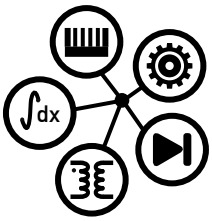

RT Box

DEMO MODEL

Multistep Model Predictive Control for NPC Inverter Driving an Induction Machine

Using the RT Box as a powerful rapid control prototyping unit

Last updated in RT Box Target Support Package 2.2.1

www.plexim.com

- Request a PLECS and PLECS Coder trial license
- Get the latest RT Box Target Support Package
- Check the PLECS and RT Box documentation

1 Overview

This RT Box model features a medium voltage three-level neutral-point-clamped voltage source inverter driving an induction machine. The squirrel cage induction machine has a rated power of 2 MVA. The objective of the model predictive current controller is to control the stator currents along their time-varying reference, by manipulating the switch position, while minimizing the switching effort. The predictive controller uses a prediction horizon of $N_\mathrm{p} = 5$. An efficient optimization algorithm is used to solve the underlying optimization problem.

The discretization step size and average execution times for this model are shown in Fig. 1. Note that the Controller execution time refers to the most loaded CPU core running the MPC algorithm on an RT Box 2 or 3.

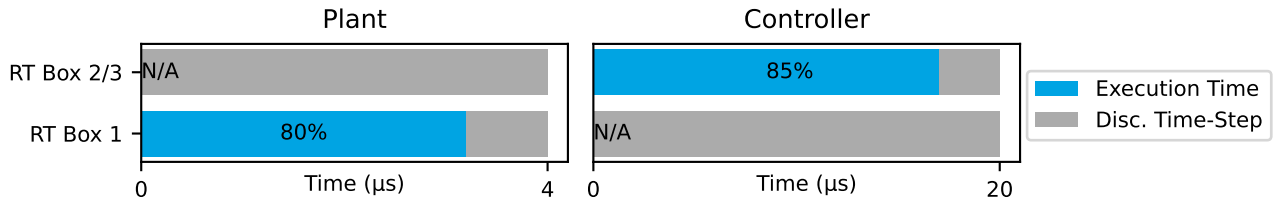

Figure 1: Performance overview for the execution on two RT Boxes

1.1 Requirements

To run this demo model, the following items are needed (available at `www.plexim.com`):

- One PLECS RT Box 2 or 3, one PLECS RT Box 1 and one PLECS and PLECS Coder license
- The RT Box Target Support Library
- Follow the step-by-step instructions on configuring PLECS and the RT Box in the Quick Start guide of the RT Box User Manual.
- Three 37 pin Sub-D cables to connect the boxes front-to-front.
- One SFP+ cable

Note that this demo model is targeted at two RT Boxes application, with an RT Box 1 running the Plant and an RT Box 2 or 3 running the Controller. In this way, the execution time of each real-time target is minimized. Besides, the setup can easily transition to a HIL or RCP test later on.

However, if the user has only one RT Box 2/3 available, please check the corresponding models targeted for one RT Box application. In this case, two 37 pin Sub-D cables are still needed to connect in front Analog Out interface with Analog In interface, and Digital Out interface with Digital In interface.

Note This model contains model initialization commands that are accessible from:

PLECS Standalone: The menu **Simulation + Simulation Parameters... + Initializations**

PLECS Blockset: Right click in the **Simulink model window + Model Properties + Callbacks + InitFcn***

---

2 Model

The top-level schematic contains two subsystems representing the controller and the plant models, as shown in Fig. 2. Both subsystems are enabled for code generation from the **Subsystem + Execution settings...** context menu. This step is necessary to generate the model code for a subsystem via the PLECS Coder.

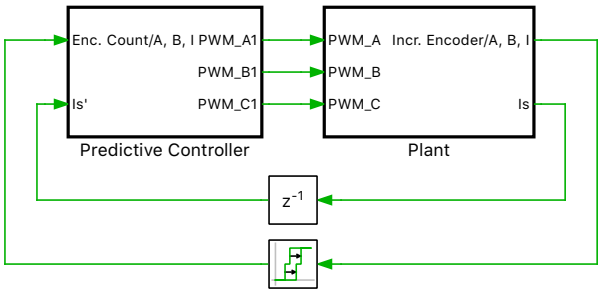

Figure 2: Top-level schematic of the demo model

2.1 Plant

The schematic as seen in Fig. 3 shows a medium voltage (MV), three-level, three-phase, neutral-point-clamped (NPC) voltage source inverter driving a squirrel cage induction machine. The neutral point potential is fixed at zero and the dc-link voltage is assumed to be constant.

