SoftwareX 24 (2023) 101591

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

OpenIMDML: Open Instance Multi-Domain Motor Library utilizing the Modelica modeling language

Fernando Fachini [a,*], Marcelo de Castro [b], Tetiana Bogodorova [a], Luigi Vanfretti [a]

[a] Rensselaer Polytechnic Institute, Department of Electrical, Computer & Systems Engineering, Troy, NY, USA
[b] Mitsubishi Electric Power Products, Inc. Power Systems Engineering Department, Warrendale, PA, USA

ARTICLE INFO

Dataset link: https://github.com/ALSETLab/OpenIMDML

Keywords:
Modelica
Multi-domain motor models
Power systems
Power grid
OpenIPSL
OpenIMDML

ABSTRACT

This paper describes OpenIMDML, an open source Modelica-based library implemented for the representation of electrically driven loads in a multi-domain engineering system. This novel approach consists of providing a physically meaningful equation-based mechanical interface of the motor's torque/speed coupled to the electrical model of the induction motor. Because the newly developed motor models have a phasor domain representation of the electrical domain and a mechanical flange for rotational component coupling, this allows the user to integrate different domain subsystems with power grid models in one single multi-domain model utilizing the Modelica language and the OpenIPSL library. The library has been developed entirely using the Modelica programming language and has been tested in both Dymola and OpenModelica software tools.

Code metadata

Current code version	v1.0
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-23-00599
Permanent link to Reproducible Capsule	
Legal Code License	GNU AGPL
Code versioning system used	git
Software code languages, tools, and services used	Modelica Language Python Dymola OpenModelica
Compilation requirements, operating environments & dependencies	OS: Windows or Linux
If available Link to developer documentation/manual	https://github.com/ALSETLab/OpenIMDML/blob/main/README.md
Support email for questions	emaildofachini@gmail.com

1. Motivation and significance

The electrical power system relies heavily upon simulation tools to study the electrical grid, in order to address both steady-state and dynamic performance issues. When creating models, several modeling assumptions and restrictions are baked into models and tools [1]. That is, models are by default a simplified version of something that is real [2]. Model variability depends of the modeling goal, and what is sought after. In the dynamic power system analysis realm, for instance, phasor domain tools represent the grid in positive sequence, with the assumption that it is possible to capture the most relevant dynamics impacting the three-phase power system's behavior.

As computer processing power increases, the need for having less computationally demanding models tends to decrease. Hence, one of the modeling simplifications that is tackled in the OpenIMDML Library is that of the representation of loads coupled to induction motors in phasor-domain tools. When using a conventional power system tool for electromechanical stability analysis like Siemens PTI PSS®E [3], e.g., the mechanical load of a typical pump system (consisting of

* Corresponding author.
E-mail addresses: fachif@rpi.edu (Fernando Fachini), marcelo.decastro@meppi.com (Marcelo de Castro), bogodt2@rpi.edu (Tetiana Bogodorova), vanfrl@rpi.edu (Luigi Vanfretti).

https://doi.org/10.1016/j.softx.2023.101591
Received 7 September 2023; Received in revised form 1 November 2023; Accepted 15 November 2023

an electrical motor and a water pump), is represented through a polynomial function that mimics the mechanical steady-state behavior impacted by load features such as Load Torque (T_L) and Load Power (P_L).

Simplified load representations are valuable for conventional electrical system studies, but fall short in assessing their impact on the load sub-system. In contrast, the Modelica modeling language allows for comprehensive multi-domain models, expanding analysis capabilities while adhering to industry modeling practices [4]. Additional Modelica libraries, such as the Modelica Standard Library [5], the Electrified Power Trains Library [6], the Electric Power Library [7], and the Electrification Library [8], include induction motor and electrical drive models. However, their purposes differ from the primary objective of the OpenIMDML library, which is to study multi-domain interactions with bulk power systems. Another notable difference is that the mentioned libraries do not depict the electrical system in the phasor-domain, a modeling simplification that allows for rapid simulation of large-scale power systems.

In contrast, the OpenIMDML Library is an open-source Modelica library that provides a wide range of induction motor models, including both single-phase and three-phase, as well as multi-domain and non-multi-domain modeling representations. It also includes a variable speed drive model, which is often missing in traditional power system tools. The library aims to enhance conventional simulation capabilities in power systems by enabling the study of inter-domain interactions. Its sister library, OpenIPSL, provides non-multi-domain induction motor models, created with reference to the Power System Analysis Toolbox (PSAT) [1,9,10]. OpenIMDML complements OpenIPSL by adding multi-domain motor and variable speed drive models, eliminating the limitations associated with co-simulation, such as numerical stability issues and orchestration challenges [11]. These libraries are designed to work with the Modelica modeling language standard specification and can be used in various compliant tools.

2. Software description

The OpenIMDML is an open-source Modelica-based library that is focused on modeling and simulation of induction motor models in the phasor domain. It was built using the Modelica language, leveraging the Modelica Standard Library [12], and functions in conjunction with OpenIPSL [13,14].

The Modelica Language is a programming language for modeling cyber–physical systems. It enables object-oriented graphical modeling, allowing the connection of components based on mathematical equations, making it easier to model these systems accurately and from the ground up [15] through a Graphical User Interface (GUI) or programmatically. It offers object-oriented structures that facilitate model reusability, especially when building models for complex systems with various components, such as power systems.

