Universidad Complutense de Madrid

Universidad Nacional de Educación a Distancia

Master's Degree in Systems and Control Engineering

Photovoltaic Power Converter Modeling in Modelica

Thesis presented by

Raúl Rodríguez Pearson

Under the supervision of

Alfonso Urquía Moraleda

Academic year of 2016/17

September 2017

Master's Degree Thesis

Master's Degree in Systems and Control Engineering

Photovoltaic Power Converter Modeling in Modelica

Type B Project

Specific project proposed by student

Thesis presented by

Raúl Rodríguez Pearson

Under the supervision of

Alfonso Urquía Moraleda

Autorización

Autorizamos a la Universidad Complutense y a la UNED a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la memoria de este Trabajo Fin de Máster, como el código, la documentación y/o el prototipo desarrollado.

Firmado: Raúl Rodríguez Pearson

Abstract

The challenges of climate change and increasing energy demand will require solutions based around renewable energy sources and a renovation of the ageing grid infrastructure. A promising solution is distributed and centralized photovoltaic systems coupled with energy storage.

Thinking up and validating new solutions to any problem greatly benefits from modelling and simulation, decreasing costs and development time. Modelica is a non-proprietary, object-oriented, equation based language to conveniently model complex physical systems.

This thesis describes the development and contents of a Modelica library for photovoltaic system modelling. The library has a focus on power electronics and is aimed at developers of control software for power converters in the photovoltaic domain.

The library includes models for a generic photovoltaic array, a simple Li-ion battery and switched and averaged power electronics. Some basic control blocks are also included to support the creation of application examples.

The library was developed using Dymola 2017 and OpenModelica 1.12 and requires Modelica Standard Library 3.2.2. The library is made freely available on GitHub using an MIT license.

Keywords: Modelica, Modelling, Simulation, Solar Photovoltaic Energy, Battery Energy Storage, Power Electronics.

Resumen

Los retos del cambio climático y una creciente demanda de energía requerirán soluciones basadas en recursos renovables y una renovación de la envejecida red eléctrica. Una solución prometedora es la tecnología solar fotovoltaica distribuida y centralizada combinada con almacenamiento de energía.

El modelado y simulación aporta muchos beneficios en la concepción y desarrollo de nuevas soluciones, como la reducción de coste y el tiempo de desarrollo. Modelica es un lenguaje no propietario, basado en ecuaciones y orientación a objetos, para el modelado físico de sistemas complejos.

Este trabajo fin de máster describe el desarrollo y el contenido de una librería Modelica para el modelado de sistemas fotovoltaicos. La librería presta especial atención a la electrónica de potencia y está dirigida a los desarrolladores de software de control de convertidores de potencia en aplicaciones fotovoltaicas.

La librería incluye modelos de un array fotovoltaico genérico, una batería sencilla de litio y electrónica de potencia conmutada y promediada. También se incluyen bloques básicos de control para permitir la creación de ejemplos de aplicación.

La librería ha sido desarrollada con Dymola 2017 y OpenModelica 1.12 y require la librería estándar de Modelica 3.2.2. La librería se hace disponible en GitHub usando una licencia MIT.

Palabras clave: Modelica, Modelado, Simulación, Energía Solar Fotovoltaica, Almacenamiento de Energía, Electrónica de Potencia.

Contents

List of Figures

List of Tables

List of Listings

1 Introduction, goals and structure

1.1 Introduction

Why photovoltaics

We need energy. That's a fact. The evolution of technology and the abundance it creates is fuelled by energy. For most of human history, we've accessed this energy by burning things. We have recently come to the realization that this is not a good idea [Urb15].

Moving to renewable energy sources seems crucial, as it provides numerous benefits and solves many problems: creation of local environmental and health benefits; facilitation of energy access, particularly for rural areas; advancement of energy security goals by diversifying the portfolio of energy technologies and resources; and improving social and economic development through potential employment opportunities [EAB14].