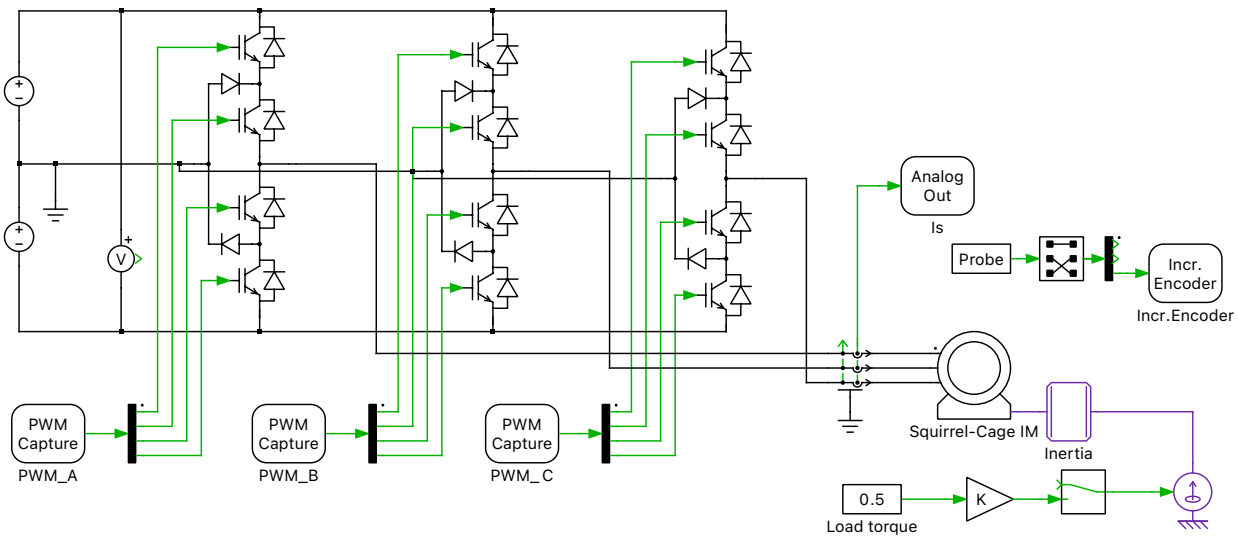

Figure 3: Power circuit of the drive system

The three-level NPC is represented by three 3-Level Half Bridge (NPC) power modules. The pulse-width modulated (PWM) switching signals are obtained from the PWM Capture block of the PLECS RT Box library. Further details on the power module components and the sub-cycle averaging of PWM signals are described in [1]. The AC output current measurements are connected to Analog Out block from the PLECS RT Box library. The rotor angular position and rotational speed are converted by the Incremental Encoder block from the PLECS RT Box library, into digital orthogonal pulses, which can be measured outside of the subsystem.

Machine Parameters

The parameters of this model are borrowed from the case-study defined in [2]. The machine is a 3.3 kV, 50 Hz squirrel-cage induction machine rated at 2 MVA. The rated values of the machine and the drive parameters are provided in the tables below.

Rated values of the induction machine

Parameter	SI value
Voltage	3300 V
Current	356 A
Real power	1.587 MW
Apparent power	2.035 MVA
Angular stator frequency	$2\pi 50$ rad/s
Rotational speed	596 rpm
Air-gap torque	26.2 kNm

Drive parameters

Parameter	SI value	pu value
Stator resistance	57.61 mΩ	0.0108
Rotor resistance	48.89 mΩ	0.0091
Stator leakage inductance	2.544 mH	0.1493
Rotor leakage inductance	1.881 mH	0.1104
Main inductance	40.01 mH	2.349
Number of pole pairs	5	
dc-link voltage	5.2 kV	1.930

2.2 Controller

The controller schematic is shown in Fig. 4. The measurements of the AC currents are imported by Analog In block. The mechanical angular speed of the rotor is obtained from the Quadrature Encoder Counter block, which converts the orthogonal digital pulses.

