

Developing grid-forming converter controller DLL for real-time HIL simulations

Singh, Utkarsh; Singh, Ravi; Popov, Marjan; Lekić, Aleksandra

DOI

[10.1007/s00502-024-01302-0](https://doi.org/10.1007/s00502-024-01302-0)

Publication date

2025

Document Version

Final published version

Published in

E & I Elektrotechnik Und Informationstechnik

Citation (APA)

Singh, U., Singh, R., Popov, M., & Lekić, A. (2025). Developing grid-forming converter controller DLL for real-time HIL simulations. *E & I Elektrotechnik Und Informationstechnik*, 142(1), 51-70.
<https://doi.org/10.1007/s00502-024-01302-0>

Important note

To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Copyright

Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy

Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.



Developing grid-forming converter controller DLL for real-time HIL simulations

Utkarsh Singh · Ravi Singh · Marjan Popov · Aleksandra Lekić

Received: 26 September 2024 / Accepted: 13 December 2024
 © The Author(s) 2025

Abstract There is a lack of knowledge on the development of a controller model into a black box dynamically-linked library (DLL) file which can be flexibly tested with a power system model through a real-time co-simulation platform. Developing a test bed to demonstrate the performance and compliance analysis enhances the approval of novel control and protection strategies. This paper presents the development of a controller DLL and a test bench that integrates this DLL with a real-time simulator. The DLL contains a previously developed grid-forming controller for the type-3 wind turbine and works in co-simulation with the power system model on Real-Time Digital Simulator (RTDS) in real-time.

Keywords RTDS · Real-time simulation · Dynamically Linked Library (DLL) · Black box control · Control-Hardware-in-the-Loop (cHIL)

Entwicklung einer netzbildenden Konverter-Controller-DLL für Echtzeit-HIL-Simulationen

Zusammenfassung Es mangelt an Wissen über die Entwicklung eines Controllermodells in eine Black-Box-DLL-Datei (Dynamic Linked Library), die flexibel mit einem Stromnetzmodell über eine Echtzeit-Co-Simulationsplattform getestet werden kann. Die Entwicklung eines Prüfstands zur Demonstration der Leistungs- und Konformitätsanalyse verbessert die

Zulassung neuer Steuerungs- und Schutzstrategien. In diesem Dokument wird die Entwicklung einer Controller-DLL und eines Prüfstands vorgestellt, der diese DLL in einen Echtzeitsimulator integriert. Die DLL enthält einen zuvor entwickelten netzbildenden Controller für die Windturbine Typ 3 und arbeitet in Echtzeit in Co-Simulation mit dem Stromnetzmodell auf einem Real Time Digital Simulator (RTDS).

Schlüsselwörter RTDS · Echtzeitsimulation · Dynamisch verknüpfte Bibliothek (DLL) · Black-Box-Steuerung · Control-Hardware-in-the-Loop (cHIL)

1 Introduction

The power grid infrastructure is changing at a fast pace. To achieve a power grid suitable to support carbon neutrality, penetration of power electronics inverter-based resources (IBRs) such as wind turbines (WTs), HVDC systems, and FACTS devices is increasing rapidly.

With the evolution of the grid, the modeling requirements of wind plants and other power electronics devices change accordingly. A Transmission System Operator (TSO) normally asks for plant and turbine electrical models for grid-code compliance and interaction studies. Such models can help avoid compliance issues before the wind turbines (WTs) connect to the grid. The functional performance of the IBRs depends a lot on the controller of the IBRs and IBR plants. Depending upon the implementation and simplifications, the representation of the controllers in software-only models might differ from the actual control code, which is embedded in the controller hardware. This difference might result in divergence in the compliance test results and interaction studies. A method to solve this problem is to assess power electronics-based devices' performance, compliance,

U. Singh · M. Popov · A. Lekić (✉)
 Department Electrical Sustainable Energy, Faculty of
 Electrical Engineering, Mathematics and Computer Science,
 Delft University of Technology, Mekelweg 4, 2628 CD Delft,
 The Netherlands
 a.lekic@tudelft.nl

R. Singh
 DNV, Arnhem, The Netherlands

and interaction using a Controller Hardware-in-Loop (CHiL) test bed [1, 2]. CHiL tests incorporate the IBR inverter controller's hardware (replica), which run the original control software. This controller interacts with the grid (measurements) and other components of the IBRs (inverters and protection) in a real-time simulation (RTS) environment. Applying such a CHiL method to simultaneously integrate multiple devices on a plant and system level along with their controllers might pose practical challenges in acquiring hardware from (different) vendors and concerns regarding Intellectual Properties (IPs) when integrating them in a single test bed. A way to still use high-fidelity controller representation in grid compliance studies and avoid such problems could be to use controller DLLs (dynamically-linked libraries) instead of hardware replicas. The controller DLLs would still contain the actual controller source code, be relatively easier to transfer, and protect the vendors' IPs. The only requirement is that the controller DLLs are run in real time to be integrated with the rest of the test bed. However, integrating a DLL into an RTS environment is a significant challenge since DLL files are based on the Windows operating system (OS), which is non-deterministic and, as such, incompatible with RTS. However, it is worth noting that the hardware used can still have the differences between the electromagnetic transient simulation performed offline [3]. In other words, in papers [1, 2], it is shown that the models show differences depending on whether the implementation of the power electronic device is in the FPGA or a CPU. Furthermore, even when using

the same real-time simulator, such as RTDS, different types of models can show deviations [4, 5]. The reasons for this are many, starting from adding reactances for cross-coupling of the model simulated with multiple parallel CPUs and others.

Another advantage of having a black-box model in DLL form is the time-saving and ease of plug-and-play capability with power system models developed in various modeling software [3] like PSCAD, Powerfactory, RSCAD [4, 5], Hypersim, MATLAB/Simulink, etc. When a black-box controller model exists, it can be easily used to show the functionality and validity of the controller algorithm without losing the IP. A black box model greatly improves interoperability and easily shares the controller models with vendors and competitors.

This paper presents a method for developing DLLs as real-time DLL (RT-DLL) and integrating them into a CHiL test bed for performance and compliance tests. The test results of RT-DLL are then compared with the original grid-forming controller developed for wind turbine converters in Simulink. A normal static or dynamically linked library file is a Windows-based file that does not support operation in real-time. To make real-time simulations possible, the Windows operating system (OS) must be temporarily converted to a real-time OS (RTOS). This is done with third-party software from IntervalZero. IntervalZero provides an RTOS platform that supports determinism or hard real-time on multi-core processors while co-residenting the Windows OS. A real-time DLL of the converter controller code and other associated

Fig. 1 Simulink model configuration for code generation

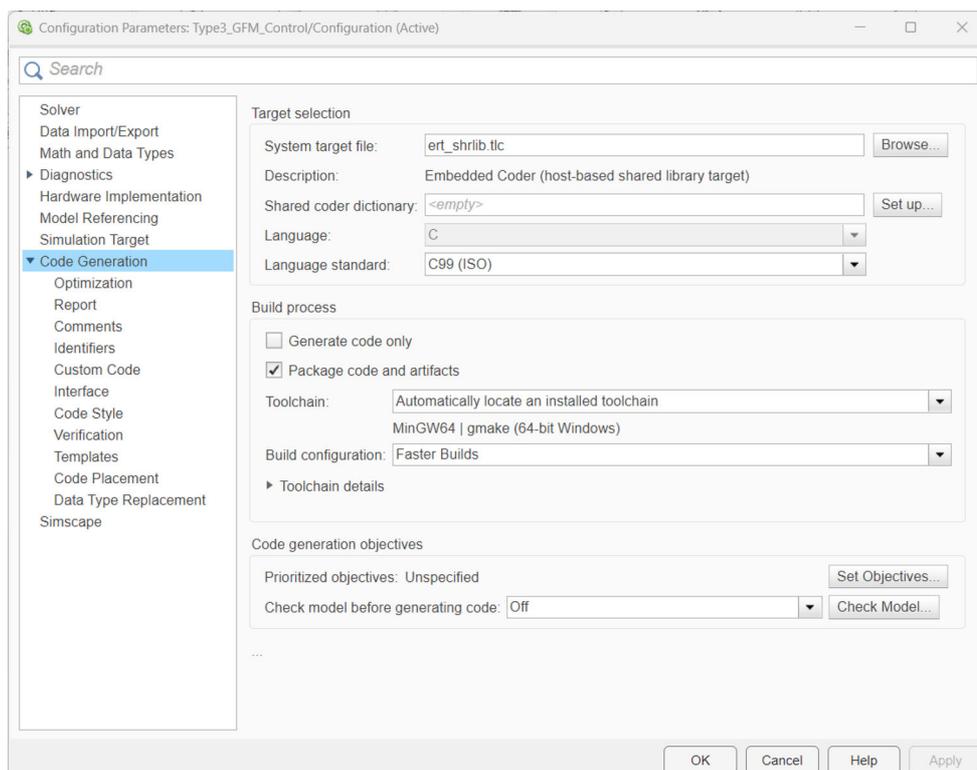
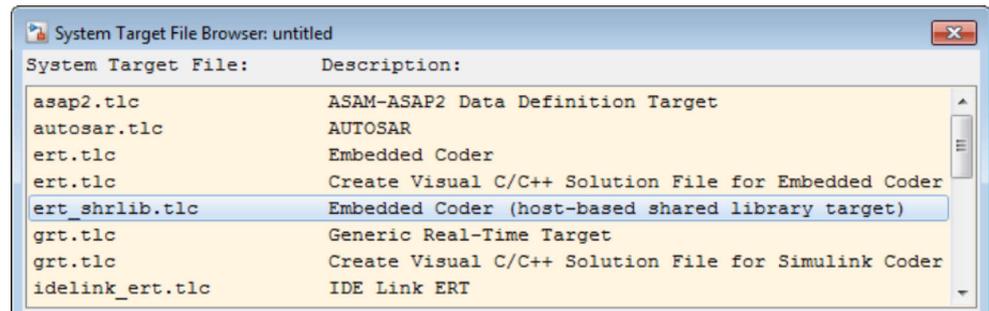


Fig. 2 Target file selection under model configuration



files is developed using the IntervalZero software development kit (SDK) and Microsoft Visual Studio (VS).

Sect. 2 presents a step-by-step process of converting a grid-forming (GFM) converter controller for the Type-3 WT into a DLL black box. This is followed by developing the software-in-the-loop (SIL) setup where the developed controller DLL is run in real-time to control the Type-3 WT simulated by a real-time simulator. Sect. 3 presents the details for the GFM control design. Accomplished tests and results obtained with the SIL setup with the black-box controller is presented in Sect. 4 followed by conclusions in Sect. 5.

2 Black-box controller design

This section provides, in detail, the step-by-step methodology followed in developing the controller as a black-box model.

2.1 Developing MATLAB/Simulink-based controller

A black-box model of a controller is a combination of an executable application (.exe/, .a/, .rtss) file along with a dynamically-linked library (.lib/, .dll/, .rtdll). The executable file is produced through an application code which is written in C-programming language in this case. The application code calls the functions exported by the library file, which in return requires the controller model's source code files (.c and .h files).

