A framework for co-simulation based optimization

Diego Alejandro Vega, Vincent Chevrier

▶ **To cite this version:**

Diego Alejandro Vega, Vincent Chevrier. A framework for co-simulation based optimization. The 35th European Modeling & Simulation Symposium, Sep 2023, Athenes, Greece. 10.46354/i3m.2023.emss.024 . hal-04413078

HAL Id: hal-04413078
https://hal.science/hal-04413078

Submitted on 23 Jan 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution - NonCommercial - NoDerivatives 4.0 International License

35th European Modeling & Simulation Symposium
20th International Multidisciplinary Modeling & Simulation Multiconference

2724-0029 © 2023 The Authors.
doi: 10.46354/i3m.2023.emss.024

A framework for co-simulation based optimization

Diego Alejandro Vega[1,*] and Vincent Chevrier[1]

[1]Université de Lorraine, CNRS, LORIA, 615 Rue du Jardin-Botanique, Vandœuvre-lès-Nancy, 54506, France

[*]Corresponding author. Email address: diego-alejandro.vega-vega@loria.fr

Abstract

Simulation-based engineering has become a cornerstone when designing new systems or products. This approach is coupled with an optimization phase in which the result of the engineering is tailored to optimize different aspects of the design (sizing components, minimizing cost, etc.). In the case of complex systems, such as cyber-physical systems, their modeling and simulation calls for multi- domains expertise. As a consequence, a simulation-based approach that integrates different simulators is needed, this is called co-simulation. This article presents a framework situated in the co-simulation basis for implementing optimization, the objective is to sustain the robustness of the multi-disciplinary approach and benefit from the large variety of simulation-based optimization algorithms. This framework will be proposed based on existing applications in engineering as well as some study cases to show the benefits of reusing architectural patterns. The implementation allows users to change optimization methods and/or co-simulation elements with the modification of fewer than 10 lines of code.

Keywords: Co-simulation; optimization; multi-disciplinary; complex systems; modeling.

1. Introduction

The growing product complexity in the industry is pushing the current design process to improve. The study of these complex systems relies on efficient multi-level comprehension that simultaneously ensures product specification (Rüdenauer et al., 2012). Furthermore, it is not enough to achieve an abstraction of the system, the idea is to be able 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 (Graja et al., 2020). Currently, truly complex engineered systems that also involve Cyber-Physical Systems (CPS) are emerging (Lee, 2008) and several fields of research have already shown interest and benefits from this kind of research, such as energy (Bharati et al., 2021), mobility (Zhao and Ioannou, 2019) and robotics (Ahmed et al., 2010).

Usually, the study of systems is done using modeling and simulation, which will provide an environment to understand and/or predict its behavior (Ramat, 2006). Also, the availability of a computational representation allows the generation of digital tests that otherwise would require the fabrication of potentially expensive or faulty prototypes. Nevertheless, with most complex systems, simulation architecture is not enough to represent the different detail levels and domains that can be present in a complex system. For this reason, co-simulation is presented as an approach to deal with the complexity (Chavalarias et al., 2009). Co-simulation is the composition of coupled simulations using synchronized computation tasks to achieve a global simulation (Gomes et al., 2017; Schirrer and Kozek, 2008). This architecture supports heterogeneous sub-models in terms of software or domains in order to maintain several levels of detail.

In 1997, the most significant emerging technology within the simulation field was optimization (Fu, 2001; Law and Kelton, 2000), this is, the problem of finding a

© 2023 The Authors. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY-NC-ND) license (https://creativecommons.org/licenses/by-nc-nd/4.0/).

value that minimizes or maximizes some specific function among all possible values that satisfies some conditions or constraints given (Boyd et al., 2004). The interest in this combination occurred when many companies started to develop an orchestrator of simulations that could be able to schedule several system configurations so that eventually an optimal or near-optimal solution was obtained (Law and McComas, 2002). Consequently, fields such as agriculture are profiting from the wide range of applicability of simulation-based optimization techniques in different domains and scales (Plà-Aragonés, 2015) as well as many widely different, albeit related, areas of operations research (Gosavi et al., 2015).

This paper is focused on making tangible the promising combination of co-simulation with optimization. Starting from the already-known benefits of the combination of optimization and simulation, and continuing with the exploration of new advantages that complex systems can bring to the table. For this reason, this article proposes a framework for the implementation of optimization in co-simulations. Alongside this proposal, the co-simulation software MECSYCO (Multi-agent Environment for Complex SYstem CO-simulation) is presented as a co-simulation tool where the framework is implemented.

The paper's organization is as follows. A review of the works relating to co-simulation and optimization is presented in Section 2 to understand the relevance and pertinence of this proposal, highlighting the emerging patterns that can help create a general framework. In Section 3 the framework proposed is described in detail as well as the study cases and the optimization algorithms implemented. In Section 4 the results of the study cases are presented and analyzed including a discussion regarding the contributions done and the challenges remaining. Finally, Section 5 contains the conclusions of the framework proposed.

2. Related work

The combined use of co-simulation and optimization is already present in the literature. In bibliographic databases, 64 publications are found linking these two concepts (search done in Web of Science, Scopus, and DBLP on November 2022 using the following equation: TITLE ("co-simulation" AND optimization)). These articles contain different examples of optimization applied to a complex system abstraction, which in these cases is the definition of some constraints on the parameters of a model to explore and find an optimal behavior of a co-simulation (Boyd et al., 2004). The domains present in the search include city mobility cases (Zhao and Ioannou, 2019), mathematical science problems (Sun et al., 2021; Wang and Ma, 2018), among others. This multidisciplinary applicability underlines the relevancy and usefulness of the topic among scientific and engineering domains.