The OpenIPSL was designed to use the phasor modeling approach for power system analysis and to facilitate its use by power engineers, it was implemented to replicate the behavior of the models from the well-known industry-grade power system tool called Siemens PTI PSS®E [3]. Moreover, an end user who is well versed with Modelica does not need to be familiar with power system-specific tools. This type of user would be comfortable of utilizing OpenIMDML and OpenIPSL. In this way, users are empowered and can satisfy their own modeling needs by building multi-domain models on top of both libraries.

In addition, another benefit of the author's approach is that OpenIMDML models can be used in any tool compliant with the open-access Modelica modeling language standard specification. A list of Modelica-compliant tools can be found in https://modelica.org/tools.html. Finally, these Modelica tools support the open-access and open-source Functional Mock-up Interface (FMI) standard for model-exchange, hence, the models developed by the user can be used by more than 100 tools, as listed here: https://fmi-standard.org/.

2.1. Software architecture

The library is structured into five subpackages: Examples, NonMultiDomain, MultiDomain, Controls, and Functions. A high-level explanation of each of the five main packages and their content is given below:

- ***Examples:*** contains Modelica model examples of all the components developed in the OpenIMDML Library. The *Examples* package contains three subpackages, namely *MultiDomainExamples*, *NonMultiDomainExamples*, and *BaseClasses*. The *MultiDomainExamples* subpackage presents examples of the single-phase and three-phase multi-domain motor developed in the library, including motor validation examples, examples including a variable speed drive and its controls, and a simple example of a multi-domain motor interacting with a pump that fills up a reservoir. The *NonMultiDomainExamples* subpackage contains examples of single-phase and three-phase non multi-domain motor models from the library, including examples that incorporate the variable speed drive. The difference between multi-domain and non multi-domain is linked to the driven load representation. Lastly, the *BaseClasses* subpackage contains partial models that are re-utilized in several of the examples from the package.
- ***NonMultiDomain:*** contains single-phase and three-phase non multi-domain induction motor models, i.e., models that lacks the mechanical interface for torque/speed. The *NonMultiDomain* package holds two subpackages, namely *SinglePhase*, and *ThreePhase*. The *SinglePhase* subpackage presents two distinct motor models — single-phase (SPIM) and dual-phase (DPIM) induction motor models. The SPIM motor is intended for cases when the motor is initialized in steady-state, while the DPIM motor can be used to simulate both start-up and steady-state initialization conditions. Both the SPIM and DPIM motor models are derived from [16]. The *ThreePhase* subpackage accommodates five different motor models — type I, type III, type V, CIM5, and CIM6. The first three are derived from PSAT [1,9,10] model manuals while the latter two are derived from PSS®E [3] model manual.
- ***MultiDomain:*** contains single-phase and three-phase multi-domain induction motor models, i.e., models that present a mechanical interface for torque/speed. It follows the same subpackage structure that is presented in the *NonMultiDomain* package, with the difference being the multi-domain nature of the models. There is one additional motor model in the *MultiDomain* package – an all-in-one motor – that implements the `replaceable` Modelica construct [17]. The user can select different motor models by simply changing a pre-set parameter in a Modelica-tool's GUI, or through Modelica Python scripting [18].
- ***Controls:*** contains power electronics and controller logic models for the variable speed drive model. The *Controls* package has two subpackages, namely *PowerElectronics*, and *ControllerLogic*. The *PowerElectronics* subpackage contains the phasor-based voltage source converter model, while the *ControllerLogic* subpackage presents the ubiquitously used Volts/Hertz controller logic [19].
- ***Functions:*** contains one function that implements an iterative method for estimating the induction motor equivalent circuit parameters using only the motor nameplate data [20].

Note that the absence of an ***Interfaces*** subpackage in OpenIMDML is attributed to the coherence of its components with those present in the sister library, OpenIPSL. Consequently, there exists no requirement for duplicating the interfaces already in use.

2.2. Software functionalities

In this section, we will present three of the major advantages and functionalities of the OpenIMDML Library that differentiate this library from conventional power system tools.

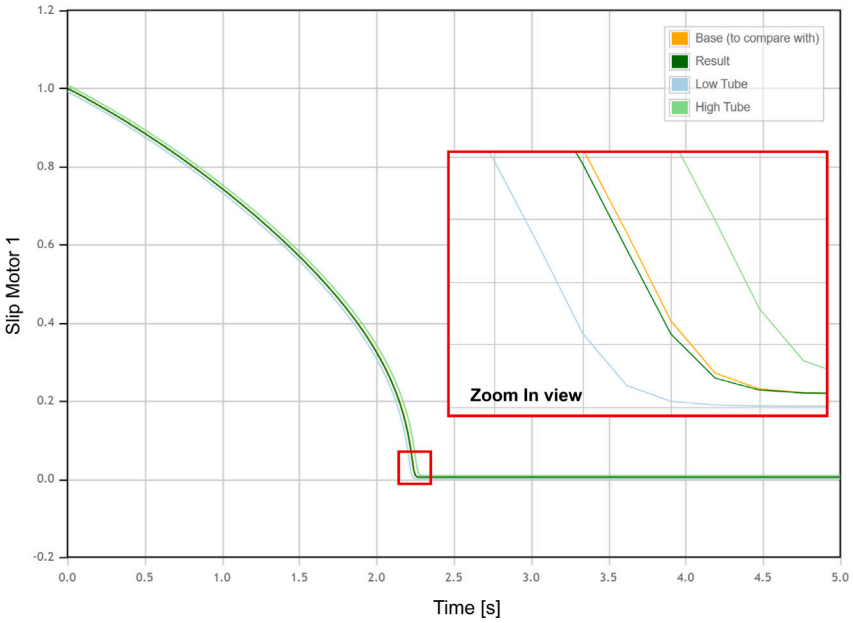


Fig. 1. *MultiDomainTypeI* example in `OpenIMDML`. Motor slip recording in Dymola (Base) and OpenModelica (Result).

