PROCEEDINGS OF ECOS 2023 - THE 36TH INTERNATIONAL CONFERENCE ON
EFFICIENCY, COST, OPTIMIZATION, SIMULATION AND ENVIRONMENTAL IMPACT OF ENERGY SYSTEMS
25-30 JUNE, 2023, LAS PALMAS DE GRAN CANARIA, SPAIN

Improving Computation Time for Optimization Runs of Modelica-based Energy Systems

Sven Klute[a], Markus Hadam[b], Mathias van Beek[c] and Marcus Budt[d]

[a] Fraunhofer Institute for Environmental, Safety, and Energy Technology UMSICHT, 46047 Oberhausen, Germany, sven.klute@umsicht.fraunhofer.de, CA
[b] Fraunhofer Institute for Environmental, Safety, and Energy Technology UMSICHT, 46047 Oberhausen, Germany, markus.hadam@umsicht.fraunhofer.de
[c] Fraunhofer Institute for Environmental, Safety, and Energy Technology UMSICHT, 46047 Oberhausen, Germany, mathias.van.Beek@umsicht.fraunhofer.de
[d] Fraunhofer Institute for Environmental, Safety, and Energy Technology UMSICHT, 46047 Oberhausen, Germany, marcus.budt@umsicht.fraunhofer.de

Abstract:
Mathematical optimization is a widespread method in order to improve, for instance, the efficiency of energy systems. A simulation approach based on partial differential equations can typically not be formulated as an optimization problem, thus requiring interfacing to an external optimization environment. This is, amongst others, also true for the programming language Modelica. Because of high computation time, such coupled approaches are often limited to small scale optimization problems. Since simulation models tend to get more complex, simulation time and, in turn, associated optimization time rise significantly. To enable proper sampling of the search space, individual optimization runs need to be solved in acceptable times. This paper addresses the search for a proper optimization approach and tool to couple with Modelica/Dymola. The optimization is carried out on an exemplary power plant model from the ClaRa-Library using an evolutionary algorithm (SPEA2-based) with Ansys optiSlang. To verify and evaluate the results, a comparison with the standard Dymola optimization library is performed. Both parallelization and indirect optimization with surrogate models achieved a significant runtime reduction by a factor of up to 5.4. The use of meta models is particularly advantageous for repetitive optimization runs of the same optimization problem but may lead to deviations due to the calculated approximations.

Keywords:
Mathematical Optimization, Runtime Reduction, Parallelization Methods, Energy Systems, Meta-model, Modelica, Dymola, Ansys optiSLang

1. Introduction

Process modeling and simulation is becoming increasingly important to achieve improvements in today´s complex energy systems and technologies [1]. The model formulation can be carried out via several programming languages. A frequently used language is the object- and equation-oriented programming language Modelica, which is particularly suitable for the representation of dynamic processes [2, 3]. To improve the handling of the modeling process, usually simulation environments such as Dymola are used [4]. Based on these models, extensive investigations such as design or control optimizations can be performed. A wide used method is the parameter variation, which is quite simple to perform, but becomes more time consuming as the complexity of the model increases. In these cases, mathematic optimization is used instead. Therefore, an optimization problem has to be defined and solved with the help of algorithms. The programming language Modelica, which is designed for simulation tasks, does not allow the formulation of such optimization problems. Thus, a coupling with external optimization languages or optimization environments is necessary [2]. In the field of energy systems, a wide variety of such environments have already been used for coupling with Dymola (see Section 3). This paper presents the first-time coupling of Dymola and Ansys optiSLang [5] in order to solve static optimization problems. For this purpose, the implemented workflow and its main features as well as the used algorithm are presented in Section 4 and 5. The verification of the results is finally done by a comparison with the standard Dymola optimization library (DLR Optimization Library [6]). This library offers a wide range of different optimization techniques that are easy to apply. However, a huge disadvantage is the

https://doi.org/10.52202/069564-0116

required runtime for optimization which will be further discussed in Section 6. For a better understanding of the results, relevant basics of mathematical optimization are presented first.

2. Mathematical optimization

The first step when dealing with optimization is the definition of the problem itself. Therefore, the objective function f(x), the optimization variable x and possible restrictions must be defined [7, 8]. For this study, we split optimization into the static and transient type, depending on the contents of the optimization variable. A typical application for static optimization is design optimization, where an optimal design of a component is determined disregarding transient features [9, 10]. Transient optimization is, for instance, used to calculate optimal process control strategies [11].

When choosing the appropriate algorithm, the specific properties of the algorithm as well as the specifications of the optimization problem must be considered [9, 12]. In application-related problems, multi-objective optimization tasks are the most common. In contrast to the single-objective optimization, several objective functions are optimized simultaneously. Usually, these functions are in conflict with each other. As a result, there is no solution where both objective functions reach their individual optimum [9, 13–15]. This context can be described by the so-called Pareto-Front (see Figure 1). Illustrated are the two fictitious objective functions $f_1(x)$ and $f_2(x)$, which are to be minimized by a variation of x. By using the single optima (x_1 and x_2), three characteristic ranges can be identified. While an increase of x in the range $x < x_1$ leads to an improvement of both functions, an increase for $x > x_2$ causes a deterioration of both functions. The relevant range is thus located between the two individual optima ($x_1 < x < x_2$). Within this range, an improvement of one objective function can only be realized by a deterioration of the other criterion. This means, that there is no longer a single optimum as in the case of the single-objective optimization. Instead, there is a set of different optima, also called the Pareto-Optimal set or Pareto-Solutions. By comparing the two objective functions, this set can be represented as a subset of the boundary curve (Pareto-Front on the right). By selecting a weighting factor, a problem-specific overall solution can then be determined. [9, 13–15]

