PLECS

Tutorial

Implicit Model Vectorization

Learn about the different vectorization techniques found in PLECS and how to implement them

Tutorial Version 1.0

1 Introduction

In PLECS, many components may be vectorized in order to be interpreted as several components and/or handle multiple signals at once. This feature is useful in avoiding the duplication of a part of a circuit several times and allows describing a large system within a smaller schematic space. This functionality may also be used more extensively to allow resizing of the system by simply changing the length of a vector or a value in a parameter mask. This reconfiguration capability is particularly useful when searching for an optimal number of cells, such as in series/parallel multilevel converters or modeling solar panels.

In this exercise you will learn the following:

- How to add vectors to component parameters.
- Which blocks perform basic vectorizing functions.
- How to understand and design circuits in a more condensed form.
- How to implement a user-defined parameter to make variable structures.

Before you begin Ensure the files `implicit_model_vectorization_1.plecs` through `implicit_model_vectorization_9.plecs` are located in your working directory.

2 Vectorization Examples

2.1 Including a Vector in a Component

Vector parameters are entered using square brackets with optional commas for separating the elements, e.g. `[1,2,3]` or `[1 2 3]`. Many components in PLECS may be vectorized, including the Sine Wave Generator block, Constant block, as well as electrical sources just to name a few. Please refer to the PLECS User Manual to find out which blocks may be vectorized.

A simple example of vectorization is shown in Fig. 1. Here, the resistor on the left is configured as a vector of three resistance values, which effectively equates this circuit to the three individual circuits shown on the right. Please note that vectorizing the voltage source in the circuit on the left of Fig. 1 isn't necessary as having a vectorized component within a circuit effectively vectorizes the additional scalar components within this circuit. The voltage of $1\,V$ is thus extended as `[1 1 1]V` and connected across the resistor vector elements of `[1 2 3]` Ω. This implicit extension of the voltage source values can circumvented by directly specifiying a vector of length 3 with the wanted values.

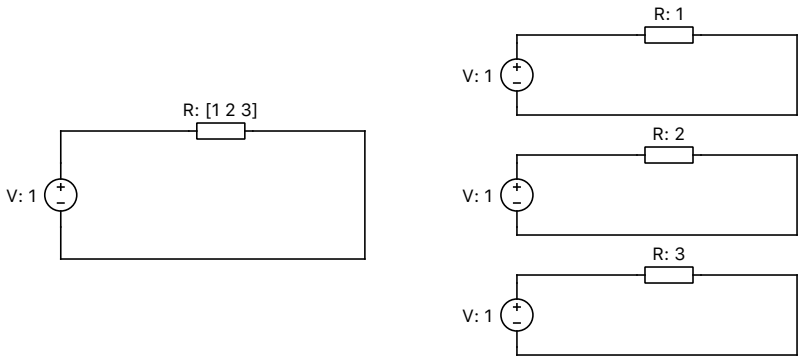

Figure 1: Vectorized resistor (left) and non-vectorized equivalent (right) circuits

Your Task: Open the model `implicit_model_vectorization_1.plecs` and confirm that the two circuit implementations are indeed equivalent by running a simulation and comparing the Display block results.

Looking at the left (vectorized) circuit, what should the value of the input voltage be in order to feed $2\,A$ to each of the resistor's vector's elements (`[1 2 3]` Ω)?

You would need to change the input voltage to a vector of `[2 4 6]` V. When connecting several vectorized components within a circuit, they must all be of the same size.

2.2 Vectorized Components in Series

Using wire multiplexer blocks

In order to interconnect several individual wires into a wire bus, Wire Multiplexers may be used. In Fig. 2, an example of a series connection of components is shown in a vectorized and an equivalent non-vectorized configuration. To follow the signal flow in the vectorized circuit on the left, trace the

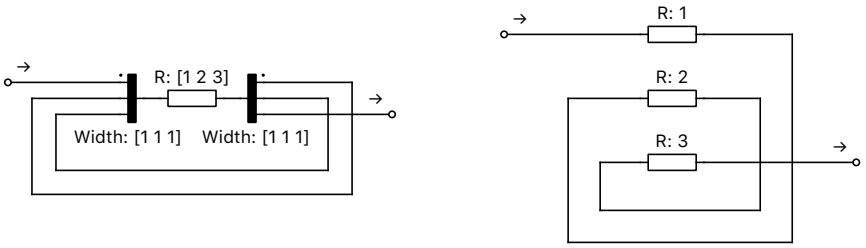

Figure 2: Vectorized resistor in series (left) and non-vectorized equivalent (right) circuits

wire from the unconnected end on the left through the first port of the Wire Multiplexer, through the first value of the resistor, out of the first (dotted) port of the second Wire Multiplexer, and so on to the other unconnected wire port on the bottom right. The non-vectorized equivalent circuit on the right is laid out in the same geometric pattern for visual aid. Notice that the width of the vector elements within the two Wire Multiplexer blocks (`[1 1 1]`) enables the current to flow through each of their ports only once. A new resistor vector element is used every time the current enters that resistor. When all elements have a width of 1, you could also use a scalar value, e.g. 3, without square brackets.