This section contains a literature survey that reveals some general patterns that could guide any co-simulation user to include an optimization process in a complex system representation. Articles provide a description of the implementation used for the application, without any concern of reuse, or any guidelines for someone wanting to redo the same kind of work in another domain or context. However, among the papers, few are related to general contributions to co-simulation architectures coupled with optimization tools. In subsection 2.1, we describe the two articles found that have a general architecture proposal for the problem. Section 2.2 presents the different ways that parameter variation is managed in the optimization process. Section 3.5 presents some examples of optimization methods used in the literature. Finally, Section 2.4 discusses all the patterns found in the optimization of co-simulations that can be generalized and reused.

2.1. Frameworks

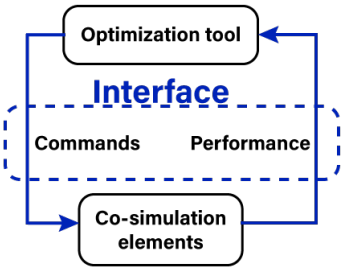

Figure 1. Robotic systems co-simulation framework, adapted from (Ahmed et al., 2010).

A Framework is a form of software that promotes the reuse of architectures within a defined application domain (Pasetti, 2002). In the literature can be found 2 examples of frameworks within the domain of this research, which are described as follows:

- A framework proposed to help design and test robotic systems using co-simulation is proposed by Ahmed et al. (2010). The architecture connects the co-simulation sub-systems (coupled simulations) with the optimization tool (in this case MATLAB/SIMULINK) as shown in Figure 1. The framework implements a loop-based communication between the co-simulation and the algorithm that allows sending feedback information from the co-simulation to the optimization tool and then commands to the co-simulation to carry out the scenario executions.
- The COSMO methodology is proposed in a domain-specific cluster of mobility scenarios (Zhao and Ioannou, 2019). This is a layer-based methodology to connect the different components that a co-simulation integration with optimization could present. As shown in Figure 2 the general structure proposes an upstream and downstream information flow between several layers, this communication handles the transition between the levels of abstraction of the problem. The

bottom part is the most tangible and concrete of the layers, containing physical systems in place, and at the other end, at the top of the layers is the most abstract one with the optimization algorithm. This hierarchy requires that the intermediate layers work as interpreters of information in a staggered manner.

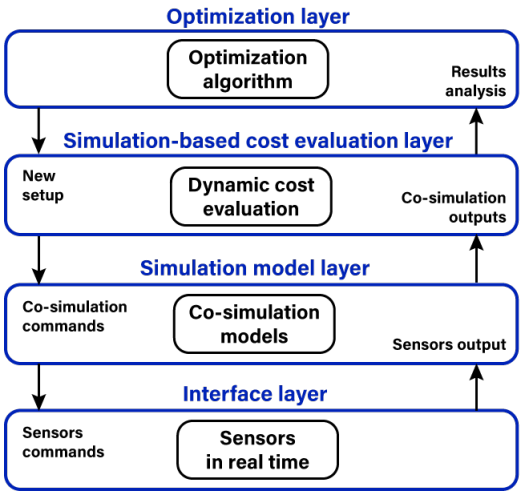

Figure 2. COSMO methodology, adapted from (Zhao and Ioannou, 2019).

These 2 frameworks show the importance of the bidirectional communication between the co-simulation and the optimization algorithm as the two main elements present in the architecture. The communication is generally concerned with providing co-simulation feedback to the optimization algorithm and then passing back instructions as a response to the feedback. The feedback part is present as co-simulation output data and the commands as new setups of the parameters or modifications to the initial state of the co-simulation.

2.2. Setup configuration in co-simulations

In simulation-based optimization, there are two elements always present that model the optimization process: the setup parameters and the outputs of the simulation model. In the case of co-simulation-based optimization, the composition of several simulations offers the opportunity of extending these elements in the process of optimization.

2.2.1. *Parameter variation*

The first element is the parameter variation which is used in the same way as simulation-based optimization. A general view of a co-simulation structure is shown in Figure 3 where a multi-model X has some sub-models A, B, C coupled between each other. A part of the optimization process is the variation of some parameters that are set at the beginning of the execution, these parameters can be a mix of one or several of the sub-models. This means that the co-simulation architecture must have the capability to determine a setup configuration changing one or several parameters located in one or several of the sub-models.

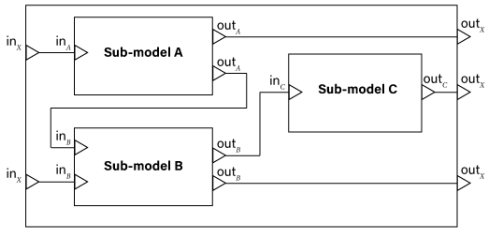

Figure 3. General co-simulation structure.

2.2.2. *Output values*

The second element is the outputs of the co-simulations that will be used as performance indicators to describe the objective function to be optimized. There are two possible scenarios for the use of outputs on optimization methods:

- The objective function of the optimization is the raw values of one of the outputs, in these cases, the optimization algorithm can be directly connected to the co-simulation. For example in the context of Figure 3, it could be executed as an optimization with an objective function defined as $min(f(x)) = out_C$.
- The objective function is a combination of several outputs, for this case, there is a necessity for an intermediate step to interpret the raw information and pass feedback suitable for the optimization algorithm. This was handled by the framework on mobility (COSMO methodology) by using intermediate elements that made the calculations necessary (Zhao and Ioannou, 2019).