An inspiring story about the power of technology to create abundance can be found in [DK14]:

Technology creates abundance

> Gaius Plinius Cecilius Secundus, known as Pliny the Elder, was born in Italy in the year AD 23. He was a naval and army commander in the early Roman Empire, later an author, naturalist, and natural philosopher, best known for his Naturalis Historia, a thirty-seven-volume encyclopedia describing, well, everything there was to describe. His opus includes a book on cosmology, another on farming, a third on magic. It took him four volumes to cover world geography, nine for flora and fauna, and another nine for medicine. In one of his later volumes, Earth, book XXXV, Pliny tells the story of a goldsmith who brought an unusual dinner plate to the court of Emperor Tiberius.
>
> The plate was a stunner, made from a new metal, very light, shiny, almost as bright as silver. The goldsmith claimed he'd extracted it from plain clay, using a secret technique, the formula known only to himself and the gods. Tiberius, though, was a little concerned. The emperor was one of Rome's great generals, a warmonger who conquered most of what is now Europe and amassed a fortune of gold and silver along the way. He was also a financial expert who knew the value of his treasure would seriously decline if people suddenly had access to a shiny new metal rarer than gold. "Therefore," recounts Pliny, "instead of giving the goldsmith the regard expected, he ordered him to be beheaded."
>
> This shiny new metal was aluminum, and that beheading marked its loss to the world for nearly two millennia. It next reappeared during the early 1800s but was still rare enough to be considered the most valuable metal in the

> world. Napoléon III himself threw a banquet for the king of Siam where the honored guests were given aluminum utensils, while the others had to make do with gold.
>
> Aluminum's rarity comes down to chemistry. Technically, behind oxygen and silicon, it's the third most abundant element in the Earth's crust, making up 8.3 percent of the weight of the world. Today it's cheap, ubiquitous, and used with a throwaway mind-set, but—as Napoléon's banquet demonstrates—this wasn't always the case. Because of aluminum's high affinity for oxygen, it never appears in nature as a pure metal. Instead it's found tightly bound as oxides and silicates in a claylike material called bauxite.
>
> While bauxite is 52 percent aluminum, separating out the pure metal ore was a complex and difficult task. But between 1825 and 1845, Hans Christian Oersted and Frederick Wohler discovered that heating anhydrous aluminum chloride with potassium amalgam and then distilling away the mercury left a residue of pure aluminum. In 1854 Henri Sainte-Claire Deville created the first commercial process for extraction, driving down the price by 90 percent. Yet the metal was still costly and in short supply.
>
> It was the creation of a new breakthrough technology known as electrolysis, discovered independently and almost simultaneously in 1886 by American chemist Charles Martin Hall and Frenchman Paul Héroult, that changed everything. The Hall-Héroult process, as it is now known, uses electricity to liberate aluminum from bauxite. Suddenly everyone on the planet had access to ridiculous amounts of cheap, light, pliable metal.
>
> Save the beheading, there's nothing too unusual in this story. History's littered with tales of once-rare resources made plentiful by innovation. The reason is pretty straightforward: scarcity is often contextual. Imagine a giant orange tree packed with fruit. If I pluck all the oranges from the lower branches, I am effectively out of accessible fruit. From my limited perspective, oranges are now scarce. But once someone invents a piece of technology called a ladder, I've suddenly got new reach. Problem solved. Technology is a resource-liberating mechanism. It can make the once scarce the now abundant.

I believe this is the role of technology and the responsibility for engineers, scientists and technologists is to tackle the most pressing problems by working on the progress of technology. Some argue this is also just a smart career choice [Tod16].

Why Modelica

To create the requisite breakthroughs to make this abundance a reality, technologists need tools that make this work easier. Model-based engineering is one of those powerful tools, even though the biggest part of its potential remains unrealized. As argued in [Vic15], modelling and simulation (M&S) could play a big role in solving climate change. One can also envision a future in which M&S is present in many ways [Mus+14]. Even today, in the field of embedded software development, the benefits of model-based

engineering are clear, despite the usability and interoperability issues of the present tools [Lie+14].

Modelica is non-propietary, object-oriented, declarative, multi-domain modelling language for component-oriented modelling of complex systems, e.g., systems containing mechanical, electrical, electronic, hydraulic, thermal, control, electric power or process-oriented subcomponents [16]. It enables equation-based description of systems, which greatly improves usability for model developers.

These ideas are the inspiration behind the effort to create `PVSystems`, on the one hand to understand photovoltaic (PV) technologies better and on the other to provide something useful to the advancement of those technologies.

1.2 Goals

This thesis has the following three goals:

1. Develop a *Modelica library of models for PV systems* focused mainly on developers of power electronic converters.

2. Provide a *review of the current technology* both related to PV systems and to M&S. In the case of photovoltaics, this includes the techniques and methods of analysis and design as well as the devices and systems currently available. In the case of M&S, the focus will be on practices, tools and languages. The intersection of both will cover the review of models relevant in the PV domain and in both cases, a review of the most immediate future will also be provided.