Table 1
Simulation information.

Hardware and simulation environment information	
Hardware	OS: Microsoft Windows 10 Processor: Intel(R) Core(TM) i7-7820HQ Installed Physical Memory (RAM) : 16.0 GB
Simulation Setup for Dymola and OpenModelica	Start Time: 0, Stop Time: 5 s Number of Intervals: 500 Algorithm : Dassl Tolerance: 0.0001
Dymola 2023X Simulation Info	Simulation Run Time: 0.014 s
OpenModelica v1.21.0 Simulation Info	Simulation Run Time: 0.032 s

2.2.1. Interoperability: Running models in different simulation environments

There are currently ten different commercial and one free Modelica simulation environments [21]. The `OpenIMDML` has been carefully developed to comply with the Modelica Language Specification [17], and it has been implemented with Modelica version 4.0. This implementation helps to take advantage of interoperability among various Modelica tools. To demonstrate that the `OpenIMDML` examples can be simulated in different environments and that the simulation results are identical, the authors ran an `OpenIMDML` example in two distinct simulation environments: Dymola [22], and OpenModelica [23]. The example in case is the *MultiDomainTypeI*, where a multi-domain motor type I is operated in start-up mode. The simulation result in Fig. 1 displays the motor slip recording from the two different environments.

The orange curve is the simulation plot from Dymola (Base), while the dark green curve is the simulation plot from OpenModelica (Result). The blue and light green curves are the lower and upper values of the tube, respectively, mentioned earlier in the subsection, with a tolerance value chosen to be $1 \cdot 10^{-4}$. The reader can notice that both the Dymola and the OpenModelica result curves are nearly identical, pointing to the fact that `OpenIMDML` models are equivalent when running in both simulation environments. This means that the user is not constrained to any particular software tool, which is common in power engineering practice. Table 1 presents details regarding computer hardware, simulation setup for both Dymola and OpenModelica, as well as the corresponding simulation runtime.

2.2.2. Multi-domain representation of driven loads

In conventional power system tools, like Siemens PTI PSS®E, all mechanical loads driven by induction motors are simplified by polynomial functions, thus not accounting for their dynamics. The polynomial function emulates the load torque of the driven load, therefore the only set of dynamic equations of the motor model is related to the electromagnetic dynamics of the induction motor. For this reason, conventional power system tools are focused solely on the electrical domain. In contrast to the aforementioned tools, the Modelica modeling language enables the creation of multi-domain models and systems. Therefore, there is an opportunity of merging multiple different domains into one model, allowing for a holistic approach to modeling multi-domain system models. For example, the library includes an illustrative example under the *Examples.MultiDomainApplications* subpackage where the driven load includes a pump filling a water reservoir. A simple example is described in Section 3. Finally, this feature also allows to avoid the need for co-simulation and to circumvent its drawbacks [11], as mentioned earlier.

2.2.3. Variable speed drive modeling

Conventional three-phase induction motors connected directly to the grid, once in steady-state condition, operate at a single speed defined by the frequency of the grid's voltage. However, in many industrial applications, one must control the motor's speed and torque

in order to fulfill a specific task, or with the overall goal of improving productivity and energy savings in pumps, fans, compressors, etc. Thus, induction motors are controlled via Variable Speed Drives (VSDs), which are power electronic converters that enable magnitude and frequency control of the motor's terminal voltage. Conventional power system tools do not model VSDs, lacking a crucial component of today's motor applications. `OpenIMDML`, on the other hand, was implemented in such a way that all three-phase induction motor models, being multi-domain or not, can operate with a VSD model from the library. The library includes multiple examples under the *Examples* subpackage, in *MultiDomainThreePhaseMotorVSDStartup* and *MultiDomainControllableMotor*, where the use of the VSD model is illustrated.

3. Illustrative example

To illustrate a few of the features that the `OpenIMDML` Modelica Library has to offer, this section describes two simulation examples:

1. **Three-Phase Multi-Domain Validation Example:** evaluates the performance of the modeled three-phase multi-domain motors.
2. **VSD Controlled Voltage and Frequency Sweep Example:** demonstrates the use of Python in conjunction with Dymola in a parameter sweep script that displays multiple *Torque* **x** *Rotor Speed* simulation results for increasing synchronous speeds and voltage values.

For more information on how to run examples in OpenIMDML, please refer to the README section in https://github.com/ALSETLab/OpenIMDML.

