# RT Box

## *Tutorial*

## Model Optimizations

Tutorial Version 1.0

# 1 Introduction

As real-time simulation models increase in complexity additional optimization steps may be required to achieve the desired performance. This tutorial provides additional optimization techniques for real-time simulation on the RT Box with a focus on hardware-in-the-loop (HIL) applications.

At the end of the tutorial you should understand how to reduce the number of switching combinations in a PLECS model. You will observe how this reduces the model execution time and reduces the size of the generated code.

**Before you begin** This exercise requires knowledge about the previous RT Box Tutorials. This tutorial requires an RT Box and a PLECS Coder license.

# 2 Optimization Concepts

When real-time simulation models increase in complexity, especially in HIL applications, the time required to compute the set of equations describing the model will naturally increase. For switching power converters, as the number of switches in a model increases, the total number of potential switching combinations will increase exponentially. With more switching combinations, the code describing the physical model of the power converter becomes larger as well. As code size increases, generally the time it takes to execute the model in real-time increases in turn.

At a certain point the execution time of the model may exceed the required discretization time step. Alternatively, the size of the generated code may exceed the storage capabilities of the real-time hardware. There are three common methods used to address these constraints:

- Model splitting
- Switch reduction techniques and sub-cycle average models
- Numerical optimization

These techniques are described in more detail in the following sections.

## 2.1 Model Splitting

For larger systems, a key optimization approach is to numerically split the model into two or more sets of state-space equations. In the electrical domain, the split commonly occurs at a capacitor or inductor. A controlled current source is placed in one part of the model and a controlled voltage source in the other part. Both sources are controlled by the respective measurements in the other part of the model. This numerically decouples the state-space matrices in each part of the model. Model splitting reduces the total number of switching combinations and decreases the size of the state-space matrices.

As an example, consider a circuit with a three phase rectifier and a single phase neutral point clamped (NPC) inverter, as shown in Figure 1. PLECS generates a unique state-space matrix describing the circuit for each switching combination. The rectifier circuit has $2^6$ potential switching combinations, as each of the six diodes can either be conducting or blocking. Similarly, the inverter circuit has $2^6$ potential switching combinations when each IGBT Diode pair is considered as a single switch.

With the inverter and rectifier directly coupled via the DC bus, each switch directly influences the current and voltages of the entire circuit. Since there are six states (four inductors and two capacitors) a $6 \times 6$ state matrix is required for each switching combination. This results in a total of $2^{12}$ or 4096 different switching combinations, and 4096 different $6 \times 6$ matrices.
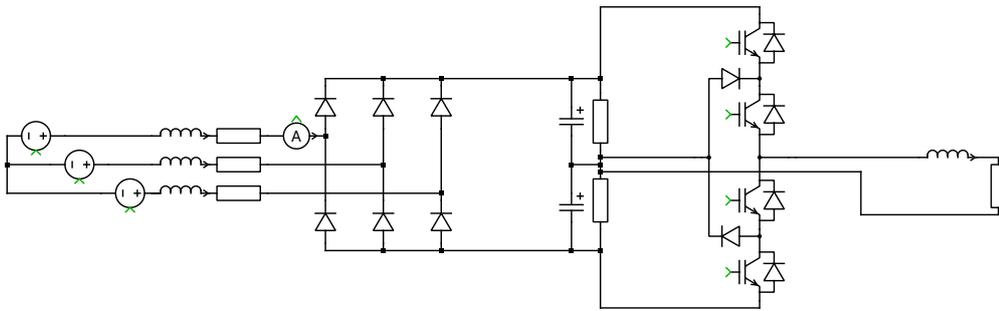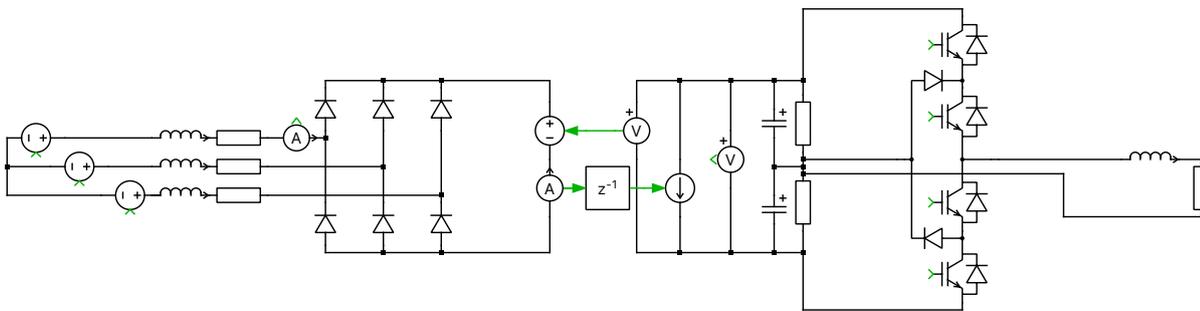
$$\text{Rectifier: } 2^6 \text{ combinations} \qquad \text{VSI: } 2^6 \text{ combinations}$$

