclusions

In the endeavor of studying complex systems, we highlight the pertinence of the co-simulation technique to comprehend and forecast the behavior of intricate mechanisms. Co-simulation offers the opportunity to compare the behavior of the system under different configurations or scenarios.

To further understand the complex system, we intend to improve the complex system in terms of an objective indicator; for this, we look at the simulation-based optimization technique. We consider the combination of co-simulation and optimization as a relevant method for studying and improving complex systems; however, a general structure of the method has not yet been established in the literature (Figure 2.13), which can lead to redundant frameworks being proposed in different domains.

In this chapter, we explored the definitions of co-simulation and optimization, as well as the formalisms usually used to link the conceptual definitions with the implementation structures.

In Figure 2.13, we gather in one structure the co-simulation, optimization, and intermediate components. The intermediate component is a crucial element to understand the appropriate combination of co-simulation and optimization in a general manner; however, this combination synergy is still lacking a generalization attempt. The definition of this combination synergy has the potential of expanding the existing capabilities of the simulation-based optimization method with the model variation feature from co-simulation.

Chapter 3

State-of-the-art

Contents

In this chapter, we make a literature review of the co-simulation-based optimization method, analyzing how existing papers are integrating co-simulation with optimization. We analyze how multidisciplinarity affects co-simulation-based optimization applications and the challenges that it presents. Then, an analysis of the optimization methods used in the applications and how they are deployed in the co-simulation environment is presented. Finally, we perform a discussion of contributions to the field and the remaining challenges to be solved.

3.1 Introduction

The co-simulation survey done by Gomes et al. [7] showed the growing interest and the advantages of co-simulation by bridging individual contributions, which improves future applications and leads to a deeper understanding of co-simulation. The same situation is currently happening on co-simulation-based optimization as a technique; there is a growing interest represented in the many individual contributions

that are not yet linked in a survey. Despite the growing interest, to the best of our knowledge, there is no state-of-the-art review focused on co-simulation-based optimization, which is causing several researchers to propose already existing approaches unknowingly. One example of this is found in the reviewed articles sample, where papers such as [36, 37, 38, 39, 40] talk about co-simulation, other papers such as [41, 42, 43, 44, 45] talk about coupled simulations, and finally the paper [46] talks of cosimulation, when the three concepts are referring to the same technique.

Even more examples can be found in the literature, where research studies are converging to the same structure described as co-simulation-based optimization in subsection 2.4.2, but using different names, such as:

- The nested and simultaneous solution for control design problems proposed in [47], where several optimization models are connected and combined into a **nested formulation** that creates a global optimization problem that is solved simultaneously. This method is combining the optimization of sub-models into a global and coupled complex optimization problem; however, this method has a stronger focus on the optimization topic, resulting in an optimization-oriented architecture.

- the OpenMDAO framework for multidisciplinary design, analysis, and optimization proposed in [48]. The framework proposed applies many of the concepts found in the co-simulation-based optimization structure, where different discipline models are combined into a **group model**, later connected to the **driver** module to perform the optimization, in the same way shown in Figure 2.10. The method in the article uses the same methodology as co-simulation-based optimization; however, it is referred to as multidisciplinary design optimization. Throughout the article, the words co-simulation or co-simulation are never mentioned; the concept of coupled simulation is mentioned once. This is the limitation that this methodology has; the nomenclature in the literature is scattered across different domains converging on the same concept. This is why we focus on a few concepts to get a general idea of the method, and we encourage the research community to converge on a uniform nomenclature.

- The resilience optimization applied in aligned and coupled multilevel decomposition strategies described in [49]. This method uses decomposition and coupling to divide a resilience optimization problem into the **design/operational model** and the **resilience model**, then an integrated structure is proposed to coordinate the optimization process. It is also discussed the possibility of a multi-level structure that allows the inclusion of additional resilience models into the formulation (this is a nesting approach compatible with the black-box approach).

- The distributed multi-agent coordination concept described in [50] has a similar structure to co-simulation. Its motivation comes from desensitizing the control of a multi-agent simulation, which benefits the physical constraints, especially in cases with limited resources and energy, short wireless communication ranges, narrow bandwidths, and large numbers of agents to be controlled. This comes at the cost of becoming far more complex in structure and organization.

These four examples of alternative nomenclatures complicate the paper compilation work; for this reason, we intend to take a sample of the methodology literature by covering the co-simulation-based optimization work where the authors use the terms **co-simulation**, **cosimulation**, or **coupled simulation** up to the year 2023 (this trio of names was already considered as a representative sample for the co-simulation field in [7]). This sample database represents a considerable part of the methodology and intends to unify the different nomenclatures of the methodology.

For the co-simulation-based optimization state-of-the-art survey, we used the same methodology as the one used for the co-simulation survey [7], by defining the equation:

$$\begin{aligned}\text{TITLE}(\ (\text{“co-simulation”}\ \text{OR}\ \text{“cosimulation”}\ \text{OR}\ \text{“coupled simulation”}\ )\ \text{AND}\ \text{“optimization”}\ )\\ \text{AND PUBYEAR} < 2024\ ).\end{aligned} \tag{3.1}$$

We use this equation on three different scientific databases: SCOPUS, Web of Science, and DBLP (we use three known scientific databases to ensure a broad coverage of the literature), which produced a total of 125 publications after checking for repeated publications, including journal articles, book chapters,

and conference proceedings. This sample is studied and reduced to 96 after excluding publications that are out of scope (i.e., those that, despite mentioning co-simulation-based optimization in the title, do not use this method as part of the scientific contribution) and those written in other languages (i.e, German or Chinese).

The study of the literature database is performed with the purpose of understanding how this combined approach is being implemented, as well as highlighting the recurring strategies, structures, and patterns found in the literature's implementations. This chapter starts with the general findings (section 3.2) from the general treatment of the database in terms of publication year and domains covered, including the finding of a new co-simulation classification. Then, we present the multidisciplinary analysis of the database in section 3.3 to understand the domain landscape, domain collaborations, and the domain-specific approaches for co-simulation-based optimization. Then, an optimization analysis is presented in section 3.4 to understand the optimization strategies landscape and implementation techniques. Finally, in section 3.5, we compile all the classifications into one to conduct the final discussion and conclusions on the literature review.

3.2 General findings

We perform a preliminary analysis of the sample database to observe the growing relevance over time. Figure 3.1 shows the publishing timeline. It illustrates the initial studies conducted on the subject between 2007 and 2009, followed by an increase in publications over time. The problem has been more relevant during the last ten years, particularly in the last three years, when the number has remained around 13. This trend demonstrates the value of the survey's findings for further research on the subject and the need to combine all of the individual efforts, particularly concerning methodology and standard operating procedures.

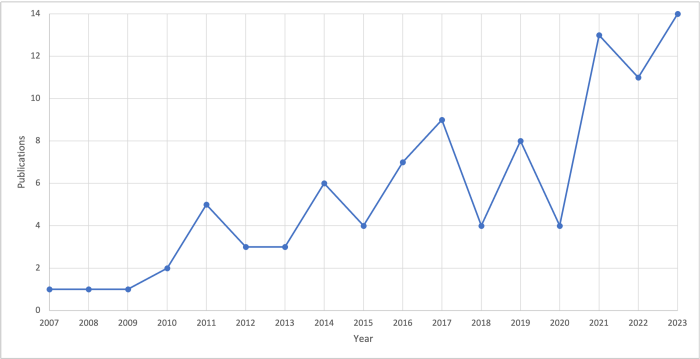


Figure 3.1: Publications about co-simulation-based optimization over the years until 2023. Combined results from SCOPUS, Web of Science, and DBLP using Equation 3.1 on February 2024.

3.2.1 Domain distribution

The domain distribution of the publications is another analysis that can provide general insights into the literature. We take Table 3.1 from the SCOPUS analysis tool, which shows the number of publications that involve each of the general domains as defined in the SCOPUS indexing (a publication may involve more than one domain). The table shows that the primary fields involved are engineering and computer science, and the applications are diverse, spanning from physics and mathematics to the natural and social sciences. Because of the wide range of applications throughout the research spectrum, co-simulation-based optimization methodology has the potential to benefit a wide range of areas. Given the multidisciplinary nature of the methodology examined in this survey, a more thorough examination of the domains and methods by which this technique is currently being applied is necessary.

Table 3.1: Number of publications per research field, using the field categories as defined in SCOPUS. Results from SCOPUS after manually removing papers out of the scope of this survey from equation 3.1 on February 2024, with a total of 85 publications.

Field	Articles
Engineering	53
Computer Science	34
Mathematics	13
Physics and Astronomy	12
Environmental Science	11
Materials Science	10
Energy	10
Agricultural and Biological Sciences	3
Chemical Engineering	3
Social Sciences	3
Earth and Planetary Sciences	2
Chemistry	2
Decision Sciences	1
Biochemistry, Genetics, and Molecular Biology	1

3.2.2 Co-simulation construction classification

In the course of reviewing the multiple applications of co-simulation-based optimization, we found that the way a co-simulation is constructed follows two possible approaches, which led us to propose a new construction classification for co-simulations. The classification follows the Bottom-up and Top-down approaches for the construction of a co-simulation from a practical point of view (complete classification of the 96 publications is available in Table 3.5 and Table 3.6).

Bottom-up approach

The bottom-up approach for a co-simulation construction uses the inductive reasoning method, which is defined as:

Definition 17. *(Inductive reasoning method.) The inductive approach moves from the specific to the general, so that particular instances are observed and then combined into a larger whole or general statement [51].*

When we apply this method to a complex system study, we reach the following definition:

Definition 18. *(Bottom-up approach for co-simulation construction.) Construction of a co-simulation through the coupling of two or more existing simulations of domain-specific systems that interact in a shared environment to produce a larger and more general behavior.*

To better understand the bottom-up approach in co-simulation construction, we present Example 8, which couples different models to build a big-picture complex model. This approach is interesting as several simulations can boost each other's accuracy with coupled interactions, which is the case of [42, 52, 43, 40]. Figure 3.2 provides a general illustration of the co-simulation bottom-up approach.

Example 8. *(Bottom-up approach to the co-simulation of a car.) Let's consider a car company that divides the design of a vehicle into branches (engine, wheels, and body), with each branch developing a simulation of the different components of the car for testing. However, now the company is interested in combining all the branch simulations into a complex vehicle co-simulation to evaluate the performance of the vehicle as a whole.*

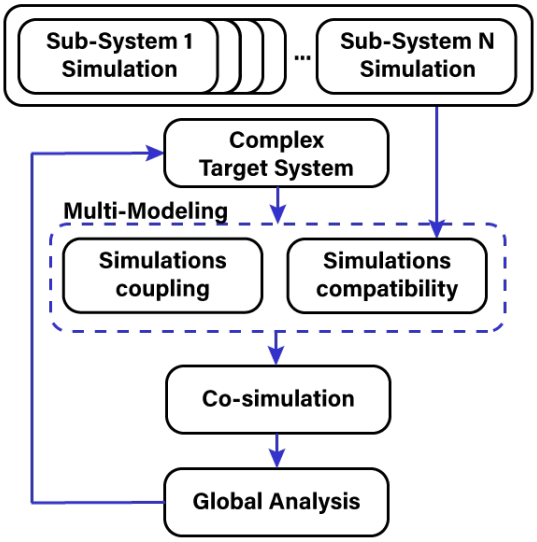


Figure 3.2: Proposed classification: Bottom-up approach for co-simulation. A target system is abstracted as a co-simulation, using pre-defined simulations coupled with each other and compared with the target system in a global analysis stage.

Top-down approach

In general, the top-down approach uses the deductive reasoning method, defined as

Definition 19. *(Deductive reasoning method.) The deductive approach is based on an earlier general theory to move to the specific, and therefore, it moves from the general theory to the specific [51].*

The application of the deductive reasoning method to co-simulation leads to the following definition:

Definition 20. *(Top-down approach for co-simulation construction.) Construction of a co-simulation that starts with the study of the complex system, using modeling and simulation to describe smaller and interconnected sub-systems.*

Figure 3.3 illustrates the general structure of this approach, which breaks down a complex target system into smaller sub-systems that are subsequently modeled and coupled using dedicated software. The resulting co-simulation is then evaluated and contrasted with the target system for validation. Example 9 showcases the approach described. We found articles in the literature database that use the top-down approach, such as [53, 41, 54, 55].

Example 9. *(Top-down approach for the design of a helicopter.) The co-simulation of a complex mechanical structure from a helicopter is co-simulated using the ANSYS software environment, where 3 different simulations are designed and coupled; each simulation describes the system under different domains: fluid, physical, and thermal mechanics [41].*

An important remark is that the use of either approach to construct a co-simulation produces the same result: a working co-simulation that aims to produce a similar behavior to that of the complex system under study. The complete classification of the literature database using these criteria is available in Table 3.5.

3.3 Multidisciplinary analysis

In this section, we present a review of the recurrent domains and software used for co-simulation-based optimization applications. We also highlight the recurrent interdisciplinary collaborations and the

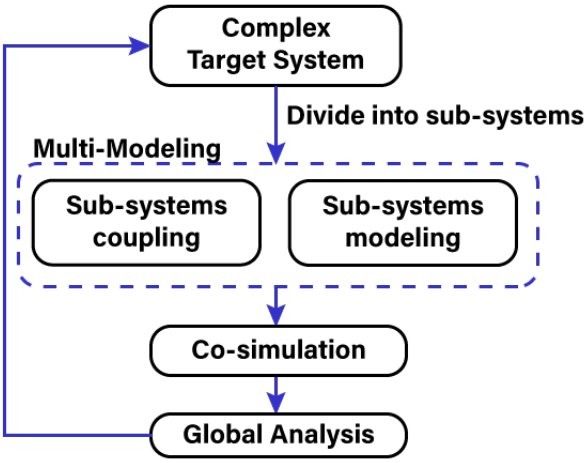


Figure 3.3: Proposed classification: Top-down approach for co-simulation. A target system is multi-modeled, i.e., it is divided into subsystems that are coupled and simulated. To provide feedback, this co-simulation is then compared to the target system using global analysis.

advantages of using co-simulation for optimizing complex multi-domain problems. An overview of the literature focused on the multidisciplinary aspects of a problem is performed; these applications usually propose a holistic representation of a system that can provide a better performance than other models. The emerging patterns found through gathering and studying the literature can lead to a conceptual architecture of the process, which is a comprehensive contribution to the field that can benefit future applications in the same area.

Example 10. *The design of a vehicle involves the interaction of its major sub-systems (engine, control, hydraulic suspension, and pressure), which involves different domain-specific dynamics like mechanics, logic, fluid, and thermal.*

The heterogeneous approach of co-simulation encourages multidisciplinary work, situation like the one proposed in example 10 can be found in the literature in [56]; this article presents a virtual prototype project proposed between companies to combine the sub-systems to evaluate the performance of the vehicle using a shared co-simulation environment to find the best configuration of the sub-systems that produces the desired vehicle performance (global optimization). To understand the multidisciplinary influence on the topic, we use the following methodology: first, we analyze the different domains interacting in the co-simulation-based optimization works in the literature. This domain-approach review highlights the synergies already present between domains and encourages new collaborations. Second, we perform a co-simulation software review to extend the analysis from this section to the software availability factor, which can influence the research path followed by each domain expert. Finally, we discuss the challenges in the multidisciplinary field of co-simulation-based optimization.

3.3.1 Domains in co-simulation

In contrast to the SCOPUS classification of domains, where the primary goal is to categorize the research fields (see Table 3.1), there are many simulation domains with different levels of detail ranging from specialized domain simulations (i.e. fluid dynamics simulation with FLUENT [42] or with COMSOL [57], numerical simulation with MATLAB [58]) to big systems domain simulations (i.e. building temperature control simulation with EnergyPlus [42, 59, 60, 61, 62, 63] or with TRNSYS [64, 65, 66, 4], vehicle simulation with CarSim [67]). As part of this survey, we provide a full classification in Table 3.5 of the simulation domains in the literature (third column) that compile a total of 49 different domains. This classification was done manually, taking into account the general purpose of the software used, as well as the document definition of the model created with the software. From these domains, we can

identify a particularly interesting one, the Computational Fluid Dynamics (CFD) [68], which is used on 21% of the articles reviewed in connection with other domains. In Table 3.2, we can see all articles from the sample database that use CFD simulations; the table shows the great variety of combinations present in terms of domains and software. Additionally, we include the integration method used to integrate the CFD simulation with the other domain software.

Table 3.2: Co-simulation collaborations with CFD by domain, simulation software, and integration method.

Article	Simulation domain	Software	Integration method
[41]	Fluid Solid	FLUENT Solidworks	Import CAD file into ANSYS
[54]	Fluid Electric	Spectre LT-Spice	Manual (coded in Verilog)
[42]	Fluid Building	FLUENT EnergyPlus	Manual
[52]	Fluid Control	FloMaster Simulink	FMU imported into Simulink
[43]	Fluid Solid	FLUENT Solidworks	Imported into ANSYS plugin
[37]	Fluid and electric	COMSOL	COMSOL framework
[38]	Fluid Mathematical	FLUENT Simulink	Imported using MATLAB plugin
[57]	Fluid Electrical Solid	COMSOL COMSOL COMSOL	COMSOL Framework
[69]	Fluid and solid	COMSOL	COMSOL framework
[70]	Fluid Mechanical	Gleeble Gleeble	Gleeble framework
[71]	Fluid Numerical	FLUENT Particle Flow Code	Manual
[72]	Solid Fluid Dynamic	ANSYS FLUENT EDEM	UDF library imported into ANSYS
[73]	Fluid Solid	FLUENT EDEM	UDF library imported into ANSYS
[74]	Solid Fluid	Pro ENGINEER FLUENT	Imported using ANSYS plugin
[75]	Fluid Solid	FLUENT EDEM	UDF library imported into ANSYS
[76]	Fluid Solid	FLUENT MATLAB	Imported using MpCCI toolset
[64]	Fluid Energy	FLUENT TRNSYS	Manual
[77]	Fluid Engine	FLUENT GT-Power	UDF library imported into ANSYS
[78]	Fluid Numerical	FrontFlow/Blue VSI Fortran	Manual
[29]	Fluid Electrochemical	AMPERES IPS	AMPERES-IPS Framework
[30]	Fluid Chemical	FLUENT Aspen Plus	APECS Framework

Table 3.2 shows that CFD simulations are widely combined with other domains such as solid (Computer-Aided Design for creating 2D or 3D virtual representations of real-world products), energy, numerical, building, and more. The diversity of domain combinations and integration methods in Table 3.2 is a

result of the complexity of the field of co-simulation. We present the different software to handle the integration of the unitary simulations from different domains, manual implementations, and the use of import/export tools to combine the simulations. The different methods highlight the methodological significance of the interoperability of the simulations, which is correlated with the software compatibility and the domains in question.

3.3.2 Interoperability within co-simulation

Interoperability, as described in [79], exists in six levels that go from low integrability to full conceptual interoperability: no compatibility, technical, syntactic, semantic, pragmatic, dynamic, and conceptual. For more details on the interoperability levels, see [25]. The interoperability concept has increased its relevance in the simulation fields to offer compatibility between software from different domains; in [80], the use of common standards is considered the best possible answer to integration and interoperability problems in co-simulation. The interoperability handles the interaction of two or more models created in different software without losing capabilities. To evaluate the current literature environment regarding co-simulation interoperability (sub-systems interaction), we identified 3 approaches within the literature reviewed, i.e., manual implementations, collaborating standards, and a co-simulation framework.

Manual implementations

This methodology covers applications in which the user manages the information transfer between the various simulation software programs, handling all aspects of the interoperability of the simulations [42, 81, 71, 82, 64, 78]. This implementation strategy solves the software interoperability issue for the specific study instance but does not offer a generic solution that can be applied to other comparable projects in the future, making it an ad hoc solution. Specific examples of manual implementations coupling using high-level programming languages to set up the co-simulation, such as ActiveX [83], Verilog [54], C++ [84], and communication protocols like Memory Mapped Files [85], or TCP/IP and DNP3 [86, 53, 87, 88].

The presence of manual implementations in the literature highlights the benefits of a fully manual interface between simulations, including control over the coupling mechanism in terms of input/output formats and event coordination. However, it also highlights the drawbacks of an ad hoc solution to the interoperability problem, which includes providing a scheduling algorithm and solving the information coherence, as well as the method's lack of reproducibility, which is of greater concern for the next two categories.

Collaborating Standards

This strategy is used by software companies to offer compatibility with other software that could improve the performance of a simulation. We identified two categories of collaboration standards: those that allow simulations to be imported in a standard format and those that offer software extensions such as toolboxes, plugins, and APIs. The collaboration approach follows the principles of the black-box approach as defined in 11, that is, software blocks that communicate with external software using a defined interface. A first type of collaboration involves simulation software that offers services to enable compatibility with other software, such as:

- SIMULINK plugin to include external software models [38, 89, 76, 90, 91].
- SIMAT interface, used to couple the SIMPACK and MATLAB models [92].
- TRNSYS library to include TRNFLOW models [66].
- ADAMS plugin to execute EDEM multi-model [93].
- CORBA-interface between MATLAB (master) and ANSYS (slave) [58].
- FEM package from COMSOL [94, 4].

- Python library to run MATLAB and OpenDSS [95].
- MATLAB feature to run MODFLOW models [96].
- ANSYS workbench tool to import Pro ENGINEER 3D design model [74].
- TRNYS library to couple a TRNYS model with MATLAB model [65].
- RecurDyn to include EDEM models [44].
- FLUENT plugin to import SolidWorks model [43].
- GT-power model toolbox to connect with MATLAB/Simulink [97].

These tools are an extension of the main software capabilities to provide a link with external software that handles the interoperability of the simulations. The second type of collaboration consists of simulation software that provides the capability of importing models created in external software using a standard format. In the literature, we find examples of domain-specific standards formats such as CAD files for 3D and 2D physical models [41, 43, 46], Materials Property Data exchange (.matml) [98], circuit simulation import using a Touchstone file standard [99] or the s-parameter format [100, 101]; we also find examples of non domain-specific formats like UDF libraries (.dll) [72, 73, 75, 102, 103, 104, 77], XML format [105], ASCII Files [106], and the Functional Mock-up Interface (FMI) standard [107] that generates ".fmu" files from any domain [52, 60, 62, 63, 80, 67].

The existence of multiple exportation standard applications and software compatibility tools is proof of the usefulness of co-simulation software. An interesting example is the exporting standard Functional Mock-up Interface (FMI).

Definition 21. *(Functional Mock-up Interface Standard.) The FMI standard is a free standard that defines a container and an interface to exchange dynamic simulation models using a combination of XML files, binaries, and C code [107].*

FMI is supported by 250+ tools and maintained as a Modelica Association Project [107]. The fact that the FMI standard is supported by many simulation software shows the relevance of this standard in the co-simulation-based optimization, especially if we consider that we found 17 programs that are both mentioned in the FMI website [107] and used in the applications from this state-of-the-art review. These programs are Dymola [108], Forge [109], Abaqus [110], Adams [111], CarSim [112], MATLAB/Simulink [113], EnergyPlus [114], Fluent [115], GT-Power [116], LS-DYNA [117], ModeFrontier [118], Model.CONNECT [119], RecurDyn [120], Amesim [121], Flomaster [122], SimPack [123], and TRNSYS [124].

Examples that we found include the Model.CONNECT [125, 119] software, which provides an easy-to-use interface for simulation model coupling and focuses on the configuration of the interaction of the models, while the definition of the individual models is imported from external software; other examples include OpenModelica [126], Simulink [113] and Adams [111] software, which combines native simulation capabilities with importation methods to combine simulations in a shared environment; this shared environment concept is especially exploited in the next category.

Software Frameworks

Definition 22. *(Software Framework.) Software frameworks are a form of software reuse that primarily promotes the reuse of entire architectures within a narrowly defined application domain. They propose an architecture that is optimized for all applications within this domain and makes its reuse across applications in the domain possible [127].*

The use of a software framework in co-simulation means the use of an architectures that encapsulate several domain-specific simulation software to coexist in one shared environment. This approach is not opposed to the previous exporting standards approach; on the contrary, it can use standards to expand its reach. The objective of these frameworks is to offer a platform to develop multi-disciplinary simulations that can be coupled easily. This approach establishes a standard for the coupling mechanisms, which means that the interoperability is automated within the framework, that is, the user does not need

Table 3.3: Co-simulation software in the literature.

Framework name	Domain covered	Used in	Proposed by
EXPPO	Cyber-Physical Systems	[128]	Article [128]
HELICS	Multi-domain	[129]	Article [130]
MATLAB	Multi-domain	[125]	Software product [131]
EV-EcoSim	Electric vehicule	[132]	Article [132]
ANSYS	Multi-domain	[133, 134, 135, 136, 137, 138]	Software product [139]
MODUS	Hardware/Software	[140, 141]	Article [141]
COMSOL	Multiphysics	[37, 57, 69]	Software product [142]
Cadence	Thermo-electric	[143]	Software product [144]
Amperes-IPS	3D Magnetic [145]	[29]	Article [29]
GUSMA	Vehicle design	[56]	Article [56]
COSMO	Mobility	[146, 147, 148, 3]	Article [149]
EPPY	Building design	[61]	Article [61]
LabView	Aerial vehicle design	[150]	Software product [151]
Rhino	Building design	[59]	Software product [152]
ADS	Circuit design	[153, 154]	Software product [155]
Gleeble	Thermomechanical	[70]	Software product [156]
COSCO	Fog computing	[39]	Article [39]
APECS	Fossil energy	[30]	Software product [157]
Not-named	Bio-fabrication	[158]	Software product [159]
Not named	Photonic and electronic	[160]	Software product [161]
Not named	Multi-domain	[162]	Article [162]
Not named	ROS-based [163]	[55]	Article [164]
Not named	OPAL-RT-based [165]	[166]	Article [166]
Not named	Robotics	[167]	Article [167]

to intervene further to ensure interoperability (for reference, in the manual implementation category, the user had to handle the entirety of the interoperability effort, and for the collaborating standards category, the user effort is reduced to following the existing collaborating standard process to ensure the interoperability, for example, installing a plugin into a software to import another software models). The advantage of co-simulation frameworks (also referred to as platform, environment, or architecture) is that they encourage reusability as they offer a set of tools to continue developing co-simulations for future researchers, which, at the same time, contributes to the validation of the framework. The examples found in the literature are presented in Table 3.3, where we classify each framework based on the domain covered, the articles that use the framework, and the source proposing the framework, which comes from published articles describing the characteristics of the framework or a software product offered in the market.

Table 3.3 shows that there are already 25 co-simulation frameworks available in the literature, each with a different target domain. The benefit of a framework over an ad hoc application is that its results serve as an incentive for the domain to continue working on the issue with the same framework in the future, thereby facilitating and encouraging progress. The table contains two cases of software that have already been used in four more applications in the literature sample database:

- ANSYS [139], an Engineering simulation software package used as a multi-domain co-simulation tool in six of the articles from the database.
- The CO-SiMulation Optimization (COSMO) [149], a layer-based architecture for the co-simulation of mobility problems, with four applications in the database articles.

This reproducibility demonstrates the value of a framework, as each iteration benefits from the co-simulation features and validates the utility of the architecture. The framework category is the most robust of the three categories this section defines, since it provides a common environment for user-friendly coupling mechanisms, opens the door for further applications following the same structure, and works towards an abstract method to perform co-simulation.

3.3.3 Challenges

In this section, we presented an analysis of the literature sample on integrating multiple domains into a co-simulation. This analysis shows the increasing presence of collaboration and frameworks in co-simulation software (Tables 3.2 and 3.3). These strategies use general patterns to couple two or more simulations that can belong to different domains.

Definition 23. *(Extensibility.) A framework is extensible when it can be easily extended to support new kinds of units with new kinds of capabilities [7].*

The co-simulation state-of-the-art [7] described extensibility (Defined in 23) as a key challenge for the co-simulation field. We find co-simulation frameworks such as APECS [30], Helics [129], and GUSMA [56] as examples of frameworks that include extensibility advantages. These frameworks integrate simulation tools supporting the import of external models and enabling the development of new compatibility modules for other software using high-level programming languages.

Therefore, we consider extensibility as a key challenge of the co-simulation software. A framework with extensibility can handle heterogeneous co-simulation, which is already found in co-simulation frameworks present in the literature; however, these frameworks lack domain flexibility.

3.3.4 Conclusions

There is remarkable progress in the field of co-simulation; we compare the conclusions of the state-of-the-art article on co-simulation from 2018 [7]. First, by 2018, the large majority of applications used ad hoc couplings between simulators, which now (at least for the co-simulation-based optimization share) is mostly dominated by import standards and co-simulation frameworks. Second, the challenges for co-simulations were many, but two specific mentions are already discussed in this section: modular coupling and finding standards for heterogeneous co-simulation. Third, Gomes et al. assert that the main conclusion of the review is that there is a lack of research into the modular, stable, and accurate coupling of simulators, which is now an important concern of software developers, as shown by the amount of co-simulation software available. The next step for the co-simulation technique that provides complex system abstractions is to offer complex insights into the system. This is the objective of implementing optimization as a general combined technique; this combination offers to explore, study, and improve complex systems.

Some of the strategies found in this section for coupling simulations are treating each simulation as a black-box as defined in 14. The idea is that by using tools like a plugin, toolbox, library, or exporting standards, a model can be included in a different software to be coupled with other models. This makes the black-box approach an appealing structure for a co-simulation architecture.

3.4 Optimization analysis

In this section, we present a review of the optimization methods used in co-simulation-based optimization (Section 3.4.1) and analyze the interaction strategies between co-simulation and optimization (Section 3.4.2) to understand the nature of the interaction illustrated in Figure 2.13. An overview of the methodologies and interactions in this field helps identify promising approaches for working with co-simulation-based optimization, which benefits both software companies and users.

As described in Section 2.4.2, the interface between the co-simulation and the optimization method can be understood using the black-box approach. The goal of this analysis is to understand the optimization strategies and interfaces used to combine optimization and co-simulation in the literature. To achieve this, we reviewed the database sample (Table 3.5).

3.4.1 Optimization methods in the literature

The optimization methodology uses a model to test and find improvements in the system by tuning the structure or parameters, which can lead to an optimal or pseudo-optimal performance configuration. This

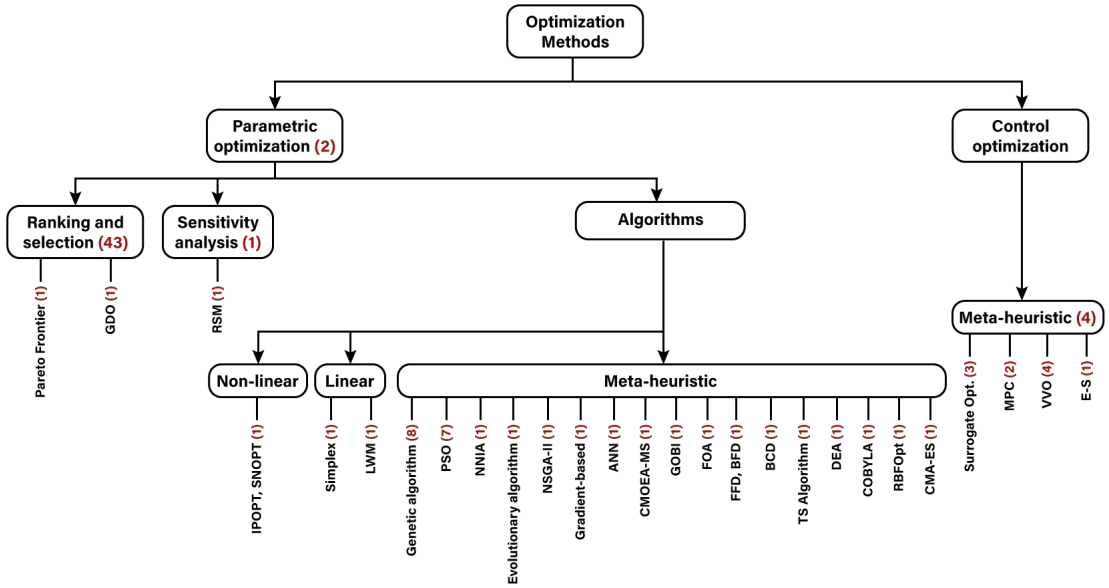


Figure 3.4: Optimization categories and specific methodologies used in the sample database (the number of articles using each method is included in red in parentheses, see table 3.5) and 3.6).

makes the process less expensive than testing on a deployed system. To understand how the optimization methodologies are being implemented in the co-simulation field, we analyze the optimization types present in the literature sample, which follows the tree structure abstracted in Figure 3.4.

From the main parametric/control optimization classification (discussed in subsection 2.3.2), we found several examples in the documents reviewed; however, there are additional sub-categories that provide more information about the characteristics of each optimization method used in the literature. The classification depends on the problem's nature as well as the approach taken to solve it [10] [1]. Now, we briefly describe each category and optimization method in Figure 3.4, including articles from the sample database that use each method (see Table A.1 for abbreviations).

In the parametric optimization category, we find approaches that aim to reduce the solution search space to a relatively small size (most of the time under 20 scenarios [10]) using intuitive or mathematical tools, to be later confronted against each other with an objective function. An example of this approach is *Ranking and selection.*

Definition 24. *(Ranking and selection optimization method.) Method designed to select the best solution among a set of competing candidates [10]. In simulation-based optimization, we refer to ranking and selection as the process of defining a limited number of possible simulation configuration candidates and selecting the best candidate based on a particular objective function.*

The ranking and selection method is the most used in the literature (with 43 of the 96 articles in this category, see Table 3.5), with two additional cases of particular ranking and selection post-processing techniques, namely, Pareto Frontier [76] and GDO [74]. Sensitivity analysis is another approach that reduces the search space by performing a mathematical treatment of the available data, like in [137] and a more specific method of sensitivity analysis, like RSM in [71].

The algorithmic approach to parametric optimization follows a specific set of steps to find an optimal or near-optimal result while avoiding exploring the search space. The linear (e.g., IPOPT, SNOPT [37])

[1] The Figure 3.4 classification is not exhaustive regarding existing optimization methods, but it is exhaustive regarding the 96 articles reviewed in the sample literature database.

and non-linear (e.g, simplex [166] and LWM [103]) algorithms depend on the assumption that we know the tendency of the solution.

In contrast, for other algorithms that deal with unknown tendencies, the meta-heuristic approach can offer interesting algorithms that employ distinct features to advance from one system's performance to a better one until a stopping criterion is satisfied. These methods are shown in Figure 3.4, and we link, in the next paragraph, each algorithm with the articles from the database.

We find in the literature sample database meta-heuristic methods using adaptive algorithms such as Particle Swarm Optimization (PSO) [133, 89, 4, 96, 150, 66, 91], also evolutionary algorithms [55] such as Fruit-Fly optimization algorithm (FOA) [134] (both part of the swarm intelligence approach), constrained multi-objective evolutionary algorithm with multiple stages (CMOEA-MS) [69], Covariance matrix adaptation evolution strategy (CMA-ES) [46], Differential Evolution Algorithm (DEA) [160], genetic algorithms [158, 132, 94, 95, 63, 65, 80, 83] including the variation NSGA-II [38]. We also have gradient-based algorithms like GOBI [39] and Block Coordinate Descent (BCD) [129], methods that approximate behavior to mathematical functions like Constrained Optimization By Linear Approximation (COBYLA) [162] and Radial Basis Function Optimization (RBFOpt) [59], methods that use Neighborhood search strategies like Non-Dominated Neighborhood Immune Algorithm (NNIA) [41] and Tabu Search (TS) [84], and finally other approaches like Artificial Neural Networks (ANN)[153], and FFD/BFD [128].

Compared to parametric optimization, the number of control optimization problems using co-simulation in the literature is quite small (14 of the 96 documents). In control optimization, due to the dynamic nature of the system, the solutions are mostly focused on dynamic or mathematical programming [10], which explains the dominance of meta-heuristics found in the literature. We found studies that implement meta-heuristics [148, 147, 3, 86] to optimize dynamic systems with techniques such as surrogate optimization [100, 154, 101], Model Predictive Control (MPC) [61, 60], Volt-var optimization (VVO) [87, 88, 104, 53], and EXTREMUM-SEEKING (E-S) [67]. The difference between the number of parametric and control optimization applications in the co-simulation-based optimization field shows that currently, the parametric methods are more relevant and require emphasis in the research efforts for co-simulation-based optimization tools.

The literature contains a variety of algorithms that demonstrate the wide range of optimization techniques that may be applied in a co-simulation scenario. These techniques involve modifying parameters or control strategies to explore the search space for optimal performance. Regardless of the approach, the interaction between co-simulation and optimization is crucial for comprehending the conceptual relationship between these elements. To investigate this further, we conduct an interoperability analysis similar to the one conducted for co-simulation interoperability in order to identify the features of this collaboration.

3.4.2 Optimization and co-simulation interoperability

As discussed in Section 3.3.3, an analysis of the software used to develop the implementations in the literature gives the opportunity to study the characteristics of the interface between components. In the previous section (Section 3.3.2), we discussed the three approaches used to handle the multi-domain interoperability within the co-simulation; now, we do the same analysis focused on the interoperability between co-simulation and optimization. To understand how the collaboration of co-simulation and optimization is enabled, we review the sample database of articles to find the interoperability management strategies in the literature.

Simulation-based optimization

The first category covers the cases where the optimization strategy used consists of the definition of a few scenarios to be tested, from which the optimal scenario is searched. For this optimization strategy, the co-simulation interacts with the user who decides on the scenarios to test, which means there is no explicit software involved in the optimization process. We find 38 articles in the database of this

review that used this Ranking and Selection optimization methodology (also known as simulation-based optimization [10]) without using any optimization software (see table 3.5). The strategy of this method is to reduce the search space as much as possible to have a reasonable number of scenarios to co-simulate, compare, and select the optimal.

There is one particular case in the database, in [103], where the co-simulation was used to define the constraints of the optimization problem, and the optimization process (Linear Weighted Method) was performed manually by the researcher. We decided to fit this strategy in this category, as it uses the co-simulation to explore the behavior of the system (in this case, to obtain the constraints), and then it uses these insights to perform the optimization (solving the mathematical model created with the constraints).

Software collaboration

The second category refers to the cases where two or more different software environments were combined to achieve the co-simulation-based optimization process. In this category, we see the presence of the black-box optimization approach (Definition 13), which consists of the separation of the co-simulation and the optimization process as two independent elements to be connected (in this case, two separate software interacting). We divide this category into three sub-categories, which differ in the way that the software is connected.

The subcategories are the case where the interaction is performed by the user (noted in Table 3.4 as user-managed), where the interaction is automatized using a coding language, where integration tools are exploited to connect the components (these tools can be APIs, Plugins, Exporting standards, or coding libraries), and where a black-box optimization software is used. All cases are classified in Table 3.4, where we can see the co-simulation software, the optimization software, and the method used to integrate these components.

The first sub-category is the user-managed interface, which refers to the cases where the researcher takes the outputs of the co-simulation software and provides the data as input to the optimization software by personally assigning the input/output files [100, 101, 81, 88, 87, 96].

The second sub-category is the coded interface that automatizes the interactions by using a high-level language such as C++ [84, 104], ActiveX [83], or Python [162, 61, 95, 160, 60, 63, 129]. The difference between the first and second sub-categories is that the coded interface can be reused in further co-simulation-based optimization applications.

The third sub-category uses existing tools to handle interoperability; that is, an additional (and often optional) tool from the co-simulation software offers a link to an external optimization software (bringing the optimization method into the co-simulation environment). In this sub-category, we find libraries, Plugins, APIs, and Exporting standards. For libraries, we identify that the TRNSYS library is used to include MATLAB modules [65, 66], and the Python library MicroGP is employed for encapsulating third-party software [158], the Datalink library for the Programming interface protocol (ADS protocol to connect the ADS environment with Python [153]). For Plugins we identify the MATLAB plugin to include FLUENT/ANSYS model [38], ANSYS aaS MATLAB Toolbox to communicate MATLAB with ANSYS [133], MATLAB using a DNP3 client software to communicate with the co-simulation in OPAL-RT [53], the MATLAB toolbox YALMIP for optimization modeling [166], the Simcoupler plugin from the PSIM software to connect moduls of PSIM with MATLAB [91], the MATLAB support package for FPGA hardware [89], GT-SUITE plugin of MATLAB to program models in GT-POWER [97], ADAMS control module to interface MATLAB with ADAMS [90]. For the export standards, we identify the CarSIM exporting tool for Simulink [67], the SolidWorks exporting tool to ADAMS [46]. For APIs, we identify the LiveLink API used as an interface between COMSOL and MATLAB [69, 94].

Finally, the fourth sub-category follows the inverse dynamic from the third category; that is, the co-simulation component is brought into the optimization environment. This is composed of three black-box optimization (as described in Definition 11) tools specially designed to import simulation models to be coupled with optimization algorithms that iteratively test co-simulation configurations to find an optimal scenario. The black-box optimization software found in the literature is GenOpt [4], OptiY [86], and modeFRONTIER [76, 80].

Table 3.4: Methods of interoperability between co-simulation and optimization software found in the literature (Software references in Table B.1).