There is no in-built option for generating source code files in an RSCAD model. Hence, an alternative strategy was followed. Generating source code files is possible with Simulink using the Code Generation and Embedded Coder toolbox. By configuring the code generator (see Fig. 1) to use the system target file `ert_shrllib.tlc` (see Fig. 2), one can generate a source code package file of the model with all the necessary .c and header files. Code generation for that system target file exports:

Fig. 3 Controller model mask in Simulink

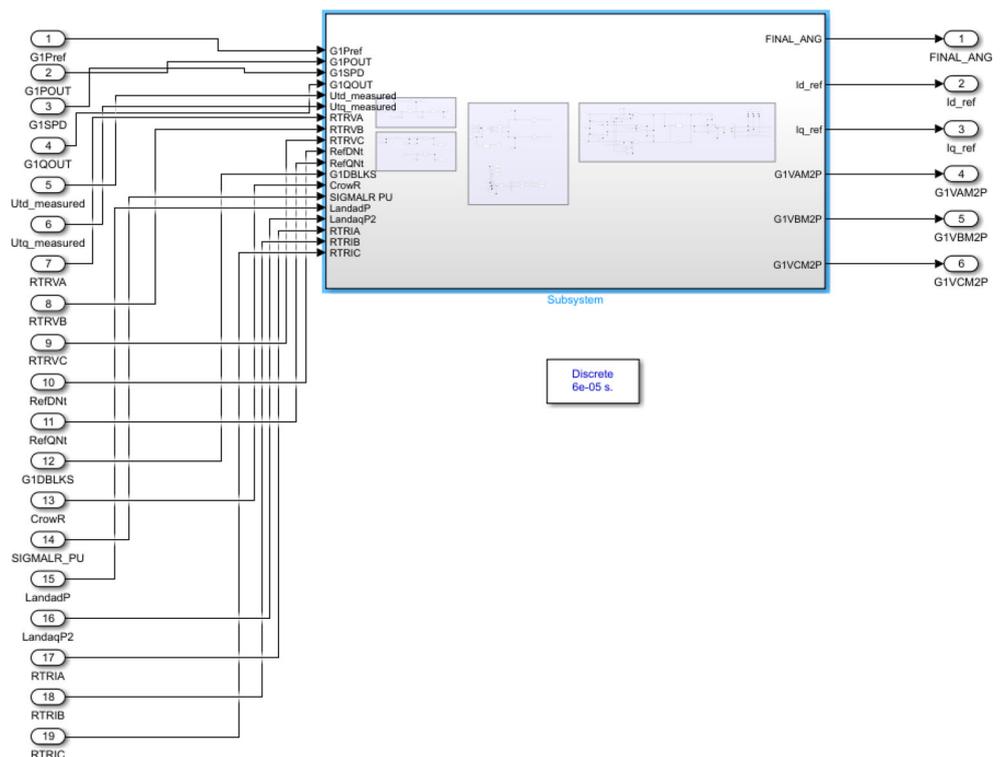


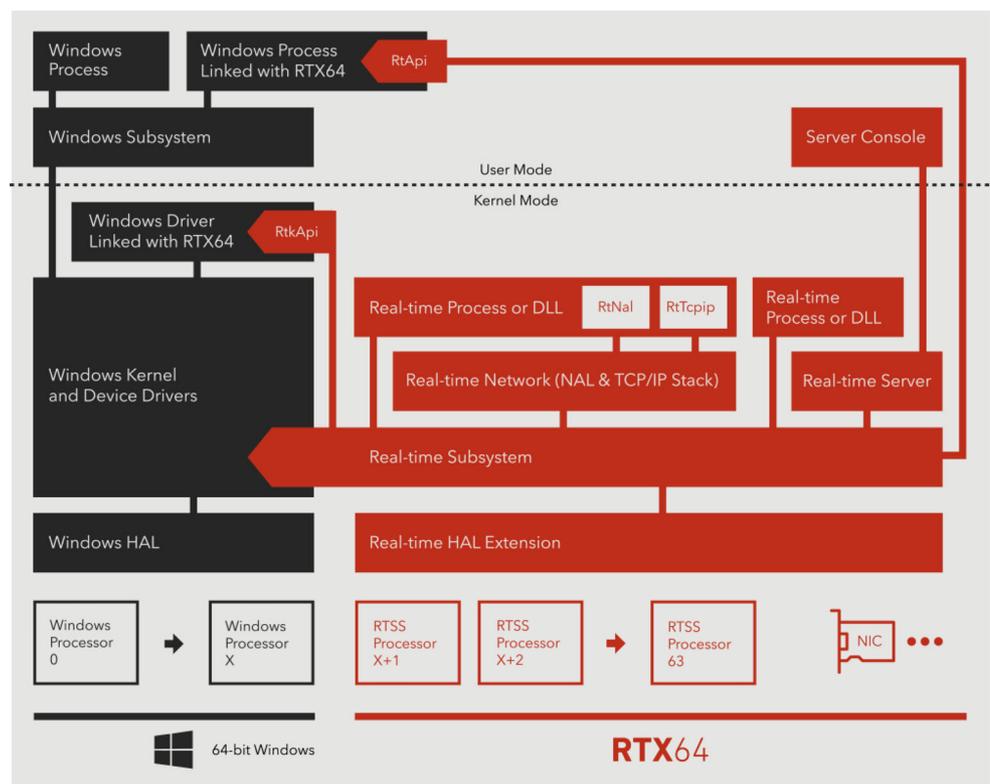
Table 1 Generated code files by simulink embedded coder

File name	Details
1) model.c	Contains entry point functions for all the code implemented in the algorithm
2) model.h	Declares model data structures and a public interface to the model entry Points and data structures
3) ert_main.c	An example main that embedded code generates by default that shows How to call the generated code
4) model_types.h	Provides forward type declarations for the model and its parameters
5) model_data.c	Contains the declarations for the parameters, data structures and the Constant block I/O data structure
6) rtwtypes.h	Defines macros, data types and structures required by embedded coder
7) model_private.h	Contains local macros and local data that are required by the model and Subsystems

1. Variables and signals of type ExportedGlobal as data.
2. Real-time model structure (model_M) as data.
3. Functions essential to executing your model code.

Hence, the control model for the grid-forming control of the Type-3 wind turbine is built on Simulink. The mask of the Simulink model can be seen below in Fig. 3.

Fig. 4 Interval zero operation, [6]



The Simulink model has 19 inputs and 6 outputs for this specific test case. Using suitable communication protocol details, these input/output signals must be communicated between the controller black-box and RSCAD models. The input and output signals were originally internal control signals in the RSCAD model.

Once the target file is configured and the constructed model built, the generated code files can be accessed in the directory where the model exists. Table 1 shows the generated modules for any Simulink model and their specific details.

2.2 Building the real-time library and application file

A normal static or dynamically linked library file is a Windows-based file that does not support operation in real-time. To make the real-time simulations possible, the Windows operating system (OS) has to be temporarily converted to a real-time OS (RTOS). This is done by a third-party software named IntervalZero. IntervalZero is an RTOS Platform that supports determinism or hard real-time on multi-core processors while co-residenting the Windows operating system. Two IntervalZero components are mainly required to achieve this:

1. **IntervalZero RTX64 Software Development Kit (SDK):** comes with headers and libraries required for developing real-time applications and DLLs, provides support for linking real-time functional-

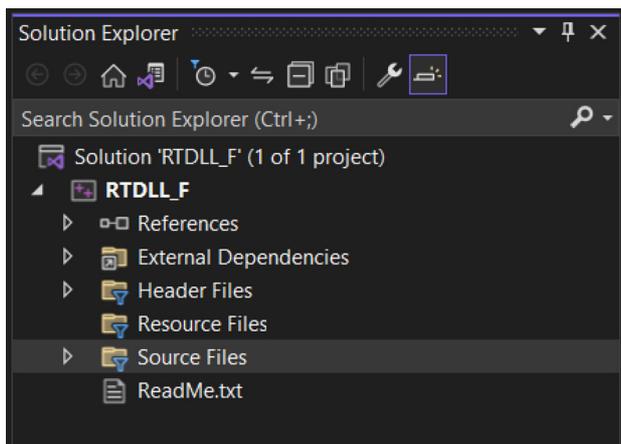


Fig. 5 Solution explorer window

and integrates seamlessly with Microsoft Visual Studio. It provides two project templates for the Visual studio – RTX64 RTDLL for building real-time dynamically-linked libraries and RTX64 Application for creating real-time applications.

2. **IntervalZero RTX64 Runtime:** Adds a real-time subsystem (RTSS) to Windows that delivers hard real-time performance. It is deployed on your target system with your real-time application. Fig. 4 below shows the overview of this process.

Using the RTX64 RTDLL template, a library file was developed. The steps to be followed are as follows:

1. Select the Real-time Dynamically-link Library (RTDLL) template from MS Visual Studio.
2. In the *Configure your new project page*, enter the project name. Leave the default *Location and Solution* name values. Set a *Solution to Create a new solution*. Uncheck *Place the solution and project in the same directory* if they are checked and then create the project.
3. When the solution is created, it can be seen that the generated project and source files in the Solution

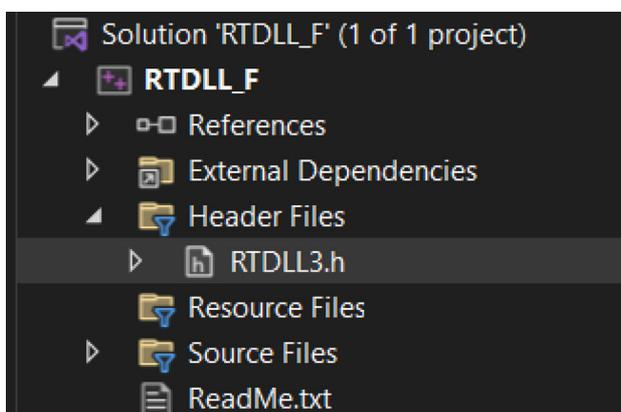


Fig. 6 Header file

Explorer window in Visual Studio (see Fig. 5 below). Next, necessary header files are created and added to declare the functions of the RTDLL exports.

4. The solution of the RTDLL template already contains a default header file. Now, declare the functions contained in the controller source code (see Fig. 6 below), including necessary APIs and declaring an example function. Remove the example function and add the function declarations contained in model.h file (seen earlier) to this skeletal header file. Then, the user can export the necessary functions and variables of interest to use in an application.
5. Next, the user can add source files containing the definitions of our declared functions to our RTDLL. The RTDLL template contains a skeletal source file in which the function definitions used in the model.c file (seen earlier) will be added. Furthermore, all the rest of .c files that were created in the source code package are added as source files here (see Fig. 7 below).
6. The final step is to update the Windows SDK version of the template (see Fig. 8) in the one in the current system to avoid build errors. Also, the source code header file's directories must be added under 'Additional Include directories' (see Fig. 9 below).
7. The final step is to build the code that generates the standard library file (.lib) and real-time dynamically-linked library file (.rtdll) in the current directory, as shown in Fig. 10 below.

To build the communication between the developed library file and the RSCAD model, a wrapper code/application code needs to be built to generate an application that interacts with RTDS. Using the RTX64 Application template, we developed the wrapper code

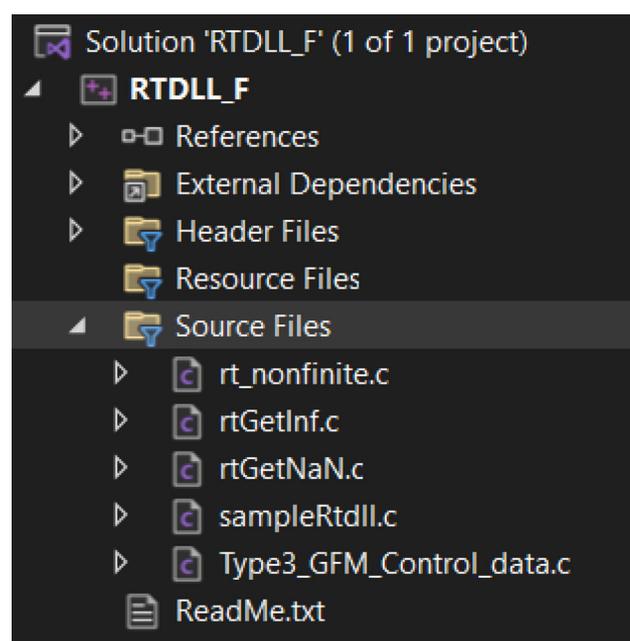
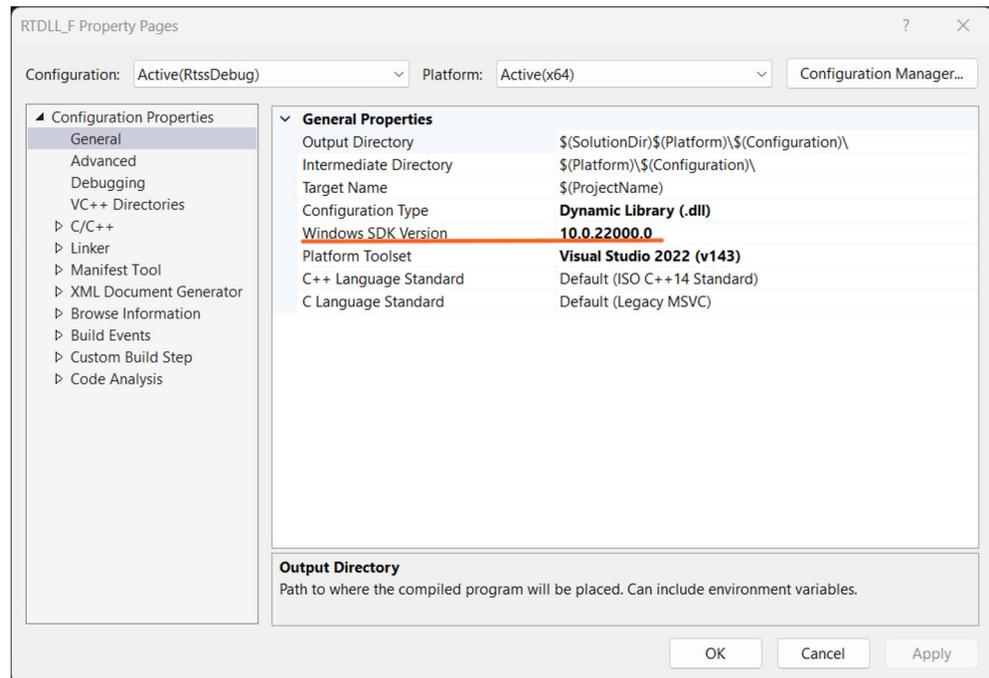