3.1. Three-phase multi-domain validation example

This section presents the results of simulating the five motor model validation examples that are located in the *Examples* package under *MultiDomainThreePhaseMotorValidation*. Each example is meant to validate one of the five developed three-phase multi-domain motor models, and their structure under the *MultiDomainThreePhaseMotorValidation* packages follows the structure below:

```
MultiDomainThreePhaseMotorValidation
    > MultiDomainTypeI
    > MultiDomaiTypeIII
    > MultiDomainTypeV
    > MultiDomainCIM
    > MultiDomainMotor
```

The names of the examples reflect the motor model used in the simulation. For instance, an induction motor Type I model assumes that only the mechanical state variable is taken into account, with the circuit dynamics neglected. For motor Types III and V, the model represents single and double cage induction motors, respectively, no longer modeling only the mechanical states, but also including the dynamics related to flux-linkages in the windings of the motor. Motor Types I, III, and V are derived from [1]. Motor CIM corresponds to the implementation of a double cage induction motor, with the addition of saturation [3]. The first four examples in the package aim to simulate the same power system model when different motor models are used and are meant for motor dynamics validation purposes. For ease of motor testing, the example shown in this subsection is the *MultiDomainMotor*, which contains an "ALL-IN-ONE" induction motor component, allowing the user to change the motor model through one press of a button. The example not only serves the purpose of comparing motor model performance, but also highlights the object-oriented capabilities of Modelica. Because different motor models contain different degrees of modeling details (number of parameters, number of variables and equations), the authors chose a set of parameter values that can be simulated from the simplest to the most complex model, while at the same time representing the same induction motor circuit in different models. Fig. 2 displays the validation test. The following is a brief description of each component in the example:

- ***inf:*** functions as an infinite bus, maintaining constant voltage and frequency. It calculates active and reactive power to maintain power balance during dynamic simulations. Essentially, it serves as a rough equivalent of the bulk power system to which the remainder of the example's components connect. This infinite bus is represented by the classical synchronous machine model GENCLS, with detailed model equations available in [3].
- ***bus1 - bus4:*** are nodes where different power lines are connected.
- ***tf1 - tf2:*** are power transformers, electrical devices that alter an initial input voltage to produce a modified output voltage.
- ***line:*** are power lines that transport electrical energy between different locations within an electric power system.
- ***Load:*** component of the power system that consumes power.
- ***Synchronous_Speed:*** is specified by a `RealExpression` Modelica block, establishing the synchronous speed of the motor. In this specific case, a constant value of 60 Hz is assumed.
- ***Torque_Equation:*** is also specified by a `RealExpression` Modelica block, establishing the mechanical torque expression used to define the load torque expression.
- ***Torque:*** the component converts a real input signal into an angle and torque in the Modelica flange connector. Consequently, the component connected to this flange experiences a torque-induced motion.
- ***Torque_Sensor:*** computes the torque between two flanges and presents the outcome as an output signal. The output signal is used as the measurement of the mechanical load in the motor model.
- ***Load_Inertia:*** rotational element featuring inertia and two flanges rigidly interconnected. This component represents the inertia of the rotor shaft of the motor.
- ***MD ALL-IN-ONE:*** an `OpenIMDML` induction motor component that allows the user to change the motor model by selecting different models in a list of models.

Note that the grid components, to the left of and including *bus4*, are from `OpenIPSL`, while all other components except the motor are from the Modelica Standard Library (MSL) [5].

Once each model is simulated, an aggregated plot in Fig. 3 can be obtained. It compares the startup slip value through time for all of the modeled multi-domain, three-phase induction motors, which can be defined by using the ALL-IN-ONE motor component.

Because the model simulates an induction motor start-up process, $slip = 1$ at $t = 0$ (the motor starts at halt condition), and the slip decreases as the rotor speed increases, until finally it reaches its steady-state, where $slip = 0.004$. The slip behavior of the *Motor1* versus all other motors can be clearly distinguished in Fig. 3, particularly when the motor reaches its steady-state value. *Motor1* does not display any oscillatory behavior around $t = 2.35$ s because of its modeling limitation, as it does not model the effect of flux linkage variation in the rotor and stator coils. The *Motor1* model contains only one differential equation that describes the slip and torque relationship, lacking the rotor cage modeling present in *Motor3*, *Motor5*, and *CIM* motor models.

Fig. 4 showcases a comparison of the active and reactive power consumption, and again, it is possible to notice that *Motor1* has a different behavior when compared to the rest. In Fig. 4 (a), there are two noticeable oscillations in the active power consumption of *Motor3*, *Motor5*, and *CIM*: one at the start of the simulation and a second one around $t = 2.35$ s. The oscillation at $t = 0$ is a result of the modeling assumption that the voltage behind the stator resistance Rs starts from zero, i.e., $e'_{re}(0) = 0$ and $e'_{im}(0) = 0$. Further analysis through linearization of the non-linear equations at $t = 0$, showed a pair of