Co-simulation software	Optimization software	Interface method	Ref
AnnAGNPS	C++	Coded in C++	[84]
EMTP-RV	C++	Coded in C++	[104]
ADS	FEKO	User-managed	[100, 101]
Matlab	GenOpt	Black-box Optimization software	[4]
SWAT	Lingo	User-managed	[81]
ANSYS	MATLAB	Plugin	[38, 133]
CarSim	MATLAB	Exporting standards	[67]
COMSOL	MATLAB	API	[69, 94]
EnergyPlus	MATLAB	Coded in Python	[61]
OPAL-RT	MATLAB	Plugin	[53, 166]
OpenDSS	MATLAB	Coded in Python	[95]
PSIM	MATLAB	Plugin	[91]
RSCAD	MATLAB	User-managed	[87, 88]
Solcore (python)	MATLAB	Coded in Python	[160]
Solidworks, ADAMS	MATLAB	Exporting standards	[46]
SPEED	MATLAB	Coded in ActiveX	[83]
SWAT	MATLAB	User-managed	[96]
TRNSYS	MATLAB	Library	[65, 66]
Xilinx	MATLAB	Plugin	[89]
Python	MicroGP library	Library	[158]
ANSYS	modeFRONTIER	Black-box Optimization software	[76]
OpenModelica	modeFRONTIER	Black-box Optimization software	[80]
SimulationX, LS-DYNA	OptiY	Black-box Optimization software	[86]
ADS	Python	Library	[153]
Aspen plus	Python	Coded in Python	[162]
EnergyPlus	Python	Coded in Python	[60, 63]
HELICS	Python	Coded in Python	[129]
GT-Power	Simulink	Plugin	[97]
Solidworks, ADAMS	Simulink	Plugin	[90]

The difference between the sub-categories is the degree to which the interoperability is already developed, or requires the user to develop it. To be specific, the first sub-category (user-managed) has no existing interoperability; it requires that the user perform the interoperability interface and implement it. The second sub-category (coded interface) has no existing interoperability either; it requires the user to develop and implement the interoperability interface in a reusable (automatic) way. The third sub-category (Plugin, library, API, and Exporting standard) has an existing interoperability automated interface; it requires the user to implement it and use it to bring the optimization tool to the co-simulation environment. The fourth sub-category (black-box optimization software) has an interoperability automated interface, which requires the user to use it to import the co-simulation model into the optimization environment.

Co-simulation-based optimization frameworks

The third and last category is the most robust of the approaches, as it aims to incorporate all aspects of the process into a single software framework (the interface is already automated and implemented in a shared software environment, so it requires no interoperability effort from the user). A software environment that fully supports the co-simulation-based optimization process is a great tool for researchers

to study and understand complex systems insightfully, with a reliable complex virtual representation and several optimization methods available to be implemented into the model effortlessly, which is a big opportunity for upcoming studies. The big-picture approach of the framework methodology provides an environment that encourages multidisciplinary collaboration and reusability. In the literature, we found a total of 11 co-simulation-based optimization frameworks that are specially designed for a domain-specific research problem, there are the electronic framework offered by ADS [155] used in [154], the process engineering dedicated framework APECS [157] used in [30], the multiphysics framework COMSOL [142] used in [37], the fog computing framework COSCO proposed in [39], the mobility framework COSMO proposed in [149] and used in [148, 147, 3], the physics framework DAKOTA-IPS [168, 169] proposed in [29], the electric vehicle power framework EV-ECOSim proposed in [132], the mathematical-based framework MATLAB [131] used in [55], the hardware and software (H/S) framework MODUS used in [141, 140], the Cyber-physical Systems framework EXPPO proposed in [128], and the Computer-aided Design framework Rhino 3D [152] used in [59].

The frameworks presented propose a generalization of the co-simulation-based optimization approach to propose an environment for a specific domain. Even more general frameworks are found in the literature that aim to use the same approach to cover multidisciplinary domains in general. The frameworks found present a software environment with the purpose of total simulation integration for design, manufacturing, and engineering problems. It offers a wide variety of domain-specific modules to create simulations that share the same environment for maximum interoperability, such as ANSYS [139] (used in [41, 71, 74, 137]), Isight [170] (used in [134]), and LabVIEW [151] (used in [150]).

We bring to attention the black-box optimization approach discussed in 2.3.3, which, as defined in 13, is relevant in optimization problems where the structure of the objective function and/or constraints is unknown, unexploitable, or nonexistent. In the literature database reviewed, the optimization method is used to improve the performance of a co-simulation based on defined indicators, but the calculation of the indicators directly relies on the execution of the co-simulation. This means that the implementations of co-simulation-based optimization reviewed fit within the black-box optimization approach structure.

3.4.3 Challenges

We summarize the main challenge as *How to combine co-simulation with optimization in a general manner?*. In chapter 2, we talked about a general definition and components of co-simulation, the types of optimization, and the black-box approach as an abstraction tool capable of conceptually combining co-simulation and optimization. We then conducted a literature review on this chapter to examine the existing applications and frameworks proposed, identifying design patterns.

In the literature, almost half of the cases (47 out of 96) choose to diminish the scenario amount to perform the analysis over a controlled number of co-simulations to find the optimal (sensitivity analysis, ranking & selection). The other half of the cases (44 out of 96) use meta-heuristics to find a near-optimal solution, as each optimization method employs different approaches, each with varied interoperability strategies (e.g., code, plugin, library, optimization framework, import/export models) to connect the optimization strategy and the co-simulation. There is a challenge to propose a generalized structure for co-simulation-based optimization that ensures interoperability of different optimization strategies.

The software variety shown in this section highlights extensibility as a relevant feature; extensibility was already mentioned in the multidisciplinary analysis as a key challenge (subsection 3.3.3) of the co-simulation field; however, the same motivation behind this importance is found in the co-simulation-based optimization field.

In Table 3.4, we see a landscape of the strategies used to combine co-simulation software with optimization software. Interestingly, most of the strategies treat both components as separate modules to be coupled, a compatible perspective with the black-box approach. This is a relevant emergent pattern, as we can determine that a challenge for any co-simulation-based optimization framework is to achieve a modular/black-box structure separating co-simulation from optimization, which will offer the advantage of replacing each module without affecting the other modules.

To summarize, the architectural patterns that are particularly relevant to handling the co-simulation optimization relation are extensibility and modularity.

3.4.4 Conclusions

We find in the literature some proposals that contribute to the field with ideas like cross-company collaboration [131, 142], domain-specific frameworks combining specialized software with optimization tools [39, 157, 149, 155, 132, 29, 140, 152, 128], multi-disciplinary sub-systems simulations in a common environment [139, 170, 151], and conceptual frameworks using a modular architecture with black-box optimization [171, 4, 118, 76, 80, 172, 86]. The many optimization categories and techniques found in the literature showed the wide variety of optimization possibilities available. This shows the complexity of deciding which optimization methodology to utilize; this is a reference to the "No free lunch theorem".

Definition 25. *(No free lunch theorem.) In the optimization field, the No Free Lunch theorem establishes that for any optimization algorithm, any elevated performance over one type of problem is offset by performance over another type [173].*

We highlight the importance of extensibility and the black-box approach as an adaptive strategy for implementations, as it offers flexibility in the integration of external simulations and optimization methods.

3.5 State-of-the-art synthesis

3.5.1 Literature classification

The effort of reviewing the publications database of publications using co-simulation-based optimization resulted in the identification of patterns in the methodologies followed. The different patterns are refined into classifications that highlight important differences between the applications. The totality of the classification is seen in Table 3.5, where we classify the 96 documents found using the equation 3.1. Each row contains a document, and the columns are defined as follows:

- Construction approach used to develop the co-simulation, the approaches are presented in subsection 3.2.2.
- Simulation domains present in the co-simulation, as discussed in subsection 3.3.1, each co-simulation is combining two or more models, these models are defined from the perspective of a scientific domain. In this column, we mention the domains included in the co-simulation.
- The simulation software used to create each of the models from the co-simulation, the way that the different simulations are integrated is described in subsection 3.3.2.
- Optimization method as shown in Figure 3.4.
- Optimization software as discussed in subsection 3.4.2.

Table 3.5 uses the following notations:

- **Non*:** Numerical simulation or system analysis where the author does not mention any software involved in the process.
- **Non**:** Manual numerical optimization.
- **Not Mentioned*:** The authors only mention they used "a commercial 3D EM simulator".
- The optimization methods notation description can be found in Table A.1.
- The software mentioned in the classification is briefly described and referenced in Table B.1.

The compilation of the 96 articles into one table serves to showcase the diversity in terms of domains, optimization methods, and software that was found in the literature. This big-picture view of the method under study makes clear the relevance of co-simulation-based optimization to deal with multi-disciplinary optimization problems.

Table 3.5: Literature sample classification (two-page table).

Article	Approach	Simulation domains	Simulation software	Optimization method	Opt. Software
[53]	Top-down	Power, control, communication	OPAL-RT, DNP3, EXATACPS	VVO	MATLAB
[41]	Top-down	Fluid, solid	FLUENT, Solidworks	NNIA	FLUENT
[54]	Top-down	Fluid, electric	Spectre, LTspice	Ranking and selection	Non
[55]	Top-down	Robotic, dynamics	Gazebo, Simulink	Evolutionary Algorithms	MATLAB
[42]	Bottom-up	Fluid, building	EnergyPlus, FLUENT	Ranking and selection	Non
[52]	Bottom-up	Fluid, control	FloMaster, Simulink	Ranking and selection	Non
[43]	Bottom-up	Fluid, solid	Solidworks, FLUENT	Ranking and selection	Non
[93]	Bottom-up	Solid, dynamic	EDEM, Adams	Ranking and selection	Non
[132]	Top-down	Power, control	GridLAB-D, python	Genetic algorithm	Python
[37]	Top-down	Electric, fluid	COMSOL, FLUENT	IPOPT, SNOPT	COMSOL
[38]	Top-down	Mathematical, fluid	Simulink, FLUENT	NSGA-II	MATLAB
[133]	Top-down	Hydraulic, electrical, control	AQWA, ANSYS	PSO	MATLAB
[5]	Top-down	Control, electrical, numerical	OpenModelica, Java	Gradient-based	Java
[44]	Top-down	Soil, solid	EDEM, RecurDyn	Ranking and selection	Non
[40]	Bottom-up	System, fluid, magnetic	Non*	Ranking and selection	Non
[153]	Top-down	Solid, circuit	ADS	ANN	Python
[57]	Top-down	Solid, fluid, electrical	COMSOL	Ranking and selection	Non
[69]	Top-down	Fluid, solid	COMSOL	CMOEA-MS	MATLAB
[146]	Bottom-up	Traffic, analysis, emissions	VISUM, MOVES	Ranking and selection	Non
[89]	Top-down	Hardware/Software, Control	Xilinx, Simulink, MATLAB	PSO	MATLAB
[70]	Top-down	Fluid, mechanical	Gleeble	Ranking and selection	Non
[39]	Top-down	IoT, fog computing	iFogSim, python, Docker	GOBI	Python
[134]	Top-down	Multi-disciplinary	ANSYS	FOA	Isight
[158]	Top-down	Boolean networks	Python, Pyboolnet	Genetic algorithms	MicroGP
[81]	Top-down	Dynamics, soil and water	SWAT, SDM	Ranking and selection	Lingo
[60]	Bottom-up	Building, control	UCEF, EnergyPlus	MPC	Python
[102]	Bottom-up	Dynamics, mechanical, numerical	UM, Simulink, Abaqus	Ranking and selection	Non
[103]	Bottom-up	Engine, solid	GT-POWER, FLUENT	LWM	Non**
[94]	Bottom-up	Optics, solid	MATLAB, COMSOL	Genetic algorithm	MATLAB
[71]	Top-down	Numerical, fluid	Particle Flow Code, FLUENT	RSM	ANSYS
[128]	Top-down	General	Python, API REST, Docker	FFD, BFD	Python
[72]	Bottom-up	Solid, fluid, dynamic	ANSYS, Fluent, EDEM	Ranking and selection	Non
[129]	Top-down	Power, communication, control	GridLAB-D, NS-3, python	BCD	Python
[73]	Bottom-up	Solid, fluid	EDEM, FLUENT	Ranking and selection	Non
[58]	Bottom-up	Dynamics, mechanical, numerical	MATLAB, ANSYS	Ranking and selection	Non
[4]	Bottom-up	Energy, sound	TRNSYS, MATLAB	PSO	GenOpt
[84]	Bottom-up	Hydrological, numerical	AnnAGNPS, C++	TS algorithm	C++
[166]	Top-down	Real-time, computer	OPAL-RT, ePHASORSIM	Simplex	MATLAB
[95]	Top-down	Power, electric	MATLAB, OpenDSS	Genetic algorithm	MATLAB
[96]	Bottom-up	Groundwater, pollution, hydrological	MODFLOW, MT3DMS,SWAT	Multi-objective PSO	MATLAB
[160]	Top-down	Photonic, electronic	Solcore, Python	DEA	MATLAB
[62]	Top-down	Control, building	Dymola, Energyplus	Ranking and selection	Non
[162]	Top-down	Solid, dynamic	Aspen plus	COBYLA	Python
[63]	Top-down	Control, building	Dymola, Energyplus	Genetic algorithm	Python
[135]	Bottom-up	Circuit, solid	Simplorer, Maxwell	Ranking and selection	Non
[125]	Bottom-up	Electrical, dynamics	MATLAB, Simulink	Ranking and selection	Non
[147]	Top-down	Transport planning	VISUM, ARENA	Control meta-heuristic	MATLAB
[74]	Top-down	Solid, fluid	Pro ENGINEER, FLUENT	GDO	ANSYS
[59]	Top-down	Building, Energy	EnergyPlus, Grasshopper	RBFOpt	Opossum
[148]	Top-down	Transport planning	VISSIM	Control meta-heuristic	MATLAB
[61]	Top-down	Building, control	EnergyPlus, MATLAB	MPC	MATLAB
[150]	Bottom-up	Physic, control	LabView	PSO	LabView

Table 3.6: Literature sample classification (continuation).

Article	Approach	Simulation domains	Simulation software	Optimization method	Opt. Software
[75]	Bottom-up	Fluid, solid	EDEM, FLUENT	Ranking and selection	Non
[174]	Bottom-up	Numerical, circuit	Non*	Ranking and selection	Non
[105]	Bottom-up	Numerical, circuits	Simulink, Xilinx	Ranking and selection	Non
[65]	Bottom-up	Numerical, energy	MATLAB, TRNSYS	Genetic algorithm	MATLAB
[80]	Top-down	General	OpenModelica	Genetic algorithm	ModeFRONTIER
[175]	Bottom-up	Numerical, optics	Non*	Ranking and selection	Non
[154]	Top-down	Electromagnetic, circuit	ADS	Surrogate opt.	ADS
[99]	Bottom-up	Circuit	HFSS, MoM	Ranking and selection	Non
[76]	Top-down	Solid, fluid	FLUENT, MATLAB	Pareto frontier	ModeFRONTIER
[140]	Top-down	Hardware/Software, language	SystemC, Simulink, Eclipse	Ranking and selection	Eclipse
[3]	Top-down	Transport planning	VISUM, ARENA	Control meta-heuristic	MATLAB
[82]	Bottom-up	Circuit, signal, power	HSPICE, CST	Ranking and selection	Non
[87]	Bottom-up	Electric	RSCAD	VVO	MATLAB
[67]	Top-down	Vehicle, control	CarSim, Simulink	EXTREMUM-SEEKING	Simulink
[143]	Bottom-up	Thermal, electrical	Sigrity PowerDC	Ranking and selection	Non
[88]	Bottom-up	Network, Control	RTDS,MATLAB	VVO	MATLAB
[136]	Top-down	Mechanical, electromagnetic	ANSYS	Ranking and selection	Non
[66]	Top-down	Building, ventilation	TRNSYS, TRNFLOW	PSO	MATLAB
[64]	Top-down	Energy, fluid	TRNSYS, FLUENT	Ranking and selection	Non
[86]	Bottom-up	Mechanical, hydraulic, control	SimulationX, LS-DYNA	Control meta-heuristic	OptiY
[90]	Bottom-up	Solid, mechanical	Solidworks, Adams, Simulink	Ranking and selection	Simulink
[137]	Bottom-up	Mechanical, numerical	Adams, Simulink	Sensitivity analysis	ModelCenter
[176]	Top-down	Mechanical, chemical, thermal	Non*	Ranking and selection	Non
[85]	Top-down	Mechanical, numerical	OSCAR, Adams	Ranking and selection	Non
[138]	Top-down	Electromagnetic, numerical	HFSS	Ranking and selection	Non
[104]	Bottom-up	Electromagnetic, network	EMTP-RV, Opnet	VVO	Opnet
[46]	Top-down	Mechanical, physic	Solidworks, Adams	CMA-ES	MATLAB
[91]	Top-down	Electric, control	PSIM, Simulink	PSO	MATLAB
[98]	Bottom-up	Mechanical, hydraulic	Adams, UG	Ranking and selection	Non
[141]	Top-down	Hardware/Software, language	SystemC, Simulink, Eclipse	Ranking and selection	Eclipse
[77]	Bottom-up	Engine, fluid	GT-Power, FLUENT	Ranking and selection	Non
[78]	Top-down	Numerical, fluid	VSI Fortran, FrontFlow/Blue	Ranking and selection	Non
[29]	Top-down	Fluid, electrochemical	AMPERES, IPS	Parametric opt.	DAKOTA
[56]	Bottom-up	Mechanical, control, hydraulic	Simpack, Simulink, AMESim, DSHPlus	Ranking and selection	Non
[83]	Bottom-up	Electromagnetic, thermal	SPEED, MotorCAD	Genetic algorithm	MATLAB
[100]	Bottom-up	Electromagnetic, circuit	FEKO, ADS	Surrogate opt.	FEKO
[177]	Top-down	Electromagnetic, circuit	Not mentioned*	Ranking and selection	Non
[97]	Top-down	Thermodynamic, solid	GT-Power	Ranking and selection	Simulink
[101]	Bottom-up	Electromagnetic, circuit	FEKO, ADS	Surrogate opt.	FEKO
[106]	Bottom-up	Forging, mechanical	Forge	Ranking and selection	Non
[167]	Top-down	Robotic, control	MONSTER, Simulink	Ranking and selection	Non
[30]	Top-down	Fluid, chemical	FLUENT, Aspen Plus	Parametric opt.	APECS
[92]	Top-down	Solid, numerical	Simpack, Simulink	Ranking and selection	Non
[178]	Bottom-up	Numerical	Non*	Ranking and selection	Non
[179]	Top-down	Control, Power	Opal-RT, RT-Lab	VVO	MATLAB
[180]	Bottom-up	Power	OpenDSS	PSO	Python
[181]	Top-down	Physic, solid	ADAMS, EDEM	NSGA-II	Python
[182]	Top-down	Hardware/Software, language	Simulink, SimuPLC	ASHO	MATLAB
[183]	Bottom-up	Fluid, combustion	SIMPIC	Ranking and selection	Non
[184]	Top-down	Circuit	Spectre, COMSOL	Surrogate opt.	Python

3.5.2 Recurrent architectural patterns

We examine framework examples found in the literature to determine some key architectural patterns that we found recurrent and useful for the implementation of co-simulation-based optimization. For this, we start with the CO-SiMulation Optimization (COSMO) approach proposed in [149] and used to optimize control problems on mobility problems (in [3] for multi-modal freight routing, in [147] for dynamic multi-modal freight routing, and in [148] for traffic light control with truck priority). The COSMO approach is shown in Figure 3.5, which presents a *layer-based* architecture that offers the identification of the main components of the process while encouraging the handling of each layer within a dedicated environment (modularity similar to the black-box approach). Another interesting characteristic of the

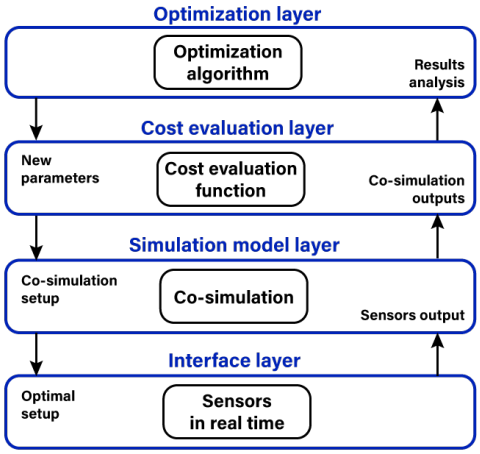


Figure 3.5: Key elements extraction of the COSMO application framework representation (adapted from [3]).

COSMO approach is the domain flexibility, defined as:

Definition 26. *(Domain flexibility in co-simulation-based optimization frameworks.) A software framework is domain flexible when the general approach of the framework enables diverse domains to apply the framework indiscriminately.*

The domain flexibility of COSMO is proven by the four applications of the framework in [3, 147, 148]. Despite the multiple applications, the framework's flexibility is still considered mid-range, as it only covers mobility-related applications that want to improve the performance of a vehicle network. This constraint is what makes the framework discipline-dependent.

The last interesting characteristic of the COSMO approach is the way the interaction between the co-simulation and optimization layers is handled. As seen in Figure 3.5, this intermediate layer between the co-simulation and optimization layers is responsible for the translation of the information between each other in a loop-like information flow that allows the iterative process to be maintained until an optimal value is found. Using this, we define an interface component as:

Definition 27. *(Interface component.) The component concentrates the configuration of the interaction between two or more components. It is intended to simplify the compatibility between the elements by generalizing the structure of the interaction.*

We examined the structure of another framework proposed in [4] for the multi-domain (thermal and acoustic) optimization of nearly zero energy buildings. The proposed framework is shown in Figure 3.6a, and the key elements are extracted in Figure 3.6b. In this framework, we see again characteristics such as a separation in components (black-box approach) and the interface components from Definition 27 (seen in Figure 3.6b). However, we see a new element in the scheme, which is the pre-processing, which is defined as:

Definition 28. *(Pre-processing in a co-simulation-based optimization framework.) The information that must be provided beforehand for the co-simulation and optimization processes to start.*

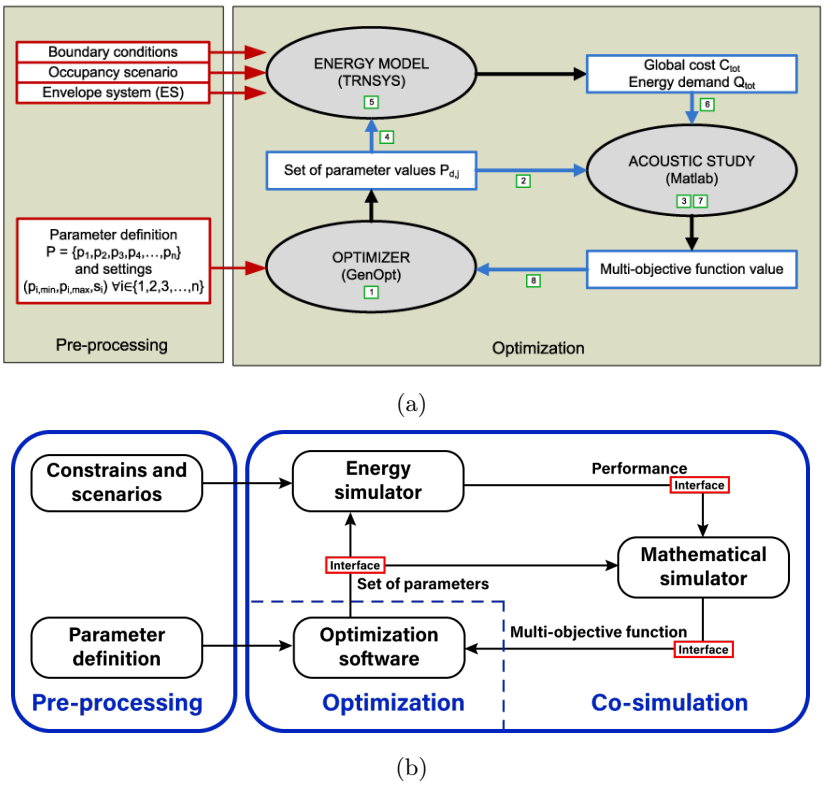


Figure 3.6: (a) Optimization framework for a co-simulation in the energy building field (taken from [4]) (b) Key elements extraction of the framework proposed by [4]

Table 3.7 contains a compilation of frameworks from the state-of-the-art, where we identify the architectural patterns present. The *modularity* column shows if the framework presents a separation between co-simulation and optimization (black-box approach from Definition 11). The *flexibility* column evaluates each framework in terms of the domain flexibility (Definition 26), which relates to the amount of different applications found in the literature (each article contains one application of the framework, except for the case of [76] where there are four applications in the same article). The *domain* column shows the discipline in which the framework is defined and by which the framework is constrained. The *interface* column shows the presence of a component between co-simulation and optimization to handle the communication (defined in 27). The *pre-processing* column shows whether the framework structure includes a stage in the framework to handle the pre-defined information required to execute the optimization process (Definition 28).

3.5.3 Discussion

The path followed to review the topic of co-simulation-based optimization started with the consolidation of a sample database of 96 articles using the equation 3.1 to find articles in the literature that combine co-simulation with optimization, and then we performed a study endeavor consisting of analyzing the articles from different points of view, general, multidisciplinary, and optimization.

We discussed the presence of different methodologies similar to co-simulation-based optimization under different names like coupled simulation, nested simultaneous design, multidisciplinary design optimization, resilience optimization, and distributed multi-agent coordination. The variety of names found in the literature makes the reviewing process complex; for this reason, we decided to establish a representative

Table 3.7: Characterization of the co-simulation-based optimization frameworks found in the literature by key elements. (ND* = The characteristic is Not Detailed in the description of the framework.)

Article	Modularity	Flexibility	Domain	Interface	Pre-processing
[3, 147, 148]	Yes	mid-range	Mobility	Yes	ND*
[4]	Yes	ND*	Energy	No	Yes
[95]	Yes	ND*	Electric	Yes	ND*
[63]	Yes	ND*	Buildings	Yes	ND*
[59]	Yes	ND*	Buildings	ND*	ND*
[150]	Yes	ND*	Vehicle design	ND*	ND*
[65]	Yes	ND*	Buildings	Yes	ND*
[76]	Yes	Mid-range	Flexible wings design	ND*	ND*
[87, 88]	Yes	Short-range	Smart grid	Yes	ND*
[67]	Yes	ND*	Cruise control	ND*	ND*
[66]	Yes	ND*	Buildings	ND*	Yes
[91]	Yes	ND*	Control system	ND*	ND*
[97]	Yes	ND*	Engine control	ND*	ND*

sample of the methodology by compiling uses of co-simulation, cosimulation, and coupled simulation. Therefore, we propose using *co-simulation-based optimization* as an unifying name for the methodology described in this thesis in future applications.

From the general analysis of the literature in section 3.2, we see that the involvement of several domains in a co-simulation-based optimization problem is a relevant and increasing situation in the field. We also identified a new co-simulation construction process (top-down and bottom-up approaches) and the subsequent conceptual compatibility with optimization using the black-box approach. This approach is supported by the fact that the implementations are already using modularity to build an architecture that handles co-simulation-based optimization.

In section 3.3 we performed a multidisciplinary analysis of the database that showed the wide variety of domains cooperating in co-simulations, many domain-specific software are used for each sub-system, which then need to interact with the others. This led us to study the interoperability of co-simulations, as it is a key factor in the performance of a complex system representation. The presence of many manual implementations shows the importance of the proposal of collaborative standards and software frameworks to simplify the implementation of co-simulation.

Then, in section 3.4, we performed an analysis of the database from the point of view of optimization that showed characteristics similar to the multidisciplinary analysis, that is, a wide variety and interoperability relevance. Furthermore, some robust frameworks found in the literature are starting to grasp a generalization of the co-simulation-based optimization process to be used in the study of complex systems. The fact that the implementations are using different ways to connect the co-simulation and optimization modules proves the importance of the question that Figure 2.13 represents, that is, how can we define the interface component in a general manner?

Finally, we present Table 3.5, which contains the compiled classification of the entire literature database using the classifications used throughout this chapter. Additionally, Appendices A and B complete the information required to understand the classification notation.

3.5.4 Conclusions

The main contribution of this survey, which presents a state-of-the-art review on co-simulation-based optimization, is to provide an analysis of how co-simulation and optimization are implemented. To do this, we examine the literature for emerging patterns in proposed strategies to identify a general abstraction of the methodologies. This abstraction showed that extensibility, export/import standardization, and frameworks are the main approaches to handling the various disciplines involved in the co-simulation-based optimization, as well as the various optimization techniques available.

In the process of developing this state-of-the-art research, we achieved several contributions relevant to co-simulation-based optimization, one of which is the construction classification for co-simulations in subsection 3.2.2, which contributes to a better understanding of the modeling process of a complex system, which helps with the identification of new complex systems that can benefit from the co-simulation method.

The number of heterogeneous domains participating in this topic showed the relevance and applicability of the co-simulation-based optimization method for understanding and predicting complex systems while also bringing attention to the way in which the components interact. This means that the collaboration of diverse domains implies an effort of interoperability, and based on the analysis done over the literature, we proposed the classification into three types of interoperability strategies: manual implementations, software collaboration standards, and frameworks that offer a multi-domain shared environment for co-simulation.

From an optimization perspective, we found and classified the optimization methods used in the literature database (Figure 3.4). This classification allowed us to examine the characteristics of the interaction between the co-simulation and optimization components to determine the three types of interoperability strategies that are utilized in the literature to integrate co-simulation and optimization; that is, manually, by software collaboration, and using co-simulation-based optimization frameworks.

We conclude that the black-box approach is a pertinent concept to study the interoperability between co-simulation and optimization. This approach can collect co-simulation and optimization into a shared conceptual environment, which is a reflection of the implementation environments present in the literature.

Another conclusion of the reviewing process of this chapter is linked to the interest in capitalizing on the many applications by proposing frameworks that offer extensibility, modularity, pre-processing, and interface handling capabilities that make possible the integration of co-simulation and optimization tools.

Chapter 4

Co-simulation-based optimization Framework proposal

Contents

In this chapter, we present the core contribution of this thesis, a general co-simulation-based optimization framework that will provide a structured architecture to study and improve a complex system by using a combination of a virtual representation of the system (co-simulation) and algorithms to optimize the performance of the system. We first discuss the motivation of the framework proposed (Section 4.1.1), then the assessment characteristics of the framework in subsection 4.1.2, followed by a description of the framework (section 4.2) in terms of the components, interactions, and dynamics. We discuss the framework scope and limitations in section 4.3. In section 4.4, we describe the software chosen to perform an implementation case of the software framework.

4.1 Framework development and assessment

4.1.1 Motivation

The literature review performed in chapter 3 showed that there is a large amount of ad hoc implementations of co-simulation-based optimization, where we identify some structural patterns that are worth

capitalizing on into one general framework proposal. The number of applications also shows the utility of general guidelines that can be reused in future implementations. To reconcile the implementation efforts of the literature, we propose a conceptual general framework; the idea is to avoid the further accumulation of framework structures similar to each other by establishing a general architecture that collects the lessons learned from existing applications and serves as a template for future implementations of the methodology.

An exciting approach in the literature is the software framework (defined in 22), which aims to provide a reusable way to replicate or expand the application of the method. This framework is intended to formalize and make explicit the reuse of the architectural patterns found in the literature.

The generalization of the framework architecture is characterized by an emphasis on design patterns in framework design and use [127]. The patterns found in chapter 3 are applied in more than one domain and more than one application in the literature; these patterns are modularity (definition 11), flexibility (definition 26), extensibility (definition 23), the interface component (definition 27), and pre-processing (definition 28).

4.1.2 Development levels

The development of the framework follows the description provided in Table 4.1, where we see the elements of framework development in contrast with the development of an application. This separation is important to understand the level of abstraction in which the general framework proposed is positioned, that is, the first level of the framework development called *Abstract interface/ Design pattern*. This means that the general framework intends to describe the abstract interfaces and design patterns to be used as guidelines for a co-simulation-based optimization framework implementation.

Table 4.1: Duality of concepts between application concept (left-hand column) and framework concepts (right-hand column). Taken from [127].

Application Development		**Framework Development**
Concrete Object	$\leftrightarrow$	*Abstract Interface/Des. Pattern*
The primary unit of design is the concrete object: the design process begins by identifying the concrete objects to be represented in the application, and they remain the focus of design efforts.		The primary units of design are the interfaces and design patterns: the design process begins by identifying the abstract interfaces and the design patterns, and they remain the focus of design efforts.
Subsystem	$\leftrightarrow$	*Framelet*
Design complexity is mastered by breaking up the application into subsystems. A subsystem is primarily a cluster of related objects.		Design complexity is mastered by breaking up the application into framelets. A framelet is primarily a cluster of related abstract interfaces and design patterns.
Use Case	$\leftrightarrow$	*Implementation Case*
A use case describes a potential usage of the application. Use cases can be used to specify the application, to check the application design, and to describe its usage.		An implementation case describes a potential usage of the framework. Implementation cases can be used to specify the framework, to check the framework design, and to describe its usage.
User Requirement	$\leftrightarrow$	*Functionality*
A user requirement describes an aspect of an application and can be mapped to the application's architectural constructs.		A functionality describes an aspect of a framework and can be mapped to the framework's architectural constructs.

In Table 4.1, we see that a method to assess the framework is by means of an *Implementation case*, which is a particular iteration of the framework to describe the usage. In the same way, the implementation case can be assessed by means of *Use cases*. In summary, from Table 4.1, we will focus on the description

of an abstract interface, an implementation case, and use cases for the development and assessment of the proposal. We decided on three key **levels of development** that will determine the logic followed in the description of the framework as follows:

1. **First level/architectural level:** We refer to the product of this stage as the *conceptual general framework*. Here, we describe the abstract interface and design patterns of the framework.

2. **Second level/implementation level:** In this level, we obtain an *implementation case* of the conceptual framework from the first level. This involves the development of a framework implementation into a software environment.

3. **Third level/application level:** In this level, we find the *use cases* developed using the framework implementation of the second level. This involves the development of a practical application that makes use of the implementation case to solve a problem and describe the usage of the framework.

To further illustrate the relation between these three levels, we propose example 11, where we define the development of a printer general framework.

Example 11. *(Example of the relation of abstract interface, implementation case, and use case.) Let's consider the problem of passing digital images or text onto paper. For this problem, a printer is a possible solution with many possible structures. However, if we want to propose a general structure of a printer, we can think about the three levels of development illustrated in Figure 4.1.*

1. *First level: we describe the general structure and design patterns of a printer. For example, we decide that a printer requires components such as a paper feed tray, a paper output tray, an ink cartridge cover structure, and a control panel.*

2. *Second level: we perform an implementation case of the structure proposed, where we build a printer using the guidelines from the first level.*

3. *Third level: we perform use cases to showcase the usage of the implementation case. In this example's context, we use the printer built in level 2 to print different images and text.*

4.1.3 Assessment of the framework using levels

The assessment of the co-simulation-based optimization framework is crucial for describing how we can validate that the proposed framework is achieving its intended purpose. This assessment is performed from the third level towards the first level as seen in Figure 4.1. The accountability of the framework's contribution must be demonstrated to the extent possible in this thesis work. We start in the **third level**, that is, the application level, where we develop use cases that pose complex systems to be improved using the software implementation. The results obtained at this level are limited to the optimization insights into the complex systems under study.

The assessment of the framework in the **second level**, that is, the implementation level, consists of the comparison of the use cases' performance, which means that we can assess the performance of the implementation case by analyzing the execution performance of the different use cases. The results of this level are interesting to argue the advantages of the framework structure of the co-simulation-based optimization implementation.

Finally, the assessment of the **first level**, the architectural level, refers to the advantages perceived in the framework development as a result of following the abstract interface and pattern designs suggested in the conceptual general framework. The conclusions of this assessment show the importance of the architectural patterns found in the literature.

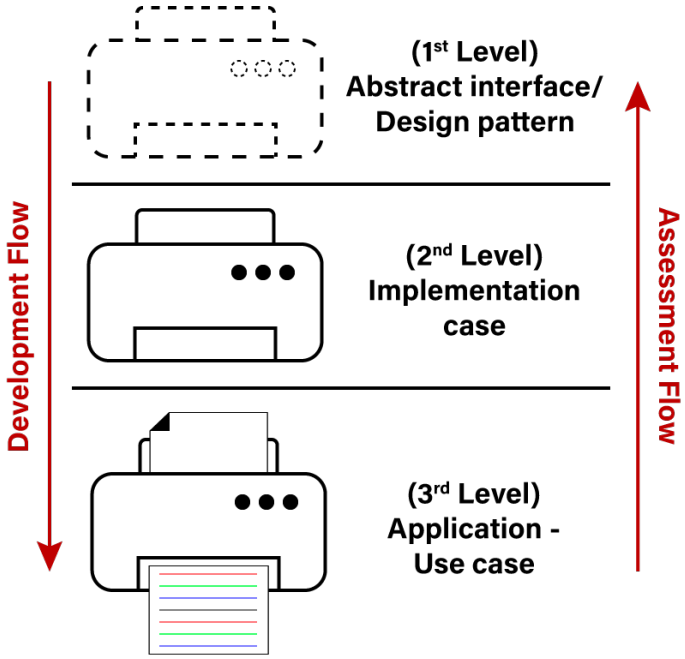


Figure 4.1: Illustration of the difference between the abstract interface, the implementation case, and the use cases for the framework. Using Example 11 as a reference for the illustrations (own authorship).

4.2 Conceptual General Framework description

In this section, we present the structure of the Conceptual General co-simulation-based optimization framework, which is the main contribution of this thesis. The framework aims to capitalize on the findings of the state-of-the-art review on section 3.5. We propose a conceptual framework that integrates co-simulation and optimization; each component will be described along with the information flow coming in and out. The structure proposed intends to unite the efforts of researchers using co-simulation-based optimization by offering a conceptual, domain-independent, and generalized architecture to guide the implementation process.

The conceptual framework that we present is represented in Figure 4.2, where we compile all the components and information involved in the co-simulation-based optimization process. Blue squares represent the main components of the framework, red arrows show the information requirements for the optimization process to start, and black arrows show the information flow of the optimization process. We recognize three major components (optimization, interface, and co-simulation).

The framework is represented with a component-based structure as seen in Figure 4.2. We use a modular/Black-box structure following the tendency of other frameworks (see modularity column in Table 3.7). This approach is an appropriate tool to describe the co-simulation-based optimization process abstractly while maintaining each domain's particularities. Another relevant detail included in the framework is the identification of the pre-processing information required to launch an optimization process (right side of Figure 4.2). We identify these elements to point out the key elements to be worked on during a software framework implementation. We describe each component in terms of the elements that compose it and the way it exchanges information with others.

4.2.1 Co-simulation component

The first component is co-simulation, which contains the multi-model of the target complex system. At the bottom of Figure 4.2, the co-simulation component shows the general structure of a co-simulation, which contains the coupled sub-models that form the multi-model (defined in 6). The co-simulation

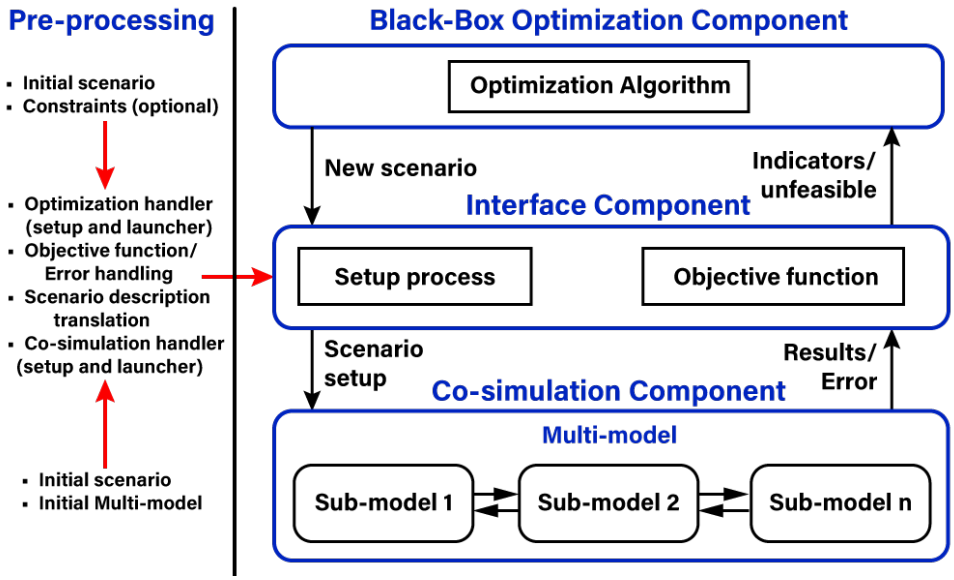


Figure 4.2: Co-simulation-based optimization Framework.

process consists of the execution of the multi-model, where the global system behavior can be measured. We define the notation of the process associated with the co-simulation component as $Cosim(\cdot)$. The role of the co-simulation in the framework is defined by its relation with the optimization process; this interaction focuses on the execution of *scenarios* of the co-simulation that generate measurements that can be interpreted as *indicators* and compared for optimization purposes. This means from the framework's perspective, the co-simulation process is defined as:

$$Cosim(N, I_{cs}) = \vec{O_{cs}}. \tag{4.1}$$

where the tuple (N, I_{cs}) contains the scenario description of the co-simulation, and vector $\vec{O_{cs}}$ contains the behavior of the indicators, which is the output to be sent to the optimization process. Now, we explain the general characteristics of these input and output values.