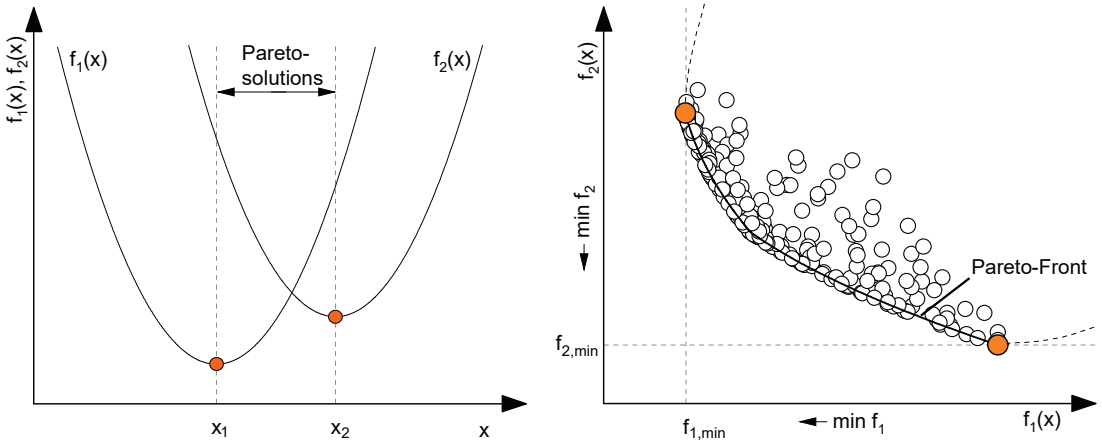

Figure. 1. Target function values of the fictitious functions $f_1(x)$ and $f_2(x)$ (left) and Pareto-Front of the fictitious functions (right) according to [9]

3. Optimization with Dymola

In the area of energy systems, various couplings between Dymola and different optimization environments have already been implemented. The choice of the appropriate environment is problem-specific and depends primarily on the type of optimization problem. Table 1 shows an overview of published investigations and lists the used optimization-tool. The DLR Optimization Library is fully integrated in Dymola and can be used to define multicriteria and trajectory optimization problems. In the three listed examples [16–18] different types of Models and optimization problems were investigated. Dymola is one of currently over 170 tools which support the FMI standard. It allows the export and exchange of simulation models into other simulation or optimization environments. The studies [19–21] are using this standard for an optimization in Matlab and Simulink with the focus on optimal control. In the study [22] Matlab is used as an optimization platform. Instead of FMI, an optimization workflow is presented. The workflow is using a Dymola script to run the simulation with different input parameters. The results are imported into Matlab from result files (dsres.mat) using pre-implemented functions. The study [23] is using the optimization environment OMOptim and Dymola as a simulation tool. To excess the model structure and adapt the input variables, OpenModelica is used. In the shown example, a

https://doi.org/10.52202/069564-0116

design optimization of a heat pump is presented. The optimization program GenOpt is a text-based system which uses the input and output files of a simulation software to perform the optimization. The studies [24–26] are presenting an optimal control problem and [27] are solving a design optimization with GenOpt. Pfeiffer [28] utilize the FISEMO environment to perform a numerical sensitivity analysis of Modelica models. Dymola is used for the modeling process and the export of the model into C-Code. The integration into the FISEMO environment is done by using the DASPK3.1 package.

Table 1. Couplings of Dymola and optimization environments in the field of energy systems.

Model	Method	Optimization-tool	Ref.
Chilled water plant	Real-Time Optimization	DLR Optimization Library	[16]
CO_2 heat pump and thermal storage	Optimal Control	DLR Optimization Library	[17]
Osmosis hybrid system	Design Optimization	DLR Optimization Library	[18]
Hybrid Energy systems (HES)	Optimal Control	Matlab	[19]
Decentralised energy supply	Optimal Control	Simulink	[20]
Thermal management system	Optimal Control	MUSCOD-II	[21]
Chiller plant	Optimal Control	Matlab	[22]
Heat pump	Design Optimization	OMOptim	[23]
Combined heat and power plant (CHP)	Optimal Control	GenOpt	[24]
Central cooling plant	Optimal Control	GenOpt	[26]
Chiller plant	Optimal Control	GenOpt	[25]
Chilled water system	System configuration	GenOpt	[27]
Discontinuously Models	Sensitivity analysis	Fisemo	[28]

Most of the listed publications deal with transient optimization and focus on optimal control problems. The static optimization is less frequent and only addressed in two publications with small scale problems. In order to test the optimization framework optiSLang and contribute to the landscape of investigations of static optimization-simulation coupling, a proof of concept with a model of a coal-fired power plant is set up and carried out in this paper. One of the main advantages of optiSLang is the support of high-order parameter spaces, and thus, extensive sensitivity analysis as well as comparatively complex design optimization calculations. In the field of mechanical construction, for instance, optimization problems with up to 30,000 independent variables were solved [29]. Since there is currently no direct connection to Dymola, a workflow will be presented in the first step (Section 4). Solving dynamic optimization problems with optiSLang is possible as well but will not be further discussed within this paper.

4. Coupling of Dymola and Ansys optiSLang

Since there is no direct coupling option for Dymola and optiSLang yet, an appropriate workflow had to be developed. The implemented workflow presented in this paper enables an automatic data exchange between Dymola and optiSLang (see Figure 2). optiSLang works according to the principle of variance calculation, which is why several model simulations and modifications of the model have to be performed [5]. Therefore, a text-based solver chain is implemented within the optiSLang environment.

Dymola is used for the modeling process of the respective system in the programming language Modelica. The model can then be translated into C-Code and saved as an executable file with an associated input file. The input file contains essential initial values, model parameters and simulation settings which are needed for the simulation run of the model. After a successful simulation, the results are saved into output files. By modifying the input file, different model parametrizations and therefore results can be generated.

