COMANDO: A Next-Generation Open-Source Framework for Energy Systems Optimization

Marco Langiu[a,b], David Yang Shu[a,c], Florian Joseph Baader[a,b], Dominik Hering[a,b], Uwe Bau[a], André Xhonneux[a], Dirk Müller[d,a,e], André Bardow[d,a,f,c], Alexander Mitsos[d,a,g], Manuel Dahmen[a,*]

[a] Forschungszentrum Jülich GmbH, Institute of Energy and Climate Research, Energy Systems Engineering (IEK-10), Jülich 52425, Germany

[b] RWTH Aachen University Aachen 52062, Germany

[c] ETH Zürich, Energy & Process Systems Engineering, Zürich 8092, Switzerland

[d] JARA-ENERGY, Jülich 52425, Germany

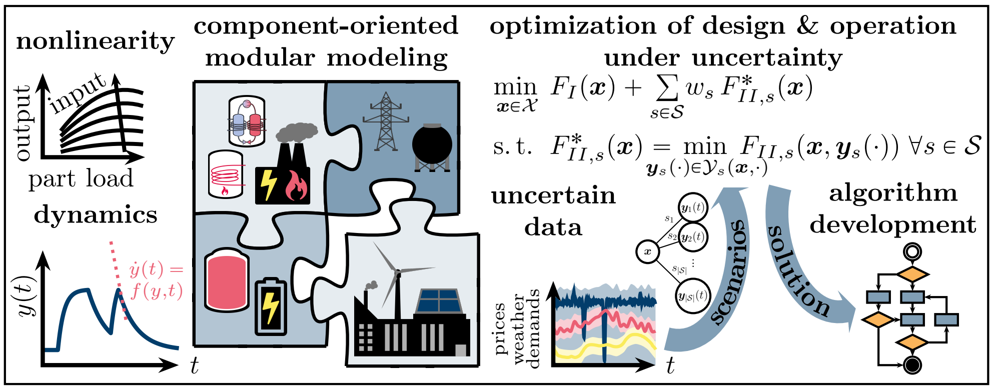
[e] RWTH Aachen University, E.ON Energy Research Center, Institute for Energy Efficient Buildings and Indoor Climate, Aachen 52056, Germany

[f] RWTH Aachen University, Institute of Technical Thermodynamics, Aachen 52056, Germany

[g] RWTH Aachen University, Process Systems Engineering (AVT.SVT), Aachen 52074, Germany

Abstract: Existing open-source modeling frameworks dedicated to energy systems optimization typically utilize (mixed-integer) linear programming ((MI)LP) formulations, which lack modeling freedom for technical system design and operation. We present COMANDO, an open-source Python package for component-oriented modeling and optimization for nonlinear design and operation of integrated energy systems. COMANDO allows to assemble system models from component models including nonlinear, dynamic and discrete characteristics. Based on a single system model, different deterministic and stochastic problem formulations can be obtained by varying objective function and underlying data, and by applying automatic or manual reformulations. The flexible open-source implementation allows for the integration of customized routines required to solve challenging problems, e.g., initialization, problem decomposition, or sequential solution strategies. We demonstrate features of COMANDO via case studies, including automated linearization, dynamic optimization, stochastic programming, and the use of nonlinear artificial neural networks as surrogate models in a reduced-space formulation for deterministic global optimization.

Keywords:
energy systems modeling, integrated energy systems, design and operation, nonlinear optimization


nonlinearity
output
input
part load
dynamics
$y(t)$
$\dot{y}(t) = f(y,t)$
t
component-oriented modular modeling
optimization of design & operation under uncertainty
$\min_{x \in \mathcal{X}} F_I(x) + \sum_{s \in \mathcal{S}} w_s F^*_{II,s}(x)$
s. t. $F^*_{II,s}(x) = \min_{y_s(\cdot) \in \mathcal{Y}_s(x,\cdot)} F_{II,s}(x, y_s(\cdot)) \; \forall s \in \mathcal{S}$
uncertain data
prices
weather
demands
t
scenarios
solution
algorithm development

Highlights:

- Open-source framework for optimization of energy systems design and operation
- Component-oriented modeling, allowing for hybrid mechanistic/data-driven models
- Optimization considering nonlinearity, dynamics and parametric uncertainty
- Four case studies, demonstrating flexibility and wide range of application

*Manuel Dahmen, Forschungszentrum Jülich GmbH, Institute of Energy and Climate Research, Energy Systems Engineering (IEK-10), Jülich 52425, Germany
E-mail: m.dahmen@fz-juelich.de

1 Introduction

Energy systems are networks of interconnected components that generate and transform energy using a set of renewable or fossil resources to satisfy various kinds of demands (Beller, 1976). The economic and ecologic performance of energy systems is strongly influenced by system design and operation. The design comprises all choices regarding the configuration, i.e., the selection and interconnection of components (discrete variables), as well as sizing and other technical specifications (continuous variables). The operation comprises commitment (discrete variables) and dispatch (continuous variables) of individual components, i.e., how their activity and output levels are chosen at different points in time. The prospective operation also needs to be taken into account during system design (Pistikopoulos, 1995; Frangopoulos et al., 2002). However, energy demands, prices, weather and other operational aspects can be highly variable and their future values are inherently uncertain, rendering the design and operation of energy systems a challenging decision process. To ensure optimal economic and ecologic performance, it is common to cast these decision processes into mathematical optimization problems (e.g. Papoulias and Grossmann, 1983; Ghobeity and Mitsos, 2012; Gunasekaran et al., 2014; Andiappan, 2017; Frangopoulos, 2018; Demirhan et al., 2019; Sass et al., 2020). This is typically done via general purpose algebraic modeling languages (AMLs), e.g., GAMS (Bussieck and Meeraus, 2004) or Pyomo (Hart et al., 2011), or via specialized energy system modeling frameworks (ESMFs), e.g., OSeMOSYS (Howells et al., 2011) or oemof (Hilpert et al., 2018). While AMLs offer flexibility in the choice of algebraic formulation and solution approach, ESMFs employ a component-oriented modeling approach, i.e., system models are created by specifying connections between component models. This approach simplifies the modeling process, model maintenance, and model re-use.

Established ESMFs typically employ linear programming (LP) (Schrattenholzer, 1981; Fishbone and Abilock, 1981; Loulou and Labriet, 2007; Bakken et al., 2007; Howells et al., 2011; Hunter et al., 2013; Dorfner, 2016) or mixed-integer linear programming (MILP) (Pfenninger and Keirstead, 2015; Hilpert et al., 2018; Atabay, 2017; Brown et al., 2018; Johnston et al., 2019) formulations, well-suited for techno-economic analysis of large-scale systems (Connolly et al., 2010; Pfenninger et al., 2014; van Beuzekom et al., 2015). In contrast, technical system design and operation must consider more detailed system behavior, often giving rise to nonlinearities and dynamic effects that are difficult or impractical to represent with MILP formulations, (see e.g., Li et al., 2011; Goderbauer et al., 2016; Schäfer et al., 2019a,b). To address the challenges of technical design and operation, we propose a next-generation ESMF for component-oriented modeling and optimization for nonlinear design and operation (COMANDO), an open source Python package (COMANDO Repository, 2020). COMANDO borrows a generic, nonlinear representation of mathematical expressions and features for algorithm development from AMLs, and the representation of differential equations and more general system model aggregation from differential-algebraic modeling frameworks (DAMFs) such as gPROMS (Process Systems Enterprise, 2019), MODELICA (Elmqvist and Mattsson, 1997), or DAE Tools (Nikolić, 2016). With this combination of features, COMANDO incorporates flexible nonlinear and dynamic modeling into the modularity of an ESMF. Additionally, COMANDO enables the simultaneous consideration of multiple operating scenarios through a two-stage stochastic programming formulation, allowing for rigorous optimization of energy system design and operation under uncertainty and/or variability of operating conditions. While the vast majority of existing ESMFs is implemented as a layer on top of an AML, COMANDO is based on the computer algebra system SymPy (Meurer et al., 2017). SymPy provides data structures for representing generic mathematical expressions and corresponding methods to analyze and manipulate expressions. These features facilitate the creation of automatic reformulation routines (e.g., automatic linearization), custom interfaces to AMLs or solvers, and user-defined solution algorithms.

This paper is structured as follows: In Section 2, we give a brief review of the state of the art in optimization-based energy-system design and operation and identify the lack of an open-source tool dedicated specifically to the technical design and operation of different types of energy systems. To this end, we present COMANDO in Section 3. In Section 4, we present four case studies highlighting important features of COMANDO. Section 5 concludes the work.

2 Optimization-based energy system design and operation

In Section 2.1 we introduce a generic mathematical programming problem for the optimal design and operation of energy systems. In Section 2.2, we briefly summarize advantages and disadvantages of the three major classes of tools that can be used to formulate and tackle variants of this problem, namely algebraic modeling languages (AMLs), energy system modeling frameworks (ESMFs) and differential-algebraic modeling frameworks (DAMFs).

2.1 Problem formulation

Realizing an optimal energy system requires optimal decisions at both the design stage and the operational stage. Due to the variability and uncertainty associated to energy system operation, there can be many relevant operational scenarios that need to be considered to obtain a reliable design. A suitable modeling approach for this setting is two-stage stochastic programming (Dantzig, 1955; Birge and Louveaux, 2011; Li and Barton, 2015; Yunt et al., 2008). It allows for the simultaneous consideration of multiple operating scenarios $s \in \mathcal{S}$, resulting in the following problem structure:

$$
\begin{aligned}
&\min_{\boldsymbol{x}} F_I(\boldsymbol{x}) + \sum_{s\in\mathcal{S}} w_s\, F^*_{II,s}(\boldsymbol{x})\\
&\text{s.t. } \boldsymbol{g}_I(\boldsymbol{x}) \leqslant \mathbf{0}\\
&\quad \boldsymbol{h}_I(\boldsymbol{x}) = \mathbf{0}\\
&\quad \left.\begin{aligned}
&F^*_{II,s}(\boldsymbol{x}) = \min_{\boldsymbol{y}_s(\cdot)}\ F_{II,s}(\boldsymbol{x},\boldsymbol{y}_s(\cdot)) = \int_{\mathcal{T}_s} \dot{F}_{II}\big(\boldsymbol{x},\boldsymbol{y}_s(t),\boldsymbol{p}_s(t)\big)\ \mathrm{d}t\\
&\qquad\qquad \text{s.t. } \boldsymbol{y}^{\mathrm{d}}_s(t=0) = \boldsymbol{y}^{\mathrm{d}}_{s,0}\\
&\qquad\qquad\quad \left.\begin{aligned}
&\dot{\boldsymbol{y}}^{\mathrm{d}}_s(t) = \boldsymbol{f}\big(\boldsymbol{x},\boldsymbol{y}_s(t),\boldsymbol{p}_s(t)\big)\\
&\boldsymbol{g}_{II}\big(\boldsymbol{x},\boldsymbol{y}_s(t),\boldsymbol{p}_s(t)\big) \leqslant \mathbf{0}\\
&\boldsymbol{h}_{II}\big(\boldsymbol{x},\boldsymbol{y}_s(t),\boldsymbol{p}_s(t)\big) = \mathbf{0}\\
&\boldsymbol{y}_s(t) = [\boldsymbol{y}^{\mathrm{d}}_s(t), \ldots]\\
&\boldsymbol{y}_s(t) \in \mathcal{Y}_s(t) \subset \mathbb{R}^{n_y}\times\mathbb{Z}^{m_y}
\end{aligned}\right\}\ \forall t\in\mathcal{T}_s\\
&\qquad\qquad\quad \mathcal{T}_s = [0, T_s]
\end{aligned}\right\}\ \forall s\in\mathcal{S}\\
&\quad \boldsymbol{x}\in\mathcal{X}\subset\mathbb{R}^{n_x}\times\mathbb{Z}^{m_x}\\
&\quad \mathcal{S} = \{s_1, s_2, \cdots, s_{|\mathcal{S}|}\}
\end{aligned}
\tag{P}
$$

The two-stage structure of (P) distinguishes between design- and operation-related variables, constraints, and objectives. We group design decisions into the vector $\boldsymbol{x}$ and operational decisions into one vector $\boldsymbol{y}_s(\cdot)$ for each scenario s, with associated probability of occurrence w_s. Further, the operational decisions are functions of time t from a continuous operating horizon $\mathcal{T}_s = [0, T_s]$ (in general, each scenario may consider a different time-horizon). Likewise, for different scenarios s the input data, i.e., the values of model parameters $\boldsymbol{p}_s(\cdot)$, may be functions of time t. The objective function of the first stage is comprised of design costs F_I and the expected value of the optimal operating costs. For a given design $\boldsymbol{x}$ and scenario s, the optimal operating costs $F^*_{II,s}$ correspond to the optimal objective value of the second stage. The operating costs are described by an integral over the operating horizon $\mathcal{T}_s$ of the momentary operating costs $\dot{F}_{II}$. The set of feasible design and operational decisions is described via constraints $\boldsymbol{g}_I$, $\boldsymbol{g}_{II}$, $\boldsymbol{h}_I$, and $\boldsymbol{h}_{II}$ (with an appropriate number of elements in $\boldsymbol{h}_{II}$, allowing for degrees of freedom), as well as bounds and integrality restrictions in the form of $\mathcal{X}$ and $\mathcal{Y}_s(t)$, with n and m corresponding to the number of continuous and discrete decisions, respectively. Additionally, for the subset of operational variables that correspond to differential states (identified via the superscript $^{\mathrm{d}}$), an initial state $\boldsymbol{y}^{\mathrm{d}}_{s,0}$ and the right hand side $\boldsymbol{f}$ of a corresponding differential equation are given.

Formulation (P) covers both mixed design and operation problems, as well as pure operational problems (with fixed design decisions $\boldsymbol{x}$). If the values of w_s are interpreted as frequencies of occurrence for a certain operational setting, the corresponding scenarios can also be interpreted as typical operating points or periods, as done e.g., in Yunt et al. (2008) and Baumgärtner et al. (2019a), respectively. Such scenarios can be derived from standardized reference load profiles, or via clustering of historical data (see, e.g., Schütz et al., 2018). If constraints coupling different scenarios are added to formulation (P), problems considering long-term effects such as seasonal storage can also be considered (see, e.g., Gabrielli et al., 2018; Baumgärtner et al., 2019b).