The co-simulation scenario description is embedded into 2 elements, the multi-model structure N, and the multi-model parameter values I_{cs}. For the definition of the multi-model structure N, we recall the same equation mentioned in Definition 10 to highlight the relevant elements for the framework as follows:

$$N = \langle X, Y, D, \{M_d | d \in D\}, EIC, EOC, IC, Select \rangle. \tag{4.2}$$

Where each element is already described in Definition 10. For the co-simulation component, we consider the possibility of varying the multi-model structure as follows:

- The set of DEVS sub-models $\{M_d | d \in D\}$. We can exchange a model d from the set M_d that has the same external structure (i.e., input X and output Y elements) but different internal dynamics. This variation has the advantage of not affecting any of the other structural elements of the multi-model (provided the external structure is compatible), making it a very interesting feature of the co-simulation component to be explored by the optimization.

- It is also possible to consider the variation of the elements EIC, EOC, or IC. However, the way that this variation would impact the feasibility of the multi-model and the exploration of the optimization process is not yet clear to us. It requires further analysis.

We therefore define the elements N and I_{cs} as the variable elements of the co-simulation for the scenario description. The set I_{cs} contains values representing the initial parameters values of the co-simulation, that is $\{I_{cs}^1, \ldots, I_{cs}^j, \ldots, I_{cs}^p\}$ where I_{cs}^j is the value of the parameter j which has a domain $Dom(I_{cs}^j) \in \mathbb{R}$ with $j \in \{1, 2, \ldots, p\}$.

In the context of the framework, we use the domain definition of the real numbers $\mathbb{R}$ to generalize the interaction as numerical values.

The outgoing flow of information for this component is the co-simulation results, which represent the behavior of one or more variables of the co-simulation. We define the outputs as a vector $\vec{O_{cs}}$ that contains a set of n values representing the behavior of the key variables of the co-simulation, that is $(O^1_{cs}, \ldots, O^i_{cs}, \ldots, O^n_{cs})$ where O^i_{cs} with $i \in \{1, 2, \ldots, n\}$.

The variable types for output values are numerical records obtained from the co-simulation behavior. Depending on the information that we want to obtain from the co-simulation, we can obtain single numerical results, time series results, or non-feasible co-simulation reports. The domain of each possibility is described as:

- The final single value of a variable of the co-simulation, with the domain $Dom(O^i_{cs}) \in \mathbb{R}$.
- A time-series of the variable's value, considering the co-simulation's time $t \in \{1, 2, \ldots, m\}$, then the domain is $Dom(O^i_{cs}) \in \mathbb{R}^m$, where m is the total time-steps of the co-simulation.
- For the possibility of infeasibility of the co-simulation, the domain is defined as $Dom(O^i_{cs}) \in \{null\}$.

4.2.2 Optimization component

The optimization component contains the optimization algorithm, which is chosen based on the necessities and characteristics of the problem. Nonetheless, as mentioned in Section 3.4, various optimization methods are employed in co-simulations, all following a general strategy of providing a *set* of parameters or scenarios to test until a specified stopping criterion is met, thereby terminating the process and providing the chosen optimal result. This component is intended to work with black-box optimization algorithms (described in Definition 13). We recall Equation 2.3 that represents an optimization problem as

$$\min \text{ or } \max \quad ObjFunc(\vec{X}) \tag{4.3}$$

where $ObjFunc(\cdot)$ represents the objective function to be maximized or minimized. However, we mentioned two types of optimization problems (parametric and control) [10]. The first type is the parametric optimization, where the vector $\vec{X}$ from equation 4.3 is defined as a set of q parameters. Each value X_k with the index $k \in \{1, 2, \ldots, q\}$ represents a parameter. The set of parameters is represented by the vector

$$\vec{X} = (X_1, \ldots, X_k, \ldots, X_q). \tag{4.4}$$

The second type is control optimization, which handles dynamic systems that change over time. The objective is to choose a set of actions to move the system along a desirable path [10], for this case, $\vec{X}$ in equation 4.3 is defined as

$$\vec{X} = (\mu(1), \ldots, \mu(k), \ldots, \mu(|\mathcal{S}|) \tag{4.5}$$

where $\mathcal{S}$ denotes the states of the system and $|\mathcal{S}|$ the number of states in the system, $\mu(k)$ refers to the action taken in state k.

Black-box optimization

As seen in equation 4.3 the optimization process depend on the calculation of $ObjFunc(\cdot)$ (commonly referred to as the objective function), however, in the context of co-simulation based optimization, the calculation of the objective function is bounded to the execution of the co-simulation, this means that the objective function has no analytical form, but it consists on a function applied to the outputs of the co-simulation. This approach is the same as the one used to describe the simulation-based optimization dynamics in Figure 2.10, where we see on the right side the simulation that generates a performance based on some scenarios passed by the *optimization algorithm* in the left side.

We use the definition of black-box optimization given in subsection 2.3.3 as the approach to be implemented in the framework. In black-box optimization, the description of the optimization strategy is

compiled in the optimization algorithm; for this reason, we can define the general form of the optimization process by modifying equation 4.3 as

$$\min \text{ or } \max \vec{I_{opt}} \tag{4.6}$$

where $\vec{I_{opt}}$ is the vector containing indicators obtained from the objective function as $(I^1_{opt}, \ldots, I^r_{opt})$ where I^a_{opt} with $a \in \{1, \ldots, r\}$ represents the $a-$indicator. These indicators are numerical values contained in the domain $Dom(I^a_{opt}) = \mathbb{R}$. Note that in cases where $r > 1$, the optimization algorithm needs to support multi-objective optimization problems.

The calculation of the indicators is achieved using the objective function with the results of the co-simulation, which defines the link between the co-simulation and optimization components as follows:

$$\vec{I_{opt}} = ObjFunc(\vec{O_{cs}}) \tag{4.7}$$

using the vector $\vec{O_{cs}}$ described in equation 4.1. This link between the co-simulation and the optimization components is delegated to the interface component. It is important to remember the consideration from black-box optimization about the possibility that the vector $\vec{I_{opt}}$ contains an invalid output ($null$), which is managed by flagging the result as $+\infty$ for minimization problems, and $-\infty$ for maximization problems.

Optimization component generalization

The objective of the optimization component is to apply the algorithm to the indicators to obtain the next scenario to be tested in the co-simulation. This means that the process performed in the optimization component can be represented as

$$Algorithm(\vec{I_{opt}}) = O_{opt}. \tag{4.8}$$

With the set $O_{opt} = \{O^1_{opt}, O^2_{opt}, \ldots, O^s_{opt}\}$ containing s elements that represent the outputs of the algorithm that provide the necessary information to describe a co-simulation scenario (see Figure 2.10). Additionally, we consider the possibility of sub-model variation within the co-simulation as described in Figure 2.11 as part of the scenario description; however, to maintain the generality of the optimization component, we maintain all the set O_{opt} elements as numerical values (we propose the use of natural numbers as category codes, that is, just as labels, not magnitudes, to refer to the sub-model variants to be used), the responsibility of translating these values into parameter and/or sub-model variation is relegated to the interface component. Thus, the domain of the set O_{opt} is defined as

$$Dom(O^b_{opt}) = \begin{cases} \mathbb{R}, & \text{if parameter value} \\ \mathbb{N}, & \text{if model variant value.} \end{cases} \tag{4.9}$$

Where $b \in \{1, 2, \ldots, s\}$ and O^b_{opt} represents the value of the $b-$th element of the scenario description. Each element represents either a parameter value or a numerical label of the variant sub-model to be used in the multi-model (the label equivalence must be defined beforehand).

We define the translation process between scenario description O_{opt} to multi-model and parameter values (N, I_{cs}) as the $setup(\cdot)$ function. We use the following notation for this process:

$$Setup(O_{opt}) = (N, I_{cs}). \tag{4.10}$$

The product of the setup process is used to generate the multi-model to be co-simulated (Equation 4.1). However, to maintain the black-box structure proposed in the framework, the optimization component only handles the algorithm process from Equation 4.8; the remaining responsibility of translation between components from Equation 4.10 is part of the interface component.

4.2.3 Interface component

The previously described two components are constantly exchanging information, which is mainly centered in the equations 4.10 and 4.7, where the processes of setup and objective function calculations are performed. The challenge involved in these two processes is the translation of the information from

one component to another while ensuring compatibility. This is necessary when the output values of one component have a different domain from the input values of the other.

Table 4.2 summarizes the processes happening in the framework; the first column contains the name of the component where the process occurs, the second and third columns show the inputs and outputs of the process, respectively, along with the mathematical notation defined, and the fourth column contains the process mathematical notation. Note that the interface component contains the two processes that transform the output information into the input information. This loop-like process is already illustrated in Figure 4.2. However, we propose a direct representation of the workflow using the mathematical notation from equations 4.1, 4.7, 4.3, and 4.10 in Figure 4.3.

Table 4.2: Processes, outgoing and incoming information representation for the framework components.

Component	Inputs → Notation	Outputs → Notation	Internal process
Co-simulation	Multi-model,Parameters → N, I_{cs}	Results → $\vec{O_{cs}}$	$Cosim(\cdot)$
Optimization	Indicators → $\vec{I_{opt}}$	Scenario → O_{opt}	$Algorithm(\cdot)$
Interface	Results → $\vec{O_{cs}}$	Indicators → $\vec{I_{opt}}$	$ObjFunc(\cdot)$
Interface	Scenario → O_{opt}	Multi-model,Parameters → N, I_{cs}	$Setup(\cdot)$

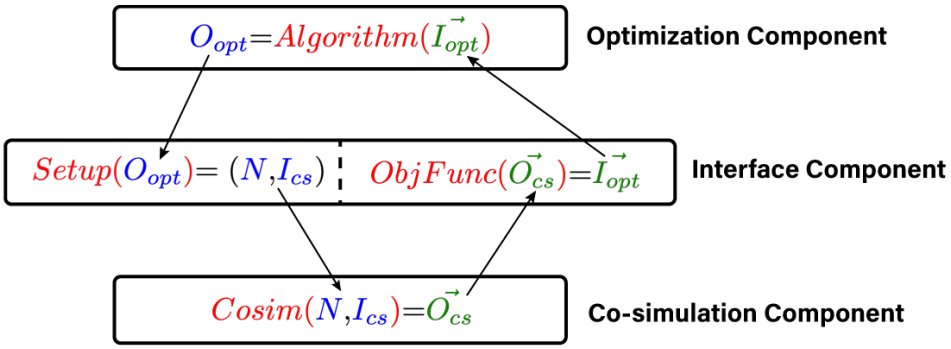


Figure 4.3: Mathematical notation of the workflow for the Co-simulation-based optimization Framework proposed in Figure 4.2. During the optimization, some elements are required to be defined from the beginning and stay fixed (in red), others that also require an initial value but that will vary during the optimization process (in blue), and elements that do not require initialization (in green).

As a consequence of the co-simulation and optimization components' generalization, the intermediate component has the broadest role in the framework, as it needs to reconcile two large-scale tools. We attempt to delimit the implementation workflow to the interactions (the optimization input $\vec{I_{opt}}$ and outputs O_{opt}), the co-simulation inputs I_{sc}, N and outputs $\vec{O_{sc}}$), and component processes (the co-simulation $Cosim(\cdot)$, the $Algorithm(\cdot)$, the $Setup(\cdot)$, and the objective function $ObjFunc(\cdot)$. The developer's effort to make a general framework is limited to the elements in the interface component; the idea is to handle the co-simulation and optimization components as independent exchangeable modules (a plug-and-play approach).

Following the modular approach for the structure of the framework is possible due to concepts such as the black-box optimization and coupled models under the DEVS formalism. These concepts are compatible with the modular approach and are used to develop an environment for co-simulation-based optimization. The Black-box Optimization and Coupled DEVS concepts are established methods for studying and improving systems. With this approach, we can reuse Black-box Optimization and Coupled DEVS models to implement the framework, thereby relegating the majority of the implementation work to the interface component.

4.2.4 Pre-processing

We discussed the relevance of the pre-processing concept proposed for the framework for multi-domain optimization of nearly zero energy building [4], where, in addition to the co-simulation and optimization components, the identification of the information needed to launch the process was considered as a useful part of the framework (see Figure 3.6a). We include this feature in the general framework definition by considering the pre-processing information of our process dynamics. For this, we take the *red* and *blue* values in Figure 4.3 that ensure the self-sustaining execution of the process. These values are the fixed process values and the process that requires an initial value, but will vary along the process. This means that for the framework proposed, the dynamic is as follows:

1. The **optimization configuration**, where all the information necessary to set in motion the optimization is provided (pre-processing), including the algorithm configuration ($Algorithm(\cdot)$), the objective function ($ObjFunc(\cdot)$), the setup process ($Setup(\cdot)$), the co-simulation process ($Cosim(\cdot)$), and the initial scenario to be used (O_{opt}) which is equivalent to the multi-model and the initial parameter values (N, I_{cs}). We identify these elements within Figure 4.3 in red and blue.

2. The **optimization execution**, where the launched optimization process follows the loop-like information flow from Figure 4.3. Throughout the execution, the static processes (in red) modify the dynamic values (in blue and green) until the optimization is finished.

3. **Optimization finalization**, where the optimization algorithm stops and we can obtain the optimal results, particularly we are interested in the tuple $O_{opt}, \vec{I_{opt}}$, which contains the minimal or maximal value obtain in the objective function, along with the co-simulation scenario description that achieve it.

In the framework dynamics, we identify the pre-processing information as the optimization configuration, that is, the elements $Algorithm(\cdot)$, $ObjFunc(\cdot)$, $Setup(\cdot)$, $Cosim(\cdot)$, O_{opt}, N, and I_{cs}). Furthermore, we decided to exploit this generalization to simplify the configuration of several co-simulation-based optimization configurations by batching processes using this pre-processing information compilation.

The intention of including the pre-processing definition in the framework is to ensure the framework's applicability across diverse use cases, as the definition of an experiment configuration structure will simplify the interaction of the user with the framework implementation by reducing the configuration effort to a few elements that define the complex system (co-simulation) that will be improved (using a particular optimization method).

4.3 Framework scope and limitations

The definition of this framework is the result of collecting and analyzing the individual contributions of co-simulation-based optimization in different domains from the literature. The framework is built to support the construction of co-simulation-based optimization by unifying the collective progress in the co-simulation-based optimization field (findings from the state-of-the-art in section 3.5).

The advantage of domain-specific frameworks is that they include features of interest to the domain that facilitate the work of the implementation process. In the case of this Conceptual General Framework (CGF), we define the domain-specific focus on the co-simulation-based optimization domain. However, we get far from a domain-specific contribution when talking about the complex system domains (e.g., fluid, solid, electric, or control dynamics), as the framework aims to be applied to any complex system regardless of the domains internally interacting. This contrast seems to present the framework from two competing interests; however, we present the CGF with the intention of working as a support tool for the application of the co-simulation-based optimization method. This means that existing domain-specific frameworks can also benefit from the CGF contribution by using the architecture proposed to exploit the potential of the co-simulation-based optimization method.

The CGF was built using the inductive reasoning method to understand the conceptual configuration of the ad hoc frameworks and establish a general structure guideline. An appropriate generalization of the conceptual structure serves as a validation of the existing ad hoc applications and offers guidance

to future implementations. The framework can be used to explore the optimization potential within co-simulation or to challenge previously positioned frameworks to validate the chosen structure and/or identify areas for improvement.

A limitation of the framework is the fact that the definition of the co-simulation structure variation needs to be reduced to sub-model labeling to facilitate the sub-model exchange process, while ensuring the generality of the architecture

The main limitation of the CGF is that the claimed generality can only be truly validated through several implementations from different multidisciplinary teams; this means that the use cases developed in this thesis, by themselves, are insufficient to confirm the full extent of the framework's generality. Still, we perform an implementation case and use cases with the objective of it being the first step towards the validation of the contribution of the CGF. The idea is to present use cases that showcase the maximum of the contributions to the co-simulation-based optimization method.

4.4 Implementation case

In this section, we describe the implementation case performed, which consists of the software decisions made to develop the framework implementation case. The objective is to have a software framework built using the CGF guidelines to assess the architecture potential for applying co-simulation-based optimization. The framework implementation is performed using a combination of the Java programming language [185] and the MECSYCO software [186] to develop a software framework for co-simulation-based optimization.

4.4.1 Software framework implementation

In Figure 4.4, we present the structure of the implementation case. The overall environment is hosted in the Eclipse platform, which uses one or more plug-ins to create a modular structure. The Eclipse Workbench offers a desktop development environment, which aims to achieve seamless tool integration and controlled openness by providing a common paradigm for the creation, management, and navigation of workspace resources [187] (in line with the modularity structure from the framework, as we intend to follow the black-box approach defined in 11 for the implementation). Eclipse is interesting for us due to its high compatibility with the Java programming language and MECSYCO, with a robust approach for project development (in line with the extensibility key property found in the literature, Definition 23).

4.4.2 Co-simulation component implementation: MECSYCO Software

We chose MECSYCO as the tool to be implemented in the co-simulation component, which is due to its rigorous approach to co-simulation. MECSYCO uses an architecture that incentivizes modularity, domain flexibility, and extensibility, which are three of the framework objectives. Now, we describe the way in which MECSYCO implements these three elements.

The co-simulation description in MECSYCO uses a formalization of the Agents and Artifacts paradigm for the representation of a multi-model (also known as a coupled model described in Section 2.2.3) [188]. MECSYCO uses a DEVS wrapper strategy, which means that instead of directly writing or transforming the models into DEVS, it provides the additional elements necessary to bridge the models with the DEVS abstract simulation structure [189]. The MECSYCO approach uses the DEVS formalism as the co-simulation guideline [190] to ensure the modularity of the DEVS coupled models as described in Definition 10. We intend to leverage this conceptual fidelity to exploit the modularity compatibility within the co-simulation component with the general framework implementation.

The construction of a multi-model in MECSYCO allows for including heterogeneous models, that is, models coming from different external software that require the definition of their DEVS wrapper to be included in multi-models in a transparent manner. MECSYCO counts with implemented DEVS wrappers for the MAS simulator NetLogo [191], the telecommunication network simulators NS-3 and

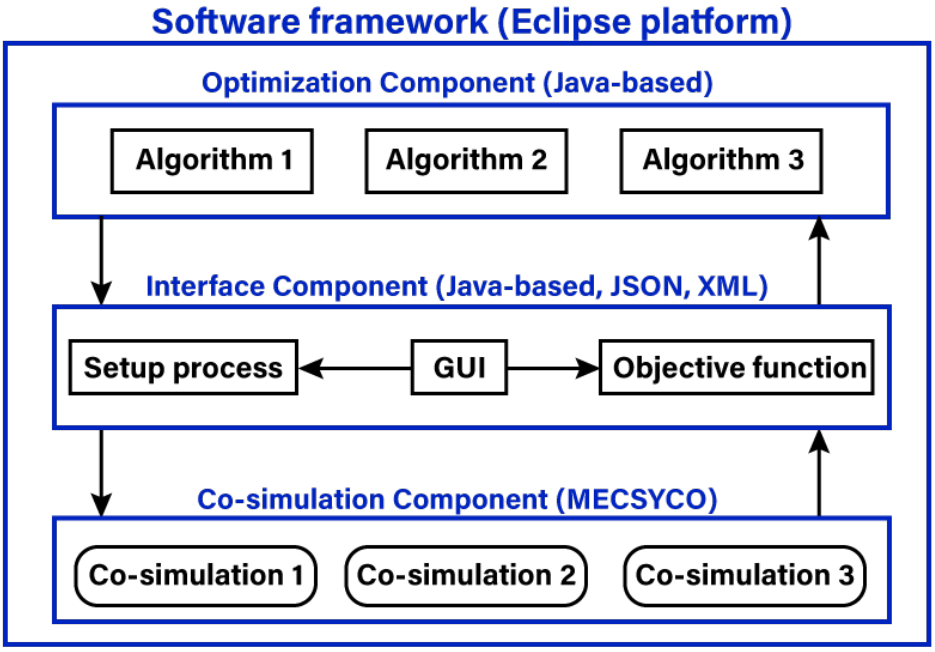


Figure 4.4: Software framework implementation structure, showing the implementation tools used to handle each of the components as well as the overall framework environment platform. For the optimization and co-simulation components, we illustrate the possibility of containing several elements to choose from to perform the optimization process.

OMNeT++/INET [192, 193], the import/export FMU standard (Definition 21) [194, 189, 195], Java-based model artifacts coded in Java [6], Neural Network API exchange using ONNX [196], SQL [197], and MATLAB/Simulink [197]. This model heterogeneity showcases the extensibility (Definition 23) and domain flexibility (Definition 26) of the MECSYCO software, which increases the compatibility with the framework approach. Furthermore, integrating a new simulator in MECSYCO requires defining the wrapper by means of 5 functions. Once done, the new simulator behaves like a DEVS atomic model [191].

The MECSYCO co-simulation file uses the XML format and can be executed using a Java-based launcher. This description file includes details on the co-simulation, multi-model, and atomic models. The approach of the framework is modularity, which we exploit in the co-simulation component by leaving the generation of the co-simulation file (XML) to MECSYCO (which means that no additional development is needed for the co-simulation component in this implementation), and the Java-based launcher configuration is relegated to the interface component; this means that the co-simulation iteration and execution are processes that require development efforts to make the integration of MECSYCO into the framework; these processes are handled from the Cosim Interface, which is located in the interface component and explained in subsection 4.4.4.

For this implementation case, we use the Java version of MECSYCO [186], which is deployed in the Eclipse environment as seen at the bottom of Figure 4.4.

4.4.3 Optimization component implementation

To implement the Black-box optimization concept into the software framework, we use a Java implementation of the concept, that is, a Java class that operates as a black-box optimization algorithm. For this, we propose the class structure from the UML diagram in Figure 4.5; the algorithm implemented (we use *AlgorithmX* as an example) has some parameters dependent on the algorithm logic requirements (examples will be shown in the use cases in the next chapter), and at least three methods:

- The stopping criterion method is a validation method usually included in the logic of an optimization algorithm. This method determines if the optimization process has finished, returning a Boolean value $\{True, False\}$.
- The finish method is in charge of the instructions necessary to conclude the optimization process, for example, the log generation of the scenarios tested along with the results obtained per scenario, and the identification of the optimal scenario found.
- The iteration method is the method that will be called in each iteration of the optimization process to calculate the new scenario. It calls the previous two methods to determine when and how to execute the end of the optimization process.

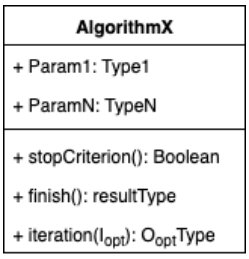


Figure 4.5: Java class UML diagram of the general structure of any black-box optimization algorithm implemented into the framework.

We are particularly interested in the iteration method, as the majority of the algorithm definition is defined in this method. The advantage of a programming language like Java is the possibility of using plugins to include external tools such as the Hexaly optimization service [198]. Hexaly is an open-source service for optimization that offers a Java-based API to define and solve optimization problems. We focus on the *external functions* feature, which is conceived for solving black-box optimization problems; therefore, we can implement the API request in the iteration method to effectively implement the Hexaly plugin into the framework implementation case.

4.4.4 Interface component implementation

The implementation of the interface component is in charge of organizing the internal interfaces, launchers, and user interface. We present the general structure of the interface component implementation in Figure 4.6. We see that the interface component is divided into two parts: the iteration launcher and the batch configuration helper.

The iteration launcher contains the core functionality of the interface, which is the connection of the co-simulation with the optimization component, and the co-simulation-based optimization launcher (called *cosim-opt launcher*). The batch configuration helper is an additional functionality of the interface component that proposes the use of a co-simulation-based optimization description to batch the execution of several instances; this is the result of the pre-processing information identification defined in subsection 4.2.4.

Optimization Interface implementation (Optimizer Interface)

We present at the right of Figure 4.6 the optimization component implemented into a Java class (as discussed in subsection 4.4.3) that is handled by an specific optimization interface, this means that for every algorithm included in the framework, it is necessary to perform the algorithm class and its respective interface, the *AlgorithmX* class in the graph represents the generalization discussed in Figure 4.5 that contains the logic of the algorithm, and the *optimizer interface* serves as the connector between the algorithm and the interface component. This is the location of the objective function $ObjFunc(\cdot)$ that we described in Equation 4.7. In the general framework's definition of the optimization component (subsection 4.2.2), we mentioned the importance of the objective function for the optimization algorithm.

For this reason, the optimizer interface has the capability of using the algorithm as a Java class and to calculate the objective function value I_{opt} based on the co-simulation outputs O_{cs} (the $ObjFunc(\cdot)$ dynamics within the framework were already described in Table 4.2 and Figure 4.3).

Co-simulation interface implementation (Cosim Interface)

In the same way as the optimization component, the co-simulation interface is represented at the left of Figure 4.6. We discussed in subsection 4.4.2 that the interface component is in charge of handling (configuration and launching) of the co-simulation process. This is done in the *Cosim interface*, which serves as the connector of the interface component with the co-simulation component. Notably, in this interface, we find the $Setup(\cdot)$ process described in Equation 4.10 responsible for translating the algorithm scenario description O_{opt} into the multi-model N and parameters I_{cs} that will be used in the co-simulation process.

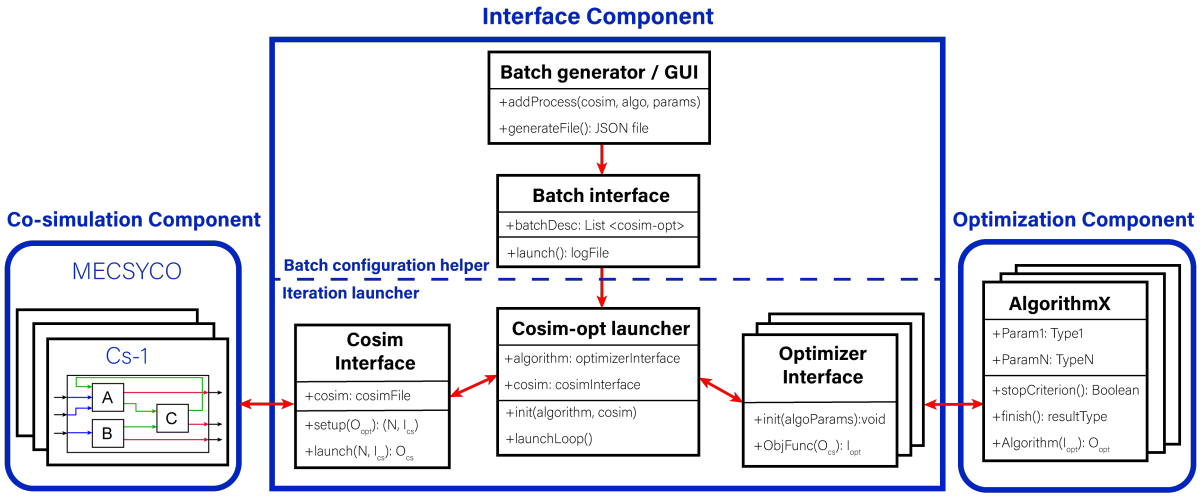


Figure 4.6: Overview of the interface implementation, the different internal interfaces converge to the co-simulation-based optimization process launcher (cosim-opt launcher).

Iteration launcher (Cosim-opt launcher)

In Figure 4.6, we represent the launcher of the co-simulation-based optimization process with the class Cosim-opt launcher, which uses the co-simulation and optimization interfaces previously defined to integrate the methods in the same loop process formally. Further validation processes for the compatibility of the co-simulation domain and/or the optimization algorithm can be defined within the interfaces, which would allow the iteration launcher to remain generic and would relegate the particular validations of each module to its own interface method.

On one hand, the implementation of the optimization component was performed using an object-based definition per algorithm; the interface also requires the definition of an optimizer interface per algorithm (which needs the algorithm parameters to instantiate it). On the other hand, the use of the MECSYCO software facilitates the interface task at the point of making possible the use of a unique co-simulation interface that requires the MECSYCO co-simulation file to initialize and launch any co-simulation. This modular definition allows the use of other co-simulation tools if necessary.

The co-simulation-based optimization launcher (cosim-opt launcher in Figure 4.6) is where the main loop of the process is executed, that is, the iterative optimization process seen in the framework structure from 4.3. To launch this loop, we require as parameters: the algorithm to be used (needed to instantiate the correct optimizer interface file), and the cosim file (used to instantiate the correct co-simulation using the cosim interface). With these parameters set, the launcher can be initialized and launch the loop process.

Batch configuration helper

The batch configuration helper is included in this implementation as a tool for simplifying the execution of several iterations of the process, which is made possible by the pre-processing information identification (see Definition 28 and subsection 4.2.4). This means that we have generalized the process to the point that we can describe a co-simulation-based optimization (Cosim-Opt) process iteration with a few key elements. The result of this generalization is the possibility of generating a queue of Cosim-Opt iterations, which can be executed in series.

For the implementation performed, we chose the JavaScript Object Notation (JSON) as a lightweight data-interchange format [199] to be used for the definition of the Cosim-Opt interactions. The structure proposed for the description is shown in the following example.

Example 12. *Let's consider the optimization of a single co-simulation using two different algorithms. For this, we define the pre-processing information as:*

- *Co-simulation file:* ***MECSYCO-cosim-file.xml****.*
- *Co-simulation parameters:*
  - ***CSparam1****.*
  - ***CSparam2****.*
- *Optimization objective value:* ***Objective1****.*
- *The algorithm keyword with its respective parameters and arbitrary value as an example:*
  - ***algorithm1****:*
    - ***Algo1Param1****: 0.1.*
    - ***Algo1Param2****: 0.01.*
    - ***Algo1Param3****: 1.0.*
  - ***algorithm2****:*
    - ***Algo2Param1****: 0.5.*
    - ***Algo2Param2****: 0.08.*

We put in bold the elements that are directly included in the configuration file. To describe the two Cosim-opt iterations, we define the following batch configuration file.

```
{
  "optimization_instances" : [ {
    "opt_name": "algorithm1",
    "opt_parameters" : {
      "indicators" : [ "Objective1" ],
      "cosimulation_parameters" : [ "CSparam1", "CSparam2" ],
      "Algo1Param1" : 0.1,
      "Algo1Param2" : 0.01,
      "Algo1Param3" : 1.0
    },
    "experiment_path" : "MECSYCO-cosim-file.xml"
  }, {
    "opt_name": "algorithm2",
    "opt_parameters" : {
      "indicators" : [ "Objective1" ],
      "cosimulation_parameters" : [ "CSparam1", "CSparam2" ],
      "Algo2Param1" : 0.5,
      "Algo2Param2" : 0.08
    },
    "experiment_path" : "MECSYCO-cosim-file.xml"
  } ]
}
```

The simplification of the description proposed with the batch configuration JSON helps the user focus on the important elements to be set in a cosim-opt process configuration; however, the correct generation of the JSON file remains the user's responsibility. For this reason, we suggest the development of a graphical user interface (GUI) to guide the user along the cosim-opt batch creation.

4.5 Conclusions

In this chapter, we defined the co-simulation-based optimization framework, whose structure comes directly from the architectural patterns found in the literature in chapter 3. These elements are flexibility, extensibility, modularity, pre-processing, and the interface component. The structural elements, such as the modularity (with a separation in boxes of the components), pre-processing (listing the information requirements), and the interface component (intermediate component to handle the communication between co-simulation and optimization), are visible in Figure 4.2.

However, the extensibility and flexibility characteristics of the framework depend on the implementation process; for this reason, we include the implementation use, which presented the co-simulation software working under a black-box architecture as a multi-disciplinary approach to include flexibility into the software framework. Additionally, we presented the implementation case of the framework in a highly versatile environment, that is, a Java-based environment, where the co-simulation and optimization components are implemented and encapsulated as usable Java objects. The core implementation effort is focused on the interface component, where we define interfaces that allow the use of the other components as plugins (as seen in Figure 4.6). The result of the implementation is a co-simulation-based optimization framework, where the user interactions converge in the interface component, mainly for the configuration of a Cosim-opt process (in the Iteration Launcher) and for the configuration of a Cosim-opt batch (in the Batch Configuration Helper).

We provided the general architecture of the framework with components and information flow in Figure 4.2, the notation used for the processes and interaction is summarized in Table 4.2, the dynamics of the framework using the mathematical notation in Figure 4.3, and finally the implementation case architecture in Figure 4.4. Nevertheless, to fully understand the utility of the framework, we require applications that can showcase the usage of the different framework implementation features. For this, we present use cases intended to explore the functionalities of the framework while studying and improving some complex systems; the definition and assessment of the use cases are provided in the next chapter.

Chapter 5

Assessment

Contents

In this chapter, the assessment of the framework is presented. For this, we describe the use cases used to test and showcase the functionalities of the framework, and then we perform the analysis of the results to compare against the literature the contributions achieved. First, we present the use cases developed in the co-simulation component implementation in section 5.1. Second, we present the use cases developed in the optimization component implementation in section 5.2. Third, we describe the usability of the interface component for creating experiments with the use cases in section 5.3, which showcases the simple usability of the software framework implemented for co-simulation-based optimization. Next, we start the assessment of the results obtained from the use cases in 5.4, which is done across the three levels of development described in subsection 4.1.2.

5.1 Co-simulation component use cases

The co-simulation component is handled with the MECSYCO software discussed in subsection 4.4.2, which offers the tools to develop a co-simulation. The idea is to showcase the flexibility of the framework by developing three use cases, which correspond to three co-simulations of different complex systems to be studied and improved. We provide a description of the target system, the scenario description characteristics, and the optimization objective for each use case.

5.1.1 First co-simulation use case: house thermal control

The first use case intends to showcase a complex system studied using co-simulation. For this, we consider a straightforward problem of the control of the internal temperature of a house, part of the MECSYCO tutorial examples [186, 190, 200]. With this use case, we use the ability of MECSYCO to represent a complex system in a multi-model. We describe the details about the multi-model, then the way that each scenario of the co-simulation is defined, and finally, the optimization interest of the complex problem.

Multi-model

The target system (Definition 1) of the first co-simulation use case is a house's air conditioning control system. We adapted a co-simulation [25] that includes the interactions between the house temperature behavior, the outside temperature (weather), an air conditioning system, and the occupancy of the house. Figure 5.1 shows the way the interconnected sub-systems represent the house thermal regulation system. Each sub-system is described as follows:

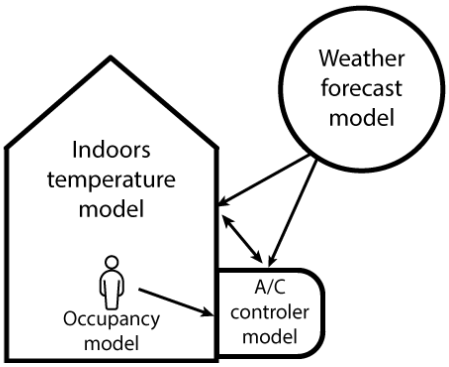


Figure 5.1: Use case 1: house air conditioning co-simulation.

- The first model represents the thermal behavior of the house, this means that it is a model that generates the **indoors temperature** of the house, this value is affected by external factors such as the external temperature that interacts with the walls of the house and the air conditioner system of the house that introduces cold or hot air into the house to modify the house temperature. The house model is developed in Open Modelica [126], then exported using the Functional Mock-up Interface (FMI) standard [107], and finally imported into the Eclipse environment to be used in the MECSYCO multi-model.
  - *(Inputs)* ET_t: Exterior temperature at moment t. P_t: power consumption used at moment t.
  - *(outputs)* T_t: internal temperature of the house at moment t.
- The **weather model** provides the exterior temperature value to the house; these values are provided in a CSV file containing the weather of the city of Nancy, France, from December 13th to 19th, 2021.
  - *(Inputs)* No inputs.
  - *(outputs)* ET_t: Exterior temperature at moment t.
- The **air conditioning model** decides whether to turn on or off the acclimatization system based on the current state of the house and the acceptable temperature boundaries of the system to cool or heat the house. The air conditioning system also considers the occupation of the house in the decision; that is, if the house is not empty, the air conditioning system remains off. The model is a Java-based code that follows the logic presented in Figure 5.2. This model has the output

- *(Inputs)* T_t: internal temperature of the house at moment t.
- *(outputs)* P_t: power consumption used at moment t.
- *(Parameters)* T_{min} and T_{max}: Represent the interval of the acceptable temperature boundaries for the house.

- The **occupancy model** is a report of the presence of people in the house model. This model provides the state of a house in terms of the absence of occupants or the presence of one or more occupants (a binary value $\{0, 1\}$). This model is a Comma-Separated Values (CSV) file that contains the report of occupancy of a house during one week.
  - *(Inputs)* No inputs.
  - *(outputs)* O_t occupation of the house at moment t.

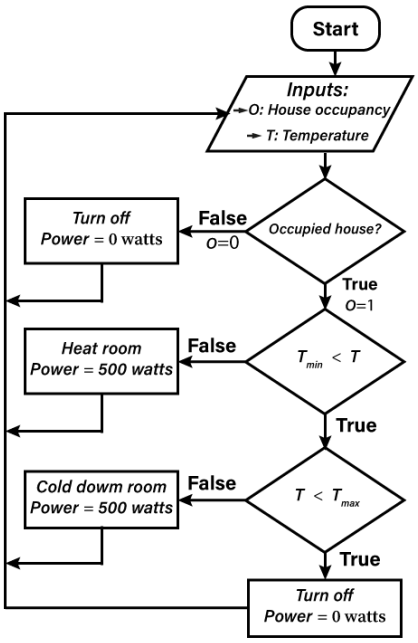


Figure 5.2: Air conditioner controller logic flowchart (taken from [5]).

Scenario definition and optimization objectives

Using the model described, we have the opportunity to test several scenarios to study the behavior of the air conditioner system under different configurations. We choose to define as a decision variable the configuration of the air conditioner model, concretely, the minimal and maximal temperatures (see values in Figure 5.2 as T_{min} and T_{max}), which renders this a parametric optimization problem. More details on the configuration of this multi-model can be found in [5].

To assess the air conditioner performance in the multi-model, we define 2 indicators: the total comfort is calculated with the equation

$$Comf_{final} = \sum_{t=1}^{N} |T_{ref} - T_t| \, O_t, \tag{5.1}$$

where the house's internal temperature during the co-simulation is represented with T_t, T_{ref} is the ideal indoor temperature of the house (we take $T_{ref} = 24,5$ as the reference temperature for the use case experimentation), O_t refers to the occupation value taken from the occupation model, t is the index of the timestamp of the co-simulation, which is executed during $N = 604800$ seconds. This equation

measures the difference between the ideal and the real temperature when the house is occupied to obtain a comfort indicator.

The second indicator is the consumption of the multi-model, using the electric power consumption of the air conditioner model as

$$Consu_{final} = \sum_{t=1}^{N} P_t, \tag{5.2}$$

where P_t is the power used by the air conditioning system at the second t of the co-simulation. Finally, we can define the objective function as f_1 with the equation

$$min \quad f_1(T_{min}, T_{max}) = w_1 \; Comf_{final} + w_2 \; Consu_{final}, \tag{5.3}$$

where we combine the indicator functions from equations 5.1 and 5.2 into one objective function to be minimized. We use $w_1 = 1$ and $w_2 = \frac{1}{300}$ to represent the weight coefficients for each indicator in the optimization process. The objective is to reduce the scale of the consumption value and have both indicators on the same scale; this way, they have approximately the same weight in the optimization process.

5.1.2 Second co-simulation use case: autonomous micro-grid

The second use case intends to showcase MECSYCO's ability to couple and reuse models as black-boxes. For this, we reuse a co-simulation developed during a Phd thesis [6]. The multi-model used in this co-simulation is particularly interesting as it uses nesting to reuse models within models without affecting the resulting co-simulation properties.