2.3. Optimization methods in co-simulation

Another subject of interest for general architectures is the type of optimization algorithms used in co-simulation problems, in the literature, are many existing methods and we can find some specific examples, such as:

- Simulation-based Optimization (Zitney, 2009; Ahmed et al., 2010; Schirrer and Kozek, 2008) where some few cases are defined, simulated, and compared to determine the optimal.
- The stopping criterion method (Haodong et al., 2021) which is an iterated method that modifies the parameters until the objective value variation is imperceptible or lower than a previously define a threshold, this method requires performance feedback to the algorithm in order to determine the marginal difference between each simulation.
- Artificial intelligence search algorithms for exploration and optimization methods in co-simulations (Sun et al., 2021; Wang and Ma, 2018; Massat et al., 2014; Prabakar

and Li, 2015; Hamiti et al., 2011), such as genetic algorithms and particle swarm algorithms, among others.
- Exhaustive optimization (Ma et al., 2018; Mou and Shen, 2017; Lu et al., 2019) where all the possible scenarios are run and the optimal parameters are found. Usually, exhaustive optimization is avoided due to the computational cost of this concept, but in some cases with some constraints, is possible to do a complete exploration of the parameters. This method is referred to as Co-optimization in some articles.
- Gradient-based optimization uses constant feedback to adjust the parameters in the right direction (Deng et al., 2015; Tuli et al., 2021).

2.4. Discussion

Despite the availability of framework proposals, one of the main drawbacks of the applied examples is the absence of a multi-disciplinary architecture. The most challenging aspect of this proposal is the generality of the idea, nevertheless, there are some patterns in the literature that can be included.

As described in Section 2.1, the COSMO methodology proposes a general structure to couple complex systems with optimization algorithms in a formal structure including intermediate layers to handle the interpretation of information for different levels of abstraction.

The revision carried out in section 3.5 shows that there is no preferred or dominant optimization algorithm that can cover all the fields using co-simulation, this poses a requirement for any general architecture proposal. The framework needs to allow access to parameter setup and output feedback in order to support as many optimization algorithms as possible as well as an intermediate element that supports the interpretation of the output information for the algorithm.

The proposal of a general framework for combining co-simulation and optimization could unify the efforts of the research community toward the progress in the study and manipulation of complex systems. This is an opportunity to propose a framework that will help users quickly implement optimization in co-simulation problems.

3. Software framework for co-simulation based optimization

In this section, we propose a software framework that incorporates the patterns found in Section 2.4 as a general architecture. This framework can guide users to implement optimization into any co-simulation problem to be used for solving engineering problems through testing and validation of configurations of the complex system.

3.1. General architecture

In Figure 4 is presented the framework proposed to implement optimization in co-simulations. It is structured in a layer-based general disposition with downstream and upstream information flow. Each layer will be described in this section, as well as the kind of information that is past from module to module.

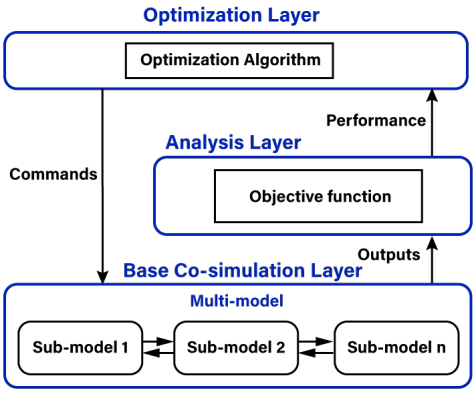

Figure 4. Software framework proposed.

3.1.1. *Base Co-simulation Layer*

The lower layer is composed of the initial co-simulation that has to be optimized. In this layer, the multi-model is defined as the representation of a complex system. Figure 5 shows a general structure of a co-simulation, which has as inputs the initial parameters setup and the output data of the multi-model. This layer contains the multi-model that describes the simulated part of the system and can contain Cyber-physical Sources (CPS).

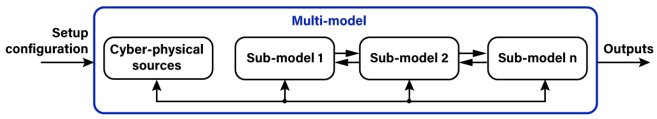

Figure 5. Base co-simulation layer.

3.1.2. *Analysis layer*

The middle layer will be the intermediary between the base-co-simulation layer (i.e. the raw output data of the co-simulation) and the high-level decisions over the model. As discussed in section 2.4, there is a necessity for an interpretation of the results of the co-simulation which is often represented as an objective function, but in general is an interpretation of the behavior of the model in a quantitative manner. In Figure 6, the outputs enter the analysis layer in order to produce an indicator of the co-simulation that will summarise the performance of the model.

3.1.3. *Optimization layer*

The optimization layer contains the optimization algorithm to be used, which depends on the necessities and characteristics of the problem. As mentioned in Section

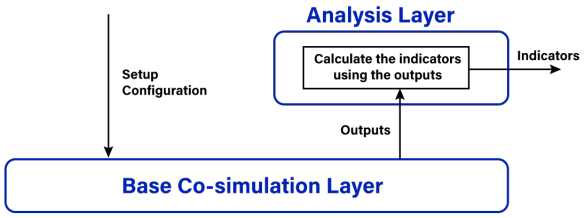

Figure 6. Analysis layer connected with base co-simulation.