Fig. 7 Source file

Fig. 8 Windows SDK version update



to generate the real-time application (.rtss) file. The steps to be followed are as follows:

1. Select the Real-time Application (RTSS) template from MS Visual Studio and then choose *Next*
2. In the *Configure your new project page*, enter the project name. Leave the default *Location and Solution* name values. Set a *Solution to Create* a new solution. Uncheck *Place solution and project* in the same directory if it's checked. Then choose *Create* to create the template project.
3. Next, to call the functions in your source code, the application project must include your library's header (.h) file (seen previously). The user is advised to copy this header file into the user-defined client app project, and then to add it to the project as an existing item. However, if the user works simultaneously on the DLL and application code, the header files could get out of synchronization. To avoid this issue, set the *Additional Include Directories* path in the user-defined project to in-

Fig. 9 Adding source code directories

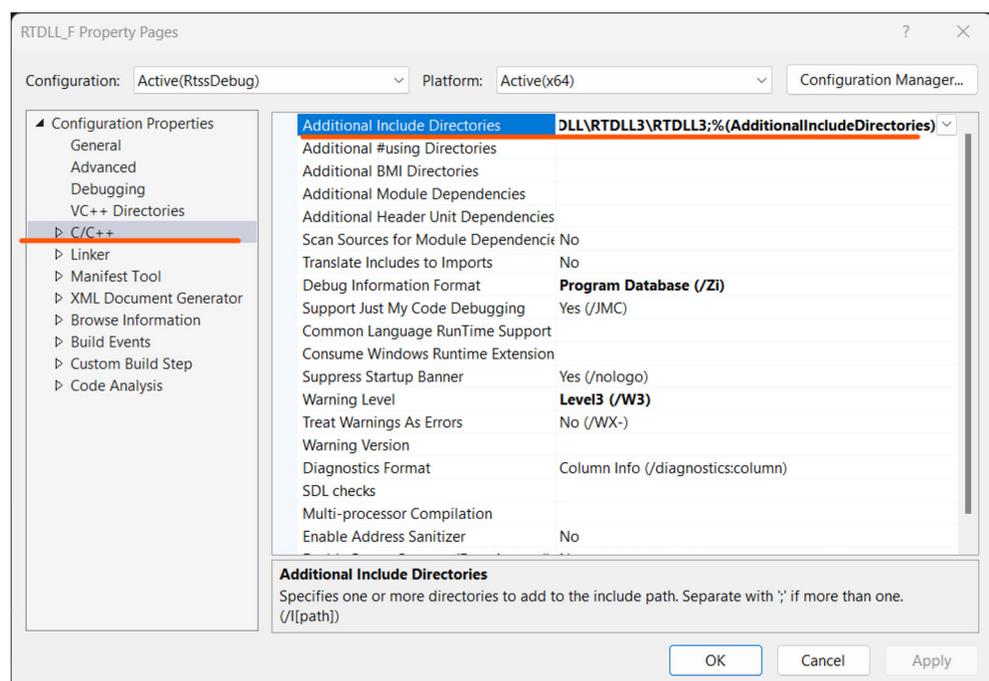


Fig. 10 Generated library files

 RTDLL_F.exp	24-03-2023 01:49	Exports Library File	2 KB
 RTDLL F	24-03-2023 01:49	lib	4 KB
 RTDLL_F	24-03-2023 01:49	Text Document	1 KB
 RTDLL_F.pdb	24-03-2023 01:49	Program Debug D...	5,652 KB
 RTDLL F	24-03-2023 01:49	RTDLL File	901 KB

- clude the path to the original header in the same way as it was shown previously in Fig. 9.
- When the user builds the client app, the error list shows several LNK2019 errors. That is because the project misses some information: You have not yet specified that the project is dependent on the RTDLL3.lib library. Besides, you have not told the linker how to find that .lib file.
 - To add the DLL import library to the project, right-click on the *Application* node in *Solution Explorer* and choose *Properties* to open the *Property Pages* dialog.
 - In the left pane, select *Configuration Properties* > *Linker* > *Input*. In the property pane, select the drop-down control next to the *Additional Dependencies* edit box, and then choose *Edit*. Add the name of the .lib file along with the extension (see Fig. 11 below).
 - In the left pane, select *Configuration Properties* > *Linker* > *General*. In the property pane, select the drop-down control next to the *Additional Library Directories* edit box, and then choose *Edit*. Double-click in the top pane of the *Additional Library Directories* dialog box to enable an edit control. In the edit control, specify the path to the location of the RTDLL.lib file (see Fig. 12 below). You might also need to add dependencies on RTX64 libraries depending on your MS Visual Studio version.
 - Now, the wrapper code needs to be built as the .c file under the source files of the application project. While doing a literature survey about connecting a dynamically linked library file and RTDS, it was known that the communication is established through a UDP connection (details of this will be discussed in the next section). There is a simple loopback example UDP_SKT_Loopback available in the Example cases of RTDS that includes the sample wrapper application udpSKT_echo.c, which simply loopbacks the incoming data from the RTDS Novacor UDP port. This code is used as the starting point as it includes the code to create the socket and handle the incoming and outgoing data.
 - First, the code defines and maps the number of incoming and outgoing signals. The part of the example code that loops back the data must be changed to include the step function of the controller source code defined in the library file (seen earlier in Fig. 10). Also, the initializing and ter-

Fig. 11 Including library name

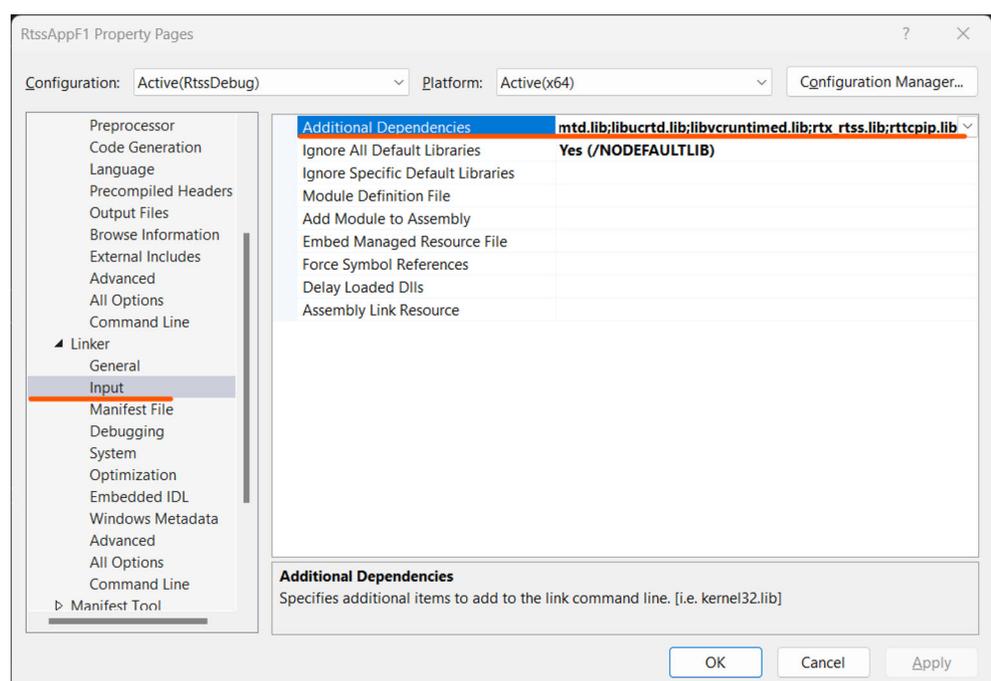
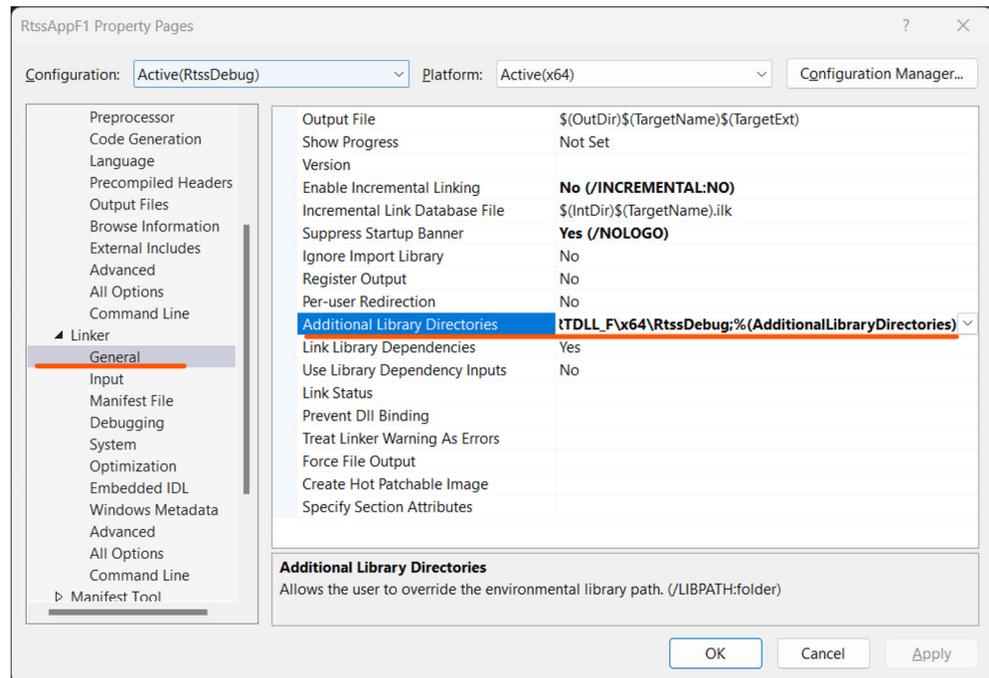


Fig. 12 Including library path



minating functions must be included outside the main loop.

- Finally, the wrapper code is built to generate the real-time application (.rtss file); in our case Rtss-AppF1.rtss file.

This concludes the subsection. The following subsection deals with the details of the process involved in building the communication between the application and the RTDS Novacor to make the controller work as a black box.

2.3 Developing the software-in-the-loop (SIL) setup

Development of the SIL setup essentially involves building communication between the controller black-box model (RTDLL file seen previously) and the RTDS model on Novacor. The communication takes place using the User Datagram Protocol (UDP). From the NovaCor side, the ETHERNET UDP port is

used for communication. This is the bottom right-most port on the back of the NovaCor (shown in Fig. 13). This port needs to have an SFP Copper transceiver. The supported copper transceivers can be found in RTDS documentation.

Shown in Fig. 14, the RSCAD component `_rtds_Ethernet_UDP_SKT` (shown in the Figure below) is used to set up a UDP interface to external equipment (PC with black-box model). The RSCAD component represents the interface from the NovaCor Chassis UDP port. The component can send data (Tx), receive data (Rx), or both. The current limit of sending and receiving signals is 100 for each. The main configuration parameters are shown in Fig. 15.