Multi-model

The target system of this use case is a network of autonomous houses connected to a shared storage system, that is, a co-simulation that represents the interactions between four houses that generate electricity through wind turbines, a coupling module, and a smart storage system that stores the surplus energy generated and provides electricity to the houses when the generated energy is not enough. This co-simulation is the result of the PhD thesis [6] and is shown in Figure 5.3. The multi-model contains three different systems interconnected and described as follows:

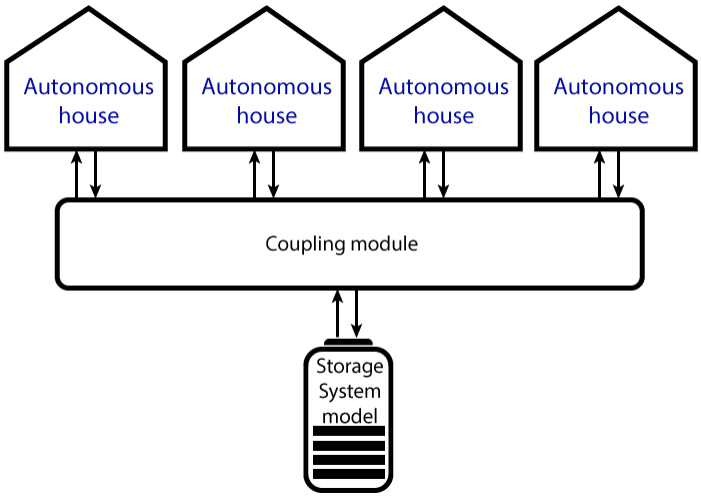
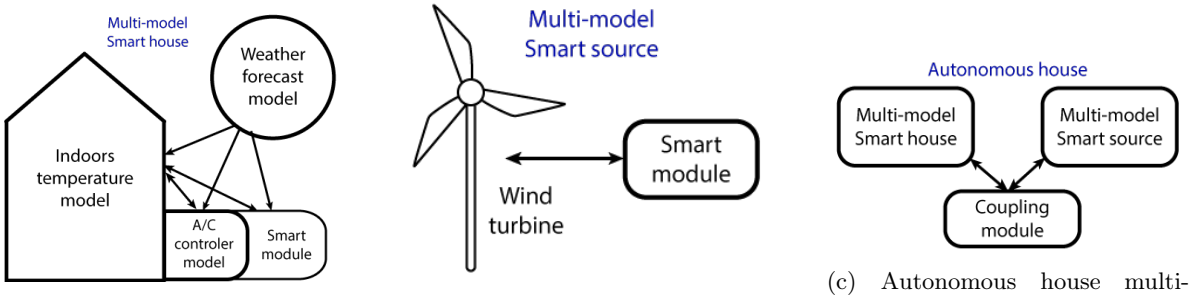


Figure 5.3: Use case 2: micro-grid co-simulation (adapted from [6]).

- The **Autonomous House model** is itself a multi-model, which is represented in Figure 5.4c. This multi-model is composed of three elements:
  - A **smart house multi-model**, which is illustrated in Figure 5.4a, where we see a similar configuration from the multi-model of the first use case, which is intentional, as we are reusing the indoor temperature model, the A/C controller model, and the weather models described in section 5.1.1. Note that in this case, the occupation model is not used; also, a smart module (which refers to a handling system in charge of making the decisions over another model) is added for the autonomous handling of the energy supply and demand.
  - A **smart source multi-model**, which is illustrated in Figure 5.4b and shows the connection between a wind turbine energy generator and a smart module.
  - A **coupling module** to handle the communication between the other two multi-models.

  The idea is to have several iterations of this model in the multi-model to represent a shared network formed with several houses close to each other.
  - *(Inputs)* In_t^i: Power instruction for the i model iteration at moment t.
  - *(outputs)* Pow_t^i: Power at moment t from the iteration i of the model.
- The **Storage System model** keeps the surplus energy from the houses to be provided in moments of high demand. This model generates a relevant output, that is, the amount of energy stored at the end of the co-simulation represented as ES_{final}, and it requires the storage system's maximal capacity as a parameter represented as SC.
  - *(Inputs)* InS_t: Power instruction for the storage System at moment t.
  - *(outputs)* $PowS_t$: Power of the storage System at moment t. ES_t: Energy stored in the system at moment t.
  - *(Parameters)* SC: Represent the storage system's maximum capacity.
- The **Coupling model** handles the equilibrium seeking in terms of energy for the system using a bidirectional constant communication between all the components. This model decides how to distribute the generated energy between the houses to meet the offers and demands from the houses. The model can decide when to store the surplus energy generated or to use the energy stored.
  - *(Inputs)* $PowS_t$: Power of the storage System at moment t. Pow_t^i: Power at moment t from the iteration i of the model.
  - *(outputs)* InS_t: Power instruction for the storage System at moment t. In_t^i: Power instruction for the i model iteration at moment t.



(a) Smart house multi-model. (b) Smart source multi-model. (c) Autonomous house multi-model.

Figure 5.4: Multi-models nested in the second use case co-simulation

The models of this use case were implemented using Java-based coding and Open Modelica models in combination with MECSYCO. More details of each model can be found in [6].

Scenario definition and optimization objectives

The multi-model presented in Figure 5.3 requires the value of the energy storage capacity SC to be used as a parameter in the co-simulation, which generates as output the energy stored at the end as ES_{final}. We use these values to calculate the state of charge as

$$SoC_{final} = ES_{final}/SC. \tag{5.4}$$

We set the objective function f_2 for this use case as finding the optimal Storage Capacity (SC) value to minimize the state of charge at the end of the co-simulation; however, to avoid the under- or over-sizing of the storage system, we set a penalty function to find the optimal storage capacity that minimizes the energy leftover at the end. The objective function f_2 is defined as

$$min\ f_2(SC) = SoC_{final} + p \cdot g(SoC_{final}) \cdot (-SC) \tag{5.5}$$

where the penalty factor is $p = 0.001$ and penalty function is $g(SoC_{final}) = \frac{1}{1+SoC_{final}}$.

5.1.3 Third co-simulation use case: Hy2Car

For the last use case, we intend to study and improve a complex system coming from a real-life problem; for this, we intervene in the Hy2Car project [201]. The Hybridized Hydrogen Car (Hy2Car) project was launched as a research initiative aimed at exploring a direct hybrid fuel cell (FC) and supercapacitors (SC) as an energy source for a vehicle.

The idea of using co-simulation-based optimization to study and improve the vehicle's design stage is to provide a platform for testing, validating, and optimizing the vehicle's performance. The expected study should consider electric, physics, control, and geographic domains to simulate and optimize the Hy2Car behavior. This work is a product of the collaboration of Professor Rael and engineering students Stanislas Mezureux and Vinicius Kamiya, who contributed to the design of several models.

Multi-model

The target system (Definition 1) of the third co-simulation use case is the vehicle using a combination of fuel cell and supercapacitor as an energy source [202]. We developed a multi-model with the objective of dividing the study of the vehicle into different sub-systems, as shown in Figure 5.5. The development of the multi-model is the result of the collaborative work with Professor of Electrical Engineering Stephane Rael. We describe the general purpose of each model as:

- The **engine model** focuses on the mechanical behavior of the vehicle, i.e., the way that the engine translates a certain power supplied from the energy source into the movement of the wheels of the vehicle.
- The **Hybrid source** model represents the behavior of the energy source, which combines the energy supply from a fuel cell and a supercapacitor [202].
- The **driver model** intends to make decisions over the vehicle to accelerate or decelerate depending on an objective velocity.
- The **path model** represents the path being traveled by the vehicle.

In the following, we describe the sub-models in detail for reproducibility reasons.

Using the division in sub-systems from Figure 5.5, the development of the co-simulation multi-model is performed in MECSYCO. The multi-model obtained is shown in Figure 5.6 using the notation defined in Figure 2.7. Each of the sub-models is described as follows:

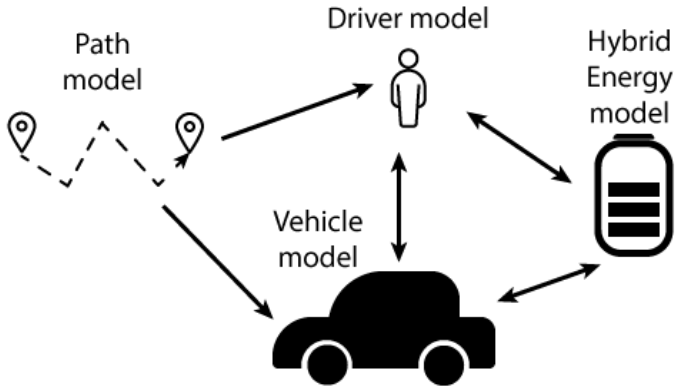


Figure 5.5: Use case 3: Hybrid vehicle sub-systems.

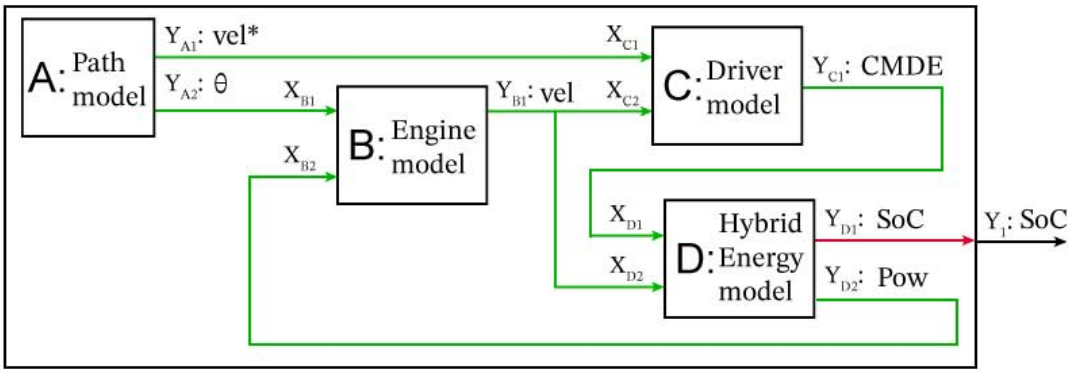


Figure 5.6: Use case 3: Multi-model structure for the Hybrid vehicle.

- The **engine model** M_B represents the mechanical behavior of the vehicle, mainly focusing on the engine, which uses a certain road inclination (input port $X_{B1} : \theta$) and amount of power (input port $X_{B2} : Pow$) to generate velocity (output port $Y_{B1} : vel$). The model calculates the change in the vehicle's velocity based on the previous velocity and the input information using the equation

$$v_t = \sqrt{v_{t-1}^2 + \Delta v_t^2}. \tag{5.6}$$

This equation calculates the velocity of the car at the moment t of the simulation. Where the calculation of Δv_t^2 is performed with the equation

$$\Delta v_t^2 = [\frac{2}{m}\ P_{meca} - 2\ g\ v_{t-1} sin(\alpha) - \frac{2}{m} fr(v_{t-1})]\Delta t. \tag{5.7}$$

Where we have values representing parameters and inputs. The parameters are: m, which is the total mass of the vehicle, and it is one of the co-simulation parameters to be defined in the scenarios, the gravity g, and the friction function $fr(v)$ that depends on the previous velocity and is calculated using the equation

$$fr(v) = \beta_1\ v + \beta_2\ v^2 + \beta_3\ v^3 \tag{5.8}$$

where $\beta_1, \beta_2, \beta_3$ are the coefficients of the friction function. For the Hy2Car project, we received the parameter values shown in Table 5.1 for the testing and optimization processes.

The inputs are represented as: P_{meca} as the power used, described as input port $X_{B2} : Pow$ in the multi-model figure, and α is the inclination of the path defined as input port $X_{B1} : \theta$, which has a positive value when the car is going uphill and negative for downhill.

Table 5.1: Parameter values for test simulations of the car model.

Parameter	Symbol	Value [units]
Gravity acceleration	g	9.81 $[m \cdot s^{-2}]$
Friction coefficient	β_1	0 $[kg \cdot m \cdot s^{-2}]$
Friction coefficient	β_2	7.2 $[kg \cdot s^{-1}]$
Friction coefficient	β_3	0.45 $[kg \cdot m^{-1}]$

- The **driver model** M_C takes as a reference the average velocity of the route to modify the current velocity of the car. We implement a PI controller to serve as the driver's behavior.

  Definition 29. *(PI Controller.) The Proportional-Integral (PI) control is a very popular control method. The model uses a Proportional action to either stabilize or enhance the convergence of an expected output. However, proportional control alone cannot remove the output steady-state offset when load disturbances affect the model. Therefore, an integral action is included to remove the steady-state offset [203].*

  The PI controller can modify an outgoing value (output port $Y_{C1} : CMDE$) to modify the value of another port, often called a real signal (input port $X_{C2} : vel$), to approach a desired value, often called a reference signal (input port $X_{C1} : vel*$). We will use the average speed of each route as a reference to change the velocity of the car. The expected behavior is an overall driving style of average speed.

  The implementation of the PI controller is developed in a Java-based model that defines the output value PI_{out} (represented as the output port $Y_{C1} : CMDE$) as

  $$PI_{out} = ke(t) + \frac{k}{T}\sum_{1}^{t} e(t)dt. \tag{5.9}$$

  Where k and T are known as the gain factor and integration time constant, respectively; to obtain the expected driving behavior, we set these parameters as $k = 3$ and $T = 0.333$. $e(t)$ is the error obtained by calculating the difference between the reference value R_t and the measured value V_t, that is,

  $$e(t) = R_t - V_t. \tag{5.10}$$

  Where the reference value is calculated with

  $$R_t = \frac{1}{2} \cdot m \cdot ref^2, \tag{5.11}$$

  and the measured value is calculated with

  $$V_t = \frac{1}{2} \cdot m \cdot mes^2, \tag{5.12}$$

  where the value ref comes from the input port $X_{C1} : vel*$, the value mes comes from the input port $X_{C2} : vel$, and m is the mass of the vehicle (same parameter from the engine model). The idea is to regulate the difference between the kinetic energies. By using the kinetic energy instead of the velocities, we linearize the output of the controller with respect to the control model input (mechanical power).

- The **path model** M_A is a model that contains the trip path to follow between two geo-located points (latitude, longitude, and altitude) representing the start and end points of the trip. The trip model calculates a path between the points and provides the road inclination (output port $Y_{A2} : \theta$ in Figure 5.6) and the average velocities for each segment of the trip as a reference to the driver to adapt the velocity accordingly (output port $Y_{A1} : vel*$). The model implemented is

a Java-based simulation that uses the OpenRoute service API [204] to obtain the path from the geo-located points A to B, the distance, and the average velocity of each trip segment. For the use cases testing, we use the trip from the ENSEM School in Nancy (latitude = 48.653053062204, longitude = 6.148272853885205, altitude = 124.3 [MASL]) to the train station in Nancy (lat = 48.69024512353616, lon = 6.174054070300207, altitude = 217 [MASL]).

- The **Hybrid source model** M_D is in fact a coupled model that we describe in detail in Figure 5.7, where we illustrate the following sub-models:
  - The **source model** M_F is a model that contains the power source model, the current power source model corresponds to a fuel cell coupled with supercapacitors. Data obtained from collaboration with Professor Stephane Rael to define the fuel cell behavior. This data provides the voltage for a single cell as a function of the current density J_{cell}, so the total voltage of the fuel cell (output port $Y_{F4} : Vfc$) is defined as

$$Vfc = f(J_{cell}) \cdot Nfc. \tag{5.13}$$

    Function f and value J_{cell} are taken from the data file provided by the Hy2car project, Nfc is the number of cells in series, which is calculated as

$$Nfc = \frac{28000}{Afc}. \tag{5.14}$$

    The calculation of the current of the fuel cell (output port $Y_{F3} : Ifc$) is performed using

$$Ifc = Afc \cdot J_{cell}. \tag{5.15}$$

    Where J_{cell} is the same value from equation 5.13, and Afc is the active area of the cell (same variable from equation 5.14), which, alongside Nfc, are the parameters that define the maximum Power of the fuel cell Pfc_{max} with the equation:

$$Pfc_{max} = 0.45[W \cdot cm^{-2}] \cdot Afc[cm^2] \cdot Nfc. \tag{5.16}$$

    With the $0.45[\frac{W}{cm^2}]$ representing the maximal power density of a cell.
    The voltage of the supercapacitor (output port $Y_{F2} : Vsc$) is calculated with

$$Vsc = Vfc - (Rw \cdot I_{fc}), \tag{5.17}$$

    where Vfc is the same as in equation 5.13, $Rw = 3,5m\Omega$ is the connection resistance between the fuel cell and the storage, and I_{fc} comes from equation 5.15. Additionally, we have the constitutive equation of the storage device as

$$Vsc = Vc - (Rc \cdot \frac{N_s}{N_p} \cdot I_{sc}), \tag{5.18}$$

    where Vc is the voltage across the capacitive element, $Rc = 0.15m\Omega$ is the internal series resistance of the supercapacitor, N_p represents the number of supercapacitors placed in parallel (this is another design parameter of the model), N_s represents the number of supercapacitors placed in series, defined by the equation

$$Ns = 1 + floor(Nfc \cdot E_0/Vc_{nom}), \tag{5.19}$$

    where the $floor(\cdot)$ function takes the integer part of a value, Nfc is the same used in equation 5.16, E_0 is the the open-circuit voltage of a cell (considered as $E_0 = 0.972[V]$ for the project), and Vc_{nom} is the nominal voltage of a supercapacitor element (consider as $Vc_{nom} = 3[V]$ for the project). And I_{sc} is defined by equation

$$I_{sc} = -(C \cdot \frac{N_p}{N_s}) \cdot \frac{\partial Vc}{\partial t}, \tag{5.20}$$

where $C = 3400[F]$ is the internal capacity of the supercapacitor, and N_p, N_s, and Vc are the variables from equation 5.18.
Finally, the current coupling equation

$$Ih = Ifc + Isc, \tag{5.21}$$

where Ifc is taken from equation 5.15, and Isc is the current of the supercapacitor from equation 5.20.

This model is developed using the software OpenModelica [126] and then exported using the FMU standard to be directly imported into the MECSYCO co-simulation software.

– The **control model** M_E is a model that contains the controller of the source model. This model is the way to make the source respect the constraints of the hybrid source system. The model defines the current nominal values Ih_{nom}, Isc_{nom}, the maximum load voltage Vsc_{max}, which is defined by

$$Vsc_{max} = Vsc_{nom}, \tag{5.22}$$

where we use Vsc_{nom} as the highest value that can be given to Vsc_{max}. For the minimum discharge voltage Vsc_{min} is defined as

$$Vsc_{min} = Vfc_{nom} - Rw \cdot Ifc_{nom}. \tag{5.23}$$

Where Rw is the value from equation 5.17, Vfc_{nom} is the nominal voltage of the fuel cell, defined by

$$Vfc_{nom} = Nfc \cdot Vc_{nom}, \tag{5.24}$$

with Vc_{nom} representing the nominal voltage of a cell unit (the value is set as $Vcnom = 3[V]$ for the Hy2car tests), and Ifc_{nom} is defined using the equation 5.15 but using the nominal value of J_{cell}.
The model calculates the power supplied by the power source using constraints and the power requested by the driver model through the input port $X_{E1} : CMDE$. With this value and the actual voltage of the supercapacitive storage device (input port $X_{E2} : Vsc$), we calculate the output port $Y_{E2} : Ih^*$ using the equation

$$Ih^* = \frac{Ph_{nom}\ \mathrm{CMDE}}{Vsc}, \tag{5.25}$$

where:

* Ph_{nom} is the nominal power from the hybrid source. In general, the term "nominal´´ refers to the normal continuous power that a device is designed to provide under standard operating conditions, without suffering overheating or abnormal deterioration. This value is calculated with the equation

$$Ph_{nom} = Ih_{nom} \cdot Vsc_{nom}, \tag{5.26}$$

where $Ihnom$ is a parameter that is constrained by the fact that we want to maintain the same proportion from a base value where $Nfc = 280$ is assumed, which gives a value of $Ihnom = 125[A]$. Therefore, to maintain this inverse proportion, we use the equation:

$$Ih_{nom} = 125 \cdot \frac{280}{Nfc}. \tag{5.27}$$

Also, $Vscnom$ is calculated with

$$Vscnom = Vc_{nom} \cdot N_s, \tag{5.28}$$

where Vc_{nom} is the same value from equation 5.24 and N_s represents the number of supercapacitors placed in series (defined in equation 5.19).

* The top part of the fraction (Ph_{nom} CMDE) is limited by interval $[\lambda_v Ph_{min}, Ph_{max}]$ previous to the application of equation 5.25.
  - Ph_{min} represents the minimal power that the hybrid source can provide at a certain moment in time. We note that $Ph_{min} < 0$, since the physical interpretation of Ph is the power provided by the source, the $-Ph_{min}$ value represents the maximum power that the hybrid source can receive (the battery is recharging). This value is calculated with the equation:

$$Ph_{min} = max(-1 \cdot Vsc \cdot Ih_{nom}, Pfc + Psc_{min}) \tag{5.29}$$

    where Pfc is defined by the input ports $X_{E3} = Ifc$ and $X_{E4} = Vfc$ as

$$Pfc = Vfc \cdot Ifc, \tag{5.30}$$

    and Psc_{min} is defined by

$$Psc_{min} = Vsc \cdot min(0, max(-Isc_{nom}, \frac{Vsc - Vsc_{max}}{10} \cdot Isc_{nom})). \tag{5.31}$$

    The Isc_{nom} is equal to

$$Isc_{nom} = N_p \cdot Ic_{nom} \tag{5.32}$$

    with $Ic_{nom} = 140[A]$ for the Hy2car testing, and N_p is the number of parallel branches in the storage system (used as a design parameter of the co-simulation).
  - Ph_{max} represents the maximal power that the hybrid source can provide at a certain moment in time; this value depends on the equation:

$$Ph_{max} = min(Vsc \cdot Ih_{nom}, Pfc + Psc_{max}), \tag{5.33}$$

    where Pfc is defined in equation 5.30, and Psc_{max} is defined by

$$Psc_{max} = Vsc \cdot max(0, min(Isc_{nom}, \frac{Vsc - Vsc_{min}}{10} \cdot Isc_{nom})). \tag{5.34}$$

  - λ_v is a limitation for energy recovery at low speed, it is defined as

$$\lambda_v = min(K \cdot vel, 1), \tag{5.35}$$

    with $K = 0.1$ and the velocity vel in $[km \cdot h^{-1}]$ for the Hy2car testing.
* After the calculation of equation 5.25, the result is limited by the interval $[-Ihnom, Ihnom]$, using the value $Ihnom$ from equation 5.26.

The source current demanded ($Ih*$) by the car will modify the source value of Ifc, Isc, Vfc, and Vsc, which will be used to calculate the electrical power provided to the round model using the equation

$$P_{elec} = Ih * Vsc. \tag{5.36}$$

Finally, we are assuming that the current regulation has a unit transfer function, which means that

$$Ih = Ih^*. \tag{5.37}$$

For the Hy2Car, the modelling of the current regulation is not necessary in the short term; however, it is expected to be added later.

– The **energy loss model** M_G is a model that handles the electro-mechanical conversion efficiency (associated with energy losses induced by the conversion of the electrical power Pow_{elec} into the mechanical power Pow). For the Hy2Car project, we use the value $\eta = 0.95$ as a conversion factor. This model calculates the output port $Y_{G1} : Pow$, which follows the equation

$$Pow = \begin{cases} \frac{P_{elec}}{\eta}, & \text{if } P_{elec} < 0 \\ P_{elec} \cdot \eta, & \text{if } P_{elec} \geq 0. \end{cases} \tag{5.38}$$

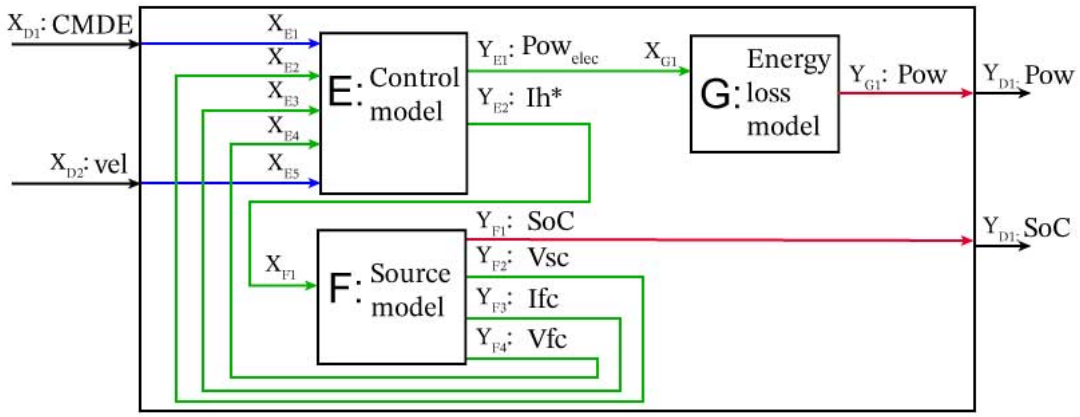


Figure 5.7: Hybrid Energy model Multi-model structure.

Scenario definition and optimization objectives

For the scenario description, we recapitulate the design variables present in the multi-model, which are the area of each fuel cell Afc, the number of fuel cells in series Nfc, and the number of branches in parallel in the supercapacitor N_p. These parameters are found in the source M_F and control M_E models. However, the modification of these parameters can affect the total mass of the vehicle, which is the parameter m found in the engine model M_B. We define the total mass of the vehicle as a function of the values Afc, Nfc, N_s, and N_p from models M_F and M_E, by using the equation

$$m = 1000 + (m_{Afc} \cdot Afc \cdot Nfc) + (N_p \cdot N_s \cdot m_{sc}), \tag{5.39}$$

where $m_{Afc} = 0.003[kg \cdot cm^{-2}]$ is the coefficient to calculate the mass from the area and number of fuel cells, and $m_{sc} = 0.75[kg]$ is the mass of each supercapacitive element. The values of these parameters are not yet validated, but we will use them for testing purposes; however, the multi-model will rest as a Hy2car project asset to test in the future with validated values for the coefficients.

The optimization parameters are therefore Afc, Nfc, and N_p. By trying different combinations of these parameters, we expect to explore the behavior of the vehicle.

The multi-model shown in Figure 5.6 contains many sub-models with different mathematical equations, from which we have a particular interest in the energy performance of the vehicle; for this, we set as indicators of the vehicle's performance the consumption and efficiency of the hybrid energy source of the Hy2Car. We start with the final State of Charge (SoC) as an indicator to be maximized, which is calculated as

$$SoC_{final} = 100 \cdot (Vsc_{final}/Vscnom). \tag{5.40}$$

The second indicator is the vehicle efficiency in terms of consumption and distance traveled, which we want to minimize; for this, we use the equation

$$Perform_{final} = \frac{\sum_{t=1}^{T}(Pow_t \cdot \Delta t)}{dist_{final}} \tag{5.41}$$

Finally, the objective function f_3 is the combination of equations 5.40 and 5.41 using the weight coefficients $w_1 = 1$ and $w_2 = -1$ to bring the same scale and optimization direction to both indicators. We intend to search for the optimal configuration of the Hy2Car under the conditions set in the parameters of the project using the equation:

$$min\ f_3(Afc, Nfc, N_p) = w_1 \cdot Perform_{final} + w_2 \cdot SoC_{final}. \tag{5.42}$$

The meaning of the objective function equation is that the optimal configuration of the vehicle results in the improvement of the energy consumption efficiency while increasing the energy saved at the end of the trip.

5.2 Optimization component use cases

The optimization component contains the optimization algorithms, which are restricted to algorithms compatible with the black-box optimization structure (as described in subsection 4.2.2). We performed the implementation of three algorithms as use cases to show the usage of the optimization component.

5.2.1 First optimization use case: Simulation-based optimization algorithm

Simulation-based optimization refers to the optimization of an objective function by the definition of a relatively small sample of scenarios to simulate and compare, that is, scenarios with variation of some parameters/policies to achieve an optimal value [10]. This method is also referred to as ranking and selection or scenario analysis, and it is widely used in simulation-based optimization [205]. Despite the simplicity of this algorithm, it is very useful when dealing with complex simulations or co-simulations. The literature review used the simulation-based optimization in 43 of the 96 articles found (representing 44% of the literature sample).

With the objective of generalizing the algorithm, we describe the algorithm in terms of a few parameters that define the optimization algorithm configuration, that is, simplifying the definition of the optimization scenarios to four elements.

- The number of scenarios to be tested N_{sb}.
- The set O_{opt} that contains the values for the scenario description (recalling the framework notation used to represent the scenario description in subsection 4.2.2).
- The initial values for the set elements, represented with the set O_{opt}^{init}.
- The step for each a-th element from the set O_{opt} represented as $step_a$. For example, for a set with three elements $O_{opt} = O_{opt}^1, O_{opt}^2, O_{opt}^3$, there are the same number of steps assigned to each element $step_1$, $step_2$, and $step_3$.

In the MECSYCO software implementation, the initial values O_{opt}^{init} are included in the multi-model definition, which means that they are set in the co-simulation component.

The implementation of this algorithm is performed in a Java-based class following the structure established in Figure 4.5. We describe the application of this algorithm to each of the co-simulation use cases described in section 5.1.

Application to the co-simulation use case 1

To apply this algorithm to the first co-simulation use case (subsection 5.1.1), the definition of the parameter values is done by modifying the interval of accepted temperature (T_{min}, T_{max}) and observing the optimization results of the objective function 5.3. We set the algorithm parameters as $N_{sb} = 12$, $O_{opt} = \{O_{opt}^1 : T_{min}, O_{opt}^2 : T_{max}\}$, $O_{opt}^{init} = \{16.0, 28.0\}$, $step_1 = 0.5$, and $step_2 = -0.5$. The resulting scenario generation for the co-simulation use case 1 is presented in Table 5.2.

Application to the co-simulation use case 2

Regarding the second use case (subsection 5.1.2), we define the amount of iteration at $N_{sb} = 20$, the only parameter to be set is the maximum capacity of the battery $O_{opt} = \{O_{opt}^1 : SC\}$ with an initial value of $O_{opt}^{init} = \{280.000\}$ and the step as $step_1 = -5000$, this means that the initial value is gradually reduced to study the behavior of the objective function. The generated scenarios from these configurations are displayed in Table 5.3.

Scenario	T_{min} [º Celsius]	T_{max} [º Celsius]
1	16.0	28.0
2	16.5	27.5
3	17.0	27.0
4	17.5	26.5
5	18.0	26.0
6	18.5	25.5
7	19.0	25.0
8	19.5	24.5
9	20.0	24.0
10	20.5	23.5
11	21.0	23.0
12	21.5	22.5

Table 5.2: Scenarios for the simulation-based algorithm in the co-simulation use case 1.

Scenario	Storage capacity [W]	Scenario	Storage capacity [W]
1	280000	11	230000
2	275000	12	225000
3	270000	13	220000
4	265000	14	215000
5	260000	15	210000
6	255000	16	205000
7	250000	17	200000
8	245000	18	195000
9	240000	19	190000
10	235000	20	185000

Table 5.3: Scenarios for the simulation-based algorithm for co-simulation use case 2.

Application to the co-simulation use case 3

We apply this algorithm to explore the third co-simulation use case (described in subsection 5.1.3), to define the configuration of the hybrid vehicle, we follow baseline values for the parameters provided by Professor Rael. We vary slightly from this configuration to understand their influence on the energy performance of the vehicle.

With this purpose, we define for the first part the number of iterations as $N_{sb} = 5$, the set $O_{opt} = \{O^1_{opt} : Afc, O^2_{opt} : Nfc, O^3_{opt} : N_p\}$, the initial values as $O^{init}_{opt} = \{200, 140, 1\}$, and the step as $step_1 = 0$, $step_2 = 0$, and $step_3 = 1$, which means that we start by increasing the number of parallel supercapacitors from 1 to 5 and maintaining static values for the other two parameters. For the second part, we use the same parameters except the initial values, which now change to $O^{init}_{opt} = \{100, 280, 1\}$. The generated scenarios from these configurations are displayed in Table 5.4.

5.2.2 Second optimization use case: Gradient descent algorithm

The gradient descent optimization algorithm uses the gradient of the result obtained from the co-simulation to explore possible scenarios [206]. The gradient descent algorithm uses parametric optimization to adjust one or more parameters to obtain an optimal value. It calculates the gradient on the results of the co-simulation iteratively to find the slope of the function curve and approach the optimal value, using n to notate the number of iterations of the algorithm. The gradient descent logic is widely known; however, since we performed a coded implementation of the algorithm, we include the definition followed during the implementation process. We define the algorithm with the equation

$$\vec{O_{opt \to n+1}} = O_{opt \to n} - \psi \nabla \vec{I_{opt \to n}}. \tag{5.43}$$

Scenario	Afc	Nfc	N_p
1	200	140	1
2	200	140	2
3	200	140	3
4	200	140	4
5	200	140	5
6	100	280	1
7	100	280	2
8	100	280	3
9	100	280	4
10	100	280	5

Table 5.4: Simulation scenarios for the simulation-based algorithm in the co-simulation use case 3.

Where we reuse the notation used in chapter 4 as follows

- O_{opt} and $\vec{I}_{opt}$ recall the notation used in equation 4.8. We notice that the application of the gradient depends on other methods of the framework, such as $ObjFunc(\cdot)$ from equation 4.7 and $Cosim(\cdot)$ (from equation 4.1). Nevertheless, the advantage of the black-box optimization approach is that the algorithm is not concerned with these functions (it is the function of the interface component). We can simplify the calculation of the gradient in the special case where only one parameter is considered (when the size of the vector O_{opt} is 1), and using the scale factor S with the equation

$$\nabla I_{\vec{opt\to n}} = \frac{I_{\vec{opt\to n}} - I_{\vec{opt\to n-1}}}{O_{opt\to n} - O_{opt\to n-1}} \cdot S. \tag{5.44}$$

  However, when the size k of the vector $\vec{O_{opt}}$ is $k \geq 2$, the calculation of one iteration of the algorithm includes k partial executions of the co-simulation. We consider the function $iter(\cdot)$ that depends on $\vec{O_{opt\to n}}$ to generate $\vec{I_{opt\to n}}$. With this, we see the partial iterations in the equation

$$\nabla I_{\vec{opt\to n}} = \nabla iter(\vec{O_{opt\to n}}) = \begin{bmatrix} \frac{\partial iter}{\partial O^1_{\vec{opt\to n}}}(\vec{O_{opt\to n}}) \\ \vdots \\ \frac{\partial iter}{\partial O^k_{\vec{opt\to n}}}(\vec{O_{opt\to n}}) \end{bmatrix}. \tag{5.45}$$

- ψ represents the learning rate.

The algorithm generalization is performed using the following parameters:

- $\psi \in \mathbb{R}_+$ represents the learning rate from equation 5.43, which defines the velocity of adaptation of the gradient function.
- The scale parameter S is used to keep a similar dimension between the parameters and the objective function values.
- The stopping criteria P_c is the threshold under which the algorithm stops the calculation of the gradient, that is, when the equation

$$(I_{\vec{opt\to n}} - I_{\vec{opt\to n-1}}) > P_c \tag{5.46}$$

  is true, the algorithm stops exploring.

The implementation in the framework is performed in a Java-based class following the structure established in Figure 4.5. The application of the gradient descent algorithm to the use cases only requires the definition of the parameters to be used in each case; we define the values shown in Table 5.5.

Cosim Use case	Parameters	Initial values	Learning rate ψ	Stopping criteria P_c	Scale S
1	T_{min}, T_{max}	24.2, 24.7	0.002	0.01	0.1
2	SC	280000	0.5	0.01	5000.0
3	Afc, Nfc, N_p	100, 140, 2	0.5	0.01	10.0

Table 5.5: Gradient descent parameters for the co-simulation use cases.

5.2.3 Third optimization use case: LocalSolver-Hexaly

This use case carries a particular interest for the framework's extensibility feature, as we intend to link an external optimization tool to the framework, which means that the implementation effort is much less than the implementation process of the previous algorithms; however, it will bring a big contribution to the framework by leveraging an external optimization tool.

We decided to use the optimization solver offered by Hexaly (previously known as LocalSolver) [207], which was implemented in the framework environment, maintaining the defined structure from Figure 4.5. The Hexaly solver has already been used in the literature [208], and by different companies listed on the Hexaly website [198].

For the black-box approach of the framework, we use the object-oriented Java API of the Hexaly solver; we particularly exploit the Hexaly **external functions** feature, which allows us to use the co-simulation execution as an *external function* that the solver can call. The interest we set in Hexaly is due to the compatibility of the optimization strategy with the framework architecture.

The implementation of the Hexaly allowed us to exploit additional features of the solver that are not implemented in the other two algorithms, that is, the possibility of adding constraints to the optimization problem. The addition of constraints remains optional (with the exception of the lower and upper bounds of each parameter, which are mandatory); however, simple constraints can help the solver converge to an answer compared to an unconstrained search space. The generalization of the algorithm is performed with the following algorithm parameters:

- $Param_i$ represents the name of the $i-$parameter of the co-simulation.
- $lower_i$ represents the lower bound of the $i-$parameter.
- $upper_i$ represents the upper bound of the $i-$parameter.
- P represents the precision to be considered in the solution.
- The exploration constraint for the algorithm is defined with two elements, from which only one is necessary to provide:
  - The maximum number of iterations It_{max}.
  - The time limit given in seconds t_{lim}.

In Table 5.6, we see the algorithm parameters used for the experiment on each of the co-simulation use cases. We see the utility of the constraints in the first use case, where the constraint $T_{min} < T_{max}$ prevents the algorithm from exploring illogical scenarios of the co-simulation.

Cosim Use case	$Param_i$	$lower_i$	$upper_i$	P	It_{max}	t_{lim}	Constraints
1	T_{min}, T_{max}	16.0 , 16.1	27.9, 28.0	0.01	-	120	$T_{min} < T_{max}$
2	SC	200000	300000	0.01	30	-	-
3	Afc, Nfc, N_p	100 , 140, 1	200, 280, 5	0.01	100	1200	-

Table 5.6: Hexaly parameters for the co-simulation use cases.

We defined the optimization configuration for all the use cases in Tables 5.2, 5.3, 5.4, 5.5, and 5.6. This completes all the setup information required to start launching the co-simulation-based optimization experiments. Now, we present the way in which this configuration is handled in the framework implementation to launch the experiments, which is mainly located in the interface component.

5.3 Configuration of the experiments with the Interface component

As described in subsection 4.2.3, the interface component is the intermediary between the co-simulation and optimization components. The interface is in charge of passing the information from the co-simulation to the optimization algorithm and serving as the configuration tool for the user. The structure of the interface component implementation was presented in 4.4.4.

The objective of the interface component is to organize the process configuration, which, if we did not have an interface component, trying to generalize the process, we would have ad hoc interfaces as shown in Figure 5.8. The intended modularity of the framework is shown in the interface component implementation shown in Figure 5.9, where we see the same elements described in the implementation of Figure 4.6. The implementation structure deployed is the result of the conceptual general framework guidelines, which allowed us to configure experiments in a simple manner, rather than developing ad hoc interfaces for each cosim-opt configuration.

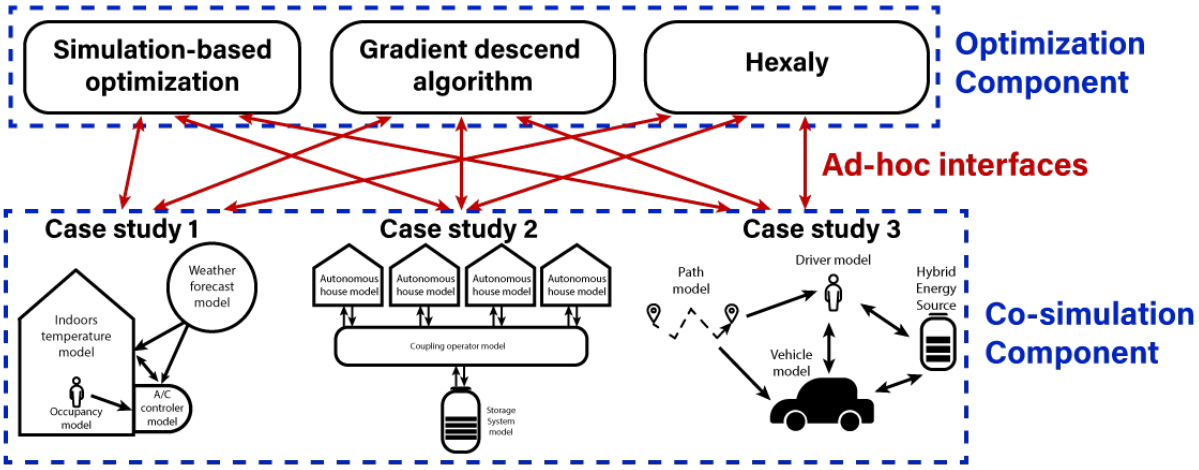


Figure 5.8: Overview of the use cases connected with ad hoc interfaces.

Each of the arrows going throughout the interface component requires the configuration of the launcher; this means that all the implementation effort has been condensed into a single document.

5.3.1 Usability of the interface component

In Figure 4.6, we discuss the batch configuration helper part of the interface component, whose purpose is the simplification of the interaction of the user with the framework; however, the user can still interact with the framework in 3 points.