Note: The port in the Wire Multiplexer that has a dot next to it refers to the first element in the (de-) multiplexed signal. This follows the convention of other library blocks like the Signal (de-) Multiplexer or the transformer components (where the dot indicates the first winding of the primary and secondary side).

Your Task:

1. Open the model `implicit_model_vectorization_2.plecs` and confirm that the two circuit implementations are indeed equivalent by running a simulation and comparing the Display block results.
2. Add a fourth value to the end of the **Resistance** vector (e.g. 4) and change the left and the right Wire Multiplexer **Width** vectors to `[1 2 1]` and `[2 1 1]`, respectively. Run a simulation and drag one of the bottom corners of the Display block down to show a fourth window.

Your model should now look like Fig. 3.

Essentially, we've just added another resistance value and re-used one of the wire loops to account for

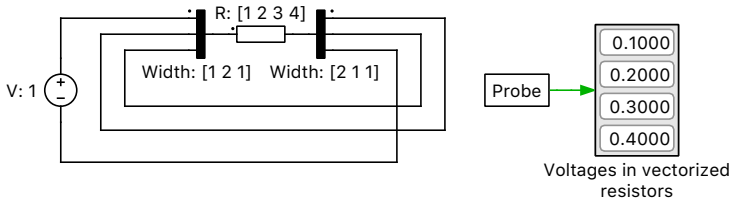

Figure 3: Vectorized resistor in series, modified

it by expanding the width of the two Wire Multiplexer ports around that wire. Widths of `[1 1 2]` and `[1 2 1]` would have produced the same results, just utilizing a different wire loop.

The numbers in Fig. 4 below are intended to help illustrate the current flow. Start with the ① at the top left, connect it with the ① to travel across the resistor bus, follow the wire loop, and only increment to ② when you reach the ② located on the left side of the resistor. Continue in the same fashion until you reach the ④ at the bottom right, which is the wire connected to the negative terminal of the voltage supply.

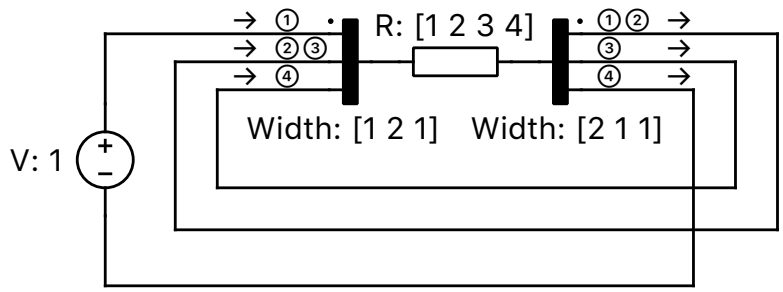

Figure 4: Current flow explanation

Further consolidation

As you just witnessed, it is possible to re-use the ports in a vectorized circuit by increasing the width of the Wire Multiplexer ports. If we reduce the number of loops to just one and place the resistor into it, the circuit will have even less elements, as seen in Fig. 5 below. The number of times current will pass through this loop will depend on the width of the two ports which the wire loop is attached to. In this case, since the width of those ports is 3, the loop is engaged three times, thus creating an equivalent, even more condensed version of the same circuit.

Your model should now look like Fig. 5.

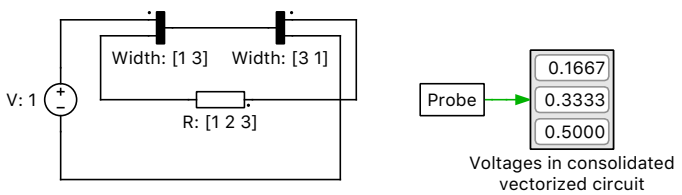

Figure 5: Further consolidated vectorized model

Your Task: Open the model `implicit_model_vectorization_3.plecs` and confirm that the condensed circuit implementation produces the same results as the original one.

How would you change this circuit to accommodate for another resistance value?

Add another value to the **Resistance** vector and change the **Width** of the Wire Multiplexer from 3 to 4 (i.e. `[1 4]` and `[4 1]`.