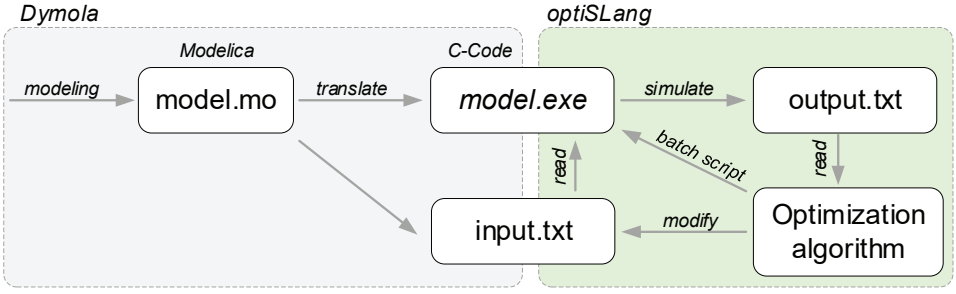

Figure. 2. Basic principle of the workflow for coupling Dymola and optiSLang

optiSlang is using these three files to perform the optimization. As a first step, the optimization problem must be defined in the optiSLang environment. In contrast to the original mathematical optimization, it is not necessary to develop a complex mathematical objective function. Instead, the optimization is performed using a black box model of the Dymola model (executable file). The input variables are set as optimization variables while the output variables are set as target criteria. The optimization variables (tuners) must be extracted from the input file. For each evaluation of the algorithm, one modified input file is created. Each of these files has different tuning values which are varied in a defined range in order to achieve the optimum. The validation of the optimum is done by using target criteria which are integrated in the model and calculated through a simulation (via batch-script). The results of each simulation are saved into individual output files. optiSLang extracts these values and evaluates them according to the chosen algorithm.

Although the black box model simplifies the definition of the optimization problem, it also means that there is no knowledge of the mathematical structure of the problem. However, this knowledge is usually necessary for the choice of a suitable algorithm. To counteract this problem, optiSLang offers the user a traffic light system which recommends an algorithm based on the determined internal optimization problem (number of inputs, single- or multi-objective problem etc.). It must be taken into account, that this recommendation depends on the mathematical solvability of the problem and does not guarantee the best possible result. Thus, it is still necessary to know about the functionality of the individual algorithms to select a suitable one for a given optimization problem.

5. Case study: coal-fired power plant

In order to validate the implemented workflow, an optimization run is performed using an example model of the open source library ClaRa [30]. The chosen model represents a coal-fired power plant with eight turbine stages including feedwater preheating. Fig. 3 shows a simplified schematic of the plant. After passing the boiler, the generated steam flows through the individual turbine stages until it is expanded to the condenser pressure. The collected condensate is then preheated by the four low-pressure preheaters (LP-PH) before it enters the steam-heated feedwatertank. The feedwater is preheated by the high-pressure preheater (HP-PH) before it enters the boiler. The power plant is designed to provide 580 MW_{el}. In the simulation scenario two load reductions to 70 percent load are performed. One simulation run of 10.000 seconds requires around 118 seconds simulation time.

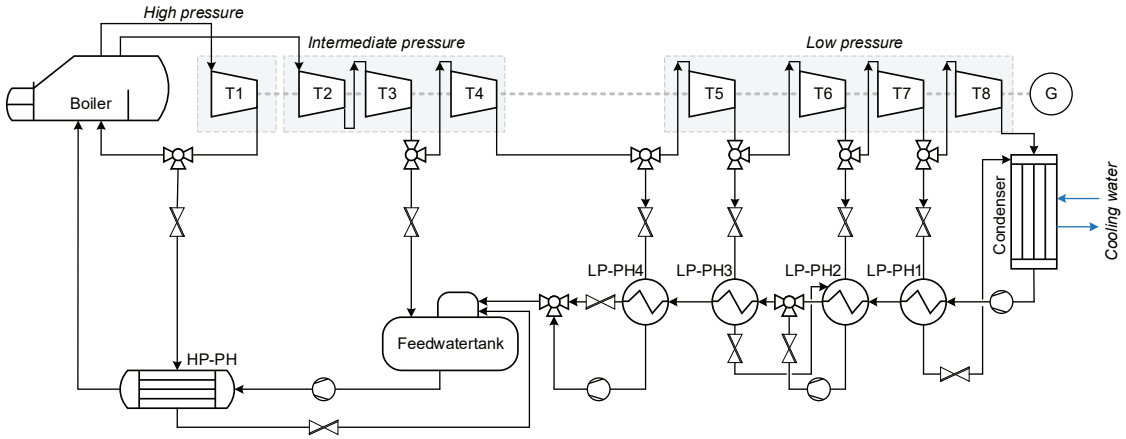

Figure. 3. *Simplified schematic of the coal fired power plant model SteamPowerPlant_01 from the open-source library ClaRa*

The optimization is carried out using the HP preheater as an example component due to its huge impact on the preheating section. The HP-PH model is a shell-and-tube heat exchanger for which selected tube parameters are to be optimized. Further information about the model can be obtained from the open source library ClaRa [30].

Table 2. Specifications of the optimization variables d_i, d_o and l.

Tuner	lower boundary [m]	initial value [m]	upper boundary [m]
d_i	0.015	0.020	0.024
d_o	0.025	0.028	0.031
l	9	10	11

https://doi.org/10.52202/069564-0116

The aim of the optimization is the improvement of the overall power plant efficiency with respect to the investment costs of the heat exchanger. Therefore, inner tube diameter d_i, outer tube diameter d_o and tube length l are to be optimized (Eq. (1)). The resulting multi-objective optimization problem is shown below. In many cases, optimization tools simplify a multi-objective problem to a single-objective problem by combining the multiple functions into a single sum-function with individual weights or by focusing on one function and introducing the remaining as constraints [31]. However, optiSLang is using an interactive method (see Section 2) instead, which allows to deal with several functions simultaneously. The resulting Pareto-Front can be used to identify the area of interest where focused optimization runs can be performed.

