Modelica-based Technologies for Modeling and Simulation of Power Systems

(Version 0.2) – Monday 22nd February, 2021

Atiyah M. G. Elsheikh*, Peter Palensky *Senior Member, IEEE*§,

*Mathemodica.com , Egypt & Germany

§ Department of Intelligent Electrical Power Grids, Delft University of Technology, Delft, The Netherlands

Abstract

This is a comprehensive but a concise and educational technical report aiming at advertising Modelica-base technologies particularly useful for power system modeling applications. What ever aspect that could be useful has been included, to the best of author's knowledge. We hope that this report to be useful not only for power system modelers desiring to get a quick idea about the benefits of employing Modelica but also for those Modelica modelers desiring a starting guide into the world of Power System Modeling applications.

ACKNOWLEDGEMENT

Atiyah Elsheikh is highly appreciating his former employer, Austrian Institute of Technology, as this report has been initially started during his role there.

Moreover, few parts of this report has been written by others. Without their contribution, the report would be definitely less valuable. Thus, I'd like to thank (in alphabetical order of family names):

- Prof. Andrea Benigni, RWTH Aachen and FZ Jülich, with his great help, this report is suitable for Electrical Engineers. Particularly, major parts of Section II and Section X was originally written by him.
- Assoc. Prof. Omar Faruque, Florida State University, for presenting this initiative at a PES general meeting
- Prof. Antonello Monti, RWTH Aachen, being the initiator of the idea of having a comprehensive report that gathers all useful aspects Modelica can provide for power system modeling applications. The introduction section was originally written by him.

CALL FOR FINANCIAL SUPPORT OR DONATION

Donation link

https://www.paypal.com/paypalme/mathemodica

This technical report has been majorly written by the first author Atiyah Elsheikh, in his free-time. If this report is useful for you, **the first author** appreciates a donation from which he is going to finance the continuation of maintaining and progressing this report among other similar activities cf. http://mathemodica.com/projects. These activities are in conformance with the spread of open science initiative.

About / License

This is a free printable and distributable copy of a comprehensive educational report on Modelica-based technologies for power systems modeling application. The latex source codes will be also accessible, modifiable and distributable. You may reuse it for whatever useful purpose you have, e.g. project proposals. However you will need to provide a clear attribution/reference to this document and the owner, the first author. Also you are highly encouraged to actively improve the state of this document whenever possible. For this reason, this document is published in Authorea, in order to allow others to collaboratively improve its contents.

CONTENTS

I. Introduction

Traditionally modeling and simulation has been conducted within a single physical domain type. Electrical engineers used to develop electrical systems primarily focusing on grid modeling and mostly neglecting other interacting domains. One of the significant exception to this rule has been a simplified representation of the rotating mass for stability analysis (Kundur et al., 1994; Ulbig et al., 2014).

In the last decade though, we have experienced a growing interest towards multi-physics design in many different areas of applications. Two concrete examples of this type are the avionics industry and ship industry in relation to programs such as More Electric Aircraft (Hafez and Forsyth, 2009; Rahrovi and Ehsani, 2019) and All Electric Ship (Thongam et al., 2013; Sulligoi et al., 2016). The development of such projects has shown the limitation of the single-domain approach and has paved the way to comprehensive approaches for multi-physical modeling and simulation (Valdivia-Guerrero et al., 2016).

One response to the problem has been the development of co-simulation interfaces and standards (Blochwitz and et al., 2011; Müller and et al., 2016; Gomes et al., 2018). Co-simulation offers the user the possibility to operate in a domain-specific environment and to relegate the integration of other simulation platforms to be mostly a software challenge. While this approach has some clear advantages from user perspective, first of all the possibility to continue operating with the same tools also in the new operating conditions, co-simulation tends to have a significant

computation and implementation overhead given by the co-presence of more than one simulation platform. Nevertheless, many tools exist simplifying the co-simulation implementation overhead, e.g. (Widl et al., 2013; Ochel and et al., 2019). Furthermore, it is quite easy to face versioning issue created by the fact that the project as a whole is stored in multiple files.

Consequently there is a growing interest towards multi-physical modeling languages. Multi-domain languages provide an interesting solution, first of all because they are languages and not simulation platforms. This means that the modeling effort and the solution effort are clearly separated. This separation brings an incredible benefit from the user point of view that is able to migrate from one simulation tool to another without repeating the modeling effort. There are many simulation languages of this type available s.a. VHDL-AMS (Ashenden et al., 2003), gPROMS (gPROMS) and Modelica (Elmqvist, 2014). The latest language has been developed as an initiation for a standard modeling language used by a large modeling community in many modeling domains. This has resulted in an open-source non-property specification language continuously maintained, supported and progressed by a large community both from academia and industry.

In this report we first review the major challenges in power system modeling and simulation and how those affect power system modeling languages and simulation tools requirements. We then dive into the Modelica language, giving a non-familiar reader though rather a minimal overview of the language but hopefully generous enough to recognize its potentials, advantages and limits in the scope of power system Modeling applications. We demonstrate some state-of-the-art modeling efforts, special Modelica-based related technologies, some conducted applications and a summary of some existing open-source libraries.

II. Challenges in Modeling and Simulation of Power Systems

Simulation is one of the most important tools in power system planning and analysis, both for the electrical system as a standalone subject and as a part of an interdependent, multi-physical system. In this section traditional approaches to power system simulation are reviewed. Afterwords modern aspects in power systems are demonstrated.

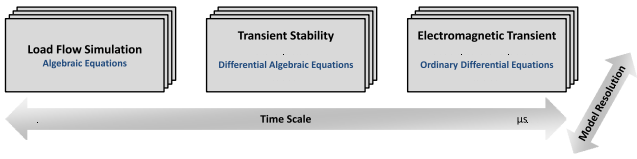

Fig. 1: Typical classical power system simulation studies w.r.t. time domain and model types

A. Traditional power system simulation studies

Traditionally, simulation analysis in support of power system design is performed by well-established mathematical techniques. Figure 1 demonstrates three commonly utilized types of simulations for power system design and analysis briefly summarized as follows:

1. Load-flow studies (Prabhu et al., 2016) are performed to determine the steady-state operation of an electric power system. It corresponds to the energy flow through each transmission line described via a set of non-linear algebraic equations. The solution determines the magnitude and phase angle of the voltage at each bus, and the real and reactive power flowing in each line. The main applications of this type of analysis is to determine if

- the system voltages remain within specified limits under various contingency conditions (e.g. N-1 analysis)
- whether equipment such as transformers and conductors are overloaded

Load-flow studies are often used for planning, economic scheduling, and control of an existing system as well as planning its future expansion identifying the need for additional generation, lines, VAR support, or the placement of capacitors and/or reactors to maintain system voltages within specified limits.

2. Transient stability simulation (Halder et al., 2018) is often referred as quasi-dynamic simulation described by a set of Differential Algebraic Equations (DAE). It is used to determine whether in consequence of a contingency the power system returns to a stable and acceptable operating point. Typical applications are:

- identifying fault clearing time
- checking generator rotor angle stability
- assessing system stability margin

- evaluating motor dynamic acceleration and reacceleration impact and
- preparing and testing load shedding schedule

Some quantities are assumed to be

- constant (e.g. LTC position)
- fast enough to be represented by algebraic relation (e.g. synchronous machine stator dynamics, voltage source converter dynamics)
- represented by differential equation (e.g. turbine and synchronous machine rotor dynamics)

Transient stability covers simulation windows that go from tens of millisecond to hundreds of seconds.

3. Electromagnetic/Electromechanical transient stability simulation(Dommel, 1969; Huang, 2016) is described with a set of nonlinear Ordinary Differential Equations (ODE) model of the network and is typically used to analyze dynamical behavior induced by rapid events. Typical applications are to analyze

- the insulation coordination as consequence of fast transient with the purpose of determining surge arrestor ratings and characteristics
- overvoltage due to circuit breaker operation or transients operations due to power electronics systems and ferroresonance phenomena

Typically the analyzed time-axis spans from hundreds of nanoseconds to tens of microseconds.

Changing the analyzed dynamics typically involves a re-modeling of the complete system with different component models and tools, therefore doubling the modeling effort and creating an additional source of errors. The situation is typically even worst because for each time scale analysis several models are created comprising a different resolution.

B. Modern aspects in power system modeling applications

The monitoring and control of future power systems are expected to be characterized by the distribution of functions and by the dependence on communications (Sabeeh and Gan, 2016). Distributed monitoring and control in increasingly involved because of the

1) features of new resources (generation and storage)
2) participation of loads in the energy management

3) the demand for fast, local reactivity for dynamic control and protection

Given the finer granularity of the controllable energy entities, and given the natural variability of some renewable sources, the sensitivity to individual load variation is greater than in traditional systems. Furthermore, the future energy grid is expected to incorporate electrical, gas and heat networks, to achieve maximum usage of the available energy in every form, and storage capacity particularly in thermal form (Leitner et al., 2019).

In this context the design of each part (subsystem, component, algorithm, logic) is a challenge due to the interactions, the interdependencies, and the dimension. This cannot be simplified via de-coupling, without losing essential dynamics. The impact of the individual element on the system, and vice versa is not straightforward to be inferred analytically (Hiskens and Pai, 2000). Consequently, the traditional design spiral cannot be applied to the considered kind of complex power systems. Overall this leads to the creation of very large systems that need to be analyzed via simulation in reasonable runtime not slowing the design process and enabling real-time execution.

The selection of the proper time scale and level of details has been based on the experience gained on well-defined operation, design conditions and on quite well-known and separate dynamic behaviors. Meanwhile considering the increasing complexity that terrestrial power systems are experiencing due to implementation of the smart grid concept and the need for a more systematic approach to simulation accuracy selection is an issue of growing importance. Future electrical subsystems will exhibit more interaction with several different physical phenomena of different time constants. Involved distribution networks will experience more dynamical behaviour dramatically enlarging the size of the system to be analyzed. Additionally, the growing presence of power electronics converters will significantly decrease the typical power system time constants.

Stochastic effects of future power systems are expected to be highly frequent mainly due to the volatility of some of the new, pervasive energy sources (particularly large and small renewable sources, and consumer owned small sources) and the effects of communication networks (e.g. delays). These stochastic phenomena are in addition to those that are typical also of classical power networks, originating from load behavior, prices and components reliability.

The distribution of control and monitoring functions, and by the remote coordination of its components leads to an increasingly relevant role of the communication infrastructure and to a heavier dependence of the performance of the entire system on communication performance. Effects such as time delay, packet loss, packet corruption, disconnections, packet re-ordering, jittering, etc. may affect the timely, stable reaction of power components and control. Conversely the operating mode of the power components affects the traffic, delay, etc. of the communication network.

Mismatches are to be expected between the models used in the design process together with field conditions and the models of the designed device and the real device. These mismatches cannot be compensated by oversizing or guaranteeing large margins from the critical operating points, as this would lead to scaling up quickly, leading to infeasibility. Furthermore, the testing of the model of the device, even within the model of the entire system, may be insufficient. Hence, physical realization must be tested in most realistic settings at the greatest possible extent prior to deployment. In this context incremental prototyping tools have to be considered a fundamental tool to support the development of new products. As consequence real-time simulation and (Power) Hardware In the Loop techniques are to play a fundamental role.

C. Mutli-physical phenomena in power systems [TO Complete]

Would be nice to have this subsection. If no text is going to be provided, subsection is going to be omitted.

III. The Rise of the Modelica Language

A. Pre-era Modelica

Before the rise of the digital computers, modeling and simulation of mathematical (ordinary differential) equations were realizable by earlier electrical analog computers. Exploiting physically continuous phenomena have enabled the simulation of ODEs. This can be achieved by the means of analog circuits components, (Ragazzini et al., 1947) as well as with mechanical based analog computers. An underlying typical block-diagram, conceptually and visually corresponding to such analog circuits, is composed of many interconnected blocks (Cellier, 1991). Each block:

- physically corresponds to a common element of analog circuits, e.g. a resistor, an inductor, a capacitor, among other possibilities
- conceptually maps a mathematical function of a set of input signals to an output signal

Practically, compositions of such blocks are capable of imitating an ODE among other continuous mathematical functions (Kolmogorov, 1956).

Due to this background, with the rise of of digital computers, the block-diagram approach was initially the dominant modeling approach by earlier general-purpose simulation languages. Nevertheless, this modeling approach, though particularly useful in some application domains like control applications (Otter and Cellier, 1996), was not the most suited for the new era. While the computational structure of the a given physical system (e.g. an electric circuit) is well-preserved (i.e. it is straightforward to map the given block-diagram to a set of equations), the topological structure of the physical system was not preserved in the block-diagram. This implies that it is not always possible to recognize the physical subsystems and their components in the block diagram, cf. (Cellier, 1991) Chapter 7. A slight modification in the physical system (e.g. insertion of a resistor in an electrical circuit) can not be easily reflected in the corresponding block-diagram.

On April 24th 1959, a new era of modeling paradigms emerged. Namely, the MIT-Professor Henry M. Paytner pioneered the physically-universal Bond graph modeling concept allowing a system representation that both reflects the computational and topological structure (Paynter, 1961). While a connection between components in a system described via a block-diagram approach corresponds to a causal relation between an output signal and a set of input signals, within a bond graph, a connection (i.e. an edge) between system components (i.e. nodes) rather corresponds to universal physical laws of Physics, e.g. Kirchhof's lows for current (so-called 1-Junction nodes) or Kirchhof's lows for voltage (so-called 0-Junction nodes). The 1-junction nodes exhibit the flow of power while 0-junction nodes maintain efforts through adjacent nodes (Åström et al., 1998; Cellier, 1991). This concept generalizes well to other physical domain by properly selecting the flow variables and efforts variables as interfaces for connectable components. For example, this would be torque and angular velocity for multibody applications. In other words this modeling concept corresponds to acausal relation among system variables with no notions of inputs-output relationship.

The bond graph concept converged to a stand-alone discipline in the Systems Engineering domain (Filippo et al., 1991). Many modeling frameworks for modeling dynamical systems have been established very early, cf. (Filippo et al., 1991) for a demonstration of such prototyping frameworks based on the bond graph concept. The majority of these frameworks were composed of a set of FORTRAN routines where the modelers need to specify subsystems and components using a set of line codes.

B. The evolve of the Modelica language

A cumbersome exploitation and extension of the bond graph concept was realized by the Dymola (DYnamic MOdeling LAguage) invented by Hilding Elmqvist in a PhD thesis (Elmqvist, 1978; Åström et al., 1998) late seventies, a predecessor of the Modelica language. Dymola was a stand-alone modeling language with object-oriented facilities allowing the description of a system in terms of hierarchical composition of subsystems and classes. Dymola enabled equations prototyping rather than assignments[1] for describing the physics of a model. Dymola compiler engine made use of graph algorithms and symbolic computation for converting a model paradigm to a solvable ODE system.

The accomplishment of the acausal modeling concept desired progress both at hardware technology and methodology levels. Even a simple model can be compiled into a high-dimensional equations system that hardware technologies at early time were not mature enough for its evaluation, yet (Elmqvist, 2014). Further methodological progress for establishing algorithms capable of efficient simulation of large-scale equation systems was required, for instance:

- Fundamental concepts for efficient equation-based compiler techniques by which a very-large scale equation system is simplified to a much smaller one by removing trivial equations and conducting simple symbolic manipulation (Murota, 1987; Maffezzoni et al., 1996; Cellier and Kofman, 2006)
- Methods for structural solvability analysis by which a large-block of an equation system is decomposed into smaller causal cascaded blocks of equation systems solved in a sequentially

[1] An equation describes an acausal relationship between variables that need to be fulfilled concurrently where as an assignment is an explicit causal relation between an output variable and a set of input variables

faster way (Cellier and Elmqvist, 1993; Leitold and Hangos, 2001). This level is also dealing with algebraic loops by which tearing algorithms are applied for solving sparse large-scale non-linear equation systems (Kron, 1963; Elmqvist and Otter, 1994)

- Index reduction algorithms by which a high-index differential algebraic equations with implicit algebraic constraints is transformed to a numerically solvable DAE system of index one or an ODE system (Pantelides, 1988; Mattsson and Söderlind, 1993)
- Consistent initialization of DAE by which consistent start values of the DAE needs to be identified through solving a nonlinear equation system using a Newton-iteration scheme. This can be a challenging problem due to the influence of initial guesses. Thus, advanced strategies are needed which is still an active area of research Casella and Bachmann (2021).

Nevertheless, by mid nighties many modeling languages existed by several working groups (Marquardt, 1996; Åström et al., 1998). This caused not only that each language or modeling tool brought its own strength and weakness, but also significant duplicated efforts were not avoidable. Model exchange and reuse among different tools were not possible. In other words as stated by (Elmqvist, 2014):

> ... Modeling requires reuse of stored knowledge, i.e. there must be a standard language. It does not make sense that various tool vendors invent their own language and that a new language is created for every Ph.D. thesis on modeling ...

In order to unify the splintered activities within the modeling community, an initiation for collaborative efforts to design a unified standard modeling language that includes the best and state-of-the-art of established modeling concepts (Elmqvist and Mattsson, 1997) including

- Support for both acausal and block-diagram approaches
- Object-oriented paradigms
- Equation-based semantics (Augustin et al., 1967) as the main building blocks for implementing model components via hybrid DAE

These efforts have converged to the Modelica language with the first open-source specification of the Modelica language published 1997, cf. Modelica Association (MA) www.modelica.org.

C. Predecessors of Modelica (To complete)

Modelica can be seen as decades of accumulation of expertise and experiences gathered from many modeling languages. To their credits the following languages should be summarized in a brief paragraph as their distinguished features have influenced Modelica in a way or another.

- CSSL and ACSL (block-diagram) then Simulink
- Omola : object-oriented features
- Simula
- 20sim and predecessors
- GPROMS and predecessors
- VHDL , VHDL-A

D. Benefits of the Modelica language

Due to the adopted features. particularly the ultimate acausal modeling concept, cf. Figure 2, the Modelica language provides several obvious benefits for modeling applications (Wetter and Haugstetter, 2006; Abel, 2010; Pollok et al., 2019; Schweiger et al., 2020a):

1) A model diagram topologically resembles the physical system
2) It is straightforward to map the diagram to an equivalent set of equations
3) It is easier to modify a system and reuse model components
4) Rapid prototyping is efficiently more productive
5) Multi-domain physical phenomena are easier to capture

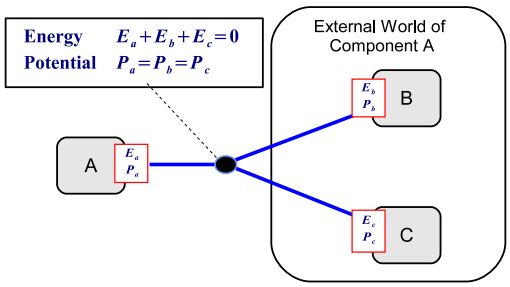

Fig. 2: Acausal modeling concepts reflecting standard universal lows of energy conservation are used to express the interactions between a system component A and the external world (components) B, C.