3.5 there are many methods of optimization used in co-simulations, this demands an architecture that supports all kinds of algorithms, providing feedback channels and continuous performance reports interpreted by the Analysis layer.

As seen in Figure 7, this layer will provide the parameters setup of the base co-simulation depending on the optimization algorithm policies. Additionally, this layer receives the indicators of the co-simulations to make decisions and has the capacity to decide when the optimization process is finished, giving as a response the optimal value according to the optimization method.

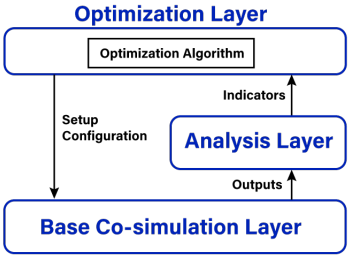

Figure 7. Optimization layer connected with the other layers.

3.2. Implementations with MECSYCO

In this section, the implementation of the architecture proposed previously is implemented in an existing co-simulation software to then propose some study cases. The objective is to use these study cases to test different configurations and possible changes that eventually could lead to improvement proposals, this will be done with optimization algorithms.

3.2.1. *Co-simulation software MECSYCO*

A software capable of implementing the proposed framework needs to have compatibility with the modular and bidirectional communication bases of the proposal. The software MECSYCO is presented due to its rigorous approach to co-simulation, using a formalization of the Agents and Artifacts paradigm in DEVS (Discrete Event System Specification) as a pivotal formalism (Camus et al., 2018). This conception allows the integration of heterogeneous formalisms within one global multi-model, this advantage can be extended to the multiple-layer structure where each layer can belong to a different domain but need to communicate constantly. Another principle that can be exploited is the idea of a DEVS wrapping for each element in order to easily handle the execution and communication of the coupled layers using the DEVS simulation protocol (Camus et al., 2018).

The software implementation is translated into DEVS-wrapped sub-models added to the co-simulation that will act as the Analysis and optimization layers shown in Figure 4. This means that using the already existing tools for the creation of sub-models in MECSYCO is possible to create each layer and establish the information flows required, as well as the execution logic that allows trying several scenarios until an optimal value is found. This software also uses DSL (Domain Specific Languages) to facilitate the programming of models for the user. The functional implementation of the framework is available through the public repositories available at http://www.mecsyco.com/.

3.3. First study case: house thermal control

In order to understand the way this framework will work, a study case is carried out where the idea is to use the tools of abstraction and definition present in the framework to structure and execute an optimization process. The first study case is a House's Air conditioning control problem, that is, a co-simulation that represents the interactions between a house, the outside temperature (weather), and an air conditioning system. This complex system is abstracted in a co-simulation that aims to optimize energy consumption and the occupants' comfort.

3.3.1. *Base Co-simulation*

As shown in Figure 8, the multi-model contains 4 interconnected systems which represent all the systems involved in the thermal regulation problem. Each system is described as follows:

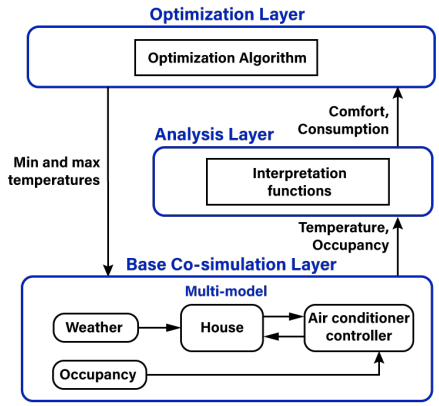

Figure 8. Study case 1: Framework application on house air conditioning co-simulation.

- The weather model is a CSV file generated using the

Open Weather Map API to collect data on the temperature weather of Nancy, France (data-set from December 13^{th} to 19^{th}, 2021), this is a Cyber-physical source as it is a direct link to the physical environment of the city.

- The house is a model that consists of 3 rooms, each room has different coefficients of outside temperature absorption and artificial heating power (controlled by an A/C system). This house model comes from the BuildSysPro /MODELICA library (state-of-the-art dedicated thermal library) (Plessis et al., 2014; Junghanns et al., 2021).
- The air conditioner controller is a Java-based model artifact that will take the decisions of turning on or off the acclimatization system based on the logic flow represented in Figure 9. The internal temperature of the house and the occupancy will trigger a new decision depending on the interval of accepted temperature (t_{min}, t_{max}). This decision will change the power the heater system uses in each room due to the energy required to heat up or cold down.
- The occupancy is a report of the occupancy of a house in Nancy, France (data-set from December 13^{th} to 19^{th}, 2021). This report actualizes the state of a house in terms of the absence of occupants (0) or the presence of one or more occupants (1).

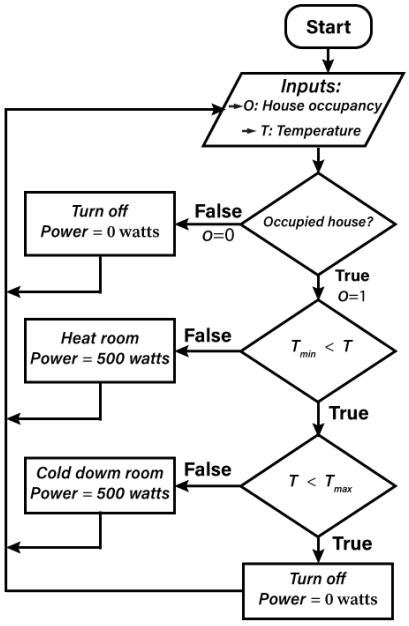

Figure 9. Air conditioner controller logic flowchart.