This model uses model predictive control (MPC) scheme for the NPC inverter. The outer flux and speed control loops provide an input to the MPC controller.

The MPC controller predicts the stator currents at the next time step for all possible switch positions. For the prediction, the sampled stator currents are required, along with the rotor flux vector. The rotor flux vector cannot be measured directly. Therefore, this vector has to be estimated based on known quantities, i.e. the stator currents, stator voltages and rotor speed.

Control Problem

The outer flux and speed controllers provide the reference d- and q-component of the stator currents to the predictive current controller. The flux controller ensures that the machine is appropriately fluxed, while the speed controller ensures that the machine is close to the desired speed reference.

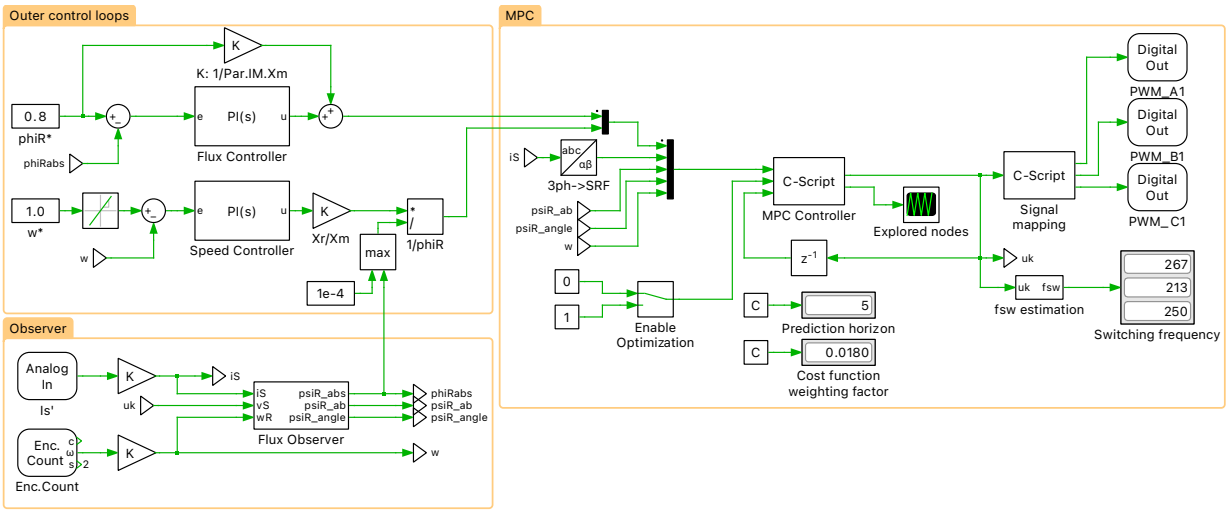

Figure 4: Predictive controller model of the drive system

The predictive current controller can be split in two main parts:

- the prediction of the future current trajectories based on a dynamic internal model of the drive system
- the optimization problem to find the unique switching vector that minimizes the cost function solved with the modified sphere decoding algorithm.

Speed Controller The speed value measured by the incremental encoder and the speed reference is provided as an input to the speed controller. The speed controller then regulates the electromagnetic torque setpoint, based on which the q-component of the stator current reference is calculated and provided to the predictive current controller.

Flux Controller The flux controller ensures that the induction machine is appropriately fluxed and can achieve fast control of the flux magnitude. The error between the estimated rotor flux and its reference is provided as an input to the flux controller. The flux controller then manipulates the d-component of the stator currents.

Rotor Flux Estimation

Based on the measured stator currents, rotor speed and stator voltages, the rotor flux vector can be estimated. The stator voltage values can be reconstructed using the upper and lower dc-link voltages and the currently applied switch positions. In this demo the rotor flux estimation is just an implementation of the machine equations. Since an estimator has no corrective feedback, this rotor flux estimation scheme will not account for parameter variations, and parameter uncertainties.