For instance, following the causal modeling approach, a power network is usually modelled using the traditional admittance matrix method. In contrary, with the acausal modeling approach, the power system topology can be visualized and modified in a graphical editor (Larsson, 2000). Analogously, an implementation of synchronous machines in the block-diagram based tool LabView results in a graphical model composed of interconnected I/O blocks each describing an (a set of) equation(s) (Karnavas and Lygouras, 2018). However, the topology of the underlying system is not directly viewable in the graphical model. On the other hand, for a similar application realizing rotating machines in Modelica with a completely transparent, accessible and modifiable equations (Kral and Haumer, 2011a), the topology of rotating machines is reflected in the graphical editor.

IV. Designing a Modelica library in Power Systems: Basic concepts

In (Bartolini et al., 2019), the Modelica library `PowerSystems` (Franke and Wiesmann, 2014) is briefly evaluated as follows:

> The very high level of abstraction, coupled with an extensive use of inheritance, allows to accommodate a very wide scope in a unified framework spanning DC, one- and three-phase AC in both quasi-static and dynamic regimes. The end result is a code base which is very general, elegant and concise; however, the price to pay for these features is that the code is quite difficult to comprehend for non-Modelica experts.

While we agree with the above statement, it encourages us to carry out two challenging but adventurous missions oriented towards non-Modelica experts:

1) initiating a first impression of Modelica syntax
2) making it easier to understand and deal with the *PowerSystems* Modelica library as well as other open-source Modelica libraries

Some capabilities of the `PowerSystems` library is demonstrated in a step-by-step manner simultaneously clarifying the features of Modelica language that make power system modeling a matter of fun. In this journey, we place ourselves in the head of a power system Modelica modeler and attempt to resemble the way he is tackling the design problem.

A. *Variables, parameters and constants*

Mathematical identities within an equation system describing a power system are classified under three types of variability:

1) Time-dependent variables whose values can change during a dynamic simulation course
2) Input parameters whose values are constant during a simulation but can change from a simulation to another
3) Constants whose values don't change at all

Modelica provides language keywords for differentiating between different variability of identities at declaration level. Declared identities in a Modelica model may look as follows:

```
Real v "Voltage";
Real i "Current";
parameter Real  R = 1.0 "Resistance";
constant  Integer n = 1 "# of independent voltage and current components";
constant  Integer m = 0 "# of reference angles";
```

Listing 1: Declaration

The keywords `parameter` and `constant` are used for distinguishing parameters and constants from variables. Typically, any declaration is briefly commented with the string `"comment ..."`. Common simulation environments generate HTML-documentation that show declared identities of all components of a library within a tabular form.

B. *Physical units*

An essential requirement for physical modeling is the capability of associating physical units to variables and parameters rather than declaring them using common primitive data types. This can be realized by defining physical units as follow:

```
type Voltage = Real(final unit = "V",
                    final quantity = "Voltage");
type Current = Real(final unit="A",
                    final quantity="Current");
```

Listing 2: Implementation of physical types

Here, a physical-unit type is a real-valued type associated with a physical unit using the attribute *unit* of type `Real`. With the above types, variables can be declared using these types:

```
Voltage v "Voltage";
Current i "Current";
```

Listing 3: Declaring a voltage variable

Consequently equations can be physically validated through unit-checking at compile time by common simulation environments. Other integer-valued physical types are also declarable but with the keyword `Integer`. Moreover, constrained types with maximum and/or minimum values can be associated to provide a mechanism of error reporting if a variable violates these constraints during simulation runtime. Usually, most of common physical units are provided within the Modelica Standard Library (MSL), introduced in Subsection VII-B, within the `Modelica.Units` package. Therefore explicit implementation of physical units is usually not needed. However, the modeler can still implement his own special types with further own attributes and special physical units.

C. Packages

A `package` in Modelica resembles the *class* concept in common object-oriented languages though a package is not instantiated by declaring an object. A Modelica package is usually composed of a set of attributes. An attribute can be a constant, a function, a declarable component among other possibilities. Declaring a Modelica library named `PowerSystems` is as follows:

```
package PowerSystems
 ...
 import Modelica.SIunits.*;
 ...
 // implementation of the library takes place here
 ...
end PowerSystems;
```

Listing 4: Declaring the library PowerSystems

The import statement implies that all physical types within the MSL package `Modelica.SIunits` can be directly employed across all components of the library:

```
Voltage v;
// instead of Modelica.SIunits.Voltage v;
```

Listing 5: Declaration of Voltage

Alternatively, since the modeler may provide own physical units, e.g. different than those within MSL, it is proper to keep it clear whether a physical unit is SI-standard or an own declared physical unit:

```
import SI = Modelica.SIunits;
...
    SI.Voltage v;
...
```

Listing 6: Declaration of Voltage

Here the package *SI* is a dummy replacement of the long package name `Modelica.SIunits` that otherwise needs to be associated to every physical variable. It is straightforward to employ different own-implementation of physical units package, e.g.

```
// import Modelica.SIunits.*;
import PowerSystems.Units.*;
...
    Voltage v;
...
```

Listing 7: Declaration of Voltage

though in this case the own package `PowerSystems.Units`:

1) should provide an implementation for all physical units employed across the library
2) does not need to follow SI standards

Mutual employment of physical units in both packages is possible with proper usage of dot notations.

D. Organization of packages

It is common that a library is composed of multiple levels of packages. Assuming that the `PowerSystems` library includes the packages `Control` and `Units` among others, and that the package `Control` includes the packages `Exciters` and `Governers`, then the corresponding textual representation may look as follows:

```
package PowerSystems
  ...
  package Control
    ...
        partial package Exciters
```

```
        ...
        // models for exciters come here
        ...
      end Exciters;

      package Governers
        ...
      end Governers;
      ...
  end Control;
  ...
  package Units
    ...
    // declaration of physical types
    ...
  end Units;
  ...
end PowerSystems;
```

Listing 8: Package hierarchies

There is no need to implement the hierarchies of several packages in a textual way. Common Modelica simulation environments provide the capabilities for flexible creation of packages and subpackages using a GUI-based editor.

E. Connections

First, in the sake of simplicity, a power system with one phase-system, i.e. direct current, is initially assumed. The generalization to other phase systems is discussed in the next Section. Before implementing power system components (i.e. lines, loads, generators etc.), it is intuitive to first consider the way these components get plugged into each other in an object diagram (rather than a block diagram). In other words, having two icons for two different components in a GUI-based object diagram editor, the question is *how to establish a physically well-defined connection mechanism enabling a connection line of these two icons in an object diagram?*

In Modelica, to make a component A connectable (either textually or graphically) to other components (say B and C), all components should include (a) connector(s) acting as attached communication port(s), cf. Figure 2. A `connector` specifies the variables of a model component (A) that:

- externally influence
- internally influenced by

the external world, i.e. the connected components B and C. All these components should share at least an identical `connector` type. In the context of a power system, a definition of a DC-connection between typical components (e.g. lines, loads, generators, etc.) may look as follows:

```
connector Terminal "DC terminal"
  Voltage v        "Voltage";
  flow Current i   "Current";
end Terminal;
```

Listing 9: Implementation of a DC-connector

Typically, two types of variables are usually[2] specified:

1) potential variables, e.g. voltage
2) energy-flow variables, e.g. current

In the object diagram of Figure 2, the potential variables of connected components resembles identity equation:

$$A.T.v = B.T.v = C.T.v$$

and the flow variables of connected components, distinguished by the keyword `flow`, result in a sum-to-zero equation:

$$A.T.i + B.T.i + C.T.i = 0$$

A model component can get connected to another model component only via included connectors of the same type (say Terminal). If there are two physically-distinct definitions of two connectors, then they can not be connected to each other.

F. Components

After establishing a well-defined physical connection mechanism, it is possible to implement typical components. Given that in an object diagram the flow of current from left to right through arbitrary component is commonly defined to be positive, cf. Figure 3, an implementation of `Resistor` may look as follows:

[2]There could be less but also more than two and there are other connection mechanisms as well (Franke et al., 2009a).

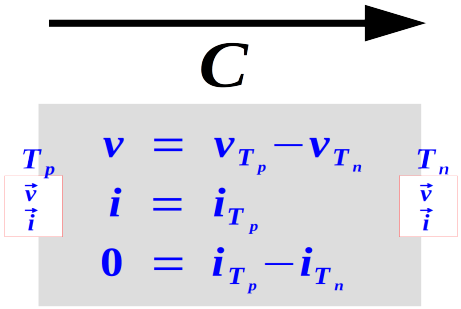

Fig. 3: The flow of current and the voltage difference within a component C with two terminals T_p and T_n

```
1  model Resistor
2    Terminal  T_p "terminal from left";
3    Terminal  T_n "terminal from right";
4    parameter Resistor R = 1.0 "Resistance";
5    Voltage   v   "Voltage";
6    Current   i   "Current";
7  equation
8    v   = T_p.v - T_n.v;
9    i   = T_p.i;
10   0   = T_p.i + T_n.i;
11   R*i = v "Ohm's law";
12 end Resistor;
```

Listing 10: Implementation of a Resistor component

The implementation of the `Resistor` component is realized as follows:

1) line 1: the keyword `model` is employed which is one of the most common ways of defining a component among other possible approaches
2) lines 2 and 3: two terminals (positive and negative) are attached to the `Resistor` component. These terminals can get connected to terminals of other components from the left and/or the right sides, correspondingly
3) lines 4-6: basic declarations of resistance, voltage and current
4) lines 8-10: the relationship between the terminals and the internal behavior of `Resistor` component is defined
5) line 11: the physical behavior (i.e. Ohm's law) is implemented using an implicit equation

The overall model apparently results in three equations in four unknown. Assuming that in a

power system network, two lines with different characteristics are connected. Then this configuration results in six equations in eight unknowns. The missing two equations are generated using the connection corresponding to Kirchhoff's low to accomplish a consistent equation system. The connection of two lines can be modeled as follows:

```
model ElectricNetwork
  ...
  Resistor line1(R=2.0);
  Resistor line2(R=4.0);
  ...
  // other components
  ...
equation
  ...
  connect(line1.T_n , line2.T_p);
  ...
  // other connections
  ...
end ElectricNetwork;
```

Listing 11: Connecting two lines

The `connect` statement expects two arguments each is an instance of a `connector`. The order of the arguments is irrelevant. The `connect` statement can be either textually implemented or generated using a GUI-based editor by dragging, dropping two components of type `Resistor` and connecting them together in a model-diagram editor.

V. Object-Oriented features

A. *Abstract Models and Inheritance*

A common object-oriented feature is the capability of introducing a generic abstract component from which several specific declarable components are derived, i.e. inheritance from abstract types. The first three equations in the implementation of `Resistor`, is almost the same in all components of a power network with few exceptions. For instance, the implementation of a transformer should convert an input current to another current value according to a given ratio. In order to avoid such repeated set of equations in multiple implementations of various components, an object-oriented design provides an abstract model for gathering such equations:

```
model PartialTwoTerminals
```

```
  Terminal  T_p  "terminal from left";
  Terminal  T_n  "terminal from right";
  Voltage   v    "Voltage";
  Current   i    "Current";
equation
  v   = T_p.v - T_n.v;
  i   = T_p.i;
  0   = T_p.i + T_n.i;
end PartialTwoTerminals;
```

Listing 12: implementation of an abstract model with two terminals

The keyword `partial` declares the model `PartialTwoTerminals` to be abstract. This prevents a modeler from declaring an instance object of such a model, given that the number of equations is less than the number of unknowns, i.e. inconsistent model. Models such as `Resistor` and `Capacitor` extend `PartialTwoTerminals` and provide an implementation only of the physical behavior, e.g.

```
model Capacitor
  extends PowerSystems.Ports.PartialTwoTerminal;
  parameter Capacitance C = 1.0 "reactive component";
equation
  C * der(v) = i;
end Capacitor;
```

Listing 13: Implementation of Capacitor

Line 2 specifies that the model `Capacitor` extends the base abstract model `PartialTwoTerminal`. The operator *der* expresses time derivative, i.e. `der(v)` corresponds to dv/dt. The `Resistor`, `Conductor`, etc. models are (re-)implemented in a similar way.

B. Arbitrary phase systems by an abstract package

Another feature of common object-oriented paradigms is the capability of providing generic templates for packages. For power systems modeling, this provides the opportunity to realize generic implementation of power systems components to be parametrized by phase systems. This enables a unique implementation of power system components, e.g. lines, for all possible phase systems. A generic phase system is as follows:

```
partial package PartialPhaseSystem  "Base package of all phase systems"
  ...
```

```
  constant String phaseSystemName = "UnspecifiedPhaseSystem";
  constant Integer n "Number of independent voltage and current components";
  constant Integer m "Number of reference angles";
  ...
end PartialPhaseSystem;
```

Listing 14: Implementation of generic phase systems

The keyword `partial` implies that this abstract package has undefined features. These features are subject to specification within extended non abstract packages. For instance, a specification is provided by the package `DirectCurrent`:

```
package DirectCurrent  "Direct Current"
  extends PartialPhaseSystem(
    phaseSystemName = "DirectCurrent",
    n=1, m=0);
  ...
end DirectCurrent;
```

Listing 15: Implementation of Direct Current phase system

Similarly, a three-phase system in dq0-representation is realized as follows

```
package ThreePhase_dq0  "AC system symmetrically loaded three phases"
  extends PartialPhaseSystem(
    phaseSystemName = "ThreePhase_dq",
    n=3, m=2);
  ...
end ThreePhase_dq0;
```

Listing 16: Implementation of dq0-based AC system

Further phase systems, e.g. dq-representation with $n = 2$ and $m = 2$, can be consulted from the available implementation of the library.

C. Interfaces

The realization of power system components depends on the specific phase system in which they operate. It is desired to associate an arbitrary generic phase system with functionalities for computing common useful quantities s.a. active power, system voltage, system current, among others. This can be realized by embedding interfaces in the `PartialPhaseSystem` package (cf. Listing 14). In this way, a specific phase system, e.g. `DirectCurrent`, is enforced to

provide a specific implementation to these interfaces. Interfaces of different phase systems usually require different implementations as shown later. This is realized in Modelica using functions:

```
1  partial package PartialPhaseSystem
2    ...
3    partial function j "Rotation by 90 degree"
4      input  Real x[:];
5      output Real y[size(x,1)];
6    end j;
7
8    partial function thetaRef "Absolute angle of rotating reference system"
9       input  Angle theta[m];
10      output Angle thetaRef;
11   end thetaRef;
12
13   // other interfaces for systemVoltage, systemCurrent,
14   // phasePowers, phaseVoltages, etc.
15   ...
16
17   partial function activePower "Total power"
18     input  Voltage v[n]  "phase voltages";
19     input  Current i[n]  "phase currents";
20     output ActivePower P "active system power";
21   end activePower;
22   ...
23 end PhaseSystem;
```

Listing 17: Interfaces for arbitrary PhaseSystems

In contrast to the implementation within an `equation` section (cf. Listing 10), functions are causal and evaluates a set of output quantities from a given set of input quantities. The following notes of the implementation of Listing 17 are highlighted:

1) The `partial function` keywords declare an interface (lines 3,8 and 17)
2) The `input` keyword declares the input(s) of a function (lines 4,9,18 and 19)
3) The `output` keyword declares the output(s) of a function (lines 5,10 and 20)
4) Arrays are declared using brackets, e.g. `theta[m]` line 9 where m is to be specified by a specific implementation of a phase system, e.g. `DirectCurrent` (cf. Listing 15)
5) The declaration `x[:]` (line 4) specifies that the function `j` accepts an array of arbitrary length

6) The built-in function `size(T,k)` (line 5) evaluates the length of the multidimensional tensor `T` along the k-th dimension
7) Thus, the declaration `y(size(x,1))` (line 5) specifies that the output vector `y` should be of the same length as the vector `x`

D. Implementation of Functions

Realization of the above functions for a component operating with a direct current is trivial due to the absence of reference angles:

```
1  package DirectCurrent
2    extends PartialPhaseSystem(...);
3    ...
4    // Direct current has no complex component
5    function j "Rotation by 90 degree"
6      extends PartialPhaseSystem.j;
7    algorithm
8       y := zeros(n);
9    end j;
10
11   // No absolute angle for DC
12   function thetaRef "Absolute angle of rotating reference system"
13     extends PartialPhaseSystem.thetaRef;
14   algorithm
15     thetaRef := 0;
16   end thetaRef;
17   ...
18   function activePower "Total power"
19     extends PartialPhaseSystem.activePower;
20   algorithm
21     P := v*i;
22   end activePower;
23   ...
24 end DirectCurrent;
```

Listing 18: Implementation of interfaces for DC

1) The implementations functions `j` (lines 5-9) and `thetaRef` (lines 12-15) are trivial for the phase system `DirectCurrent`
2) A function inherits the declarations from the base class by the `extend` statement (lines 6, 13 and 19)

3) The implementation of a function takes place in an `algorithm` section, e.g. lines 20-22
4) In an algorithm section the assignment notation `:=`, e.g. line 21 is employed, in contrary to common the equation notation = in `equation` section
5) the built-in function `zeros(n)` (line 8) assigns a one dimensional vector to the output vector `y`

Non-trivial implementation of such functions is carried out in e.g. AC phase system in dq0-representation:

```
1  ...
2  function j "Rotation(pi/2) of vector around {0,0,1}"
3    extends PartialPhaseSystem.j;
4  algorithm
5    y := cat(1, {-x[2], x[1]}, zeros(size(x,1)-2));
6  end j;
7
8  function thetaRef "Absolute angle of rotating reference system"
9    extends PartialPhaseSystem.thetaRef;
10 algorithm
11   thetaRef := theta[2];
12 end thetaRef;
13 ...
14 function activePower "Total power"
15   extends PartialPhaseSystem.activePower;
16 algorithm
17       P := v[1:2]*i[1:2];
18 end activePower;
19 ...
```

Listing 19: Implementation of interfaces for DC

1) The function `cat(k,...)` concatenates multiple arrays along the k-th axis
2) Active power is calculated using a dot product of the first two entries from the vectors v and i (line 17)