- At **Cosim-opt launcher**, the user can directly create a cosim-opt launcher configuration and execute it. This interaction demands from the user knowledge of the Java programming language, as it requires going into a *.java* file and changing the values of 7-10 variables to later execute the Java file Cosim-opt launcher.
- At the **Batch interface**, the user can configure a JSON file (with the structure described in subsection 4.4.4) containing the description of 1 or more cosim-opt descriptions, which are launched

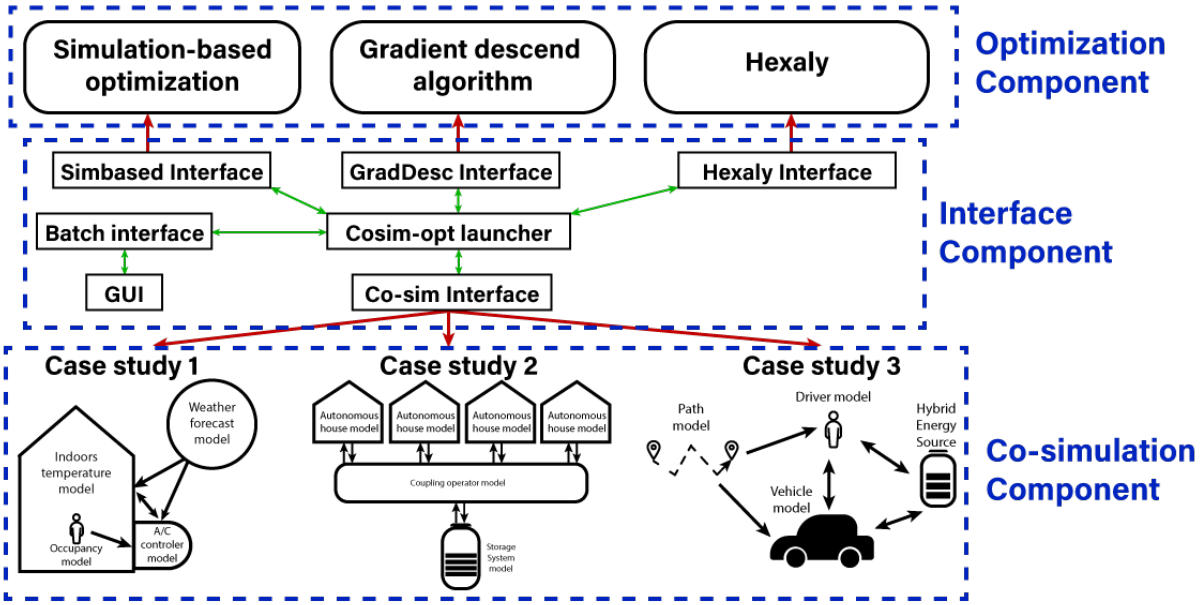


Figure 5.9: Overview of the use cases connected with the interface component from the implementation case developed for this thesis. We use red arrows to represent the connection between components, and green arrows for the connection between elements from the same component.

in series using the cosim-opt launcher. This interaction demands from the user a basic understanding of the JSON format to correctly modify the description file, and the capability of executing the Java file Batch interface.

- Using the **Graphical User Interface** (GUI) to create the batch of cosim-opt iterations. This interaction only demands that the user fill in the fields asked for in the GUI form.

5.3.2 Graphical User Interface structure and application

The structure of the Graphical User Interface (GUI) is presented in Figure 5.10, where we intend to summarize the relevant information to be decided by the user when configuring a co-simulation-based optimization experiment:

1. The co-simulation to be used. Due to the modularity of the framework, all the co-simulations available in the implementation are reachable in the same folder, which simplifies the task of the GUI of exploring the co-simulation files (field 3 of the Figure 5.10).

2. The optimization algorithm to be used. For this, the same principle applies, as the framework is organized in a manner that the algorithms are available in the same environment (under the structure defined in Figure 4.5) and should be reachable by the GUI (field 1 and field 2 for the algorithm configuration).

3. The co-simulation and optimization component links, which are the indicators and parameters keyword to be used (fields 4 and 5 of the Figure 5.10).

4. The possibility of including the sub-model variation defined in Figure 2.12 (in field 6 from Figure 5.10).

5. We also include the possibility of creating a queue of co-simulation-based optimization processes (this batching can be configured using the fields 7 and 8 from the figure).

6. Finally, the GUI can generate the JSON file containing the batch configuration to be executed as defined in example 12 (fields 9 and 10 from the figure).

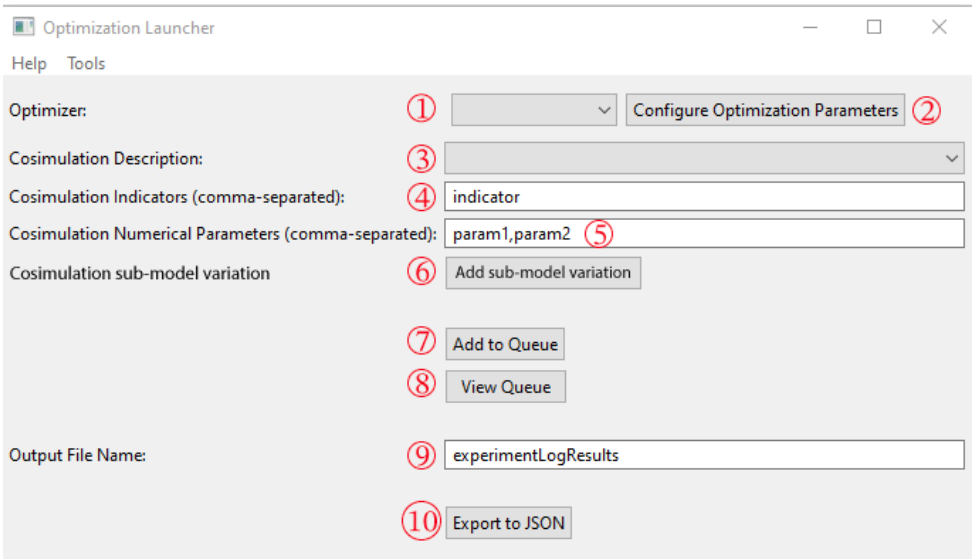


Figure 5.10: Overview of the GUI of the framework aiming at simplifying the input of the co-simulation-based optimization iteration description.

The screenshots shown in Figure 5.11 and Figure 5.12 illustrate the result of applying the modular approach. This allows us to see that the interface component can reach all of the use cases that were developed in the framework, which allows us to create a co-simulation-based optimization iteration using any combination of use cases.

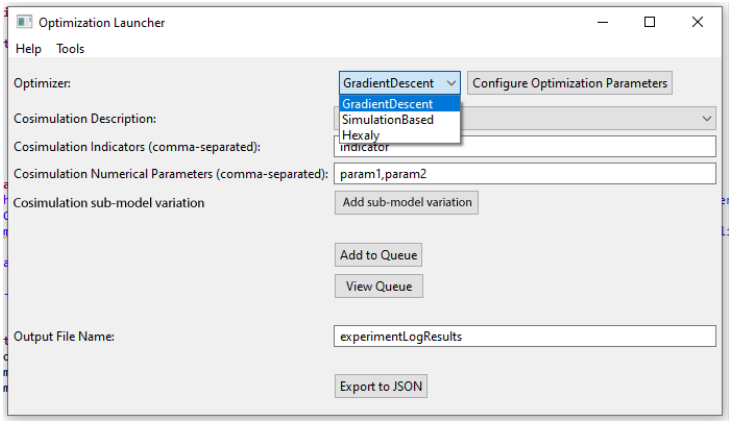


Figure 5.11: GUI implementation of the optimizer options implemented in the optimization component.

5.4 Framework Assessment

For the assessment of the framework, we recall Figure 4.1, where we propose the assessment flow to be conducted starting from the use case scale towards the abstract interface. To follow this order, we start in the **third level/application level** of development, where we can assess the results of the optimization processes.

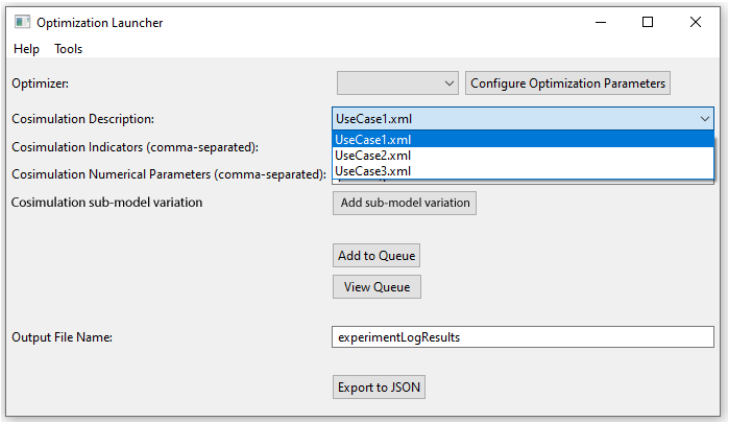


Figure 5.12: GUI implementation of the co-simulation options implemented in the co-simulation component.

5.4.1 Framework assessment: Application level

At this level, we intend to study and improve (in terms of a particular objective function) a complex system, and for this reason, we are interested in indicators such as the feasibility, which depends on software resources usage (was the improvement process finished?), and the optimal configuration performance (was the complex system improved?). The interest is mainly in the scenario description used to obtain the best result from the objective function.

With these elements, we are capable of drawing conclusions about the behavior of the complex system and the way that we can improve the performance of the systems in terms of the objective function. Additionally, in case of unfeasibility or poor optimization results, we can conclude on the conditions that make the problem not fit to be solved under the use case configuration.

To start the validation of this research, we now present the results of the study and improvement of the complex systems included in the use cases. For that, we perform an analysis of the optimization results focused on making explicit the improvement opportunities for each complex system.

Framework application to the co-simulation use case 1

The first use case defined in subsection 5.1.1 is optimized using the 3 different algorithms described in section 5.2. We generate one batch with the 3 cosim-opt instances that can be seen in Appendix C.

In Figure 5.13, we present the results of the optimization process using the simulation-based optimization algorithm, where we use the scenarios already established in Table 5.2. The 12 cases show a non-linear behavior that gives an optimal value of $f_1 = 1847$, which corresponds to the scenarios from 1 to 7. This gives us a first insight into the model performance, which appears to have an optimal zone when $t_{min} \leq 19$ and $t_{max} \geq 25$.

From the Figure 5.13, we can appreciate the considerable size of the search space that 2 parameters bring to the problem; it was necessary to do a projection of 3 axes to include both parameters and the objective function, where it is visible that the exploration performed was only a diagonal. With a bi-dimensional search space, the simulation-based algorithm is not robust enough to explore the function's behavior. The algorithm is useful to provide an initial look at the optimal values.

The next algorithm applied to this use case is the Gradient descent, which in this case is a multi-parametric optimization (T_{min}, T_{max}), using the algorithm parameters described in Table 5.5, and for this case the partial derivatives are used (equation 5.45). The resulting exploration performed is shown in Figure 5.14, where we observe an interesting difference between the exploration of the parameters. The T_{min} parameter is highly explored by the gradient, which even found a convex curve when $T_{max} \approx 24.6$. However, the exploration of the T_{max} parameter was just explored in 3 points different from $T_{max} \approx 24.6$,

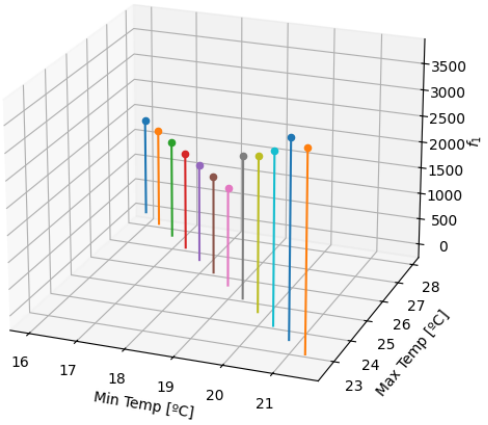


Figure 5.13: Results of the co-simulation-based optimization process using co-simulation use case 1 and the Simulation-based algorithm.

which suggests that the partial derivative of the parameter T_{max} found a minimum in a few iterations, but the algorithm continued solving the partial derivative of the parameter T_{min}.

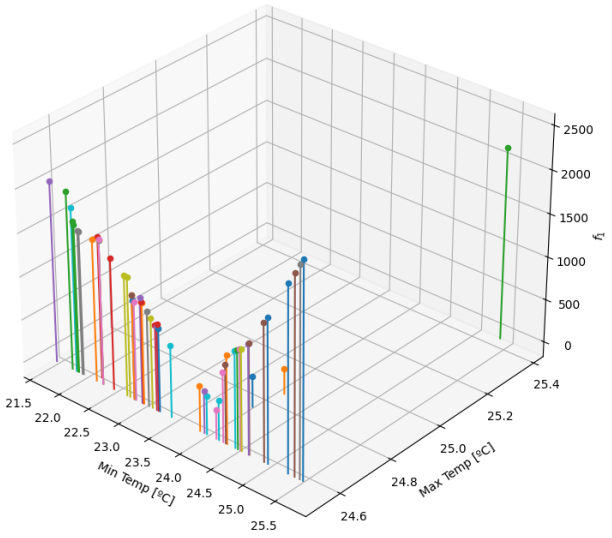


Figure 5.14: Results of the co-simulation-based optimization process using co-simulation use case 1 and the Gradient descent algorithm.

The Figure 5.14 shows that the optimal minimization result found is $f_1 = 299$ with the parameters $T_{min} = 24.3$ and $T_{max} = 24.8$. We see a big improvement in the minimization compared to the simulation-based algorithm; the gradient was effective in exploring blindly the 2-dimensional search space.

Next, we apply the Hexaly algorithm to this use case using the algorithm parameters described in Table 5.6. The resulting exploration performed is shown in Figure 5.15, where we observe an exploration pattern much more spread. Given the boundaries defined as $T_{min} \in [16.0, 27.9]$ and $T_{max} \in [16.1, 28.0]$, we can observe that Hexaly tries to explore solutions along the boundaries and then follow some descending gradients to find the optimal solution.

The solution found in Figure 5.15 is $f_1 = 292$ (objective function defined in equation 5.3) with the parameters $T_{min} = 24.3$ and $T_{max} = 24.88$. This is an interesting result because it found a solution very close to the one obtained from the Gradient descent algorithm; however, it is slightly better. The spread

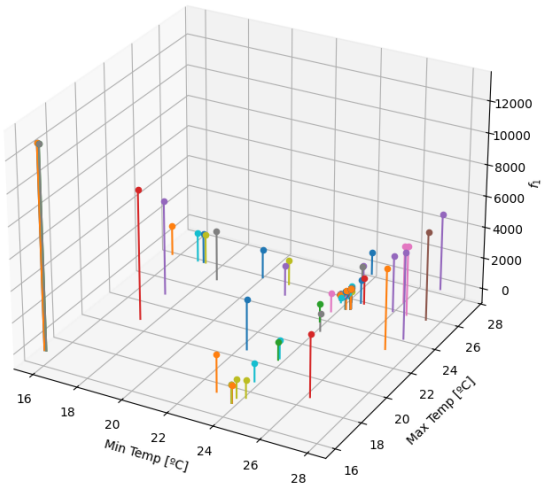


Figure 5.15: Results of the co-simulation-based optimization process using co-simulation use case 1 and the Hexaly algorithm.

exploration of the search space gives more reliability to the value found and gives more insights into the objective function behavior, that is, we can observe that there is a region with near-optimal values when $T_{min} \approx 24$.

The conclusion of the house thermal control problem is that to optimize the comfort and consumption of the house under the conditions defined for the optimization, the recommended temperature limits for the A/C controller are $[24.3, 24.88]$. To find this value, we find equally useful the gradient descent and Hexaly algorithms. The simulation-based algorithm was not very useful in this use case, due to the unknown and bi-dimensional nature of the search space.

Framework application to the co-simulation use case 2

The second use case described in subsection 5.1.2 is optimized using the three algorithms. We present the results from each algorithm, starting from the simulation-based optimization algorithm described in subsection 5.2.1. For the simulation-based algorithm, 20 scenarios were defined in Table 5.3, which produced the results shown in Figure 5.16, where the Y-axis shows the value of the objective function $f_2(SC)$ (defined in equation 5.5).

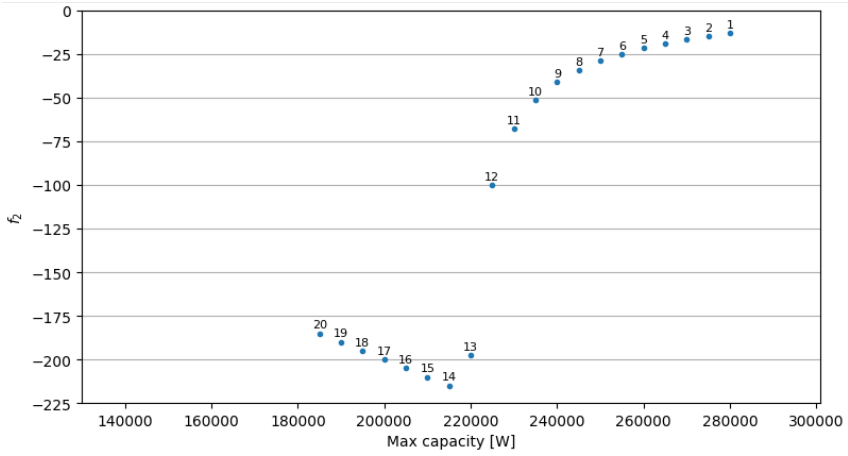


Figure 5.16: Results of the co-simulation-based optimization process using co-simulation use case 2 and the simulation-based algorithm.

The optimal (minimal) value from the objective function is $f_2(SC) = -215$ using the parameter value $MaxCapacity = SC = 215000$, which corresponds to scenario 14. The results show that the performance of the storage system according to the objective function starts too high; however, the function has a pivot point around scenarios 13 and 14, where the function starts to increase again.

The next optimization process for the second co-simulation use case uses the same objective function, but the second algorithm, which is the gradient descent algorithm described in subsection 5.2.2. We use the algorithm parameters shown in Table 5.5 and in the JSON file in Appendix D. The results of the optimization process are shown in Figure 5.17, which used the same layout configuration as Figure 5.16.

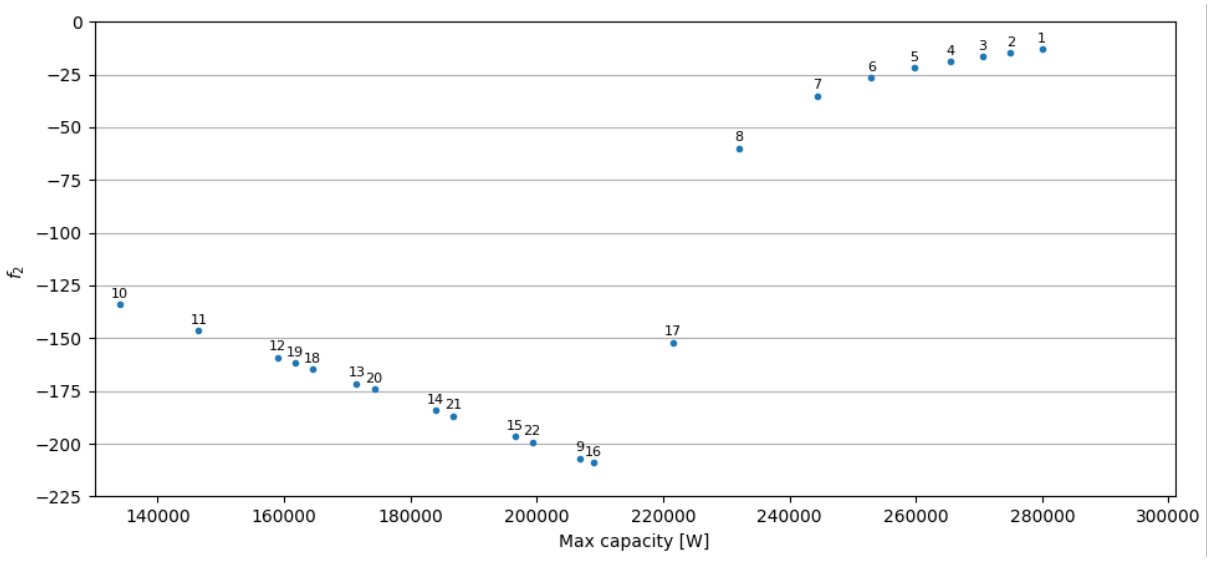


Figure 5.17: Results of the co-simulation-based optimization process using co-simulation use case 2 and the gradient descent algorithm.

The optimal (minimal) value from the objective function shown in Figure 5.17 is $f_2(SC) = -209$ using the parameter value $MaxCapacity = SC = 209008$, which corresponds to scenario 16. The results show similar exploration points to the ones set for the simulation-based algorithm, but we see the behavior of the algorithm of following the gradient direction, that is, we see the starting value very high, then the step taken grows as it approaches the minimum; however, it surpasses the minimum, which changes the direction of the gradient, causing the algorithm to go back until it reaches a point where the stopping criterion is reached. The use of the gradient descent for this use case is very useful for the exploration of the behavior of the objective function when we completely ignore the shape of the objective function; it is, however, possible to fall into a local minimum value.

The final optimization process for the second co-simulation use case uses the same objective function, but the third algorithm, which is the Hexaly implementation described in subsection 5.2.3. We use the algorithm parameters shown in Table 5.6 and in the JSON file in Appendix D. The results of the optimization process are shown in Figure 5.18 using the same layout configuration as Figure 5.16, except for the notation of the scenario number over each point in the graph; the reason for this is the fact that the Hexaly solver exploration system is a black-box (the company solver is a product sold in the marker, which means the exploration mechanism is a secret), we observed that the algorithm sometimes revisits the same point, which made the graph not readable and therefore we decided to not include the scenario number notation for this algorithm.

The optimal (minimal) value from the objective function shown in Figure 5.18 is $f_2(SC) = -218$ using the parameter value $MaxCapacity = SC = 219529.7$, which was used during scenarios 13, 18, 25, 30, 32, and 33. The results show that the Hexaly exploration system pays attention to the boundaries given as parameters (Table 5.6) and it explores the interval in an efficient way, shown by the optimal

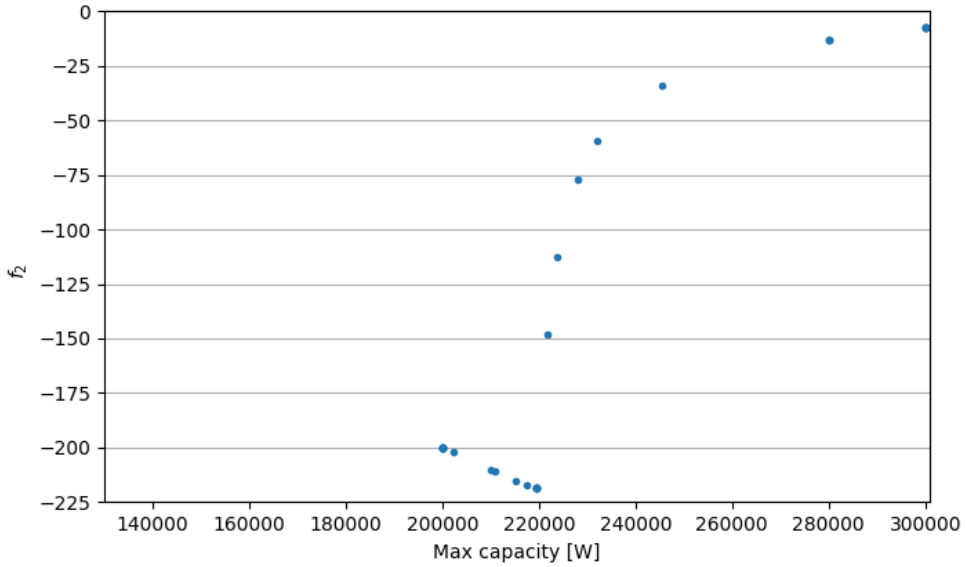


Figure 5.18: Results of the co-simulation-based optimization process using co-simulation use case 2 and the Hexaly algorithm.

value found (the best from the 3 algorithms).

The conclusion of the study and improvement of the autonomous micro-grid complex system is that, for the parameters defined in the co-simulation, the optimal size of the storage system should be close to $SC = 219529.7$, depending on the actual values available in the market. To reach this conclusion, we observed that the Hexaly and Gradient Descent are powerful self-sufficient exploration algorithms; however, Hexaly was capable of finding a more precise answer for the optimization problem.

Framework application to the co-simulation use case 3

We perform the optimization of use case 3 (described in subsection 5.1.3) using the algorithms to explore the performance behavior of the model. We start with the simulation-based algorithm, which has the scenarios defined in Table 5.4. We evaluate the results using the objective function f_3 defined in equation 5.42. The result of the scenarios is shown in Figure 5.19, where we see on the horizontal axis the value of the parameter N_p, as the other parameters are fixed, the values for each scenario can be found in Table 5.4.

The simulation-based optimal value obtained in Figure 5.19 is the scenario 2 where the parameters are $N_p = 2$, $Afc = 200$, and $Nfc = 140$; producing a value of $f_3(Afc, Nfc, N : p) = -305.5$. The behavior of the objective function does not seem clear, as the values go up and down on different scales. The optimal value found is the result of the scenarios proposed by the Hy2Car Professor Stephane Rael.

The next optimization process is performed using the gradient descent algorithm using the configuration defined in Table 5.5. The results are shown in Figure 5.20, where we see the result of the 9 scenarios explored. To illustrate the exploration, we generated a 3D figure where each axis is one of the parameters. Using color-coded points, we represent the value of the objective function f_3, the scale on the right contains the conversion of the colors for the f_3 values.

The optimal value obtained in Figure 5.20 is the scenario 3 where the parameters are $N_p = 9$, $Afc = 104$, and $Nfc = 142$; producing a value of $f_3(Afc, Nfc, N_p) = -69.8$. According to Professor Rael, while this configuration is possible, the solution is not better than the previous algorithm, and the space that 9 parallel supercapacitor branches take would affect the comfort of the vehicle owner, as it would be necessary to use space from the vehicle trunk.

We observe that the exploration of the search space was wide; notably, we notice that the gradient led the Afc value to $Afc = 1$, which is not a realistic value in the context of its representation (a cell with an active area of 1 cm^2 is too small for the source of the vehicle). Then the gradient performed the

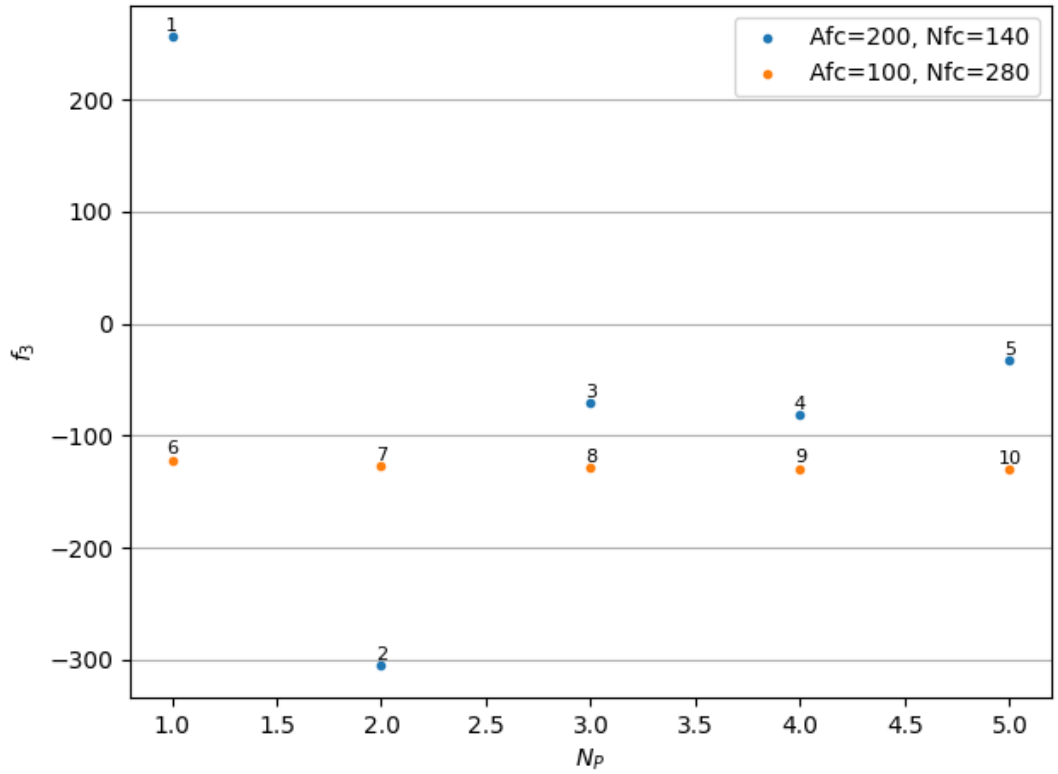


Figure 5.19: Results of the co-simulation-based optimization process using co-simulation use case 3 and the simulation-based algorithm.

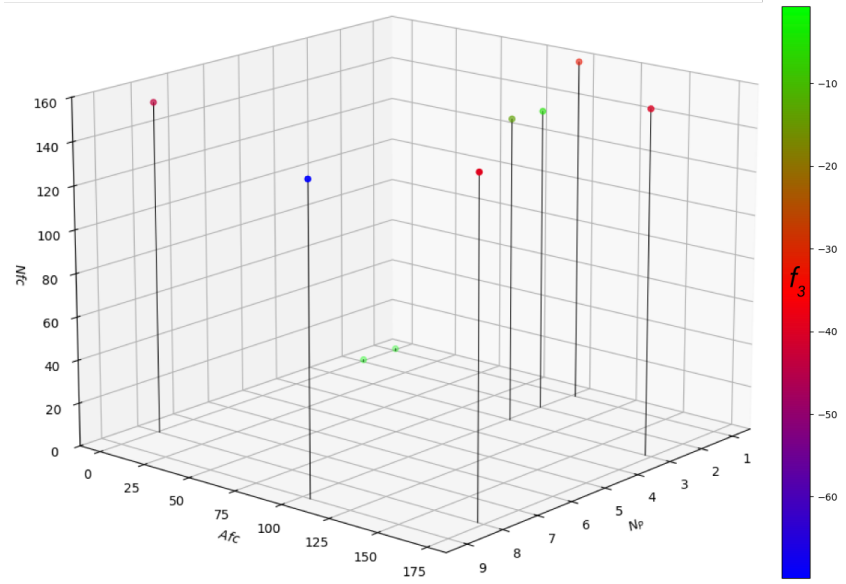


Figure 5.20: Results of the co-simulation-based optimization process using co-simulation use case 3 and the gradient descent algorithm. The 3 axes of the graph represent each of the parameters, and the color of the points represents the objective function value (f_3) using the scale on the right.

same operation on the Nfc value, resulting in the two green points in the middle of the figure, which triggered the stopping criterion of the algorithm.

The interesting conclusion from this algorithm's exploration is that the gradient descent produced scenarios that are not physically feasible. This means that the gradient is challenged when working in highly restricted problems, which is understandable, as it follows the mathematical gradient exclusively to decide on the scenarios, not the physical feasibility.

The last optimization process is performed using the Hexaly algorithm using the configuration defined in Table 5.6. The results are shown in Figure 5.21, where we use the same illustration technique as the gradient descent results.

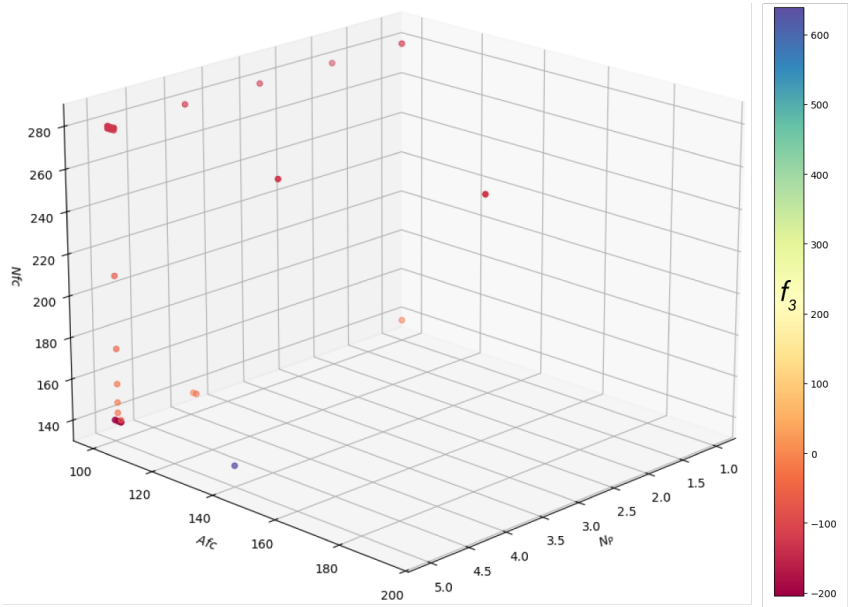


Figure 5.21: Results of the co-simulation-based optimization process using co-simulation use case 3 and the Hexaly algorithm. The 3 axes of the graph represent each of the parameters, and the color of the points represents the objective function value (f_3) using the scale on the right.

In Figure 5.21, the optimal value for the objective function is $f_3(Afc, Nfc, N:p) = -203.9$, obtained with the parameters $N_p = 5$, $Afc = 101$, and $Nfc = 140$. We can also observe that the majority of the exploration is found when $Afc \approx 100$, which is the lower bound of this variable, suggesting that the best scenarios are found when the active surface of the cell tends to be smaller. It is also interesting that the value found by Hexaly is still not superior to that of the simulation-based algorithm, which highlights the complexity of the 3-dimensional search space of the objective function.

The conclusion of the study and improvement of the Hy2Car vehicle is that, with the current model, we can suggest the configuration of $Afc = 200$, $Nfc = 140$, and $N_p = 2$ produces the optimal performance of the vehicle, that is, minimizing the ratio of energy used per kilometer and maximizing the energy left in the storage at the end of the trip (logic behind the function 5.42). However, we mentioned that some of the values used in the study are not validated by the Hy2Car project, which means that the main result of this study is the multi-model development and implementation in the framework, which can be further tuned with validated parameters to find the real optimal values (exploiting the algorithms available in the framework) to be used in the design of the vehicle.

Conclusions of the application level

The combination of the 3 co-simulation use cases and the 3 algorithm use cases generated 9 different experiments, which bring insights into the behavior of the different complex systems in terms of the objective functions defined. The application of the framework led to many insights found in the 9

optimization processes.

Study case	Optimization method	Results
1	Simulation-based	Figure 5.13
1	Gradient descent	Figure 5.14
1	Hexaly	Figure 5.15
2	Simulation-based	Figure 5.16
2	Gradient descent	Figure 5.17
2	Hexaly	Figure 5.18
3	Simulation-based	Figure 5.19
3	Gradient descent	Figure 5.20
3	Hexaly	Figure 5.21

Table 5.7: Study cases results.

We present Table 5.7 as a compilation of the results obtained in the application level of the framework. The 9 figures show different behaviors and search strategies, which showcase the importance of including a variety of algorithms (recalling the No free lunch theorem described in Definition 25). Also, the necessity of variety in co-simulations to challenge the generality of the framework's usability.

The conclusion of the application level of the framework is that when working with complex systems, the behavior of an objective function can be highly unpredictable; therefore, the exploration of solutions with several strategies is a useful approach to improving complex systems. This multi-strategic exploration is possible thanks to the framework's modular approach to integrating the co-simulation and optimization tools.

5.4.2 Framework assessment: Implementation level

The assessment of the *second level* (i.e., implementation level described in subsection 4.1.3) is performed by comparing the performance of the different use cases in terms of:

- Best algorithm per use case.
- Run-time per experiment.
- Number of iterations of the co-simulation per experiment.

In Table 5.8, we compile all the experiments' performance to be compared. The first column assigns an experiment number to each of the optimization processes performed for reference purposes. The second and third columns show the co-simulation and optimization use case per experiment.

Then, the fourth column shows the experiment run-time, which is measured from the launching of the optimization process until the optimal value is returned. The experiments were run on a computer with an Intel Xeon W-2223 processor running at 3.6 GHz, using 64GB of RAM, running Windows version 11 Enterprise. We can see that the co-simulation use cases 2 and 3 are heavy in comparison with the first use case. This is evident in experiment 3, where 212 co-simulations of use case 1 were executed, with an overall execution time that is still faster than 33 executions of use cases 2 (experiment 6) or 3 (experiment 8).

The fifth column shows the number of scenarios executed by the algorithm, that is, how many co-simulation executions were performed for the algorithm to decide on the optimal value. We can see the impact of including multiple parameters in the gradient descent, which resulted in many extra co-simulation executions to solve the derivatives and generate the next scenario (visible in experiments 2 and 8). We also see that for the co-simulation use cases 1 and 2, more scenarios tested by the algorithm resulted in better optimal values, which means that the exploration depth is rewarded with better results. However, for the third co-simulation use case, this is not consistent with the simulation-based algorithm, whose success is credited to the domain-specific knowledge of the person proposing the scenarios to test. Apart from that algorithm, we see that the extra exploration effort of the Hexaly algorithm was rewarded with a better result closer to the one obtained with the simulation-based algorithm.

Experiment	Cosim Use case	Algorithm	Run-time [sec]	Scenarios	Optimal value
1	1	Simulation-based	6.7	12	1847
2	1	Gradient descent	72.7	52 (154)*	299
3	1	Hexaly	121.6	212	292
4	2	Simulation-based	589.8	20	-215
5	2	Gradient descent	672.8	22	-209
6	2	Hexaly	973.1	33	-218
7	3	Simulation-based	375.6	10	-305.5
8	3	Gradient descent	1211.7	9 (33)*	-69.8
9	3	Hexaly	1340.2	37	-203.9

Table 5.8: Comparison of the experiments' performance in terms of run-time, number of scenarios, and optimal value found. ***:** In the case of the gradient descent algorithm, when multiple parameters are used, there is a necessity for partial derivatives, which implies additional co-simulations. In the parentheses, we include the total number of co-simulation executions, including the partial derivative calculations.

The last column contains the optimal value obtained in each experiment; this value is useful to visualize the expected trade-off between algorithm depth and the optimal value found. This is the main challenge when choosing the algorithm to use in an optimization problem. In general, we can conclude on some elements that can help decide on which algorithm to choose for a particular problem:

- Simulation-based:
  - Recommended for an initial exploration of the search space.
  - Useful with highly constrained problems.
  - Recommended in problems with the availability of a domain expert who can suggest interesting scenarios to explore.
- Gradient descent:
  - Benefit of being widely known and accepted in the optimization community.
  - Not recommended with highly constrained problems (it requires a gradient descent variant that considers constrained optimization problems [209]).
  - Problems with many parameters, which will increase the scenario calculation due to the partial derivatives calculation.
  - The objective function has to be feasible at all points in a defined interval. This means that the algorithm can have trouble with integer-defined parameters.
- Hexaly:
  - Private solver that requires a license to access its services.
  - Good performance in the experiments performed in this thesis.
  - Simple integration to the framework due to the API standard (compatible with black-box approach).

Another conclusion for the implementation level comes from the run-time analysis of the experiments, which is that if the co-simulation execution run-time is relatively low, it is worth using several optimization strategies to explore the search space. However, with heavy co-simulation, it is recommended to exploit mathematical optimization heuristics, such as gradient descent or commercial solvers, such as Hexaly. It is worth noting that in the context of complex systems, it is highly expected for the co-simulations to be heavy, as the co-simulation approach encourages complex multi-model implementations.

The main result obtained at the implementation level was the simplification of the process of creating and launching the different co-simulation-based optimization experiments. This is visible when

considering that all the results and analyses obtained so far were generated with only 3 standardized files (included in Appendices C, D, and E). The objective of the framework implementation was to experience the simplification of the process; this was achieved by using the GUI shown in Figure 5.11 to generate the JSON files that launch the whole process.

5.4.3 Framework assessment: Architectural level

The final assessment level is the level where we can analyse the performance of the abstract interface and design patterns included in the Conceptual General Framework proposed. For this, we retake the 5 main insights found in the literature review performed in chapter 3, and we present each element's influence in the results obtained.

Modularity

We implemented the modularity in the framework by following a black-box approach when handling the main elements of the process (co-simulation, optimization, and interface). This approach is an appropriate method for defining modular architectures that include entities with unknown internal behavior, known inputs, outputs, and parameters. The correct implementation of a black-box architecture has the potential to handle diverse co-simulation domains combined with diverse optimization methods.

The results of this modularity are seen in the creation of the experiments 5.12, and in the appendices C, D, E. The creation of the 9 experiments described in Table 5.8 was possible by only modifying between 6 and 8 fields; an example of this is the difference between the configuration of experiments 1, 4, and 7 from Table 5.8. We gather those configurations in Figure 5.22 for comparison. From these experiment configurations, we see highlighted in yellow the values that differ between configurations. The figure shows that it is possible to change between co-simulation use cases with the same algorithm by only changing 4 elements (the parameters, iterations, steps, and the co-simulation file name).