Direct Model Predictive Current Control

MPC incorporates a dynamic model of the system to be controlled. This model is needed to predict a sequence of future system states and outputs based on the actual state and a sequence of manipulated variables. Based on a cost function the optimum future input sequence in order to drive the system towards a reference state is searched. The control objective is translated into a cost function, which maps the sequence of future states, outputs, and inputs into a scalar value. This scalar value is used to compare different possible future input values together and to chose the optimum one, i.e the solution with the lowest cost.

A commonly used MPC approach in power electronics is to directly manipulate the switch positions of the semiconductors in order to track a reference. In this demo the direct MPC formalism regulates the

stator currents towards the reference values provided by the outer speed and flux controllers. In direct MPC the current controller and the modulator are grouped into one computational stage. Therefore, no modulator is needed in direct MPC.

The major disadvantage of direct MPC is the computational burden of solving the optimization problem especially for longer prediction horizons, since the number of possible switching sequences grows exponentially.

Tuning Variables Contrary to classical control schemes, in direct MPC there are only two parameters that can be tuned:

- the sampling interval T_s and
- the switching penalty λ_u

The switching penalty is always a trade-off between tracking accuracy and switching effort for a given sampling interval. This weighting coefficient is used directly in the cost function as shown below. A lower bound on the sampling interval is mainly given by the computational power of the controller, and secondly the minimum on-time of the semiconductors.

Cost Function and Optimization Problem

The future values of the load currents are predicted for all the switching states generated by the inverter. For this purpose, it is necessary to measure the present load currents. After obtaining the predictions, a cost function g is evaluated for each switching state. The switching state that minimizes the cost function is selected and applied during the next sampling period.

The control requirements for the NPC inverter are:

- Load current reference tracking
- Reduction of the switching frequency to reduce switching losses.

These requirements can be formulated in the form of a cost function to be minimized. The cost function for the NPC inverter with a prediction horizon of 1 has the following composition:

$$g = |\mathrm{i}_\alpha^* - \mathrm{i}_\alpha^\mathrm{p}| + |\mathrm{i}_\beta^* - \mathrm{i}_\beta^\mathrm{p}| + \lambda_\mathrm{n}\mathrm{n}_\mathrm{c} \tag{1}$$

where, the first two terms are the load current errors in orthogonal coordinates, where $\mathrm{i}_\alpha^\mathrm{p}$ and $\mathrm{i}_\beta^\mathrm{p}$ are the real and imaginary components of the predicted current vector i^p, respectively, and i_α^* and i_β^* are the real and imaginary components of the reference current vector i^*.

The last term in Eq. (1) is the number of commutations n_c required to switch from the present switching state to the switching state under evaluation. A switching state that implies fewer commutations of the power semiconductors will be preferred. In this manner, the use of this term will have a direct effect on the switching frequency of the converter. Weighting factor λ_n handles the reduction of switching frequency within the cost function g. A larger weighting factor value implies greater priority to that objective.

If the prediction horizon is increased, the cost function must be calculated for each step and finally summed up.

Computational Burden The above optimization problem can be solved based on exhaustive search. In doing so, the cost function is evaluated for each possible switching combination. As the prediction horizon is enlarged and the number of decision variables is increased, the computational complexity grows exponentially. Therefore, exhaustive search is computationally feasible only for very short predictions horizons. However, a far more efficient method based on branch-and-bound for finding the optimal switching sequence can be used. The second method is an adapted version of a so-called sphere decoding algorithm.