3. Explore and showcase *best practices of software engineering* relevant to the development of the library. This includes practices pertaining documentation, library architecture, as well as validation techniques.

1.3 Document structure

This document can be conceptually divided in two main parts: the first part includes Chapters 1–3 and provides some context for the development of this work; the second part includes Chapters 4–7 and details the work undertaken, that is, the development of the `PVSystems` Modelica library.

This first chapter, *Introduction, goals and structure*, provides the motivation for this thesis, and outlines its goals and structure. In the second chapter, *Technology review*, a brief technology review is presented both of photovoltaics and of M&S. This chapter address goal 2. In the third chapter, *Photovoltaic systems modelling*, the physics of the different photovoltaic system elements is discussed.

This physics discussion serves as the basis for the models that are presented in the fourth chapter, *`PVSystems` library*, which presents a detailed description of each of the components of the `PVSystems` library. The fifth chapter, *Development and verification*,

describes the work undertaken to validate the correctness of the models developed. This chapter also provides a short discussion of best practices related with the development of libraries of models, addressing goal 3. In chapter six, *Application*, application of the library is presented through the description of the example systems included with it. Finally, chapter seven, *Conclusions and future work*, presents some concluding remarks and ideas for future work.

In the electronic version of this document, most of the cross-references, citations and URLs are clickable.

The library discussed in this document is available for download at GitHub:

```
https://github.com/raulrpearson/PVSystems
```

An online version of the documentation for the latest release is available at:

```
https://raulrpearson.github.io/PVSystems/
```

2 Technology review

2.1 Introduction

In this chapter, a review of photovoltaic as well as modelling and simulation technologies will be presented.

2.2 PV systems

2.2.1 PV technology

A good high-level overview of the state and prospects of PV technology is provided in [Sme+16]. As presented in Figure 2.1, installed PV is growing exponentially and currently dominated by European countries.

Several metrics are used to establish how energy inputs relate to energy outputs of an energy technology, of which two are most prominent. First, the Net Energy Ratio (NER) value, expressed as a ratio, which evaluates the amount of energy an energy source contributes to society over its life-cycle, relative to the inputs required to establish the technology. Second, Energy Payback Time (EPT), an estimate of the duration of time expressed in months or years at which an energy source has "paid back" its initial energy input. It is expressed by taking the energy input necessary to produce and operate the energy technology and dividing by the outputs produced over a fixed period of time.

It seems that studies frequently underestimate the performance of current PV technology and it seems that current EPT for PV plants falls around 2.4 years, and its NER

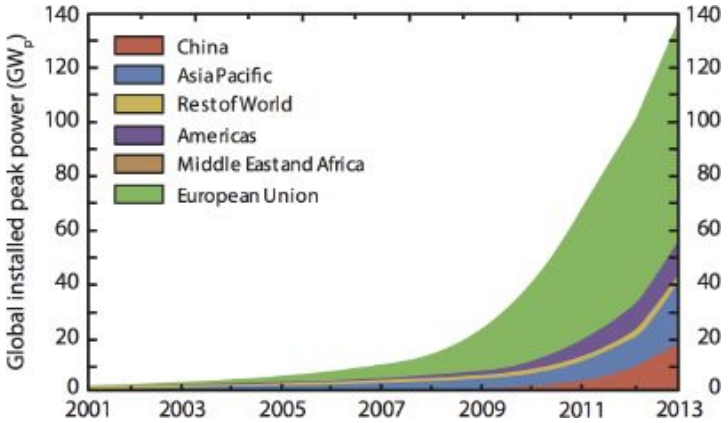


Figure 2.1: Global installed PV capacity [Sme+16]

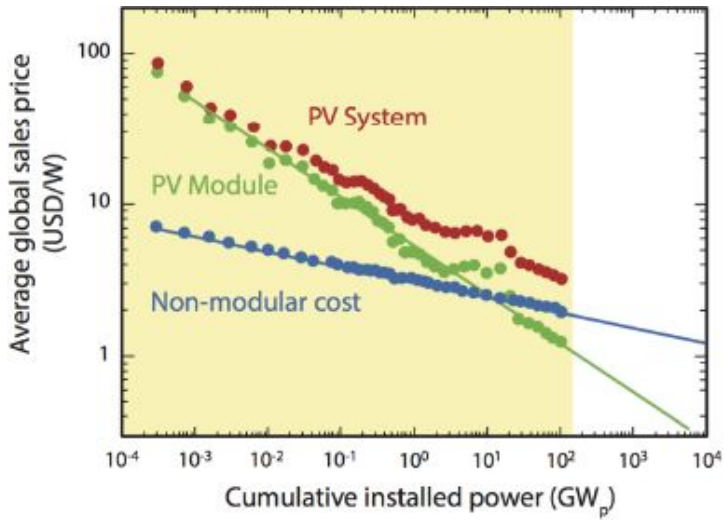


Figure 2.2: Learning curve for PV modules and systems [Sme+16]

around 11.4 [Kop16].