In general, we summarize the items that need to be changed to alternate between co-simulation or algorithms, as follows (we include the examples from the experiments performed with the use cases):

- A path to point at the file containing the co-simulation (e.g., CompleteHouseCosim.xml, completeAutonomousHouseNetwork_Opt.xml, hy2carOptim.xml).
- Keywords that reference values from the co-simulation file:
  - The indicator value keyword (e.g., objective)
  - The co-simulation parameter keywords (e.g., minTemp, maxTemp, maxCapacity, Afc, Nfc, Np_sc)
- A keyword to refer to the algorithm to use (e.g, simulationbased, gradientdescent, Hexaly).
- Keywords and values to set the algorithm parameters (e.g, iterations:12, steps:[0.5, -0.5], learning_rate:0.002, stop_criteria:0.01, etc).

The generalization of the components allowed us to change from one optimization method or co-simulation to another with only a few parameters to change. The advantage is that the optimization algorithms and co-simulations use the same source code, which means that the modular approach allows the algorithms and co-simulations to be easily interchangeable.

Domain flexibility

The results obtained in the application level (subsection 5.4.1) provided a deeper understanding of the complex system studied (i.e., house internal temperature, smart grid system, and hybrid source vehicle). Nevertheless, the success of the domain flexibility approach of the framework is only validated by the reuse of the architecture proposed in future applications across diverse domains. The validation of this framework is an effort that requires the involvement of more research teams that implement the framework and can further challenge the advantages of the proposal in this thesis. The extent of the

```
{ "optimization_instances" : [ {
    "opt_name" : "SimulationBased",
    "opt_parameters" : {
      "indicators" : [ "Objective" ],
      "cosimulation_parameters" : [ "minTemp", "maxTemp" ],
      "iterations" : 12,
      "steps" : [ 0.5, -0.5 ]
    },
    "experiment_path" : "Experiments/CompleteHouseCosim.xml"
}]}
```

```
{ "optimization_instances" : [ {
    "opt_name" : "Simulationbased",
    "opt_parameters" : {
      "indicators" : [ "Objective" ],
      "cosimulation_parameters" : [ "maxCapacity" ],
      "iterations" : 20,
      "steps" : [ -5000.0 ]
    },
    "experiment_path" : "Experiments/completeAutonomousHouseNetwork_Opt.xml"
}]}
```

```
{ "optimization_instances" : [ {
    "opt_name" : "SimulationBased",
    "opt_parameters" : {
      "indicators" : [ "Objective" ],
      "cosimulation_parameters" : [ "Afc", "Nfc", "Np_sc" ],
      "iterations" : 5,
      "steps" : [ 0.0, 0.0, 1.0 ]
    },
    "experiment_path" : "Experiments/hy2carOptim.xml"
}]}
```

Figure 5.22: Comparison of the configuration files that describe experiments 1, 4, and 7 from Table 5.8. All the text is the same for the files, with the exception of the text highlighted in yellow.

flexibility validated in this thesis is shown with the 3 co-simulation use cases developed (i.e., thermal control, electric grid design, vehicle design), in addition to the articles from Table 3.7, which are the frameworks where we found the importance of the flexibility domain approach (i.e., mobility, energy, electric, buildings, aerodynamics, cruise control, etc).

Extensibility

The ability to easily accommodate new features [7] of the framework is a key element in handling the heterogeneity of the components in co-simulation-based optimization. To demonstrate this feature in the framework, we used the MECSYCO software and included optimization algorithms in different ways. We observed the extensibility of MECSYCO with the compatibility of the software to integrate models imported from external software (e.g., OpenModelica with the FMU exporting standard, API service using the CSV format, Java-based models, among others). The extensibility of the optimization component was observed with the use cases, where we developed hard-coded algorithms (simulation-based and gradient descent), and we also used a third-party optimization solver implemented into the framework (Hexaly).

The idea was to leverage different co-simulation and optimization tools to maximize the reach of the framework. As seen in Table 3.1, diverse domains are combining co-simulations with optimisation to improve complex systems. This is why any contribution to the co-simulation-based optimization method is for the benefit of many domain researchers.

Pre-processing

We defined the pre-processing as the information required beforehand to start the co-simulation and optimization processes, but in the framework, we intend to combine both methods into one method called co-simulation-based optimization. For this, we define the general structure of the pre-processing information, which allowed us to propose a standardization of the information in a simple format (JSON chosen for the implementation, but others can be further tested), shown in Example 12. This standardization allowed us to launch 9 experiments with only 3 JSON files, where only some fields needed to be changed; we also exploited this generalization to queue the experiments in a batch structure, and finally, we developed a GUI to simplify the co-simulation-based optimization batch creation. Something as simple as enunciating the pre-processing information resulted in a standardization of a process that led to a user interface that makes the use of the framework simple.

Interface component

This research work started with the question illustrated in Figure 2.13, that is, what is the composition and dynamics of that intermediate step between co-simulation and optimization. This led us to a literature review that collected some architectural patterns to be taken into account when defining the intermediate component. Finally, we proposed a general structure of the framework where the majority of the efforts to include modularity, extensibility, flexibility, and pre-processing are condensed into the *interface component*.

The correct implementation of the interface component allows the combination of any co-simulation with any black-box optimization algorithm. The interface's main capability is the ability to run a co-simulation-based optimization process and then exchange the co-simulation and/or optimization components with little effort.

Chapter 6

Conclusions and Future Work

The context of this thesis is the growing presence of product complexity in the industry, which is expanding the need to understand and improve the behavior of systems while considering the internal complexity. A complex big-picture problem can be studied by combining the deep knowledge of various discipline experts through interdisciplinary collaboration. This calls for tools such as co-simulation and optimization that, combined, can serve as a means to study and improve complex systems.

The *research question* that motivated the research is how we can combine in a general manner co-simulation and optimization?. For this, we review the literature for a state-of-the-art on the co-simulation-based optimization method, which, to our knowledge, does not exist in the literature.

The *first contribution* of this thesis was the development of a state-of-the-art review of the co-simulation-based optimization method, where we provide an overview of approaches with a wide and heterogeneous landscape of optimization strategies and domains involved in the co-simulation, that is, we found more than 25 different algorithms and 14 disciplines being combined in different ways within each document.

The *second contribution* was a new classification of the co-simulation application, where we identified that the way a co-simulation is developed can follow an inductive reasoning modeling process (Bottom-up approach), starting from small models to build a big complex model, or a deductive reasoning modeling process (Top-down approach), which starts from a complex system and builds the components as smaller models connected.

The *third and core contribution* of this thesis is a generic framework for co-simulation-based optimization. The literature review showed the presence of ad hoc implementation and domain-specific frameworks for the field; however, this is the first framework attempting to provide a global decision support for the implementation of co-simulation-based optimization with a domain-independent approach. For the construction of the framework, we integrate the design patterns found in the literature review. The framework includes a modular, domain-flexible, and extensible architecture, with an interface component and pre-processing. The framework implements the black-box and coupled DEVS models formalisms to include the parametric and structural (sub-model variation potential) optimization process. The structure is divided into the co-simulation, optimization, and interface components. The novelty of the framework proposed is the generalization approach of the architecture. To assess the extent of the generalization success, we performed a software framework implementation along with 3 co-simulation use cases, combined with 3 optimization algorithms. The implementation allowed us to confront the advantages of the co-simulation component (the tool for studying the use cases), the optimization component (the tool for the improvement of the use cases), and interface components (where all the efforts to handle the interactions between the other components are condensed).

The *assessment* of the framework is performed in three levels. For the *application level*, we focus on

the use case results, which means we display the improvements achieved on each complex system after using the 3 algorithms. For the *implementation level*, we provided conclusions about the characteristics of each component's implementation by finding the optimal configuration for each use case under the conditions of each co-simulation; that is, for the house thermal control problem, we find the parameters that improve the comfort and reduce the energy consumption. For the autonomous micro-grid, we find the optimal storage system size that minimizes the leftover energy production while avoiding the under-dimensioning of the storage. And for the Hy2Car project, we find the optimal configuration of the hybrid source system that improves the efficiency of the energy consumption in terms of the energy used and generated. For the co-simulation component, we identified the effect of the co-simulation computational resource requirements on the optimization results. For the optimization component, we considered the No Free Lunch Theorem (Definition 25) as a motivation to include several optimization methods, which allowed us to identify the conditions that make each algorithm more suitable for a certain type of problem. The interface component produced a notable result of the simplification of the co-simulation-based optimization process, which is the possibility to change from one algorithm to another, with only 3-4 keywords needing to be modified, and to change from one co-simulation to another; also, only 3-4 fields need to be modified. For the *architectural level*, we identified the diversity of domains included (domain flexibility), the number of software tools working on the same shared environment (extensibility), and the capacity of the framework to exchange co-simulation or algorithms (manifestation of the modularity) by changing between 7-10 keywords (pre-processing) in a single file or graphic interface (interface component).

This thesis has contributed to the scientific community through two major publications. A conference paper was presented at the 35th European Modeling & Simulation Symposium (EMSS 2023), where the proposed framework was demonstrated using two representative use cases: residential thermal control and a micro-grid system [5]. Furthermore, a journal article providing a comprehensive state-of-the-art review on co-simulation-based optimization was submitted to *ACM Computing Surveys*. The article received an initial decision of *major revision* and is currently under its second round of peer review.

During the framework assessment, we mentioned some limitations of the framework structure and implementation. First, we mention the possibility of varying the structure of the multi-model used in the co-simulation component; in this description, we proposed the variation of the sub-models and the variation of the coupling structure of the multi-model. We included the conceptual strategy to use the sub-model variation in the optimization process; it is already possible to perform this kind of optimization by using the batching function of the framework to optimize a co-simulation under different structures. However, it was not included in the experiments developed for the thesis, which leaves this interesting feature without validation. We mentioned the idea of the variation of the coupling structure, but we did not attempt a conceptual or practical definition, as its influence on the black-box optimization methods is not clear to us.

The research problem treated in this thesis opened *perspectives* that require further research, that is, challenges in the short-term, mid-term, and long-term.

- Short-term:
  - The Hy2Car study can still be extended to include more details of the hybrid source behavior, especially in terms of the trade-off between the energy source configuration and its consequences in the vehicle performance on the road. This will produce more meaningful insights for the vehicle design project.
  - Consider a use case that exploits the sub-model variation feature of the framework. For example, in the Hy2Car project, we included a driver model; however, we can include different driving style models to assess the behavior of the vehicle under different driving scenarios.
- Mid-term: The application of the software framework implemented to study and improve a real problem from the industry would be interesting. This would put the framework in the position of combining several domain experts' experience into a study that can be later improved and confront the results with a team of experts.

- Long-term: The validation process of this thesis for the framework was focused on parametric optimization; however, it would be interesting to extend the framework validation process to the control optimization category. For this, the definition of use cases that follow the control structure is required (usually defined as Markov chain problems), in combination with control-based optimization algorithms (usually referred to as dynamic programming methods). This process will possibly challenge the implementation performed to ensure the intended generality of the framework.

Résumé en français

Introduction

La complexité croissante des produits dans l'industrie nécessite de comprendre et de prédire le comportement des systèmes, compte tenu de leur complexité interne. Un problème complexe peut être étudié en combinant les connaissances d'experts de différentes disciplines grâce à une collaboration multidisciplinaire. La complexité peut résider dans la structure d'un système, mais elle peut aussi résider dans l'œil de l'observateur de ce système. Même lorsqu'un système est « intrinsèquement » simple, c'est-à-dire qu'il peut être décrit en principe en termes simples, un observateur peut ne pas découvrir cette simple description et ne peut caractériser le système que de manière très complexe [15]. Pour gérer cette complexité avec une approche multidisciplinaire, un outil capable de capturer les mécanismes essentiels du système et de produire un comportement similaire est nécessaire. Nous nous intéressons à l'outil de simulation pour développer cette représentation dans un environnement virtuel, cependant, dans certains cas une simulation ne suffit pas à capter la complexité, c'est pourquoi nous avons recours à la co-simulation. La co-simulation consiste en l'interconnexion de plusieurs simulations qui génèrent un comportement global pouvant être combiné à une optimisation pour fournir des informations de haut niveau sur le système avant un déploiement réel.

Le sujet de recherche de cette thèse est la combinaison d'outils de co-simulation et d'optimisation comme méthode d'étude et d'amélioration de systèmes complexes. La co-simulation est définie comme la théorie et les techniques qui permettent la simulation générale d'un système à travers la composition de simulateurs [7]. Le résultat de la co-simulation est constitué de simulations couplées qui exécutent un multi-modèle présentant un comportement similaire à celui du système étudié. Grâce à cette co-simulation, il est possible de modifier ses paramètres et sa structure pour explorer d'éventuelles modifications du système; pour ce processus d'exploration, des algorithmes d'optimisation capables de trouver des configurations idéales peuvent être utilisés. Nous appelons cette méthode combinée «optimisation basée sur la co-simulation»; cependant, dans la littérature, nous ne trouvons que des applications ad hoc ou spécifiques à un domaine, ce qui signifie que le domaine manque d'une méthode d'intégration systématique et indépendante du domaine pour une optimisation basée sur la co-simulation.

La structure du manuscrit commence par une introduction à la problématique de recherche, suivie d'une revue de la littérature où la pertinence de la méthode d'optimisation basée sur la co-simulation est questionnée, les types de simulation utilisés et les types de stratégies d'optimisation utilisées sont également énoncés. En utilisant les résultats de la revue de la littérature, un *framework* général pour l'optimisation basée sur la co-simulation est proposé, dans le but de définir la structure abstraite générale à suivre pour tout domaine intéressé par l'utilisation de l'optimisation basée sur la co-simulation. Suite à cela, une évaluation du *framework* est réalisée, dans laquelle une implémentation est réalisée dans un *framework* logiciel, où nous effectuons la mise en œuvre de 3 études de cas de co-simulation, combinées à 3 algorithmes d'optimisation. Enfin, les conclusions générales de la thèse sont présentées.

Revue de la littérature

Nous effectuons une analyse préliminaire de la littérature concernant l'optimisation basée sur la co-simulation. Pour cela nous utilisons l'équation de recherche suivante dans les bases de données Scopus, Web of Science et DBLP

TITLE-ABS-KEY(("co-simulation" OR "cosimulation" OR "coupled simulation") AND "optimization").

Le résultat de cette recherche, présenté dans la figure 3.1, montre le calendrier de publication. Il illustre les premières études menées sur le sujet entre 2007 et 2009, suivies d'une augmentation du nombre de publications au fil du temps. Le problème a pris davantage d'importance au cours des dix dernières années, en particulier au cours des trois dernières, où le nombre est resté autour de 13. Cette tendance démontre la valeur des résultats de l'enquête pour la poursuite des recherches sur le sujet et la nécessité de combiner tous les efforts individuels, en particulier en ce qui concerne la méthodologie et les procédures opérationnelles standards.

La répartition des publications par domaine est une autre analyse qui peut fournir des informations générales sur la littérature. Nous présentons le tableau 1 provenant de l'outil d'analyse SCOPUS, qui indique le nombre de publications relevant de chacun des domaines généraux définis dans l'indexation SCOPUS (une publication peut relever de plusieurs domaines). Le tableau montre que les principaux domaines concernés sont l'ingénierie et l'informatique, et que les applications sont diverses, allant de la physique et des mathématiques aux sciences naturelles et sociales. En raison du large éventail d'applications dans tout le spectre de la recherche, la méthodologie d'optimisation basée sur la co-simulation a le potentiel de bénéficier à un large éventail de domaines. Compte tenu de la nature multidisciplinaire de la méthodologie examinée dans cette étude, un examen plus approfondi des domaines et des méthodes dans lesquels cette technique est actuellement appliquée est nécessaire.

Table 1: Nombre de publications par domaine de recherche, selon les catégories définies dans SCOPUS. Résultats obtenus à partir de SCOPUS après suppression manuelle des articles hors du champ d'application de cette enquête à partir de l'équation de recherche en février 2024, avec un total de 85 publications.

Domaine	Articles
Ingénierie	53
Informatique	34
Mathématiques	13
Physique et astronomie	12
Sciences de l'environnement	11
Science des matériaux	10
Énergie	10
Sciences agricoles et biologiques	3
Génie chimique	3
Sciences sociales	3
Sciences de la Terre et des planètes	2
Chimie	2
Sciences de la décision	1
Biochimie, génétique et biologie moléculaire	1

L'examen de la base de données des publications utilisant l'optimisation basée sur la co-simulation a permis d'identifier des modèles dans les méthodologies suivies. Les différents modèles sont affinés en classifications qui mettent en évidence les différences importantes entre les applications. La classification complète est présentée dans Table 3.5 et Table 3.6, où nous classons les 96 documents trouvés à l'aide de

l'équation

TITLE((“co-simulation” OR “cosimulation” OR “coupled simulation”) AND “optimization”) AND PUBYEAR $<$ 2024).

Chaque ligne contient un document et les colonnes sont définies comme suit :

- Approche de construction utilisée pour développer la co-simulation, les approches sont: i) l'approche ascendante pour la construction d'une co-simulation utilise la méthode du raisonnement inductif (l'approche inductive part du particulier pour aller vers le général, de sorte que des cas particuliers sont observés puis combinés dans un ensemble plus large ou une affirmation générale [51]). Lorsque nous appliquons cette méthode à l'étude d'un système complexe, nous arrivons à la définition suivante : *Approche ascendante pour la construction d'une co-simulation : construction d'une co-simulation par le couplage de deux ou plusieurs simulations existantes de systèmes spécifiques à un domaine qui interagissent dans un environnement partagé pour produire un comportement plus large et général.*

  Et ii) l'approche descendante utilise la méthode du raisonnement déductif (l'approche déductive est basée sur une théorie générale antérieure pour passer au spécifique, et donc, elle passe de la théorie générale au spécifique [51]). L'application de la méthode du raisonnement déductif à la co-simulation conduit à la définition suivante : *Approche descendante pour la construction d'une co-simulation : construction d'une co-simulation qui commence par l'étude du système complexe, en utilisant la modélisation et la simulation pour décrire des sous-systèmes plus petits et interconnectés.*

- Domaines de simulation présents dans la co-simulation, chaque co-simulation combine deux ou plusieurs modèles, ces modèles sont définis du point de vue d'un domaine scientifique. Dans cette colonne, nous mentionnons les domaines inclus dans la co-simulation.

- Le logiciel de simulation utilisé pour créer chacun des modèles de la co-simulation.

- La méthode d'optimisation.

- Le logiciel d'optimisation.

La principale contribution de cette revue, qui présente une analyse approfondie sur l'optimisation basée sur la co-simulation, est de fournir une analyse de la manière dont la co-simulation et l'optimisation sont mises en œuvre. Pour ce faire, nous passons en revue la littérature pour identifier les tendances émergentes dans les stratégies proposées et arrivons à une abstraction générale des méthodologies. Cette abstraction a montré que l'extensibilité, la normalisation de l'exportation/importation de modèles de simulation d'un logiciel à un autre et des cadres de conception (frameworks) sont les principales approches pour aborder les différentes disciplines impliquées dans l'optimisation basée sur la co-simulation, ainsi que les différentes techniques d'optimisation disponibles.

Au cours du développement de cette recherche approfondie, nous avons apporté plusieurs contributions pertinentes à l'optimisation basée sur la co-simulation, dont l'une est la classification de construction pour les co-simulations, qui contribue à une meilleure compréhension du processus de modélisation d'un système complexe, ce qui aide à identifier de nouveaux systèmes complexes pouvant bénéficier de la méthode de co-simulation.

Du point de vue de l'optimisation, nous avons identifié et classé les méthodes d'optimisation utilisées dans la base de données bibliographique (Figure 1). Cette classification nous a permis d'examiner les caractéristiques de l'interaction entre les composants de co-simulation et d'optimisation afin de déterminer les trois types de stratégies d'interopérabilité utilisées dans la littérature pour intégrer la co-simulation et l'optimisation, à savoir manuellement, par collaboration logicielle et à l'aide de *frameworks* d'optimisation basés sur la co-simulation. Nous concluons que l'approche « boîte noire » est un concept pertinent pour étudier l'interopérabilité entre la co-simulation et l'optimisation. Cette approche permet de regrouper la co-simulation et l'optimisation dans un environnement conceptuel commun, qui reflète les environnements de mise en œuvre présents dans la littérature. Une autre conclusion de cette analyse de la littérature est

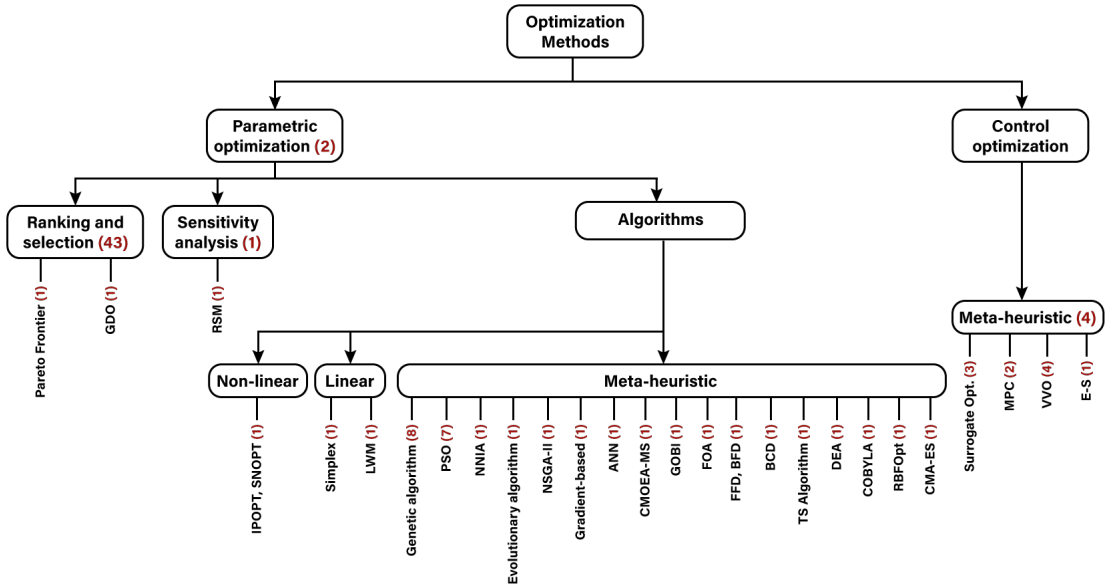


Figure 1: Catégories d'optimisation et méthodologies spécifiques utilisées dans la base de données échantillon (le nombre d'articles utilisant chaque méthode est indiqué en rouge entre parenthèses, voir tableau 3.5) et 3.6).

liée à l'intérêt de tirer parti des nombreuses applications en proposant des *frameworks* offrant des capacités d'extensibilité, de modularité, de prétraitement et d'interfaçage qui rendent possible l'intégration des outils de co-simulation et d'optimisation.

Nous nous appuyons sur les conclusions de cette revue de la littérature pour proposer un nouveau *framework* général réutilisable, afin d'accélérer et de simplifier la mise en œuvre de l'optimisation basée sur la co-simulation pour l'étude et l'amélioration de systèmes complexes. Nous utilisons des modèles architecturaux existants pour proposer un *framework* général d'optimisation basé sur la co-simulation, visant à fournir une structure flexible en termes d'algorithmes d'optimisation et de domaines de simulation.

Cela a conduit à la principale contribution de cette thèse : un *framework* modulaire, flexible et extensible pour l'optimisation basée sur la co-simulation, incluant un composant d'interface généralisé pour gérer l'interaction entre la co-simulation et l'optimisation, ainsi qu'une définition des informations de prétraitement nécessaires pour l'exécution simple de l'ensemble. Nous réalisons une implémentation sous la forme d'un *framework* logiciel déployé dans un environnement Java, suivant les lignes directrices du *framework* conceptuel, et étudions trois systèmes complexes hétérogènes (études de cas) en combinaison avec trois algorithmes d'optimisation afin de démontrer la fonctionnalité du *framework*.

Proposition de *framework* général

Dans ce chapitre, nous définissons le *framework* d'optimisation basé sur la co-simulation, dont la structure provient directement des modèles architecturaux trouvés dans la revue de la littérature. Ces éléments sont la flexibilité, l'extensibilité, la modularité, le prétraitement et le composant d'interface. Les éléments structurels, tels que la modularité (avec une séparation des composants dans des boîtes), le prétraitement (énumérant les besoins en informations) et le composant d'interface (composant intermédiaire pour gérer la communication entre la co-simulation et l'optimisation).

Afin de garantir l'extensibilité et la flexibilité du *framework* quelle que soit son implémentation, nous avons opté pour une architecture de type « boîte noire ». Pour illustrer son implémentation, nous fournissons un exemple qui sert de fil conducteur. Nous présentons une mise en œuvre du *framework* dans

un environnement basé sur Java, où les composants de co-simulation et d'optimisation sont encapsulés sous forme d'objets Java. L'effort de mise en œuvre principal se concentre sur le composant d'interface, où nous définissons des interfaces qui permettent d'utiliser les autres composants comme des plugins. Le résultat est un *framework* d'optimisation basé sur la co-simulation, où les interactions de l'utilisateur convergent vers le composant d'interface, principalement pour la mise en place d'un processus d'optimisation basé sur la co-simulation (CS-O) et pour la mise en place d'un lot CS-O (dans l'assistant de configuration des lots).

Le *framework* est représenté par une structure basée sur des composants. Nous utilisons une structure modulaire/boîte noire, conformément à la tendance des autres *frameworks* issus de la revue de la littérature. Cette approche est appropriée pour décrire de manière abstraite le processus d'optimisation basé sur la co-simulation tout en conservant les particularités de chaque domaine. Un autre détail pertinent inclus dans le *framework* est l'identification des informations de prétraitement nécessaires au lancement d'un processus d'optimisation. Nous identifions les composants de co-simulation, d'optimisation et d'interface comme les principales entités interagissant dans le *framework*. En outre, la partie prétraitement du *framework* compile toutes les informations nécessaires au lancement du processus d'optimisation.

La Table 2 récapitule les processus de traitement de l'information qui se déroulent dans le *framework*; la première colonne contient le nom du composant où le processus se déroule, les deuxième et troisième colonnes indiquent respectivement les entrées et les sorties du processus, ainsi que la notation mathématique définie, et la quatrième colonne contient la notation mathématique du processus. Notons que le composant d'interface contient les deux processus qui transforment les informations de sortie en informations d'entrée. Pour ce processus en boucle nous proposons une représentation directe du flux à l'aide de la notation mathématique des équations $(Cosim(N, I_{cs}) = \vec{O_{cs}})$, $(\vec{I_{opt}} = ObjFunc(\vec{O_{cs}}))$, $(Algorithm(\vec{I_{opt}}) = O_{opt})$ et $(Setup(O_{opt}) = (N, I_{cs}))$ dans Figure 2.

Table 2: Processus, représentation des informations sortantes et entrantes pour les composants du *framework*.

Composant	Entrées → Notation	Sorties → Notation	Processus interne
Co-simulation	Multi-modèle, Paramètres → N, I_{cs}	Résultats → $\vec{O_{cs}}$	$Cosim(\cdot)$
Optimisation	Indicateurs → $\vec{I_{opt}}$	Scénario → O_{opt}	$Algorithm(\cdot)$
Interface	Résultats → $\vec{O_{cs}}$	Indicateurs → $\vec{I_{opt}}$	$ObjFunc(\cdot)$
Interface	Scénario → O_{opt}	Multimodèle, paramètres → N, I_{cs}	$Setup(\cdot)$

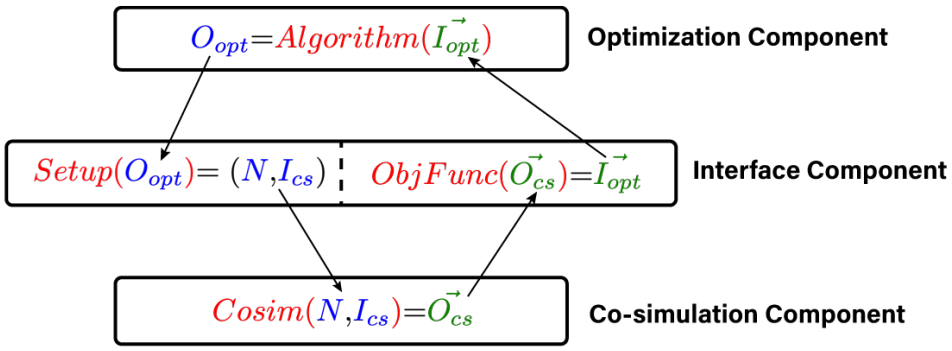


Figure 2: Notation mathématique des flux pour le *framework* d'optimisation basé sur la co-simulation. Au cours de l'optimisation, certains éléments doivent être définis dès le début et rester fixes (en rouge), d'autres nécessitent également une valeur initiale mais varient au cours du processus d'optimisation (en bleu), et d'autres encore ne nécessitent aucune initialisation (en vert).

Nous avons présenté l'architecture générale du *framework* avec ses composants et le flux d'informations dans Figure 2 et la notation utilisée pour les processus et les interactions est résumée dans Table 4.2.

Néanmoins, pour bien comprendre l'utilité du *framework*, nous avons besoin d'applications qui peuvent montrer l'utilisation des différentes fonctionnalités de mise en œuvre du *framework*. Pour cela, nous présentons des cas d'utilisation destinés à évaluer les fonctionnalités du *framework* tout en étudiant et en améliorant certains systèmes complexes.

Évaluation du *framework*

Définition des cas d'utilisation et des algorithmes de cosimulation

Nous présentons l'optimisation pour 3 études de cas. L'étude de cas n° 1 est définie comme l'optimisation du système de climatisation afin d'améliorer le confort des occupants d'une maison. Dans l'étude de cas n° 2, nous déterminons la capacité optimale du système de stockage d'un réseau de 4 maisons autonomes afin de minimiser les pertes d'énergie. Dans l'étude de cas n° 3, nous développons l'optimisation basée sur la co-simulation pour la conception d'un véhicule hybride à hydrogène (dans le cadre du projet Hy2Car). Le résultat est la co-simulation du véhicule et les connaissances en matière d'optimisation contribuant à la proposition optimale pour la configuration de la source hybride.

Cas d'Utilisation 1 : Système de Contrôle de la Température d'une Maison

Le premier cas d'utilisation démontre l'étude d'un système complexe par co-simulation, centré sur le contrôle de la température interne d'une maison. En utilisant le logiciel MECSYCO, ce cas illustre comment représenter un système multi-modèle. Le système cible est le contrôle de l'air conditionné d'une maison, intégrant les modèles de températures interne et externe, le système de climatisation et l'occupation de la maison. L'optimisation vise à améliorer le problème complexe de régulation thermique.

Cas d'Utilisation 2 : Réseau de Maisons Autonomes

Le deuxième cas met en avant la capacité de MECSYCO à coupler et à réutiliser des modèles sous forme de "boîtes noires". Ici, un réseau de maisons autonomes, interconnectées à un système de stockage partagé, est exploré. La co-simulation analyse les interactions entre quatre maisons générant de l'électricité via des éoliennes et un module de stockage intelligent. Ce modèle, issu d'une thèse de doctorat, utilise l'imbrication pour réutiliser des modèles sans affecter les propriétés de co-simulation.

Cas d'Utilisation 3 : Projet Hy2Car

Dans le troisième cas, nous étudions et améliorons un système complexe lié au projet Hy2Car, qui explore l'utilisation directe d'une pile à hydrogène hybride et de supercondensateurs comme source d'énergie pour un véhicule. L'optimisation basée sur la co-simulation vise à tester et à valider le design du véhicule, en simulant divers domaines (électrique, physique, contrôle, géographique) pour optimiser le comportement énergétique.

Algorithmes d'Optimisation 1 : Optimisation basée sur la simulation

L'optimisation basée sur la simulation utilise un échantillon limité de scénarios pour optimiser une fonction objectif. Cette méthode, connue sous le terme de "ranking and selection", est utile pour les simulations complexes. Elle a été appliquée dans 44% des articles examinés dans la littérature.

Algorithmes d'Optimisation 1 : Algorithme de Descente de Gradient

Cet algorithme ajuste les paramètres en utilisant le gradient des résultats de la co-simulation. Il calcule itérativement le gradient pour approcher une valeur optimale. Bien que simple, la logique de descente de gradient est essentielle pour l'optimisation paramétrique.

Algorithmes d'Optimisation 1 : Intégration d'un Outil d'Optimisation Externe (Hexaly)

Ce cas d'utilisation est particulièrement intéressant pour l'extensibilité du *framework*, visant à lier un outil d'optimisation externe au *framework*. L'implémentation du solveur Hexaly, offrant des améliorations significatives, facilitera l'intégration avec un minimum d'effort par rapport aux algorithmes précédents.

Niveaux d'évaluation

Cette évaluation d'un *framework* logiciel comprend trois niveaux : le niveau d'application, le niveau d'implémentation et le niveau architectural. Chacun de ces niveaux offre une perspective unique sur les aspects de performance et d'optimisation des systèmes complexes.

Au niveau d'application, nous développons des use cases visant à améliorer des systèmes complexes grâce à l'implémentation logicielle. Les résultats ici fournissent des aperçus d'optimisation, limités aux systèmes étudiés, mettant en avant les solutions concrètes issues de la co-simulation.

Le niveau d'implémentation se concentre sur la comparaison des performances des différents use cases. Cette analyse permet d'évaluer l'efficacité de chaque implémentation, fournissant des arguments solides sur les avantages de la structure du *framework* pour l'optimisation basée sur la co-simulation.

Enfin, le niveau architectural aborde les bénéfices découlant de l'application d'interfaces abstraites et de motifs de conception structurels. Les conclusions de cette analyse soulignent l'importance des choix architecturaux, renforçant la maintenabilité et la flexibilité du *framework*. Ensemble, ces trois niveaux forment une évaluation complète des capacités du *framework* logiciel.

Résultats au niveau de l'application

Nous présentons Table 3 comme une compilation des résultats obtenus au niveau de l'application du *framework*. Les 9 figures montrent différents comportements et stratégies de recherche, qui soulignent l'importance d'inclure une variété d'algorithmes qui rappelle le théorème « No free lunch » (Dans le domaine de l'optimisation, le théorème « No Free Lunch » établit que pour tout algorithme d'optimisation, toute performance élevée sur un type de problème est compensée par la performance moins élevée sur un autre type [173].). Elles montrent également la nécessité de varier les co-simulations afin de remettre en question la généralité de l'utilisabilité du *framework*.

Étude de cas	Méthode d'optimisation	Résultats
1	Simulation	Figure 5.13
1	Descente de gradient	Figure 5.14
1	Hexaly	Figure 5.15
2	Simulation	Figure 5.16
2	Descente de gradient	Figure 5.17
2	Hexaly	Figure 5.18
3	Simulation	Figure 5.19
3	Descente de gradient	Figure 5.20
3	Hexaly	Figure 5.21

Table 3: Résultats des cas étudiés.

La conclusion du niveau d'application du *framework* est que, lorsqu'on travaille avec des systèmes complexes, le comportement d'une fonction objectif peut être très imprévisible ; par conséquent, l'exploration de solutions à l'aide de plusieurs stratégies est une approche utile pour améliorer les systèmes complexes. Cette exploration multi-stratégique est possible grâce à l'approche modulaire du *framework* pour intégrer les outils de co-simulation et d'optimisation.

Résultats au niveau de l'implémentation

L'évaluation du *deuxième niveau* (c'est-à-dire le niveau de l'implémentation) est effectuée en comparant les performances des différents cas d'utilisation en termes de :

- Meilleur algorithme par cas d'utilisation.
- Durée d'exécution par expérience.
- Nombre d'itérations de la co-simulation par expérience.

Dans Table 4, nous compilons toutes les performances des expériences. La première colonne attribue un numéro d'expérience à chacun des processus d'optimisation effectués à des fins de référence. Les deuxième et troisième colonnes indiquent le cas d'utilisation de la co-simulation et de l'optimisation.

Expérience	**Cas d'étude**	**Algorithme**	**Temps d'exécution [s]**	**Scénarios**	**Valeur optimale**
1	1	Simulation	6.7	12	1847
2	1	Descente de gradient	72.7	52 (154)*	299
3	1	Hexaly	121.6	212	292
4	2	Simulation	589.8	20	-215
5	2	Descente de gradient	672.8	22	-209
6	2	Hexaly	973.1	33	-218
7	3	Simulation	375.6	10	-305.5
8	3	Descente de gradient	1211.7	9 (33)*	-69.8
9	3	Hexaly	1340.2	37	-203.9

Table 4: Comparaison des performances des expériences en termes de temps d'exécution, de nombre de scénarios et de valeur optimale trouvée. ***:** Dans le cas de l'algorithme de descente de gradient, lorsque plusieurs paramètres sont utilisés, il est nécessaire de calculer les dérivées partielles, ce qui implique des co-simulations supplémentaires. Entre parenthèses, nous indiquons le nombre total d'exécutions de co-simulation, y compris les calculs de dérivées partielles.

Enfin, la quatrième colonne indique le temps d'exécution de l'expérience, mesuré entre le lancement du processus d'optimisation et l'obtention de la valeur optimale. Les expériences ont été réalisées sur un ordinateur équipé d'un processeur Intel Xeon W-2223 cadencé à 3,6 GHz, avec 64 Go de RAM, sous Windows version 11 Enterprise. Nous pouvons constater que les cas d'utilisation de co-simulation 2 et 3 sont lourds par rapport au premier cas d'utilisation. Cela est évident dans l'expérience 3, où 212 co-simulations du cas d'utilisation 1 ont été exécutées, avec un temps d'exécution global qui reste plus rapide que celui des 33 exécutions des cas d'utilisation 2 (expérience 6) ou 3 (expérience 8).

Résultats au niveau architectural

Le niveau d'évaluation final est celui où nous pouvons analyser les performances de l'interface abstraite et des modèles de conception inclus dans le *framework* général conceptuel proposé. Pour cela, nous reprenons les 5 principales conclusions tirées de l'analyse documentaire et nous présentons l'influence de chaque élément dans les résultats obtenus.

Nous avons mis en œuvre la modularité dans le *framework* en suivant une approche de type « boîte noire » lors du traitement des principaux éléments du processus (cosimulation, optimisation et interface). Cette approche est une méthode appropriée pour définir des architectures modulaires qui incluent des entités dont le comportement interne est inconnu, mais dont les entrées, les sorties et les paramètres sont connus. La mise en œuvre correcte d'une architecture de type « boîte noire » permet de traiter divers domaines de cosimulation combinés à diverses méthodes d'optimisation.

Les résultats obtenus au niveau de l'application ont permis de mieux comprendre le système complexe étudié (à savoir la température interne des maisons, le système de réseau intelligent et les véhicules à source hybride). Néanmoins, la flexibilité du *framework* n'est validée que par la réutilisation de l'architecture proposée dans de futures applications dans divers domaines. La validation de ce *framework* est un effort qui nécessite la participation d'autres équipes de recherche qui mettent en œuvre le *framework* et peuvent remettre en question les avantages de la proposition présentée dans cette thèse. L'étendue de la flexibilité validée dans cette thèse est illustrée par les trois cas d'utilisation de co-simulation développés (à savoir, le contrôle thermique, la conception du réseau électrique et la conception de véhicules).

La capacité à intégrer facilement de nouvelles fonctionnalités [7] dans le *framework* est un élément clé pour gérer l'hétérogénéité des composants dans l'optimisation basée sur la co-simulation. Pour démontrer cette fonctionnalité, nous avons utilisé le logiciel MECSYCO et inclus des algorithmes d'optimisation de différentes manières. Nous avons observé l'extensibilité de MECSYCO grâce à la compatibilité du logiciel pour intégrer des modèles importés à partir de logiciels externes (par exemple, OpenModelica avec la norme d'exportation FMU, le service API utilisant le format CSV, les modèles basés sur Java, entre autres). L'extensibilité du composant d'optimisation a été observée avec les cas d'utilisation, où nous avons développé et implanté des algorithmes (basés sur la simulation et la descente de gradient), et nous avons également utilisé un solveur d'optimisation tiers implémenté dans le *framework* (Hexaly).

Nous avons défini le prétraitement comme les informations requises au préalable pour démarrer les processus de co-simulation et d'optimisation, mais dans le cadre de cette thèse, nous avons l'intention de combiner de manière générale les deux méthodes en une seule appelée « optimisation basée sur la co-simulation ». Pour cela, nous définissons la structure générale des informations de prétraitement, ce qui nous a permis de proposer une normalisation des informations dans un format simple (JSON choisi pour la mise en œuvre, mais d'autres peuvent être testés ultérieurement), comme le montre l'exemple 12. Cette normalisation nous a permis de lancer 9 expériences avec seulement 3 fichiers JSON, dans lesquels seuls certains champs devaient être modifiés ; nous avons également exploité cette généralisation pour mettre les expériences en file d'attente dans une structure par lots, et enfin, nous avons développé une interface graphique pour simplifier la création de lots d'optimisation basée sur la co-simulation. Une chose aussi simple que l'énoncé des informations de prétraitement a abouti à la normalisation d'un processus qui a conduit à une interface utilisateur qui simplifie l'utilisation du *framework*.