Modified Sphere Decoding Algorithm The modified sphere decoding algorithm is based on the branch-and-bound approach. It is dentified to be a very efficient algorithm to find the optimal switching sequence U_{opt} for direct model predictive control optimization problems with long prediction horizons, by only exploring a very small subset of the search tree. Although only a small part of the possible nodes is calculated, the algorithm still returns the optimal solution in each sampling period. For completeness, the pseudo-code of the modified sphere decoding algorithm introduced in [2] is given in Fig. 5.

```
function U_opt = mSDA(U, d^2, i, ρ^2, Ū_unc):
    for each u ∈ U do
        u_i = u;
        d'^2 = ||ū_unc,i − v_i,1:i u_1:i|||| + d^2;
        if d'^2 ≤ ρ^2 then
            if i < 3N_p then
                mSDA(U, d'^2, i + 1, ρ^2, Ū_unc);
                else
                    if U meets then
                        U_opt = U;
                        ρ^2 = d'^2;
                    end
                end
            end
        end
    end
```

Figure 5: Modified sphere decoding algorithm

To invoke the search algorithm, it must be initialized. For this, first the unconstrained optimal solution $\bar{U}_{\mathrm{unc}}$ can be calculated explicitly, as shown in [2]. This unconstrained solution neglects the fact that a valid switching vector can only contain discrete values as given in U_{set}[1]. Direct componentwise rounding of the unconstrained solution towards the nearest integer value given in U_{set}

$$U_{\mathrm{ini}} = round(U_{\mathrm{unc}}) \tag{2}$$

leads to a possible sub-optimal candidate solution. However, this sub-optimal solution can be used as an initial guess to invoke the optimization routine. A sphere centered around $\bar{U}_{\mathrm{unc}}$ is spanned and the radius ρ_{ini} is set in such a way that the initial solution U_{ini} just lies within the sphere. This ensures that at least one valid solution will be found. As shown in the pseudo-code of the modified sphere decoding algorithm, a possible switching vector U is built up componentwise in a recursive way as long as it lies within a sphere described by the radius ρ. This is the case, if the squared distance between the unconstrained solution vector and the current switching vector candidate $U_{1:i}$ is less than the current squared distance ρ^2. If the candidate vector is not inside the current sphere, the search in this branch of the tree is aborted immediately since it will lead to a sub-optimal solution. Once the vector has full length, e.g. $i = 3N_{\mathrm{p}}$, a better solution was found and the radius of the sphere is reduced to the new squared distance between the unconstrained solution and the new best solution. By doing so, the sphere radius is tightened every time a better switching vector is found, and at the end of the routine only the single switching vector minimizing the cost-function lies within the sphere. A complete mathematical derivation of the control problem is given in [2].

[1]For a three-level NPC topology valid entries are: $U_{\mathrm{set}} = \{-1, 0, 1\}$

The sphere decoding algorithm is invoked as

$$U_{\text{opt}} = \text{mSDA}([\,], 0, 0, \rho^2_{\text{ini}}, \bar{U}_{\text{unc}}) \tag{3}$$

and returns the optimal switching vector U_{opt} that is applied to the gate drivers.

3 Simulation

This model can run both in offline mode on a computer, and in real-time mode on the PLECS RT Box. For the real-time operation, two RT Boxes (referred to as “Plant” and “Controller”) need to be set up as demonstrated in Fig. 6 using three DB37 cables.

Please follow the instructions below to run a real-time simulation on two RT Boxes (one RT Box 1 for the “Plant” and one RT Box 2 or 3 for the “Controller”):

- Connect the Analog Out interface of the “Plant” RT Box 1 to the Analog In interface of the “Controller” RT Box 2/3, the Digital Out interface of the “Plant” RT Box 1 to the Digital In interface of the “Controller” RT Box 2/3, and the Digital In interface of the “Plant” RT Box to the Digital Out interface of the “Controller” RT Box.
- Connect the SFP+ innterconnect port A of the “Plant” RT Box to the SFP+ innterconnect port A of the “Controller” RT Box.
- From the **System** tab of the **Coder options...** window, select the “Plant” and **Build** it onto the “Plant” RT Box 1. Then, select “Controller" and **Build** it onto the “Controller” RT Box 2/3.
- Once both models are uploaded, from the **External Mode** tab of the **Coder options...** window, **Connect** to both RT Boxes and **Activate autotriggering**.