$$\begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ . & . & . & . & . & . \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} 1 & 1 & 0 & 0 & & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ . & . & . & . & . & . \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} = 4096 \text{ switching combinations}$$

**Figure 1: Rectifier and inverter circuit without a model split**

Figure 2 shows the same circuit but split at the DC bus via controlled voltage and current sources. The use of controlled current and voltage sources electrically decouples the state-space matrices of the rectifier circuit and the inverter circuit. Now the switching state of the rectifier does not directly couple to the switching state of the inverter, and vice versa. The total number of switching combinations drops to $2^7$ or 128. Additionally, the size of the state-space matrix for each switching combination decreases from a $6 \times 6$ matrix to two $3 \times 3$ matrices; a factor of two reduction in matrix size and associated computations.



$$\text{Rectifier: } 2^6 \text{ combinations} \qquad \text{VSI: } 2^6 \text{ combinations}$$

$$\begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ . & . & . & . & . & . \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 1 & 1 & 0 & 0 & & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ . & . & . & . & . & . \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} = 128 \text{ switching combinations}$$

**Figure 2: Rectifier and inverter circuit split with a gyrator**

## 2.2  Switch Reduction Techniques and Hybrid Models

The hybrid power modules described in earlier tutorials not only increase the effective range of duty cycles, but also can significantly reduce the number switching combinations for many topologies.

For example, the conventional implementation of an NPC half-bridge is shown in Figure 3. The NPC half-bridge consists of six switches when each IGBT Diode pair is considered as a single switch. However, hybrid power module implementation for this circuit, shown in Figure 4, effectively requires only two diodes [2].

When this concept is applied to multi-phase circuits, such as the three-phase NPC inverter in Figure 5, there is a drastic reduction in the number of switches and switching combinations. Using conventional switch models there will be a total of $2^{18}$ different switching combinations in the three-phase NPC inverter. When hybrid switch models are used there are only $2^6$ different switching combinations for the same circuit. The overall model complexity is significantly decreased.
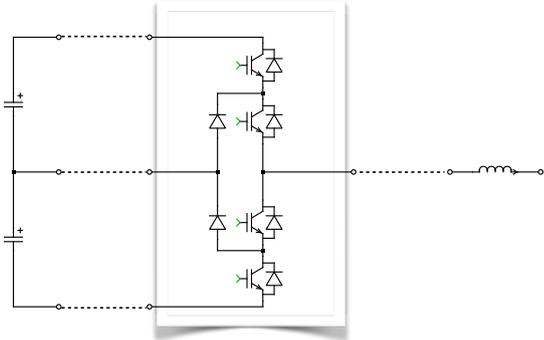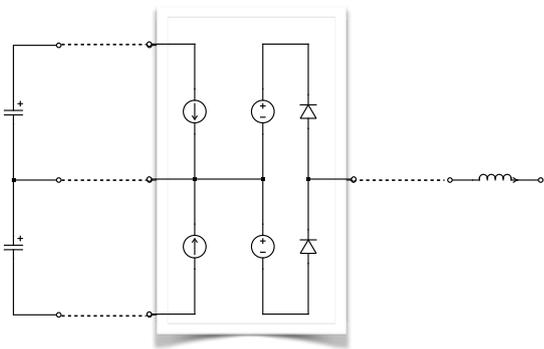