3.3.2. *Indicators*

The first indicator is energy consumption, this is directly related to the decisions of the air-conditioned system of turning ON and OFF as well as the power (in Watts) that is used in order to heat up or cool down the rooms. The power outputs will be accumulated to determine the overall energy use of the model, represented by the objective function $min\, f(x) = \sum_t |P_t|$, where P_t represents the power used in the timestamp t of the co-simulation.

The second indicator is the Comfort of the occupant, for the quantification of this thermal perception, the Thermal Comfort Standard ASHRAE 55 Comfort Zone (Olesen and Brager, 2004) will be used. This standard simplifies the many thermo-physical variables present in a thermal perception problem in order to determine a general goal temperature for the 2 common seasons that require internal temperature control in buildings. As the input weather data collected corresponds to the winter season of the year (December), the optimum temperature is 24.5°C with an acceptable range of 23-26°C. With this goal, Equation 1 is proposed as a second objective function. Where T_{Opt} is the optimal temperature and T_t is the temperature at moment t of the co-simulation. Thus maintaining the minimization premise for the optimization objective.

$$min \sum_t |T_{Opt} - T_t| \tag{1}$$

3.4. Second study case: electrical network

The second study case is a network of Houses connected to an energy supply and storage system, that is, a co-simulation that represents the interactions between 4 houses and a smart storage system, using as an intermediate a coupling operator to handle the internal communication. This co-simulation is the result of a doctoral thesis regarding energy exchanges in microgrids done by Wiart (2023).

3.4.1. *Base Co-simulation*

As shown in Figure 10, the multi-model contains 3 main systems interconnected and described as follows:

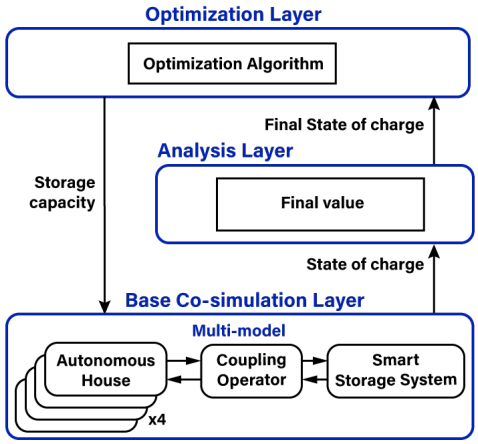

Figure 10. Study case 2: Framework application on microgrid co-simulation.

- Autonomous House: This component is by itself a complex system, as it is a multi-model involving a house energy consumption model, weather data, a smart energy generation source composed of a wind turbine, and photovoltaic cells.
- The Smart Storage System is in charge of helping with the energy supply process by storing the energy surplus and then supplying it in moments of high demand.
- Coupling operator: Handles the equilibrium seeking in terms of energy for the system using a bidirectional constant communication between all the components, in this case, the balance of energy between the autonomous houses and the energy storage system.

3.4.2. *Indicators*

The indicator is the final state of charge (SoC), as this model is conceived as a self-balanced co-simulation, there is no long-term objective function to optimize. Nevertheless, the matter of the optimal energy capacity of the storage system is relevant, as it would prevent oversized storage facilities. For the co-simulation scenario defined, the objective is to find the energy capacity that will have at the end of the co-simulation the minimum remaining charge in the storage system.

3.5. Optimization methods

The two optimization methods implemented in the MECSYCO framework are the basic simulation-based optimization method and the gradient descent optimization algorithm. These methods are described as follows.

3.5.1. *Simulation-based optimization*

Simulation-based optimization refers to the parametric optimization of an objective function, that is, the variation of some parameters to achieve an optimal value (Gosavi et al., 2015). In the case of the first study case, the definition of some parameter values is done by extending the interval of accepted temperature (T_{min}, T_{max}) and observing the optimization results. The 10 simulation scenarios chosen for this experiment are shown in Table 1.

Case	T_{min} [° Celsius]	T_{max} [° Celsius]
1	24.4	24.6
2	24.3	24.7
3	24.2	24.8
4	24.1	24.9
5	24.0	25.0
6	23.9	25.1
7	23.8	25.2
8	23.7	25.3
9	23.6	25.4
10	23.5	25.5

Table 1. Simulation scenarios for study case 1.

Regarding the second study case, the only parameter to be set is the maximum capacity of the battery to be charged starting from the default value of the co-simulation (280.000) and gradually reduced to approach the optimal value, the scenarios are displayed in Table 2.

Case	Charge capacity [W]	Case	Charge 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 2. Simulation scenarios for study case 2.

3.5.2. *Gradient descend*

The gradient descent optimization algorithm is a method to minimize an objective function by updating the parameters using the gradient of the objective function, which in this case depends on the results of the co-simulation (Ruder, 2017). The gradient descent algorithm allows to perform parametric optimization to more than 1 parameter, this makes the gradient updating process use partial derivatives in mathematical problems and partial executions in simulation problems. For the application in the study cases, only an initial value of the parameters and a learning rate are necessary to launch the optimization process, the values used for the study cases are shown in Table 3.

Study case	Parameter	Initial value	Learning rate
1	T_{min}, T_{max}	24.4, 24.6	0.3
2	Charge capacity	280000	0.001

Table 3. Gradient descend parameters.