All the incoming signals to the RSCAD from the controller model and outgoing signals from the RSCAD to the controller model are defined with the same names used initially in the RSCAD and Simulink models.

Fig. 13 NovaCor Chassis, UDP PORT 32



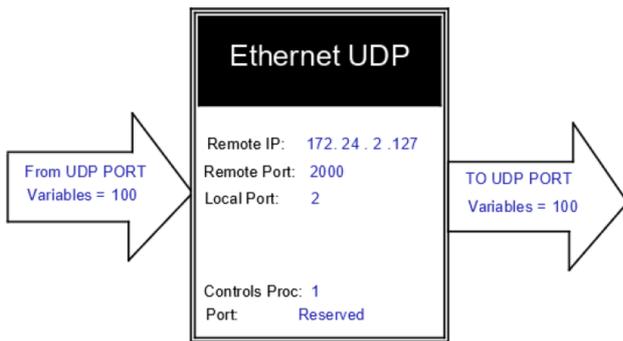


Fig. 14 UDPSKT component in RSCAD software

With the addition of the UDP Ethernet port, the NovaCor simulator can communicate to a remote device over a LAN/WAN using UDP socket protocol. A typical connection configuration (through the switch) is shown in Fig. 16. Similarly, an Ethernet cable can be directly connected from the NovaCor UDP port to the PC network card, i.e., without an intermediate switch. We have made a direct connection using a PC NIC card.

From the PC side, the developed application (.rtss file) running on IntervalZero is needed. This would be the wrapper code that accepts the data from NovaCor, maps the incoming data to the user controller inputs, calls the main controller step function, maps the controller outputs to the outgoing data, and sends the data to NovaCor.

For communication with the PC, one can use either the RTX64 RT-TCP Stack layer or the RTX64 Network Abstract Layer (NAL) for the interface. In this work, the TCP stack and a socket connection are used between the PC NIC card and the NovaCor UDP port. To use IntervalZero successfully, a compatible PC and network card hardware was required. The list of compatible PC network cards can be found on the IntervalZero website [8]. The Intel I210 Copper-only Ethernet Network Card is used in this case, as this was recommended in the IntervalZero manual [8]. For the PC, the CPU virtualization and Hyper-threading are to be switched off in the BIOS. The original drivers of

the installed PC ethernet card must be replaced by the IntervalZero **RTNALIGB.rtdll** driver which comes with IntervalZero installation; this is done through the configuration settings of the PC card inside the Ethernet adapter settings menu. Also, one has to make sure that the port numbers and the IP address assigned to both the NovaCor UDP port and PC ethernet port match in the wrapper code (in Remote PC) and in the UDP_SKT component (In RSCAD model). The NOVACOR UDP port IP can be set in the RSCAD manual [7].

To avoid latency in the black box controller output signal, the number of cores assigned to the windows and the IntervalZero must be re-assigned through the IntervalZero control panel. As a rule of thumb, at least three cores should be assigned to Windows to avoid serious latency issues.

3 Grid forming control design

With the rapid development of wind power worldwide, the unique characteristics of wind power's static and dynamic responses have emerged new major challenges related to the adequacy and stability of the modern power system. One of the prominent concerns regarding the increasing penetration of wind power is its impact on power system frequency stability due to the lack of positive response from the installed WTs to frequency disturbances, i.e., large load fluctuations or generating unit losses occurring in the power grid [9]. Therefore, the dynamic frequency support capability of wind power will be much needed in the near future to ensure that frequency stability is not compromised. It is known that almost all of the installed WTs currently are adopting the typical vector control method based on the phase-locked-loop (PLL) synchronizing technique. The PLL performs well for a strong grid where the grid dictates voltage angle magnitudes to the controls of the WT's converter. Such an arrangement where converter controls follow the voltage angle and voltage magnitude inputs from a strong grid and behave as a current source is known as a grid-following (GFL) control strategy.

Fig. 15 UDPSKT component main configuration menu

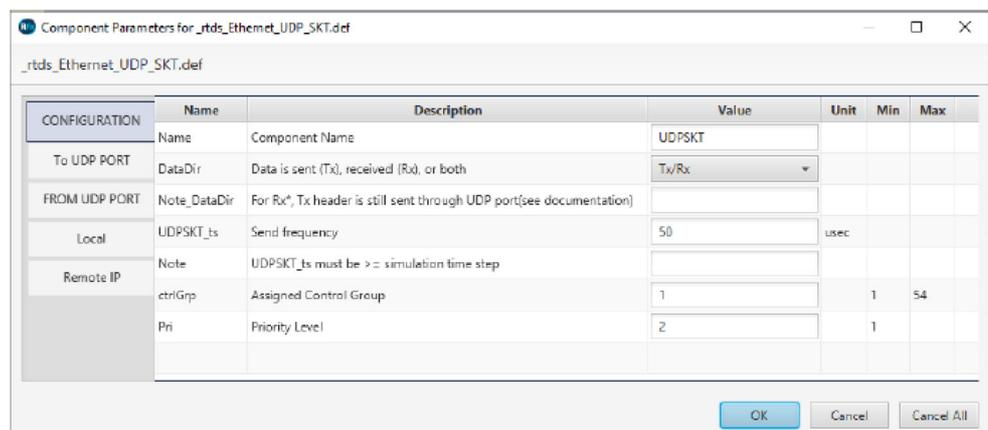


Fig. 16 UDP connection between RTDS Novacor and host PC, (RSCAD User manual [7])

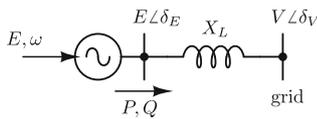
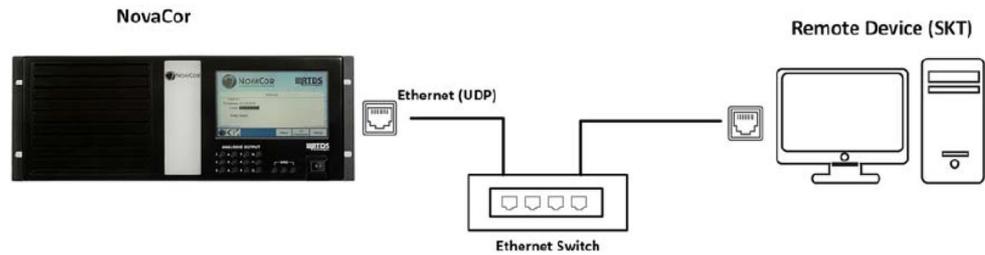


Fig. 17 Basic concept of grid forming [10]

$$Q = \frac{E^2 - EV \cos \delta_p}{X_L} = \frac{E(E - V)}{X_L} \quad (1b)$$

where P is nearly linear with the phase difference δ_p , and Q is almost linear with the internal voltage magnitude E . This decoupling reduces the controller design complexity and is known as the power-frequency ($P - f$) droop relationship and the reactive power-voltage magnitude ($Q - V$) droop relationship, respectively.

On the other hand, power fluctuations may occur when large-scale WTs are located in a weak grid. As the connected AC system's short circuit ratio (SCR) is reduced, the resonances and nonlinearity due to the PLL become more prominent and potentially make the Voltage Source Converter (VSC) more difficult to control. A modern power grid with a high penetration of renewable generators resembles a weak grid with a low SCR value intermittently due to the lower reliability of renewable sources. Hence, the concept of novel grid forming (GFM) control has evolved.

A GFM converter fundamentally behaves as a voltage source behind a coupling reactance X_L , which controls both the voltage magnitude E and angular frequency ω , as shown in Fig. 17 [10, 11]. The coupling reactance X_L plays a critical role in controller design. By properly sizing the coupling reactance X_L , the active power P and reactive power Q are decoupled [12]. As shown in Eqs. (1a) and (1b):

$$P = \frac{EV}{X_L} \sin \delta_p = \frac{EV}{X_L} \delta_p \quad (1a)$$

3.1 MIGRATE benchmark test case

The MIGRATE project [10] benchmark model consists of a point-to-point terminal HVDC link connected to a grid with loads, WTs, photovoltaic solar farms, and conventional generation. An infinite grid with the possibility of modifying its strength has also been included [13]. A typical network looks like a benchmark model of the IEEE upgraded with renewable sources (i.e., Type-3 WT, Type-4 WT, Solar) and an HVDC link. The schematic of the model can be seen below in Fig. 18.

3.1.1 Type-3 wind turbine

The implemented RSCAD Type-3 WT model in the MIGRATE benchmark is located together with five hierarchy boxes containing the applied controls as depicted in Fig. 19. The black box depicted in Fig. 19 contains

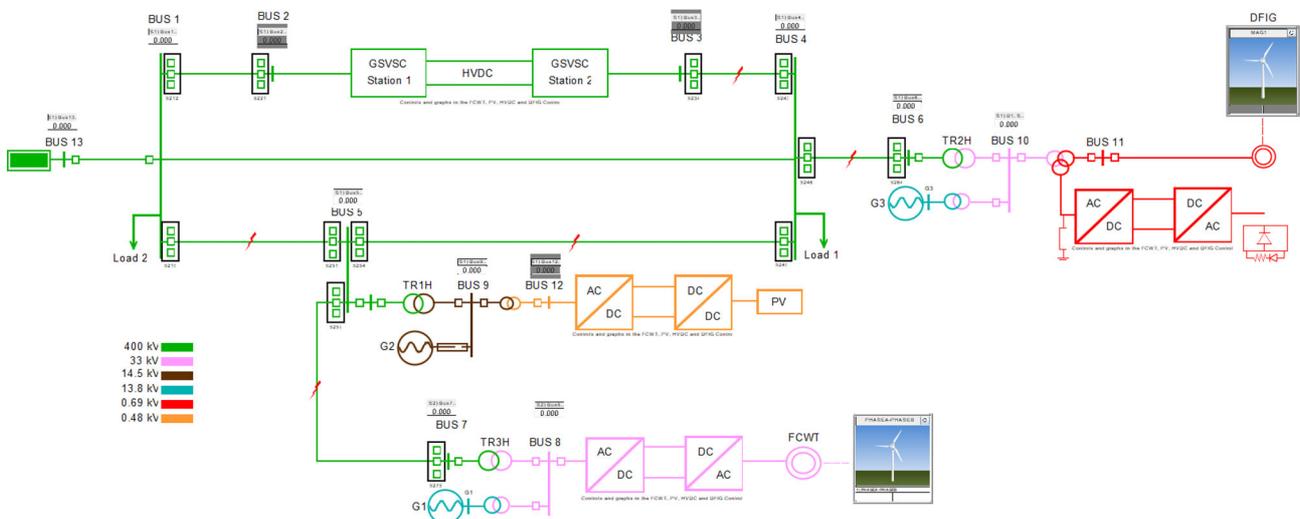


Fig. 18 Migrate Benchmark model, taken from RSCAD/RTDS draft [10]

Fig. 19 Type-3 WT Model, [10]

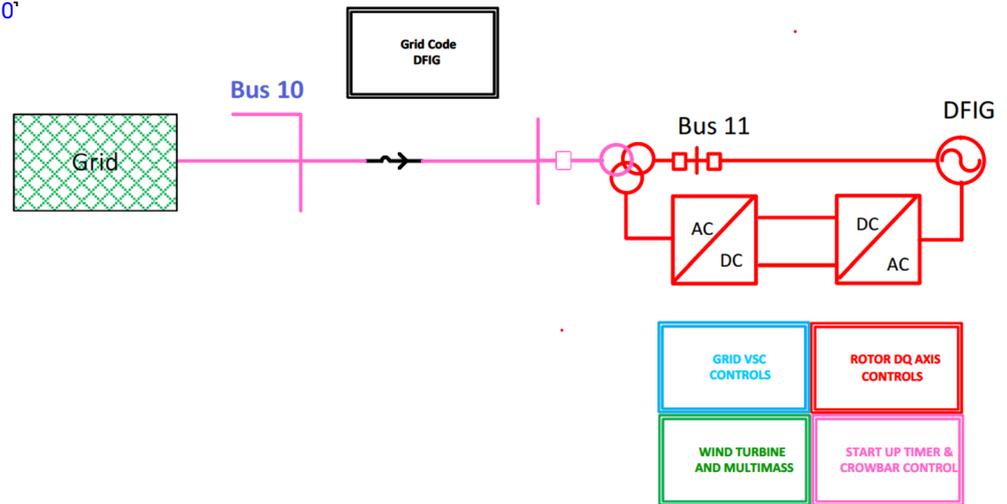


Table 2 Turbine Data

DESCRIPTION	VALUE
Rated Generator Power (GR)	2.2 MVA
Rated Turbine Power (TR)	2.0 MW
PU Gen Speed at Rated Turbine Speed (WR)	1.2 p.u
Rated wind speed (WSR)	12 m/s
Cut-in wind speed (WSC1)	6 m/s

the grid code controls. The blue and red boxes are related to the grid and rotor VSC controls. The WT with its controls can be found in the green box, and the Type-3 WT generator protection (chopper and rotor crowbar), controls, and converter blocking control are located in the purple box.