The two-stage formulation (P) can be cast into an equivalent single-stage formulation, also referred to as the *deterministic equivalent*, see e.g., (Yunt et al., 2008), that can be solved with general-purpose solvers. While solvers interfaced from DAMFs, as well as some specialized dynamic optimization solvers, e.g., DyOS (Caspari et al., 2019), directly accept continuous-time problem formulations and take care of time-discretization internally, almost all solvers available via AMLs and ESMFs require discrete-time formulations as input. To obtain a discrete-time formulation, a particular discretization scheme is chosen, and $\boldsymbol{y}_s(\cdot)$, $F_{II,s}(\boldsymbol{x}, \boldsymbol{y}_s(\cdot))$, and $\boldsymbol{f}\big(\boldsymbol{x}, \boldsymbol{y}_s(t), \boldsymbol{p}_s(t)\big)$ are replaced by corresponding discrete-time counterparts.

An alternative to solving the deterministic equivalent is to employ an algorithm capable of exploiting the special constraint structure of the two-stage formulation (P). Such an algorithm decomposes (P) into multiple subproblems that are solved iteratively to obtain increasingly tighter bounds on the solution of (P). Different decomposition algorithms are applicable, depending on the presence and location of nonlinearity, nonconvexity and integrality; for a concise overview, see Li and Grossmann (2019).

2.2 Tools

Both deterministic equivalent formulations as well as suitable decomposition algorithms can be implemented in AMLs such as AMPL (Fourer et al., 1990), GAMS (Bussieck and Meeraus, 2004), or AIMMS (Bisschop, 2006). In recent years, several AML extensions have been developed that can be leveraged for energy system modeling. In particular, stochastic programming related functionality has been incorporated widely, both in commercial AMLs such as AMPL (Fourer et al., 1990) and GAMS (Bussieck and Meeraus, 2004) (through SAMPL (Valente et al., 2009) and Extended Mathematical Programming (Ferris et al., 2009), respectively), as well as in the open-source AMLs Pyomo (Hart et al., 2011) and JuMP (Dunning et al., 2017) (through PySP (Watson et al., 2012) and StructJuMP, formerly StochJuMP, (Huchette et al., 2014) or StochasticPrograms.jl (Biel and Johansson, 2019), respectively). Further modeling constructs tailored towards special problem structures have been incorporated through block-oriented modeling (Friedman et al., 2013) in Pyomo and through Plasmo.jl (Jalving et al., 2017, 2019) in JuMP. Finally, Pyomo.DAE (Nicholson et al., 2018) enables the direct representation of differential equations within optimization problems expressed in Pyomo and provides various options for automatic discretization. Through the combination of features offered by these extensions, newer AMLs are in principle well suited to model and optimize energy system design and operation. However, their abstract nature can complicate implementation, code maintenance, and re-use, and renders the resulting problem formulations difficult to comprehend. Development on the Pyomo AML has resulted in the modeling tool IDAES (Miller et al., 2018), which employs methodologies from process systems engineering with the aim of advancing fossil energy systems (IDAES homepage, 2020). In particular, IDAES provides models for thermal power plants and associated components. These systems are considered in the form of process flowsheets, i.e., components are modeled as control volumes with in- and outflows, whose steady-state and dynamic behavior can be specified via so-called property packages.

Compared to AMLs and their various extensions, ESMFs provide an even higher level of abstraction, allowing to model generic energy systems comprised of utilities for generating, converting, or storing different energy forms. This higher level of abstraction is commonly achieved via an interface layer on top of an AML that separates component and system modeling from problem formulation. In a first modeling step, models of energy system components, e.g., boilers, combined-heat-and-power units, or heat pumps are created. These component models contain variables, parameters and constraints specifying possible in- and outputs as well as the internal component behavior. In a second modeling step, system models are aggregated by specifying the connections between different components. Finally, component- and system-level constraints are combined with an objective, e.g., the minimization of total annualized cost (TAC) or global warming impact (GWI), yielding a problem formulation that can be passed to an appropriate solver.

The modular, object-oriented nature of modern ESMFs such as oemof (Hilpert et al., 2018) allows component and system models to be implemented as classes, inheriting reoccurring functionality, e.g., from generic models representing generation, transformation, storage or consumption of different energy commodities. Such inheritance allows for more structured modeling, thereby simplifying model maintenance and re-use compared to AMLs, e.g., through the creation of component libraries. However, the vast majority of ESMFs is based on either linear programming (LP) or mixed-integer linear programming (MILP) formulations, i.e., all participating functions must be linear in the decision variables $\boldsymbol{x}$ and $\boldsymbol{y}$. In such ESMFs, the user must provide linear approximations for all nonlinear expressions. While this is usually not considered a limitation in the context of *system analysis*, i.e., the principal focus of most ESMFs (cf. Pfenninger et al., 2014), problems concerned with *technical design and operation* need to represent systems in more detail, often giving rise to nonlinearities that are difficult or impractical to linearize. In the presence of such nonlinearities, it is often sensible to use the original nonlinear equations or nonlinear surrogate models such as artificial neural networks (ANNs), as, e.g., in Schäfer et al. (2020), which however is not possible in MILP-based ESMFs.

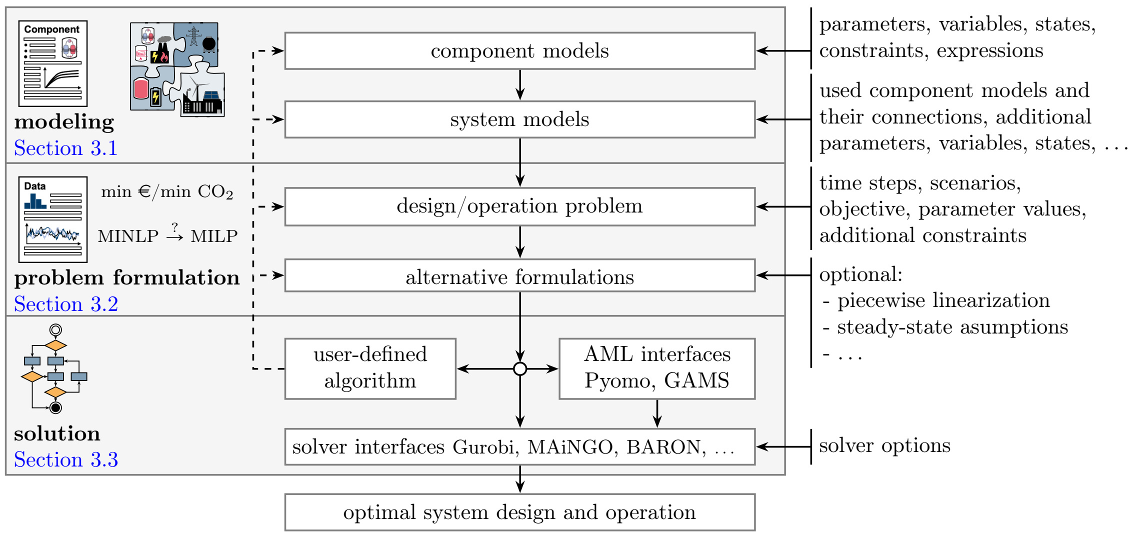
Besides AMLs and ESMFs, differential-algebraic modeling frameworks (DAMFs) constitute a third class of tools that can be used to model energy systems. DAMFs also employ a component-oriented modeling approach, which, however, is more general than in a typical ESMF: In DAMFs, components may correspond to actual physical machinery or to a particular physical phenomenon (e.g., heat transfer) and can constitute subsystems, which are themselves composed of other components. Additionally, the information exchanged between components is not restricted to a particular kind of quantity, such as energy. DAMFs are particularly focused on detailed operational aspects, allowing for differential equations and nonlinear expressions within component models. They provide powerful features for operational *simulation* of the resulting models for which a fixed design is assumed. Design *optimization* is also possible in several DAMFs, (e.g., Smith, 1997; Pfeiffer, 2012), and the commercial tool gPROMS (Process Systems Enterprise, 2019) even allows for the direct consideration of parametric uncertainty using formulations similar to Problem (P), (see, e.g., Bansal et al., 2000). In contrast, noncommercial, open-source tools such as Open-Modelica (Thieriot et al., 2011) or Optimica (Åkesson et al., 2010) are currently limited to a single set of operational data, impeding design optimization under uncertainty. Furthermore, DAMFs usually offer less freedom in the choice of problem formulation, solver or algorithm in comparison to AMLs. In particular, many tools employ gradient-based methods, (e.g., Pfeiffer, 2012; Navarro and Vassiliadis, 2014; Magnusson and Åkesson, 2015) yielding only local solutions, or heuristic global optimization methods, e.g., random search, genetic algorithms, or simulated annealing (Thieriot et al., 2011; Pfeiffer, 2012; Kim et al., 2018), which treat the system model as a black box and cannot reliably locate global solutions.

AMLs, ESMFs and DAMFs each exhibit strengths related to a particular aspect of modeling and optimizing energy system design and operation. ESMFs are tailored to energy systems modeling and offer a component-oriented approach that benefits model maintenance and re-usability. However, their principal focus is on system analysis. In particular, their restriction to LP or MILP formulations makes them less suited for applications concerned with technical design and operation. Both AMLs and DAMFs lift the restriction to (MI)LP formulations, but AMLs lack high-level component-oriented abstractions for generic energy systems and DAMFs lack control over the choice of problem formulation and optimization algorithm. We therefore propose a next-generation ESMF that allows for flexible, component-oriented modeling, including nonlinear and differential-algebraic formulations, parametric uncertainty, and the possibility to specify specialized solution algorithms. Its basic structure is presented in the following Section.

3 The COMANDO ESMF

The goal of COMANDO is to provide an open-source framework which allows to generate detailed models of energy system components, including differential-algebraic and nonlinear elements, and aggregate them to system models for the purpose of optimization. Unlike most ESMFs, which are commonly based on an AML, COMANDO is implemented as a flat layer on top of the computer algebra system SymPy (Meurer et al., 2017). This choice provides: i) data structures for the mathematical expressions used to describe components and systems, as well as ii) several routines useful for creating automatic reformulations and user-defined algorithms, such as automatic differentiation, substitution of expressions or solution of nonlinear systems of equations. An overview of the structure of COMANDO and the typical workflow of modeling, problem formulation and optimization is given in Fig. 1.

In Section 3.1 we describe the process of creating models for components and systems in COMANDO.



Component
modeling
Section 3.1
Data
min €/min CO$_2$
MINLP $\xrightarrow{?}$ MILP
problem formulation
Section 3.2
solution
Section 3.3
component models
system models
design/operation problem
alternative formulations
user-defined algorithm
AML interfaces Pyomo, GAMS
solver interfaces Gurobi, MAiNGO, BARON, ...
optimal system design and operation
parameters, variables, states, constraints, expressions
used component models and their connections, additional parameters, variables, states, ...
time steps, scenarios, objective, parameter values, additional constraints
optional:
- piecewise linearization
- steady-state assumptions
- ...
solver options

Fig. 1. Workflow for modeling, problem formulation, and optimization using COMANDO.

Section 3.2 provides details on how optimization problems can be created from a system model and how alternative formulations of these problems can be obtained. Finally, the different options for solving the formulated problems are given in Section 3.3.

3.1 Modeling process

The goal of the modeling phase is to generate a model describing the behavior of a given energy system. For the creation of such a system model, models for its constituting components as well as information on their connectivity are required.

We begin with the description of component models, which are used to represent elementary parts of an energy system. Fig. 2 depicts the structure of the `Component` class used for that purpose. A model of a component i consists of several types of mathematical expressions, given in symbolic form. Following the notation introduced in Section 2, the expressions describing the component contain different symbols corresponding to quantities which are either *parameters* ($\boldsymbol{p}_i$), i.e., placeholders for values that are assumed to be given before an optimization, or design or operational *variables* ($\boldsymbol{x}_i$ and $\boldsymbol{y}_i$, respectively), i.e., placeholders for scalar and vector values that are to be determined during optimization.

To instantiate a `Component`, a unique name must be provided, which serves as an identifier for the component. The names of parameters, variables, and constraints associated to the component are prepended with this identifier, in order to distinguish quantities from different instances of the same component model. The `Component` class can either be used directly or subclassed to specify specialized component classes with custom behavior. To create and add symbols to a component, the `Component` class provides three methods:

- `make_parameter`,
- `make_design_variable`, and
- `make_operational_variable`.

All three methods require a name for the symbol that is used to represent the quantity. The methods for creating variables provide optional arguments for the specification of variable bounds, domain (integer/real) and a scalar value for initialization, while the parameter creation method only provides a single optional argument for the specification of its value. Note that time- and scenario-specific values for operational quantities are set in the problem formulation phase after the time and scenario structure, has been specified, see Section 3.2.

Based on variables and parameters, mathematical expressions can be formed using the overloaded Python operators `+`, `-`, `*`, `/`, `**`, or any of the functions implemented in SymPy (e.g., exp, log, trigono-

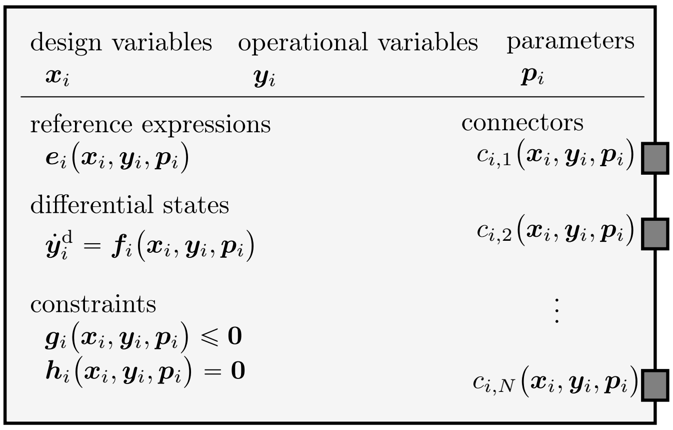

design variables
$\boldsymbol{x}_i$
operational variables
$\boldsymbol{y}_i$
parameters
$\boldsymbol{p}_i$
reference expressions
$\boldsymbol{e}_i(\boldsymbol{x}_i, \boldsymbol{y}_i, \boldsymbol{p}_i)$
differential states
$\dot{\boldsymbol{y}}_i^{\mathrm{d}} = \boldsymbol{f}_i(\boldsymbol{x}_i, \boldsymbol{y}_i, \boldsymbol{p}_i)$
constraints
$\boldsymbol{g}_i(\boldsymbol{x}_i, \boldsymbol{y}_i, \boldsymbol{p}_i) \leqslant \boldsymbol{0}$
$\boldsymbol{h}_i(\boldsymbol{x}_i, \boldsymbol{y}_i, \boldsymbol{p}_i) = \boldsymbol{0}$
connectors
$c_{i,1}(\boldsymbol{x}_i, \boldsymbol{y}_i, \boldsymbol{p}_i)$
$c_{i,2}(\boldsymbol{x}_i, \boldsymbol{y}_i, \boldsymbol{p}_i)$
$\vdots$
$c_{i,N}(\boldsymbol{x}_i, \boldsymbol{y}_i, \boldsymbol{p}_i)$

Fig. 2. Structure of a generic component i in COMANDO. Mathematical expressions are specified based on symbols that are either parameters, design variables, or operational variables. These expressions can be kept for later reference, constitute the right-hand side of differential equations, form part of algebraic constraints, or describe possible in- and/or outputs through connectors.

metric, and hyperbolic functions). Any intermediate expressions $\boldsymbol{e}_i$ that are of interest can be assigned an identifier and stored in the component using the `add_expression` method. These expressions can simply be used for evaluation or as parts of more complex expressions, e.g., system-level constraints, or an objective function, cf. Section 3.2. Vectors $\boldsymbol{g}_i$ and $\boldsymbol{h}_i$ contain inequality and equality constraints associated to the component i and their elements can be specified using the methods

- `add_le_constraint`,
- `add_eq_constraint`, and
- `add_ge_constraint`.