4. Results and discussion

The results of the optimization process for each experiment done with MECSYCO to the study cases are presented and analyzed in this section. The source code for the study cases and optimization algorithms are available in the repository Vega (2023).

4.1. Optimization results

Study case	Optimization method	Results	Runtime [seconds]
1	Simulation-based	Figure 11	3.1025612
1	Gradient descend	Figure 12	19.9121719
2	Simulation-based	Figure 13	175.7024563
2	Gradient descend	Figure 14	217.3359421

Table 4. Study cases results.

Following the simulation scenarios defined in Section 3.5 the results are presented as shown in Table 4. In Figure 11 a Pareto graph compares the comfort with the consumption final values, the 10 cases show a non-linear behavior that gives a minimization result of T_{min} = 24.4° and T_{max} = 24.6° which is the case 1.

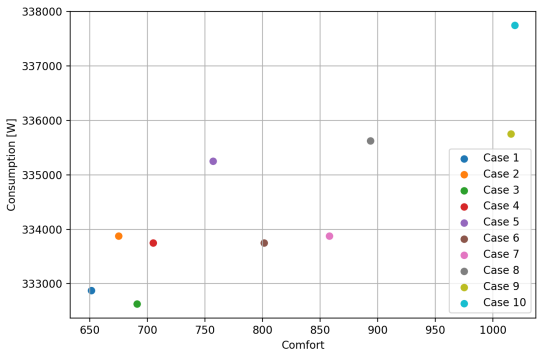

Figure 11. Pareto graph of the scenarios on study case 1.

The next results concern the same study case, but with the use of the Gradient descend algorithm, which in this case is a multi-parametric optimization, as 2 parameters are being variated at the same time a 3-dimension graph is presented in Figure 12 to show the different combinations of parameters (horizon axes) compare with the performance in terms of comfort (vertical axe). The results show that the optimal minimization result is T_{min} = 24.4° and T_{max} = 24.6°, this is the same result as the previous method but 13 iterations were executed.

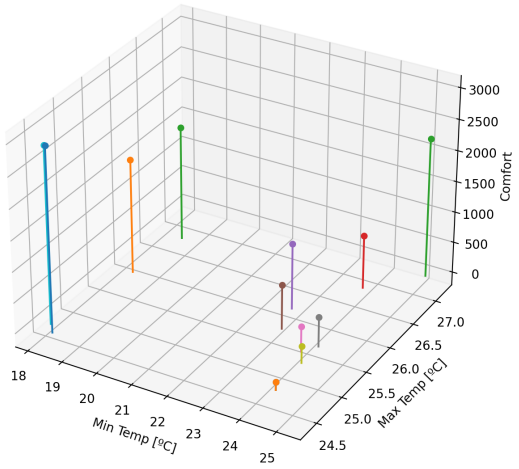

Figure 12. Comfort results of the gradient descent algorithm for study case 1.

The second study case results using the simulation-base optimization algorithm are shown in Figure 13 where can be seen that the optimal minimization value of charge capacity was achieved in several of the cases, as can be seen, any value under 220.000 will give a complete use of the battery storage, for this reason, 8 of the simulation results are redundant.

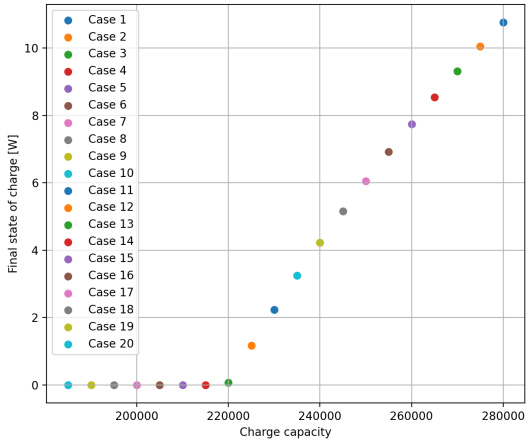

Figure 13. Simulation-based optimization results for study case 2.

For the second study case with the gradient descent algorithm, Figure 14 shows a similar approximation to the response as the previous method, but it is done at a faster rate, also after reaching the optimal point the algorithm recognized the convergence and stopped the process.

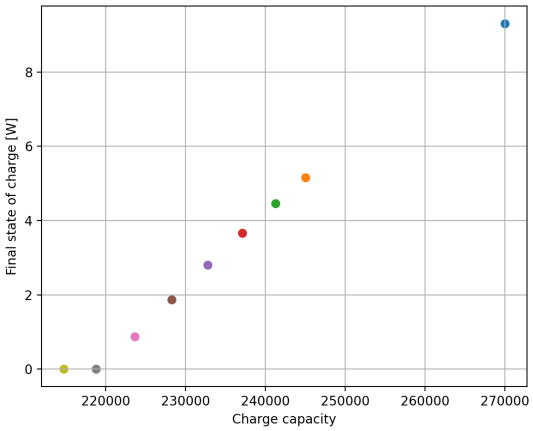

Figure 14. Gradient-based optimization results for study case 2.