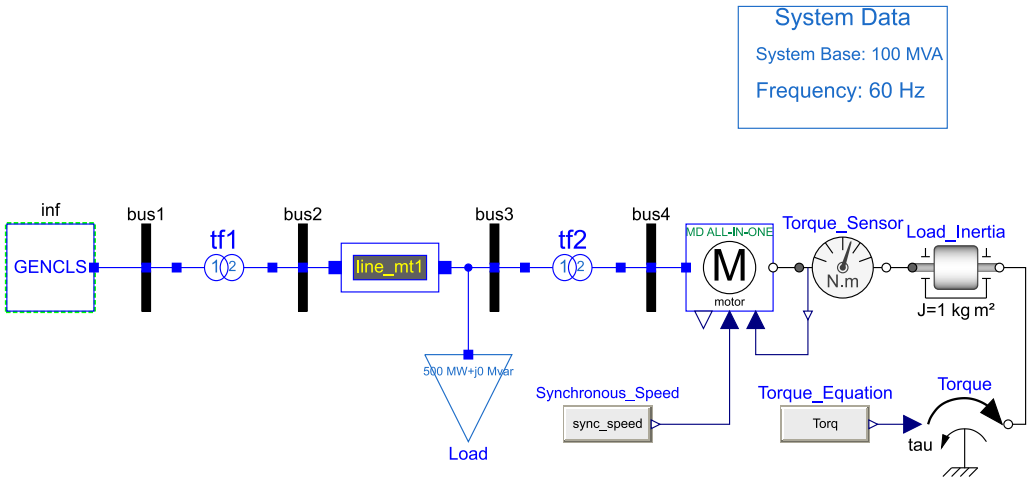


Fig. 2. Multi-Domain Induction Motor Validation Example.

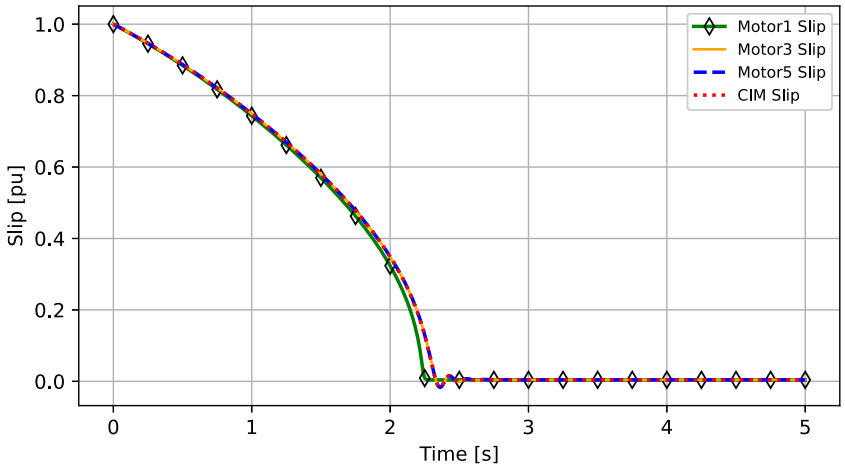


Fig. 3. Comparison of induction motor slip values in time.

complex eigenvalues that are stable (negative real component) with an oscillation frequency of 60.0684 Hz, validating the oscillation present in Fig. 4. The results depicted in the plots within this subsection align with the characteristics observed in Example 11.7 from [24], which was initially simulated using the PST software tool [25]. However, an exception is noted in the simulation results when employing *Motor1* due to modeling simplifications inherent to this component.

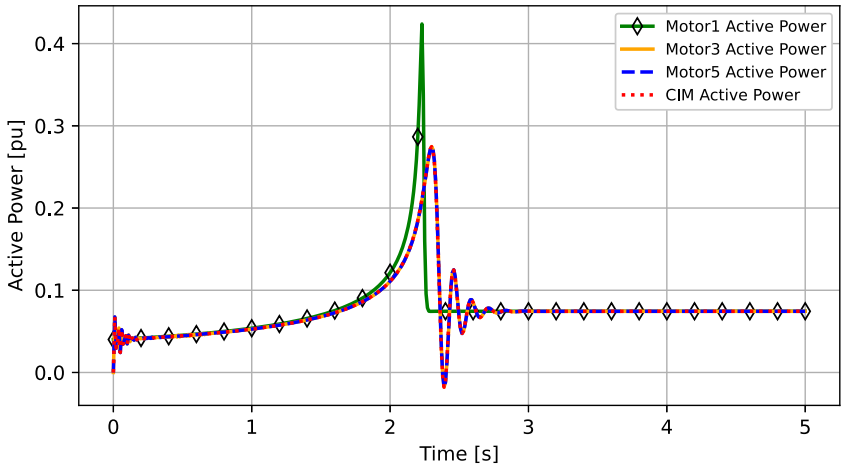
3.2. VSD controlled voltage and frequency sweep example

This section discusses two simulations, identified as Test 1 and Test 2, to facilitate clear tracking of the discussion. Both simulations focus on demonstrating the relationship between the voltage and synchronous speed of an induction motor. As described in [26], the magnetic field strength is proportional to the ratio between the voltage on the induction motor and its frequency. The volts/hertz control of a VSD increases voltage and frequency at a constant rate, maintaining the magnetic field close to a constant value, which guarantees a constant torque region. If voltage is held constant and frequency is lowered, then both magnetic field and torque increase. The opposite is also true, if voltage is held constant and frequency is increased, both magnetic field and torque decrease. Fig. 5 depicts the system utilized to illustrate the aforementioned behavior.