Each of these methods takes two expressions and an optional name for the resulting relation as arguments. Explicit distinction into first and second stage expressions and constraints is not necessary and occurs automatically, based on the symbols present in the respective expressions.

Dynamic behavior can be represented by specifying right-hand side expressions $\boldsymbol{f}_i$ for the time derivatives of differential states $\boldsymbol{y}_i^{\mathrm{d}}$ (recall that $\boldsymbol{y}_i^{\mathrm{d}}$ constitutes a subset of the operational variables $\boldsymbol{y}_i$). Previously created operational variables may be declared differential states using the `declare_state` method or differential states may be created directly using the `make_state` method. The first method requires an existing variable and an expression, corresponding to entries of the vectors $\boldsymbol{y}_i^{\mathrm{d}}$ and $\boldsymbol{f}_i$ as mandatory arguments and allows for the specification of an initial state as well as bounds and an initial guess for the value of the derivative. The method results in the creation of a new operational variable, corresponding to an element in $\dot{\boldsymbol{y}}_i^{\mathrm{d}}$, and an equality constraint, linking the time-derivative with the given expression in $\boldsymbol{f}_i$. An explicit relation between the state and its derivative is not specified at this point, as it depends on the desired time-discretization which is handled by the solution interfaces, cf. Section 3.2. The `make_state` method creates a new operational variable corresponding to the differential state and then calls `declare_state`.

To allow for the aggregation of components to systems, individual expressions in $\boldsymbol{c}_i$ can be assigned to connectors (cf. Fig. 2). Connectors are generally bidirectional, but may be specified to only allow for in-, or output. In- and output connectors restrict the assigned expression to a nonnegative or nonpositive range, respectively.

A system model can be created as an instance of the `System` class, whose instantiation again requires a unique label that serves as an identifier. Optionally, a list of components and the connections between them can be passed to the constructor of the `System` class. Each connection is specified via a label and a list of associated connectors. The connectors are connected to a 'bus' at which the quantities associated to them are balanced and a corresponding constraint is created automatically, see the graphical notation in Fig. 3, which is also used for the case studies in Section 4. Instead of specifying the complete structure

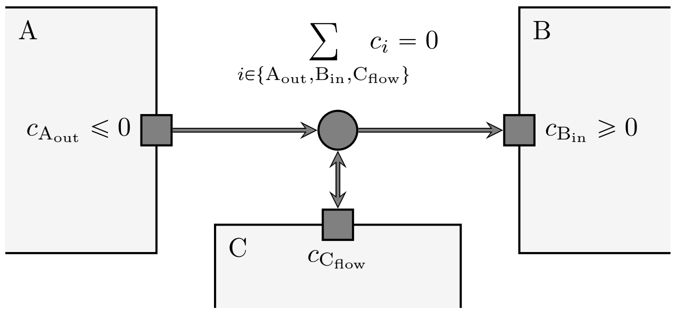

A
B
C
$\sum_{i \in \{A_{out}, B_{in}, C_{flow}\}} c_i = 0$
$c_{A_{out}} \leqslant 0$
$c_{B_{in}} \geqslant 0$
$c_{C_{flow}}$

Fig. 3. A connection formed by connecting three connectors to a bus: The components A, B, and C each define a connector for a particular quantity. The connectors of A and B are marked as outputs and inputs, respectively, restricting the sign of the associated expression, while the connector of C is not restricted. The connection of A_{out}, B_{in}, and C_{flow} via a bus results in the creation of a balance constraint in the system model. This graphical notation is also used for the case studies in Section 4.

during construction of a `System` instance, components and connections can also be added sequentially via corresponding methods, allowing for procedural model generation. As in DAMFs, a nested creation of systems from subsystems is possible by exposing connectors of individual components or extending existing connections via additional connectors. For instance, a neighborhood can be represented as a system composed of buildings as subsystems, which are in turn composed of heating, cooling and power equipment. As with component models, system models can be assigned their own variables, parameters, expressions and constraints describing their behavior. These two features are accomplished by letting the `System` class inherit from the `Component` class.

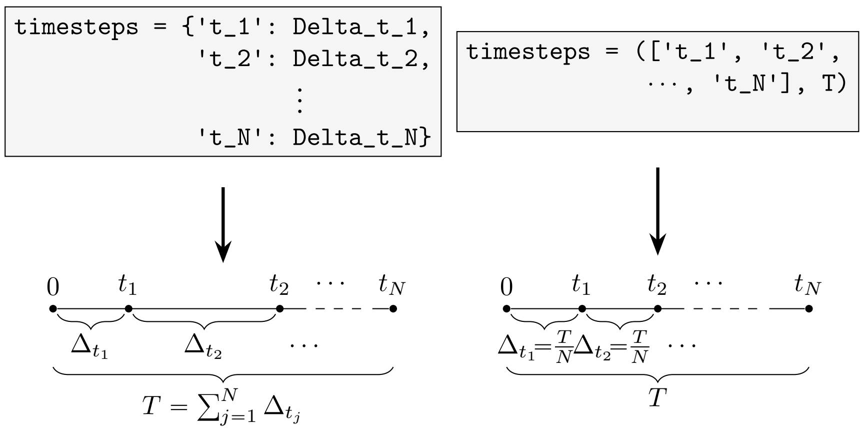
3.2 Problem formulation

Based on a system model, different kinds of optimization problems considering system design and/or operation can be created. To this end, COMANDO provides the `Problem` class, instances of which can be created by the `create_problem` method of the `System` class. As the system model defines a constraint set which is parameterized by the parameters $\boldsymbol{p}$, only the objective terms F_I and $\dot{F}_{II}$ as well as a time and scenario structure and appropriate data (i.e., values for the parameters $\boldsymbol{p}$) need to be specified in the `create_problem` method to obtain a complete problem formulation, corresponding to (P). Note that the user may decide which units to use for data and time-steps but must ensure they match. Units given in the Nomenclature are those used for the case studies in Section 4.

To define the objective terms, the `System` class provides the `aggregate_component_expressions` method. For a given expression identifier, it returns the sum of all expressions stored under that identifier in the individual components. The resulting expressions can be used for the objective terms F_I and $\dot{F}_{II}$, depending on whether they consist exclusively of first stage (i.e., scalar) quantities or not. A second use for the `aggregate_component_expressions` method is to create expressions for system-level constraints involving contributions from multiple components.

The time and scenario structure is specified in terms of the considered scenarios $s \in \mathcal{S}$ and the corresponding discretized time horizons $\widehat{\mathcal{T}}_s$. The $\widehat{\mathcal{T}}_s$ are required by COMANDO's solver or AML interfaces for the automatic discretization of the differential equations. If more than one operational scenario is considered, the different scenarios can either be specified as a list of M scenario identifiers, corresponding to scenarios with probability $1/M$, or by a series of scenario identifiers and associated weights w_s. In the latter case, the weights are not required to sum to one, allowing for a more general weighting. Similarly, individual time-points for each time horizon are either specified via a mapping of time-point labels t to the corresponding lengths $\Delta_{s,t}$ or in the case of equidistant time-steps via a list of labels and an end-time T_s, see Fig. 4. If the time horizons are identical for all scenarios, a single time-horizon can be specified, otherwise, one specification per scenario is required.

Parameter values corresponding to the resulting time and scenario structure can be specified during problem creation and may later be updated using the `data` attribute of the `Problem` instance. Similarly,



```
timesteps = {'t_1': Delta_t_1,
             't_2': Delta_t_2,
                         ⋮
             't_N': Delta_t_N}
```

```
timesteps = (['t_1', 't_2',
             ..., 't_N'], T)
```

Fig. 4. Alternative ways to specify time-steps for a particular scenario: For variable length an ordered mapping (left) and for constant length a list and the total length (right) can be specified. If multiple scenarios with different time-structures are to be considered, one such description is given per scenario.

design and operational variable values can be updated using the `design` and `operation` attributes, respectively. Values for design variables must be scalar while values for parameters and operational variables may be provided as scalars or as time- and/or scenario-dependent data.

After the abovementioned steps, a problem in the form of (P) is fully specified. However, it may be desirable to adapt the original problem formulation in different ways. Adaptations to the problem formulation range from simply adding further constraints to the reformulation of expressions in the problem. One generic reformulation routine implemented in COMANDO is the automatic linearization of arbitrary continuous multivariate expressions via convex-combination or multiple-choice linearization (Vielma et al., 2010). More generally, custom reformulations may be created making use of existing algorithms provided by SymPy (Meurer et al., 2017), e.g., for automatic differentiation, analytic solution of different kinds of nonlinear equation systems, or symbolic substitution of subexpressions. Note that reformulations do not have to result in approximations but can also be used to create alternative formulations that possess better properties than the original one, e.g., tighter relaxations for deterministic global optimization.

3.3 Problem solution

A fully specified problem formulation can be directly passed to a suitable solver or to an AML. In this step, the problem structure and data are translated from the COMANDO representation to a new representation, matching the syntax of the target solver or AML. For this purpose, COMANDO contains a generic parsing routine that can be used to create new interfaces based on target-specific representations of the symbols and operations occurring within the different expressions of the problem formulation. Interfaces may be text-based, resulting in an input file for a solver or AML, or they can be object-oriented, resulting in a translation of the problem formulation using the target-API. Currently implemented interfaces are:

- text-based:
  - BARON (Sahinidis, 2020) (solver)
  - GAMS (Bussieck and Meeraus, 2004) (AML)
  - MAiNGO (Bongartz et al., 2018) (solver)
- API-based:
  - Pyomo (Hart et al., 2011) (AML)
  - Pyomo.DAE (Nicholson et al., 2018) (AML)
  - Gurobi (Gurobi Optimization, LLC, 2020) (solver)
  - MAiNGO (Bongartz et al., 2018) (solver)

All of these interfaces provide methods to solve the deterministic equivalent formulation of Problem (P) with a given set of options, and to write back the obtained results to COMANDO. Note that a problem formulation may contain differential equations if states were defined in the component or system model. Since most solvers and AMLs do not support differential equations, the corresponding interfaces can specify different schemes for automatic time discretization. All existing interfaces implement implicit Euler discretization. More advanced schemes are available through the Pyomo.DAE interface.

Instead of directly solving a problem, it can also be addressed with a user-defined algorithm. User-defined algorithms can range from simple preprocessing routines based on the system model and available data to more advanced methods, such as decomposition techniques, commonly used in stochastic programming (see, e.g., Li and Grossmann, 2019). The architecture of COMANDO allows for manipulation at the level of component and system models as well as at the level of the resulting optimization problems. In particular the `Problem` class can also be used to specify the sub-problems that may occur within user-defined algorithms, allowing them to be passed to any of the available interfaces.

4 Case Studies

We now demonstrate key features of COMANDO in four case studies, which are illustrative of the kinds of design and operation problems we address with COMANDO. The case studies focus on different aspects of energy systems and vary in their approaches for modeling the considered systems and their components. The complete source code for all case studies can be found in the `examples` directory of the COMANDO Repository (2020).

The first case study, based on our previous work (Voll et al., 2013; Sass and Mitsos, 2019), consists of the greenfield design and operation of an industrial energy system considering both economic and environmental impact. The component models account for nonlinearities in part-load behavior and investment cost, and differential equations for the state of charge of battery and thermal energy storage units, resulting in a mixed-integer dynamic optimization (MIDO) problem. Here, the automatic implicit Euler discretization as well as the automatic linearization implemented in COMANDO are employed to obtain a MILP formulation, and a simple user-defined algorithm for multi-objective optimization is demonstrated.

In the second case study, the operation of a simple building energy system is optimized, considering forecasts for electricity price and ambient temperature. The system model makes use of differential equations to describe the thermal behavior of the building, allowing to represent dynamic aspects of demand response via a MIDO problem. The interface to Pyomo.DAE (Nicholson et al., 2018) is used to apply orthogonal collocation on finite elements as an advanced time-discretization method.

The third case study is a variation of the benchmark problem from (Saelens et al., 2020), integrating low-temperature waste heat into a district heating network via heat pumps. The explicit consideration of thermal losses and temperatures at different points of the network results in a nonconvex mixed-integer quadratically-constrained quadratic programming (MIQCQP) problem. For the implementation in COMANDO, repeated structures within the system are abstracted via subsystems, allowing for re-use of the models and reducing modeling effort. A stochastic formulation considering multiple operational scenarios based on clustered historical data is solved to obtain an optimal system design.

The fourth case study is a reimplementation of our previous work (Huster et al., 2019), where the power production of an organic Rankine cycle is maximized. The detailed thermodynamic behavior of the working fluid is described via artificial neural networks (ANNs), capable of predicting fluid properties with high accuracy. The ANNs result in a highly nonconvex, but reduced-space NLP formulation that can be solved to global optimality with our inhouse solver MAiNGO (Bongartz et al., 2018).

All case studies are solved on a desktop PC with an i7-8700 CPU (3.20GHz), 32 GB RAM, running Windows 10 Enterprise LTSC.

4.1 Case study 1: Greenfield design of an industrial energy system

This case study is inspired by our previous work (Sass et al., 2020). For demonstration, we consider a simpler system, allowing only up to one component of each type. We make use of inheritance to abstract common model aspects of conversion and storage components into generic classes and then derive specialized variants that implement more specific behavior. Furthermore, we take advantage of automatic linearization and discretization routines to obtain MILP problems from the originally dynamic and nonlinear component models of Sass et al. (2020).