4.2. Discussion

The results obtained by the several experiments done, show the usual advantages of simulation-based optimization, that is, to show the impact of the parameter variation over the performance of the system which allows us to take decisions to improve it. The experiments also show the advantages of using one general framework for several cases, such as the capacity of changing from one opti-

mization method to another with only a few parameters to set. Also, these optimization algorithms are using the same source code, which means that the layered approach allows the algorithms to be interchangeable with many optimization problems. In Figures 15 and 16, it is possible to see that using the same file dedicated to simulation-based optimization, it is only required to change 6 lines of code and remove 2 lines to alternate between study case applications. For the gradient descent configuration file, it is required the modification of 6 lines and the removal of 1 line as shown in Figures 17 and 18. On one hand, the transition between study cases depends only on setting the corresponding parameters, while on the other hand, the transition between optimization methods requires a different dedicated configuration file for each optimization method, a generalization of the optimization method parameters would simplify the process of optimization even more.

```
49        experimentPath = "Experiments/ComparisonHouses3.xml"; // Co-simulation file
50        List<String> nameParameters = new ArrayList<String>();
51        nameParameters.add("minTemp"); // First parameter name as defined in multi-model
52        nameParameters.add("maxTemp"); // Second parameter name, add more if necessary
53        List<String> nameIndicators = new ArrayList<String>();
54        nameIndicators.add("Comfort"); // First indicator name as defined in multi-model
55        nameIndicators.add("Consumption"); // Second indicator name, add more if necessary
56        String resultsPath = sourcePath + "/outputsMecsyco/results";  // Individual case reports
57        int runs = 10; // Number of iterations
58        double step = 0.1; // Parameters step variation for each iteration
```

Figure 15. Simulation-based configuration in study case 1.

```
64        experimentPath = "Experiments/ReseauAutonome_PoC_Opt.xml"; // Co-simulation file
65        List<String> nameParameters = new ArrayList<String>();
66        nameParameters.add("maxCapacity"); // First parameter name as defined in multi-model
67        List<String> nameIndicators = new ArrayList<String>();
68        nameIndicators.add("powerMode"); // First indicator name as defined in multi-model
69        String resultsPath = sourcePath + "/outputsMecsyco/results";  // Individual case reports
70        int runs = 10; // Number of iterations
71        double step = -50000; // Parameters step variation for each iteration
```

Figure 16. Simulation-based configuration in study case 2.

This general framework performance is expected to be of benefit to all co-simulation users in the optimization implementation process in any field of application. Particularly, the co-simulation methodology is relevant for multi-disciplinary problems, where the communication and global analysis of complex systems is the key to proposed improvements.

```
51        experimentPath = "Experiments/ComparisonHouses3.xml"; // Co-simulation file
52        List<String> nameParameters = new ArrayList<String>();
53        nameParameters.add("minTemp"); // First parameter name as defined in multi-model
54        nameParameters.add("maxTemp"); // Second parameter name, add more if necessary
55        String nameIndicator= "Comfort"; // // Indicator to minimize as defined in multi-model
56        String resultsPath = sourcePath + "/outputsMecsyco/results"; // Individual case reports
57        String resumeFilePath = sourcePath + "/outputsMecsyco/logResults.csv"; // General optimization report
58        double scale = 9000000.0; // Handle dimension differences between parameters and indicators
```

Figure 17. Gradient descent configuration in study case 1.

5. Conclusions

The contribution of this paper is a general architecture that will be able to contain any co-simulation problem to then connect an optimization process, this means that

```
51        experimentPath = "Experiments/ReseauAutonome_PoC_Opt.xml"; // Co-simulation file
52        List<String> nameParameters = new ArrayList<String>();
53        nameParameters.add("maxCapacity");  // Parameter name as defined in multi-model
54        String nameIndicator= "powerMode"; // Indicator to minimize as defined in multi-model
55        String resultsPath = sourcePath + "/outputsMecsyco/results";  // Individual case reports
56        String resumeFilePath = sourcePath + "/outputsMecsyco/logResults.csv"; // General optimization report
57        double scale = 9000000.0; // Handle dimension differences between parameters and indicators
```

Figure 18. Gradient descent configuration in study case 2.

the users will be able to exchange optimization methods to explore and compare the many existing algorithms in terms of performance and velocity. This architecture is implemented in MECSYCO.

In this work, several examples of co-simulation-based optimization were used to build the bases of the framework as well as 2 study cases for testing the capabilities of the proposal. For future work, we are working on more examples from different domains that will help validate the multi-disciplinary approach of the model as well as reveal more advantages of the framework, we plan to work on an Electric Car Co-simulation study case. Also, the implementation of more optimization algorithms could reveal more requirements for the optimization layer.

6. Funding

Research funded by the doctoral contract of Diego Alejandro Vega with the Université de Lorraine.

References

Ahmed, M., Yoo, Y.-H., and Kirchner, F. (2010). 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)*, pages 1–6. VDE.

Bharati, G. R., Chakraborty, S., and Darrah, J. (2021). 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)*, pages 1–5. IEEE.

Boyd, S., Boyd, S. P., and Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press.

Camus, B., Paris, T., Vaubourg, J., Presse, Y., Bourjot, C., Ciarletta, L., and Chevrier, V. (2018). Co-simulation of cyber-physical systems using a devs wrapping strategy in the mecsyco middleware. *Simulation*, 94(12):1099–1127.

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

Deng, W., Mao, B., Liang, B., and Song, P. (2015). Co-simulation of stabilization accuracy optimization of overhead weapon station. In *2015 International conference on Applied Science and Engineering Innovation*, pages 972–977. Atlantis Press.

Fu, M. C. (2001). Simulation optimization. In *Proceed-*

ing of the 2001 Winter Simulation Conference (Cat. No. 01CH37304), volume 1, pages 53–61. IEEE.

Gomes, C., Thule, C., Broman, D., Larsen, P. G., and Vangheluwe, H. (2017). Co-simulation: State of the art. *arXiv preprint arXiv:1702.00686*.