Ce travail de recherche a débuté par la question illustrée dans Figure 2.13, à savoir quelle est la composition et la dynamique de cette étape intermédiaire entre la co-simulation et l'optimisation. Cela nous a conduits à une revue de la littérature qui a permis de recueillir certains modèles architecturaux à prendre en compte lors de la définition du composant intermédiaire. Enfin, nous avons proposé une structure générale du *framework* dans laquelle la plupart des efforts visant à inclure la modularité, l'extensibilité, la flexibilité et le prétraitement sont condensés dans le *composant d'interface*.

La mise en œuvre correcte du composant d'interface permet de combiner n'importe quelle co-simulation avec n'importe quel algorithme d'optimisation de boîte noire. La principale capacité de l'interface est de pouvoir exécuter un processus d'optimisation basé sur la co-simulation, puis d'échanger les composants de co-simulation et/ou d'optimisation avec peu d'efforts.

Conclusions

L'évaluation du *framework* a été réalisée à trois niveaux. Pour le niveau d'application, nous nous sommes concentrés sur les résultats des cas d'utilisation, à savoir les améliorations obtenues sur chaque système complexe après avoir utilisé les trois algorithmes. Au niveau de la mise en œuvre, nous avons tiré des conclusions sur les caractéristiques de la mise en œuvre de chaque composant en trouvant la configuration optimale pour chaque cas d'utilisation dans les conditions de chaque co-simulation ; c'est-à-dire que pour le problème de contrôle thermique de la maison, nous avons trouvé les paramètres qui améliorent le confort et réduisent la consommation d'énergie. Pour le micro-réseau autonome, nous avons trouvé la taille optimale du système de stockage qui minimise la production d'énergie restante tout en évitant le sous-dimensionnement du stockage. Et pour le projet Hy2Car, nous avons trouvé la configuration optimale du système de source hybride qui améliore l'efficacité de la consommation d'énergie en termes d'énergie utilisée et générée. Pour le composant de co-simulation, nous avons identifié l'effet des besoins en ressources informatiques de la co-simulation sur les résultats de l'optimisation. Pour la composante d'optimisation, nous avons considéré le théorème « No Free Lunch » (définition 25) comme une motivation pour inclure plusieurs méthodes d'optimisation, ce qui nous a permis d'identifier les conditions qui rendent chaque algorithme plus adapté à un certain type de problème. Le composant d'interface a produit un résultat notable en simplifiant le processus d'optimisation basé sur la co-simulation, à savoir la possibilité de passer d'un algorithme à un autre en ne modifiant que 3 à 4 mots-clés, et de changer d'une co-simulation à une autre en modifiant également seulement 3 à 4 champs. Pour le niveau architectural, nous avons identifié la diversité des domaines inclus (flexibilité des domaines), le nombre d'outils

logiciels fonctionnant dans le même environnement partagé (extensibilité) et la capacité du *framework* à échanger des co-simulations ou des algorithmes (manifestation de la modularité) en modifiant entre 7 et 10 mots-clés (prétraitement) dans un seul fichier ou une seule interface graphique (composant d'interface).

Cette thèse a contribué à la communauté scientifique à travers deux publications majeures. Un article a été présenté lors du 35e Symposium européen sur la modélisation et la simulation (EMSS 2023), où le *framework* proposé a été démontré à l'aide de deux cas d'utilisation représentatifs : le contrôle thermique résidentiel et un système de micro-réseau [5]. En outre, un article de revue fournissant une revue complète de l'état de l'art sur l'optimisation basée sur la co-simulation a été soumis à *ACM Computing Surveys*. L'article a reçu une décision initiale de *révision majeure* et fait actuellement l'objet d'un deuxième cycle d'évaluation par les pairs.

Au cours de l'évaluation du *framework*, nous avons identifié certaines limites concernant à la fois la structure et la mise en œuvre du *framework*. Nous avons souligné la possibilité de modifier la structure du multimodèle utilisé dans le composant de co-simulation. Dans ce contexte, deux aspects ont été proposés:

- Variation des Sous-Modèles : Nous avons introduit la stratégie conceptuelle d'utiliser la variation des sous-modèles dans le processus d'optimisation. Il est déjà possible d'effectuer ce type d'optimisation en utilisant la fonction de traitement par lots du *framework*, ce qui permet d'optimiser une co-simulation sous différentes structures. Cependant, cette fonctionnalité n'a pas été incluse dans les expériences menées pour la thèse, laissant ainsi cette capacité sans validation.
- Variation de la Structure de Couplage : Bien que nous ayons évoqué l'idée de varier la structure de couplage, nous n'avons pas tenté d'en fournir une définition conceptuelle ou pratique. L'influence de cette variation sur les méthodes d'optimisation de boîte noire reste encore floue. Ces perspectives soulignent des opportunités pour de futures recherches afin de valider et d'expérimenter ces approches prometteuses.

Appendix A

Abbreviations

Table A.1: Abbreviations.

Abbreviation	Full name
ANN	Artificial Neural Networks
BCD	Block Coordinate Descent
BFD	Best-Fit Decreasing
CMA-ES	Covariance matrix adaptation evolution strategy
CMOEA-MS	Constrained multi-objective evolutionary algorithm with multiple stages
COBYLA	Constrained Optimization By Linear Approximation
DEA	Differential Evolution Algorithm
E-S	EXTREMUM-SEEKING
FFD	First-Fit Decreasing
FOA	Fruit fly Optimization Algorithm
GDO	Goal-Driven Optimization
GOBI	Gradient-Based Optimization using Back-propagation to Input
IPOPT	Interior-Point Optimizer
LWM	Linear Weighted Method
MOR	Model Order Reduction
MPC	Model Predictive Control
NNIA	Non-Dominated Neighborhood Immune Algorithm
NSGA-II	Non-dominated Sorting Genetic Algorithm
OPF	Optimal Power Flow
PSO	Particle swarm optimization
RBFOpt	Radial Basis Function Optimization
RSM	Response Surface Methodology
SNOPT	Sparse Non-linear Optimizer
TOPSIS	Technique for Order Preference by Similarity to Ideal Solution
TS	Tabu Search algorithm
VVO	Volt-var optimization

Appendix B

Software compilation

Table B.1: Software description (part 1 of 2).

Software	Definition of the software	Ref
Abaqus	Finite Element Analysis Software	[110]
Adams	Multibody dynamics simulation	[111]
ADS	Advanced Design System, Electronic circuit simulation	[155]
AMESim	Thermo-hydraulic simulation software	[121]
AMPERES	Multi-Physics simulation for Electrochemical and Renewable Energy Storage	[145]
ANSYS	Engineering simulation software package	[139]
AnnAGNPS	Simulation software to evaluate management decisions on watersheds	[210]
APECS	Advanced Process Engineering Co-Simulator	[157]
API REST	Set of protocols for building and integrating applications software	[211]
AQWA	Hydrodynamic simulation for offshore and marine structures (Part of ANSYS)	[212]
ARENA	Discrete event simulation software	[213]
Aspen Plus	Process Simulation Software for the Chemical Industry	[214]
CarSim	Simulation Software of vehicle behavior	[112]
COMSOL	Software for Multiphysics Simulation	[142]
CST	3D electromagnetic software for design, analysis and optimization	[215]
C++	High-level and general-purpose programming language	[216]
DAKOTA	Software for optimization using advanced parametric analyses	[168]
DNP3	Client-Master Simulator	[217]
Docker	OS-level virtualization to deliver software in packages called containers.	[218]
DSHplus	Dynamic simulation of fluid power systems	[219]
Dymola	Dynamic complex systems simulation	[108]
Eclipse	Integrated development environment (IDE) used in computer programming	[220]
EDEM	Discrete Element Method (DEM) Software	[221]
EMTP-RV	Electromagnetic Transients Program	[222]
EnergyPlus	Building simulation software	[114]
ePHASORSIM	Real-time transient stability simulation	[223]
EXata CPS	Network communications emulator	[224]
FEKO	Electromagnetic Simulation	[225]
Flomaster	Thermo-fluid simulation	[122]
FLUENT	Fluid simulation software (Part of ANSYS)	[115]
FMI	Functional Mock-up Interface standard for model exchange and co-simulation	[107]
Forge	Hot and cold forging simulation	[109]
FrontFlow/Blue	FEM solver for flow prediction in complicated geometries and resulting sound	[226]
Gazebo	2D/3D robotics simulator	[227]
GenOpt	Optimization software that connects to external simulation programs	[171]

Table B.2: Software description (part 2 of 2).

Software	Definition of the software	Ref
Gleeble	Thermal-Mechanical Simulators	[156]
Grasshopper	Software for Parametric Modeling of Structural Engineering	[228]
GridLAB-D	Power distribution system simulator	[229]
GT-Power	Engine performance simulation	[116]
HFSS	High-Frequency Structure Simulation Software (Part of ANSYS)	[230]
HSPICE	Industry's 'gold standard' for accurate circuit simulation	[231]
iFogSim	Resource Management for IoT, Edge, and Fog Computing Simulation	[232]
IPS	Integrated Plasma Simulator Framework	[169]
Isight	Software for automating and optimizing simulation process flows.	[170]
Java	High-level, class-based, object-oriented programming language	[185]
LabView	Dynamic systems simulation	[151]
Lingo	Optimization Software for Linear, Non-linear, and Integer Programming	[233]
LS-DYNA	Multiphysics simulation software package (Part of ANSYS)	[117]
LTspice	Simulation Program with Integrated Circuit Emphasis	[234]
MATLAB	Numeric multi-paradigm programming language	[131]
MicroGP	A multi-purpose extensible self-adaptive evolutionary optimization tool	[235]
Maxwell	Low-Frequency EM-Field Simulation (Part of ANSYS)	[236]
ModeFRONTIER	Simulation software for process automation and optimization	[118]
ModelCenter	Multi-model optimization software (Part of ANSYS)	[237]
MODFLOW	Geological modular finite-difference simulation for groundwater flow	[238]
MoM	3D planar electromagnetic (EM) simulator (part of ADS)	[239]
MONSTER	Microkernel for scabrous terrain exploring robots	[240]
MotorCAD	Electric Machine design simulation	[241]
MOVES	MOtor Vehicle Emission Simulator (MOVES)	[242]
MT3DMS	Transport Model for Simulation of Contaminants in Groundwater Systems	[243]
NS-3	Discrete-event network simulator for Internet systems	[244]
OPAL-RT	Open Real-Time Digital Simulator	[165]
OpenDSS	Electric power distribution system simulator (DSS)	[245]
OpenModelica	Mathematical equations based modeling and simulation environment	[126]
Opnet	Network simulator	[246]
Opossum	Optimization Solver with Surrogate Models (Part of Grasshopper)	[247]
OptiY	Multidisciplinary design environment with modern optimization strategies	[172]
OSCAR	Pantograph/catenary interaction simulation software	[248]
Particle Flow Code	Discrete-Element Method simulation for synthetic materials	[249]
Pro ENGINEER	Simulation software for mechanical engineering design	[250]
PSIM	Electronic circuit simulation software package	[251]
PyBoolNet	Python package for the simulation of Boolean networks	[252]
Python	High-level, general-purpose programming language.	[253]
RecurDyn	Interdisciplinary computer-aided engineering software	[120]
RSCAD	Real-time simulation software	[254]
RTDS	Electromagnetic transient (EMT) simulations	[255]
SDM	Java program for System Dynamics models (.mdl)	[256]
Sigrity PowerDC	Electrical and thermal simulation software	[144]
Simpack	Multibody system Simulation	[123]
Simplorer	Multi-physics circuit simulator (Part of ANSYS)	[257]
SimulationX	Mechanical, hydraulic, pneumatic, electrical, and combined systems simulation	[258]
Simulink	Numerical simulation software	[113]
Solcore	Python-based library for modeling solar cells and semiconductor materials	[161]
Solidworks	Mechanical modeling en 2D y 3D	[259]
Spectre	Circuit simulator from Cadence	[260]
SPEED	Electric Machine Design Simulator	[261]
SWAT	Quasi-physically-based water quality simulator	[262]
SystemC	System design and verification language that spans hardware and software	[263]
TRNFLOW	Ventilation dynamics simulator	[264]
TRNSYS	Transient systems simulation	[124]
UCEF	Universal Cyber-Physical Systems Environment for Federation Simulation	[265]
UG	Advanced high-end Design, engineering, and manufacturing simulation software	[266]
UM	Dynamics of planar and spatial mechanical systems simulation	[267]
VISSIM	Microscopic traffic simulation software	[268]
VISUM	Macroscopic traffic simulation software	[269]
VSI Fortran	Implementation of the Fortran programming language software	[270]
Xilinx	Simulink add-on that enables the development of FPGA designs	[271]

Appendix C

Batch description for the first co-simulation use case

```
{ "optimization_instances" : [ {
    "opt_name" : "SimulationBased",
    "opt_parameters" : {
      "indicators" : [ "Objective" ],
      "cosimulation_parameters" : [ "minTemp", "maxTemp" ],
      "iterations" : 12,
      "steps" : [ 0.5, -0.5 ]
    },
    "experiment_path" : "Experiments/CompleteHouseCosim.xml"
  }, {
    "opt_name" : "GradientDescent",
    "opt_parameters" : {
      "indicators" : [ "Objective" ],
      "cosimulation_parameters" : [ "minTemp", "maxTemp" ],
      "learning_rate" : 0.002,
      "stop_criteria" : 0.01,
      "scale" : 0.1
    },
    "experiment_path" : "Experiments/CompleteHouseCosim.xml"
  }, {
    "opt_name" : "Hexaly",
    "opt_parameters" : {
      "indicators" : [ "Objective" ],
      "cosimulation_parameters" : [ "minTemp", "maxTemp" ],
      "lower_bounds" : [ 16.0, 16.1 ],
      "upper_bounds" : [ 27.9, 28.0 ],
      "precision" : 0.01,
      "time_limit" : 120.0,
      "constraints" : [ {
        "op" : "<",
        "lhs" : "minTemp",
        "rhs" : "maxTemp"
      } ]
    },
    "experiment_path" : "Experiments/CompleteHouseCosim.xml"
  }]
}
```

Appendix D

Batch description for the second co-simulation use case

```
{
  "optimization_instances" : [ {
    "opt_name" : "Simulationbased",
    "opt_parameters" : {
      "indicators" : [ "Objective" ],
      "cosimulation_parameters" : [ "maxCapacity" ],
      "iterations" : 20,
      "steps" : [ -5000.0 ]
    },
    "experiment_path" : "Experiments/completeAutonomousHouseNetwork_Opt.xml"
  },{
    "opt_name" : "GradientDescent",
    "opt_parameters" : {
      "indicators" : [ "Objective" ],
      "cosimulation_parameters" : [ "maxCapacity" ],
      "learning_rate" : 0.5,
      "stop_criteria" : 0.01,
      "scale" : 5000.0
    },
    "experiment_path" : "Experiments/completeAutonomousHouseNetwork_Opt.xml"
  }, {
    "opt_name" : "Hexaly",
    "opt_parameters" : {
      "indicators" : [ "Objective" ],
      "cosimulation_parameters" : [ "maxCapacity" ],
      "lower_bounds" : [ 200000.0 ],
      "upper_bounds" : [ 300000.0 ],
      "precision" : 0.01,
      "iterations" : 30
    },
    "experiment_path" : "Experiments/completeAutonomousHouseNetwork_Opt.xml"
  }]
}
```

Appendix E

Batch description for the third co-simulation use case

```
{ "optimization_instances" : [ {
    "opt_name" : "SimulationBased",
    "opt_parameters" : {
      "indicators" : [ "Objective" ],
      "cosimulation_parameters" : [ "Afc", "Nfc", "Np_sc" ],
      "iterations" : 5,
      "steps" : [ 0.0, 0.0, 1.0 ]
    },
    "experiment_path" : "Experiments/hy2carOptim.xml"
  }, {
    "opt_name" : "SimulationBased",
    "opt_parameters" : {
      "indicators" : [ "Objective" ],
      "cosimulation_parameters" : [ "Afc", "Nfc", "Np_sc" ],
      "iterations" : 5,
      "steps" : [ 0.0, 0.0, 1.0 ]
    },
    "experiment_path" : "Experiments/hy2carOptim.xml"
  }, {
    "opt_name" : "GradientDescent",
    "opt_parameters" : {
      "indicators" : [ "Objective" ],
      "cosimulation_parameters" : [ "Afc", "Nfc", "Np_sc" ],
      "learning_rate" : 0.5,
      "stop_criteria" : 0.01,
      "scale" : 10.0
    },
    "experiment_path" : "Experiments/hy2carOptim.xml"
  }, {
    "opt_name" : "Hexaly",
    "opt_parameters" : {
      "indicators" : [ "Objective" ],
      "cosimulation_parameters" : [ "Np_sc", "Afc", "Nfc" ],
      "lower_bounds" : [ 1.0, 100.0, 140.0 ],
      "upper_bounds" : [ 5.0, 200.0, 280.0 ],
      "precision" : 0.01,
      "iterations" : 100,
```

```
38        "time_limit" : 1200.0
39      },
40      "experiment_path" : "Experiments/hy2carOptim.xml"
41    }]
42  }
```

Bibliography

[1] B. P. Zeigler, H. Praehofer, and T. G. Kim, *Theory of modeling and simulation: integrating discrete event and continuous complex dynamic systems*. Academic press, 2000.

[2] J. A. Snyman, *Practical mathematical optimization*. Springer, 2005.

[3] Y. Zhao, P. A. Ioannou, and M. M. Dessouky, "A hierarchical co-simulation optimization control system for multimodal freight routing," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1–6, IEEE, 2017.

[4] M. Ferrara, J. C. Vallee, L. Shtrepi, A. Astolfi, and E. Fabrizio, "A thermal and acoustic co-simulation method for the multi-domain optimization of nearly zero energy buildings," *Journal of Building Engineering*, vol. 40, p. 102699, 2021.

[5] D. Vega and V. Chevrier, "A framework for co-simulation based optimization," in *Proceedings of the 35th European Modeling & Simulation Symposium (EMSS 2023) Article 024*, p. 10, CALTEK, 2023.

[6] J.-B. Wiart, *Approche hierarchique de co-simulation pour l'etude des echanges d'energie des micro-reseaux*. phdthesis, Université de Lorraine, 2023.

[7] C. Gomes, C. Thule, D. Broman, P. G. Larsen, and H. Vangheluwe, "Co-simulation: A survey," *ACM Computing Surveys*, vol. 51, may 2018.

[8] I. Graja, S. Kallel, N. Guermouche, S. Cheikhrouhou, and A. Hadj Kacem, "A comprehensive survey on modeling of cyber-physical systems," *Concurrency and Computation: Practice and Experience*, vol. 32, no. 15, p. e4850, 2020.

[9] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge University Press, 2004.

[10] A. Gosavi, *Simulation-Based Optimization*. Operations Research/Computer Science Interfaces Series, Springer US, 2nd ed., 2015.

[11] M. C. Fu, "Simulation optimization," in *Proceeding of the 2001 Winter Simulation Conference (Cat. No. 01CH37304)*, vol. 1, pp. 53–61, IEEE, 2001.

[12] A. M. Law and W. D. Kelton, *Simulation modeling and analysis*, vol. 3. Mcgraw-hill New York, 2000.

[13] L. M. Plà-Aragonés, *Handbook of operations research in agriculture and the agri-food industry*, vol. 224. Springer, 2015.

[14] E. Ramat, *Contributions à la modélisation et à la simulation des systèmes complexes*. Accreditation to supervise research, Université du Littoral Côte d'Opale - ULCO, Dec. 2003.

[15] H. A. Simon, "How complex are complex systems?," *PSA: Proceedings of the Biennial Meeting of the Philosophy of Science Association*, vol. 1976, no. 2, p. 507–522, 1976.

[16] D. Chavalarias, P. Bourgine, E. Perrier, F. Amblard, F. Arlabosse, P. Auger, J.-B. Baillon, O. Barreteau, P. Baudot, E. Bouchaud, *et al.*, "French roadmap for complex systems 2008-2009," *arXiv preprint arXiv:0907.2221*, 2009.

[17] E. Mumford, R. Hirschheim, G. Fitzgerald, and T. Wood-Harper, *Research methods in information systems*. North-Holland Amsterdam, 1985.

[18] H. A. Simon, "The Organization of Complex Systems," in *Models of Discovery: And Other Topics in the Methods of Science*, pp. 245–261, Dordrecht: Springer Netherlands, 1977.

[19] E. Ramat, "Introduction à la simulation: principaux concepts," *Modélisation et Simulation Multi-Agent: application pour les Sciences de l'Homme et de la Société*, pp. 37–60, 2006.

[20] M. L. Minsky, "Matter, minds and models," in *Semantic Information Processing* (M. L. Minsky, ed.), MIT Press, 1965.

[21] B. P. Zeigler, A. Muzy, and E. Kofman, *Theory of modeling and simulation: discrete event & iterative system computational foundations*. Academic press, 2018.

[22] L. Yilmaz and T. Oren, "Dynamic model updating in simulation with multimodels: A taxonomy and a generic agent-based architecture," *Simulation Series*, vol. 36, no. 4, p. 3, 2004.

[23] P. Palensky, A. van der Meer, C. Lopez, A. Joseph, and K. Pan, "Applied cosimulation of intelligent power systems: Implementing hybrid simulators for complex power systems," *IEEE Industrial Electronics Magazine*, vol. 11, no. 2, pp. 6–21, 2017.

[24] H. S. Sarjoughian and B. Zeigler, "Devsjava: Basis for a devs-based collaborative m&s environment," *Simulation Series*, vol. 30, pp. 29–36, 1998.

[25] T. Paris, *Modelisation de Systemes complexes par composition*. PhD thesis, Universite de Lorraine, 2019.

[26] B. Poggi, *Développement de concepts et outils d'aide à la décision pour l'optimisation via simulation*. PhD thesis, SPE UMR CNRS 6134; Université de Corse, 2014.

[27] P.-A. Bisgambiglia, E. de Gentili, P. A. Bisgambiglia, and J. F. Santucci, "Fuzz-idevs: towards a fuzzy toolbox for discrete event systems," in *Proceedings of the 2nd International Conference on Simulation Tools and Techniques*, Simutools '09, (Brussels, BEL), ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2009.

[28] P. Von Hilgers, "The History of the Black Box: The Clash of a Thing and Its Concept," *Cultural Politics*, vol. 7, pp. 41–58, Mar. 2011.

[29] W. R. Elwasif, D. E. Bernholdt, S. Pannala, S. Allu, and S. S. Foley, "Parameter sweep and optimization of loosely coupled simulations using the dakota toolkit," in *2012 IEEE 15th International Conference on Computational Science and Engineering*, pp. 102–110, IEEE, 2012.

[30] S. E. Zitney, "Advanced co-simulation for computer-aided process design and optimization of fossil energy systems with carbon capture," in *Design for Energy and the Environment*, pp. 211–228, CRC Press, 2009.

[31] S. Alarie, C. Audet, A. E. Gheribi, M. Kokkolaras, and S. L. Digabel, "Two decades of blackbox optimization applications," *EURO Journal on Computational Optimization*, vol. 9, p. 100011, 2021.

[32] A. M. Law and M. G. McComas, "Simulation-based optimization," in *Proceedings of the Winter Simulation Conference*, vol. 1, pp. 41–44, IEEE, 2002.

[33] E. Kaddoum, *La coopération pour l'adaptation autonome et continue dans les systèmes multi-agents.* Habilitation à diriger des recherches, Université Toulourse Jean Jaurès, Mar. 2025.

[34] D. Weyns, M. U. Iftikhar, S. Malek, and J. Andersson, "Claims and supporting evidence for self-adaptive systems: A literature study," in *2012 7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, pp. 89–98, IEEE, 2012.

[35] S. Galland, N. Gaud, J. DEMANGE, and A. Koukam, "Environment model for multiagent-based simulation of 3d urban systems," in *7th European Workshop on Multiagent Systems (EUMAS09), Ayia Napa, Cyprus*, 04 2013.

[36] D. Zhou, L. Lu, and C. Yang, "Optimization of high speed permanent magnet synchronous motor based on co-simulation method," in *2023 5th Asia Energy and Electrical Engineering Symposium (AEEES)*, pp. 460–465, IEEE, 2023.

[37] S. A. Assadi, O. Tayyara, J. Palumbo, A. Chen, M. S. Zaman, C. Da Silva, S. Chandra, C. H. Amon, and O. Trescases, "Electro-thermal co-design of a high-density power-stage for a reconfigurable-battery-assisted electric-vehicle fast-charger using multi-physics co-simulation and topology optimization," in *2023 IEEE Applied Power Electronics Conference and Exposition (APEC)*, pp. 1808–1815, IEEE, 2023.

[38] J. Hou, S. Li, W. Pan, and L. Yang, "Co-simulation modeling and multi-objective optimization of dynamic characteristics of flow balancing valve," *Machines*, vol. 11, no. 3, p. 337, 2023.

[39] S. Tuli, S. R. Poojara, S. N. Srirama, G. Casale, and N. R. Jennings, "Cosco: Container orchestration using co-simulation and gradient based optimization for fog computing environments," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 1, pp. 101–116, 2022.

[40] U. Oh, N. Nonaka, and J. Ishimoto, "Valve optimization with system-fluid-magnetic co-simulation and design of experiments," *International Journal of Automotive Technology*, vol. 23, no. 3, pp. 683–692, 2022. Publisher: Korean Society of Automotive Engineers.

[41] W. Tan, Z. Chen, Z. Li, and H. Yan, "Thermal-fluid-solid coupling simulation and oil groove structure optimization of wet friction clutch for high-speed helicopter," *Machines*, vol. 11, no. 2, p. 296, 2023.

[42] Q. Zhang, Q. Deng, X. Shan, X. Kang, and Z. Ren, "Optimization of the thermal environment of large-scale open space with subzone-based temperature setting using bem and cfd coupling simulation," *Energies*, vol. 16, no. 7, p. 3214, 2023.

[43] C. Hu, Y. Zhao, Z. Zhang, H. Zhang, and D. Chen, "Optimization of flow field structure for proton exchange membrane fuel cell stack by multi-physics coupling simulation," *International Journal of Electrochemical Science*, vol. 18, no. 7, p. 100195, 2023.

[44] J. Zhao, Y. Lu, X. Wang, J. Zhuang, and Z. Han, "A bionic profiling-energy storage device based on MBD-DEM coupled simulation optimization reducing the energy consumption of deep loosening," *Soil and Tillage Research*, vol. 234, 2023. Publisher: Elsevier B.V.

[45] B. Zheng, S. Fu, and J. Lei, "Titanium alloy piston thermo-mechanical coupling simulation and multi-objective optimization design," *ADVANCES IN MECHANICAL ENGINEERING*, vol. 14, no. 4, 2022.

[46] M. Ahmed, C. Oekermann, and F. Kirchner, "Cosimulation environment for mechanical design optimization with evolutionary algorithms," in *World Symp. Comput. Appl. Res., WSCAR*, pp. 1–6, Institute of Electrical and Electronics Engineers Inc., 2014. Journal Abbreviation: World Symp. Comput. Appl. Res., WSCAR.

[47] D. R. Herber and J. T. Allison, "Nested and simultaneous solution strategies for general combined plant and control design problems," *Journal of Mechanical Design*, vol. 141, no. 1, p. 011402, 2019.

[48] J. S. Gray, J. T. Hwang, J. R. Martins, K. T. Moore, and B. A. Naylor, "Openmdao: An open-source framework for multidisciplinary design, analysis, and optimization," *Structural and Multidisciplinary Optimization*, vol. 59, no. 4, pp. 1075–1104, 2019.

[49] D. Hulse and C. Hoyle, "Understanding resilience optimization architectures: alignment and coupling in multilevel decomposition strategies," *Journal of Mechanical Design*, vol. 144, no. 11, p. 111704, 2022.

[50] Y. Cao, W. Yu, W. Ren, and G. Chen, "An overview of recent progress in the study of distributed multi-agent coordination," *IEEE Transactions on Industrial informatics*, vol. 9, no. 1, pp. 427–438, 2012.

[51] S. Elo and H. Kyngäs, "The qualitative content analysis process," *Journal of advanced nursing*, vol. 62, no. 1, pp. 107–115, 2008.

[52] X. Gui and X. Fu, "Optimization of heat dissipation and control scheme for wind turbine gearbox based on flomaster-simulink co-simulation," in *Journal of Physics: Conference Series*, vol. 2584, p. 012049, IOP Publishing, 2023.

[53] M. A. Aftab, A. Chawla, P. P. Vergara, S. Ahmed, and C. Konstantinou, "Volt/var optimization in the presence of attacks: A real-time co-simulation study," in *2023 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, pp. 1–7, IEEE, 2023.

[54] H. Xu, J. Wang, Z.-F. Zhou, Z. Yi, M. Qin, and Q.-A. Huang, "System-level optimization of mems thermal wind sensor based on the co-simulation of macromodel and board-level interface circuits," *Journal of Microelectromechanical Systems*, 2023.

[55] E. Arslan, M. Suveren, and S. T. H. Moghaddam, "Ros gazebo and matlab/simulink co-simulation for cart-pole system: A framework for design optimization," in *2023 7th International Symposium on Innovative Approaches in Smart Technologies (ISAS)*, pp. 1–7, IEEE, 2023.

[56] A. Rüdenauer, S. Han, M. Geimer, *et al.*, "Optimization of the development process of mobile machines using a standardized co-simulation.," *Landtechnik*, vol. 67, no. 2, pp. 122–126, 2012.

[57] Y. Wang, X. Fan, and Q. Cai, "Optimization for ev fuse based on thermal-electrical coupled simulation," in *2022 7th International Conference on Power and Renewable Energy (ICPRE)*, pp. 786–790, IEEE, 2022.

[58] R. Schemmel, V. Krieger, T. Hemsel, and W. Sextro, "Co-simulation of matlab and ansys for ultrasonic wire bonding process optimization," *Microelectronics Reliability*, vol. 119, p. 114077, 2021.

[59] C. Waibel, R. Evins, and J. Carmeliet, "Co-simulation and optimization of building geometry and multi-energy systems: Interdependencies in energy supply, energy demand and solar potentials," *Applied Energy*, vol. 242, pp. 1661–1682, 2019.

[60] P. J. McCurdy, K. Pattawi, C. Wang, T. Roth, C. Nguyen, Y. Liu, and H. Lee, "Validation approach for energy optimization models of grid-interactive buildings using co-simulation," in *ASME International Mechanical Engineering Congress and Exposition*, vol. 85642, p. V08BT08A041, American Society of Mechanical Engineers, 2021.

[61] D. Lee, M. C. Lim, L. Negash, and H.-L. Choi, "Eppy based building co-simulation for model predictive control of hvac optimization," in *2018 18th International Conference on Control, Automation and Systems (ICCAS)*, pp. 1051–1055, IEEE, 2018.

[62] G. Cucca and A. Ianakiev, "Assessment and optimisation of energy consumption in building communities using an innovative co-simulation tool," *Journal of Building Engineering*, vol. 32, p. 101681, 2020.

[63] E. Borkowski, M. Donato, G. Zemella, D. Rovas, and R. Raslan, "Optimisation of controller parameters for adaptive building envelopes through a co-simulation interface: A case study," in *Proceedings of Building Simulation 2019: 16th Conference of IBPSA*, pp. 1372–1379, International Building Performance Association (IBPSA), 2019.

[64] Z. Du, P. Xu, X. Jin, and Q. Liu, "Temperature sensor placement optimization for vav control using cfd–bes co-simulation strategy," *Building and Environment*, vol. 85, pp. 104–113, 2015.

[65] H. Elarga, A. Dal Monte, R. K. Andersen, and E. Benini, "Pv-pcm integration in glazed building. co-simulation and genetic optimization study," *Building and Environment*, vol. 126, pp. 161–175, 2017.

[66] M. P. Fanti, A. M. Mangini, M. Roccotelli, F. Iannone, and A. Rinaldi, "A natural ventilation control in buildings based on co-simulation architecture and particle swarm optimization," in *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 002621–002626, IEEE, 2016.

[67] A. Rahnama, M. Xia, S. Wang, and P. J. Antsaklis, "Passivation and performance optimization using an extremum seeking co-simulation framework with application to adaptive cruise control systems," in *2016 American Control Conference (ACC)*, pp. 6109–6114, IEEE, 2016.

[68] H. H. Hu, "Chapter 10 - computational fluid dynamics," in *Fluid Mechanics (Fifth Edition)* (P. K. Kundu, I. M. Cohen, and D. R. Dowling, eds.), pp. 421–472, Boston: Academic Press, fifth edition ed., 2012.

[69] J. Tan, Q. Li, B. Zhao, C. Ma, Y. Wu, Y. Li, J. Zhang, Y. He, T. Ye, C. Zhang, *et al.*, "Multi-objective optimization design of plate-type fuel via co-simulation method," *Annals of Nuclear Energy*, vol. 169, p. 108914, 2022.

[70] J.-S. Park, S.-W. Kim, H.-C. Lim, and J.-H. Kang, "Flow stress optimization of inconel 718 based on a coupled simulation of material-forming analysis and joule heating analysis," *Metals*, vol. 12, no. 12, p. 2024, 2022.

[71] Y. Li, X. Qin, Z. Zhang, and H. Dong, "Operation parameters optimization of a separating system for non-ferrous metal scraps from end-of-life vehicles based on coupled simulation," *Waste Management*, vol. 120, pp. 667–674, 2021.

[72] H. Hu, Z. Zhou, W. Wu, W. Yang, T. Li, C. Chang, W. Ren, and X. Lei, "Distribution characteristics and parameter optimisation of an air-assisted centralised seed-metering device for rapeseed using a cfd-dem coupled simulation," *Biosystems Engineering*, vol. 208, pp. 246–259, 2021.

[73] Y. Wang, H. Li, H. Hu, J. He, Q. Wang, C. Lu, P. Liu, D. He, and X. Lin, "Dem–cfd coupling simulation and optimization of a self-suction wheat shooting device," *Powder Technology*, vol. 393, pp. 494–509, 2021.

[74] Q. Zhaoju, L. Yingsong, Y. Zhenzhong, D. Junfa, and W. Lijun, "Diesel engine piston thermo-mechanical coupling simulation and multidisciplinary design optimization," *Case Studies in Thermal Engineering*, vol. 15, p. 100527, 2019.

[75] D. Han, D. Zhang, H. Jing, L. Yang, T. Cui, Y. Ding, Z. Wang, Y. Wang, and T. Zhang, "Dem-cfd coupling simulation and optimization of an inside-filling air-blowing maize precision seed-metering device," *Computers and Electronics in Agriculture*, vol. 150, pp. 426–438, 2018.

[76] N. Wirth, P. Bayrasy, B. Landvogt, K. Wolf, F. Cecutti, and T. Lewandowski, "Analysis and optimization of flow around flexible wings and blades using the standard co-simulation interface mpcci," *Recent Progress in Flow Control for Practical Flows: Results of the STADYWICO and IMESCON Projects*, pp. 283–321, 2017.

[77] X. J. Hou, S. Chen, and Z. Liu, "Egr system performance optimization of diesel engine based on gt-power and fluent co-simulation," *Advanced Materials Research*, vol. 860, pp. 1703–1709, 2014.

[78] K. Nozaki, H. Ohsaki, M. Tanaka, A. Vanhirtum, and E. Sakane, "Strategic optimization of computation nodes allocation in a coupled simulation: Computational fluid dyanmics and aero acoustic simulations," in *2012 IEEE/IPSJ 12th International Symposium on Applications and the Internet*, pp. 211–214, IEEE, 2012.

[79] A. Tolk, S. Y. Diallo, and C. D. Turnitsa, "Applying the levels of conceptual interoperability model in support of integratability, interoperability, and composability for system-of-systems engineering," *Journal of Systems, Cybernetics, and Informatics*, vol. 5, no. 5, 2007.

[80] M. Inzillo and C. Kavka, "Multi-disciplinary optimization with standard co-simulation interfaces.," in *ICSOFT*, pp. 453–458, 2017.

[81] B. Zhang, H. Guo, Y. Zhang, Z. Li, Y. Liu, S. Wang, and Z. Fu, "A coupling simulation and optimization method developed for environmental-economic management of lake watershed," *Journal of Environmental Management*, vol. 318, p. 115546, 2022.

[82] A. C. Scogna and L. Teoh, "Sipi co-extraction and spice co-simulation for package on-die decap optimization," in *2016 IEEE 20th Workshop on Signal and Power Integrity (SPI)*, pp. 1–4, IEEE, 2016.

[83] T. Hamiti, C. Gerada, and M. Rottach, "Weight optimisation of a surface mount permanent magnet synchronous motor using genetic algorithms and a combined electromagnetic-thermal co-simulation environment," in *2011 IEEE energy conversion congress and exposition*, pp. 1536–1540, IEEE, 2011.

[84] S. Zhang, J. Zhang, M. Meng, P. Chen, X. Liu, G. Liu, and Z. Gu, "A multi-objective decision making system (mdms) for a small agricultural watershed based on meta-heuristic optimization coupling simulation," *Water*, vol. 13, no. 10, p. 1338, 2021.

[85] J.-P. Massat, C. Laurent, J.-P. Bianchi, and E. Balmès, "Pantograph catenary dynamic optimisation based on advanced multibody and finite element co-simulation tools," *Vehicle System Dynamics*, vol. 52, no. sup1, pp. 338–354, 2014.

[86] T. Schulze, J. Weber, K. Großmann, L. Penter, and C. Schenke, "Hydraulic die cushions in deep drawing presses: Analysis and optimization using coupled simulation," in *Fluid Power Systems Technology*, vol. 57236, p. V001T01A066, American Society of Mechanical Engineers, 2015.

[87] M. Manbachi, A. Sadu, H. Farhangi, A. Monti, A. Palizban, F. Ponci, and S. Arzanpour, "Real-time co-simulation platform for smart grid volt-var optimization using iec 61850," *IEEE Transactions on Industrial Informatics*, vol. 12, no. 4, pp. 1392–1402, 2016. cited By 56.

[88] M. Manbachi, A. Sadu, H. Farhangi, A. Monti, A. Palizban, F. Ponci, and S. Arzanpour, "Impact of ev penetration on volt–var optimization of distribution networks using real-time co-simulation monitoring platform," *Applied Energy*, vol. 169, pp. 28–39, 2016.

[89] R. F. Hassan, A. R. Ajel, S. J. Abbas, and A. J. Humaidi, "Fpga based hil co-simulation of 2dof-pid controller tuned by pso optimization algorithm," *ICIC Express Letters*, vol. 16, no. 12, pp. 1269–1278, 2022.

[90] X. Zhou, B. Zhao, and G. Gong, "Control parameters optimization based on co-simulation of a mechatronic system for an ua-based two-axis inertially stabilized platform," *Sensors*, vol. 15, no. 8, pp. 20169–20192, 2015.

[91] J. Darvill, A. Tisan, and M. Cirstea, "A novel psim and matlab co-simulation approach to particle swarm optimization tuning of pid controllers," in *2014 International Conference on Optimization of Electrical and Electronic Equipment (OPTIM)*, pp. 784–789, IEEE, 2014.

[92] A. Schirrer and M. Kozek, "Co-simulation as effective method for flexible structure vibration control design validation and optimization," in *2008 16th Mediterranean conference on control and automation*, pp. 481–486, 2008.

[93] Y. Wang, L. Wang, C. Li, Z. Xue, Y. Sun, R. Ma, D. Wang, M. Cui, X. Wei, L. Tang, *et al.*, "Optimization of excavator bucket structure by a coupled simulation method," *Applied Sciences*, vol. 13, no. 20, p. 11336, 2023.

[94] L. Sun, R. Zhang, C. Du, W. Rong, X. Li, Y. Chen, T. Fu, S. Cao, and D. Shi, "Optical optimization of ultra-thin crystalline silicon solar cells by a co-simulation approach of fem and ga," *Applied Physics A*, vol. 127, no. 7, pp. 1–9, 2021.

[95] S. M. Mohseni-Bonab, A. Hajebrahimi, A. Moeini, and I. Kamwa, "Optimization application in integrated transmission and distribution operation: Co-simulation approach," in *2020 IEEE Power & Energy Society General Meeting (PESGM)*, pp. 1–5, IEEE, 2020.

[96] K. Norouzi Khatiri, M. H. Niksokhan, A. Sarang, and A. Kamali, "Coupled simulation-optimization model for the management of groundwater resources by considering uncertainty and conflict resolution," *Water Resources Management*, vol. 34, no. 11, pp. 3585–3608, 2020.

[97] I. Arsie, I. Criscuolo, L. De Simio, and S. Iannaccone, "Optimization of control parameters for a heavy-duty cng engine via co-simulation analysis," tech. rep., SAE Technical Paper, 2011.

[98] W. Sun and S. S. Zhu, "The mechanical excavator-hydraulic system of the establishment of co-simulation and optimization," in *Advanced Materials Research*, vol. 706, pp. 1651–1656, Trans Tech Publ, 2013.

[99] J. Mou and Z. Shen, "Co-simulation and co-optimization strategy for active absorber with periodic structure," in *2017 IEEE Electrical Design of Advanced Packaging and Systems Symposium (EDAPS)*, pp. 1–4, IEEE, 2017.

[100] S. Koziel, "Robust optimization of microwave structures using co-simulation-based surrogate models," in *2011 IEEE International Symposium on Antennas and Propagation (APSURSI)*, pp. 2924–2927, IEEE, 2011.