**Figure 3: Conventional NPC electrical configuration**



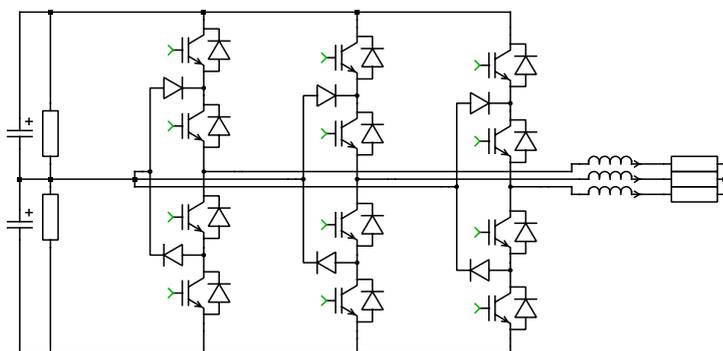**Figure 4: Hybrid power module NPC electrical configuration**



**Figure 5: Hybrid power module NPC electrical configuration**

## 2.3  Special Numerical Optimization

There are certain calculation tasks that require more computation time than others. Generally, non-linear calculations are especially time consuming, for example certain trigonometric functions.

While developing a real-time simulation model it is important to keep the total number of computationally intensive tasks in mind, remove any unnecessary calculations, and to consider more efficient numerical implementations of these tasks.

Note that at the code generation step some functions automatically use discrete optimizations. For example, the sine wave generator does not make a function call to the `sin(x)` function from the `math.h` library, but uses a more efficient discrete sine wave calculation technique.

## 2.4  Model Settings Block

The Model Settings block lets you configure parameter settings that influence the code generation for a particular physical system (electrical, rotational, translational) that is attached to the terminal of the block. The Model Settings block affects the physical circuit that it is attached to by its terminal. An physical model can be split into multiple state-space systems if the underlaying model equations are fully decoupled. To allow PLECS to split the state-space model into smaller independent models navigate to the **Simulation + Simulation parameters...** drop-down menu. In the **Simulation Parameters** window click the **Options** tab and check the **Enable state-space splitting** option. In the same window check also the **Display state-space splitting** parameter. With this configuration, PLECS will issue diagnostic messages that highlight the components that make up the individual state-space models after splitting. At most one Model Settings block may be connected to each individual state-space system.

### Switching algortithm

When generating code for a physical model, PLECS lets you choose between two switching algortithms, **Iterativ** and **Direct Look-up**. You can specify the algortihm indivdually for each independent state-space model. The default switching algrorithm is **Direct Look-up**. See also the **Code Generation for Physical Systems** chapter in the PLECS User Manual for further details.

### Matrix Coding Style

This setting allows you to specify the format used for storing the state-space matrices for a physical system. When set to `sparse`, only the non-zero matrix entries and their row and column indicies are stored. When set to `full`, matrices are stored as full $m \times n$ arrays. When set to `full (inlined)`, the matrices are additionally embedded in helper functions, which may enable the compiler to further optimize the matrix-vector-multiplications at the cost of increased code size.

# 3  Optimization of an Induction Machine Drive

In this exercise you will look at a series of optimization steps for an induction machine drive. Initially you will reduce the number of switching combinations by changing the switch type. Then, you will electrically split the model at the DC link to further reduce the number of switching combinations. You will compare the execution time and size of the generated code following both optimization steps.

**Your Task:**

**1** Open the provided `dtc_start.plecs` case and explore the model including the "Circuit" and "Direct Torque Control" subsystems. After reviewing the model, run the simulation with a variable-step solver and then hold all scope traces.