A simple degression function is implemented for estimating the investment costs (see Eq. (2)) [32, 33]. Therefore, the heat transfer surface is set in relation to a theoretical reference heat exchanger (A_{ref} = 1,500 m² and $K_{i,ref}$ = 800,000 €) and is simplified by Eq. (3) The degression exponent n was set to 0.6 [32, 33]. The price index effect from PI and PI_{ref} is neglected in this example. The efficiency is automatically calculated within in the model and is equal to the outlet-power and inlet-power ratio (see Eq. (4)). The negative equation sign is used to transform the maximization problem into a minimization problem. Table 2 shows the individual limits of the tuner variables. The initial values of the overall system efficiency η_{ref} and the investment costs $K_{i,ref}$ are 51,58 % and 802,545 €.

$$x = (d_o, d_i, l)^T \tag{1}$$

$$\min f_1(x) = K_{i,ref} \cdot \left(\frac{A(x)}{A_{ref}}\right)^n \cdot \left(\frac{PI}{PI_{ref}}\right) \tag{2}$$

$$A(x) = 2\pi \cdot \frac{d_o + d_i}{2} \cdot l \cdot n_T \tag{3}$$

$$\min f_2 = -P_{out}/P_{in} \tag{4}$$

5.1. Optimization algorithm

As solving algorithm a modified implementation of the Strength Pareto Evolutionary Algorithm 2 (SPEA2 [34]) is used. Figure 4 shows the general flowchart of the algorithm and indicates the interaction with the implemented workflow from Section 4.

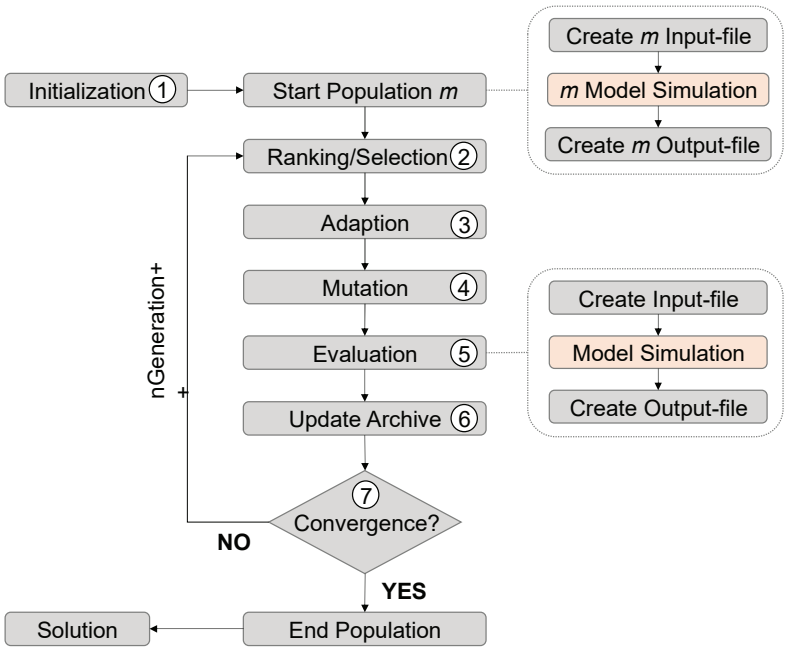

Figure. 4. General flowchart of the evolutionary algorithm (according to [14] and [35]) and the interaction with the implemented workflow

After initialization of the simulation model, a start population of the size *m* is generated (Step 1). Therefore, *m* modified input-files are created and *m* respective simulation runs are performed. A dominance-based ranking system is used to assign a corresponding fitness factor to each solution, which are extracted from the *m* output-files (Step 2). The iteration loop starts after the assignment process is done by using adaption (e.g. crossover or swarm movement) and mutation methods. The additional designs must be evaluated and ranked once more (Step 5). Therefore, new model simulations are required. Beneficial solutions are used to update the archive (containment for good solutions) and checked for convergence (Step 7). If the convergence criteria is reached, the optimization terminates and the end population is stated. In the negative case, step 2 to 7 are repeated until convergence or a termination criterion are reached. Since each evaluation requires one simulation of the model, optimization runtime is highly affected by simulation runtime and the number of total simulations. For a time-efficient optimization, both should be reduced as far as possible.

The described steps can be modified further by changing the algorithm parameter (e.g. mutation-rate or crossover method). This allows the user to adapt the algorithm to the specific problem. The optimization runs presented in this paper use the optiSLang standard settings. All calculations are carried out under Windows 10 on an Intel Core i5-6500 (3.2 GHz processor) with 8 GB RAM.

6. Results

6.1. Direct and indirect optimization

The optimization can be done as a direct or indirect optimization (Figure 5). In case of the direct optimization, an individual model simulation is carried out for each evaluation of the algorithm. The indirect optimization, on the other hand, uses a surrogate model (meta-model) of the simulation model. The meta-model is determined based on n model simulations (100 by default) via a sensitivity analysis (SA). The SA provides several response surfaces which mathematically describe the correlation between the target criteria and the chosen optimization variables. Both variants have individual advantages and disadvantages which are described in more detail below.