E. Generic connectors

Considering arbitrary phase systems, the actual implementation of the connector `Terminal` in Listing 9 is:

```
1 connector Terminal "General power terminal"
2   replaceable package PhaseSystem =
```

```
3     PhaseSystems.PartialPhaseSystem "Phase system";
4   Voltage v[PhaseSystem.n]      "Voltage vector";
5   flow Current i[PhaseSystem.n] "Current vector";
6   ReferenceAngle theta[PhaseSystem.m]
7     if PhaseSystem.m > 0  "Phase angles vector";
8 end Terminal;
```

Listing 20: Implementation of generic connector

1) This is a generic connector parameterized by the replaceable package *PhaseSystem* assigned with the value `PartialPhaseSystem` (lines 2 and 3)
2) The voltage and the current are declared as arrays of size `PhaseSystem.n` components (lines 4 and 5)
3) The value of n within the abstract package `PhaseSystem` is initially not specified
4) Thus, the keyword `replaceable` indicates that a re-declaration of the connector is desired to provide a well-defined connector
5) The `if` condition expresses a conditional declaration only if `PhaseSystem.m` is positive (lines 6 and 7)

An AC-specific connector in dq0-representation is declared as follows:

```
Terminal T(redeclare package PhaseSystem =  PhaseSystems.ThreePhase_dq0);
```

Listing 21: Declaration of a specific connector

Alternatively, an AC-specific connector type is designed as follows:

```
connector Terminal_AC3dq
  extends Terminal(redeclare package PhaseSystem = PhaseSystems.ThreePhase_dq);
```

Listing 22: Declaration of a specific connector

F. Generic components

A DC-specific partial model `PartialTwoTerminals` in Listing 12 is generalized to arbitrary phase systems as follows:

```
1 partial model PartialTwoTerminal
2   replaceable package PhaseSystem = PhaseSystems.PartialPhaseSystem;
3   Terminal T_p(redeclare package PhaseSystem = PhaseSystem);
4   Terminal T_n(redeclare package PhaseSystem = PhaseSystem);
5   Voltage v[PhaseSystem.n];
```

```
6     Current i[:];
7   equation
8     v = T_p.v - T_n.v;
9     i = T_p.i;
10    zeros(PhaseSystem.n) = T_p.i + T_n.i;
11    if PhaseSystem.m > 0 then
12      T_p.theta = T_n.theta;
13    end if;
14  end PartialTwoTerminal;
```

Listing 23: Generic component with two terminals

1) Within `redeclare` statements (lines 3 and 4), the first `PhaseSystem` refers to the re-declarable package in the connector `Terminal`, while the second `PhaseSystem` refers to re-declarable package of `PartialTwoTerminal`
2) Equations are expressed in terms of vectors (lines 8-10)
3) The last `if`-conditional equation is considered only if `PhaseSystem.m` is positive

Generic phase-system independent components are designed in a similar fashion. The component `Resistor` in Listing 10 is renamed and generalized to `Impedance`:

```
1   model Impedance
2     extends PowerSystems.Ports.PartialTwoTerminal;
3     parameter Resistance R = 1 "active component";
4     parameter SI.Inductance L = 1/314 "reactive component";
5     AngularFrequency omegaRef;
6   equation
7     if PhaseSystem.m > 0 then
8       omegaRef = der(PhaseSystem.thetaRef(T_p.theta));
9     else
10      omegaRef = 0;
11    end if;
12    v = R*i + omegaRef*L*j(i);
13    if PhaseSystem.m > 0 then
14      T_p.theta = T_n.theta;
15    end if;
16  end Impedance;
```

Listing 24: Implementation of impedance

Careful readers should be capable of undrestanding the previous implementation without further illustration. Phase system-specific components can be declared or designed in similar way as in

Listings 21 and 22.

VI. Examples

A. A power flow study

The first example demonstrates a typical load flow study of a power network model (Oeding and Oswald, 2016) using the OpenModelica simulation environment (Fritzson et al., 2020). Having accomplished the implementation of basic components of a power system library, a typical power system dynamical network model can be constructed by two ways:

1) Either by using a graphical modeling approach by dragging, dropping and connecting components together in the graphical model editor called OMEdit (Ashgar and et al., 2011), cf. Figure 4, resulting in the model diagram of Figure 5
2) or by assembling the network model in a textual manner, cf. Listing 25

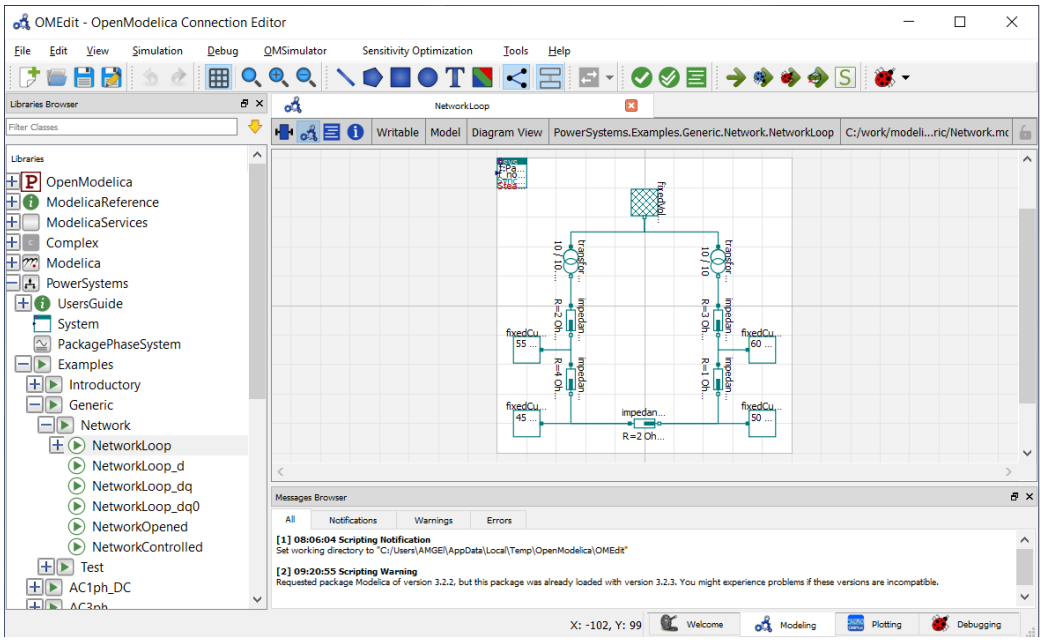

Fig. 4: The graphical and textual modeling editor OMEdit: a front-end for the OpenModelica simulation environment. Modelers are capable of choosing between several solvers and adjusting their settings. Advanced plotting capabilities including animation exist.

```
model NetworkLoop

  FixedVoltageSource fixedVoltageSource1(V=10e3)
```

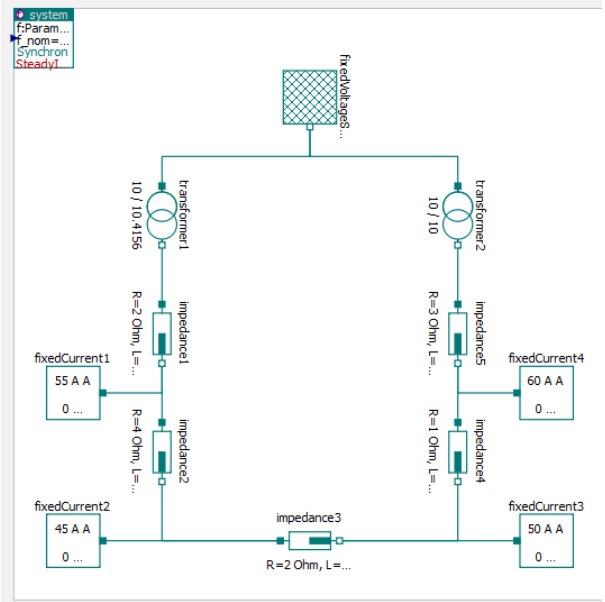

Fig. 5: A typical power system network *PowerSystems.Examples.Network.NetworkLoop* composed of fixed voltage source (the shadowed square), fixed current sources (the squares), transformers (two intersected circles) and impedances (dashed rectangles)

```
  Impedance line1(R=2, L=0)
  Impedance line2(R=4, L=0)
  Impedance line3(R=2, L=0)
  Impedance line4(L=0, R=1)
  Impedance line5(L=0, R=3)

  FixedCurrent fixedCurrent1(I=55);
  FixedCurrent fixedCurrent2(I=45);
  FixedCurrent fixedCurrent3(I=50);
  FixedCurrent fixedCurrent4(I=60);

  VoltageConverter transformer1(ratio=10/10.4156);
  VoltageConverter transformer2(ratio=10/10);

equation

  // lines
  connect(line1.T_n , line2.T_p);
  connect(line2.T_n , line3.T_p);
  connect(line4.T_p , line5.T_n);
```

```
  connect(line4.T_n , line3.T_n);

  // fixed current & voltage sources
  connect(fixedCurrent1.T , line1.T_n);
  connect(fixedCurrent2.T , line3.T_p);
  connect(fixedCurrent3.T , line3.T_n);
  connect(fixedCurrent4.T , line5.T_n);
  connect(fixedVoltageSource1.T , transformer1.T_p);

  // transformers
  connect(transformer1.T_n, line1.T_p);
  connect(transformer2.T_n, line5.T_p);
  connect(transformer2.T_p, fixedVoltageSource1.T);

end NetworkLoop;
```

Listing 25: Implementation of network model of Figure 5

This textual representation is automatically generated by the editor when constructing the network in graphical manner. The network model assumes that the component are associated with a specific phase system, say AC in dq0-representation. However, they can be also generic in terms of phase systems. In such a case, different models in different representations can be easily established:

```
model NetworkLoop_dq "NetworkLoop example with phase system ThreePhase_dq"
  extends NetworkLoop(
    redeclare replaceable package PhaseSystem =
      PowerSystems.PhaseSystems.ThreePhase_dq);
end NetworkLoop_dq;
```

Listing 26: Implementation of network model in dq representation

Having constructing the network model with the set of parameter values in Listing 25 with inductance being neglected, the compilation of this apparently simple model results in a nonlinear system of equations of dimension 267 equations out of which 121 equations are trivial, i.e. (equality equations $x = y$ or $x = -y$). Such trivial equations are subject to removal and only an optimized set of equations is evaluated (Maffezzoni et al., 1996). The simulation of this model produces the following samples of results:

Line	Voltage (kV)	Current (A)
Line 1	0.269	314.159
Line 2	0.319	79.633
Line 3	0.069	34.633
Line 4	0.015	15.367
Line 5	0.226	75.367

B. Power generation and consumption

Examples for dynamic simulation can be consulted in the `PowerSystems` library, e.g. the s.c. `PowerWorld` example (Franke and Wiesmann, 2014) shown in Figure 6. Simulation results of the power generation of the power plants and the power consumption of the city is shown in Figures 7 and 8.

So far the paper is concerned with illustrating fundamental concepts of the Modelica language. This can help interested readers to further explore the detailed implementation of the example that can be consulted by the library source code under https://github.com/modelica-3rdparty/PowerSystems.

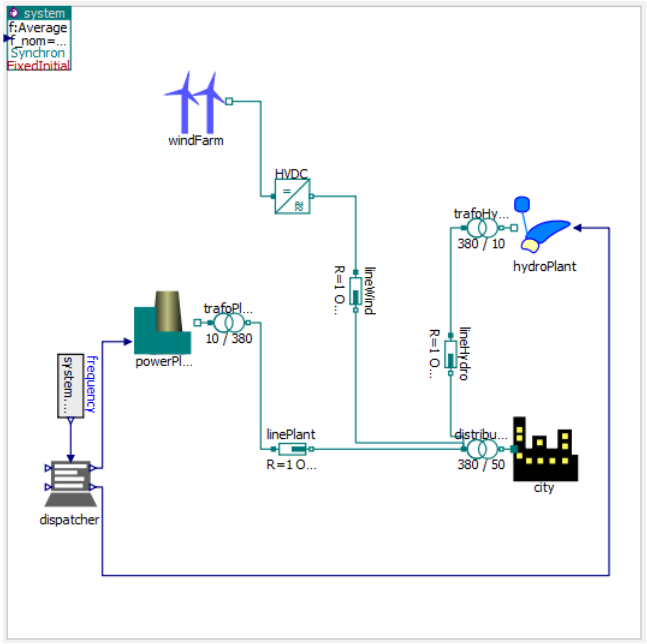

Fig. 6: A power system network from *PowerSystems.Examples.PowerWorld* composed of a power plant, wind farm, a hydro power plant and a city

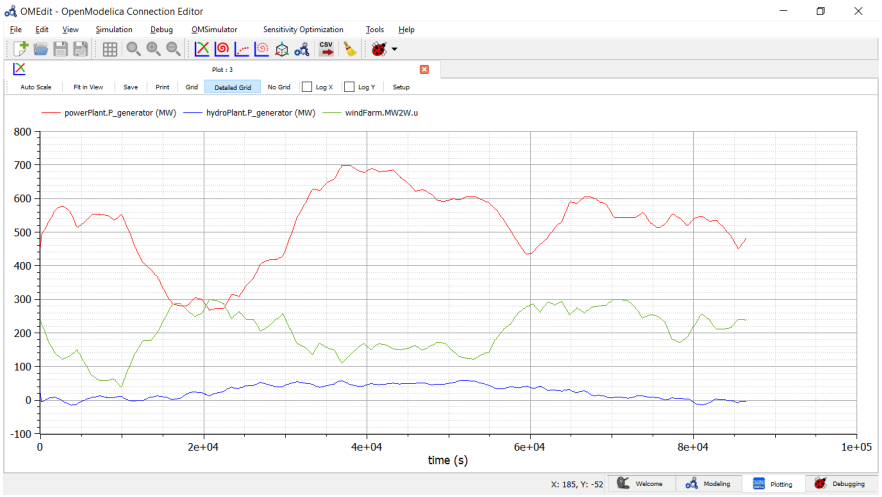

Fig. 7: Power generation of the plants

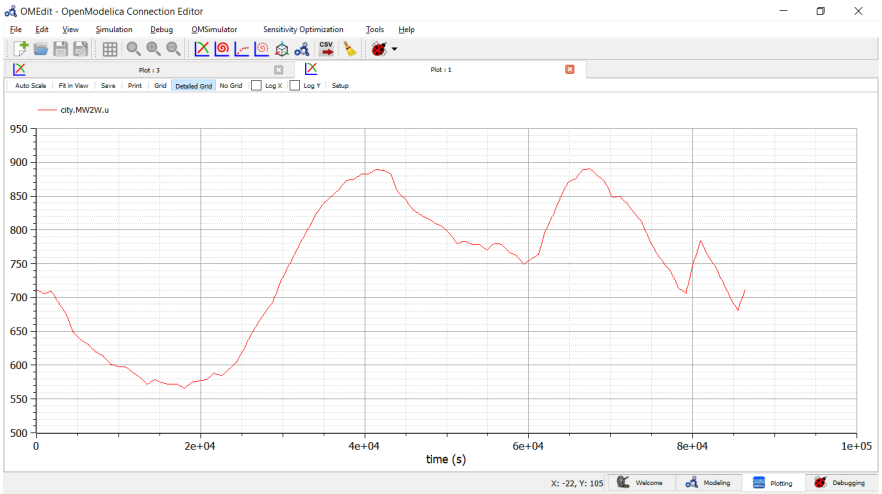

Fig. 8: Power consumption of the city

VII. Current State of Modelica

A. *Language specification*

While the first version of the language specification released September 1997 focused on continuous-time systems, the specification is continuously subject to improvement and enhancement, cf. www.modelica.org/documents. The evolve of specification versions is motivated by

1) attempting a unique interpretation of the language among various tool vendors and
2) providing further improved and new features and capabilities

A sample of established new features in the past were for describing and supporting:

- discrete systems (versions 1.1, 1.2 and 1.4)
- external interfaces to C/Fortran routines (V1.2)
- array semantics (V1.3, V2.0)
- enumeration types (V2.0)
- hybrid paradigms and state machines (Ferreira and Estima de Oliveira, 1999; M. Otter and Årzén, 2005) (V2.2)
- stream connectors exhibiting bi-directional flow (Franke et al., 2009a) (V3.1)
- operator overloading (Olsson et al., 2009) (V3.1)
- synchronous controllers (Elmqvist, 2012) (V3.3)

The maintenance and progress of the language design among other issues are initiated through well-defined procedures and are discussed in periodic meetings organized by MA typically 3-4 times a year, cf. https://modelica.org/events/design_meetings. Decisions are officially electable by MA members in a transparent way. Organizational and individual membership of MA is practically open for any one or organization though under the conditions announced in https://modelica.org/association.

B. The Modelica Standard Library

An open-source Modelica Standard Library (MSL) was developed and continuously maintained and enhanced by MA. The library comprises a large set of various model components in many physical domains, cf. Table I.

TABLE I: Overview of some (sub)packages within MSL

(Sub)Package	Description
Blocks	Basic components for building block diagrams:
*.Interface	Connectors and abstract models for I/O blocks
*.Math	Mathematical functions as I/O blocks
*.Continuous	Basic continuous-blocks described by DAEs, e.g. Integrator, Derivative, PI & PID controllers, etc.
*.Noise (Klökner et al., 2014)	Random noise generators / statistical distribution

*.Sources	Signal generation blocks
*.Tables	Interpolation of one- & two-dimensional tables
Other subpackages:	Logical, IntegerMath, MathBoolean, Nonlinear, ...
Electrical	Diverse electrical components:
*.Analog (Clauß et al., 2000; Majetta et al., 2009a)	Basic electrical components s.a. resistors, capacitors, inductors, simple transistors and diodes, heat storage and dissipation, etc. (Cellier et al., 2007)
*.Machines (Kral and Haumer, 2005)	Electric machine models including loss effects from various sources (Haumer et al., 2009) with consistent thermodynamically established concept.
*.QuasiStationary (Haumer et al., 2008)	Quasi-stationary analysis with sinusoidal excitation
*.Spice3 (Majetta et al., 2009b, 2011)	General devices (resistors, capacitors, inductors, sources) and semiconductor devices
*.*Converters	Subpackages for converting ACDC, DCDC, ...
Other subpackages	Digital, Multiphase, PowerConverters, ...
Fluid (Casella et al., 2006; Franke et al., 2009b)	Zero- and one-dimensional thermo-fluid flow components within subpackages: Vessels, Pipes, Machines, Valves, Dissipation, etc.
Magnetic	Magnetic components in couple of subpackages s.a.
*.FluxTubes	Electromagnetic devices with lumped magnetic networks
*.Fundamental-Wave (Kral and Haumer, 2011b)	Rotating electrical three-phase machines relying on magnetic potential and flux fundamental waves
Math	Mathematical functions operating on matrices

Mechanics	Diverse one- and three-dimensional mechanical components within the subpackages: Multibody, Rotational Translational
Media (Elmqvist et al., 2003)	Definitions for ideal gases, mixtures, water & air models incompressible & compressible media etc.
Thermal	Thermodynamics within many packages s.a.
*.HeatTransfer	Heat transfer
*.FluidHeatFlow	Thermo-fluid pipe flow
SIunits	Type and unit definitions based on SI units
StateGraph (M. Otter and Årzén, 2005)	A framework for modeling discrete event systems in a structured way via finite state machines based on Grafcet principles (David and Alla, 1992) with capabilities for representing state charts (Årzén, 1996). Earlier attempts were conducted (Mosterman et al., 1998; Ferreira and Estima de Oliveira, 1999; Elmqvist et al., 2001).