The industrial energy system needs to satisfy given time-dependent demands for heating, cooling, and electricity with minimal total annualized costs (TAC) and global warming impact (GWI). To satisfy these demands, multiple conversion and storage components are available in the superstructure of the system (Fig. 5). For self-containment, we briefly repeat the description of the conversion and storage components here. More detailed information can be found in Sass et al. (2020) and in the source code for this case study, available in the COMANDO Repository (2020).

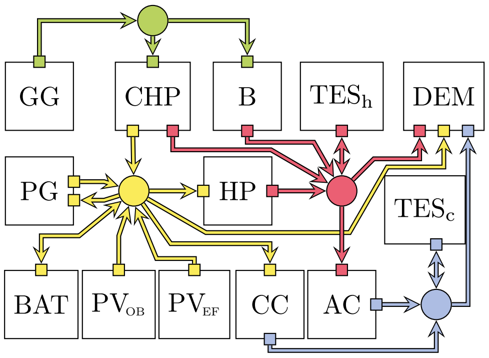


Fig. 5. Superstructure for the industrial energy system case study: gas-grid (GG), power-grid (PG), boiler (B), combined heat-and-power unit (CHP), compression chiller (CC), absorption chiller (AC), heat pump (HP), photovoltaic units on office buildings (PV$_{\mathrm{OB}}$) and on experimental facilities (PV$_{\mathrm{EF}}$), thermal energy storage for hot water (TES$_\mathrm{h}$) and cooling water (TES$_\mathrm{c}$), a battery (BAT), and a demand (DEM). Natural gas is shown in green, electricity in yellow, hot water in red, and cooling water in blue.

The conversion components $i \in \mathcal{I}^{\mathrm{conv}}\{\mathrm{AC, B, CC, CHP, HP}\}$ (cf. Fig. 5) are modeled with nonlinear investment cost and part-load efficiency curves. Additionally, minimal part-load requirements are considered by introducing binary variables. The investment cost reflect decreasing marginal investment costs C_i^{I} with increasing nominal component output $\dot{E}_i^{\mathrm{nom}}$, i.e.,

$$C_i^{\mathrm{I}} = C_i^{\mathrm{ref}} {\dot{E}_i^{\mathrm{nom}}}^{M_i} \quad \forall i \in \mathcal{I}^{\mathrm{conv}}, \tag{1}$$

where C_i^{ref} and M_i are technology-specific parameters. The part-load efficiency η_i is expressed via a base efficiency multiplied with a rational function of the part-load fraction $\dot{E}_i^{\mathrm{out}}/\dot{E}_i^{\mathrm{nom}}$, and describes the relationship of input $\dot{E}_i^{\mathrm{in}}$ and output $\dot{E}_i^{\mathrm{out}}$:

$$\dot{E}_i^{\mathrm{out}} = \eta_i \dot{E}_i^{\mathrm{in}} \quad \forall i \in \mathcal{I}^{\mathrm{conv}} \tag{2}$$

The HP and CHP models have variable base efficiencies that depend on temperatures and the nominal size, respectively. We create a generic conversion component class with an unparametrized nonlinear efficiency and investment cost model (Eqs. (1) and (2)). From this conversion component class, we derive the individual conversion technologies as subclasses. Three instances of the CHP model with different ranges for the nominal size are considered, accounting for the size-dependence of the conversion efficiencies for heat and electricity. The three CHP models are aggregated into a subsystem which enforces that at most one of them is built. The subsystem can then be incorporated into other system models like any other component.

The storage components $i \in \mathcal{I}^{\mathrm{sto}} = \{\mathrm{BAT, TES_h, TES_c}\}$ are modeled with the differential equation

$$\frac{\mathrm{d}E_i}{\mathrm{d}t} = \eta_i^{\mathrm{in}} \dot{E}_i^{\mathrm{in}} - \frac{1}{\eta_i^{\mathrm{out}}} \dot{E}_i^{\mathrm{out}} - \frac{1}{\tau_i} E_i \quad \forall i \in \mathcal{I}^{\mathrm{sto}}, \tag{3}$$

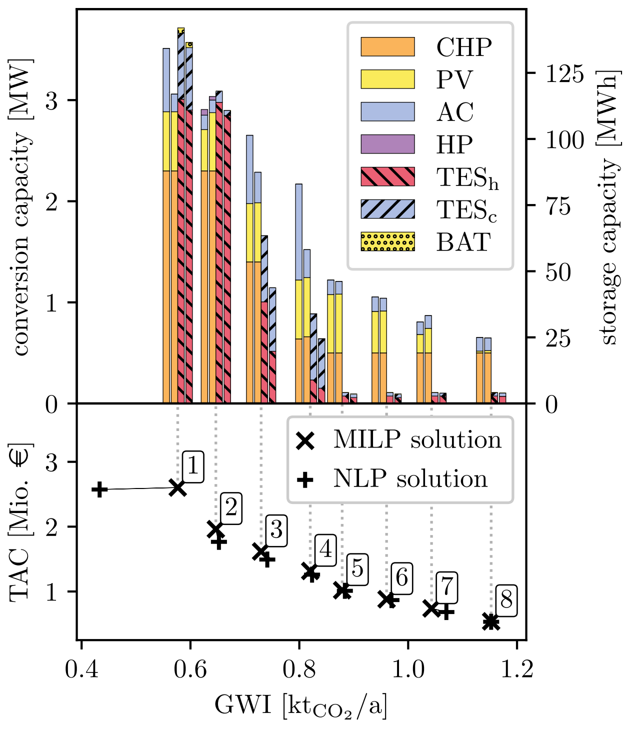


Fig. 6. Bottom: eight Pareto-optimal designs, determined from multi-objective optimization regarding total annualized cost (TAC) and global warming impact (GWI). Top: corresponding capacities of conversion (left) and storage components (right) from the MILP (inner bars) and NLP (outer bars) formulations. CHP: Combined heat and power unit, PV: photovoltaic array, AC: absorption chiller, HP: heat pump, TES_h: hot thermal energy storage, TES_c: cold thermal energy storage, BAT: battery. Note that boilers and compression chillers are not part of any design and thus excluded from the legend.

where the state E_i is the stored energy, η_i^{in} and η_i^{out} are constant charging and discharging efficiencies, $\dot{E}_i^{\text{in}}$ and $\dot{E}_i^{\text{out}}$ are the charging and discharging rates, and τ_i is a time constant describing self-discharging. As with the conversion components, we create a generic storage component class and derive technology-specific sub-classes, e.g., batteries. For each component we additionally consider a binary variable and associated constraints, representing whether the component is built or not.

We use aggregated data from the supplementary material of Sass et al. (2020), obtained via clustering of a full year of data for demands, weather, prices, and global warming impacts for electricity. The aggregated data represent the full year via four typical days, each with four time-steps of varying length, and two isolated time-points representing peak heating and cooling demands. In COMANDO we can consider such a time structure via six scenarios, corresponding to the four typical days and the two isolated time-points for peak demands. The scenarios corresponding to typical days are weighted by number of days associated to them during clustering, and the scenarios for peak demands are assigned a weight of zero, i.e., they have no effect on the objective but are considered for feasibility, cf. formulation (P).

Due to the storage dynamics Eq. (3), problems derived from this system model will be MIDO problems. In our previous work (Sass et al., 2020), we manually implemented the MILP formulation resulting from explicit Euler discretization and a case-specific linearization in GAMS. As this process and subsequent changes are labor-intensive and error-prone, we instead make use of COMANDO's automatic routines for discretization and piecewise linearization.

The augmented ε-constraint method (Mavrotas, 2009) is implemented as a user-defined algorithm, in which two design optimization problems with either TAC or GWI as objective function are repeatedly solved. For the solution of the two problems we use Gurobi 9.1.1 with a relative optimality tolerance of 1%. Generating 8 designs from the Pareto front for TAC and GWI, shown in Fig. 6, takes about 3.6

hours. Note that a Pareto-optimal design can only improve upon one of the two objectives by worsening the other.

The total GWI can be reduced by 50% (from 1.152 to 0.577 kt/a) when accepting a fourfold increase in TAC (from 0.539 to 2.6 Mio. €) (Fig. 6, bottom). Solutions with lower TAC are characterized by small component capacities with lower investment costs, whereas solutions with lower GWI rely on large conversion and storage components (Fig. 6 inner bars, top). As these results were obtained with a linearization of the original model, they are only approximate and the corrsponding designs may not be feasible with respect to the nonlinear model.

However, correcting the infeasibilities is straightforward in COMANDO as the original, nonlinear model formulation is available. We first obtain the MINLP problem resulting from implicit Euler discretization of the original formulation with TAC as the objective. We then repeat the multi-objective optimization with the same algorithm but using the MINLP formulation. For each iteration, we set the appropriate upper bound on GWI and fix binary variables to the values of the corresponding MILP solution, obtaining an NLP formulation. The values of the remaining variables are used as an initial point and the resulting formulation is passed to BARON 20.10.16 using default options, except for a relative optimality tolerance of 1% and a time limit of one hour for the subproblems.

In three cases the subproblems are terminated due to the time limit (with 3.5% relative gap for the TAC minimization of iteration 3 and 4, and 7.5% relative gap for the GWI correction of iteration 3). The remaining subproblems take at most 78 s to be solved to the desired optimality. Thus, all cases result in a design and an operational strategy that are feasible with respect to the original nonlinear formulation. The resulting solutions exhibit slightly lower TAC values and slightly higher GWI values than their MILP counterparts, with the exception of iteration 1, where the GWI value is 25% lower than for the MILP approach (433 t/a vs. 577 t/a). The corresponding designs can be seen in the outer bars in Fig. 6 (top). While the MILP and NLP solutions of iterations 2 and 5–8 are similar, iterations 1, 3 and 4 exhibit larger conversion components and smaller storages in the NLP case. In summary, the approach provides MINLP-feasible system designs that allow a trade-off between the TAC and GWI of the resulting system.

4.2 Case study 2: Demand response of a building energy system

To illustrate how to formulate and solve optimization problems with more pronounced dynamic effects in COMANDO, we model an illustrative building energy system. The system is heated by a heat pump (HP) and is capable to perform load shifting via concrete core activation, i.e., a concrete core with a high thermal inertia can be heated directly. We investigate a demand response (DR) case, where we optimize the operation of the building energy system over the horizon of one day with given profiles for electricity price and ambient temperature.

The considered building energy system consists of three thermal zones: air, outside wall, and concrete core. Occupant comfort has to be ensured by maintaining the air temperature between minimal and maximal temperatures $T_{\mathrm{air}}^{\mathrm{min}}$ and $T_{\mathrm{air}}^{\mathrm{max}}$, respectively. To do so, the air in the room can be heated via a direct heat flow to the air $\dot{Q}_{\mathrm{air,in}}$, or indirectly through the concrete core, which can be heated via the heat flow $\dot{Q}_{\mathrm{core,in}}$. We consider a zero-dimensional model of each thermal zone. For instance, the energy balance of the air zone is given by

$$\rho_{\mathrm{air}} V_{\mathrm{air}} c_{p,\mathrm{air}} \frac{\mathrm{d}T_{\mathrm{air}}}{\mathrm{d}t} = \dot{Q}_{\mathrm{core,air}} - \dot{Q}_{\mathrm{air,wall}} + \dot{Q}_{\mathrm{air,in}}, \tag{4}$$

where $T_{\mathrm{air}}, V_{\mathrm{air}}, \rho_{\mathrm{air}}$, and $c_{p,\mathrm{air}}$ are the air temperature, volume, density, and specific heat capacity, respectively, and $\dot{Q}_{\mathrm{core,air}}$ and $\dot{Q}_{\mathrm{air,wall}}$ are heat exchange flows with the adjacent zones. The heat flow $\dot{Q}_{A,B}$ between two zones A and B is calculated depending on the temperatures T_A and T_B, the area $A_{A,B}$, and the heat transfer coefficient $U_{A,B}$:

$$\dot{Q}_{A,B} = U_{A,B} A_{A,B} (T_A - T_B) \tag{5}$$

The structure of the model is shown in Fig. 7. To model thermal masses, we introduce a component M, which is instantiated by specifying volume, density, and specific heat capacity, and optionally allows to specify minimal and maximal temperatures. The heat transfer is abstracted as a component HT,

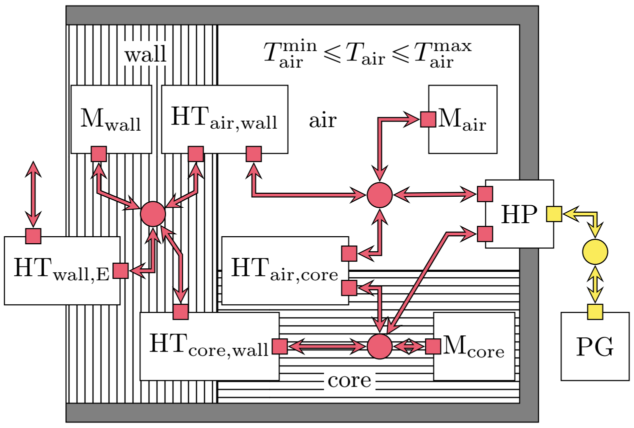

wall
$T_{\text{air}}^{\text{min}} \leqslant T_{\text{air}} \leqslant T_{\text{air}}^{\text{max}}$
M_{wall}
$\text{HT}_{\text{air,wall}}$
air
M_{air}
HP
$\text{HT}_{\text{wall,E}}$
$\text{HT}_{\text{air,core}}$
$\text{HT}_{\text{core,wall}}$
M_{core}
PG
core

Fig. 7. Structure of the considered building energy system and implementation in COMANDO: three instances of the thermal mass class (M_{air}, M_{core}, M_{wall}), four instances of the heat transfer class ($\text{HT}_{\text{air,wall}}$, $\text{HT}_{\text{wall,E}}$, $\text{HT}_{\text{air,core}}$, $\text{HT}_{\text{core,wall}}$), heat pump (HP), and power grid (PG). Red arrows represent heat flows and yellow arrows electric power flows.

implementing Eq. (5), and the heat pump is again modeled with a temperature-dependent efficiency, but with the option of splitting the output to multiple connectors.