Gosavi, A. et al. (2015). *Simulation-based optimization*. Springer.

Graja, I., Kallel, S., Guermouche, N., Cheikhrouhou, S., and Hadj Kacem, A. (2020). A comprehensive survey on modeling of cyber-physical systems. *Concurrency and Computation: Practice and Experience*, 32(15):e4850.

Hamiti, T., Gerada, C., and Rottach, M. (2011). 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*, pages 1536–1540. IEEE.

Haodong, L., Sujun, D., Hongsheng, J., and Yunhua, L. (2021). A cooling capacity distribution method for liquid cooling cycle based on co-simulation and optimization of amesim and modefrontier. pages 317–322. cited By 0.

Junghanns, A., Gomes, C., Schulze, C., Schuch, K., Pierre, R., Blaesken, M., Zacharias, I., Pillekeit, A., Wernersson, K., Sommer, T., et al. (2021). The functional mock-up interface 3.0-new features enabling new applications. In *Modelica Conferences*, pages 17–26.

Law, A. M. and Kelton, W. D. (2000). *Simulation modeling and analysis*, volume 3. Mcgraw-hill New York.

Law, A. M. and McComas, M. G. (2002). Simulation-based optimization. In *Proceedings of the Winter Simulation Conference*, volume 1, pages 41–44. IEEE.

Lee, E. A. (2008). Cyber physical systems: Design challenges. In *2008 11th IEEE international symposium on object and component-oriented real-time distributed computing (ISORC)*, pages 363–369. IEEE.

Lu, H., Zhao, H., Huang, A., Kim, D., Sun, J., Hu, J., and Kim, H. (2019). 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)*, pages 1–3. IEEE.

Ma, Y., Li, B., Wu, Z., Wu, H., and Chen, Z. (2018). A co-simulation method of power amplifier for reliability optimization. In *2018 IEEE International Conference on Electron Devices and Solid State Circuits (EDSSC)*, pages 1–2. IEEE.

Massat, J.-P., Laurent, C., Bianchi, J.-P., and Balmès, E. (2014). Pantograph catenary dynamic optimisation based on advanced multibody and finite element co-simulation tools. *Vehicle System Dynamics*, 52(sup1):338–354.

Mou, J. and Shen, Z. (2017). Co-simulation and co-optimization strategy for active absorber with periodic structure. In *2017 IEEE Electrical Design of Advanced Packaging and Systems Symposium (EDAPS)*, pages 1–4. IEEE.

Olesen, B. W. and Brager, G. S. (2004). A better way to predict comfort: The new ashrae standard 55-2004.

Pasetti, A. (2002). *Software frameworks and embedded control systems*. Number 2231 in Lecture notes in computer science. Springer. OCLC: ocm49036010.

Plà-Aragonés, L. M. (2015). *Handbook of operations research in agriculture and the agri-food industry*, volume 224. Springer.

Plessis, G., Kaemmerlen, A., and Lindsay, A. (2014). Buildsyspro: a modelica library for modelling buildings and energy systems. In *Proceedings of the 10 th International Modelica Conference; March 10-12; 2014; Lund; Sweden*, number 096, pages 1161–1169. Linköping University Electronic Press.

Prabakar, K. and Li, F. (2015). Proportional integral controller gain tuning using real time digital simulation models and multi-objective optimization based co-simulation. *IFAC-PapersOnLine*, 48(30):473–478.

Ramat, E. (2006). Introduction à la simulation: principaux concepts. *Modélisation et Simulation Multi-Agent: application pour les Sciences de l'Homme et de la Société*, pages 37–60.

Rüdenauer, A., Han, S., Geimer, M., et al. (2012). Optimization of the development process of mobile machines using a standardized co-simulation. *Landtechnik*, 67(2):122–126.

Ruder, S. (2017). An overview of gradient descent optimization algorithms.

Schirrer, A. and Kozek, M. (2008). Co-simulation as effective method for flexible structure vibration control design validation and optimization. pages 481–486.

Sun, L., Zhang, R., Du, C., Rong, W., Li, X., Chen, Y., Fu, T., Cao, S., and Shi, D. (2021). Optical optimization of ultra-thin crystalline silicon solar cells by a co-simulation approach of fem and ga. *Applied Physics A*, 127(7):1–9.

Tuli, S., Poojara, S. R., Srirama, S. N., Casale, G., and Jennings, N. R. (2021). Cosco: Container orchestration using co-simulation and gradient based optimization for fog computing environments. *IEEE Transactions on Parallel and Distributed Systems*, 33(1):101–116.

Vega, D. A. (2023). MECSYCO study cases repository. https://gitlab.inria.fr/Simbiot/mecsyco/mecsycoscholar/-/tree/co-sim-opt-study-cases/. [Online; accessed 12-May-2023].

Wang, X. and Ma, H. (2018). Unidirectional coupler optimization of surface plasmon polaritons based on co-simulation of genetic algorithm and comsol. *Optik*, 156:408–415.

Wiart, J.-B. (2023). Approche hierarchique de co-simulation pour l'etude des echanges d'energie des micro-reseaux.

Zhao, Y. and Ioannou, P. (2019). A co-simulation, optimization, control approach for traffic light control with truck priority. *Annual Reviews in Control*, 48:283–291. cited By 3.

Zitney, S. E. (2009). Advanced co-simulation for computer-aided process design and optimization of fossil energy systems with carbon capture. In *Design for Energy and the Environment*, pages 211–228. CRC Press.