MSL can be directly employed or extended by modelers for developing their own libraries and applications. MSL, being developed by experienced modelers, is subject to intensive discussion and regression testing. MSL also acts as instructive guidelines for tackling modeling tasks using the state of the art Modelica-based design techniques. The progress of the library state can be monitored through the github website https://github.com/modelica/ModelicaStandardLibrary. Justified contributions in the form of pull requests are encouraged.

C. The functional mockup interface

Due to the increasing demand of model-based technologies in various application domains there is no need to follow an ad-hoc approach for involving Modelica-based components in external model-based applications, e.g. model predictive control, embedded systems and co-simulation among others. In contrary, utilization of the so-called Functional Mockup Interface

(FMI) technology (Blochwitz et al., 2011) is the standard approach. This is achieved by exporting Modelica models as well as other dynamic modeling tools to a simulation program in C that follows a unified API, recommended by FMI, cf. https://fmi-standard.org/tools/ for an actual list of tools supporting FMI.

The FMI-compliant exported model together with a descriptive XML file is referred to as Functional Mockup Unit (FMU), cf. Figure 9. The FMU can be independently simulated or

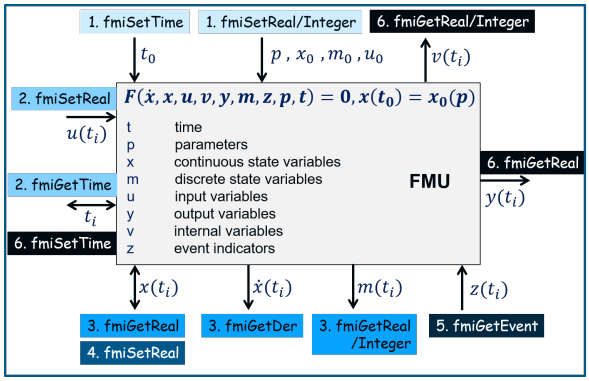

Fig. 9: Functional mockup unit: Mathematical representation and some C-functions operating on them

integrated into other simulation platforms.Advanced numerical solvers requiring the Jacboian for step-size adaptive numerical integration have been served since the second version of FMI specification (Blochwitz and et al., 2012). This version recommends exported FMUs to provide directional derivatives, which even allows the evaluation of parameter sensitivities for model-based predictive control (Franke et al., 2015) among other possible applications.

The functionalities demonstrated in Figure 9 refers to *FMI for model exchange*. Another flavor is to allow the exportation of a bundeld numerical solver within the FMU. This is referred to as *FMI for Co-simulation*. Using FMI standards, generic application-independent uniform solutions for employing FMI-based models in many model-based applications can be followed, e.g. (Pfeiffer et al., 2012; Vanfretti et al., 2014). Particularly, the utilization of external design and optimization tool becomes a software engineering task e.g. (Vanfretti et al., 2016a).

Still there is a need for a further level of standardization for interacting FMUs, e.g. in power system modeling applications (Gómez et al., 2020). When describing a complex system comprising multi-physical phenomena, it is ideal to have the whole system described in only one language. This is however almost impossible due to many reasons. First many other well-established long-rooted simulation tools optimized for specific domains exist. Their involvements in system modeling is highly beneficial. The in-depth accumulation of expertise with such domain-specific tools can not be excluded. Thus, a practical solution that enables the collaboration of different departments is to employ co-simulation where each tool export its models in FMUs. It is beneficial to design such an FMI-based co-simulation of a technically complex system in a standard manner allowing the orchestration among multi FMI-based simulations. For this purpose, there are two FMI-related projects maintained by MA for standardizing the orchestration of FMI-based co-simulations:

1) System Structure and Parameterization (SSP): .

SSP proposes an XML-based specification for describing the system FMU-based components and their interrelations (Köhler et al., 2016). Moreover, the specification covers also the overall parameterization in a consistent rather than isolated manner. This is particularly useful for MiL, SiL and HiL applications.

2) Distributed Co-simulation Protocol (DCP): While SSP is more related to the modeling and descriptive part of a complex system, DCP is more related to the technical part of the co-simulation including interoperability aspects in a heterogeneous system (Krammer et al., 2019). DCP follows a master-slave architecture along a finite discrete state machine characterizing the operation of the system. A standardized I/O data exchange model and description for communication protocol among slaves are provided.

D. Modelica simulation environments

Many mature Modelica simulation environments, commercial and open-source exist, typically enhanced with additional services for model design and optimization, cf. www.modelica.org/tools. Large set of documentation, free and commercial books are available in couple of languages, cf. www.modelica.org/publications.

E. Projects

The increasing significance of Modelica-based technologies can be best reflected by the amount of involvement in research projects and organized conferences. Table II lists some large-scale research industry-oriented projects primarily for developing and improving the Modelica language and its related technologies, cf. www.modelica.org/external-projects for an actualized list.

TABLE II: Summary of some large-scale research projects around Modelica-based technologies

Project	**Budget** M €	**Objective**
(EUROSYSLIB, 2007-2010)	16	Modelica libraries for embedded systems
(MODELISAR, 2008-2011)	27	Developing FMI standards (www.fmi-standard.org)
(PEGASE, 2008-2012)	13.59	Defining the most appropriate state estimation, optimization and simulation for power grid (Villella et al., 2012; Chieh et al., 2011)
(OPENPROD, 2009-2012)	11	Integration of open model-driven M&S environments with industrial applications
(DYNCAP:, 2011-2014)		Dynamic operation of stream power plants with CO2 capture technologies
(MODRIO, 2012-2016)	21	Extending M&S tools from system design to system operation
(ITESLA, 2012-2016)	19.43	Unified Modelica-based tools for pan European security assessments of large-scale power systems
(Annex 60, 2012-2017)		New generation computational tools for building and community energy systems
(TransiEnt, 2013-2017)		Efficient integration of renewable energies into urban energy systems
(ACOSAR, 2015-2018)		Advanced co-simulation interface for offline and online system integration
(OPENCPS, 2015-2018)		Verification/Validation and interoperability of key standard technologies (Modelica, SysML, FMI,...)
(DYNSTART, 2015-2019)	0.234	Startup and shutdown processes of steam power plants with fluctuating share of renewable energy
(INDIGO, 2016-2020)	2.79	More-efficient next generation of district cooling systems (Costa et al., 2020)
MISSION (2016-2024)		Modeling and simulation tools for more-electrical aircraft design (Valdivia-Guerrero et al., 2016)
(EMPHYSIS, 2017-2020)	14	New FMI standards for embedded systems, e.g. electronic control units and micro controllers
(ResiliEntEE, 2017-2021)		The resilience of coupled energy systems in the presence high share of renewable energy
(IBPSA, 2018-2022)		Modelica framework for building and community energy system design and operation (Wetter et al., 2019)

F. Conferences and user groups

Despite the fact of Modelica is viewed as an European product, the internationalization of Modelica through increasing activities in Asia and North America are remarkably present in the form of active user groups and periodic user meetings, cf. www.modelica.org/users-groups. Meanwhile, there are periodically organized international conferences in three continents, cf. www.modelica.org/events:

1) The 13^{th} European version of the conference attracted couple of hundreds of participants largely from the industry with nearly one hundred peer-reviewed accepted papers
2) The first Japanese international conference was held in Tokoyo May 2016
3) The first North American Modelica conference was held in Cambridge October 2018

Despite of the actual state of pandemics, latest Modelica conferences were successfully held in a vertual manner.

G. Modelica Association Membership

MA memberships are practically provided for any interested party whether a tool provider or an end user after accomplishing some fair conditions published in the bylaws, cf. https://github.com/modelica/MA-Bylaws/releases. The memberships enables to take a part in the maintenance and progress of the Modelica language and its projects in a coordinated and organized manners according to the published bylaws. There are two forms of memberships:

1) Organizational memberships for universities, research institutes and companies, cf. https://modelica.org/association for an actual list of organizations
2) individual membership for persons

The MA organization board taking care of the association common activities is periodically elected. Their members and duties are listed in the given previous website.

H. Modelica Newsletters

Newsletters on the Modelica language, tools, libraries, etc. is periodically published, typically three times a year. The contents are collaboratively written by many parties through pull requests at github, cf. https://github.com/modelica/newsletter. Already published newsletters and email subscriptions can be obtained at https://newsletter.modelica.org/

VIII. Open-source Modelica libraries

A large growing set of third-party open-source libraries exist and they get monitored whenever an update takes place (Tiller and Winkler, 2014), cf. www.modelica.org/libraries or https://github.com/modelica-3rdparty. Most of these libraries are distributed under Modelica BSD-like licenses, cf. https://www.modelica.org/licenses for more details. To the best of our knowledge, we provide a list of most existing open-source libraries that are / can be useful for power system modeling applications. In this section, a quick overview of existing open-source libraries are summarized.

A. Power systems libraries

Power system modeling is an active application area within the Modelica community since an early stage of the Modelica language (Bachmann and Wiesmann, 2000). The MSL provides some primary capabilities based on first-principle models (e.g. resistors) allowing construction of power system networks, cf. Table I. Table III provides an overview of existing open-source Modelica libraries for various power system modeling applications including classical power network modeling. The underlying assumptions would usually assume that (Casella et al., 2017):

1) three-phase voltages and currents are always balanced, and can thus be described through the phasor approach (Steinmetz, 1893; de Araujo and Tonidandel, 2013)
2) network frequency remains within an acceptable range around its reference value and thus a constant impedance is considered
3) a fixed network topology is assumed throughout network simulation

These libraries commonly realize most of the following aspects:

- capabilities of browsing, reusing, modifying or extending existing model components
- flexible way of modeling, e.g. acausal modeling principles make networks construction very cumbersome, by simply connecting loads, busses, generators and lines together rather than relying on traditional admittance matrix methods
- uniform framework gathering detailed mechanical power generation (e.g. wind turbines), thermal aspects (e.g. PVs and thermal flow), weather forecast data, classical grid components, control components and strategies, faults, switches and events all together
- flexible mixing of different components from different libraries

Special attention has been neutrally devoted to the *PowerSystem* library. This library additionally realizes the following unique advantages:

- can provide several analysis aspects only in one package including detailed dynamic transient analysis, quasi-steady state and steady state (i.e. transient and voltage stability) analysis and power flow calculations.‘
- Uniform treatment of phase-systems (DC, AC one-phase, AC three phases). Modelers will be able to easily switch between different phase systems.
- Uniform treatment of different coordinate representation (e.g. natural abc representation, model dq- and dq0 representations). In other words, modelers will be able to easily simulate the same models with different representation of phase systems.

More detailed comparisons among some of the listed Modelica libraries can be found in (Winkler, 2017).

TABLE III: Open-source Modelica libraries in power system modeling applications

Clara (Brunnemann et al., 2012; Gottelt et al., 2017) www.claralib.com/	Thermo-hydraulic behavior of stream power plants with coal dust firing with focus on technologies for carbon capture and its storage. Processes governing fuel handling, cooling of combustion chamber and electrification of the stream in turbo generators among others are treated. Components include heat exchangers, pumps, valves, reservoirs, turbines, compressors, absorbers, steam accumulator (Richter et al., 2019), gases and solvents. Transient analysis of power outputs can be performed. Capabilities for controller testings and investigation of start-up and shut-down processes are provided (Marx-Schubach and Schmitz, 2019).
ComplexLib (Enge et al., 2006)	Steady-state analysis of electro-mechanical drives with electrical AC subsystems employing the phasor method
ModPowerSystems (Minz et al., 2016)	A modeling framework facilitating the increasing employment of power electronics in power systems. Capabilities for static and dynamic phasors simulation as well as electromagnetic transient are present.

ObjectStab (Larsson, 2000)	Modeling transient and voltage stability analysis. Components include generators as slack or PV nodes, 3rd or 6th order dq-models of excitation and governor control systems, transmission lines, reactive power compensation devices; shunt reactors, shunt capacitance and series capacitance, transformers, static and dynamic loads, buses and faulted lines among others. Although maintenance and progress is inactive, the library is suitable for educational purposes as it does not exploit later established advanced language features.
OpenIPSL (Vanfretti et al., 2016b; Baudette et al., 2018)	Validated set of power-system components based on phasor-time approach validated against reference models from common power system tools. Power system models, including common known bus-systems models, electro-mechanical transients comprising power generation, transmission and consumption together with equipment components for control and stabilization can be constructed. Initial guesses corresponding to power-flow simulations are provided through other simulation tools. The library is not only suited for realistic applications but also for educational purposes (Murad et al., 2017).
PhotoVoltaics (Brkic et al., 2019)	Photovoltaic components including cells, modules and plants, enhanced with power converters based on MSL components among other features. Validation was conducted using selected industrial module data sheets. (Brkic et al., 2019) includes a good summary on the features of other PV libraries and their limitations.

PowerGrids (Bartolini et al., 2019)	Modeling of electro-mechanical phenomena in power systems including power generation and transmission targeting European transmission system and distribution system operators. Power flow analysis is expressed as initial equations which solution is the initial values of the power system network. Providing a good-initial guess, transient stability analysis can be conducted. The library was used for feasibility studies of realistic large-scale networks with thousands of nodes, generators and lines represented by over half-million equations (Casella et al., 2017).
PowerSystems (Franke and Wiesmann, 2014)	Arbitrary phase systems in one single framework covering a large set of components for DC and three-phase AC. Steady-state, transient and asymmetric analysis can be conducted. An extensive overview is presented in Sections **??** and V.
SPOT	One of the earliest libraries for power systems with extensive set of model components suited for detailed modeling of transient effects (Bachmann and Wiesmann, 2000). Its implementation has influenced many subsequent libraries including the `PowerSystems` library. SPOT provided a uniform treatment of natural (abc) and modal (dq0) coordinates in one framework. Per-unit system for parameterization is supported (Franke and Wiesmann, 2014). Although current development is inactive, its contents has educational values as it is using less advanced language features that did not exist in earlier versions of the Modelica language. Actually, before implementing a new library, it is encouraged to explore and learn from the implementation.

SolarTherm (la Calle et al., 2018)	Typical components, with varying degree of complexity, for building a solar thermal power plant. Components include receivers, plates, pumps, heat-transfer, fluids, energy storage, power generation, whether data, control. The library is associated with a range of Python-based tools and scripts that automate the process of testing, simulating, optimising and visualising the results.
ThermoPower (Casella and Leva, 2006; Schiavo and Casella, 2007)	Dynamic modelling of thermal power plants with components for water and gas properties. Heat exchangers and electric generators are included. The library has been particularly useful for the study of control systems in traditional and innovative power plants and energy conversion systems including steam generators, coal-fired power plants, combined-cycle power plants, nuclear power plants, solar plants, organic Rankine cycle plants, and cryogenic circuits for nuclear fusion applications.
TransiEnt (Andresen et al., 2015, 2019; Heckel and Becker, 2019)	Coupled energy supply grids: electricity, district heating and gas grids (Andresen et al., 2018) with attention in energy storages in the presence of fluctuating renewable energies (Dubucq and Ackermann, 2017; Bode and Schmitz, 2018). Components include conventional and renewable power plants, electrical and thermal loads, buildings, districts, cities, electric- heat- and gas distribution elements and storages. Conversion technologies s.a. power-to-gas (involving electrolyzer (Webster and Bode, 2019)), power-to-heat, gas-to-heat and gas-to-power are tools for enabling the transition to renewable energy source powered society (Bode and Schmitz, 2018).

WindPowerPlant (Eberhart et al., 2015)	Simulation of wind power plants with components for wind turbines, generators and control

B. Energy in buildings and/or districts

Modern energy systems modeling applications may view buildings as a part of the electrical grid. They are not only energy consumers, but can be also energy producers within the electrical grid. While building modeling applications are usually focused on thermal behavior, still small-order building models can be employed in the context of large-scale energy simulation at district-level (Frayssinet et al., 2017) including communication and transportation (Lu et al., 2019; Bollinger et al., 2018). The interactions between thermal, gas and electrical networks and the opportunity of storing excess of electrical power in a form of thermal energy and/or hydrogen among other possibilities make libraries from the buildings modeling community of a vital interest for modern power systems modeling applications. Table IV provides an overview of existing open-source libraries in the buildings/districts domain.

TABLE IV: Open-source Modelica libraries in building and district (thermal-)energy modeling

AixLib (Müller et al., 2016)	High and reduced order building models with focus on HVAC systems. It has been employed in the field of district heating and cooling networks (Mans et al., 2019) as well as in hardware-in-the-loop (Schneider et al., 2015) among others.
Buildings (Wetter et al., 2014) simulationresearch.lbl.gov	A comprehensive library with extensive set of components for modeling aspects of thermal energy behavior in buildings including HVAC, multizone heat transfer and airflow among others. The library is enhanced with a package interfacing and interacting with the electrical grid in two directions (Bonvini et al., 2014). Package includes weather modeling for renewable production, PVs, wind turbines among others.

BuildingSystems (Nytsch-Geusen and et al., 2016)	Main focus on thermal energy (heating and cooling) in buildings and districts. Building models range from simplified 0-dimensional low-order model to 3-dimensional multi-zones models. Components for HVAC, solar thermal systems and photo-voltaic systems are provided.
DisHeatLib (Basciotti and Pol, 2011)	Modeling thermal energy aspects (Heating, cooling, pipes, storage, control, etc.) in districts with optional interfaces to the electrical power grid (Leitner et al., 2019).
DCOL (del Hoyo Arce et al., 2018) https://zenodo.org/record/818289	District heating and cooling systems including distribution networks and thermal energy storage. The library was used in establishing a virtual test-bed of a district cooling production plant with compressor and absorption chillers (Zabala et al., 2020). Components include water storage, pipes, multi-zone air-flow, actuators, dampers, valves, radiators, heat pumps, etc.
FastBuildings (Baetens et al., 2015)	low-order building models employing common components of electrical circuits ideal for smart grid modeling applications
GreenHouse (Altes-Buch et al., 2019)	Modeling greenhouse climate including indoor climate, ventilation, climate control, generation and storage units (Combined Heat and Power (Altes-Buch et al., 2018) and heat pumps), thermal energy storage and corp yield.