2.3 Vectorized Components in Parallel

To vectorize components connected in a parallel fashion, a Wire Selector block is used. As you can see in Fig. 6, the resistor is vectorized and surrounded by a Wire Selector component from each side. The output indices of the Wire Selector blocks match the vector size of the resistor.

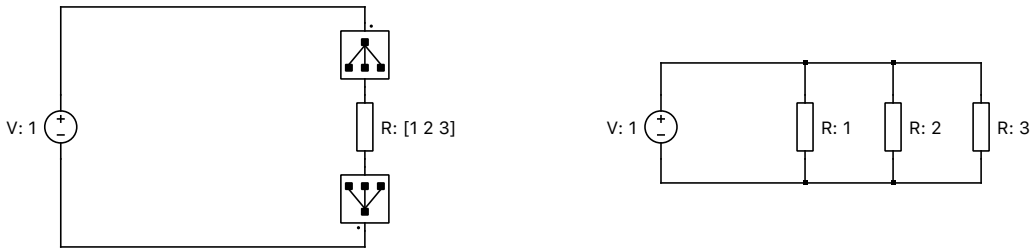

Figure 6: Vectorized resistor and non-vectorized equivalent

Your Task:

1. Open the model `implicit_model_vectorization_4.plecs` and confirm that the two circuit implementations are equivalent by running a simulation and comparing the Display block results.
2. Add a fourth value to the end of the **Resistance** vector (i.e. 4) and add a fourth element to the Wire Selector output vectors. Run a simulation and drag one of the bottom corners of the Display block down to show a fourth window.

Your model should now look like Fig. 7.

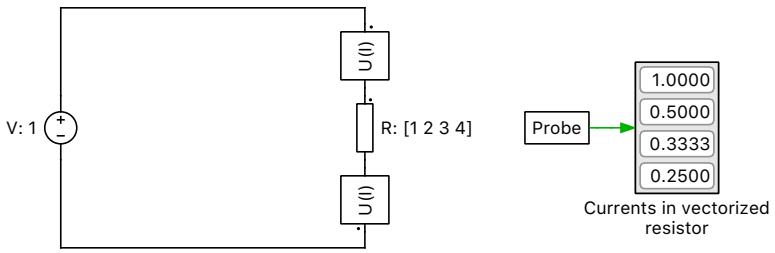

Figure 7: Vectorized resistor in parallel, modified

2.4 Vectorizing Circuits

Entire circuits themselves may be vectorized in a few different ways depending on what you're trying to achieve.

Your Task: Open the model `implicit_model_vectorization_5.plecs` and compare the three vectorized circuits provided.

What's the difference between these circuits?

- In the first (top-most) circuit, the vectorized component (inductor) effectively vectorizes the rest of the scalar components, thus functionally making this equivalent to having three separate circuits. This is why you see three load voltages in the Scope. This is also what happens in Fig. 1 above.
- In the second case, the vectorized inductor is separated from the rest of the scalar components via Wire Selector blocks. Such a configuration is equivalent to having three inductors in parallel with each other – the same concept as Fig. 6 above.
- In the third model, an entire part of the Buck converter is separated from the capacitor and the load via Wire Selector components. This creates an interleaved buck converter, where three instances of that left part of the circuit are all parallelized and connected to a common capacitor and load resistor. A phase delay was added in this case in order to differentiate this circuit from the one above it.

2.5 Combining Series and Parallel Vectorization

Next, lets use the concepts from Fig. 5 and Fig. 6 to follow a vectorized circuit consisting of series and parallel resistors, as seen on the left circuit in Fig. 8. If we follow the current flow, we see that

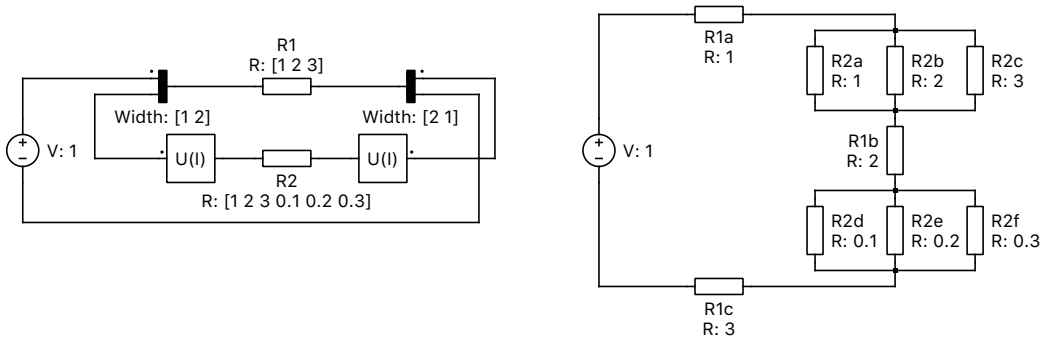

Figure 8: Vectorized (left) and equivalent (right) circuits of series and parallel resistors

it would first enter "`R1`" (corresponding to "`R1a`" in the circuit on the right with the value of $1\,\Omega$), then flow through the wire loop containing paralleled resistors "`R2`" (corresponding to "`R2a`", "`R2b`", and "`R2c`" on the right), then flow back into "`R1`", which obtains the next value from its vector ("`R1b`", which is $2\,\Omega$), and so on.