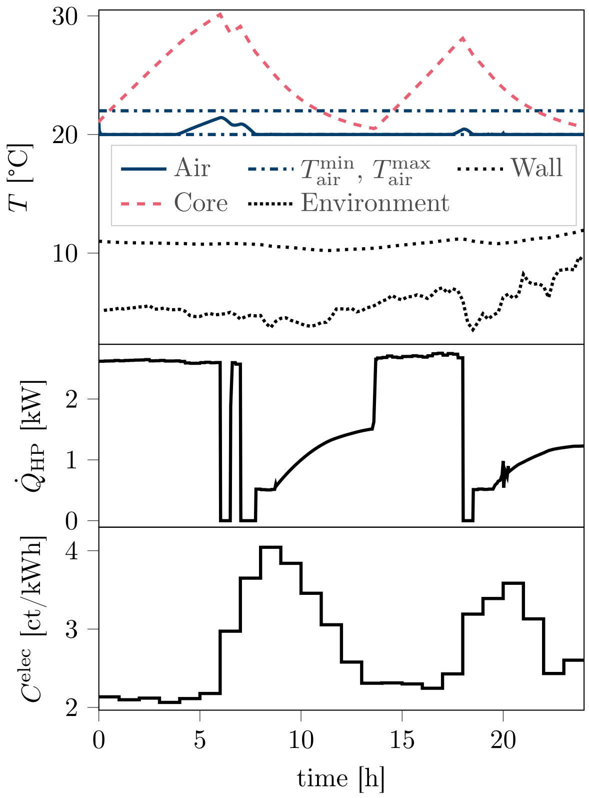


Fig. 8. Results of the demand response optimization for the building energy system: the temperatures of the different zones together with the air temperature comfort bounds $T_{\text{air}}^{\text{min}}$ and $T_{\text{air}}^{\text{max}}$ (top), the heat flow supplied by the heat pump $\dot{Q}_{\text{HP}}$ (center), and the electricity costs C^{elec} (bottom).

Based on the model of the building energy system, we define a DR optimization problem, i.e., we minimize the integral over the electricity costs for a given electricity price profile. The resulting operational objective function is thus chosen as $\dot{F}_{II} = C^{\mathrm{elec}} P_{\mathrm{HP}}$, where C^{elec} and P_{HP} are the electricity costs and electric input power of the heat pump, respectively.

As we consider a minimum part-load constraint for the heat pump, the resulting problem is a MIDO problem. The time horizon is a 24 hour period considered at quarter-hourly resolution and the input data consists of hourly electricity price data and ambient temperature data at quarter-hourly resolution. We use a full discretization approach (Cuthrell and Biegler, 1987) via the COMANDO interface to Pyomo.DAE (Nicholson et al., 2018). Specifically, we use Legendre-Radau collocation with four elements per hour and fourth-order polynomials. Since the model contains exclusively linear expressions and we use collocation with a fixed time grid, we obtain a MILP problem after discretization. The resulting formulation has 6931 constraints and 6257 variables, 96 of which are binary. The problem can be solved with Gurobi 9.1.1 to global optimality in less than one second of CPU time. Results are visualized in Fig. 8, where the temperatures of the three thermal zones, the ambient temperature, the heat flow supplied by the heat pump, and the electricity price are shown. During times of low prices, the concrete core is heated to store energy. During times of high prices, the concrete core transfers the stored heat to the air zone and cools down such that the heat pump has to supply less heat. Thus, load is shifted to times of favorable prices, while the air temperature remains within the comfort range.

Using the introduced component models for general thermal masses and heat transfers, the extension to a larger building energy system with several rooms, thermal masses, and heat transfers is straightforward.

4.3 Case Study 3: Design of a low-temperature district heating network

In this case study, we extend components of previous work (Hering et al., 2020) to describe a district heating network and apply them to a design optimization of the network described by Saelens et al. (2020). The system comprises a source of waste heat, a distribution network, and 16 consumers. We aggregate the 16 consumers into four consumer groups, comprising four consumers each, and assume linear heating curves for the flow temperature T^{fl}. The heating curves are described by the flow temperatures $T^{\mathrm{fl,max}}$ and $T^{\mathrm{fl,min}}$, at ambient air temperatures of $-12\,^{\circ}\mathrm{C}$ and $20\,^{\circ}\mathrm{C}$, respectively, see Tab. 1.

Tab. 1. Clustering of neighbouring buildings into consumer groups. Buildings within a group are assumed to have identical heating curves.

Consumer group	$T^{\mathrm{fl,max}}$ ($T_{\mathrm{air}} = -12\,^{\circ}\mathrm{C}$)	$T^{\mathrm{fl,min}}$ ($T_{\mathrm{air}} = 20\,^{\circ}\mathrm{C}$)
CG_{40}	40°C	35°C
CG_{50}	50°C	40°C
CG_{70}	70°C	50°C
CG_{85}	85°C	60°C

The source of waste heat supplies heat to a network to which each consumer group may be connected or not. Both waste heat and consumer groups are linked to the network via a heat exchanger or a heat pump, and connecting a consumer group additionally requires the necessary pipes to be built. Independently of whether a consumer group is connected or not, it may also be equipped with a gas-fired boiler or an electric heating rod. The superstructure of the heating network is shown in Fig. 9.

The system is modeled using components for a source of waste heat (WH), the distribution network (NW), the power grid (PG) and the gas grid (GG). As both the linking unit and the consumer groups are composed of multiple components and occur more than once, they are modeled as subsystems. The linking subsystem (L) contains a heat pump (HP) and a heat exchanger (HE) and the consumer group subsystem (CG) contains a linking subsystem, a demand (DEM), and two instances of a generic heat source with different parametrizations, representing a boiler (HS_{B}) and a heating rod ($\mathrm{HS}_{\mathrm{HR}}$).

The design decisions comprise binary variables for the type of linking component (heat exchanger, heat pump, or none) and the type of additional heat source (gas boiler, electric heater, or none) to be built, as well as continuous variables for component sizing and the maximum and minimum return

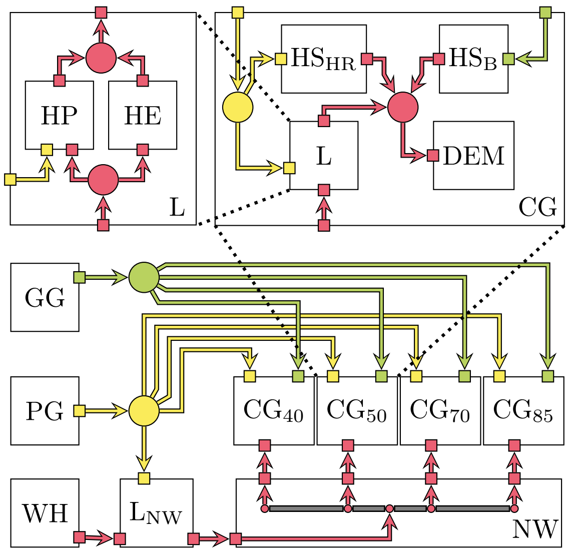


Fig. 9. Superstructure with components for the gas grid (GG), power grid (PG), waste heat source (WH) and network (NW) as well as subsystems for linking (L) and consumer groups (CG), see top. The superstructure of the linking subsystem contains a heat pump (HP) and a heat exchanger (HE) and that of the consumer group subsystems contains two heat source (HS) components parameterized as a heating rod (HS$_{\mathrm{HR}}$) and a boiler (HS$_{\mathrm{B}}$) and demand (DEM) as well as a decentral linking subsystem. To connect the different consumer groups, the necessary pipe segments (depicted as gray bars within NW) need to be built.

temperature of the network $T_{\mathrm{NW}}^{\mathrm{re,max}}$ and $T_{\mathrm{NW}}^{\mathrm{re,min}}$, respectively. Finally, four pipe segments can be added to the network model separately using the decision variables, $b_{\mathrm{NW}}^{\mathrm{s}}$. The linking components for the consumer groups can only be built if all necessary pipe segments of the network are built. The demand component has a parameter for the required heat demand and computes the required flow temperature based on the ambient air temperature. The heat demand is based on Saelens et al. (2020), while the flow temperature is assumed to depend linearly on the ambient air temperature (cf. Tab. 1). The network return temperature $T_{\mathrm{NW}}^{\mathrm{re}}$ also depends linearly on the ambient air temperature T_{air} and the design variables $T_{\mathrm{NW}}^{\mathrm{re,max}}$ and $T_{\mathrm{NW}}^{\mathrm{re,min}}$, while the network flow temperature $T_{\mathrm{NW}}^{\mathrm{fl}}$ is assumed to be 15 K higher than $T_{\mathrm{NW}}^{\mathrm{re}}$.

We aggregate the whole network into one pipe network with two branches, cf. Fig. 9. The central linking component is connected to the center of the network with $T_{\mathrm{NW}}^{\mathrm{fl}}$ and $T_{\mathrm{NW}}^{\mathrm{re}}$. Despite being located at different distances from the center, we assume that all consumer groups receive and reject water at the same flow and return temperatures, $T_{\mathrm{NW}}^{\mathrm{fl}} - \Delta T_{\mathrm{NW}}^{\mathrm{fl,loss}}$ and $T_{\mathrm{NW}}^{\mathrm{re}} + \Delta T_{\mathrm{NW}}^{\mathrm{re,loss}}$, respectively. For this simplification to be conservative, we use the total length of the network, l_{NW}, calculated as

$$l_{\mathrm{NW}} = \sum b_{\mathrm{s}} l_{\mathrm{s}}, \tag{6}$$

to calculate the temperature drops, where b_{s} is the build decision and l_{s} is the length of each network segment s, cf. Fig. 9. To obtain the temperature differences in the flow and return pipes, $\Delta T_{\mathrm{NW}}^{\mathrm{fl,loss}}$ and $\Delta T_{\mathrm{NW}}^{\mathrm{re,loss}}$, respectively, we consider energy balances of the water for both the flow (fl) and return (re) pipe of the network, i.e.,

$$\dot{m}_{\mathrm{NW}}\, c_p\, \Delta T_{\mathrm{NW}}^{\mathrm{fl,loss}} = U_{\mathrm{NW}}\, l_{\mathrm{NW}}\, (T_{\mathrm{NW}}^{\mathrm{fl}} - T_{\mathrm{gr}}) \tag{7}$$

$$\dot{m}_{\mathrm{NW}}\, c_p\, \Delta T_{\mathrm{NW}}^{\mathrm{re,loss}} = U_{\mathrm{NW}}\, l_{\mathrm{NW}}\, (T_{\mathrm{NW}}^{\mathrm{re}} + \Delta T_{\mathrm{NW}}^{\mathrm{re,loss}} - T_{\mathrm{gr}}) \tag{8}$$

where $\Delta T_{\mathrm{NW}}^{\mathrm{fl,loss}}$ and $\Delta T_{\mathrm{NW}}^{\mathrm{re,loss}}$ are operational variables describing the temperature drop in the respective pipe, c_p is the constant specific heat capacity of water, $U_{\mathrm{NW}} = 0.035 \frac{\mathrm{W}}{\mathrm{mK}}$ is the specific heat transfer coefficient and l_{NW} is the pipe network length, and $T_{\mathrm{gr}} = 8$°C is the average ground temperature.

The heat pump model in each linking component is modeled via the following set of equations:

$$\dot{Q}_{\mathrm{HP}} \leqslant b_{\mathrm{HP}}\, 400\,\mathrm{kW} \tag{9}$$

$$P_{\mathrm{HP}}\, T_{\mathrm{con}}^{\mathrm{re}}\, \eta_{\mathrm{COP}} = \dot{Q}_{\mathrm{HP}} (T_{\mathrm{con}}^{\mathrm{re}} - T_{\mathrm{eva}}^{\mathrm{re}}) \tag{10}$$

$$\begin{aligned} &\dot{m}_{\mathrm{eva}} c_p\, (T_{\mathrm{eva}}^{\mathrm{fl}} - T_{\mathrm{eva}}^{\mathrm{re}}) + P_{\mathrm{HP}} \\ &\quad = \dot{m}_{\mathrm{con}} c_p\, (T_{\mathrm{con}}^{\mathrm{re}} - T_{\mathrm{con}}^{\mathrm{fl}}) \end{aligned} \tag{11}$$

Here, $\dot{Q}_{\mathrm{HP}}$, P_{HP}, $\dot{m}_{\mathrm{eva}}$, $\dot{m}_{\mathrm{con}}$, $T_{\mathrm{eva}}^{\mathrm{fl}}$, $T_{\mathrm{eva}}^{\mathrm{fl}}$, $T_{\mathrm{con}}^{\mathrm{re}}$ and $T_{\mathrm{con}}^{\mathrm{fl}}$ are operational variables, and $\eta_{\mathrm{COP}} = 0.6$ is the heat pump efficiency relative to the carnot efficiency. The outgoing heat flow for the heat pump ($\dot{Q}_{\mathrm{HP}}$) is bounded by zero or the maximum allowable nominal size of 400 kW through Eq. (9). The input power P_{HP} is coupled to $\dot{Q}_{\mathrm{HP}}$ via Eq. (10). In the energy balance Eq. (11), enthalpy differences at the evaporator and condenser side are described by the associated mass flows $\dot{m}_{\mathrm{eva}}$ and $\dot{m}_{\mathrm{con}}$, and flow and return temperatures $T_{\mathrm{eva}}^{\mathrm{fl}}$, $T_{\mathrm{eva}}^{\mathrm{fl}}$, $T_{\mathrm{con}}^{\mathrm{fl}}$ and $T_{\mathrm{con}}^{\mathrm{re}}$.

For the investment cost, we assume linear cost correlations with a specific cost $\mathrm{c}_{\mathrm{spec}}$ and a fixed cost $\mathrm{c}_{\mathrm{fix}}$ according to Tab. 2.

Tab. 2. Specific and fixed costs for heating equipment according to (BMVBS, 2012) and (BBSR, 2014)

Component	$\mathrm{c}_{\mathrm{spec}}$	$\mathrm{c}_{\mathrm{fix}}$
central HP	500 €/kW	0 €
decentral HP	620 €/kW	0 €
HE	90 €/kW	0 €
$\mathrm{HS}_{\mathrm{HR}}$	10 €/kW	100 €
HS_{B}	111 €/kW	4300 €

Additionally, we consider the costs for each pipe segment of the network based on Jentsch et al. (2008). Thus, the total investment costs of the system includes the investments into heating components and piping.

To obtain an economical design, we minimize TAC. We use k-means clustering (Pedregosa et al., 2011) to aggregate the original set of ambient temperatures and heat demands into representative clusters. Each resulting cluster center is a pair of daily mean values for temperature and heat demand and can be considered as a representative operating scenario. To reduce computational demand, the data is clustered into 11 such scenarios, including one scenario representing the maximum heat demand. Demand data with zero heat demand are dropped from the dataset. Fig. 10 shows the resulting 11 clusters.