IDEAS (Jorissen et al., 2018a,b)	Main focus is on district thermal energy simulations with interfaces to electrical systems, for instance the impact of PV and heat pumps on low-voltage electrical grids can be investigated (Protopapadaki and Saelens, 2017, 2019).
modelica-ibpsa (Wetter and et al., 2015)	Basic interfaces and components common for modeling applications of energy of buildings. It provides guidelines and standardization for advanced Modelica libraries in the Energy in Buildings domain. By relying on this standard, it easier to exchange components among various Modelica libraries targeting different scope of applications. There are many libraries listed in this Table based on modelica-ibpsa.

C. Useful libraries

Table V provides a list of some 3rd-parties open-source Modelica libraries providing fundamental methodologies and/or tools for useful purposes. These libraries are not directly related to modeling of power systems but still they can be useful in some aspects. Many of these libraries are actively maintained, but few of them are discontinued. Thus, for those who are involved in similar activities, they can

- build on these efforts rather than starting from scratch
- directly contribute to these libraries and/or make these efforts active again

TABLE V: Open-source Modelica libraries potentially useful for some Power Systems Modeling applications

ABMLib (Sanz et al., 2018)	Agent-based modeling framework. Source can be obtained under www.euclides.dia.uned.es.

ADMSL (Elsheikh, 2014a)	A demonstrative example for computing dynamic parameter sensitivities of Modelica models using equation-based algorithmic differentiation techniques (Elsheikh, 2015)
AdvancedNoise (Klöckner et al., 2017)	Additional features to the existing Noise library within MSL incorporating more stochastic distributions and tools
AlgebraTestSuite (Sielemann et al., 2013)	Models for testing initialization problems of typical large-scale differential algebraic equations using a proposed initialization heuristic (Sielemann and Schmitz, 2011; Sielemann, 2012)
CellularAutomataLib (Sanz et al., 2014a, 2016)	Describing dynamic phenomena dependent on spatial coordinates. One- or two-dimensional cellular automates are present.
DESLIB (Sanz et al., 2012) www.euclides.dia.uned.es	Discrete-event systems following DEVS formalism. This has been used to reproduce some parts of the SIMAN language (Sanz et al., 2013). A library corresponding to the arena simulation language (Prat et al., 2006) suited for industrial dynamics (Forrester, 1961), e.g. describing processes of factory automation, is included.
ExternalMedia (Casella and Richter, 2008)	Enabling inclusion of external fluid propery code in Modelica compatible to interfaces provided by MSL. Focus is devoted on two-phase single-substance fluids.

FailureMode (Pop et al., 2012)	A collection of failure modes assisting modelers to systematically recognize the root-cause of common runtime errors through useful debugging information by a Modelica simulation tool. Typical errors include incorrect parameterization, invalid initialization, boundaries violation among others. This is particularly useful in the development stage.
FaultTriggering (van der Linden, 2014)	proposed standardization for triggering common faults during simulation runtime. This is realized by fault-output blocks attached to model components to trigger user-chosen faults.
FMITest	Testing connections of FMUs particularly during initialization and iteration
LinearMPC (Hölemann and Abel, 2009)	Model-based predictive control for linear processes
Modelica_DeviceDriver (Thiele et al., 2017; Bellmann, 2009)	Allows access to hardware devices such as input devices (keyboard, 3D-connecion, SpaceMouse, etc.) within Modelica models. This is particularly useful for Human-in-the-Loop and/or Hardware-in-the-Loop simulators as well as control applications. Part of the library is included in MSL 4.0.0.

Modelica_ LinearSystems2 (Otter, 2006; Baur et al., 2009)	Data structures for linear time-invariant differential and difference equation systems as well as polynomial-based operations on complex numbers. Typical operations on these structures are provided enabling description of transfer functions for common linear control analysis.
Modelica_Requirements (Otter and et al., 2015)	Formal modeling of requirements and verifying them during simulation
Optimisers	Specification of dynamical optimization using Modelica components. The optimization problem is solved using external solvers
ModelicaDEVS (Beltrame and Cellier, 2006)	A realization of discrete event system simulation formalism (Ziegler, 1976) in Modelica, cf. Subsection IX-B3
SystemDynamics (Cellier, 2008)	Modeling various dynamical phenomena in many areas including economics, industry, environments, according to the principles of system dynamics of J. Forrester (Forrester, 1969, 1971a,b)
ThermalSeparation (Joos et al., 2009)	Gas-separation, absorption, adsorption and rectification processes including heat and mass transfer within mixed-phases mediums (Marx-Schubach and Schmitz, 2019).

XogenyTest	Unit testing of Modelica models and libraries using Modelica components

IX. Scalability and runtime performance

A. Limitations

Energy power systems together with their boundary conditions are naturally of decentralized large-scale nature. This aspect is further emphasized by the current transformation of the energy systems, where smaller units (e.g. active buildings, wind energy, farms, solar farms etc.) enhance the complexity of the system. Currently, multi-domain universal simulation environments may suffer from scalability issues when enlarging the modeling size and/or considering low-level detailed micro aspects, e.g. the charging of an electrical car in a smart home within a smart grid. In (Palensky et al., 2012), a demonstrative smart grid model comprising various common concepts:

- Physical world / continuous models: power generation, consumption, heating/cooling, etc.
- Stochastic aspects: weather prediction, stochastic distribution for collective description of individuals, etc.
- Information technology / discrete models: controllers, communications, etc.
- Smart agents: human behavior, management systems, energy markets, etc.

It was found that such large-scale models badly scale up with Modelica-based environments (nearly $O(N^3)$) (Palensky et al., 2013). Moreover, due to excessive memory requirements, maximum number of state variables in a Modelica simulation was in the order of $O(10^4)$.

In order to identify the reasons behind the low scalability of such models, it is notable that object-oriented models are of sparse nature. Every model typically consists of hierarchical composition of subsystems each with a limited set of components. Each of these components is connected to only few components. When such a model translates to a system of equations, each equation consists of few variables. Thus, the Jacobian, corresponding to the partial derivatives of equations w.r.t. state variables, is largely sparse with high-amount of zeros, typically $> 99\%$ of the entries are zero (Casella, 2015a). By comparing the runtime performance of both of acausal modeling and multi-agent based modeling paradigms (Palensky et al., 2013), the reasons behind such low

runtime performance of Modelica environments can be deduced as follows (Casella, 2015a):

1) Translation to one single big block of equations: Modelica models get usually translated to one single ODE/DAE block and not as a cascaded system of much smaller ODEs/DAEs like the case with agent-based modeling paradigms. There could be some isolated blocks of linear and nonlinear equations that would be separately evaluated. However, the resulting ODE/DAE block would be usually large and computationally dominant.

2) Single-rate numerical integration: For large-scale networked models, the resulting ODE/-DAE comprises phenomena that runs at very different runtime scales. The subsystem with the quickest dynamics (i.e. with smallest time constant) would cause a variable-step numerical integration scheme to pick a tiny integration step-size that is irrelevant for a large portion of the system with slower dynamics. While this common simulation paradigm guarantees a precise accuracy, still accuracy of agent-based modeling paradigms were reliable due to the highly loose-coupled nature of such models.

3) No exploitation of sparsity patterns: The resulting ODE/DAE would usually require an implicit numerical integration scheme due to its underlying stiffness. The corresponding sparse Jacobian is a part of the underlying nonlinear equation to be solved by a Gauss-Newton iteration scheme. Apparently until few years ago, Modelica simulation environments didn't consider exploiting sparsity patterns of Jacobian both at computation and memory level, at least in their provided default set of solvers. For example, describing the evaporation of water to steam in a tube for a plant model comprising 800 state variables takes nearly 2 hours with a Modelica simulation environment in comparison with 80 seconds by an in-house simulation tool making use of sparse solvers (Link et al., 2009). Algorithmic complexity of matrix inversion for solving a nonlinear equation system at every integration step is of order $O(N^3)$. The memory complexity for storing a dense Jacobain is of order $O(N^2)$, not even considering approaches relying on the Hessian matrix. A system with 10000 state variables would require 1.6 GB of storage (Casella, 2015a). This has also significant penalty on runtime performance due to the excessive transfer of data among the various levels of data storage and caches.

4) Insignificant local events cause tremendous computation: Low-level local events (e.g. opening a window in a flat) which are insignificant for large-scale portion of a smart grid model still cause large-scale intensive computation for identifying a single state-event (say automatic switching of a heater in a smart grid model). The underlying root-finding algorithm causes the evaluation of the entire equation system for couple of iterations for detecting each state event (Eich-Soellner and Führer, 1998). A state-event can even trigger further state-events complicating the evaluation process, i.e. chattering (Mehrmann and Wunderlich, 2009). Having a high number of events extremely slowdowns the runtime simulation performance.

B. Active research agenda for improving runtime performance

Achieving significant simulation runtime improvement is not impossible. In contrary, classical/modern sparse techniques for huge-scale linear and nonlinear equation systems have been established in the scientific computing community many decades ago. Apparently, Modelica tool vendors were largely focused on the tedious task of fulfilling a full support of the always growing set of language features with less attention on runtime performance capabilities. In 2015, Casella addressed some research directions for the Modelica community in order to overcome some addressed runtime performance obstacles including (Casella, 2015a):

1) Exploiting sparsity patterns and sparse solvers: Integration of sparse solvers can reduce the overall computational complexity of large-scale decentralized networked models from $O(N^{2.6})$ to $O(N^{1.3}) - O(N^{1.8})$ according to (Casella, 2015a). The OpenModelica simulation environment reported experimental integration of sparse solvers for linear and nonlinear systems (Braun et al., 2017). Accordingly, transient simulations of realistic power system network models with thousands of nodes, generators and lines were conducted with runtime performance comparable to domain-specific tools (Casella et al., 2017). The Dymola simulation environment recently reported the utilization of sparse solvers with emphasize on root findings for identifying state events (Henningsson et al., 2019). Other Modelica simulation tools are expected sooner or later to follow similar directions, if not already done. A comparative power system modeling benchmarking between Dymola and OpenModelica were recently reported (Dorado-Rojas et al., 2020).

2) Multi-rate numerical solvers: The main stream of numerical solvers follow a single-rate numerical integration scheme, where at every integration step the entire equation system is evaluated. On the other hand, employing multi-rate numerical solvers (Rice, 1960) for systems with mixed slow and fast dynamics including stiff systems can largely improve the performance of large-scale decentralized network models. Multi-rate numerical integration is an active research area, e.g. (Savcenco, 2009; Roberts et al., 2018), which has attracted the Modelica community, e.g. (Bonaventura et al., 2020).

In multi-rate solvers, numerical integration of the entire equation system takes place globally only at few time steps. However, for many sub-iterations, only those states variables that violate desired accuracy and/or stability threshold are numerically integrated at smaller step sizes. Practically every time step aims at the evaluation of different subset of state variables, while employing interpolation or extrapolation for the rest of variables. In (Casella, 2015b), employing common graph-based algorithms models are suggested for identifying strongly interrelated subsets of equations related to object oriented models. Each of such strongly connected equation subsystems is numerically integrated using smaller local step-sizes while the whole equation system is globally evaluated at larger step-sizes. A case study in (Rande and Casella, 2014) shows that the computational complexity of numerical integration via a multi-rate implicit solver was of a quadratic rather than a cubic order. Another approach for realizing multi-rate algorithms is by exploiting synchronous language constructs that allow the modeler to define different time clocks with different numerical solvers to each component of a system (Thiele et al., 2014). This approach however requires explicit specification of the subsystems.

3) Solvers for massive number of state-events: Quantized States System (QSS) methods (Zeigler and Lee, 1998), though have been first established in the context of distributed simulation, is one of the successful methods suited for networked models with a massive number of state events. The main idea behind QSS methods is that discritisization of state variables rather than time is considered. Quantized state variables follow piece-wise constant trajectories rather than continuous trajectories. In contrary to classical methods, not all states get updated in a synchronous manner, but only those violating ”quantum level crossings”. In other words, each state variable gets updated upon its own pace in an asynchronous manner. In such a dense evaluation scheme, zero crossings corresponding to state-events are not evaluated using a Gauss-

Newton iteration as the case with classical approaches, but they are simply predicted leading to a significant improvement of the overall computational performance, cf. (Kofman, 2004; Floros et al., 2010) for more details. Earlier QSS methods used explicit iteration schemes for numerical integration (Kofman and Junco, 2001). Second and third-order QSS methods were further established by letting states variables follow parabolic and cupic piece-wise constant trajectories, respectively (Kofman, 2002, 2006). Implicit QSS methods suited for stiff systems have been later established (Migoni and Kofman, 2009).

QSS methods have been integrated to the OpenModelica simulation environment (Bergero et al., 2012). Two benchmarks (Floros et al., 2014) corresponding to district cooling system (Ceriani et al., 2013) and a smart grid model (Elsheikh et al., 2012) demonstrated the power of integrating QSS to OpenModelica. In both benchmarks, the simulation runtime scale quadratically (from ~$5N^2$ to ~$6N^2$) with the well-known DASSL solver while it scales linearly (from ~N to ~$3N$) with QSS methods. In another benchmark (Bergero et al., 2018) concerned with stiff hybrid systems, several cases studies were evaluated involving demand-side centralized power management of a population of thermostatically-controlled air conditioners in buildings (Perfumo et al., 2012) as well as a centralized cooling power distribution system in a building adopted from (Ceriani et al., 2013). It was shown that Implicit QSS methods exhibit a scalabiltiy of a linear order, while classical solvers demonstrate a nearly quadratic runtime performance complexity. Similar conclusions have been drawn for a renewable hybrid energy system models where the presence of DC-DC converters, diods and transformers cause stiffness (Migoni et al., 2016).

Alternatively, techniques that attempt to solve a much smaller but significant set of equation systems for identifying state-events of large-scale systems have been established, e.g. (Bergero et al., 2018). This can be achieved by exploiting structural information of the model including the way equations are causally sorted (Höger, 2013; Sanz et al., 2014b). Further significant improvements in the context of stiff systems can be achieved by considering a mixed-mode solver that automatically switches between an implicit QSS method and a classical implicit solver based on numerical criteria (Di Pietro et al., 2020).

4) Hybrid modeling paradigms [TO COMPLETE]: a QSS integration-scheme can be imitated by decomposing a loosly-coupled hybrid systems into several subsystems and applying a

Modelica hybrid paradigm. follow a discrete event simulation (DEVS) scheme (Ziegler, 1976), see e.g. (Beltrame and Cellier, 2006).

5) Agent-based modeling paradigms [TO COMPLETE]: An agent-based modeling paradigm would allow decomposing a loosy-coupled hybrid systems into several smaller subsystems. Each of these subsystems is separately integrated, though communicating intermediate values among subsystems follow explicit specified conditions. By employing agent-based modeling paradigms rather than Modelica implementation (Palensky et al., 2013), it was possible to simulate systems with millions of state variables in reasonable amount of time using standard workstations (remember with Modelica it the number of state variables was at most $O(10^4)$). The scalabilty factor tended to be semi-linear in comparison with a quadratic order using Modelica.

6) Parallelization [TO COMPELETE]: Parallelization (Elmqvist et al., 2014, 2015)

- a heuristic based on numerical approach for decoupling a large-scale system, cf. Papadopoulos et al. Model separability indices for efficient dynamic simulation
- Achieve parallelism by 1. library calls 2. language constructs or 3. automatic parallelization, Sjölund
- By contrast, distributed modeling, where solvers can beassociated with or embedded in subsystems, and even com-ponent models, has almost linear scaling properties, Spe-cial considerations are needed, however, to connect thesubsystems to each other in a way that maintains stabil-ity properties without introducing unwanted numerical ef-fects. Sjölund

Further comprehensive and detailed discussion can be consulted in (Casella, 2015a).

X. Summary and Outlook

In this section, some significant advantages of Modelica in the context of future power systems modeling applications under the lights of the demonstration in previous sections are highlighted. Then some research future directions regarding Modelica applications and language features are summarized.

A. Advantages of the Modelica language

1) Object-oriented paradigm: Modelica enables rapid prototyping by supporting:

- high-level descriptive language elements allowing rapid prototyping as well as flexible modification of complex models and their components
- flexible organization of model components within libraries of browsable packages and subpackages
- graphical modeling facilities allowing elegant experimentation of submodel variants
- hierarchical multi-level modeling corresponding to different resolution levels possibly within hybrid paradigms enabling both bottom-up and top-down of model design

among others, cf. Section IV and Section V.

2) Domain-independent multi-physical modeling concepts: Future power systems incorporate various physical perspectives, including mechanics, thermodynamics, hydraulics, etc. Modelica supports various physical domains and meanwhile providing the capabilities for describing the interrelationships among these physical domains. This is realized by relying on universal domain-independent physical concepts rather than domain-dependent principles, cf. section III.

3) Advanced methods for efficient runtime simulation: Definitely, the large-scale dimensions of future power systems models enforce simulation tools to employ state-of-the-art methodologies for improving runtime performance. Typical Modelica simulation environments (strive to) support:

- advanced graph algorithms for simplifying and optimizing large-scale equation systems
- common and various standard solvers for numerical integration of hybrid ODEs/DAEs with adaptive error control and treating potential presence of stiffness
- features for stochastic simulations via MSL, cf. Subsection VII-B
- capabilities for (hybrid) modeling paradigms

Nevertheless, common Modelica simulation environments are still subject to active progress in order to efficiently handle modern large-scale power system applications, cf. Section IX.

4) Standardized (co-)simulation interfaces: Co-simulation (Gomes et al., 2018; Schweiger et al., 2020b) has been widely used in the last decade to interface power system simulators with communication network simulators (Müller and et al., 2016; Palensky et al., 2017). It is expected to play a significant role also at power level simulation when the need of highly specialized tools for specific subsection of the system does not allow the use of a single simulation environment. Employing Modelica models enable co-simulation via the standardized FMI specification, established under the Modelica Association Umbrella. Standardization allows

for general tool-independent methodologies to take place leading to significant efforts reduction when realizing co-simulation, (cf. Subsection VII-C).