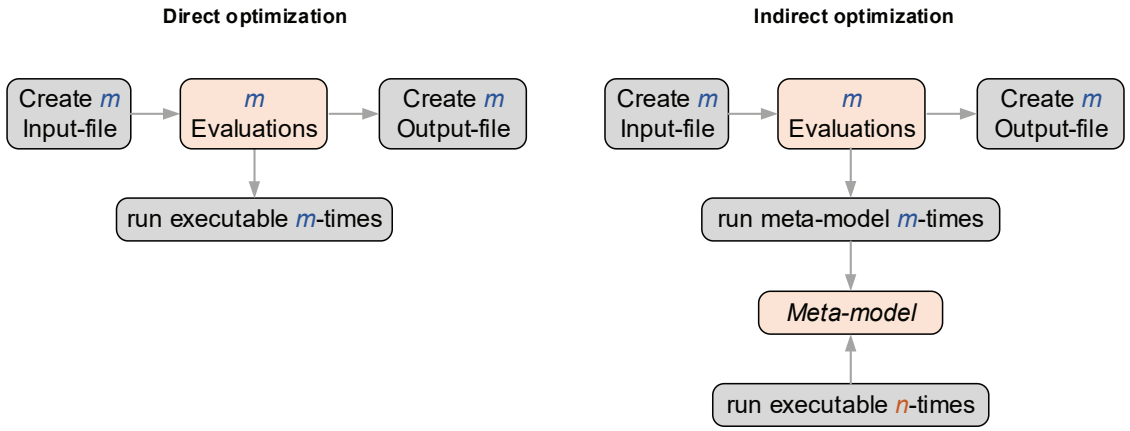

Figure. 5. Direct and indirect optimization using the executable file and surrogate models

Figure 6 shows an example response surface of the overall system efficiency and its correlation to the tube length and the inner tube diameter of the HP-PH. For the approximation a combination of several approximation techniques such as Kriging or neuronal networks are used. In this case the color scale indicates the local forecast quality of the approximation. Therefore, the so-called Coefficient of Prognosis (COP) is used. This value is a validation criterion developed by optiSLang to describe the quality of the approximation. It is calculated by using a cross-validation procedure [5]. In this case the COP is on a very high level across the entire surface. In systems with more complex and discontinuous process behavior generally lower COP values are reached. In such cases, it is possible to use other Design of Experiment (DoE) methods and/or to increase the number of samples (*n*). Another feature of the SA is the filtering of non-relevant parameters to simplify the optimization problem.

https://doi.org/10.52202/069564-0116

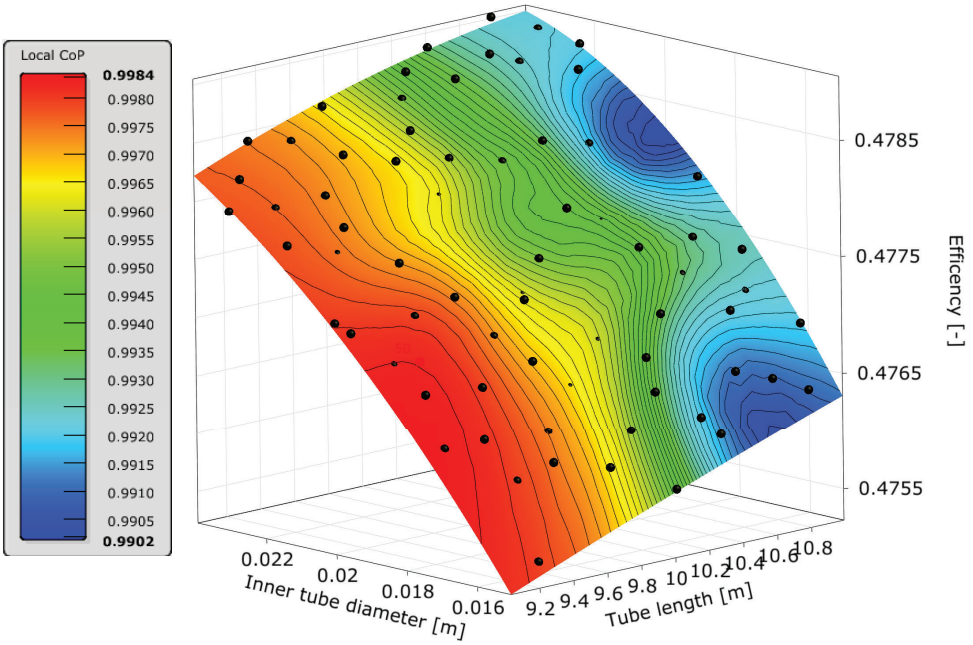

Figure. 6. Change of overall system efficiency depending on the tube length and the inner tube diameter of the HP preheater. Indication of the local COP by means of color scale

6.2. Runtime comparison of direct and indirect optimization

A comparison of the direct and indirect optimization method indicates a time advantage in favor of the indirect method. Both calculations are carried out with the same number of model simulations and algorithm evaluations. For the comparison, the overall runtime (t_{dir} and t_{indir}) is further separated into different categories (see Eq. (5) and Eq. (6)). Hereby a more detailed understanding of the optimization processes is achieved. The different categories are defined as the simulation time, the approximation time and the optimization time. The simulation time t_{sim} equals the total time for model simulation and can be calculated from the runtime for one simulation t_{model} and the total number of model simulations n_s (see Eq. (7)). The approximation time t_{approx} of the indirect variant corresponds to the time period for generating the surrogate model and is affected by the number of simulation and the complexity of the correlations. The remaining time is assigned to the optimization time. Both, the approximation time and the total runtime can be taken from the optiSLang log file.

$$t_{dir} = t_{sim} + t_{opt,dir} \tag{5}$$

$$t_{indir} = t_{sim} + t_{opt,indir} + t_{approx} \tag{6}$$

$$t_{sim} = t_{model} \cdot n_s \tag{7}$$

The resulting runtimes of the direct and indirect method are shown in Figure 7. The displayed percentage values describe the share of the respective category in the total runtime of the method. Due to the fixed number of model simulations, the direct and indirect optimization require the same simulation time. Although the indirect variant requires additional time due to the creation of the meta-model, the optimization can be done significantly faster because no further time-consuming simulation runs are needed. Compared to the direct method, the optimization time can be reduced by 97 % and the total runtime by 12 %. With increasing number of evaluations this time advantage increases even more. In addition, simulation and approximation are omitted in a new optimization run of the same problem since the determined approximation is reusable.