The Scherbius drive is a standard drive option in the high-power application used in the Type-3 WT. This drive is an AC–DC–AC converter in the rotor circuit where the rotor side converter (RSC) is the AC to DC converter, and the grid side converter (GSC) is the DC to AC converter. In this drive, the rotor wind-

ings are supplied through slip rings from two inverters (grid VSC and rotor VSC) connected back-to-back. Three single-phase interface transformers connect the Scherbius drive to the rest of the MIGRATE benchmark.

In the case of the Type-3 WT system, only 2 masses are modeled: the turbine itself and the DFIG. Through the configuration menu for the multi-mass, it is possible to select the high pressure and turbine mass inertia constants such that adding the two must match the value of the total inertia H requested in the mechanical constant [14]. Shaft spring constant, self, and mutual dampings are other characteristics that can be specified. At the turbine data menu in Table 2, values such as rated turbine power (MW), rated generator power (MVA), and rated wind speed (during m/s) that gives 1 pu power at 1 per unit rotational speed are given. Those parameters are considered constant during the simulation time.

The current inner regulators are the fundamental parts of the control strategies applied to the GSC and

Fig. 20 General overview of the applied Type-3 WT converter control schemes, [10]

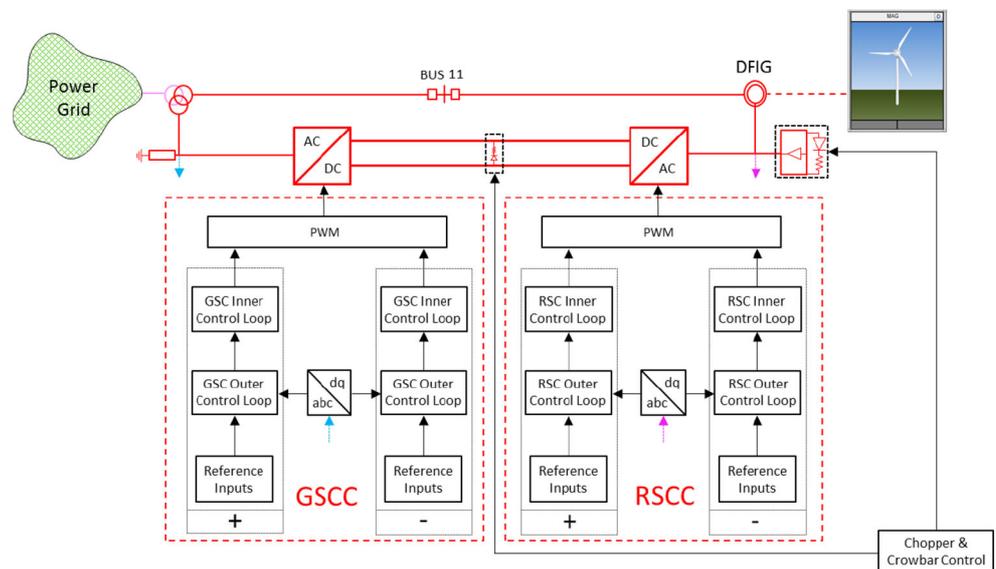


Fig. 21 RSC VSM control, [15–17]

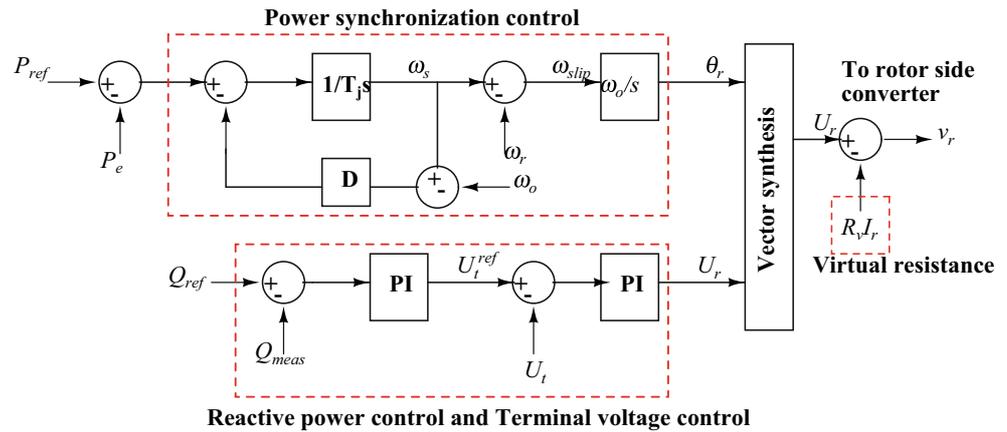


Fig. 22 RSC control that includes GFM capabilities

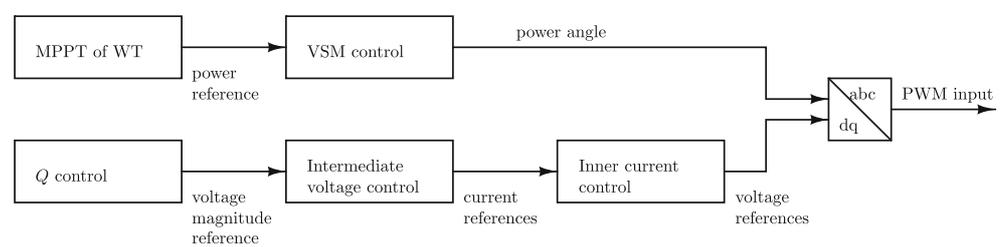


Fig. 23 MPPT logic for VSM control

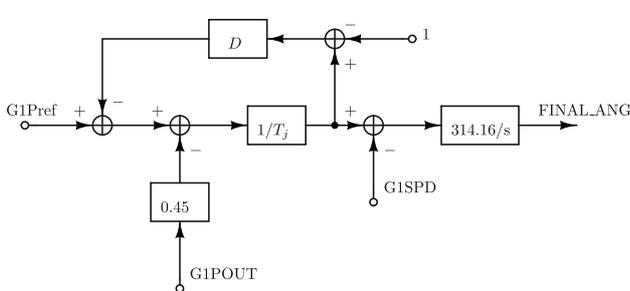
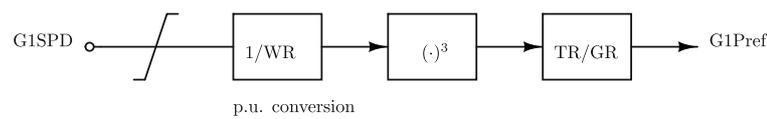


Fig. 24 VSM control strategy

RSC. The inner controllers receive their reference signals provided by the outer controllers. Each GSC and RSC contains four inner controllers. Two controllers regulate the positive sequence current components, and the other two control the negative sequence ones. The positive sequence outer controllers in GSC maintain constant DC-link voltage and provide the required reactive power for the Type-3 WT generator point at the point of common coupling (PCC) based on the applied grid code. Simultaneously, the RSC positive sequence outer controllers regulate the active and reactive power based on the optimal WT power extraction and applied grid code. Similarly to the positive sequence controllers, two regulators are considered to control the negative sequence current components. To provide the reference control signals for negative sequence regulators, double frequency minimization is considered the main target. Fig. 20

shows a general overview of the applied converter control schemes.

3.1.2 GFM control implementation

This section summarizes the application of a virtual synchronous machine (VSM) grid-forming control strategy on the existing Type-3 WT in the MIGRATE benchmark model. The basic GFM Type-3 model control mainly includes WT, RSC, and GSC control. The reference value of the active power is generated by the control of the wind turbine, which imitates the prime mover system of the traditional synchronous machine, and the independent control of rotor excitation voltage amplitude and frequency is realized through the virtual synchronous control of the rotor side. The GSC is mainly used to maintain the sta-

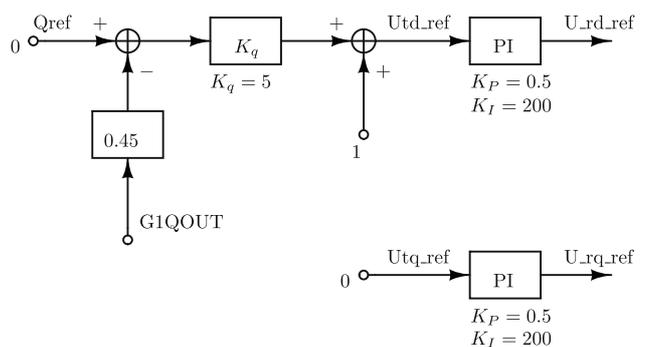


Fig. 25 Reactive power control loop

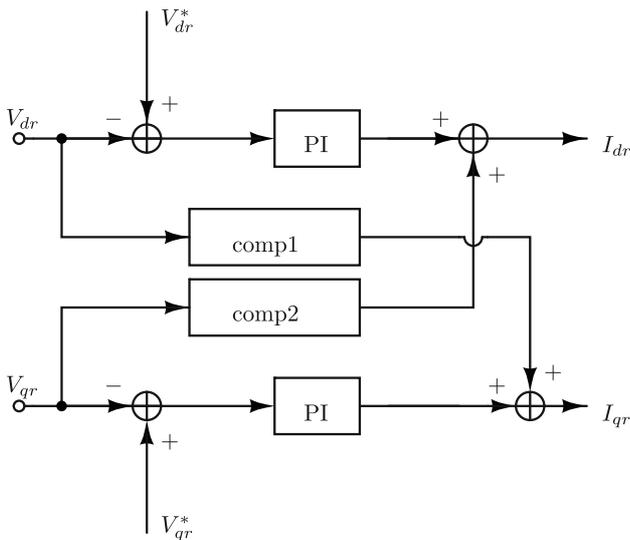


Fig. 26 Intermediate voltage controller block diagram

bility of DC voltage. Thus, the Type-3 WT under the GFM control is made available through the control of the RSC, and the vector control mode based on PLL synchronization is adopted to control GSC.

RSC control The main objective of the RSC control is to extract the optimal power from the WT. In addition to this, it can inject reactive power (according to the defined grid code) to the grid PCC to boost the voltage under fault conditions.

The RSC control of the GFM Type-3 WT model adopts the method of VSM control. The frequency/phase of rotor excitation voltage is adjusted by unbalanced active power through a virtual rotor motion equation (swing equation), to realize the adjustment of frequency/phase of internal potential. The amplitude of rotor excitation voltage is adjusted by unbalanced reactive power through the PI controller, to adjust the amplitude of internal potential. It mainly includes power synchronization control, reactive power/terminal voltage control, damping control, and virtual resistance current limiting control. The overall control diagram is shown in the Fig. 21.