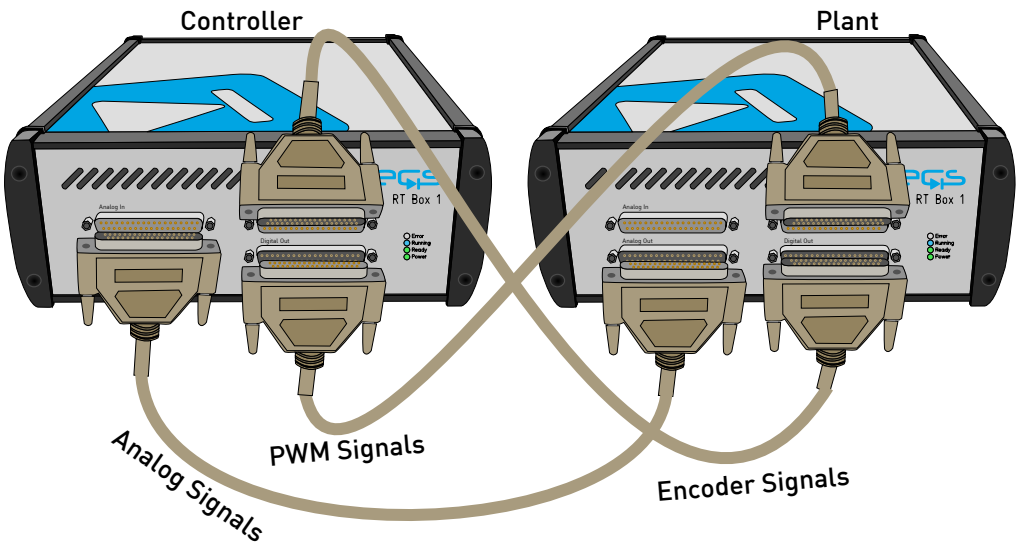

Figure 6: Hardware configuration for the real-time operation of the demo model

Key waveforms can be observed in the different Scopes individually placed in the “Plant” and “Controller” subsystems. The optimization of the predictive current controller can be enabled by toggling the “Enable Optimization” manual switch inside the controller subsystem from 0 to 1. If the optimization is disabled, the sphere decoding algorithm is not invoked and a sub-optimal explicit switching vector is applied to the inverter.

The Scope “Explored nodes” shows the number of switching combinations the optimization algorithm has explored before the optimal solution was found. This Scope is only relevant once the optimization has been enabled as explaind above.

The Manual switch to enable a load torque step in the plant subsystem is inlined and can therefore be toggled during a real-time simulation. When toggling the manual switch a load step of 0.5 p.u is applied to the drive system.

The main simulation results are shown in Fig. 7.