Your Task: Open the model `implicit_model_vectorization_6.plecs` and confirm that the two circuit implementations are equivalent by running a simulation and comparing the Display block results for the resistor currents and voltages.

2.6 Using a Variable

Instead of hardcoding the number of iterations of a wire loop or the number of components in parallel, you may take a more dynamic approach using a variable. This variable could either be defined within the **Simulation Parameters (Ctrl+E) + Initialization** tab, or within the mask of a Subsystem block.

Within the initialization tab

Your Task:

1. Open the model `implicit_model_vectorization_7.plecs`, which contains the final series and parallel models from Sections 2.2 and 2.3.
2. Replace both **Resistance** vectors as well as the **Output Indices** of the Wire Selector blocks with `ones(1,n)*1`, which is an array of all ones with the length of n, which we will soon define. Since this is an array, the square brackets are optional in this case. The `*1` at the end is optional in this case as the vector is already a series of ones, but that's the value you would change if you'd like to initialize the Resistance vector with a uniform value different from 1.
3. In the Wire Multiplexers' parameter window change the **Width** parameter to `n` as well, i.e., `[1 n]` and `[n 1]`. Remember that the first element of the Wire (de-) Multiplexer refers to the value given to the port which has the dot next to it.
4. Under **Simulation Parameters (Ctrl+E) + Initialization** tab, define `n=5;`. This is now the parameter you are able to change to dynamically reconfigure your vectorized circuits.
5. Run a simulation and verify that the currents and voltages you are getting make sense.

At this stage, your model should be the same as the reference model `implicit_model_vectorization_8.plecs`.

Within a masked subsystem

You may instead choose to create a Subsystem out of your circuit and pass the parameter value through its mask. To do that, follow the steps below for both circuits.

Your Task:

1. For one of the circuits, select everything but the voltage source, right-click on the resistor and select **Create subsystem (Ctrl+G)**.
2. An optional step could be to move the newly created ports `1` and `2` to a better suited location on the Subsystem's perimeter by clicking and dragging them while holding **Shift**. You may also move the Subsystem's label outside of the block itself.
3. Right-click on the Subsystem and navigate to **Subsystem + Create mask... (Ctrl+M)**.
4. Within the **Dialog** tab click on the **+** icon to create a new dialog parameter. If you are working with the "series" circuit, you may enter something like `Number of resistors in series` for the **Prompt** and `n` for the **Variable**. You may leave the **Type** as **Edit**. Click **OK**. For more information on using Subsystem masks to create custom components in PLECS, please refer to the `Custom Components` tutorial or the **Help** button within the Subsystem component's parameter window.

5. Double-click on the Subsystem now and enter the value that you would like for `n` now, such as 5. Simulating should produce 5 display block rows now with appropriate currents/voltages. Note that the `n` which you define within the mask of the Subsystem component will only affect any instances of `n` within that Subsystem block itself.
6. Add a variable `R` to the mask in the same fashion, which defines the resistance value.

You can see this concept in application in the `Statcom Cascaded H-Bridge Converter` demo model for implementing multiple H-bridge cells in series. For more examples on creating complex vectorized models in PLECS, you may refer to [1].

At this stage, your model should be the same as the reference model `implicit_model_vectorization_9.plecs`.

3 Conclusion

Vectorized circuits are useful for creating structures of repeated components where the number of repeated components should be configurable by a single parameter. The main advantage of a vectorized implementations over a standard form with individual components is that vectorized designs are far less prone to "copy&paste" errors. On the other hand, the condensed form of such vectorized models may also make a bit more difficult to read at a first glance. This tutorial covered the basic vectorization examples as well as the functionality of the blocks which are used to create such structures. We hope you will bring the concepts seen in this tutorial to your future PLECS models.

References

[1] Thierry Meynard, *Analysis and Design of Multicell DC/DC Converters Using Vectorized Models*, ISTE Ltd, 2015, pp 23-24.

Revision History:

Tutorial Version 1.0	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

PLECS Tutorial