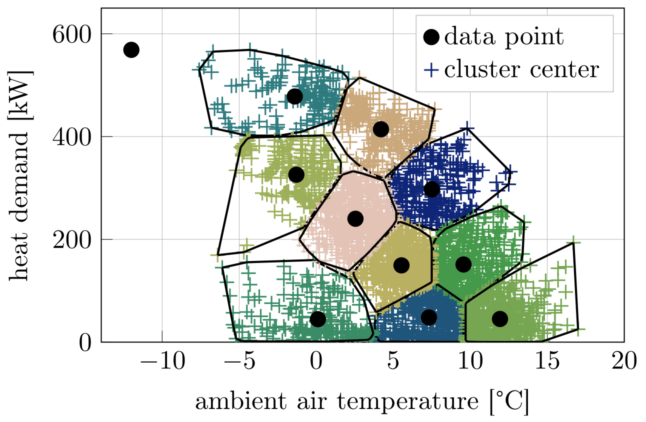


Fig. 10. Heat demand clusters: Each cross represents one pair of measurements of total daily mean heat demand and daily mean ambient air temperature. Colors and boundaries are used to aid visual distinction of the clusters whose centers are mean values depicted as black dots.

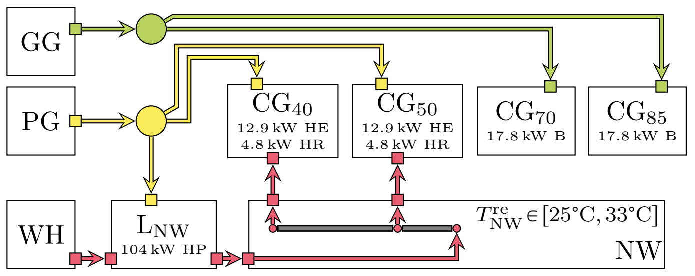


Fig. 11. Optimal system structure: A central heat pump HP supplies waste heat from WH to the network NW. Consumer groups CG_{40} and CG_{50} are connected to NW via heat exchangers (HE) and use heating rods (HR) for peak demands. Consumer groups CG_{70} and CG_{85} are not connected and satisfy their heat demand via boilers (B).

We use the clusters as scenarios in the COMANDO framework, with the fraction of data points in each cluster as the corresponding scenario weight. Considering the data in this way ensures that the final design is feasible for all considered scenarios and is optimized with regards to the expected value of TAC. The resulting problem is a MIQCQP with 526 continuous variables, 147 binary variables and 275 quadratic constraints. An optimal design is obtained within three minutes of CPU time and a 1% optimality gap, using the Gurobi API interface with Gurobi 9.1.1. The global optimal solution corresponds to the system shown in Fig. 11.

The network is designed with a variable return temperature between 25°C and 33°C and is connected to the waste heat source using a 104 kW heat pump. Consumer groups CG_{40} and CG_{50} are connected to the network using heat exchangers and have additional electric heating rods installed. Consumer groups CG_{70} and CG_{85} are not connected but satisfy their heat demand using gas-fired boilers instead. The TAC of this design are 22 193 €. At an annual heat demand of 322 MWh this corresponds to a specific heating cost of 68.77 €/MWh.

4.4 Case study 4: Optimal operating point of an organic Rankine cycle (ORC)

Finally, we consider a case study from our previous work (Huster et al., 2019), where an optimal operating point of an organic Rankine cycle (ORC) with respect to net power production is sought. With this case study, we demonstrate how COMANDO can handle complex modeling features such as accurate fluid properties via artificial neural networks (Schweidtmann and Mitsos, 2018; Schweidtmann et al., 2019) and a sequential modeling approach that gives rise to reduced-space formulations beneficial for global optimization (Bongartz and Mitsos, 2017).

Again, we give a short overview of the case study for self-containment. In the considered process, the working fluid isobutane (ib) is first pressurized by a pump and then preheated in a recuperator before being heated to evaporation temperature, evaporated and superheated by cooling geothermal brine (gb) from 408 K to 357 K. After expanding in a turbine, the working fluid is used in the recuperator to preheat the pressurized fluid and is finally condensed and cooled to its original state using cooling water at 288 K. The heat passed from the condenser to the cooling water (cw) is dissipated by a cooling system consisting of multiple fans.

The ORC is modeled as a system consisting of 4 types of components, i.e., a pump (P), a turbine (T), a cooling system (CS), and five heat exchangers (condenser HE_{con}, recuperator HE_{rec}, economizer HE_{eco}, evaporator HE_{eva}, and superheater HE_{sup}). All components have connectors for enthalpy in- and out-flows that are connected as depicted in Fig. 12 to obtain the system model.

As discussed in Bongartz and Mitsos (2017), reduced-space formulations, i.e., formulations in which a large number of variables and constraints are eliminated by substitution, are well suited for global optimization of power cycles such as the present ORC. To obtain a reduced-space formulation, model generation begins with an empty system model to which different component models are added sequentially. First, the decision variables are specified at the system level as follows: The mass flow $\dot{m}$ of the

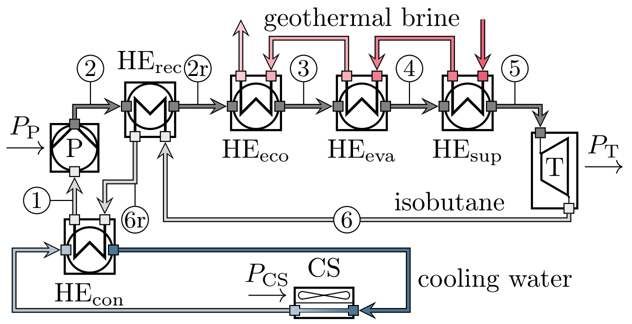


Fig. 12. System model of the ORC process from Huster et al. (2019). The components are a pump (P), a recuperator ($\mathrm{HE_{rec}}$), an economizer ($\mathrm{HE_{eco}}$), an evaporator ($\mathrm{HE_{eva}}$), a superheater ($\mathrm{HE_{sup}}$), a turbine (T), a condenser ($\mathrm{HE_{con}}$), and a cooling system (CS). Flows of geothermal brine, the working fluid isobutane, and cooling water are depicted in red, gray, and blue, respectively. Electrical power is consumed by pump (P_P) and cooling system (P_CS) and produced by the turbine (P_T).

working fluid, the pressures p_1 and p_2 before and after the pump, and the specific enthalpy after the recuperator $h_{2\mathrm{r}}$, as well as the isentropic specific enthalpy after the turbine h_6^{is}. All other quantities of interest are defined in terms of these five variables.

In our previous work (Schweidtmann and Mitsos, 2018; Schweidtmann et al., 2019), the use of artificial neural networks (ANNs) in combination with our inhouse global MINLP solver MAiNGO (Bongartz et al., 2018) has been shown to result in tight relaxations, beneficial for deterministic global optimization. In Huster et al. (2019), we trained several ANNs to learn the relations between various quantities of different thermodynamic phases of the working fluid isubutane using data from the thermophysical property library CoolProp (Bell et al., 2014). Each ANN expresses one output quantity in terms of either pressure p, pressure and specific enthalpy h, or pressure and specific entropy s, as inputs. As a result of training, we thus obtain explicit analytical expressions for various quantities. In this case study, eight of the ANNs from Huster et al. (2019) are used as analytical surrogate models for the following quantities:

$h^{\mathrm{liq}}(p,s)$	liquid enthalpy
$T^{\mathrm{liq}}(p,h)$	liquid temperature
$h^{\mathrm{sat,liq}}(p)$	enthalpy of saturated liquid
$s^{\mathrm{sat,liq}}(p)$	entropy of saturated liquid
$T^{\mathrm{sat}}(p)$	saturation temperature
$h^{\mathrm{sat,vap}}(p)$	enthalpy of saturated vapor
$s^{\mathrm{vap}}(p,h)$	vapor entropy
$T^{\mathrm{vap}}(p,h)$	vapor temperature

The enthalpy flows of pump and turbine are described via mass flow and specific enthalpies, and the electrical power consumed by the pump (P_P) and provided by the turbine (P_T) are modeled as

$$P_\mathrm{P} = \dot{m}\,\frac{h_\mathrm{P}^{\mathrm{is,out}} - h_\mathrm{P}^{\mathrm{in}}}{\eta_\mathrm{P}^{\mathrm{is}}}, \tag{12}$$

$$P_\mathrm{T} = \dot{m}\,(h_\mathrm{T}^{\mathrm{in}} - h_\mathrm{T}^{\mathrm{is,out}})\,\eta_\mathrm{T}^{\mathrm{is}}, \tag{13}$$

where $\eta_\mathrm{P}^{\mathrm{is}}$ and $\eta_\mathrm{T}^{\mathrm{is}}$ are known, constant isentropic efficiencies and the required specific enthalpies h are determined via the appropriate ANNs.

For each heat exchanger, the differences of enthalpy flows at the hot (h) and cold (c) side are either defined in terms of a mass flow and specific enthalpies (ib) or in terms of a specific heat capacity flow

$\dot{m}c_p$ and temperatures (cw and gb):

$$\dot{Q}_\mathrm{h} = \begin{cases} \dot{m}_\mathrm{h}\,(h_\mathrm{h}^\mathrm{in} - h_\mathrm{h}^\mathrm{out}), & \mathrm{h} = \mathrm{ib} \\ (\dot{m}c_p)_\mathrm{h}\,(T_\mathrm{h}^\mathrm{in} - T_\mathrm{h}^\mathrm{out}), & \mathrm{h} \in \{\mathrm{cw},\, \mathrm{gb}\} \end{cases} \tag{14}$$

$$\dot{Q}_\mathrm{c} = \begin{cases} \dot{m}_\mathrm{c}\,(h_\mathrm{c}^\mathrm{out} - h_\mathrm{c}^\mathrm{in}), & \mathrm{c} = \mathrm{ib} \\ (\dot{m}c_p)_\mathrm{c}\,(T_\mathrm{c}^\mathrm{out} - T_\mathrm{c}^\mathrm{in}), & \mathrm{c} \in \{\mathrm{cw},\, \mathrm{gb}\} \end{cases} \tag{15}$$

As heat losses are neglected, the energy balance reduces to

$$\dot{Q}_\mathrm{h} = \dot{Q}_\mathrm{c}. \tag{16}$$

Since we aim for a reduced-space formulation, no variables are introduced for the left-hand sides of Eqs. (12)–(15) and the corresponding right-hand side expressions are used directly, avoiding the addition of constraints. In particular, where possible, Eq. (16) is automatically reformulated to obtain a definition for one of the temperatures or specific enthalpies in the right-hand sides of Eqs. (14) and (15) in terms of the other quantities. The heat-exchanger model is configured to perform the appropriate reformulation automatically, based on the provided quantities.

A pinch point is assumed in the condenser, i.e., the temperature of the cooling water at the pinch point, T_pinch, is assumed to lie $\Delta T_\mathrm{min} = 10\,\mathrm{K}$ below the evaporation temperature $T^\mathrm{sat}(p_1)$. Through this assumption, it is possible to compute the heat capacity flow of the cooling water, $(\dot{m}c_p)_\mathrm{cw}$, as

$$\begin{aligned} (\dot{m}c_p)_\mathrm{cw} &= \frac{\dot{m}\,(h_\mathrm{pinch} - h_1)}{\max(10^{-5}\,\mathrm{K},\ T_\mathrm{pinch} - T_\mathrm{cw}^\mathrm{in})} \\ &= \frac{\dot{m}\,\left(h^\mathrm{sat,vap}(p_1) - h^\mathrm{sat,liq}(p_1)\right)}{\max\left(10^{-5}\,\mathrm{K},\ T^\mathrm{sat}(p_1) - 10\,\mathrm{K} - 288\,\mathrm{K}\right)}. \end{aligned} \tag{17}$$

Note that the max function and the constant 10^{-5} in Eq. (17) are introduced to avoid division by zero. The electrical power P_CS, required to run the fans of the cooling system, is modeled to be proportional to the specific heat capacity flow of the air $(\dot{m}c_p)_\mathrm{air}$ passing through them and is computed as

$$\begin{aligned} P_\mathrm{CS} &= \frac{\dot{V}_\mathrm{air}\,\Delta p_\mathrm{fan}}{\eta_\mathrm{fan}} \\ &= \frac{(\dot{m}c_p)_\mathrm{air}\,\Delta p_\mathrm{fan}}{c_{p,\mathrm{air}}\,\rho_\mathrm{air}\,\eta_\mathrm{fan}}, \end{aligned} \tag{18}$$

where $\Delta p_\mathrm{fan} = 170\,\mathrm{Pa}$ and $\eta_\mathrm{fan} = 0.65$ are the pressure drop and efficiency of the fan, $\dot{V}_\mathrm{air}$, $c_{p,\mathrm{air}} = 1000\,\frac{\mathrm{J}}{\mathrm{kg\,K}}$ and $\rho_\mathrm{air} = 1.2\,\frac{\mathrm{kg}}{\mathrm{m}^3}$ are the volume flow, specific heat capacity and density of the air, respectively. With the assumption that

$$(\dot{m}c_p)_\mathrm{air} = (\dot{m}c_p)_\mathrm{cw}, \tag{19}$$

the power of the cooling system is fully determined. For the complete formulation, the reader is referred to the model source code.

The reduced space formulation results in a system model with relatively few expressions, however, since several quantities that are described by ANNs are themselves inputs to other ANNs or used in reformulations within the heat exchangers, the model expressions become deeply nested. For this particular use case, the standard SymPy backend (implemented in pure Python) proved to be inefficient as model generation takes about 45 minutes. Therefore, SymEngine (Čertík et al., 2019), a C++ implementation of a subset of SymPy, was integrated as an alternative backend for COMANDO. Although SymEngine has a reduced feature set compared to SymPy, all functionality relevant for the presented case study is provided. The use of SymEngine reduces the model generation time to about 0.1 seconds. Nevertheless, the nested expressions in the model result in very large input files that can take substantial time when written to disk. For instance, when using only a single scenario and operating point and maximizing the net power production

$$P_\mathrm{net} = P_\mathrm{T} - P_\mathrm{P} - P_\mathrm{CS}, \tag{20}$$

the resulting optimization problem has only 5 variables and 32 constraints.

In order to solve this problem with BARON (Sahinidis, 2020), the nonsmooth max function in Eq. (17) is approximated with $\max(a, b) \approx 0.5(a + b + [(a - b + 10^{-4})^2]^{0.5})$ and the $\tanh(x)$ function present in the ANNs is equivalently expressed as $1 - 2/[\exp(2x) + 1]$. Generating the BARON input file takes around 1 minute and results in a file size of about 40 MB. This input file is passed to BARON 20.10.16 with absolute and relative optimality tolerances set to 1e-3. BARON reports finding a feasible solution with an objective value of $P_{\mathrm{net}} = 16.48$ MW during preprocessing and terminates after the first iteration and 8 s of CPU time. Although a lower bound within the optimality tolerance is given in the log file, BARON states that it cannot guarantee global optimality due to missing bounds for certain nonlinear subexpressions.