5) Code generation capabilities: Code generation capabilities are fundamental to allow simulation execution on dedicated or embedded hardware platforms so providing a flexible way of incorporating high-level model components for real-time and hardware in the loop as well as for control applications (e.g. model-based predictive control). Again, the FMI standards allow Modelica models to be exported as FMI-compliant units (FMUs) as stand-alone simulation programs portable to any environments (Brembeck, 2019; Lange et al., 2021) including embedded systems (Bertsch et al., 2015) and control applications (Franke et al., 2015).

6) Considerable amount of open-source libraries in power-system (related-) domain(s): At the early phase of the evolve of Modelica, already first works for modeling specific aspects in power systems (e.g. transient simulations, steady state analysis, etc.) have been implemented (Bachmann and Wiesmann, 2000; Kalaschnikow, 2002). With the maturity Modelica specification has achieved with respect to object oriented facilities, further modern power systems libraries have provided novel solutions that cover many power systems modeling applications in one single framework, cf. Subsection VIII-A.

7) Further useful open-source libraries: In addition to modeling libraries concerned with classical power systems modeling applications, Section VIII summarized additional libraries that cover various aspects, e.g. renewable energy, buildings and districts, required for realizing modern power systems modeling applications. The Modelica Standard Library, actively maintained and progressed by the Modelica Association, contains a considerable amount of useful components, cf. Subsection VII-B. Additionally, a considerable amount of third-party covering fundamental modeling paradigms up to optimization and unit testing exist, cf. Subsection VIII-C.

8) Modelica for power system modeling applications: For an extended discussion regarding the suitability of Modelica for power system modeling applications, the reader is conferred to (Gómez et al., 2020). Francisco J. Gómez and et al. have extensively discussed and highlighted the feasibility of the Modelica language for fulfilling and formalizing functional and technical requirements addressed by the European Network of Transmission System Operators of Electricity (ENTSO-E), cf. www.entose.eu/Pages/default.aspx).

B. Challenges and Future directions

In addition to the increasing demand on improving simulation runtime performance of large-scale systems, cf. Section IX, progress of current state-of-the-art and ongoing research and activities can be categorized as follows:

1) Enabling and developing further modeling libraries, in particular for cyber physical systems (Zimmer, 2016), e.g. stochastic, communications, control etc. with which Modelica wins further ground in the potential scope of applications
2) Conceptualization and implementation of simulation-based mathematical algorithms and tools with which model-based applications can be realized (Elsheikh, 2014b, 2015)
3) Realization of model-based applications in particular with FMI e.g. (Gräber et al., 2012; Nouidui and Wetter, 2014)

Further research topics attempt to extend the Modelica language with additional capabilities not supported yet. These are non-standardized attempts for inserting additional language features for describing[3]:

- Partial Differential Equations (Saldamli et al., 2002; Šilar et al., 2018)
- Numerical optimization problems (Åkesson, 2008)
- Uncertainty quantification (Bouskela et al., 2011)
- Parallelization (Gebremedhin et al., 2012)
- Variable structure dynamics (Zimmer, 2013; Benveniste et al., 2019)
- Message passing communication for DEVS formalism (Sanz and Urquia, 2016)
- Stochastic differential equations incorporating Wiener and Poisson generators externally called from Modelica code (Gevorkyan et al., 2016) [4]
- Julia-based Modelica package (Elmqvist et al., 2017)
- Modelica-based meta programming for symbolic computations, model transformation, compiler specification among others (Fritzson et al., 2019)
- Model selection (Brüger, 2019)
- Dynamic parameter sensitivities (Elsheikh, 2019)

[3]Items is sorted according to publication date of the chosen reference (not necessarily the first-ever possible reference)

[4]The Modelica libraries Noise and `AdvancedNoise` are also acknowledged for modeling of stochastic processes

REFERENCES

D. Abel, *Object-Oriented Modelling for Simulation and Control of Energy Transformation Processes*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 327–344.

ACOSAR, "Advanced Co-simulation Open System ARchitecture," https://itea3.org/project/acosar.html, 2015-2018.

J. Åkesson, "Optimica an extension of Modelica supporting dynamic optimization," in *Modelica'2008: the 6th International Modelica Conference*, Bielefeld, Germany, March 2008.

Q. Altes-Buch, S. Quoilin, and V. Lemort, "Modeling and control of chp generation for greenhouse cultivation including thermal energy storage," in *In Proceedings of the 31st International Conference on Efficiency, Cost, Optimization, Simulation and Environmental Impact of Energy Systems*, no. 13, Guimaraes, Portugal, Jun. 2018.

——, "Greenhouses: a Modelica library for the simulation of greenhouse climate and energy systems," in *Modelica'2019: The 13th International Modelica Conference*, Regensburg, Germany, Mar. 2019.

A. Andresen, P. Dubucq, R. P. Garcia, G. Ackermann, A. Kather, and G. Schmitz, "Status of the TransiEnt library: Transient simulationof coupled energy networks with high share of renewable energy," in *Modelica'2015: The 11th International Modelica Conference*, Paris, France, Sep. 2015.

——, "Status of the TransiEnt library: Transient simulation of coupled energy networks with high share of renewable energy," in *Modelica'2019: The 13th International Modelica Conference*, Regensburg, Germany, Mar. 2019.

L. Andresen, C. Bode, and G. Schmitz, "Dynamic simulation of different transport options of renewable hydrogen to a refinery in a coupled energy system approach," *International Journal of Hydrogen Energy*, vol. 43, no. 42, pp. 19 600 – 19 614, 2018.

Annex 60, "IEA EBC Annex 60: New generation computational tools for building and community energy systems based on the Modelica and Functional Mockup Interface standards," www.iea-annex60.org/index.htmll, 2012-2017.

K.-E. Årzén, "Grafchart: A graphical language for sequential supervisory control applications," *IFAC Proceedings Volumes*, vol. 29, no. 1, pp. 4831 – 4836, 1996, 13th World Congress of IFAC, 1996, San Francisco USA, 30 June - 5 July.

P. J. Ashenden, G. D. Peterson, and D. A. Teegarden, *The system designers guide to VHDL-AMS*.

Morgan Kaufmann, 2003.

A. Ashgar and et al., "An open source modelica graphic editor integrated with electronic notebooks and interactive simulation," in *Modelica'2011, the 8th International Modelica Conference*, Dresden, Germany, Mar. 2011.

K. J. Åström, H. Elmqvist, and S. E. Mattsson, "Evolution of continuous-time modeling and simulation," in *ESM'1998: The 12th European Simulation Multiconference - Simulation - Past, Present and Future*, Manchester, United Kingdom, June 1998.

D. C. Augustin, M. S. Fineberg, B. B. Johnson, R. N. Lineberger, F. J. Sansom, and J. C. Strauss, "The sci continuous system simulation language (cssl)," *SIMULATION*, vol. 9, no. 6, pp. 281–303, 1967.

B. Bachmann and H. Wiesmann, "Advanced modeling of electromagnetic transients in power systems," in *Modelica'2000: The Modelica Workshop*, Lund, Sweden, September 2000.

R. Baetens, R. D. Coninck, F. Jorissen, D. Picard, L. Helsen, and D. Saelens, "OPENIDEAS – an open framework for integrated district energy," in *Proceedings of BS2015*, Hyderabad, India, Dec. 2015.

A. Bartolini, F. Casella, and A. Guironnet, "Towards pan-european power grid modelling in Modelica: Design principles and a prototype for a reference power system library," in *Modelica'2019: The 13th International Modelica Conference*, Regensburg, Germany, 2019.

D. Basciotti and O. Pol, "A theotetical study of the impact of using small-scale thermo chemical storage units in district heating networks," in *The 2011 International Conference for Sustainable Energy Storage*, 2011.

M. Baudette, M. Castro, T. Rabuzin, J. Lavenius, T. Bogodorova, and L. Vanfretti, "OpenIPSL: Open-instance power system library – update 1.5 to itesla power systems library (ipsl): A modelica library for phasor time-domain simulations," *SoftwareX*, vol. 7, pp. 34–36, 2018.

M. Baur, M. Otter, and B. Thiele, "Modelica libraries for linear control systems," in *Modelica'2009: The 7th International Modelica Conference*, Como, Italy, 2009.

T. Bellmann, "Interactive simulations and advanced visualization with Modelica," in *Modelica'2009: The 7th International Modelica Conference*, Como, Italy, 2009.

T. Beltrame and F. E. Cellier, "Quantised state system simulation in Dymola/Modelica using the DEVS formalism," in *Modelica'2006: The 5th International Modelica Conference*, Vienna, Austria, Sep. 2006.

A. Benveniste, B. Caillaud, H. Elmqvist, K. Ghorbal, M. Otter, and M. Pouzet, *Multi-Mode DAE*

Models - Challenges, Theory and Implementation. Cham: Springer International Publishing, 2019, pp. 283–310.

F. Bergero, X. Floros, J. Fernàndez, E. Kofman, and F. E. Cellier, "Simulating modelica models with a standalonequantized state systems solver," in *Modelica'2012: The 9th International Modelica Conference*, Munich, Germany, Sep. 2012.

F. M. Bergero, F. Casella, E. Kofman, and J. Fernàndez, "On the efficiency of quantization-based integration methods for building simulation," *Building Simulation*, vol. 11, pp. 405–418, 2018.

C. Bertsch, J. Neudorfer, E. Ahle, S. S. Arumugham, Karthikeyan, Ramachandran, and A. Thuy, "FMI for physical models on automotive embedded targets," in *Modelica'2015: The 11th International Modelica Conference*, Paris, France, Sep. 2015.

T. Blochwitz and et al., "The functional mockup interface for tool independent exchange of simulation models," in *Modelica'2011: The 8th International Modelica Conference*, Dresden, Germany, Mar. 2011.

——, "Functional mockup interfae 2.0: The standard for tool independent exchange of simulation models," in *Modelica'2012: The 9th International Modelica Conference*, Munich, Germany, Sep. 2012.

T. Blochwitz, M. Otter, M. Arnold, C. Bausch, C. Clauß, H. Elmqvist, A. Junghanns, J. Mauss, M. Monteiro, T. Neidhold, D. Neumerkel, H. Olsson, J.-V. Peetz, and S. Wolf, "The functional mockup interface for tool independent exchange of simulation models," in *Modelica'2011: The 8th International Modelica Conference*, Dresden, Germany, Mar. 2011.

C. Bode and G. Schmitz, "Dynamic simulation and comparison of different configurations for a coupled energy system with 100 *Energy Procedia*, vol. 155, pp. 412 – 430, 2018, 12th International Renewable Energy Storage Conference, IRES 2018, 13-15 March 2018, Düsseldorf, Germany.

L. Bollinger, C. Davis, R. Evins, E. Chappin, and I. Nikolic, "Multi-model ecologies for shaping future energy systems: Design patterns and development paths," *Renewable and Sustainable Energy Reviews*, vol. 82, pp. 3441 – 3451, 2018.

L. Bonaventura, F. Casella, L. D. Carciopolo, and A. Ranade, "A self adjusting multirate algorithm for robust time discretization of partial differential equations," *Computers & Mathematics with Applications*, vol. 79, no. 7, pp. 2086 – 2098, 2020, advanced Computational methods for PDEs.

M. Bonvini, M. Wetter, and T. S. Nouidui, "A Modelica package for building-to-electrical grid

integration," in *BauSim'2014: The 5th BauSim Conference*, Aachen, Germany, Sep. 2014.

D. Bouskela, J. Audrey, Z. Benjelloun-Touimi, P. Aronsson, and P. Fritzson, "Modelling of uncertainities with modelica," in *Modelica'2011: The 8th International Modelica Conference*, Dresden, Germany, Mar. 2011.

W. Braun, F. Casella, and B. Bachmann, "Solving large-scale Modelica models: new approaches and experimental results using OpenModelica," in *Modelica'2017: The 12th International Modelica Conference*, Prague, Czech Republic, May 2017.

J. Brembeck, "A physical model-based observer framework for nonlinear constrained state estimation applied to battery state estimation," *Sensors*, vol. 19(20):4402, Oct. 2019.

J. Brkic, M. Ceran, M. Elmoghazy, R. Kavlak, A. Haumer, and C. Kral, "Open source PhotoVoltaics library for systemic investigations," in *Modelica'2019: The 13th International Modelica Conference*, Regensburg, Germany, 2019.

C. Brüger, "Modelica language extensions for practical non-monotonic modelling: on the need for selective model extension," in *Modelica'2019: The 13th International Modelica Conference*, Regensburg, Germany, 2019.

J. Brunnemann, F. Gottelt, K. Wellner, A. Renz, A. Thüring, V. Roeder, C. Hasenbein, C. Schulze, G. Schmitz, and J. Eiden, "Status of ClaRaCCS: Modelling and simulationof coal-fired power plants with co2 capture," in *Modelica'2012: The 9th International Modelica Conference*, Munich, Germany, Sep. 2012.

F. Casella and C. Richter, "ExternalMedia: A library for easy re-use of external fluid property code in Modelica," in *Modelica'2008: the 6th International Modelica Conference*, Bielefeld, Germany, March 2008.

F. Casella, M. Otter, K. Proelss, C. Richter, and H. Tummescheit, "The Modelica Fluid and Media library fpr modeling of incompressible and compressible thermo-fluid pipe networks," in *Modelica'2006: The 5th International Modelica Conference*, Vienna, Austria, Sep. 2006.

F. Casella, A. Leva, and A. Bartolini, "Simulation of large grids in OpenModelica: reflections and perspectives," in *Modelica'2017: The 12th International Modelica Conference*, Prague, Czech Republic, May 2017.

F. Casella, "Simulation of large-scale models in modelica: State of the art and future perspective," in *Modelica'2015: The 11th International Modelica Conference*, Paris, France, Sep. 2015.

——, "Efficient computation of state derivatives for multi-rate integration of object-oriented models," *IFAC-PapersOnLine*, vol. 48, no. 1, pp. 262 – 267, 2015, 8th Vienna International

Conferenceon Mathematical Modelling.

F. Casella and B. Bachmann, “On the choice of initial guesses for the newton-raphson algorithm,” *Applied Mathematics and Computation*, vol. 398, p. 125991, 2021.

F. Casella and A. Leva, “Modelling of thermo-hydraulic power generation processes using Modelica,” *Mathematical and Computer Modelling of Dynamical Systems*, vol. 12, no. 1, pp. 19–33, Aug. 2006.

F. E. Cellier, *Continuous System Modeling*. Springer Verlag, 1991.

——, “World3 in Modelica: Creating system dynamics models in the Modelica framework,” in *Modelica’2008: the 6th International Modelica Conference*, Bielefeld, Germany, March 2008.

F. E. Cellier and H. Elmqvist, “Automated formula manipulation supports object-oriented continuous-system modeling,” *IEEE Control Systems Magazine*, vol. 13, no. 2, pp. 28–38, 1993.

F. E. Cellier, C. Clauß, and A. Urquia, “Electronic circuit modeling and simulation in Modelica,” in *Congress on Modelling and Simulation*, Ljublijana, Sep. 2007.

F. E. Cellier and E. Kofman, *Continuous System Simulation*. Berlin, Heidelberg: Springer-Verlag, 2006.

N. M. Ceriani, R. Vignali, L. Piroddi, and M. Prandini, “An approximate dynamic pro-gramming approach to the energy managementof a building cooling system,” in *European Control Conference*, Zurich, Switzerland, Jul. 2013, pp. 2026 – 2031.

A. Chieh, P. Panciatici, and D. Picard, “Power system modeling in modelica for time-domain simulation,” in *2011 IEEE Trondheim PowerTech*, Trondheim, Norway, Jun. 2011, pp. 1–8.

C. Clauß, T. Leitner, A. Schneider, and P. Schwarz, “Modelling of electronic circuits with Modelica,” in *Modelica’2000: The Modelica Workshop*, Lund, Sweden, September 2000.

A. Costa, A. Bassani, J. Febres, S. López Perez, S. Herrero López, M. Rämä, K. Klobut, Y. Stauffer, R. Carrillo, G. Naveran, M. Keane, and R. Sterling, “Indigo project: A simulation based approach to support district cooling design and operation,” in *Proceedings of the 16th IBPSA Conference*, V. Corrado, E. Fabrizio, A. Gasparella, and F. Patuzzi, Eds. International Building Performance Simulation Association - IBPSA, 2020, pp. 1913–1920, 16th IBPSA International Conference and Exhibition, Building Simulation 2019, BS 2019 ; Conference date: 02-09-2019 Through 04-09-2019.

R. David and H. Alla, *Petri Nets and Grafcet: Tools for Modelling Discrete Event Systems*. Prentice Hall, 1992.

A. E. A. de Araujo and D. A. V. Tonidandel, “Steinmetz and the concept of phasor: A forgotten story,” *International Journal of Control, Automation and Electrical Systems*, vol. 24, pp. 388–395, Apr. 2013.

I. del Hoyo Arce, S. Herrero López, S. López Perez, M. Rämä, K. Klobut, and J. A. Febres, “Models for fast modelling of district heating and cooling networks,” *Renewable and Sustainable Energy Reviews*, vol. 82, pp. 1863 – 1873, 2018.

F. Di Pietro, J. Fernndez, G. Migoni, and E. Kofman, “Mixed-mode statetime discretization in ode numerical integration,” *Journal of Computational and Applied Mathematics*, vol. 377, p. 112911, 2020.

H. V. Dommel, “Digital computer solution of electromagnetic transients in single-and multiphase networks,” *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-88, no. 4, pp. 388–399, Apr. 1969.

S. A. Dorado-Rojas, M. N. Catalán, M. de Castro Fernandes, and L. Vanfretti, “Performance benchmark of Modelica time-domain power system automated simulations using Python,” in *Proceedings of the American Modelica Conference 2020*, Boulder, Colorado, USA, Mar. 2020.

P. Dubucq and G. Ackermann, “Optimal use of energy storage potentials in a renewable energy system with district heating,” *Energy Procedia*, vol. 135, pp. 158 – 171, 2017, 11th International Renewable Energy Storage Conference, IRES 2017, 14-16 March 2017, Düsseldorf, Germany.

DYNCAP:, “Dynamic Capture of CO2 Dynamic investigation of steam power processes with CO2 capturing for providing balancing energy,” http://www.kraftwerkforschung.info/en/mehr-flexibilitaet-fuer-emissionsarme-kohlekraftwerke, 2011-2014.

DYNSTART, “Start-up behavir of Power Plants - Subtopic CO2,” https://www.tuhh.de/alt/technische-thermodynamik/research/recent-projects/dynstart.html, 2015-2019.

P. Eberhart, T. S. Chung, A. Haumer, and C. Kral, “Open source library for the simulation of wind power plants,” in *Modelica’2015: The 11th International Modelica Conference*, Paris, France, Sep. 2015.