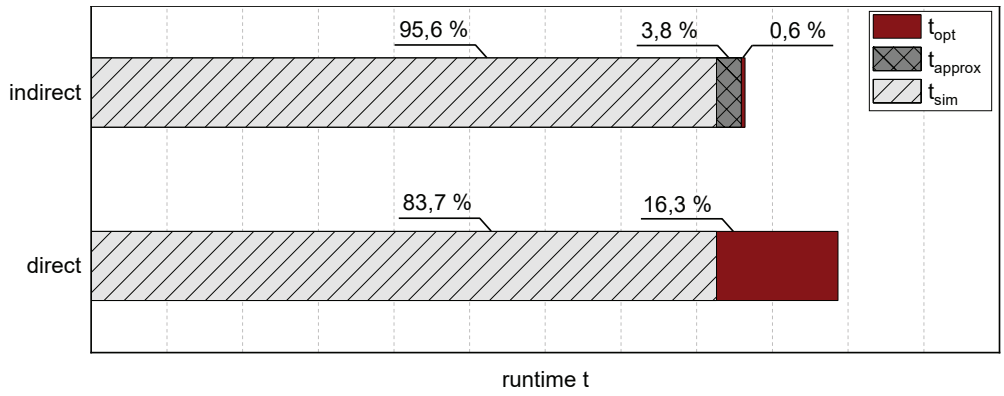

Figure. 7. Overall runtime of the direct and indirect optimization and the respective proportions of the simulation, approximation and optimization time

When using approximations, it must be taken into account that the results are affected by an error seen in Figure 8 which compares the Pareto-Front of the two methods. While the indirect Pareto-Front generates a very smooth graph, the direct method shows a much more complex behavior. The use of the approximation simplifies the system response, which in this example leads to areas where the investment costs are underestimated or overestimated. Because of these discrepancies, the indirect method should be used primarily as a first optimization step in order to reduce the overall computing time and to locate/constrain the relevant solution space [5].

With increasing model runtime, shortening methods are becoming more important. If no pre-optimization is desired, parallelization methods can be used alternatively. Since the simulations of each evaluation loop are executed parallel instead of sequentially, runtime reductions can be achieved without affecting the result. In the present work the runtime of the direct method could be reduced by up to 75 % using parallelization. A further reduction can be realized by using a computing cluster which grants access to more cores.

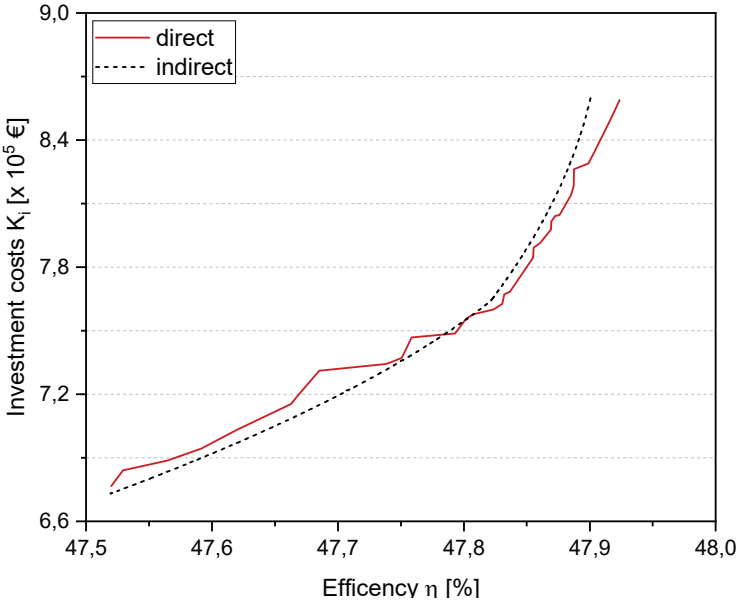

Figure. 8. Comparison of the direct and indirect Pareto-Front of the overall system efficiency and the investment costs of the HP preheater

6.3. Comparison of the DLR Optimization Library and optiSLang

In the following section, a comparison of the optiSLang workflow and the DLR Optimization Library is presented. Since the DLR Optimization Library uses weighting factors to transform multi-objective problems into single-objective problems, no comparison to the previous results could be made. Instead, a single-

https://doi.org/10.52202/069564-0116

objective optimization using the overall efficiency as the optimization goal is carried out. An evolutionary algorithm with standard specifications is used for the calculation and parallelization methods are considered.

All setups manage to improve the overall system efficiency (see Table 3). The optimized efficiency η_{opt} as well as the difference to the initial value $\Delta\eta$ are shown in the second column of the table and only differ slightly. For the further comparisons of the setups, the direct DLR setup is defined as reference. With regard to the optimized efficiency only minimal changes occur. On the other hand, large differences in the runtime were found. The direct optimization of the DLR Optimization Library is the most time-consuming setup. However, by using parallelization, the runtime can be more than halved. The direct optiSLang setup reduces the runtime by almost 40 %. By using additional parallelization, the runtime can be further reduced by a factor of four without influencing the results. The indirect optiSLang setup is faster than the direct DLR and direct optiSLang setup but slightly influences the optimized results negatively. A benefit of this setup is the generated surrogate model, which can be reused for further optimization runs.

Table 3. Overall runtime and maximum efficiency of the investigated setups for the single-objective optimization of the HP Preheater.