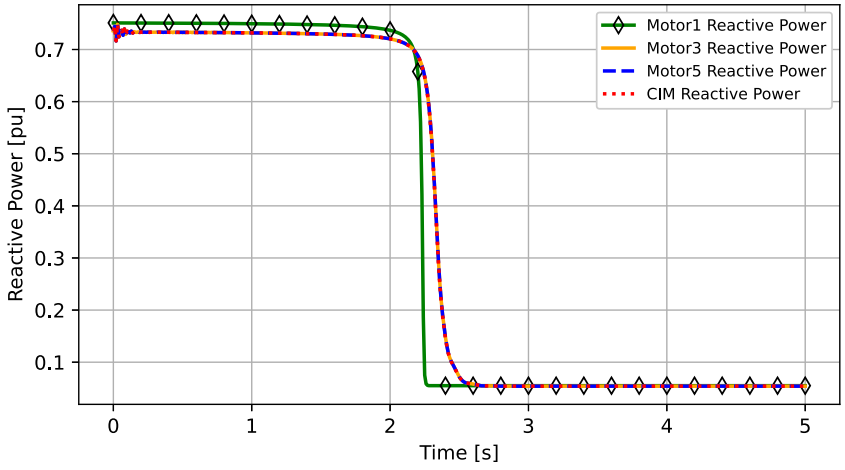
The difference in the model shown in Fig. 5 compared to Fig. 2 is the addition of the power electronics component (*AC/DC - DC/AC*) of the variable speed drive, which allows the user to control the terminal voltage of the motor. The reader can notice that the example does not contain the Volts/Hertz controller component, however, the motor's behavior is altered by changing the parameters *v_start* and *t_start* from real expression components *SVR* and *SSR* respectively. The *SVR* block represents the stator voltage ratio, while the *SSR* represents the synchronous speed ratio. This configuration allows the user to test the motor response via changes in parameters *v_start* and *t_start*. This configuration was chosen to have separate controllable variables for behavior validation purposes, allowing the authors to generate the typical curves presented in the literature [26].

The aforementioned parameters are modified via a Python script that interacts with Dymola. For test 1, the authors increased voltage and frequency (synchronous speed) at a constant ratio until reaching its rated value, which is shown in the black curves in Fig. 6(a). The air gap flux is constant, and thus, the peak electromagnetic torque is also constant in this region. If the stator frequency is increased beyond its rated value, as shown in the red curves (see Fig. 6(a)), then the air gap flux reduces, and consequently, the electromagnetic torque is also reduced.

For test 2, with NEMA type D motors (high slip motors with high peak loads and low efficiency), typically used to drive pumps and fans, a cheap and simple way of achieving variable speed control is through variable-voltage operation. To simulate such scenario in the model displayed in Fig. 5, the *TorqueEquation* was modified to reflect such type of load ($T_L \propto w_r^2$) and the motor parameters were set to represent a NEMA type D motor. In the following simulations, the frequency is



(a) Active Power.



(b) Reactive Power.

Fig. 4. Comparison of induction motor active and reactive power consumption in time.

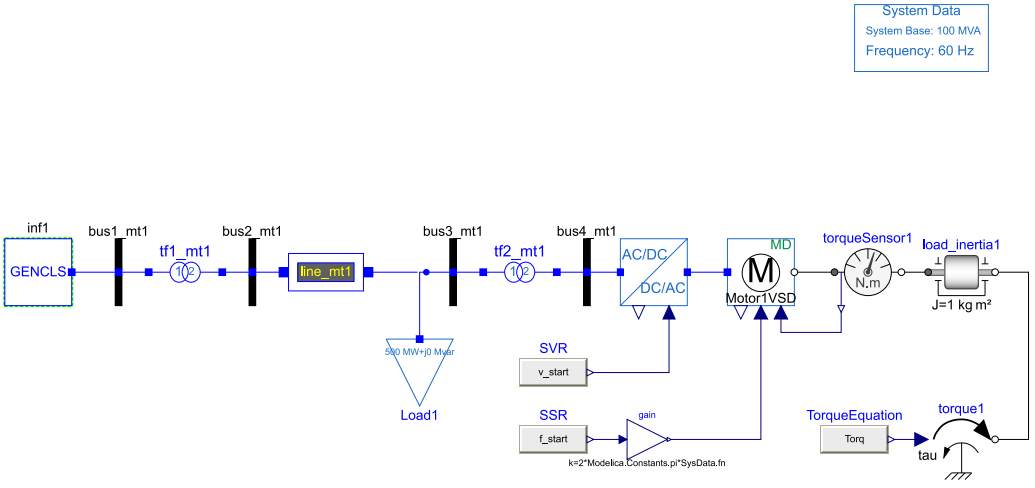
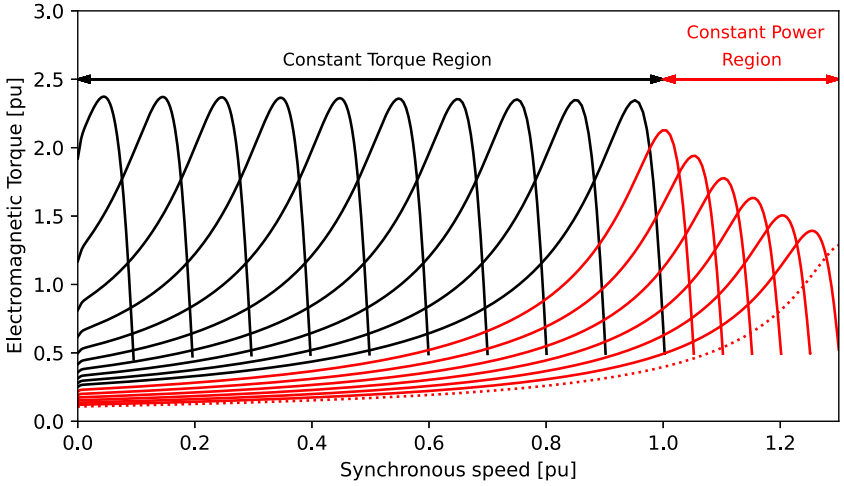
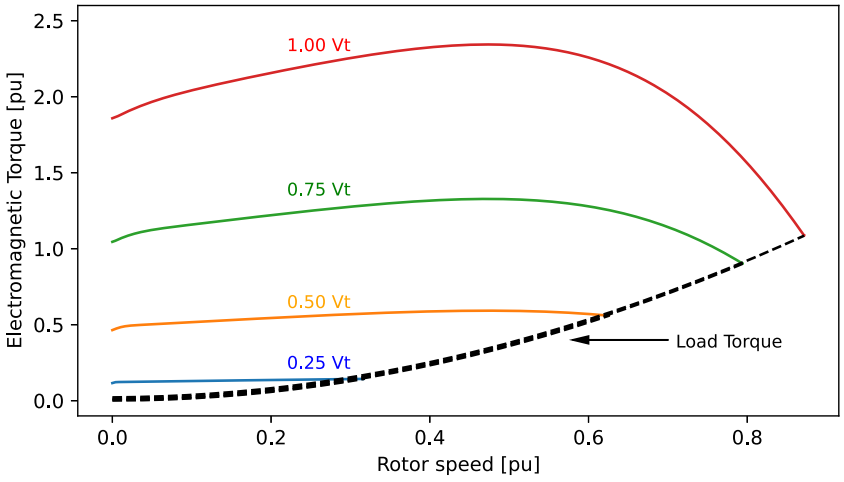