**2** Change to a fixed-step solver from the **Simulation + Simulation parameters** drop-down menu and choose a simulation time step. Compare the results against the variable-step solution.

**(?)** What is a reasonable time step?

**[A]** Typically the time step is at least an order of magnitude less than the switching frequency. However, with the hysteretic controller used in this model the switching frequency is variable. A $5e{-}6$ second fixed-step solution is reasonable. Further reduction in the step size can improve the accuracy of the simulation, but at the cost of additional computational overhead.

Figure 6 shows the results for a $5e{-}6$ second fixed-step solution and a variable-step solution. Due to the limited time resolution of the fixed-step solver and a hysteretic controller, there will always be some error between the two numerical methods.
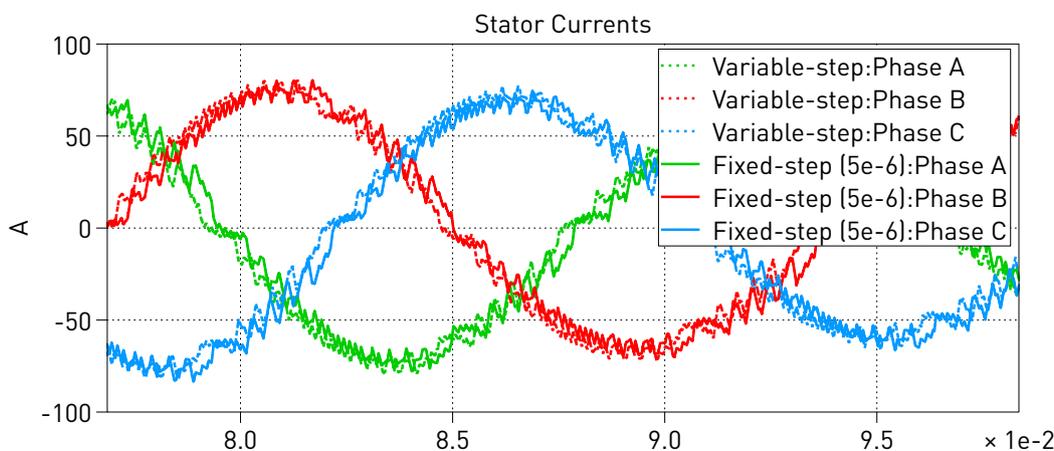


**Figure 6: Comparison between variable-step and fixed-step solution**

**3** With the discrete time step you have chosen, generate code for the "DTC" subsystem without any modifications. You may cancel code generation if you deem the process is taking too long.

**(?)** Why is the generation of code for this system problematic?

**[A]** With six diodes on the rectifier input and six IGBTs on the motor drive, there are $2^{12}$ or 4096 different switching combinations, as the Coder by default assumes any switching combination is possible. The large number of switching combinations results in increased code size and model complexity.

**4** Next, reduce the number of switching combinations in the inverter system by replacing the IGBT with diodes with the Double Switch component from the PLECS library. See Figure 7.