Method	$\boldsymbol{\eta_{opt}}$ $(\boldsymbol{\Delta\eta})$ [%]	$\boldsymbol{\eta_{DLR,direct} - \eta_{opt}}$ [%]	$\boldsymbol{t}$ [h]	$\frac{t}{t_{DLR,direct}}$ [-]
DLR Optimization Library direct	51.688 (+ 0.105)	0	11.4	1
DLR Optimization Library direct and parallel	51.688 (+ 0.105)	0	5.3	0.46
optiSLang direct	51.689 (+ 0.106)	0.001	8.2	0.72
optiSLang direct and parallel	51.689 (+ 0.106)	0.001	2.1	0.18
optiSLang indirect	51.686 (+ 0.103)	- 0.002	7.2	0.63

7. Conclusion

Since the programming language Modelica does not allow the definition of optimization problems, it is necessary to link up with external optimization environments or optimization languages. The simulation environment Dymola offers the integrated DLR Optimization Library which contains ready-made methods for static and transient optimization. These optimization methods are easy to use but only customizable to a limited extent. Therefore, it is barely possible to adapt the algorithm individually for different optimization tasks. Since the optimization is done directly, large simulation models cause long optimization times. For a runtime reduction parallelization can be used. To ensure a better optimization performance with higher flexibility, tools such as Ansys optiSLang can be used. An implementation of an appropriate workflow for coupling Modelica/Dymola and optiSLang is presented in this paper. It has been shown that the optimization via optiSLang can be more time efficient compared to the Dymola standard DLR Optimization Library. When using parallelization methods, the runtime of the examined test case has been reduced by the factor 4. This improvement is necessary to deal with large scale optimization problems in a suitable time. By using the sensitivity analysis, it is possible to obtain additional information about the system behavior. The determined surrogate model (meta-model) can also be used for an indirect optimization which is especially advantageous for a repeated optimization of the same model. Due to the deviations from the model behavior indirect methods should only be used for pre-optimizations.

The comparison with the DLR Optimization Library shows that the developed workflow makes it possible to solve static optimization problems very efficiently. Further investigations should focus on optimization problems with a higher complexity to identify the limits of the presented workflow. In addition, first investigations showed that dynamic optimization problems may be realized by adjusting the workflow. For this, the input signals/vectors need to be discretized into a finite number of static points, which can later be used like the tuners of the static optimization.

Nomenclature

f	Objective function	[-]
x	Optimization variable	[-]
d_i	Inner tube diameter	[m]
d_o	Outer tube diameter	[m]
l	Tube length	[m]

η	Overall system efficiency in nominal operation	[%]
η_{opt}	Optimized overall system efficiency in nominal operation	[%]
K_i	Investment costs	[€]
$K_{i,ref}$	Reference investment costs	[€]
A	Heat exchange surface	[m^2]
A_{ref}	Reference heat exchange surface	[m^2]
PI	Price index	[-]
PI_{ref}	Reference price index	[-]
P_{in}	Input Power	[MWh]
P_{out}	Output Power	[MWh]
n	Degression exponent	[-]
n_T	Number of tubes	[-]
n_s	Number of model simulations	[-]
t	Runtime	[h]
t_{model}	Model runtime	[h]
t_{sim}	Simulation time	[h]
t_{approx}	Approximation time	[h]
t_{opt}	Optimization time	[h]
t_{dir}	Runtime of the direct method	[h]
t_{indir}	Runtime of the indirect method	[h]
$t_{DLR,direct}$	Runtime of the direct DLR optimization Library method	[h]

References

[1] A. Subramanian, T. Gundersen, and T. Adams, Eds., *Modeling and Simulation of Energy Systems: A Review*. Processes, 2018, doi: 10.3390/pr6120238.

[2] J. Akesson, *Optimica - An Extension of Modelica Supporting Dynamic Optimization*. The Modelica Association, 2008. [Online]. Available: https://www.researchgate.net/publication/253089188_Optimica-An_Extension_of_Modelica_Supporting_Dynamic_Optimization

[3] Modelica Association, *Modelica - A Unified Object-Oriented Language for Physical Systems Modeling Version 3.2 Revision 2*, 2013. [Online]. Available: https://www.modelica.org/

[4] Dassault Systèmes, *Dymola - Dynamic Modeling Laboratory: Dymola Release Notes*, 2022. [Online]. Available: https://www.3ds.com/

[5] Dynardo. “optiSLang - Produktübersicht.” https://www.dynardo.de/software/optislang.html (accessed Jan. 28, 2020).

[6] A. Pfeiffer, *Documentation, Optimization Library for Dymola: Version 2.2.2 - Tutorial*, 2016.

[7] K. Siebertz, D. van Bebber, and Thomas Hochkirchen, *Statistische Versuchsplanung: Design of Experiments (DoE)*. Springer-Verlag Berlin Heidelberg, 2010.

[8] L. Göllmann *et al., Mathematik für Ingenieure: Verstehen Rechnen Anwenden: Band 2: Analysis in mehreren Variablen, Differenzialgleichungen, Optimierung* (Lehrbuch). Berlin: Springer Vieweg, 2017.

[9] A. Wünsch, “Effizienter Einsatz von Optimierungsmethoden in der Produktentwicklung durch dynamische Parallelisierung,” Dissertation, Fakultät für Maschinenbau, Otto-von-Guericke-Universität Magdeburg, 2016.

[10] A. M. Vogelsang, “Mehrzieloptimierung von solarthermischen Parabolrinnenkraftwerken unter Berücksichtigung variabler Vergütungsschemata mit Hilfe technischer Auslegungsparameter,” Dissertation, Universität Flensburg, 2014.