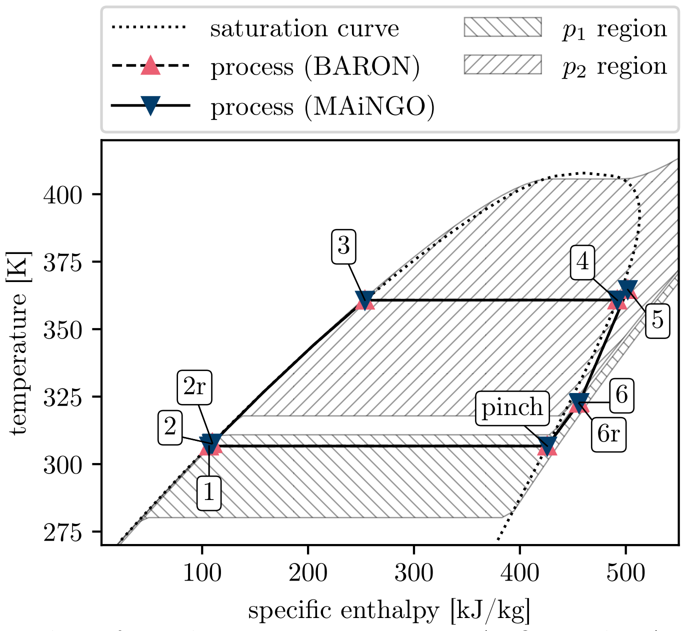


Fig. 13. Processes resulting from the optimization using BARON and MAiNGO and boundaries of pressure variables p_1 and p_2.

To prove the global optimality of this solution, we use the COMANDO interface to the API of our inhouse solver MAiNGO (Bongartz et al., 2018). MAiNGO automatically provides relaxations of the nested expressions by propagating McCormick relaxations through subexpressions (Mitsos et al., 2009). The COMANDO interface uses a SymEngine implementation of *common subexpression elimination* to find subexpressions that occur more than once within the problem description. By creating intermediate variables and replacing all occurrences of these subexpressions, a small (21 kB) input file for MAiNGO can be created. Since MAiNGO is capable of propagating McCormick relaxations, the user does not need to provide bounds on these intermediate variables and they are not treated as decision variables, maintaining the reduced-space formulation. Solving the resulting problem via MAiNGO version 0.3 with the solution returned by BARON as an initial point takes 22 s and confirms its global optimality (see Fig. 13), matching the results reported in Huster et al. (2019).

5 Conclusion

We present COMANDO, our flexible open-source framework for component-oriented modeling and optimization for nonlinear design and operation of energy systems. COMANDO combines desirable features of existing tools and provides layers of abstraction suitable for structured model generation and flexible problem formulation. The behavior of individual components can be represented with detailed models, including dynamic and nonlinear effects based on mechanistic, data-driven or hybrid modeling approaches.

The component models are then aggregated to energy system models, based on which different optimization problems concerning the design and/or operation of the energy system can be formulated. COMANDO natively allows to consider multiple operating scenarios via stochastic programming formulations, allowing to find system designs that are suitable for operation under uncertainty. The resulting problem formulations can either be manipulated in user-defined algorithms, or be passed to algebraic modeling languages or directly to solvers.

COMANDO allows for flexible model creation beyond the capabilities of existing MILP-based energy-system modeling tools and provides a wide range of options for problem formulation. Contrary to classical algebraic modeling frameworks, it allows for modular component and system representations, and is dedicated to energy system design and operation.

In four case studies, we demonstrate how COMANDO can be used to create modular and reusable component and system models of various types of energy systems. Further, we formulate and solve associated optimization problems. With COMANDO, we facilitate and enhance workflows of computer-based analysis of future integrated energy systems. We plan to continuously improve and expand COMANDO's capabilities, with future versions being published via the COMANDO Repository (2020).

Author Contribution

- ML developed COMANDO, and wrote Sections 1–3, 4.4 and 5 in close collaboration with MD and help & guidance from AM.
- DS and ML contributed the code for automatic linearization, the first case study and wrote Section 4.1 with help and guidance from MD and AB.
- FB incorporated Pyomo.DAE into the Pyomo interface, contributed the second case study and wrote Section 4.2 with help and guidance from ML, MD and AB.
- DH and ML contributed the third case study and wrote Section 4.3 with help and guidance from MD, AX and DM.
- UB and MD gave conceptual input for the creation of COMANDO.
- MD supervised the writing process.
- All authors reviewed and edited the manuscript

Declaration of Competing Interest

We have no conflict of interest.

Acknowledgements

We would like to thank Alexander Holtwerth (Forschungszentrum Jülich GmbH, Institute of Energy and Climate Research, Energy Systems Engineering (IEK-10)) for providing an initial version of the Gurobi interface. This work was funded by the Helmholtz Association of German Research Centers through program-oriented funding, the Joint Initiative "Energy System 2050 – A Contribution of the Research Field Energy", and the Initiative "Energy System Integration".

Nomenclature

Acronyms

AML	algebraic modeling language
ANN	artificial neural network
API	application programming interface
COP	coefficient of performance
DAMF	differential-algebraic modeling framework
ESMF	energy system modeling framework
GWI	global warming impact
LP	linear programming
MIDO	mixed-integer dynamic optimization
MILP	mixed-integer linear programming
MINLP	mixed-integer nonlinear programming
MIQCQP	mixed-integer quadratically constrained quadratic programming
NLP	nonlinear programming
ORC	organic Rankine cycle
TAC	total annualized costs

Component labels

AC	absorption chiller
B	boiler
BAT	battery
CG	consumer group subsystem
CC	compression chiller
CHP	combined heat-and-power unit
CS	cooling system
DEM	demand
GG	gas grid
HE	heat exchanger
HP	heat pump
HR	heating rod
HS	heat source
L	linking subsystem
NW	network
P	pump
PG	power grid
PV	photovoltaic unit
T	turbine
TES	thermal energy storage
WH	waste heat

Latin symbols

A	contact area [m^2]
b	build decision (1: build, 0: do not build)
c	generic connector expression
c_p	heat capacity [J/kg/K]
C	cost [€]
$\boldsymbol{e}$	generic algebraic expression
$E, \dot{E}$	generalized energy, energy flow [J], [W]
F	generic objective function
$\boldsymbol{g}$	left-hand side of generic inequality constraints
h	specific enthalpy [J/kg]
$\boldsymbol{h}$	left-hand side of generic equality constraints
$\mathcal{I}$	set of components
$\dot{m}$	mass flow rate [kg/s]
M	investment cost exponent
p	pressure [Pa]
$\boldsymbol{p}$	generic parameters
P	electric power [W]
$\dot{Q}$	heat transfer rate [W]
s	specific entropy [W/kg/K]
t	time point
T	temperature [K]
U	heat transfer coefficient [W/m^2/K]
$V, \dot{V}$	volume, volumetric flow [m^3], [m^3/s]
$\boldsymbol{x}$	generic design variables
$\boldsymbol{y}$	generic operational variables
$\mathcal{T}$	set of all considered time-points
$\mathcal{X}$	host-set of generic design variables
$\mathcal{Y}$	host-set of generic operational variables
w	scenario weight

Greek symbols

$\Delta_{s,t}$	time-step [h]
ΔT	temperature difference [K]
η	efficiency
ρ	density [kg/m^3]
τ	self-discharge of storage component [h]

Subscripts

0	initial point
1, 2, 2r, 3, 4, 5, 6, 6r, pinch	working fluid states in the fourth case study
40, 50, 70, 85	design temperatures in the third case study
A,B	thermal zones A and B in the second case study
c	cool
con	condenser
core	concrete core
cw	cooling water
eva	evaporator
eco	economizer
gb	geothermal brine
h	hot
i	generic system component
ib	isobutane
I, II	first- and second-stage quantities
rec	recuperator
s	scenario
sup	superheater

Superscripts

conv	conversion components
d	differential states
elec	electricity
fl	flow
gr	ground
I	investment
in	input, in-flowing stream
is	isentropic
liq	liquid
max	maximum value
min	minimum value
nom	nominal value
out	output, out-flowing stream
re	return
ref	reference value
sat	saturation
sto	storage components
vap	vapor

Bibliography

Åkesson, J., Årzén, K.-E., Gäfvert, M., Bergdahl, T., and Tummescheit, H. (2010). Modeling and optimization with Optimica and JModelica.org - Languages and tools for solving large-scale dynamic optimization problems. *Comput. Chem. Eng.*, 34(11):1737–1749.

Andiappan, V. (2017). State-Of-The-Art Review of Mathematical Optimisation Approaches for Synthesis of Energy Systems. *Process Integr. Optim. Sustain.*, 1(3):165–188.

Atabay, D. (2017). An open-source model for optimal design and operation of industrial energy systems. *Energy*, 121:803–821.

Bakken, B. H., Skjelbred, H. I., and Wolfgang, O. (2007). eTransport: Investment planning in energy supply systems with multiple energy carriers. *Energy*, 32(9):1676–1689.

Bansal, V., Perkins, J. D., Pistikopoulos, E. N., Ross, R., and van Schijndel, J. M. G. (2000). Simultaneous design and control optimisation under uncertainty. *Comput. Chem. Eng.*, 24(2-7):261–266.

Baumgärtner, N., Bahl, B., Hennen, M., and Bardow, A. (2019a). RiSES3: Rigorous Synthesis of Energy Supply and Storage Systems via time-series relaxation and aggregation. *Comput. Chem. Eng.*, 127:127–139.

Baumgärtner, N., Temme, F., Bahl, B., Hennen, M., Hollermann, D., and Bardow, A. (2019b). RiSES4: Rigorous Synthesis of Energy Supply Systems with Seasonal Storage by relaxation and time-series aggregation to typical periods. In *Proc. ECOS 2019*, pages 263–274, Wrocław, Poland.

Bell, I. H., Wronski, J., Quoilin, S., and Lemort, V. (2014). Pure and Pseudo-pure Fluid Thermophysical Property Evaluation and the Open-Source Thermophysical Property Library CoolProp. *Ind. Eng. Chem. Res.*, 53(6):2498–2508.

Beller, M. (1976). Reference energy system methodology. Technical report, Brookhaven National Lab., Upton, NY (USA).

Biel, M. and Johansson, M. (2019). Efficient Stochastic Programming in Julia. *arXiv preprint arXiv:1909.10451v3*.

Birge, J. R. and Louveaux, F. (2011). *Introduction to stochastic programming.* Springer Science & Business Media.

Bisschop, J. (2006). *AIMMS optimization modeling.* Lulu.com.

Bongartz, D. and Mitsos, A. (2017). Deterministic global optimization of process flowsheets in a reduced space using McCormick relaxations. *J. Global Optim.*, 69(4):761–796.

Bongartz, D., Najman, J., Sass, S., and Mitsos, A. (2018). MAiNGO: McCormick based Algorithm for mixed integer Nonlinear Global Optimization. Technical report, Process Systems Engineering (AVT. SVT), RWTH Aachen University.

Brown, T., Hörsch, J., and Schlachtberger, D. (2018). PyPSA: Python for Power System Analysis. *J. Open Res. Softw.*, 6(1):4.

Bundesinstitut für Bau-, Stadt- und Raumforschung (BBSR) (2014). Kosten energierelevanter Bau- und technischer Anlagenteile bei der energetischen Sanierung von Nichtwohngebäuden/ Bundesliegenschaften. BBSR-Online-Publikation, Nr. 06/2014.

Bundesministerium für Verkehr, Bau und Stadtentwicklung (BMVBS) (2012). Ermittlung von spezifischen Kosten energiesparender Bauteil-, Beleuchtungs-, Heizungs- und Klimatechnikausführungen bei Nichtwohngebäuden für die Wirtschaftlichkeitsuntersuchungen zur EnEV 2012. BMVBS-Online-Publikation, Nr. 08/2012.

Bussieck, M. R. and Meeraus, A. (2004). General algebraic modeling system (GAMS). In *Modeling languages in mathematical optimization*, pages 137–157. Springer.

Caspari, A., Bremen, A. M., Faust, J. M. M., Jung, F., Kappatou, C. D., Sass, S., Vaupel, Y., Hannemann-Tamás, R., Mhamdi, A., and Mitsos, A. (2019). DyOS - A Framework for Optimization of Large-Scale Differential Algebraic Equation Systems. *Comput. Aided Chem. Eng.*, 46:619–624.

Čertík, O., Peterson, D. L., Rathnayake, T. B., Dembia, C., Rioux, J., Hiray, S., Hisch, T., Steinberg, V., Fernando, I., Brady, P., Vats, S., Kulal, S., Rasnayaka, S., Meher, A., Sahai, G., Kumar, A., Biscani, F., Behan, C., Dahlgren, B., Stephan, R., Mandre, I., Agarwal, A., Trehan, A., Garg, S., Siwach, A., Prakash, P., malayaleecoder, Nikhil, N., Yuning, Z., Chen, C., Luszczak, M., Lui, I., Vidanaarachchi, R., Singh, K., Luo, V., Stojic, J., Parsoya, A., Kumar, R., Jaiswal, S., Sidana, V., Bhat, S., He, T., Mills, C., Pelteret, J.-P., Kumar, R., Manohar, K., Ruwanpathirana, K., Saroad, M., Reusch, D., Ansmann, G., Ma, J., Pochhi, N., Gupta, E., Yan, Z., Humenberger, A., Flowing, C., Corlay, S., Kaempen, K., Hu, A., Singh, R. R., Bonazzi, F., Stelter, S., Bocklund, B., Mansueto, M., and Lee, S. (2019). symengine 0.4.0. https://github.com/symengine/symengine (accessed February 02 2021).

COMANDO Repository (2020). https://jugit.fz-juelich.de/iek-10/public/optimization/comando (accessed February 02 2021).

Connolly, D., Lund, H., Mathiesen, B. V., and Leahy, M. (2010). A review of computer tools for analysing the integration of renewable energy into various energy systems. *Appl. Energy*, 87(4):1059–1082.

Cuthrell, J. E. and Biegler, L. T. (1987). On the optimization of differential-algebraic process systems. *AIChE J.*, 33(8):1257–1270.

Dantzig, G. B. (1955). Linear programming under uncertainty. *Manage. Sci.*, 1(3-4):197–206.

Demirhan, C. D., Tso, W. W., Ogumerem, G. S., and Pistikopoulos, E. N. (2019). Energy systems engineering - a guided tour. *BMC Chem. Eng.*, 1(1):11.