**(?)** What are the limitations of using the Double Switch component?

**[A]** The Double Switch component does not have built in diodes to represent inverter blanking states. The Double Switch component is not compatible with the PWM Capture block and cannot detect shoot-through states. Hybrid power modules overcome these limitations.
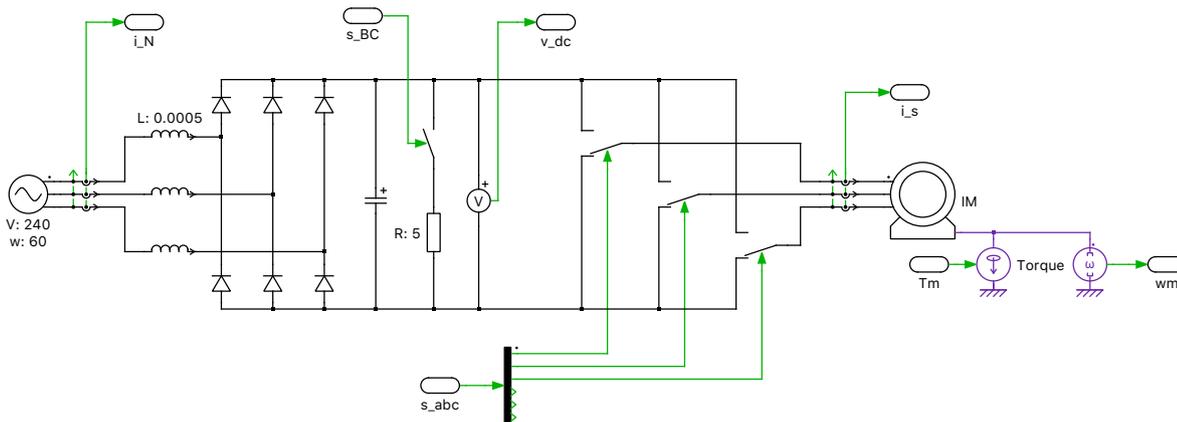
**Figure 7: Induction motor drive with Double Switches**

**5** Generate code and run the model on the RT Box.

**(?)** What is the execution time?

**[A]** The execution time will be close to $5.5\,\mu s$. This exceeds the $5\,\mu s$ discretization step size. Note that the current cycle time is actually the peak cycle time over several simulation time steps. The maximum number of consecutive overruns that may occur before the simulation is aborted is defined in the **Coder options + Target** tab.

**6** In your file browser navigate to where you have saved your PLECS model. Open the _codegen folder and look for the DTC.c file. This file contains the generated code for the "DTC" subsystem.

**(?)** What is the file size of DTC.c in megabytes (MB)?

**[A]** The file is $15.2\,\text{MB}$.

**⚑** At this stage your model should be the same as the reference model dtc_1.plecs.

**☑ Your Task:**

**1** Split the model into two electrically separate parts as shown in Figure 8.

**(?)** Why is this a suitable place to electrically split the model?

**[A]** Due to the large DC link capacitance, the capacitor voltage dynamics are slow compared to the simulation time step; the voltage does not change significantly from time step to time step. Therefore it is suitable to model the capacitor as controlled voltage and current sources as the latency introduced by the controlled sources has a negligible on impact the voltage dynamics.
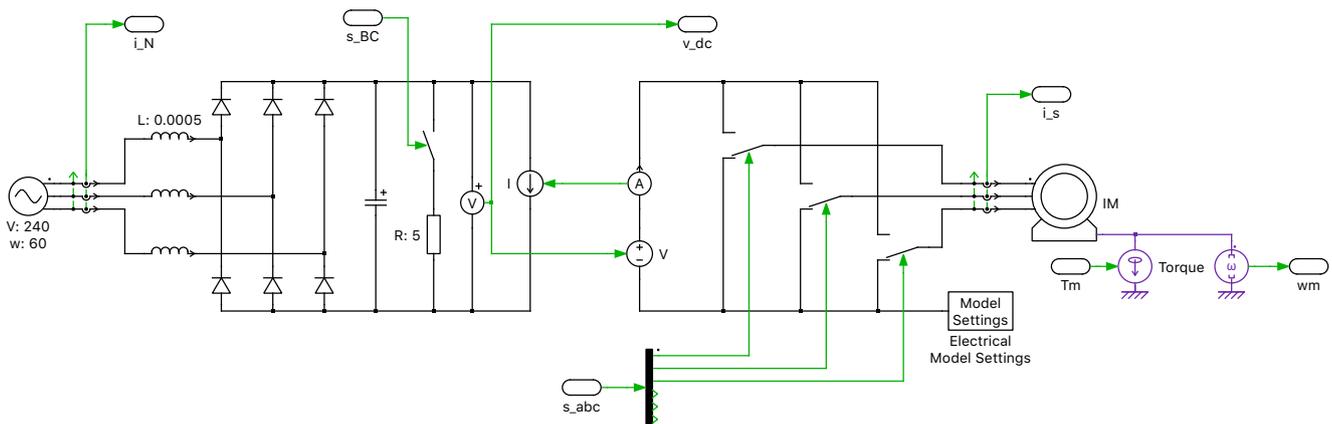
**Figure 8: Induction motor drive with Double Switches and DC bus split.**