The active power control loop of the VSM control strategy takes the active power reference as the input and gives the voltage angle as the output. Hence, the position of the VSM control has to be placed after the MPPT control so that the voltage angle produced can be used for park transformations of the inner controllers. Furthermore, the reactive power control gives rotor voltage magnitudes as the output, which cannot be used as references to the inner current controller

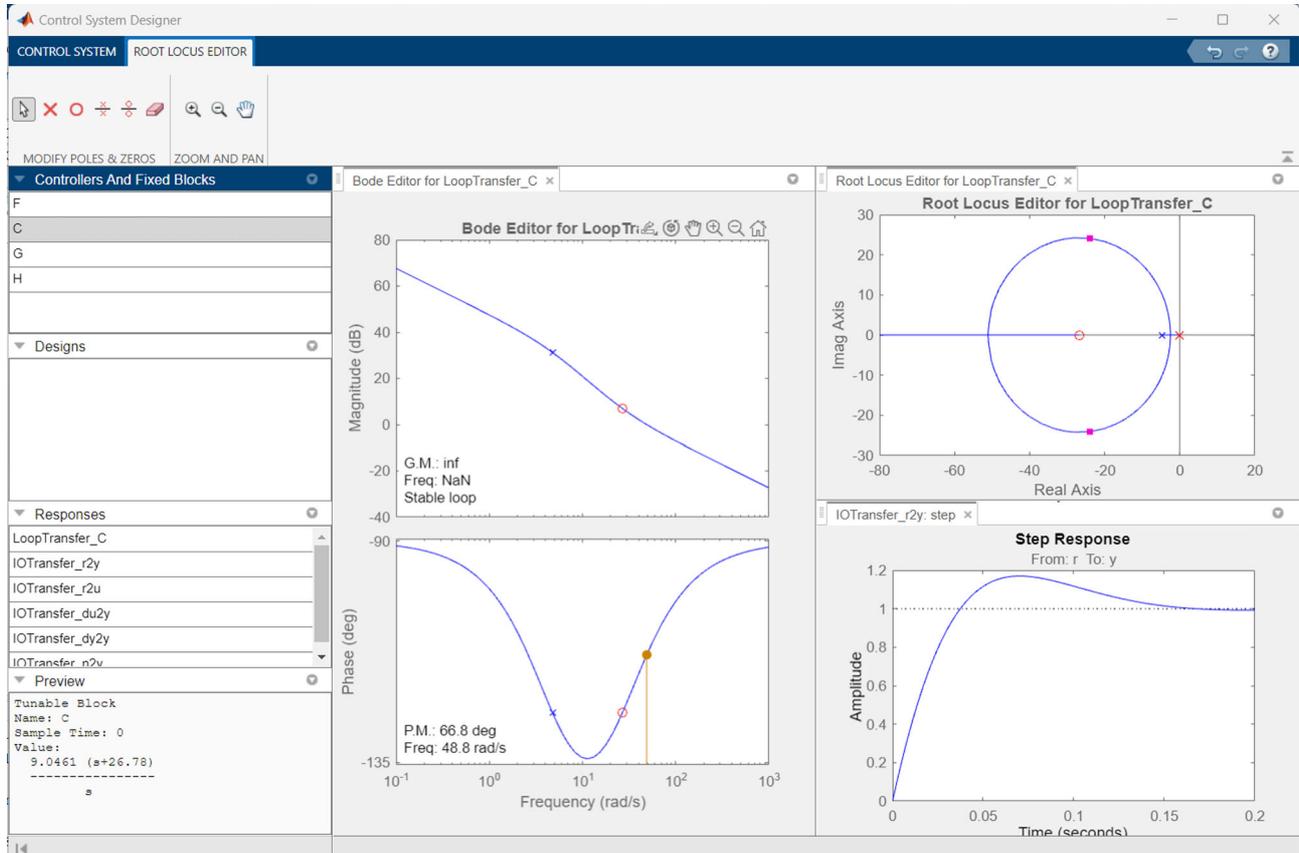


Fig. 27 PI controller tuning to design the intermediate voltage control

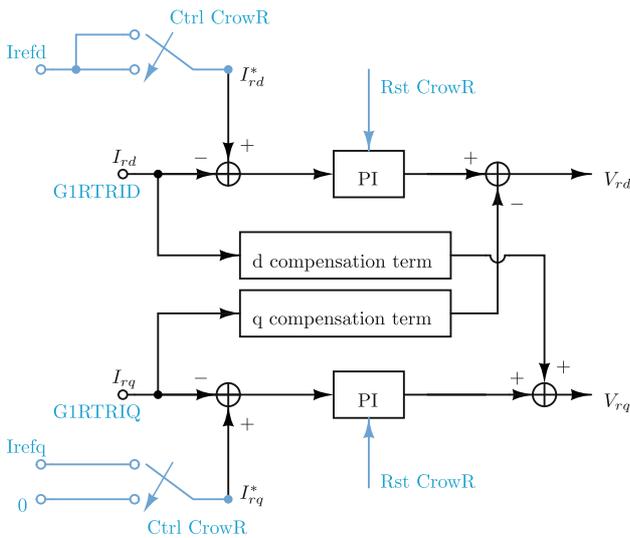


Fig. 28 RSC inner current controller scheme

which needs direct and quadrature axis current reference. Hence, based on the characteristic equations of the induction generator, an intermediate voltage controller has to be inserted after the reactive power control of the VSM control and before the inner current controller so that current references for the inner current controller can be produced. The new proposed control strategy to enable GFM capabilities can be summarized as in Fig. 22.

MPPT Algorithm The new VSM control strategy takes in the reference power as an input (see Fig. 21) and provides the power reference as its output. The optimal power to be extracted from a wind turbine is a function of turbine shaft speed raised to the third power as:

$$P_{opt} = K_{opt} \omega_r^3, \quad (2)$$

considering the torque expression ($T = \frac{P}{\omega_r}$), the optimal electrical torque is a function of turbine shaft speed raised to the second power. Hence, after applying the above logic, the new MPPT control logic looks as in Fig. 23.

VSM Control and Q Control The VSM control presented in [17] was implemented for active and reactive power control loops, as shown in Figs. 24 and 25. The damping constant $D = 150$ as it provides the lowest peak response, i.e., the highest damping effect crucial for the DFIG stability [17]. The typical inertia constants for SGs of the large conventional power plants are in the range of 2–9 s, and the physical inertia constant of the modern large WT is more or less equal to the average of the conventional power plants. In our analysis, the inertia constant $T_j = 10$ mimics SG’s inertial response behavior with $H = 5$ s. The reference reactive power is taken as zero.

In Figs. 24 and 25, the gains in front of the inputs are factors to convert in per unit values. It has to be noted carefully that the signal **FINAL_ANG** represents the generated slip voltage angle that a PLL originally generated. This angle is used for Park transformations and inverse Park’s transformation of Input and output rotor voltages and currents for inner controllers. When it is to be used for Park transformations of stator quantities, it must first be added to the rotor angle to generate the full voltage angle. The signals **U_rd_ref** and **U_rq_ref** will be referenced in the intermediate voltage controller.

Intermediate Voltage Controller The intermediate voltage controller was designed to take the reference voltage signals generated by the reactive power con-

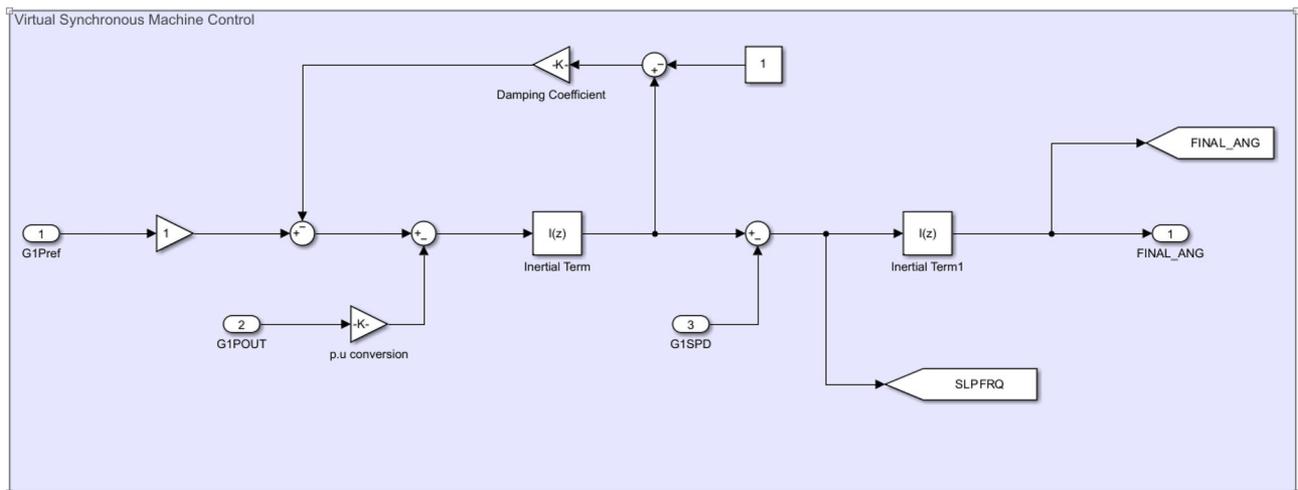


Fig. 29 VSM control in MATLAB/Simulink

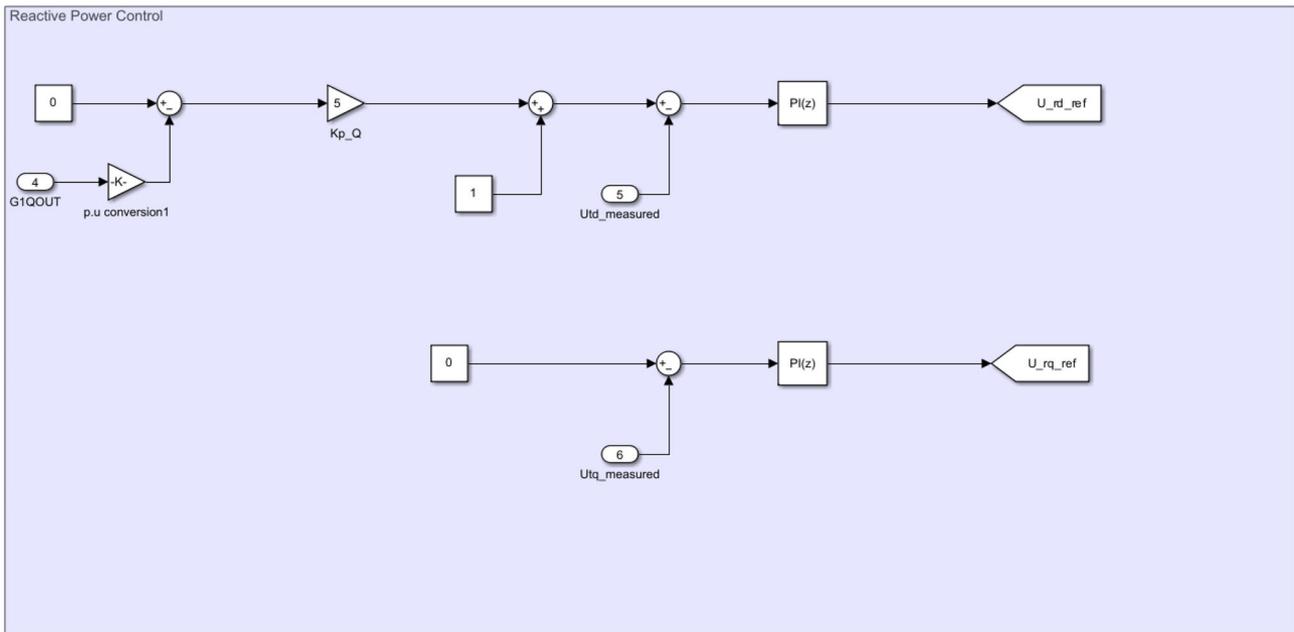


Fig. 30 Q control in MATLAB/Simulink

trol and generate current references for the inner current control. The DFIG rotor voltage equations can be written in the Laplace domain as:

$$v_{dr}(s) = R_r i_{dr}(s) + \sigma L_r s i_{dr}(s) - \omega_{\text{slip}} \sigma L_r i_{qr}(s), \quad (3)$$

$$v_{qr}(s) = R_r i_{qr}(s) + \sigma L_r s i_{qr}(s) + \omega_{\text{slip}} (L_m s i_{ms}(s) + \sigma L_r i_{dr}(s)), \quad (4)$$

using the principle of superposition, we can derive the transfer function and corresponding compensation terms as:

$$\frac{i_{dr}(s)}{v_{dr}(s)_{i_{qr}=0}} = \frac{i_{qr}(s)}{v_{qr}(s)_{i_{dr}, i_{ms}=0}} = \frac{1}{R_r + s \sigma L_r}, \quad (5)$$

$$\frac{i_{qr}(s)}{v_{dr}(s)_{i_{dr}=0}} = \frac{-1}{\omega_{\text{slip}} \sigma L_r}, \quad (6)$$

$$\frac{i_{dr}(s)}{v_{qr}(s)_{i_{qr}=0}} = \frac{1}{\omega_{\text{slip}} \sigma L_r} + \omega_{\text{slip}} L_m i_{ms}. \quad (7)$$

The above Eqs. (3), (4) can be represented in the control block form in the Fig. 26 below. The compensation terms are represented by the right-hand side of Eqs. (6) and (7) and are denoted in the figure by names 'comp1' and 'comp2', respectively.