Fig. 5. Example utilized in sweeping voltage and synchronous speed variables.



(a) Mechanical and Electromagnetic Torque comparison for Test 1.



(b) Mechanical and Electromagnetic Torque comparison for Test 2.

Fig. 6. Comparison of induction motor active and reactive power consumption in time.

kept at nominal value while the terminal voltage is changed in order to operate the motor at different speeds. Another validation test that reflects variable-voltage operation in NEMA type D induction motors is the simple and economic method of speed control through terminal voltage variation, as shown in section 2.1 from [26]. Fig. 6(b) depicts four resulting electromagnetic torque curves for four different variable-voltage operations ($0.25Vt$, $0.50Vt$, $0.75Vt$, and $1.00Vt$) all driving a load torque that reflects a fan/blower load.

Similarly to what is shown in [26], the electromagnetic starting and maximum torque increases as the terminal voltage also increases. Because of the high slip operation capability of such type D motor, it operates within a large rotor speed margin, reflected in the four different plots in Fig. 6(b).

4. Impact

`OpenIMDML` is a library that complements the current `OpenIPSL` Modelica Library, adding multi-domain capability to induction motor models. While the positive sequence modeling of motors is the de-facto modeling approach in conventional power system tools used for dynamic analysis, the motor model with an interface to couple with another domain using the Modelica language is novel and allows studying the dynamical interaction between the electrical power grid and other domains. This flexibility is only possible thanks to the implementation utilizing the Modelica modeling language.

Another impactful contribution of the library is in the implementation of a VSD model that allows simulating the speed control operation over induction motors, a capability that is not found nor modeled in standard phasor domain power system simulation tools. Because the new motor models can be coupled to the VSD model, this also implies that the motor models have frequency-varying stator and rotor impedances, which is also an improvement to the de-facto modeling tools. Considering the numerous advantages of using VSD's in current induction motor applications, the Modelica-based VSD model is a crucial addition for emerging research areas such as grid-interactive building energy systems. As an example of the usage of such models, paper [4] discusses the implementation and utilization of a multi-domain Type I induction motor along with a variable speed drive. These models served as the foundation for the development of the `OpenIMDML` Library and have become integral components of the library. The incorporation of thermo-fluid and mechanical interfaces introduces the capability to explicitly simulate transient load disturbances in accordance with their constitutive physics, thereby facilitating dynamic interactions between electrical and hydraulic contingencies. Furthermore, the paper presents a straightforward example of a multi-domain power and thermo-fluid model, featuring a motor interconnected with a water pump and its associated piping system.

5. Conclusions

This paper provides a brief overview of OpenIMDML, a Modelica-based library that was created to complement the OpenIPSL Modelica Library, with the possibility of representing induction motors, variable speed drives, and their loads in a multi-domain manner. The implemented motor models, in conjunction with the VSD, bridge the gap that is currently experienced with conventional power system modeling approaches using phasor-based tools. The new models give the user control over the rotational speed of the motor models and further enhance the portrayal of load processes that are interlinked with the mechanical domain of the motor and other domains downstream. Additionally, because of the implementation of the OpenIMDML Library utilizing the Modelica modeling language, the user is capable of utilizing the library and its models in different Modelica-and-FMI-compliant simulation environments. Users are free to select whatever simulation environment they are most comfortable with, being free or commercial.