Dorfner, J. (2016). *Open source modelling and optimisation of energy infrastructure at urban scale.* PhD thesis, Technical University of Munich.

Dunning, I., Huchette, J., and Lubin, M. (2017). JuMP: A Modeling Language for Mathematical Optimization. *SIAM Review*, 59(2):295–320.

Elmqvist, H. and Mattsson, S.-E. (1997). Modelica-the next generation modeling language-an international design effort. In *Proc. 1st World Congr. Syst. Simul.*, pages 1–3, Singapore.

Ferris, M. C., Dirkse, S. P., Jagla, J.-H., and Meeraus, A. (2009). An extended mathematical programming framework. *Comput. Chem. Eng.*, 33(12):1973–1982.

Fishbone, L. G. and Abilock, H. (1981). Markal, a linear-programming model for energy systems analysis: Technical description of the bnl version. *Int. J. Energy Res.*, 5(4):353–375.

Fourer, R., Gay, D. M., and Kernighan, B. W. (1990). A Modeling Language for Mathematical Programming. *Manage. Sci.*, 36(5):519–554.

Frangopoulos, C., Von Spakovsky, M., and Sciubba, E. (2002). A Brief Review of Methods for the Design and Synthesis Optimization of Energy Systems. *Int. J. Thermodyn.*, 5:151–160.

Frangopoulos, C. A. (2018). Recent developments and trends in optimization of energy systems. *Energy*, 164:1011–1020.

Friedman, Z., Ingalls, J., Siirola, J. D., and Watson, J.-P. (2013). Block-oriented modeling of superstructure optimization problems. *Comput. Chem. Eng.*, 57:10–23.

Gabrielli, P., Gazzani, M., Martelli, E., and Mazzotti, M. (2018). Optimal design of multi-energy systems with seasonal storage. *Appl. Energy*, 219:408–424.

Ghobeity, A. and Mitsos, A. (2012). Optimal design and operation of a solar energy receiver and storage. *J. Sol. Energy Eng.*, 134(3).

Goderbauer, S., Bahl, B., Voll, P., Lübbecke, M. E., Bardow, A., and Koster, A. M. C. A. (2016). An adaptive discretization MINLP algorithm for optimal synthesis of decentralized energy supply systems. *Comput. Chem. Eng.*, 95:38–48.

Gunasekaran, S., Mancini, N. D., and Mitsos, A. (2014). Optimal design and operation of membrane-based oxy-combustion power plants. *Energy*, 70:338–354.

Gurobi Optimization, LLC (2020). Gurobi Optimizer Reference Manual. http://www.gurobi.com (accessed February 02 2021).

Hart, W. E., Watson, J.-P., and Woodruff, D. L. (2011). Pyomo: modeling and solving mathematical programs in Python. *Math. Program. Comput.*, 3(3):219–260.

Hering, D., Xhonneux, A., and Müller, D. (2020). Design optimization of a heating network with multiple heat pumps as mixed integer quadratically constrained program. In *Proc. ECOS 2020*, pages 1745–1755, Osaka, Japan.

Hilpert, S., Kaldemeyer, C., Krien, U., Günther, S., Wingenbach, C., and Plessmann, G. (2018). The Open Energy Modelling Framework (oemof) - A new approach to facilitate open science in energy system modelling. *Energy Strategy Rev.*, 22:16–25.

Howells, M., Rogner, H., Strachan, N., Heaps, C., Huntington, H., Kypreos, S., Hughes, A., Silveira, S., DeCarolis, J., Bazillian, M., et al. (2011). OSeMOSYS: the open source energy modeling system: an introduction to its ethos, structure and development. *Energy Policy*, 39(10):5850–5870.

Huchette, J., Lubin, M., and Petra, C. (2014). Parallel Algebraic Modeling for Stochastic Optimization. In *1st Workshop High Perform. Tech. Comput. Dyn. Lang.*, pages 29–35, New Orleans, LA. IEEE.

Hunter, K., Sreepathi, S., and DeCarolis, J. F. (2013). Modeling for insight using tools for energy model optimization and analysis (Temoa). *Energy Econ.*, 40:339–349.

Huster, W. R., Schweidtmann, A. M., and Mitsos, A. (2019). Impact of Accurate Working Fluid Properties on the Globally Optimal Design of an Organic Rankine Cycle. *Comput. Aided Chem. Eng.*, 47:427–432.

IDAES homepage (2020). https://idaes.org/ (accessed February 02 2021).

Jalving, J., Abhyankar, S., Kim, K., Hereld, M., and Zavala, V. M. (2017). A graph-based computational framework for simulation and optimisation of coupled infrastructure networks. *IET Gener. Transm. Distrib.*, 11:3163–3176.

Jalving, J., Cao, Y., and Zavala, V. M. (2019). Graph-based modeling and simulation of complex systems. *Comput. Chem. Eng.*, 125:134–154.

Jentsch, A., Bohn, K., Pohlig, A., Dötsch, C., Richter, S., and Manderfeld, M. (2008). Handbuch zur Entscheidungsunterstützung - Fernwärme in der Fläche: Leitungsgebundene Wärmeversorgung im ländlichen Raum. Technical report, Fernwärmeversorgung Niederrhein GmbH, Dinslaken and Forschungszentrum Jülich, Projektträger Material und Rohstofforschung (PLR) and Fraunhofer-Institut für Umwelt-, Sicherheits- und Energietechnik (UMSICHT), Oberhausen and GEF Ingenieur AG, Leimen.

Johnston, J., Maluenda, B., Henríquez, R., and Fripp, M. (2019). Switch 2.0: A Modern Platform for Planning High-Renewable Power Systems. *SoftwareX*, 10:100251.

Kim, H., Kim, S., Kim, T., Lee, T. H., Ryu, N., Kwon, K., and Min, S. (2018). Efficient design optimization of complex system through an integrated interface using symbolic computation. *Adv. Eng. Software*, 126:34–45.

Li, C. and Grossmann, I. E. (2019). A generalized Benders decomposition-based branch and cut algorithm for two-stage stochastic programs with nonconvex constraints and mixed-binary first and second stage variables. *J. Global Optim.*, 75(2):247–272.

Li, X., Armagan, E., Tomasgard, A., and Barton, P. I. (2011). Stochastic pooling problem for natural gas production network design and operation under uncertainty. *AIChE J.*, 57(8):2120–2135.

Li, X. and Barton, P. I. (2015). Optimal design and operation of energy systems under uncertainty. *J. Process Control*, 30:1–9.

Loulou, R. and Labriet, M. (2007). ETSAP-TIAM: the TIMES integrated assessment model Part I: Model structure. *Comput. Manag. Sci.*, 5(1-2):7–40.

Magnusson, F. and Åkesson, J. (2015). Dynamic Optimization in JModelica.org. *Processes*, 3(2):471–496.

Mavrotas, G. (2009). Effective implementation of the ϵ-constraint method in Multi-Objective Mathematical Programming problems. *Appl. Math. Comput.*, 213(2):455–465.

Meurer, A., Smith, C. P., Paprocki, M., Čertík, O., Kirpichev, S. B., Rocklin, M., Kumar, A., Ivanov, S., Moore, J. K., Singh, S., Rathnayake, T., Vig, S., Granger, B. E., Muller, R. P., Bonazzi, F., Gupta, H., Vats, S., Johansson, F., Pedregosa, F., Curry, M. J., Terrel, A. R., Roučka, Š., Saboo, A., Fernando, I., Kulal, S., Cimrman, R., and Scopatz, A. (2017). SymPy: symbolic computing in Python. *PeerJ Comput. Sci.*, 3:e103.

Miller, D. C., Siirola, J. D., Agarwal, D., Burgard, A. P., Lee, A., Eslick, J. C., Nicholson, B., Laird, C., Biegler, L. T., Bhattacharyya, D., Sahinidis, N. V., Grossmann, I. E., Gounaris, C. E., and Gunter, D. (2018). Next Generation Multi-Scale Process Systems Engineering Framework. In *13th Int. Symp. Process Syst. Eng.*, pages 2209–2214, San Diego, CA. Elsevier.

Mitsos, A., Chachuat, B., and Barton, P. I. (2009). McCormick-Based Relaxations of Algorithms. *SIAM J. Optim.*, 20(2):573–601.

Navarro, A. K. W. and Vassiliadis, V. S. (2014). Computer algebra systems coming of age: Dynamic simulation and optimization of DAE systems in Mathematica™. *Comput. Chem. Eng.*, 62:125–138.

Nicholson, B., Siirola, J. D., Watson, J.-P., Zavala, V. M., and Biegler, L. T. (2018). pyomo.dae: a modeling and automatic discretization framework for optimization with differential and algebraic equations. *Math. Program. Comput.*, 10(2):187–223.

Nikolić, D. D. (2016). DAE Tools: equation-based object-oriented modelling, simulation and optimisation software. *PeerJ Comput. Sci.*, 2:e54.

Papoulias, S. A. and Grossmann, I. E. (1983). A structural optimization approach in process synthesis—I: Utility systems. *Comput. Chem. Eng.*, 7(6):695–706.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Édouard Duchesnay (2011). Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.*, 12(85):2825–2830.

Pfeiffer, A. (2012). Optimization Library for Interactive Multi-Criteria Optimization Tasks. In *Proc. 9th Int. MODELICA Conf.*, pages 669–680, Munich, Germany. Linköping University Electronic Press.

Pfenninger, S., Hawkes, A., and Keirstead, J. (2014). Energy systems modeling for twenty-first century energy challenges. *Renew. Sustain. Energy Rev.*, 33:74–86.

Pfenninger, S. and Keirstead, J. (2015). Renewables, nuclear, or fossil fuels? Scenarios for Great Britain's power system considering costs, emissions and energy security. *Appl. Energy*, 152:83–93.

Pistikopoulos, E. N. (1995). Uncertainty in process design and operations. *Comput. Chem. Eng.*, 19:553–563.

Process Systems Enterprise (1997-2019). gPROMS. https://www.psenterprise.com/products/gproms (accessed February 02 2021).

Saelens, D., de Jaeger, I., Bünning, F., Mans, M., Maccarini, A., Garreau, E., Rønneseth, Ø., Sartori, I., Vandermeulen, A., van der Heijde, B., and Helsen, L. (2020). Towards a DESTEST: a District Energy Simulation Test Developed in IBPSA Project 1. In Corrado, V., Fabrizio, E., Gasparella, A., and Patuzzi, F., editors, *Proc. 16th Int. Conf. Build. Simul.*, Building Simulation Conference proceedings, pages 3569–3577, Rome, Italy. IBPSA.

Sahinidis, N. V. (2020). *BARON 20.10.16: Global Optimization of Mixed-Integer Nonlinear Programs,* User's Manual.

Sass, S., Faulwasser, T., Hollermann, D. E., Kappatou, C. D., Sauer, D., Schütz, T., Shu, D. Y., Bardow, A., Gröll, L., Hagenmeyer, V., Müller, D., and Mitsos, A. (2020). Model Compendium, Data, and Optimization Benchmarks for Sector-Coupled Energy Systems. *Comput. Chem. Eng.*, 135:106760.

Sass, S. and Mitsos, A. (2019). Optimal operation of dynamic (energy) systems: When are quasi-steady models adequate? *Comput. Chem. Eng.*, 124:133–139.

Schrattenholzer, L. (1981). The energy supply model MESSAGE. Technical report, International Institute for Applied Systems Analysis, Laxenburg, Austria.

Schweidtmann, A. M., Huster, W. R., Lüthje, J. T., and Mitsos, A. (2019). Deterministic global process optimization: Accurate (single-species) properties via artificial neural networks. *Comput. Chem. Eng.*, 121:67–74.

Schweidtmann, A. M. and Mitsos, A. (2018). Deterministic Global Optimization with Artificial Neural Networks Embedded. *J. Optim. Theory Appl.*, 189:925–948.

Schäfer, P., Caspari, A., Kleinhans, K., Mhamdi, A., and Mitsos, A. (2019a). Reduced dynamic modeling approach for rectification columns based on compartmentalization and artificial neural networks. *AIChE J.*, 65(5):e16568.

Schäfer, P., Caspari, A., Mhamdi, A., and Mitsos, A. (2019b). Economic nonlinear model predictive control using hybrid mechanistic data-driven models for optimal operation in real-time electricity markets: In-silico application to air separation processes. *J. Process Control*, 84:171–181.

Schäfer, P., Schweidtmann, A. M., Lenz, P. H. A., Markgraf, H. M. C., and Mitsos, A. (2020). Wavelet-based grid-adaptation for nonlinear scheduling subject to time-variable electricity prices. *Comput. Chem. Eng.*, 132:106598.

Schütz, T., Schraven, M. H., Fuchs, M., Remmen, P., and Müller, D. (2018). Comparison of clustering algorithms for the selection of typical demand days for energy system synthesis. *Renew. Energy*, 129:570–582.

Smith, E. M. d. B. (1997). *On the optimal design of continuous processes.* PhD thesis, Imperial College London.

Thieriot, H., Nemura, M., Fritzson, P., Singh, R., Kocherry, J. J., and Torabzadeh-Tari, M. (2011). Towards design optimization with OpenModelica emphasizing parameter optimization with genetic algorithms. In *Proc. 8th Int. MODELICA Conf.*, pages 756–762, Dresden, Germany. Linköping University Electronic Press.

Valente, C., Mitra, G., Sadki, M., and Fourer, R. (2009). Extending Algebraic Modelling Languages for Stochastic Programming. *INFORMS J. Comput.*, 21(1):107–122.

van Beuzekom, I., Gibescu, M., and Slootweg, J. G. (2015). A review of multi-energy system planning and optimization tools for sustainable urban development. In *IEEE PowerTech*, pages 1–7, Eindhoven, The Netherlands. IEEE.

Vielma, J. P., Ahmed, S., and Nemhauser, G. (2010). Mixed-Integer Models for Nonseparable Piecewise-Linear Optimization: Unifying Framework and Extensions. *Oper. Res.*, 58(2):303–315.

Voll, P., Klaffke, C., Hennen, M., and Bardow, A. (2013). Automated superstructure-based synthesis and optimization of distributed energy supply systems. *Energy*, 50:374–388.

Watson, J.-P., Woodruff, D. L., and Hart, W. E. (2012). PySP: modeling and solving stochastic programs in Python. *Math. Program. Comput.*, 4(2):109–149.

Yunt, M., Chachuat, B., Mitsos, A., and Barton, P. I. (2008). Designing man-portable power generation systems for varying power demand. *AIChE J.*, 54(5):1254–1269.