[11] K. Graichen, “Methoden der Optimierung und optimalen Steuerung (WS 2017/2018),” Skriptum, Institut für Mess–; Regel– und Mikrotechnik, Universität Ulm, Institut für Mess–, Regel– und Mikrotechnik, Universität Ulm, 2017.

[12] S. Vajna, H. Bley, P. Hehenberger, C. Weber, and K. Zeman, *CAx für Ingenieure: Eine praxisbezogene Einführung,* 2nd ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. [Online]. Available: http://site.ebrary.com/lib/alltitles/docDetail.action?docID=10274774

[13] J. Kallrath, *Gemischt-ganzzahlige Optimierung: Modellierung in der Praxis: Mit Fallstudien aus Chemie, Energiewirtschaft, Papierindustrie, Metallgewerbe, Produktion und Logistik,* 2nd ed., 2013.

[14] A. Schumacher, *Optimierung mechanischer Strukturen: Grundlagen und industrielle Anwendungen,* 2nd ed. Berlin, Heidelberg: Springer, 2013.

https://doi.org/10.52202/069564-0116

[15] M. Pieper, *Mathematische Optimierung: Eine Einführung in die kontinuierliche Optimierung mit Beispielen* (essentials). Wiesbaden: Springer Spektrum, 2017.
[16] B. Mu, Y. Li, J. M. House, and T. I. Salsbury, Eds., *Real-time optimization of a chilled water plant with parallel chillers based on extremum seeking control*. Applied Energy, 2017, doi: 10.1016/j.apenergy.2017.09.072.
[17] F. Liu, J. Deng, and W. Pan, Eds., *Model-based Dynamic Optimal Control of an Ejector Expansion CO2 Heat Pump Coupled with Thermal Storages*. Energy Procedia, 2018, doi: 10.1016/j.egypro.2018.09.074.
[18] S. Senthil and S. Senthilmurugan, Eds., *Reverse Osmosis–Pressure Retarded Osmosis hybrid system: Modelling, simulation and optimization*. Desalination 389, 2016, doi: 10.1016/j.desal.2016.01.027.
[19] J. Chen and H. E. Garcia, Eds., *Economic optimization of operations for hybrid energy systems under variable markets*. Applied Energy, 2016, doi: 10.1016/j.apenergy.2016.05.056.
[20] C. Hoffmann and H. Puta, Eds., *Dynamic optimization of energy supply systems with modelica models: Energy saving control in plants and buildings*, 2006, doi: 10.3182/20061002-4-BG-4905.00009.
[21] M. Gräber, Ed., *A tool chain for the efficient solution of optimal control point*. Linköping University Electronic Press, 2017, doi: 10.3384/ecp17132249.
[22] M. Karami and L. Wang, Eds., *Particle Swarm optimization for control operation of an all-variable speed water-cooled chiller plant*. Applied Thermal Engineering, 2018, doi: 10.1016/j.applthermaleng.2017.11.037.
[23] H. Thieriot. “Towards Design Optimization with OpenModelica Emphasizing Parameter Optimization with Genetic Algorithms.” https://ep.liu.se/ecp/063/084/ecp11063084.pdf
[24] G. Delikaya, “Supervisory control of a combined heat and power plant by economic optimization,” Masterarbeit, Fraunhofer ISE, 2015.
[25] S. Huang, W. Zuo, and M. D. Sohn, Eds., *Improved cooling tower control of legacy chiller plants by optimizing the condenser water set point*. Buildings and Enviroment, 2017, doi: 10.1016/j.buildenv.2016.10.011.
[26] J. Granderson, G. Lin, D. Blum, J. Page, M. Spears, and M. A. Piette, Eds., *Integrating diagnostics and model-based optimization*. Energy & Buildings, 2019, doi: 10.1016/j.enbuild.2018.10.015.
[27] M. Ali, V. Vukovic, M. H. Sahir, and G. Fontanella, Eds., *Energy analysis of chilled water system configurations using simulation-based optimization*. Energy and Buildings, 2013, doi: 10.1016/j.enbuild.2012.12.011.
[28] A. Pfeiffer, *Numerische Sensitivitätsanalyse unstetiger multidisziplinärer Modelle mit Anwendungen in der gradientenbasierten Optimierung* (Zugl.: Halle, Univ., Naturwissenschaftliche Fak. III, Diss.) (Berichte aus dem Institut für Robotik und Mechatronik 417). Düsseldorf: VDI-Verl., 2008.
[29] J. Riedel, *Gewichtsoptimierung eines Kreuzfahrtschiffes unter Spannungsrestriktionen*. Institutskolloquium: Dynardo, 2000. Accessed: Jan. 25, 2020. [Online]. Available: www.dynardo.de
[30] TLK-Thermo GmbH and XRG-Simulation GmbH, *Clara Lib - Version 1.4.0*. Accessed 20 March 2020, 2019. [Online]. Available: https://www.claralib.com/
[31] J. Branke, K. Deb, K. Miettinen, and R. Słowiński, *Multiobjective Optimization* (5252). Berlin, Heidelberg: Springer Berlin Heidelberg, 2008.
[32] Lehner. “Betriebstechnik - BET -Wirtschaftlichkeitsrechnungen und Betriebliches Rechnungswesen.” https://docplayer.org/14249839-Betriebstechnik-bet-teil-2-wirtschaftlichkeitsrechnungen-und-betriebliches-rechnungswesen.html (accessed Jan. 16, 2023).
[33] C. Lühe, “Modulare Kostenschätzung als Unterstützung der Anlagenplanung für die Angebots- und frühe Basic Engineering Phase,” Dissertation, Technischen Universität Berlin, 2013.
[34] E. Zitzler, M. Laumanns, and L. Thiele, Eds., *SPEA2: Improving the strength pareto evolutionary algorithm*. ETH Zurich, 2001, doi: 10.3929/ethz-a-004284029.