[101] S. Kozie and J. W. Bandler, "Fast design optimization of microwave structures using co-simulation-based tuning space mapping," *Applied Computational Electromagnetics Society Journal*, vol. 26, no. 8, p. 631, 2011.

[102] Z. Zhao, K. Wei, W. Ding, W. Du, and H. Li, "Um-simulink co-simulation for the vibration reduction optimization of a magnetorheological damping steel-spring floating slab track," *Construction and Building Materials*, vol. 307, p. 124923, 2021.

[103] H. He, X. Ruan, J. Wu, F. Zhao, F. Wu, L. Wang, and J. Yin, "Structure optimization of natural gas engine's intake pipe based on one-dimensional and three-dimensional coupled simulation," in *IOP Conference Series: Earth and Environmental Science*, vol. 781, p. 042029, IOP Publishing, 2021.

[104] R. Bottura, A. Borghetti, F. Napolitano, and C. A. Nucci, "Ict-power co-simulation platform for the analysis of communication-based volt/var optimization in distribution feeders," in *ISGT 2014*, pp. 1–5, IEEE, 2014.

[105] L. Zhang, "System generator model-based fpga design optimization and hardware co-simulation for lorenz chaotic generator," in *2017 2nd Asia-Pacific Conference on Intelligent Robot Systems (ACIRS)*, pp. 170–174, IEEE, 2017.

[106] C. Brecher, W. Klein, and M. Tannert, "Optimization of multi-stage closed-die forging processes by coupled simulation of the machine and the forging processes," *Production Engineering*, vol. 4, pp. 279–286, 2010.

[107] "Fmi standard." https://fmi-standard.org/, 2024. Modelica Association Project. Accessed: 31.05.2024.

[108] "Dymola software." https://www.3ds.com/products/catia/dymola, 2002. Dassault Systèmes. Accessed: 31.05.2024.

[109] "Forge software." https://www.transvalor.com/en/forge, 2024. TRANSVALOR. Accessed: 31.05.2024.

[110] "Abaqus software." https://www.3ds.com/products/simulia/abaqus, 2002. Dassault Systèmes. Accessed: 28.05.2024.

[111] "Adams software." https://hexagon.com/products/product-groups/computer-aided-engineering-software/adams, 2024. Hexagon AB and/or its subsidiaries. Accessed: 28.05.2024.

[112] "Carsim software." https://www.carsim.com/, 2005. Mechanical Simulation Corporation. Accessed: 31.05.2024.

[113] "Simulink software." https://www.mathworks.com/products/simulink-online.html, 1994. The MathWorks Inc. Accessed: 31.05.2024.

[114] "Energyplus software." https://energyplus.net/, 2021. U.S. Department of Energy's and Building Technologies Office. Accessed: 31.05.2024.

[115] "Fluent software." https://www.ansys.com/products/fluids/ansys-fluent, 2024. ANSYS Inc. Accessed: 31.05.2024.

[116] "Gt-power software." https://www.gtisoft.com/gt-power/, 2024. Gamma Technologies, LLC. Accessed: 03.06.2024.

[117] "Ls-dyna software." https://lsdyna.ansys.com/, 2023. LS-DYNA. Accessed: 31.05.2024.

[118] "modefrontier software." https://engineering.esteco.com/modefrontier/, 2024. ESTECO SpA. Accessed: 31.05.2024.

[119] "Model.connect software." https://www.avl.com/en/simulation-solutions/software-offering/simulation-tools-a-z/modelconnect, 2024. AVL. Accessed: 05.06.2024.

[120] "Recurdyn software." https://functionbay.com/en/page/single/2/recurdyn-overview, 2024. FunctionBay Inc. Accessed: 31.05.2024.

[121] "Amesim software." https://plm.sw.siemens.com/en-US/simcenter/systems-simulation/amesim/, 2024. Siemens Digital Industries Software. Accessed: 28.05.2024.

[122] "Simcenter flomaster software." https://plm.sw.siemens.com/es-ES/simcenter/systems-simulation/flomaster/, 2024. Siemens. Accessed: 31.05.2024.

[123] "Simpack software." https://www.3ds.com/products/simulia/simpack, 2002. Dassault Systèmes. Accessed: 31.05.2024.

[124] "Transient system simulation tool software." https://www.trnsys.com/, 2019. TRNSYS. Accessed: 31.05.2024.

[125] F. R. Holzinger and M. Benedikt, "Hierarchical coupling approach utilizing multi-objective optimization for non-iterative co-simulation," in *Proceedings of the 13th International Modelica Conference, Regensburg, Germany, March 4–6, 2019*, vol. 157, pp. 1–6, Linköping University Electronic Press, 2019.

[126] "Openmodelica software." https://openmodelica.org/, 2024. OpenModelica. Accessed: 31.05.2024.

[127] A. Pasetti, *Software frameworks and embedded control systems*. No. 2231 in Lecture notes in computer science, Berlin ; New York: Springer, 2002. OCLC: ocm49036010.

[128] Y. Barve, H. Neema, Z. Kang, H. Vardhan, H. Sun, and A. Gokhale, "Exppo: Execution performance profiling and optimization for cps co-simulation-as-a-service," *Journal of Systems Architecture*, vol. 118, 2021. cited By 0.

[129] R. Sadnan, N. Gray, A. Dubey, and A. Bose, "Distributed optimization for power distribution systems with cyber-physical co-simulation," in *2021 IEEE Power & Energy Society General Meeting (PESGM)*, pp. 1–5, IEEE, 2021.

[130] T. D. Hardy, B. Palmintier, P. L. Top, D. Krishnamurthy, and J. C. Fuller, "Helics: A co-simulation framework for scalable multi-domain modeling and analysis," *IEEE Access*, vol. 12, pp. 24325–24347, 2024.

[131] "Matlab software." `https://www.mathworks.com/products/matlab.html`, 1994. MathWorks Inc. Accessed: 31.05.2024.

[132] E. Balogun, E. Buechler, S. Bhela, S. Onori, and R. Rajagopal, "Ev-ecosim: A grid-aware co-simulation platform for the design and optimization of electric vehicle charging infrastructure," *IEEE Transactions on Smart Grid*, vol. 15, no. 3, pp. 3114–3125, 2024.

[133] H. Mao, M. Jing, Z. Fan, J. Lu, X. Xiao, Z. Fang, Z. Liu, M. Chen, and Y. Jiang, *A Multi-Body Structure Optimization and Fully Coupled Hydrodynamic Simulation Method for Marine Semi-Submersible Equipment*, pp. 740–749. IOS Press, 11 2023.

[134] H. Sun, W. Li, L. Zheng, S. Ling, and W. Fu, "Adaptive co-simulation method and platform application of drive mechanism based on fruit fly optimization algorithm," *Progress in Nuclear Energy*, vol. 153, p. 104397, 2022.

[135] H. Lu, H. Zhao, A. Huang, D. Kim, J. Sun, J. Hu, and H. Kim, "High power wireless power transfer efficiency and emi co-optimization based on fast field-circuit co-simulation," in *2019 IEEE International Conference on Computational Electromagnetics (ICCEM)*, pp. 1–3, IEEE, 2019.

[136] Y. Hou, Z. Shi, S. Li, Y. Zhang, J. Bai, S. Jia, and L. Wang, "Co-simulation on the optimization design of high-speed electromagnetic repulsion mechanism of vacuum circuit breaker," in *2016 27th International Symposium on Discharges and Electrical Insulation in Vacuum (ISDEIV)*, vol. 2, pp. 1–4, IEEE, 2016.

[137] W. Deng, B. Mao, B. Liang, and P. Song, "Co-simulation of stabilization accuracy optimization of overhead weapon station," in *2015 International conference on Applied Science and Engineering Innovation*, pp. 972–977, Atlantis Press, 2015.

[138] X. Yan, C. Ma, L. Shi, Y. Zhuo, X. J. Zhou, L. Wei, and R. Xue, "Optimization of an 8-channel loop-array coil for a 7 t mri system with the guidance of a co-simulation approach," *Applied Magnetic Resonance*, vol. 45, pp. 437–449, 2014.

[139] "Ansys software." `https://www.ansys.com/`, 2024. ANSYS Inc. Accessed: 28.05.2024.

[140] M. Loupis, "A toolset for the design of embedded systems, enabling hw/sw co-simulation, performance optimisation and source code generation," in *2017 Panhellenic Conference on Electronics and Telecommunications (PACET)*, pp. 1–4, IEEE, 2017.

[141] M. S. Berger, J. Soler, H. Yu, M. Tsagkaropoulos, Y. Leclerc, and C. Olma, "Methodology and toolset for model verification, hardware/software co-simulation, performance optimisation and customisable source-code generation," *WSEAS Transactions on Information Science and Applications*, vol. 6, no. 10, pp. 169–178, 2013.

[142] "Comsol software." `https://www.comsol.com/`, 2024. COMSOL. Accessed: 31.05.2024.

[143] F. Hou, Y. Zhou, F. Liu, M. Su, C. Chen, J. Li, T. Lin, and L. Cao, "Optimization design of 2.5 d tsv package using thermo-electrical co-simulation method," in *2016 IEEE 66th Electronic Components and Technology Conference (ECTC)*, pp. 1964–1969, IEEE, 2016.

[144] "Sigrity powerdc software." `https://www.cadence.com/zh_CN/home/tools/ic-package-design-and-analysis/si-pi-analysis-point-tools/sigrity-powerdc.html`, 2024. Cadence Design Systems Inc. Accessed: 03.06.2024.

[145] "Amperes software." `https://www.integratedsoft.com/products/Amperes`, 2021. INTEGRATED Engineering Software. Accessed: 28.05.2024.

[146] P. Chen, P. Ioannou, and M. Dessouky, "Mixed freight dynamic routing using a co-simulation optimization approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 8, pp. 12833–12845, 2021.

[147] Y. Zhao, P. A. Ioannou, and M. M. Dessouky, "Dynamic multimodal freight routing using a co-simulation optimization approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 7, pp. 2657–2667, 2019.

[148] Y. Zhao and P. Ioannou, "A co-simulation, optimization, control approach for traffic light control with truck priority," *Annual Reviews in Control*, vol. 48, pp. 283–291, 2019. cited By 3.

[149] Y. Zhao, P. Ioannou, and M. Maged, "Routing of multimodal freight transportation using a co-simulation optimization approach," in *Transportation Research Board 96th Annual Meeting*, 2017.

[150] K. B. Khoud, S. Bouallègue, and M. Ayadi, "Design and co-simulation of a fuzzy gain-scheduled pid controller based on particle swarm optimization algorithms for a quad tilt wing unmanned aerial vehicle," *Transactions of the Institute of Measurement and Control*, vol. 40, no. 14, pp. 3933–3952, 2018.

[151] "Labview software." `https://www.ni.com/en/support/downloads/software-products/download.labview-control-design-and-simulation-module.html#460270`, 2024. NATIONAL INSTRUMENTS CORP. Accessed: 31.05.2024.

[152] "Rhino 3d." `https://www.rhino3d.com/`, 1993. Robert McNeel & Associates TLM Inc. Accessed: 03.07.2024.

[153] S. Yang, A. Khusro, W. Li, M. Vaseem, M. Hashmi, and A. Shamim, "Optimization of ann-based models and its em co-simulation for printed rf devices," *International Journal of RF and Microwave Computer-Aided Engineering*, vol. 32, no. 3, p. e23012, 2022.

[154] P. Kurgan, S. Koziel, L. Leifsson, and X. Du, "Expedite design of variable-topology broadband hybrid couplers for size reduction using surrogate-based optimization and co-simulation coarse models," *Procedia Computer Science*, vol. 108, pp. 1483–1492, 2017.

[155] "Ads software." `https://www.keysight.com/us/en/products/software/pathwave-design-software/pathwave-advanced-design-system.html`, 2000. Keysight Technologies. Accessed: 28.05.2024.

[156] "Gleeble software." `https://gleeble.com/`, 2024. Dynamic Systems Inc. Accessed: 31.05.2024.

[157] "Advanced process engineering co-simulator (apecs)." `https://www.colan.org/experiences-projects/advanced-process-engineering-co-simulator-apecs/`, 2001. CAPE OPEN LABORATORIES NETWORK. Accessed: 28.05.2024.

[158] L. Giannantoni, R. Bardini, and S. Di Carlo, "A methodology for co-simulation-based optimization of biofabrication protocols," in *Bioinformatics and Biomedical Engineering* (Rojas I., Valenzuela O., Rojas F., Herrera L.J., and Ortuño F., eds.), vol. 13347 LNBI, pp. 179–192, Springer Science and Business Media Deutschland GmbH, 2022. Journal Abbreviation: Lect. Notes Comput. Sci.

[159] I. de Jong, "Pyro - python remote objects - 4.82." https://pyro4.readthedocs.io/en/stable/index.html, 2024. Accessed: 23.07.2024.

[160] T. H. Anderson, B. J. Civiletti, P. B. Monk, and A. Lakhtakia, "Coupled optoelectronic simulation and optimization of thin-film photovoltaic solar cells," *Journal of Computational Physics*, vol. 407, p. 109242, 2020.

[161] D. Alonso-Alvarez, T. Wilson, P. Pearce, M. Fuhrer, D. Farrell, and N. Ekins-Daukes, "Solcore software." https://www.solcore.solar/, 2018. Accessed: 31.05.2024.

[162] M. Yamanee-Nolin, A. Löfgren, N. Andersson, B. Nilsson, M. Max-Hansen, and O. Pajalic, "Single-shooting optimization of an industrial process through co-simulation of a modularized aspen plus dynamics model," in *Computer Aided Chemical Engineering*, vol. 46, pp. 721–726, Elsevier, 2019.

[163] "Ros: Robot operating system." https://www.ros.org/, 2021. Open Robotics. Accessed: 22.07.2024.

[164] B. Udugama, "Mini bot 3d: A ros based gazebo simulation," *arXiv preprint arXiv:2302.06368*, 2023.

[165] "Opal-rt software." https://www.opal-rt.com/, 2024. OPAL-RT TECHNOLOGIES Inc. Accessed: 31.05.2024.

[166] G. R. Bharati, S. Chakraborty, and J. Darrah, "A co-simulation-based hardware-in-the-loop architecture for validating power systems optimization with large-scale grid models," in *2021 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT)*, pp. 1–5, IEEE, 2021.

[167] M. Ahmed, Y.-H. Yoo, and F. Kirchner, "A co-simulation framework for design, test and parameter optimization of robotic systems," in *ISR 2010 (41st International Symposium on Robotics) and ROBOTIK 2010 (6th German Conference on Robotics)*, pp. 1–6, VDE, 2010.

[168] "Dakota software." https://dakota.sandia.gov/, 2024. National Technology and Engineering Solutions of Sandia, LLC. Accessed: 31.05.2024.

[169] "Integrated plasma simulator." https://ipsframework.sourceforge.net/doc/html/, 2011. SWIM Project. Accessed: 31.05.2024.

[170] "Isight software." https://www.3ds.com/products/simulia/isight, 2002. Dassault Systèmes. Accessed: 31.05.2024.

[171] "Genopt software." https://simulationresearch.lbl.gov/GO/, 2021. Lawrence Berkeley National Laboratory. Accessed: 03.06.2024.

[172] "Optiy software." https://www.optiy.eu/default.htm, 2005. OptiY GmbH. Accessed: 31.05.2024.

[173] D. Wolpert and W. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.

[174] Y. Ma, B. Li, Z. Wu, H. Wu, and Z. Chen, "A co-simulation method of power amplifier for reliability optimization," in *2018 IEEE International Conference on Electron Devices and Solid State Circuits (EDSSC)*, pp. 1–2, IEEE, 2018.

[175] J. M. Kovitz, V. Manohar, and Y. Rahmat-Samii, "Feed horn optimization using feed+ reflector co-simulation for advanced reflector antennas," in *2017 International Applied Computational Electromagnetics Society Symposium-Italy (ACES)*, pp. 1–2, IEEE, 2017.

[176] E. Abisset-Chavanne and F. Chinesta, "Toward an optimisation of the reactive resin transfer molding process: thermo-chemico-mechanical coupled simulations," *International journal of material forming*, vol. 7, pp. 249–258, 2014.

[177] G. Manzi and U. Muehlmann, "Passive uhf rfid sensor/transponder antenna optimization for backscatter operation by electromagnetic-circuital co-simulation," in *Proceedings of the 11th International Conference on Telecommunications*, pp. 17–22, IEEE, 2011.

[178] G. Ayad, J. Song, T. Barriere, B. Liu, and J. Gelin, "A fully coupled simulation and optimization scheme for the design of 3d powder injection molding processes," in *AIP Conference Proceedings*, vol. 908 (1), pp. 507–512, American Institute of Physics, 2007.

[179] M. M.-U.-T. Chowdhury, M. S. Hasan, and S. Kamalasadan, "A Novel Voltage Optimization Co-Simulation Framework for Electrical Distribution System With High Penetration of Renewables," *IEEE Transactions on Industry Applications*, vol. 61, pp. 1080–1090, Jan. 2025.

[180] M. Dubravac, M. Žnidarec, K. Fekete, and D. Topié, "A New Co-Simulation Approach of Active Power Curtailment for Voltage Optimization in PV-Rich Distribution Networks," in *2023 IEEE 6th International Conference and Workshop Óbuda on Electrical and Power Engineering (CANDO-EPE)*, (Budapest, Hungary), pp. 000065–000070, IEEE, 2023.

[181] J. Guo, J. Qi, J. Hu, C. Hong, Y. Shen, H. Huang, and W. Liu, "Robust optimization method for co-simulation of equipment based on EDEM-ADAMS," *Simulation Modelling Practice and Theory*, vol. 142, p. 103124, July 2025.

[182] J. Lan, W. Zou, Q. Xu, Y. Lai, and S. Zhu, "Model-Based Co-Simulation Method for PLC Programming: Interaction Design and Optimization," in *2024 IEEE 33rd International Symposium on Industrial Electronics (ISIE)*, (Ulsan, Korea, Republic of), pp. 1–6, IEEE, June 2024.

[183] A. Powell and G. R. Mudalige, "Predictive Analysis of Code Optimisations on Large-Scale Coupled CFD-Combustion Simulations using the CPX Mini-App," in *2023 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, (St. Petersburg, FL, USA), pp. 579–589, IEEE, May 2023.

[184] P. D. Romero, C. Toprak, L. Duipmans, S. Van Waasen, and L. Geck, "Co-Simulation for Automated Optimization of Integrated Cryogenic Qubit Electronics," in *2025 21st International Conference on Synthesis, Modeling, Analysis and Simulation Methods, and Applications to Circuits Design (SMACD)*, (Istanbul, Turkiye), pp. 1–4, IEEE, July 2025.

[185] "Java software." `https://www.java.com/en/`, 2024. Oracle. Accessed: 31.05.2024.

[186] "Mecsyco software." `http://www.mecsyco.com/`, 2017. Université de Lorraine and Inria. Accessed: 03.07.2024.

[187] "Eclipse help." `https://help.eclipse.org/latest/index.jsp`, 2025. Eclipse Foundation AISBL. Accessed: 13.08.2025.

[188] B. Camus, *Multi-agent environment for multi-modeling and simulation of complex systems.* PhD thesis, Universite de Lorraine, 2016.

[189] B. Camus, T. Paris, J. Vaubourg, Y. Presse, C. Bourjot, L. Ciarletta, and V. Chevrier, "Co-simulation of cyber-physical systems using a devs wrapping strategy in the mecsyco middleware," *Simulation*, vol. 94, no. 12, pp. 1099–1127, 2018.

[190] B. Camus, T. Paris, J. Vaubourg, Y. Presse, C. Bourjot, L. Ciarletta, and V. Chevrier, "MECSYCO: a Multi-agent DEVS Wrapping Platform for the Co-simulation of Complex Systems," research report, LORIA, UMR 7503, Université de Lorraine, CNRS, Vandoeuvre-lès-Nancy ; Inria Nancy - Grand Est (Villers-lès-Nancy, France), Sept. 2016.

[191] B. Camus, C. Bourjot, and V. Chevrier, "Combining devs with multi-agent concepts to design and simulate multi-models of complex systems." working paper or preprint, Jan. 2015.

[192] J. Vaubourg, V. Chevrier, L. Ciarletta, and B. Camus, "Co-simulation of ip network models in the cyber-physical systems context, using a devs-based platform," in *Communications and Networking Simulation Symposium*, ACM, 2016.

[193] J. Vaubourg, *Intégration de modèles de réseaux IP à un multi-modèle DEVS, pour la co-simulation de systèmes cyber-physiques*. PhD thesis, Université de Lorraine, 2017.

[194] J. Vaubourg, Y. Presse, B. Camus, C. Bourjot, L. Ciarletta, V. Chevrier, J.-P. Tavella, and H. Morais, "Multi-agent multi-model simulation of smart grids in the ms4sg project," in *International Conference on Practical Applications of Agents and Multi-Agent Systems*, pp. 240–251, Springer, 2015.

[195] B. Camus, V. Galtier, and M. Caujolle, "Hybrid co-simulation of fmus using dev &dess in mecsyco," in *2016 Symposium on Theory of Modeling and Simulation (TMS-DEVS)*, pp. 1–8, 2016.

[196] "Open neural network exchange onnx api." `https://onnx.ai/`, 2019. The Linux Foundation. Accessed: 12.09.2025.

[197] T. Paris, J.-B. Wiart, D. Netter, and V. Chevrier, "Teaching co-simulation basics through practice," in *2019 SUMMER SIMULATION CONFERENCE*, 2019.

[198] "Hexaly software overview." `https://www.hexaly.com/docs/last/index.html`, 2025. Hexaly. Accessed: 27.08.2025.

[199] "Ecma-404 - the json data interchange syntax." `https://ecma-international.org/publications-and-standards/standards/ecma-404/`, 2013. ECMA. Accessed: 02.09.2025.

[200] T. Paris, L. Ciarletta, and V. Chevrier, "Designing co-simulation with multi-agent tools: a case study with netlogo," in *European Conference on Multi-Agent Systems*, pp. 253–267, Springer, 2017.

[201] CNRS, "Retour sur l'inauguration de la plateforme HyMob : pour l'élaboration de véhicules électriques familiaux adaptés à des courts trajets | Délégation Centre-Est du CNRS." `https://www.centre-est.cnrs.fr/fr/cnrsinfo/retour-sur-linauguration-de-la-plateforme-hymob-pour-lelaboration-de-vehicules-electriques`, 2025. Accessed: 19.08.2025.

[202] F. Lapicque, M.-F. Agnoletti, C. Bonnet, M. Bouizakarne, V. Chevrier, O. Dufaud, M. Hinaje, M.-N. Pons, S. Raël, and Y.-Q. Song, "Low demand solution for (sub)urban personal transport by a direct hybrid (fuel cell-supercapacitors) source," *Electrochimica Acta*, vol. 487, p. 144128, May 2024.

[203] J. Alvarez-Ramirez, A. Morales, and I. Cervantes, "Robust proportional-integral control," *Industrial & Engineering Chemistry Research*, vol. 37, no. 12, pp. 4740–4747, 1998.

[204] "Openroute service." `https://openrouteservice.org/`, 2025. HeiGIT gGmbH. Accessed: 27.08.2025.

[205] D. A. Vega and V. Chevrier, "Co-simulation-based optimization: state of the art." preprint, Aug. 2024.

[206] S. Ruder, "An overview of gradient descent optimization algorithms," 2017.

[207] T. Benoist, B. Estellon, F. Gardi, R. Megel, and K. Nouioua, "Localsolver 1. x: a black-box local-search solver for 0-1 programming," *4or*, vol. 9, no. 3, pp. 299–316, 2011.

[208] A. Khanna, F. Liu, S. Gupta, S. Pavia, A. Mukhopadhyay, and A. Dubey, "Pdptw-db: Milp-based offline route planning for pdptw with driver breaks," in *Proceedings of the 26th International Conference on Distributed Computing and Networking*, pp. 73–83, 2025.

[209] L. Chen, K.-U. Bletzinger, N. R. Gauger, and Y. Ye, "A gradient descent akin method for constrained optimization: algorithms and applications," *Optimization Methods and Software*, vol. 0, no. 0, pp. 1–28, 2024.

[210] "Annagnps software." `https://www.appsolutelydigital.com/ModelPrimer/chapter2_section1.html`, 2010. United States Department of Agriculture. Accessed: 28.05.2024.

[211] A. Soni and V. Ranga, "Api features individualizing of web services: Rest and soap," *International Journal of Innovative Technology and Exploring Engineering*, vol. 8, no. 9, pp. 664–671, 2019.

[212] "Aqwa software." `https://www.ansys.com/webinars/ansys-simulation-of-hydrodynamic-wave-loading-on-offs` 2024. ANSYS Inc. Accessed: 28.05.2024.

[213] "Arena software." `https://www.rockwellautomation.com/en-us/products/software/arena-simulation.html`, 2024. Rockwell Automation. Accessed: 28.05.2024.

[214] "Aspen plus software." `https://www.aspentech.com/en/products/engineering/aspen-plus`, 2024. Aspen Technology Inc. Accessed: 28.05.2024.

[215] "Cst studio suite." `https://www.3ds.com/products/simulia/cst-studio-suite`, 2002. Dassault Systèmes. Accessed: 31.05.2024.

[216] "Standard c++." `https://isocpp.org/`, 2024. C++ Foundation. Accessed: 28.06.2024.

[217] "Dnp3 client master simulator." `https://www.freyrscada.com/dnp3-ieee-1815-Client-Simulator.php`, 2024. FreyrSCADA. Accessed: 31.05.2024.

[218] "Docker." `https://www.docker.com/`, 2024. Docker Inc. Accessed: 31.05.2024.

[219] "Dshplus software." `https://fluidon.com/en/tools/dshplus`, 2024. Altair Engineering Inc. Accessed: 31.05.2024.

[220] "Eclipse." `https://www.eclipse.org/downloads/`, 2024. Eclipse Foundation AISBL. Accessed: 31.05.2024.

[221] "Altair edem software." `https://altair.com/edem/features`, 2024. Altair Engineering Inc. Accessed: 31.05.2024.

[222] "Emtp-rv software." `https://www.emtp.com/`, 2024. ELECTRICITE DE France and HYDRO-QUEBEC and RTE RESEAU DE TRANSPORT D'ELECTRICITE, Now called EMTP, Accessed: 31.05.2024.

[223] "ephasorsim software." `https://www.opal-rt.com/systems-ephasorsim/`, 2024. OPAL-RT Technologies Inc. Accessed: 31.05.2024.

[224] "Exata cps software." `https://www.keysight.com/us/en/product/SN0500ECPA/exata-network-modeling-critical-infrastructure.html`, 2000. Keysight Technologies. Accessed: 31.05.2024.

[225] "Feko software." `https://altair.com/feko`, 2024. Altair Engineering Inc. Accessed: 31.05.2024.

[226] "Frontflow/blue software." `http://www.ciss.iis.u-tokyo.ac.jp/project/rss/software/07_info.html`, 2024. Collaborative Center for Research on Innovative Simulation Software. Accessed: 03.06.2024.

[227] "Gazebo software." `https://gazebosim.org/home`, 2002. Open Robotics. Accessed: 31.05.2024.

[228] "Grasshopper." `https://www.rhino3d.com/en/features/#grasshopper`, 1993. Robert McNeel & Associates (TLM Inc.) Accessed: 03.06.2024.

[229] "Gridlab-d software." https://www.gridlabd.org/, 2023. U.S. Department of Energy. Accessed: 31.05.2024.

[230] "Hfss software." https://www.ansys.com/products/electronics/ansys-hfss, 2024. ANSYS Inc. Accessed: 31.05.2024.

[231] "Hspice software." https://www.synopsys.com/implementation-and-signoff/ams-simulation/primesim-hspice.html, 2024. Synopsys Inc. Accessed: 31.05.2024.

[232] "ifogsim software." http://www.cloudbus.org/cloudsim, 2017. Cloud Computing and Distributed Systems (CLOUDS) Laboratory. Accessed: 31.05.2024.

[233] "Lingo software." https://www.eostrack.com/lingo#:~:text=OVERVIEW,faster%2C%20easier%20and%20more%20efficient., 2023. EOSTrack Engineering & Agencies. Accessed: 31.05.2024.

[234] "Ltspice software." https://www.analog.com/en/resources/design-tools-and-calculators/ltspice-simulator.html, 2024. Analog Devices Inc. Accessed: 03.06.2024.

[235] G. Squillero, "Microgp3 software." https://ugp3.sourceforge.net/, 2006. Accessed: 31.05.2024.

[236] "Maxwell software." https://www.ansys.com/products/electronics/ansys-maxwell#:~:text=What%20is%20Ansys%20Maxwell%3F,varying%20magnetic%20and%20electric%20fields., 2024. ANSYS Inc. Accessed: 31.05.2024.

[237] "Modelcenter software." https://www.ansys.com/products/connect/ansys-modelcenter, 2024. ANSYS Inc. Accessed: 31.05.2024.

[238] "Modflow." https://www.usgs.gov/mission-areas/water-resources/science/modflow-and-related-programs, 2022. U.S. Department of the Interior. Accessed: 31.05.2024.

[239] "Mom software." https://www.remcom.com/resources/articles-and-papers/3d-electromagnetic-simulation-vs-planar-mom, 2024. Remcom. Accessed: 31.05.2024.

[240] D. Spenneberg, M. Albrecht, and T. Backhaus, "Monster: A new behavior-based microkernel for mobile robots," *Proceedings of the ECMR*, vol. 2005, 2005.

[241] "Motor-cad software." https://www.ansys.com/products/electronics/ansys-motor-cad, 2024. ANSYS Inc. Accessed: 31.05.2024.

[242] "Moves software." https://www.epa.gov/moves/latest-version-motor-vehicle-emission-simulator-move 2024. U.S. Environmental Protection Agency. Accessed: 31.05.2024.

[243] "Mt3dms software." https://www.aquaveo.com/software/gms-mt3dms, 2021. Aquaveo, LLC. Accessed: 31.05.2024.

[244] "ns-3 software." https://www.nsnam.org/, 2011. nsnam. Accessed: 31.05.2024.

[245] "Opendss software." https://www.epri.com/pages/sa/opendss, 2001. Electric Power Research Institute Inc. Accessed: 31.05.2024.

[246] "Opnet software." https://opnetprojects.com/opnet-network-simulator/, 2005. OPNET SIMULATOR PROJECTS. Accessed: 31.05.2024.

[247] "Optimization solver with surrogate models software." https://www.food4rhino.com/en/app/opossum-optimization-solver-surrogate-models, 2024. McNeel Europe. Accessed: 31.05.2024.

[248] "Oscar software." https://www.sdtools.com/solutions/oscar/, 2002. SDTools. Accessed: 03.06.2024.

[249] "Particle flow code software." https://www.itasca.fr/en/software/pfc, 2024. Itasca Consultants. Accessed: 31.05.2024.

[250] "Proengineer software." https://www.ptc.com/en/products/creo/pro-engineer, 2024. PTC. Accessed: 31.05.2024.

[251] "Psim software." https://altair.com/psim, 2024. Altair Engineering Inc. Accessed: 31.05.2024.

[252] H. Klarner, "Pyboolnet software." https://pyboolnet.readthedocs.io/en/master/installation.html, 2024. Accessed: 31.05.2024.

[253] "python software." https://www.python.org/, 2001. Python Software Foundation. Accessed: 31.05.2024.

[254] "Rscad software." https://www.rtds.com/technology/graphical-user-interface, 2024. RTDS Technologies Inc AMETEK. Accessed: 31.05.2024.

[255] "Rtds software." https://www.rtds.com/, 2024. RTDS Technologies Inc AMETEK. Accessed: 31.05.2024.

[256] "Sdm software." https://systemdynamics.org/resources-old/sdm-doc/, 2024. SDS. Accessed: 31.05.2024.

[257] "Simplorer software." https://www.ansys.com/training-center/course-catalog/electronics/introduction-to-ansys-simplorer, 2024. ANSYS Inc. Accessed: 31.05.2024.

[258] "Simulationx software." https://www.esi-group.com/products/simulationx, 2024. ESI Group. Accessed: 31.05.2024.

[259] "Solidworks software." https://www.solidworks.com/, 2002. Dassault Systèmes. Accessed: 03.06.2024.

[260] "Spectre software." https://www.cadence.com/en_US/home/tools/custom-ic-analog-rf-design/circuit-simulation/spectre-x-simulator.html, 2024. Cadence Design Systems Inc. Accessed: 31.05.2024.

[261] "Speed software." https://www.maccon.com/cae-software/simcenter-speed.html, 2024. MACCON Bespoke electric motors and drive electronics. Accessed: 31.05.2024.

[262] "Swat software." https://swat.tamu.edu/software/, 2024. College of Agriculture and Life Sciences and Texas A&M AgriLife Extension Service and Texas A&M AgriLife Research. Accessed: 31.05.2024.

[263] "Systemc software." https://systemc.org/overview/systemc/, 2024. Accellera Systems Initiative. Accessed: 31.05.2024.

[264] "Trnflow software." https://aiguasol.coop/energy-software/trnflow-ventilation-module-for-trnsys-18/?privacy=updated, 2024. Aiguasol. Accessed: 31.05.2024.

[265] "Ucef: Universal cps environment for federation software." https://www.nist.gov/ctl/smart-connected-systems-division/iot-devices-and-infrastructures-group/ucef-universal-cps, 2021. National Institute of Standards and Technology. Accessed: 31.05.2024.

[266] "Unigraphics (ug) nx software." https://toolingtechgroup.com/stamping-design-division/capabilities/software/unigraphics-ug-nx/, 2024. Tooling Tech Group. Accessed: 03.06.2024.

[267] "Universal mechanism software." https://universalmechanism.com/en/pages/index.php?id=1, 2024. Laboratory of Computational Mechanics. Accessed: 31.05.2024.

[268] "Vissim software." https://www.ptvgroup.com/en/products/ptv-vissim, 2024. PTV Planung Transport Verkehr GmbH. Accessed: 31.05.2024.

[269] "Visum software." https://www.ptvgroup.com/en/products/ptv-visum, 2024. PTV Planung Transport Verkehr GmbH. Accessed: 31.05.2024.

[270] "Vsi fortran software." https://vmssoftware.com/products/fortran/, 2014. VMS Software Inc. Accessed: 31.05.2024.

[271] "Xilinx software." https://www.xilinx.com/support/download.html, 2024. Advanced Micro Devices Inc. Accessed: 31.05.2024.

Abstract

The growing presence of product complexity in the industry is expanding the need to understand and predict the behavior of systems while considering the internal complexity. A complex big-picture problem can be studied by combining the knowledge of discipline experts through interdisciplinary collaboration. We present a tool to study complex systems that uses interdisciplinary collaboration to capture the essential mechanisms of the system and produce a similar behavior, known as co-simulation. The result of the co-simulation process is a complex model that can be combined with optimization to offer high-level insights into the system before a real-life deployment is needed. This combination of methodologies is known as co-simulation-based optimization and is already present in the literature, as shown by the 96 articles using co-simulation-based optimization found in the literature review conducted.

The literature review produced findings such as a new classification of the co-simulation application, where we identified that the way a co-simulation is developed can follow an inductive reasoning modeling process (Bottom-up approach), starting from small models to build a big complex model, or a deductive reasoning modeling process (Top-down approach), which starts from a complex system and builds the components as smaller models connected. Another result of the review is a wide and heterogeneous landscape of optimization strategies and domains involved in the co-simulation, that is, we found more than 25 different algorithms and 14 disciplines being combined in different ways within each co-simulation.

We leverage the findings of the literature review as challenges to propose a new general framework that can be reused for a faster and simpler implementation of co-simulation-based optimization for the study and improvement of complex systems. We use the architectural patterns found in the literature to propose a general co-simulation-based optimization framework that intends to provide a flexible structure in terms of optimization algorithms and simulation domains.

This led to the main contribution of this thesis, a modular, domain-flexible, and extensible framework for co-simulation-based optimization that includes a generalized interface component to handle the interaction between co-simulation and optimization, and a definition of the pre-processing information required to launch the process in a summarized and simplified manner. We perform an implementation as a software framework deployed in a Java-based environment following the conceptual framework guidelines, and finally, we study three heterogeneous complex systems (case studies) in combination with 3 optimization algorithms to showcase the functionality of the framework.

We analyze the results of the case studies' optimization. For case study 1, we optimized the air-conditioning system to improve the comfort of the occupants of a house. In case study 2, we found the optimal capacity of the storage system of a network of 4 autonomous houses to ensure minimum energy loss. In case study 3, we developed the co-simulation-based optimization for the design of a hydrogen hybrid vehicle (part of the project Hy2Car). The result is the co-simulation of the vehicle and the optimization insights that contributed to the optimal proposal for the hybrid source configuration.

We assess the big picture contribution of the thesis, which is the relevance of the architecture proposed when working with different complex systems being studied in the same environment. The proposed architecture achieved the optimization with three different algorithms of the three case studies, which required the configuration of a single configuration file, completely avoiding the necessity for code modification to launch an experiment, effectively simplifying the study and improvement of the target complex systems.

Keywords: Co-simulation, optimization, general framework, black-box approach, complex systems.

Résumé

La complexité croissante des produits dans l'industrie conduit à la nécessité de comprendre et de prédire le comportement des systèmes tout en tenant compte de leur complexité interne. Un problème global complexe peut être étudié en combinant les connaissances d'experts de différentes disciplines via une collaboration interdisciplinaire. Nous présentons un outil permettant d'étudier des systèmes complexes, utilisant la collaboration interdisciplinaire pour saisir les mécanismes essentiels du système et produire un comportement similaire, connu sous le nom de co-simulation. Le résultat du processus de co-simulation est un modèle complexe, pouvant être combiné avec l'optimisation pour offrir des informations de haut niveau sur le système préalablement à un déploiement réel. Cette combinaison de méthodologies est connue sous le nom d'optimisation basée sur la co-simulation et est déjà présente dans la littérature, comme le montrent les 96 articles utilisant l'optimisation basée sur la co-simulation.

L'exploration de la littérature a permis d'aboutir à certaines conclusions, telles qu'une nouvelle classification des applications de co-simulation. Nous avons identifié la manière dont une co-simulation est développée peut suivre un processus de modélisation par raisonnement inductif (approche ascendante), partant de petits modèles pour en construire de plus complexes ; ou un processus de modélisation par raisonnement déductif (approche descendante), partant d'un système complexe et construisant les composants sous forme de petits modèles connectés. Un autre résultat de cette analyse est le large et hétérogène panel des stratégies d'optimisation et des domaines impliqués dans la co-simulation. En effet, nous avons trouvé plus de 25 algorithmes différents et 14 disciplines combinés de différentes manières au sein de chaque co-simulation.

Nous nous appuyons sur les conclusions de la littérature pour proposer un nouveau cadre général réutilisable, afin d'accélérer et de simplifier la mise en œuvre de l'optimisation basée sur la co-simulation pour l'étude et l'amélioration de systèmes complexes. Nous utilisons des modèles architecturaux existants pour proposer un cadre général d'optimisation basé sur la co-simulation, visant à fournir une structure flexible en termes d'algorithmes d'optimisation et de domaines de simulation.

Cela a conduit à la principale contribution de cette thèse : un cadre modulaire, flexible et extensible pour l'optimisation basée sur la co-simulation, incluant un composant d'interface généralisé pour gérer l'interaction entre la co-simulation et l'optimisation, ainsi qu'une définition des informations de prétraitement nécessaires pour l'exécution simple de l'ensemble. Nous réalisons une implémentation sous la forme d'un cadre logiciel déployé dans un environnement Java, suivant les lignes directrices du cadre conceptuel, et étudions trois systèmes complexes hétérogènes (études de cas) en combinaison avec trois algorithmes d'optimisation afin de démontrer la fonctionnalité du cadre.

Nous analysons les résultats de l'optimisation pour chaque cas. Pour l'étude de cas n° 1, nous optimisons le système de climatisation afin d'améliorer le confort des occupants d'une maison. Dans l'étude de cas n° 2, nous déterminons la capacité optimale du système de stockage d'un réseau de 4 maisons autonomes afin de garantir une perte d'énergie minimale. Dans l'étude de cas n° 3, nous développons l'optimisation basée sur la co-simulation pour la conception d'un véhicule hybride à hydrogène (dans le cadre du projet Hy2Car). Le résultat est la co-simulation du véhicule et les connaissances en matière d'optimisation contribuant à la proposition optimale pour la configuration de la source hybride.

Nous évaluons la contribution globale de la thèse, à savoir la pertinence de l'architecture proposée lorsqu'on travaille avec différents systèmes complexes étudiés dans le même environnement. L'architecture proposée a permis d'optimiser les trois études de cas à l'aide de trois algorithmes différents, nécessitant la configuration d'un seul fichier de configuration et évitant ainsi complètement le besoin de modifier le code pour lancer une expérience, simplifiant efficacement l'étude et l'amélioration des systèmes complexes cibles.

Mots-clés: Co-simulation, optimisation, architecture générale, approche boîte noire, systèmes complexes.