In Fig. 26, the dq voltage references denoted by V_{dr}^* and V_{qr}^* are output from the preceding reactive power controller of the VSM control, as seen from Fig. 22. The PI controllers for the intermediate controller are tuned using the Zeigler-Nichols step response method. The proportional and integral gain values obtained from the Zeigler-Nichols step response tuning method serve as a starting point, and the values are further optimized through multiple iterations and observing the controller behavior. The initially tuned

PI controller plots on the SISO-TOOL of MATLAB can be seen in Fig. 27.

Inner Current Control Simple PI controllers were used for the inner current controller as illustrated in Fig. 28 in black. The derivation of compensation terms is done based on the generator's characteristic equations based on the same approach as in the previous paragraph describing the intermediate voltage controller.

A crowbar protection scheme is also implemented here, which controls the reference current inputs and reset value for the integrator. The input signal **CrowR** in Fig. 28 comes from the crowbar protection control, which controls the reference current inputs and reset value for the integrator. The crowbar control system is a conventional approach to protect the RSC from DFIG rotor over-current. This system is connected to the rotor windings, and during a fault, the rotor currents increase. When the currents exceed the surge capability of each IGBT, the crowbar system is activated and diverts the currents through the resistor to dissipate the initial energy outflow from the machine.

3.1.3 GSC control

The GSC control of the GFM Type-3 WT model adopts traditional vector control, and its main function is to maintain DC bus voltage stability. In the case of terminal voltage orientation, the active and reactive power control is decoupled, and the d -axis current controls the active power, and the q -axis current controls the reactive power. To reduce the capacity of the grid-connected converter as much as possible, the refer-

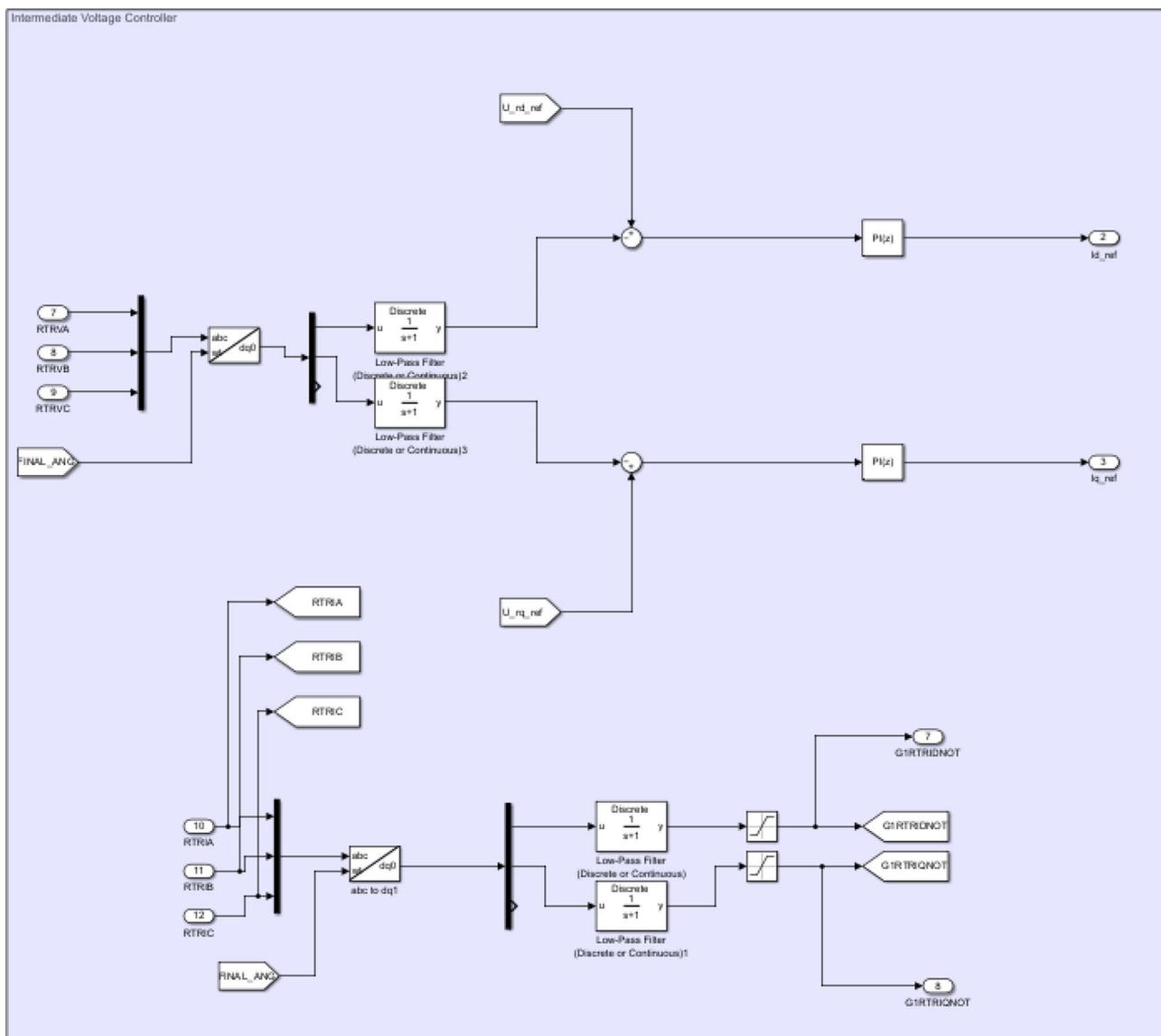


Fig. 31 Intermediate voltage control in MATLAB/Simulink

ence value of the q -axis component of the current is often set to 0. The control of DC bus voltage is mainly to maintain the stability of DC voltage by adjusting the reference value of the d -axis component of the current.

4 DLL test for validation

4.1 Black box control implementation

The GFM controller implemented in MATLAB/Simulink has the blocks depicted in Fig. 22: VSM control, Q control, intermediate voltage control, and inner current control. The realization of these controls in MATLAB/Simulink with denoted input and output signals is already depicted in Fig. 3. Inside the mask, four controllers are mentioned as shown in Figs. 29–32.

For building DLL control, the procedure from Sect. 2 is followed.

4.2 Type-3 wind turbine black box control validation

To validate the developed black box model, two tests are performed with the black-box controller connected to the RSCAD model: 1) wind speed step up from 9 to 12 m/s at low SCR value of 9 (SGs off, HVDC link disconnected); and 2) load step from 70 to 370 MW up at $SCR = 20$. These tests are performed for the cases when the GFM controller is implemented as an internal RSCAD controller as described in Sect. 3, and when it is implemented as a black box in MATLAB/Simulink as explained in Sect. 4.1, and thus, interconnected in SIL manner as described in Sect. 2. In both cases, the simulated power system is the same as in the MIGRATE project [10].

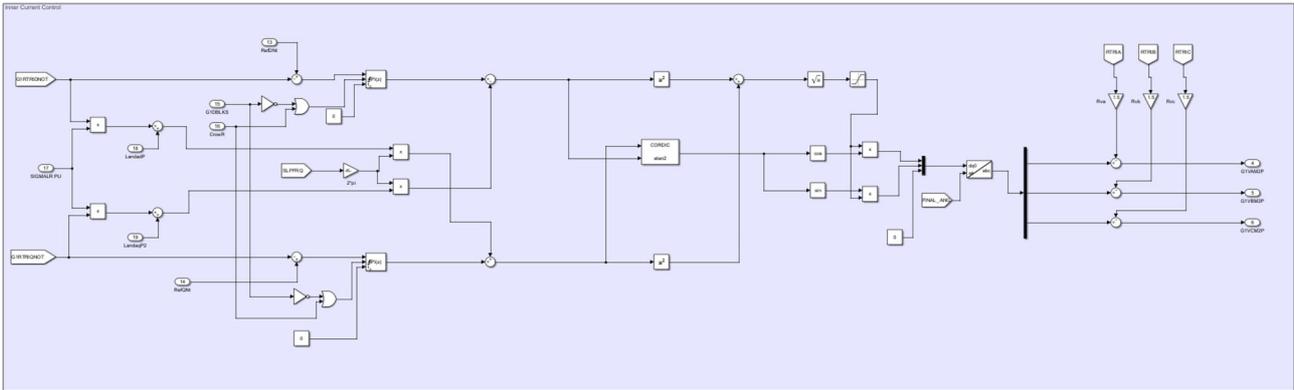
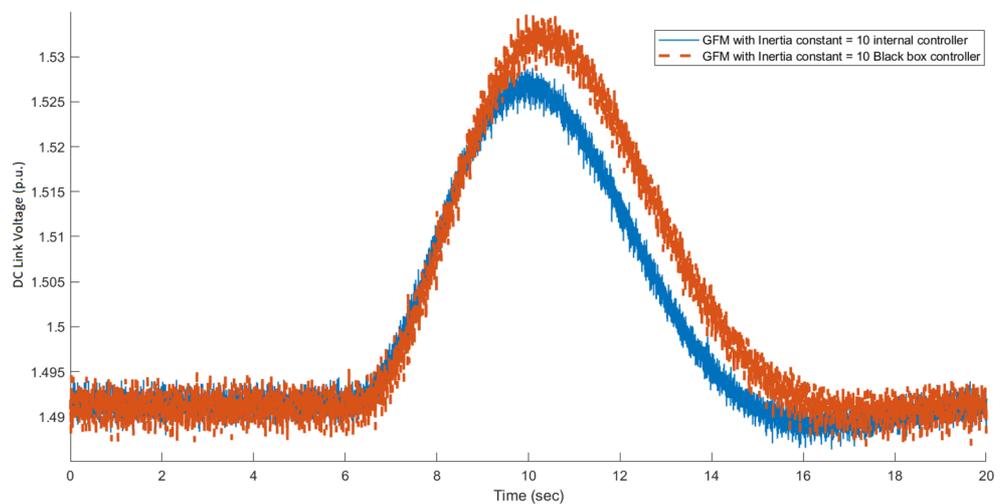


Fig. 32 Current control in MATLAB/Simulink

Fig. 33 DC-link voltage variation for $SCR = 9$ and wind speed variation from 9 m/s to 12 m/s



The results are plotted against the previous results obtained for the same tests when the controller was implemented within the model. Figs. 33 and 34 show the DC-link voltage and active power variation when wind speed increases from 9 m/s to 12 m/s.

We see that the performance of the developed black-box controller follows the performance of the original controller. The error of the active power variation is close to 6 %, and that of DC-link voltage is at 0.32 %.

Figure 35 shows a plot to show instant active power support in response to a load step from 70 to 370 MW up at $SCR = 20$.

Here, we can see almost an absolute overlap of performance.

The differences shown in Figs. 33–35 can be many [3]. Namely, the original model is developed in RSCAD/RTDS and the black-box model is developed in MATLAB/Simulink. Although the controller structure is the same, some modeling aspects in the RTS models cannot be easily checked, such as reactance cross-coupling when simulating with multiple parallel CPUs (cores). We plan to check this in our future work.