E. Eich-Soellner and C. Führer, *Numerical Methods in Multibody Dynamics*. Springer Fachmedien Wiesbaden, 1998.

H. Elmqvist, “A structured model language for large continuous systems,” Ph.D. dissertation, Lund Institute of Technology, Lund, Sweden, 1978.

H. Elmqvist and M. Otter, “Methods for tearing systems of equations in object oriented

modeling," in *ESM'94: European Simulation Multiconference*, Barcelona, Spain, Jun. 1994.

H. Elmqvist, S. E. Mattsson, and M. Otter, "Object-oriented and hybrid modeling in Modelica," *Journal Europen des systmes automatiss*, vol. 35, no. 1, pp. 1–X, 2001.

H. Elmqvist, H. Tummescheit, and M. Otter, "Object-oriented modeling of thermo-fluid systems," in *Modelica'2003: The 3rd International Modelica Conference*, Linköping, Sweden, 2003.

H. Elmqvist, "Modelica evolution - from my perspective," in *Modelica'2014: The 10th International Modelica Conference*, Lund, Sweden, Apr. 2014.

H. Elmqvist and S. E. Mattsson, "Modelica - The next generation modeling language: an international design effort," in *ESS97: The 9th European Simulation Symposium*, Passau, Germany, Oct. 1997.

H. Elmqvist, S. E. Mattsson, and H. Olsson, "Parallel model execution on many cores," in *Modelica'2014: The 10th International Modelica Conference*, Lund, Sweden, Apr. 2014.

H. Elmqvist, H. Olsson, A. Goteman, V. Roxling, D. Zimmer, and A. Pollok, "Automatic gpu code generation of modelica functions," in *Modelica'2015: The 11th International Modelica Conference*, Paris, France, Sep. 2015.

H. Elmqvist, T. Henningsson, and M. Otter, "Innovations for future Modelica," in *Modelica'2017: The 12th International Modelica Conference*, Prague, Czech Republic, May 2017.

M. u. M. S. E. Elmqvist, Hilding und Otter, "Fundamentals of synchronous control in modelica," in *Modelica'2012: The 9th International Modelica Conference*, Munich, Germany, Sep. 2012.

A. Elsheikh, E. Widl, and P. Palensky, "Simulating complex energy systems with modelica: A primary evaluation," in *2012 6th IEEE International Conference on Digital Ecosystems Technologies (DEST)*. IEEE, 2012, pp. 1–6.

A. Elsheikh, "An equation-based algorithmic differentiation technique for differential algebraic equations," *Journal of Computational and applied Mathematics*, vol. 281, pp. 135 – 151, June 2015.

——, "Derivative-based hybrid heuristics for continuous-time simulation-optimization," *Simulation Modelling Practice and Theory*, vol. 46, pp. pp. 164 – 175, Aug. 2014.

——, "Modeling parameter sensitivities using equation-based algorithmic differentiation techniques: The ADMSL.Electrical.Analog library," in *Modelica'2014: The 10th International Modelica Conference*, Lund, Sweden, Mar. 2014.

——, "Dynamic Parameter Sensitivities: Summary of computation methods for continuous-time Modelica models," in *Modelica'2019: The 13th International Modelica Conference*,

Regensburg, Germany, Mar. 2019.

EMPHYSIS, "Embedded systems with physical models in the production code software," https://emphysis.github.io, 2017-2020.

O. Enge, C. Clauß, P. Schneider, P. Schwarz, M. Vetter, and S. Schwunk, "Quasi-stationary AC analysis using phasor description with Modelica," in *Modelica'2006: The 5th International Modelica Conference*, Vienna, Austria, Sep. 2006.

EUROSYSLIB, "European Leaderhip in System Modeling and Simulation through advanced Modelica Libraries," https://itea3.org/project/eurosyslib.html, 2007-2010.

J. Ferreira and J. Estima de Oliveira, "Modelling hybrid systems using statecharts and Modelica," in *The 7th IEEE International Conference on Emerging Technologies and Factory Automation, 1999. Proceedings. ETFA '99.*, vol. 2, 1999, pp. 1063 –1069 vol.2.

J. M.-D. Filippo, M. Delgado, C. Brie, and H. M. Paynter, "A survey of bond graphs : Theory, applications and programs," *Journal of the Franklin Institute*, vol. 328, no. 5, pp. 565 – 606, 1991.

X. Floros, F. E. Cellier, and E. Kofman, "Discretizing time or states?: A comparative study between DASSL and QSS (work in progress paper)," in *EOOLT'2010: The 3rd International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools*, Oslo, Norway, Oct. 2010.

X. Floros, F. Bergero, N. Ceriani, F. Casella, E. Kofman, and F. E. Cellier, "Simulation of smart-grid models using quantization-based integration methods," in *Modelica'2014: The 10th International Modelica Conference*, Lund, Sweden, Apr. 2014.

J. W. Forrester, *Industrial Dynamics*. M.I.T. Press, 1961.

——, *Urban Dynamics*. M.I.T. Press, 1969.

——, *Principles of Systems*. M.I.T. Press, 1971.

——, *World Dynamics*. M.I.T. Press, 1971.

R. Franke and H. Wiesmann, "Flexible modeling of electrical power systems – the Modelica PowerSystems library," in *Modelica'2014: The 10th International Modelica Conference*, Lund, Sweden, Apr. 2014.

R. Franke, F. Casella, M. Otter, M. Sielemann, H. Elmqvist, S. E. Mattsson, and H. Olsson, "Stream connectors – an extension of Modelica for device-oriented modeling of convective transport phenomena," in *Modelica'2009: The 7th International Modelica Conference*, Como, Italy, 2009.

R. Franke, F. Casella, M. Sielemann, K. Proelss, M. Otter, and M. Wetter, "Standardization of themo-fluid modeling in Modelica.Fluid," in *Modelica'2009: The 7th International Modelica Conference*, Como, Italy, 2009.

R. Franke, M. Walther, N. Worschech, W. Braun, and B. Bachmann, "Model-based control with fmi and a c++ runtime for modelica," in *Modelica'2015: The 11th International Modelica Conference*, Paris, France, Sep. 2015.

L. Frayssinet, F. Kuznik, J.-L. Hubert, M. Milliez, and J.-J. Roux, "Adaptation of building envelope models for energy simulation at district scale," *Energy Procedia*, vol. 122, pp. 307 – 312, 2017, cISBAT 2017 International ConferenceFuture Buildings Districts Energy Efficiency from Nano to Urban Scale.

P. Fritzson, A. Pop, M. Sjölund, and A. Ashgar, "MetaModelica – a symbolic-numeric Modelica language and comparison to Julia," in *Modelica'2019: The 13th International Modelica Conference*, Regensburg, Germany, Mar. 2019.

P. Fritzson, A. Pop, K. Abdelhak, A. Asghar, B. Bachmann, W. Braun, D. Bouskela, R. Braun, L. Buffoni, F. Casella, R. Castro, R. Franke, D. Fritzson, M. Gebremedhin, A. Heuermann, B. Lie, A. Mengist, L. Mikelsons, K. Moudgalya, L. Ochel, A. Palanisamy, V. Ruge, W. Schamai, M. Sjölund, B. A. Thiele, J. Tinnerholm, and P. Östlund, "The OpenModelica integrated environment for modeling, simulation, and model-based development," *Modeling, Identification and Control*, vol. 41, no. 4, pp. 241–295, 2020.

M. Gebremedhin, A. H. Moghadam, P. Fritzson, and K. Stavåker, "A data-parallel algorithmic Modelica extension for efficient execution on multi-core platforms," in *Modelica'2012: The 9th International Modelica Conference*, Munich, Germany, Sep. 2012.

M. Gevorkyan, M. Hnatich, I. M. Gostev, A. V. Demidova, A. V. Korolkova, D. S. Kulyabov, and L. A. Sevastianov, "The stochastic processes generation in OpenModelica," in *Distributed Computer and Communication Networks*, V. M. Vishnevskiy, K. E. Samouylov, and D. V. Kozyrev, Eds. Cham: Springer International Publishing, 2016, pp. 538–552.

C. Gomes, C. Thule, D. Broman, P. Larsen, and H. Vangheluwe, "Co-simulation: A survey," *ACM Computing Surveys*, vol. 51, 05 2018.

F. J. Gómez, M. A. Aguilera, S. H. Olsen, and L. Vanfretti, "Software requirements for interoperable and standard-based power system modeling tools," *Simulation Modelling Practice and Theory*, vol. 103, p. 102095, 2020.

F. Gottelt, T. Hoppe, and L. Nielsen, "Applying the power plant library ClaRa for control

optimisation," in *Modelica'2017: The 12th International Modelica Conference*, Prague, Czech Republic, May 2017.

gPROMS, "Process systems enterprise," http://www.psenterprise.com/gproms/index.html.

M. Gräber, C. Kirches, D. Scharff, and W. Tegethoff, "Using functional mocku-up units for nonlinear model predictive control," in *Modelica'2012: The 9th International Modelica Conference*, Munich, Germany, Sep. 2012.

A. Hafez and A. Forsyth, "A review of more-electric aircraft," *13th International Conference on Aerospace Sciences and Aviation Technology*, vol. 13, 04 2009.

A. Halder, N. Pal, and D. Mondal, "Transient stability analysis of a multimachine power system with tcsc controller a zero dynamic design approach," *International Journal of Electrical Power & Energy Systems*, vol. 97, pp. 51 – 71, 2018.

A. Haumer, C. Kral, J. Gragger V., and H. Kapeller, "Quasi-Stationary modeling and simulation of electrical circuits using complex phasors," in *Modelica'2008: the 6th International Modelica Conference*, Bielefeld, Germany, March 2008.

A. Haumer, C. Kral, H. Kapeller, B. T., and J. Gragger V., "The AdvancedMachines library: Loss models for electric," in *Modelica'2009: The 7th International Modelica Conference*, Como, Italy, 2009.

J.-P. Heckel and C. Becker, "Advanced modeling of electric components in integrated energysystems with the TransiEnt library," in *Modelica'2019: The 13th International Modelica Conference*, Regensburg, Germany, Mar. 2019.

E. Henningsson, H. Olsson, and L. Vanfretti, "DAE solvers for large-scale hybrid models," in *Modelica'2019: The 13th International Modelica Conference*, Regensburg, Germany, 2019.

I. A. Hiskens and M. A. Pai, "Trajectory sensitivity analysis of hybrid systems," *IEEE Transactions on Circuits and Systems – Part I: Fundamental Theory and Applications*, vol. 47, no. 2, Feb. 2000.

C. Höger, "Sparse causalisation of differential algebraic equations for efficient event detection," in *The 8th EUROSIM Congress on Modelling and Simulation*. Cardiff, UK: IEEE Computer Society, Sep. 2013, pp. 351–356.

S. Hölemann and D. Abel, "Modelica predective control - An MPC library for Modelica," *Automatisierungstechnik*, vol. 57, no. 4, pp. 187–194, Sep. 2009.

Q. Huang, "Electromagnetic transient and electromechanical transient stability hybrid simulation: Design, development and its applications," Ph.D. dissertation, 11 2016.

IBPSA, “IBPSA Project 1: BIM/GIS and Modelica Framework for building and community energy system design and operation,” https://ibpsa.github.io/project1/index.html, 2018-2022.

INDIGO, “New Generation of Intelligent Efficient District Cooling Systems,” https://www.indigo-project.eu/, 2016-2020.

ITESLA, “Innovative Tools for Electrical System Security within Large Areas,” https://cordis.europa.eu/project/id/283012, 2012-2016.

A. Joos, K. Dietl, and G. Schmitz, “Thermal separation: An approach for a Modelica library for absorption, adsorption and rectification,” in *Modelica’2009: The 7th International Modelica Conference*, Como, Italy, 2009.

F. Jorissen, G. Reynders, R. Baetens, D. Picard, D. Saelens, and L. Helsen, “Implementation and verification of the IDEAS building energy simulation library,” *Journal of Building Performance Simulation*, vol. 11, no. 6, pp. 669–688, 2018.

——, “Implementation and verification of the ideas building energy simulation library,” *Journal of Building Performance Simulation*, vol. 11, no. 6, pp. 669–688, 2018.

S. N. Kalaschnikow, “PQLib - a Modelica library for power quality analysis in networks,” in *Modelica’2002: The 2nd International Modelica Conference*, Munich, Germany, 2002.

Y. L. Karnavas and E. I. Lygouras, “Synchronous machine analysis and modelling in LabVIEW: An educational tool for transient stability studies,” *The International Journal of Electrical Engineering & Education*, vol. 57, no. 3, pp. 202–229, 2018.

A. Klöckner, A. Knoblach, and A. Heckmann, “How to shape noise spectra for continuous system simulation,” *Mathematical and Computer Modelling of Dynamical Systems*, vol. 23, no. 3, pp. 284–300, 2017.

A. Klökner, F. L. J. van der Linden, and D. Zimmer, “Noise generation for continuous system simulation,” in *Modelica’2014: The 10th International Modelica Conference*, Lund, Sweden, Apr. 2014.

E. Kofman, “A third order discrete event simulation method for continuous system simulation,” *Latin American Applied Research*, vol. 36, no. 2, pp. 101–108, 2006.

——, “A second-order approximation for DEVS simulation of continuous systems,” *Simulation*, vol. 78, no. 2, pp. 76–89, 2002.

——, “Discrete event simulation of hybrid systems,” *SIAM Journal of Scientific Computing*, vol. 25, no. 5, p. 17711797, May 2004.

E. Kofman and S. Junco, “Quantized-state systems: A DEVS approach for continuous system

simulation," *Simulation: Transactions of the Society for Computer Simulation International*, vol. 18, no. 3, pp. 123–132, 2001.

J. Köhler, H.-M. Heinkel, P. Mai, J. Krasser, M. Deppe, and M. Nagasawa, "Modelica-Association-Project System Structure and Parameterization early insights," in *The First Japanese Modelica Conferences*, Tokyo, Japan, May 2016.

A. Kolmogorov, "On the representation of continuous functions of several variables by superpositions of continuous functions of a smaller number of variables," *Proceedings of the USSR Academy of Sciences*, vol. 108, p. 179182, 1956, (Russian).

C. Kral and A. Haumer, "Modelica libraries for dc machines, three phase and polyphase machines," in *Modelica'2005: The 4th International Modelica Conference*, Hamburg, Germany, 2005.

——, *Object Oriented Modeling of Rotating Electrical Machines*. InTech, 2011, ch. Advances in Computer Science and Engineering, pp. 135–160.

——, "The new FundamentalWave library for modeling rotating electrical three phase machines," in *Modelica'2011: The 8th International Modelica Conference*, Dresden, Germany, Mar. 2011.

M. Krammer, K. Schuch, C. Kater, K. Alekeish, T. Blochwitz, S. Materne, A. Soppa, and M. Benedikt, "Standardized integration of real-time andnon-real-time systems: The distributed co-simulationprotocol," in *Modelica'2019: The 13th International Modelica Conference*, Regensburg, Germany, Mar. 2019.

G. Kron, *Duakoptics - The Piecewise Solution of Large-scale Systems*. London, UK: MacDonald & Co., 1963.

P. Kundur, N. Balu, and M. Lauby, *Power System Stability and Control*, ser. Discussion Paper Series. McGraw-Hill Education, 1994.

A. D. la Calle, J. Hinkley, P. Scott, and J. D. Pye, "SolarTherm: a new Modelica library and simulation platform for concentrating solar thermal power systems," in *The 9th Eurosim Congress*, Oulu, Finland, Sep. 2018.

R. Lange, S. Traversaro, O. Lenord, and C. Bertsch, *Integrating the Functional Mock-Up Interface with ROS and Gazebo*. Cham: Springer International Publishing, 2021, pp. 187–231.

M. Larsson, "ObjectStab - a Modelica library for power system stability studies," in *Modelica'2000: The Modelica Workshop*, Lund, Sweden, September 2000.

B. Leitner, E. Widl, W. Gawlik, and R. Hofmann, "A method for technical assessment of power-to-heat use cases to couple local district heating and electrical distribution grids," *Energy*, vol.

182, pp. 729–738, Sep. 2019.

A. Leitold and K. M. Hangos, "Structural solvability analysis of dynamic process models," *Computers & Chemical Engineering*, vol. 25, pp. 1633–1646, 2001.

K. Link, H. Steuer, and A. Butterlin, "Deficiencies of Modelica and its simulation environments for large fluid systems," in *Modelica'2009: The 7th International Modelica Conference*, Como, Italy, 2009.

X. Lu, K. Hinkelman, Y. Fu, J. Wang, W. Zuo, Q. Zhang, and W. Saad, "An open source modeling framework for interdependent energy-transportation-communication infrastructure in smart and connected communities," *IEEE Access*, vol. 7, 4 2019.

K.-E. M. Otter and I. D. Årzén, "Stategraph-a modelica library for hierarchical state machines," in *Modelica'2005: The 4th International Modelica Conference*, Hamburg, Germany, 2005.

C. Maffezzoni, R. Girelli, and P. Lluka, "Generating efficient computational procedures from declarative models," *Simulation Practice and Theory*, 1996.

K. Majetta, S. Böhme, C. Clauß, and P. Schneider, "SPICE3 Modelica library," in *Modelica'2009: The 7th International Modelica Conference*, Como, Italy, 2009.

K. Majetta, C. Clauß, M. Franke, and P. Schneider, "Improvement of MSL electrical analog library," in *Modelica'2009: The 7th International Modelica Conference*, Como, Italy, 2009.

K. Majetta, S. Böhme, C. Clauß, and P. Schneider, "MSL electrical Spice3 - Status and further development," in *Modelica'2011: The 8th International Modelica Conference*, Dresden, Germany, Mar. 2011.

M. Mans, T. Blacha, P. Remmen, and D. Müller, "Automated model generation and simplification for district heating and cooling netwroks," in *Modelica'2019: The 13th International Modelica Conference*, Regensburg, Germany, Mar. 2019.

W. Marquardt, "Trends in computer-aided process modeling," *Computers & Chemical Engineering*, vol. 20, no. 6, pp. 591 – 609, 1996, fifth International Symposium on Process Systems Engineering.

T. Marx-Schubach and G. Schmitz, "Modeling and simulation of the start-up process of coal fired power plants with post-combustion co2 capture," *International Journal of Greenhouse Gas Control*, vol. 87, pp. 44 – 57, 2019.

S. E. Mattsson and G. Söderlind, "Index reduction in differential-algebraic equations using dummy derivatives," *SIAM Journal on Scientific Computing*, vol. 14, no. 3, pp. 677 – 692, 1993.