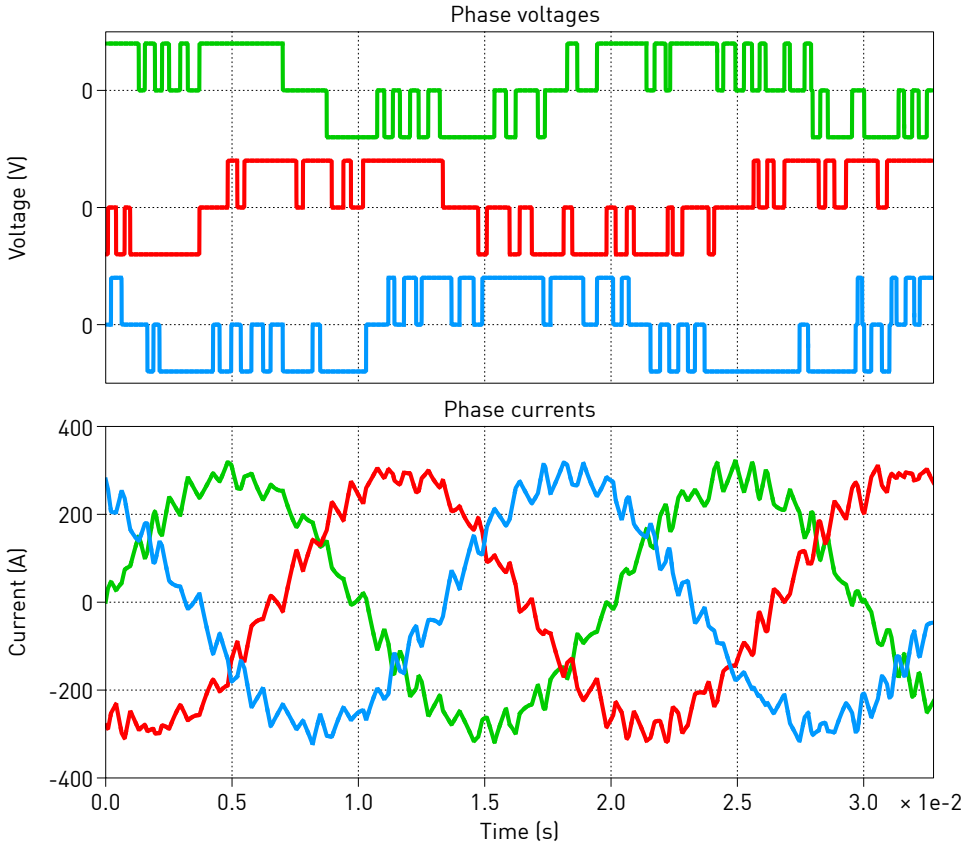

Figure 7: Steady-state voltage and current waveforms for a prediction horizon of $N_\mathrm{p} = 5$ and a load torque of 0.5 p.u. The cost function weighting factor was chosen to achieve a switching frequency around 300 Hz

In addition, the prediction horizon can be changed prior to an offline and real-time simulation. From the **Model initialization commands** window of **Simulation Parameters... + Initializations** tab from the **Simulation** menu, change the value of `Par.Ctr.Np` to a value between 1 and 5, to choose the desired prediction horizon. You must also configure the prediction horizon in the "MPC Controller" C-Script in the controller subsystem accordingly. Open the C-Script dialog and navigate to the **Code declarations** section in the **Code** tab. Change the value of `Np` to the same value as in the **Model initialization commands**. You can see the effect of different prediction horizons in the "Explored nodes" scope. The lower the prediction horizon is, the less nodes have to be explored by the sphere decoding algorithm. Please note, that the model must be reloaded on both RT boxes after changing the prediction horizon:

- Open the web interface of each RT Box by clicking on the icon under the **Target** or the **External Mode** tabs of the Coder Options dialog.
- Stop the execution of the real-time simulation by clicking on the **Stop** button at the bottom of the **Application** tab in the web interface.
- Re-build the "Plant" and "Controller" subsystems on the respective RT Boxes as described in Section 3.

4 Conclusion

This model demonstrates a medium voltage three-level neutral-point-clamped voltage source inverter driving an induction machine. The drive system is closed-loop controlled by means of outer flux and speed controllers and an inner predictive current controller. The model can run both in offline and real-time operation for Hardware-in-the-loop testing and rapid control prototyping.

References

[1] J. Allmeling, and N. Felderer, “Sub cycle average models with integrated diodes for real-time simulation of power converters,” IEEE Southern Power Electronics Conference (SPEC), 2017.

[2] T. Geyer and D. E. Quevedo, “Multistep Finite Control Set Model Predictive Control for Power Electronics,” in IEEE Transactions on Power Electronics, vol. 29, no. 12, pp. 6836-6846, Dec. 2014, doi: 10.1109/TPEL.2014.2306939.

[3] L. Capponi, “Direct Model Predictive Control Using Model Based Design,” IEEE International Conference on Predictive Control of Electrical Drives and Power Electronics (PRECEDE) , 2021.

Revision History:

RT Box Target Support Package 2.2.1	First release

How to Contact Plexim:

☎	+41 44 533 51 00	Phone
	+41 44 533 51 01	Fax
✉	Plexim GmbH Technoparkstrasse 1 8005 Zurich Switzerland	Mail
@	`info@plexim.com`	Email
	`http://www.plexim.com`	Web

RT Box Demo Model

© 2002–2022 by Plexim GmbH

The software PLECS described in this document is furnished under a license agreement. The software may be used or copied only under the terms of the license agreement. No part of this manual may be photocopied or reproduced in any form without prior written consent from Plexim GmbH.

PLECS is a registered trademark of Plexim GmbH. MATLAB, Simulink and Simulink Coder are registered trademarks of The MathWorks, Inc. Other product or brand names are trademarks or registered trademarks of their respective holders.