5 Conclusion

This paper presented a comprehensive methodology for developing a black-box model in DLL for a converter algorithm initially developed using MATLAB/SIMULINK. Third-party software was used to run the under-test DLL file in real time on a Windows computer. In the end, the designed controller is tested in a control hardware-in-the-loop setup for the test case of a Type-3 grid-forming wind turbine. A limitation of the presented method is that it depends on third-party software whose license needs to be purchased. The method thus depends on the development and availability of the used third-party software. Currently, the communication between the black-box controller model and the power system model is happening through the User Data Protocol (UDP), which is a relatively slower protocol. Studies can be conducted to see whether this can be done through other high-level communication protocols. Different real-time simulator manufacturers are also coming up with customized solutions to enable the running of controller source code in real-time. However, this also comes at a cost and sometimes is not suitable to the needs of the user. The presented method provides

Fig. 34 Active power variation for $SCR = 9$ and wind speed variation from 9 m/s to 12 m/s

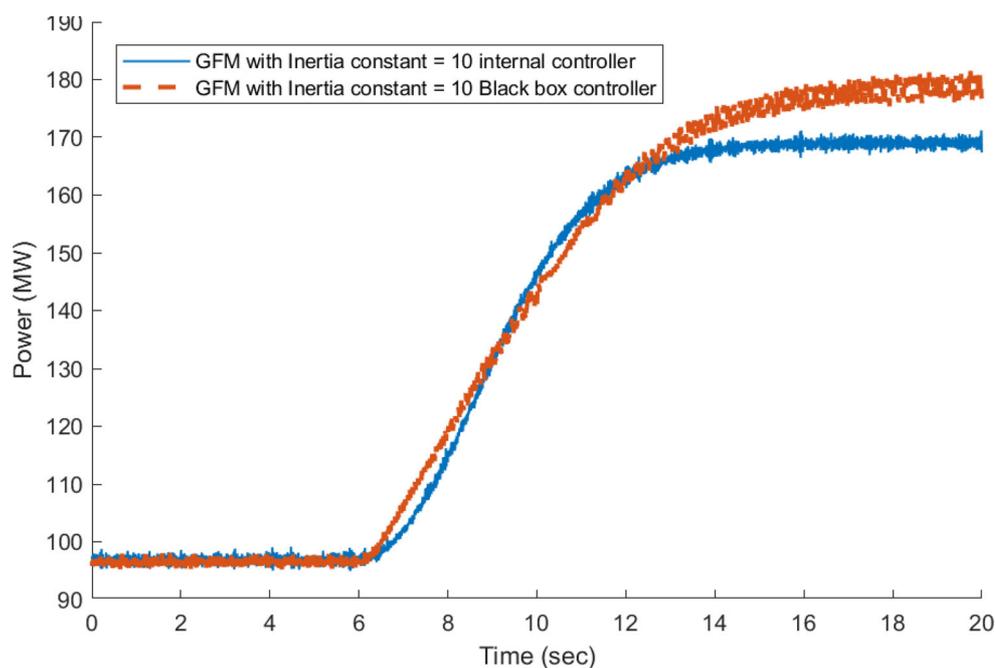
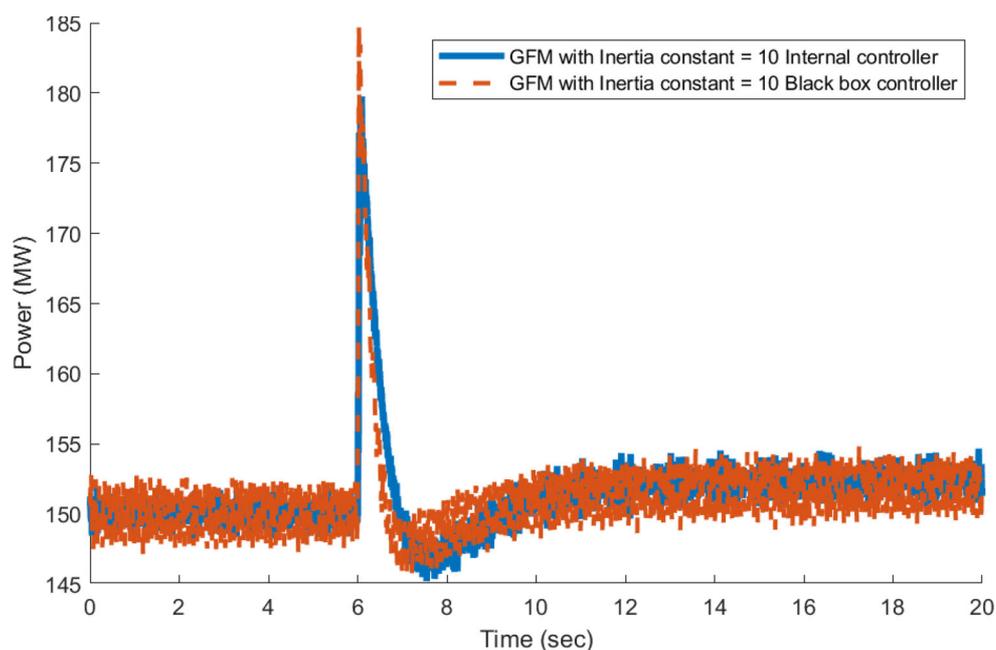


Fig. 35 Power output in case of high $SCR = 20$, load step change



a new real-time simulator platform independent way to test actual converter control software in DLL form on a hardware-in-the-loop test bed. The validated approach is in the form of a black box, using the test case of the MIGRATE project implemented in RSCAD/RTDS. However, the development of the black box controller model is the same one applied in different real-time hardware-in-the-loop platforms, which makes this approach easy to extend to different platforms and applications.

Open Access Dieser Artikel wird unter der Creative Commons Namensnennung 4.0 International Lizenz veröffentlicht, welche die Nutzung, Vervielfältigung, Bearbeitung, Verbre-

itung und Wiedergabe in jeglichem Medium und Format erlaubt, sofern Sie den/die ursprünglichen Autor(en) und die Quelle ordnungsgemäß nennen, einen Link zur Creative Commons Lizenz beifügen und angeben, ob Änderungen vorgenommen wurden. Die in diesem Artikel enthaltenen Bilder und sonstiges Drittmaterial unterliegen ebenfalls der genannten Creative Commons Lizenz, sofern sich aus der Abbildungslegende nichts anderes ergibt. Sofern das betreffende Material nicht unter der genannten Creative Commons Lizenz steht und die betreffende Handlung nicht nach gesetzlichen Vorschriften erlaubt ist, ist für die oben aufgeführten Weiterverwendungen des Materials die Einwilligung des jeweiligen Rechteinhabers einzuholen. Weitere Details zur Lizenz entnehmen Sie bitte der Lizenzinformation auf <http://creativecommons.org/licenses/by/4.0/deed.de>.

References

- De Carne G, Lauss G, Syed MH, Monti A, Benigni A, Karari S, Kotsampopoulos P, Faruque MO (2022) On modeling depths of power electronic circuits for real-time simulation—a comparative analysis for power systems. *Ieee Open Access J Power Energy* 9:76–87
- Benigni A, Strasser T, De Carne G, Liserre M, Cupelli M, Monti A (2020) Real-time simulation-based testing of modern energy systems: A review and discussion. *Ieee Ind Electron Mag* 14(2):28–39
- (2024) BiGER Explore project. <https://cresym.eu/biger-explore/>
- Shetgaonkar A, Lekić A, Rueda Torres JL, Palensky P (2021) Microsecond enhanced indirect model predictive control for dynamic power management in mmc units. *Energies* 14(11):3318
- Liu L, Shetgaonkar A, Lekić A (2023) Interoperability of classical and advanced controllers in mmc based mtdc power system. *Int J Electr Power Energy Syst* 148:108980
- (2021) IntervalZero products. <https://www.intervalzero.com/en-products/en-rtx64/>
- (2023) RSCAD, M.: Training & Support. <https://www.rtds.com/resource-centre/training-support/>
- (2019) IntervalZero, d.: RTX64 supported network interface cards – intervalzero. https://www.intervalzero.com/library/guides_and_mini_tutorials/rtx64_documentation/RTX64SupportedNIC4.1.pdf
- Wang S, Hu J, Yuan X, Sun L (2015) On inertial dynamics of virtual-synchronous-controlled dfig-based wind turbines. *Ieee Trans Energy Convers* 30(4):1691–1702. <https://doi.org/10.1109/tec.2015.2460262>
- Popov M, Zagar M, Chavez J, Martinez E, Borroy S, Terzija V, Azizi S, Sun M (2016) Migrate—massive integration of power electronic devices, d4.1: Accurate models for desktop protection studies and hardware-in-the-loop (hil) tests. Technical report, Red Eléctrica de España
- Kikusato H, Orihara D, Hashimoto J, Takamatsu T, Oozeki T, Matsuura T, Miyazaki S, Hamada H, Miyazaki T (2023) Performance evaluation of grid-following and grid-forming inverters on frequency stability in low-inertia power systems by power hardware-in-the-loop testing. *Energy Reports*, vol 9, pp 381–392 <https://doi.org/10.1016/j.egy.2022.10.434>
- Du W, Chen Z, Schneider KP, Lasseter RH, Pushpak Nandanoori S, Tuffner FK, Kundu S (2020) A comparative study of two widely used grid-forming droop controls on microgrid small-signal stability. *IIEEE J Emerg Sel Topics Power Electron* 8(2):963–975. <https://doi.org/10.1109/jestpe.2019.2942491>
- Chavez JJ, Popov M, López D, Azizi S, Terzija V (2021) S-transform based fault detection algorithm for enhancing distance protection performance. *Int J Electr Power Energy Syst* 130:106966
- Popov M, Chavez J, Carrasco EM, Martinez MT, Vicente SB, Lopez D, Azizi S, Terzija V (2020) Enhancing distance protection performance in transmission systems with renewable energy utilization. *ISGT-Europe*, vol 2020. *IIEEE, PES Innovative Smart Grid Technologies Europe* <https://doi.org/10.1109/isgt-europe47291.2020.9248896>
- Hu J, Lei Y, Chi Y, Tian X (2022) Analysis on the inertia and the damping characteristics of dfig under multiple working conditions based on the grid-forming control. *Reports*, vol 8. *Energy*, pp 591–604 <https://doi.org/10.1016/j.egy.2022.09.200>
- Hu J, Chi Y, Tian X, Zhou Y, He W (2022) A coordinated and steadily fault ride through strategy under short-circuit fault of the wind power grid connected system based on the grid-forming control. *Reports*, vol 8. *Energy*, pp 333–341 <https://doi.org/10.1016/j.egy.2022.01.166>
- Wang S, Hu J, Yuan X (2015) Dfig-based wind turbines with virtual synchronous control: Inertia support in weak grid. *Meeting*, vol 2015. *IIEEE, Power and Energy Society General Meeting* <https://doi.org/10.1109/pesgm.2015.7286451>

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Utkarsh Singh, works as converter controls software engineer in HVDC control & protection team at Hitachi Energy in Ludvika, Sweden. He finished his master's degree in sustainable energy technology from TU Delft. He has done his MSc thesis in collaboration with DNV in Netherlands.



Ravi Singh, is an experienced electrical engineer with knowledge in the domain of modelling, measurements and testing of power grid components. He has several years of experience in the field of HV substation engineering and design. He obtained his PhD from TU Eindhoven in Netherlands in 2021. Since 2021 he is working with DNV Group R&D in Arnhem, Netherlands. The focus of his work is on assurance of assets in power grids using cyber-physical modeling and simulations. He is particularly involved in using hardware-in-loop and software-in-loop testbeds to test and assess controllers, control algorithms for various assets in power grids including Wind, PV, BESS, STATCOM, etc.



Marjan Popov, is a professor at the Delft University of Technology. His research interests are in future power systems, large-scale power system transients, intelligent protection for future power systems, and wide-area monitoring and protection. He received the Hidde Nijland Prize for extraordinary research and education achievements in smart grids in 2010. He received the IEEE PES Prize Paper Award and IEEE Switchgear Committee Award in 2011. He is an Associate

Editor of Elsevier's International Journal of Electric Power and Energy Systems, and Co-Editor in Chief for the Elsevier's e-Prime journal, Advances for e-Prime journal, Advances in Electrical Engineering, Electronics and Energy. He is a member of Cigre, a steering committee member of the Dutch Cigre, and an IEEE Fellow.



Aleksandra Lekić, received B.S., M.S., and Ph.D. degrees in electrical engineering from the School of Electrical Engineering, University of Belgrade, Belgrade, Serbia, in 2012, 2013, and 2017, respectively. Since January 2020, Aleksandra works as an Assistant Professor at TU Delft, Faculty of Electrical Engineering, Mathematics and Computer Science. Aleksandra leads the Control of the HVDC/AC power systems team, which conducts advanced research in the field of

power electronics and power system control. She is a recipient of the prestigious NWO Veni 2022 grant in the Netherlands. She is an Associate Editor in the International Journal of Electrical Power & Energy Systems, Elsevier.