V. Mehrmann and L. Wunderlich, “Hybrid systems of differential-algebraic equations analysis and numerical solution,” *Journal of Process Control*, vol. 19, no. 8, pp. 1218 – 1228, 2009, special Section on Hybrid Systems: Modeling, Simulation and Optimization.

G. Migoni and E. Kofman, “Linearly implicitdiscrete event methods for stiff odes,” *Latin American Applied Research*, Dec. 2009.

G. Migoni, P. Rullo, F. Bergero, and E. Kofman, “Efficient simulation of hybrid renewable energy systems,” *International Journal of Hydrogen Energy*, vol. 41, no. 32, pp. 13 934 – 13 949, 2016.

M. Minz, L. Netze, and A. Monti, “A multi-level approach to power system Modelica models,” in *2016 IEEE 17th Workshop on Control and Modeling for Power Electronics (COMPEL)*. Trondheim, Norway: Norwegian University of Science and Technology, Jun. 2016.

MODELISAR, “From System Modeling to S/W running on the Vehicle,” https://itea3.org/project/modelisar.htmll, 2008-2011.

MODRIO, “Model Driven Physical Systems Operation,” https://itea3.org/project/modrio.html, 2012-2016.

P. J. Mosterman, M. Otter, and H. Elmqvist, “Modeling petri nets as local constraint equations for hybrid systems using modelica,” in *In: proceedings of the Summer Computer Simulation Conference -98*, 1998, pp. 314–319.

D. Müller, M. Lauster, A. Constantin, M. Fuchs, and P. Remmen, “AixLib - an open-source Modelica library within the IEA-EBC Annex 60 framework,” in *BAUSIM’2016 IBPSA Germany*, Dresden, Germany, Sep. 2016.

S. C. Müller and et al., “Interfacing power system and ICT simulators: Challenges, state-of-the-art, and case studies,” *IEEE Transactions on Smart Grid*, vol. 9, no. 1, pp. 14–24, Jan. 2016.

M. A. A. Murad, L. Vanfretti, M. Rokonuzzaman, and R. A. Tuhin, “Enhancing engineering studies in developing countries using OpenModelica,” in *2017 4th International Conference on Advances in Electrical Engineering (ICAEE)*, 2017.

K. Murota, *Systems Analysis by Graphs and Matroids*. Berlin: Springer, 1987.

T. S. Nouidui and M. Wetter, “Tool coupling for the design and operation of building energy and control systems based on the functional mock-up interface standard,” in *Modelica’2014: The 10th International Modelica Conference*, Lund, Sweden, Apr. 2014.

C. Nytsch-Geusen and et al., “BuildingSystems Eine modular hierarchische Modell-Bibliothek

zur energetischen Gebäude- und Anlagensimulation," in *BAUSIM'2016 IBPSA Germany*, Dresden, Germany, Sep. 2016.

L. Ochel and et al., "Omsimulator – integrated fmi and tlm-based co-simulation with composite model editing and ssp," in *Modelica'2019: The 13th International Modelica Conference*, Regensburg, Germany, 2019.

D. Oeding and B. R. Oswald, *Elektrische Kraftwerke und Netze*, 8th ed. Springer, 2016.

H. Olsson, M. Otter, H. Elmqvist, and D. Brück, "Operator overloading in modelica 3.1," in *Modelica'2009: The 7th International Modelica Conference*, Como, Italy, 2009.

OPENCPS, "Open Cyber-Physical System Model-Driven Certified Development," https://www.opencps.eu/, 2015-2018.

OPENPROD, "Open Model-Driven Whole-Product Development and Simulation Environment," https://itea3.org/project/openprod.html, 2009-2012.

M. Otter and F. E. Cellier, *Software for Modeling and Simulating Control Systems*. Boca Raton, FL, US: CRC Press, 1996, ch. The Control Handbook, pp. 415–428.

M. Otter, "The LinearSystems library for continuous and discrete control systems," in *Modelica'2006: The 5th International Modelica Conference*, Vienna, Austria, Sep. 2006.

M. Otter and et al., "Formal requirement modeling for simulation-based verification," in *Modelica'2015: The 11th International Modelica Conference*, Paris, France, Sep. 2015.

P. Palensky, A. A. Van Der Meer, C. D. Lopez, A. Joseph, and K. Pan, "Cosimulation of intelligent power systems: Fundamentals, software architecture, numerics, and coupling," *IEEE Industrial Electronics Magazine*, vol. 11, no. 1, pp. 34–50, 2017.

P. Palensky, E. Widl, and A. Elsheikh, "Simulating cyber-physical energy systems: challenges, tools and methods," *IEEE Transactions on Systems, Man and Cybernetics – Part C: Applications and Reviews, S. I. on Industrial Applications of Distributed Intelligent Systems*, 2012.

P. Palensky, E. Widl, A. Elsheikh, and M. Stifter, "Modeling inteligent energy systems: Co-simulation platform for validating flexible-demand ev charging management," *IEEE Transactions on Smart Grid, SI: Real-Time Demand Response*, 2013.

C. C. Pantelides, "The consistent initialization of differential-algebraic systems," *SIAM Journal on Scientific and Statistical Computing*, vol. 9, no. 2, pp. 213–231, Mar. 1988.

H. M. Paynter, *Analysis and Design of Engineering Systems*. Cambridge, Massachusetts: MIT Press, 1961.

PEGASE, "Pan European Grid Advanced Simulation and state Estimation," https://ieeexplore.

ieee.org/document/6465783, 2008-2012.

C. Perfumo, E. Kofman, J. H. Braslavsky, and J. K. Ward, "Load management: Model-based control of aggregate power for populations of thermostatically controlled loads," *Energy Conversion and Management*, vol. 55, pp. 36 – 48, 2012.

A. Pfeiffer, M. Hellerer, S. Hartweg, M. Otter, and M. Reiner, "PySimulator – A simulation and analysis environment in Python with plugin infrastructure," in *Modelica'2012: The 9th International Modelica Conference*, Munich, Germany, Sep. 2012.

A. Pollok, A. Klöckner, and D. Zimmer, "Psychological aspects of equation-based modelling," *Mathematical and Computer Modelling of Dynamical Systems*, vol. 25, no. 2, 25 March 2019.

A. Pop, M. Sjölund, A. Ashgar, P. Fritzson, and F. Casella, "Static and dynamic debugging of Modelica models," in *Modelica'2012: The 9th International Modelica Conference*, Munich, Germany, Sep. 2012.

J. A. X. Prabhu, S. Sharma, M. Nataraj, and D. P. Tripathi, "Design of electrical system based on load flow analysis using etap for iec projects," in *2016 IEEE 6th International Conference on Power Systems (ICPS)*, March 2016, pp. 1–6.

V. S. Prat, A. Urquia, and S. Dormido, "ARENALib: A modelica library for discrete-event system simulation," in *Modelica'2006: The 5th International Modelica Conference*, Vienna, Austria, Sep. 2006.

C. Protopapadaki and D. Saelens, "Heat pump and pv impact on residential low-voltage distribution grids as a function of building and district properties," *Applied Energy*, vol. 192, pp. 268 – 281, 2017.

——, "Towards metamodeling the neighborhood-level grid impact of low-carbon technologies," *Energy and Buildings*, vol. 194, pp. 273 – 288, 2019.

J. R. Ragazzini, R. H. Randall, and F. A. Russell, "Analysis of problems in dynamics by electronic circuits," *Proceedings of the IRE*, vol. 35, no. 5, pp. 444–452, 1947.

B. Rahrovi and M. Ehsani, "A review of the more electric aircraft power electronics," 02 2019, pp. 1–6.

A. Rande and F. Casella, "Multi-rate integration algorithms: a path towards efficient simulation of object-oriented models of very large systems," in *EOOLT'2014: The 6h International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools*, Berin, Germany, Apr. 2014.

ResiliEntEE, "Resilience of integrated energy networks with a high share of renewable Energies,"

https://www.tuhh.de/transient-ee/en/projectdescription.html, 2017-2021.

J. R. Rice, “Split Runga-Kutta method for simultaneous equations,” *Journal of Research of the National Bureau of Standards – B. Mathematics and Mathematical Physics*, vol. 64B, no. 3, Jul. 1960.

M. Richter, G. Oeljeklaus, and K. Grner, “Improving the load flexibility of coal-fired power plants by the integration of a thermal energy storage,” *Applied Energy*, vol. 236, pp. 607 – 621, 2019.

S. Roberts, A. Sarshar, and A. Sandu, “Coupled multirate infinitesimal GARK schemes for stiff systems with multiple time scales,” *SIAM Journal on Scientific Computing*, vol. 42, p. A1609A1638, Nov. 2018.

B. Sabeeh and C. Gan, “Power system frequency stability and control: Survey,” vol. 11, pp. 5688–5695, 05 2016.

L. Saldamli, P. Fritzson, and B. Bachmann, “Extending Modelica for partial differential equations,” in *Modelica’2002: The 2nd International Modelica Conference*, Munich, Germany, 2002.

V. Sanz and A. Urquia, “Modelica extensions for supporting message passing communication and dynamic data structures,” in *EOOLT’2016: The 7h International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools*, Milan, Italy, Apr. 2016.

V. Sanz, A. Urquia, F. E. Cellier, and S. Dormido, “Modeling of hybrid control systems using the DEVSLIB Modelica library,” *Control Engineering Practice*, vol. 20, no. 1, pp. 24–34, Jan. 2012.

——, “Hybrid system modeling using the SIMANLib and ARENALib modelica libraries,” *Simulation Modelling Practice and Theory*, vol. 37, pp. 1–17, Sep. 2013.

V. Sanz, A. Urquia, and F. Casella, “Improving efficiency of hybrid system simulation in Modelica,” in *EOOLT’2014: The 6h International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools*, Berin, Germany, Apr. 2014.

V. Sanz, A. Urquia, and A. Leva, “1d/2d cellular automata modeling with modelica,” in *Modelica’2014: The 10th International Modelica Conference*, Lund, Sweden, Apr. 2014.

——, “CellularAutomataLib2 : improving the support for cellular automata modelling in Modelica,” *Mathematical and Computer Modelling of Dynamical Systems*, vol. 22, no. 3, pp. 244–264, May 2016.

V. Sanz, F. Bergero, and A. Urquia, “An approach to agent-based modeling with Modelica,”

Simulation Modelling Practive and Theory, vol. 83, pp. 65–74, Apr. 2018.

V. Savcenco, "Construction of a multirate RODAS method for stiff ODEs," *Journal of Computational and Applied Mathematics*, vol. 225, no. 2, pp. 323 – 337, 2009.

F. Schiavo and F. Casella, "Object-oriented modelling and simulation of heat exchangers with finite element methods," *Mathematical and Computer Modelling of Dynamical Systems*, vol. 13, no. 3, pp. 211–235, May 2007.

G. F. Schneider, J. Oppermann, A. Constantin, R. Streblow, and D. Müller, "Hardware-in-the-Loop-Simulation of a building energy and control system to investigate circulating pump control using modelica," in *Modelica'2015: The 11th International Modelica Conference*, Paris, France, Sep. 2015.

G. Schweiger, G. Engel, S. J., I. Hafner, T. S. Nouidui, and C. Gomes, "Co-simulation - an empirical survey: Applications, recent developments and future challenges," *Simulation Notes Europe*, vol. 30, no. 2, pp. 73–76, Jun. 2020.

G. Schweiger, H. Nilsson, J. Schoeggl, W. Birk, and A. Posch, "Modeling and simulation of large-scale systems: A systematic comparison of modeling paradigms," *Applied Mathematics and Computation*, vol. 365, 2020.

M. Sielemann and G. Schmitz, "A quantitative metric for robustness of nonlinear algebraic equation solvers," *Mathematics and Computers in Simulation*, vol. 81, no. 12, pp. 2673 – 2687, 2011.

M. Sielemann, F. Casella, and M. Otter, "Robustness of declarative modeling languages: Improvements via probability-one homotopy," *Simulation Modelling Practice and Theory*, vol. 38, pp. 38–57, Nov. 2013.

M. Sielemann, "probability-one homotopy for robust initialization of differential-algebraic equations," in *Modelica'2012: The 9th International Modelica Conference*, Munich, Germany, Sep. 2012.

C. P. Steinmetz, "Complex quantities and their use in Electrical Engineering," in *Proceedings of the International Electrical Congress, Conferece of AIEE (American Institute of Electrical Engineers Proceedings)*, Chicago, USA, 1893, pp. 33–74.

G. Sulligoi, A. Vicenzutti, and R. Menis, "All-electric ship design: From electrical propulsion to integrated electrical and electronic power systems," *IEEE Transactions on Transportation Electrification*, vol. 2, no. 4, pp. 507–521, Dec 2016.

B. Thiele, M. Otter, and S. E. Mattsson, "Modular multi-rate and multi-method real-time

simulation," in *Modelica'2014: The 10th International Modelica Conference*, Lund, Sweden, Apr. 2014.

B. Thiele, T. Beutlich, V. Waurich, M. Sjölund, and T. Bellmann, "Towards a standard-conform platform-generic and feature-rich Modelica device drivers library," in *Modelica'2017: The 12th International Modelica Conference*, Prague, Czech Republic, May 2017.

J. S. Thongam, M. Tarbouchi, A. F. Okou, D. Bouchard, and R. Beguenane, "All-electric ships – a review of the present state of the art," in *2013 Eighth International Conference and Exhibition on Ecological Vehicles and Renewable Energies (EVER)*, March 2013, pp. 1–8.

M. Tiller and D. Winkler, "impact – A Modelica Package Manager," in *Modelica'2014: The 10th International Modelica Conference*, Lund, Sweden, Apr. 2014.

TransiEnt, "Transient Behavior of Integrated Energy Networks with a High Share of Renewable Energies," https://www.tuhh.de/transient-ee/en/project.html, 2013-2017.

A. Ulbig, T. S. Borsche, and G. Andersson, "Impact of low rotational inertia on power system stability and operation," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 7290 – 7297, 2014, 19th IFAC World Congress.

V. Valdivia-Guerrero, R. Foley, S. Riverso, P. Govindaraju, A. Elsheikh, L. Mangeruca, G. Burgio, A. Ferrari, M. Gottschall, T. Blochwitz, S. Bloch, D. Taylor, D. Hayes-McCoy, and A. Himmler, "Modelling and simulation tools for systems integration on aircraft," in *SAE Technical Paper*. SAE International, 09 2016.

F. L. J. van der Linden, "General fault triggering architecture to trigger model faults in Modelica using a standardized blockset," in *Modelica'2014: The 10th International Modelica Conference*, Lund, Sweden, Apr. 2014.

L. Vanfretti, T. Bogodorova, and M. Baudette, "A Modelica power system component library for model validation and parameter identification," in *Modelica'2014: The 10th International Modelica Conference*, Lund, Sweden, Apr. 2014.

L. Vanfretti, T. Rabuzin, M. Baudette, and M. M., "itesla power systems library (iPSL): A Modelica library for phasor time-domain simulations," *SoftwareX*, vol. 5, pp. 84–88, 2016.

L. Vanfretti, M. Baudette, A. Amazouz, T. Bogodorova, T. Rabuzin, J. Lavenius, and F. J. Gomz-Lpez, "Rapid: A modular and extensible toolbox for parameter estimation of modelica and fmi compliant models," *SoftwareX*, vol. 5, pp. 144 – 149, 2016.

F. Villella, S. Leclerc, I. Erlich, and S. Rapoport, "Pegase pan-european test-beds for testing of algorithms on very large scale power systems," in *2012 3rd IEEE PES Innovative Smart Grid*

Technologies Europe (ISGT Europe), Berlin, Germany, Oct. 2012.

J. Šilar, F. Ježek, and J. Kofránek, "PDEModelica: a modelica language extension for partial differential equations implemented in openmodelica," *International Journal of Modelling and Simulation*, vol. 38, no. 2, pp. 128–137, 2018.

J. Webster and C. Bode, "Implementation of a non-discretized multiphysics PEM electrolyzer model in Modelica," in *Modelica'2019: The 13th International Modelica Conference*, Regensburg, Germany, Mar. 2019.

M. Wetter and C. Haugstetter, "Modelica versus trnsys – a comparison between an equation-based and a procedural modeling language for building energy simulation," in *The 2nd SimBuild Conference*, Cambridge, MA, USA, August 2006.

M. Wetter, C. Van Treeck, L. Helsen, A. Maccarini, D. Saelens, D. Robinson, and G. Schweiger, "Ibpsa project 1: Bim/gis and modelica framework for building and community energy system design and operation - ongoing developments, lessons learned and challenges," *IOP Conference Series / Earth and Environmental Science*, vol. 323, no. 1, 9 2019.

M. Wetter and et al., "IEA EBC Annex 60 Modelica Library – An international collaboration to develop a free open-source model library for buildings and community energy systems," in *14th IBPSA Conference*, Hyderabad, India, Dec. 2015.

M. Wetter, W. Zuo, T. S. Nouidui, and X. Pang, "Modelica Buildings library," *Journal of Building Performance Simulation*, vol. 7, no. 4, pp. 253–270, 2014.

E. Widl, W. Müller, A. Elsheikh, M. Hörtenhuber, and P. Palensky, "The FMI++ library: A high-level utility package for FMI for model exchange," in *The IEEE Workshop on Modeling and Simulation of Cyber-Physical Energy Systems*, Berkeley, USA, May 2013.

D. Winkler, "Electrical power system modelling in Modelica - comparing open-source library options," in *SIMS2017: The 58th Conference on Simulation and Modelling*, Reykjavik, Iceland, Sep. 2017.

L. Zabala, J. Febres, R. Sterling, S. Lpez, and M. Keane, "Virtual testbed for model predictive control development in district cooling systems," *Renewable and Sustainable Energy Reviews*, vol. 129, p. 109920, 2020.

B. P. Zeigler and J. S. Lee, "Theory of quantized systems: formal basis for DEVS/HLA distributed simulation environment," in *Enabling Technology for Simulation Science II*, A. F. Sisti, Ed., vol. 3369, International Society for Optics and Photonics. SPIE, 1998, pp. 49 – 58.

B. P. Ziegler, *Theory of Modeling and Simulation*. New York, USA: John Wiley & Sons, 1976.

D. Zimmer, “A new framework for the simulation of equation-based models with variable structure,” *SIMULATION*, vol. 89, no. 8, pp. 935–963, 2013.

——, “Equation-based modeling with Modelica – principles and future challenges,” *Simulation Notes Europe*, vol. 26, no. 2, pp. 67–74, Jun. 2016.