CRediT authorship contribution statement

Fernando Fachini: Conceptualization, Formal analysis, Investigation, Methodology, Software, Validation, Writing – original draft. **Marcelo de Castro:** Conceptualization, Methodology, Software, Writing – review & editing. **Tetiana Bogodorova:** Conceptualization, Project administration, Supervision, Writing – review & editing. **Luigi Vanfretti:** Conceptualization, Software, Funding acquisition, Investigation, Project administration, Resources, Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The library can be found on GitHub in the following webpage: https://github.com/ALSETLab/OpenIMDML.

Acknowledgments

This paper is based upon work supported by the U.S. Department of Energy's Office of Energy Efficiency and Renewable Energy (EERE) under the Advanced Manufacturing Office, Award Number DE-EE0009139.

This paper was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

References

[1] Milano F. Power system modelling and scripting. Springer Science & Business Media; 2010.
[2] Schichl H. Models and the history of modeling. In: Modeling languages in mathematical optimization. Springer; 2004, p. 25–36.
[3] Siemens PTI. PSS®E 34.2.0 model library. Schenectady, NY: Siemens Power Technologies International; 2017.
[4] Fachini F, De Castro M, Liu M, Bogodorova T, Vanfretti L, Zuo W. Multi-domain power and thermo-fluid system stability modeling using modelica and openipsl. In: 2022 IEEE power & energy society general meeting. IEEE; 2022, p. 1–5.
[5] Otter M, Elmqvist H. Modelica overview. 2001, URl: https://www.modelica.org/education/educationalmaterial/lecture-material/english/ModelicaOverview.pdf.
[6] Electrified power train library, 3DEXPERIENCE platform. 2023, URL https://my.3dexperience.3ds.com/welcome/compass-world/rootroles/systems-electrified-power-train-library?role=true. [Online; Accessed 25 October].
[7] Electric power library. Modelon; 2023, URL https://modelon.com/library/electric-power-library/. [Online; Accessed 25 October].
[8] Electrification library. Modelon; 2023, URL https://modelon.com/library/electrification-library/. [Online; Accessed 25 October].
[9] Milano F. An open source power system analysis toolbox. IEEE Trans Power Syst 2005;20(3):1199–206.
[10] Milano F, Vanfretti L, Morataya JC. An open source power system virtual laboratory: The PSAT case and experience. IEEE Trans Educ 2008;51(1):17–23.
[11] Schweiger G, Gomes C, Engel G, Hafner I, Schoeggl J, Posch A, et al. An empirical survey on co-simulation: Promising standards, challenges and research needs. Simul Model Pract Theory 2019;95:148–63.
[12] Modelica libraries. Modelica Libraries - Modelica Association; 2023, URL https://modelica.org/libraries.html. [Online; Accessed 31 August].
[13] Baudette M, et al. OpenIPSL: Open-instance power system library—update 1.5 to "itesla power systems library (iPSL): A modelica library for phasor time-domain simulations". SoftwareX 2018;7:34–6.
[14] de Castro M, Winkler D, Laera G, Vanfretti L, Dorado-Rojas SA, Rabuzin T, et al. Version [OpenIPSL 2.0. 0]-[itesla power systems library (iPSL): A modelica library for phasor time-domain simulations]. SoftwareX 2023;21:101277.
[15] Fritzson P, Engelson V. Modelica—A unified object-oriented language for system modeling and simulation. In: European conference on object-oriented programming. Springer; 1998, p. 67–90.
[16] Fitzgerald AE, Kingsley C, Umans SD, James B. Electric machinery, vol. 5. McGRAW-hill New York; 2003.
[17] Modelica specification. Modelica; 2023, URL https://specification.modelica.org/. [Online; Accessed 31 August].
[18] Fritzson P. Principles of object-oriented modeling and simulation with modelica 3.3: A cyber-physical approach. John Wiley & Sons; 2014.
[19] Del Rosso AD, et al. Stability assessment of isolated power systems with large induction motor drives. In: 2006 IEEE/PES T&D conf. & expo.: Latin America. 2006, p. 1–6.
[20] Lee K, Frank S, Sen PK, Polese LG, Alahmad M, Waters C. Estimation of induction motor equivalent circuit parameters from nameplate data. In: 2012 North American power symposium. IEEE; 2012, p. 1–6.
[21] Modelica tools. Modelica Tools - Modelica Association; 2023, URL https://modelica.org/tools.html. [Online; Accessed 31 August].
[22] Brück D, Elmqvist H, Mattsson SE, Olsson H. Dymola for multi-engineering modeling and simulation. In: Proceedings of modelica. 2002, 2002, p. 1–6.
[23] Fritzson P, Aronsson P, Lundvall H, Nyström K, Pop A, Saldamli L, et al. The OpenModelica modeling, simulation, and development environment. In: 46th conference on simulation and modelling of the scandinavian simulation society (SIMS2005), Trondheim, Norway, October 13-14, 2005. 2005.
[24] Chow JH, Sanchez-Gasca JJ. Power system modeling, computation, and control. John Wiley & Sons; 2020.
[25] Chow JH, Cheung KW. A toolbox for power system dynamics and control engineering education and research. IEEE Trans Power Syst 1992;7(4):1559–64.
[26] Bose BK. Power electronics and AC drives. Englewood Cliffs; 1986.