💡 **Note:** Note that the **Discretization behavior** of the controlled current and voltage sources in Figure 8 has to be set to **Zero-order hold**. In this configuration, only the input value from the previous simulation step is required to calculate the current state variables.

**2**  Generate code and deploy the model to an RT Box.

❓ What is the execution time of the model? How does this compare with the results in Step 5?

🅐 The execution time is reduced to $3.5\,\mu s$ from $5.5\,\mu s$ in Step 5.

❓ How can the blanking state of the inverter be included?

🅐 Replace the Double Switches with IGBT Half Bridge components from the "Electrical", "Power Modules" section of the PLECS Library.

**3**  Inspect the size of the DTC.c file.

❓ What is the file size of DTC.c now? How does this compare with the results in Step 6?

🅐 The DTC.c file is now $0.660\,\text{MB}$. This is 20 times smaller than the results in Step 6.

🏁 At this stage your model should be the same as the reference model dtc_2.plecs.

**4**  Open the provided dtc_1.plecs and dtc_2.plecs models. Navigate to the **Simulation + Simulation parameters...** drop-down menu. In the **Simulation Parameters** window click the **Options** tab and notice that the **Display state-space splitting** option is checked. With this configuration, PLECS will issue diagnostic messages that highlight the components that make up the individual state-space models after splitting. This is a useful diagnostic to confirm you have successfully split the state-space matrix.

Run the models and open the PLECS Diagnostics window by clicking the exclamation icon ⓘ in the bottom right hand corner of the simulation window.

(?) What is the total number of state-space models in `dtc_1.plecs` and `dtc_2.plecs` as indicated in the PLECS Diagnostics window? Does this confirm your expectations based on the previous steps?

[A] You will observe three state-space systems. One state-space system is associated with the rectifier circuit, a second represents the motor and drive system, and a third represents the mechanical domain including the motor shaft. This confirms the electrical network was successfully split into two smaller state-space systems that are numerically coupled through controlled voltage and current sources.

**5** Connect an Electrical Model Settings block to the drive system on the right side of the splitting circuit, as indicated in Figure 8. Set the **Switching algorithm** parameter to `direct look-up` and the **Matrix coding style** to `full (inlined)`. Generate Code and deploy the model to an RT Box.

(?) What is the execution time of the model? How does this compare with the results in Step 2?

[A] The execution time is reduced to $3.1\,\mu s$ from $3.5\,\mu s$ in Step 2.

🏁 At this stage your model should be the same as the reference model `dtc_3.plecs`.

# 4 Conclusion

In this tutorial you learned different optimization techniques for real-time models and applied these in a practical circuit. The execution time and code size was evaluated at each optimization step. You were able to directly observe how reducing the number of switching combinations in a PLECS model improves the real-time performance.

With an understanding of these topics, you should now be able to optimize your own real-time models.

# References

[1] Jost Allmeling, Niklaus Felderer "Sub-cycle average models with integrated diodes for real-time simulation of power converters", 2017 IEEE Southern Power Electronics Conference (SPEC), pp. 1-6, 2017.

[2] Jost Allmeling, Niklaus Felderer, Min Luo, "High Fidelity Real-Time Simulation of Multi-Level Converters", Power Electronics Conference (IPEC-Niigata 2018 –ECCE Asia) 2018 International, pp. 2199-2203, 2018.

**Revision History:**

Tutorial Version 1.0    First release
Tutorial Version 1.1    Added explanation of Model Set-
                        tings block

**How to Contact Plexim:**

*RT Box Tutorial*