An uncontroversial fact is that the cost of PV systems is falling. Figure 2.2 shows the learning curve of the technology.

The PV effect

Even though PV systems comprise a whole host of technologies, including power electronics and energy storage, the main stage is occupied by PV cells and modules. The working principle of solar cells is based on the photovoltaic effect, i.e. the generation of a potential difference at the junction of two different materials in response to electromagnetic radiation. The photovoltaic effect is closely related to the photoelectric effect, where electrons are emitted from a material that has absorbed light with a frequency above a material-dependent threshold frequency.

The photovoltaic effect can be divided intro three basic processes [Sme+16]:

1. *Generation of charge carriers due to the absorption of photons in the materials that form a junction*: since electrons can only occupy specific energy levels, only photons with a certain amount of energy are absorbed. The absorption of a photon leads to the creation of an electron-hole pair.

2. *Subsequent separation of the photo-generated charge carriers in the junction*: usually, the electron-hole pair will recombine. If one wants to use the energy stored in the electron-hole pair for performing work in an external circuit, semipermeable membranes must be present on both sides of the absorber, such that electrons can only flow out through one membrane and holes can only flow out through the other membrane. In most solar cells, these membranes are formed by n- and p-type materials. A solar cell has to be designed such that the electrons and holes can reach the membranes before they recombine, i.e. the time it requires the charge carriers to reach the membranes must be shorter than their lifetime. This requirement limits the thickness of the absorber.

3. *Collection of the photo-generated charge carriers at the terminals of the junction*:

finally, the charge carriers are extracted from the solar cells with electrical contacts so that they can perform work in an external circuit.

Generations of PV cells

Solar cell technologies are traditionally divided into three generations [Mad]. First generation solar cells are mainly based on silicon wafers and typically demonstrate a performance about 15-20%. These types of solar cells dominate the market and are mainly those seen on rooftops. The benefits of this solar cell technology lie in their good performance, as well as their high stability. However, they are rigid and require a lot of energy in production.

The second generation solar cells are based on amorphous silicon, CIGS and CdTe, where the typical performance is 10 - 15%. Since the second generation solar cells avoid use of silicon wafers and have a lower material consumption it has been possible to reduce production costs of these types of solar cells compared to the first generation. The second generation solar cells can also be produced so they are flexible to some degree. However, as the production of second generation solar cells still include vacuum processes and high temperature treatments, there is still a large energy consumption associated with the production of these solar cells. Further, the second generation solar cells are based on scarce elements and this is a limiting factor in the price.

Third generation solar cells uses organic materials such as small molecules or polymers. Thus, polymer solar cells are a sub category of organic solar cells. The third generation also covers expensive high performance experimental multi-junction solar cells which hold the world record in solar cell performance. This type has only to some extent a commercial application because of the very high production price. A new class of thin film solar cells currently under investigation are perovskite solar cells and show huge potential with record efficiencies beyond 20% on very small area. Polymer solar cells or plastic solar cells, on the other hand, offer several advantages such as a simple, quick and inexpensive large-scale production and use of materials that are readily available and potentially inexpensive. Polymer solar cells can be fabricated with well-known industrial roll-to-roll (R2R) technologies that can be compared to the printing of newspapers. Although the performance and stability of third generation solar cells is still limited compared to first and second generation solar cells, they have great potential and are already commercialized.

PV cells and modules

PV cells are grouped into solar modules, which are clustered into solar panels. A group solar panels is called a PV array. Together with the rest of the components needed to convert solar radiation into useful and safe electrical energy, they form the PV system, as depicted in Figure 2.3.

The connection of cells to form modules, of modules to form panels and panels to form arrays can be made in series or parallel, or a mixture of both, at the different levels. This interconnection is made in a way that provides the required voltage and current levels for the final PV array.

A group of elements connected in series is also called a *string*. The total current in a string of solar cells is equal to the smallest current generated by one single solar cell, but their voltages add up. On the other hand, if cells are connected in parallel, the voltage is the same over all solar cells, while the currents of the solar cells add up.

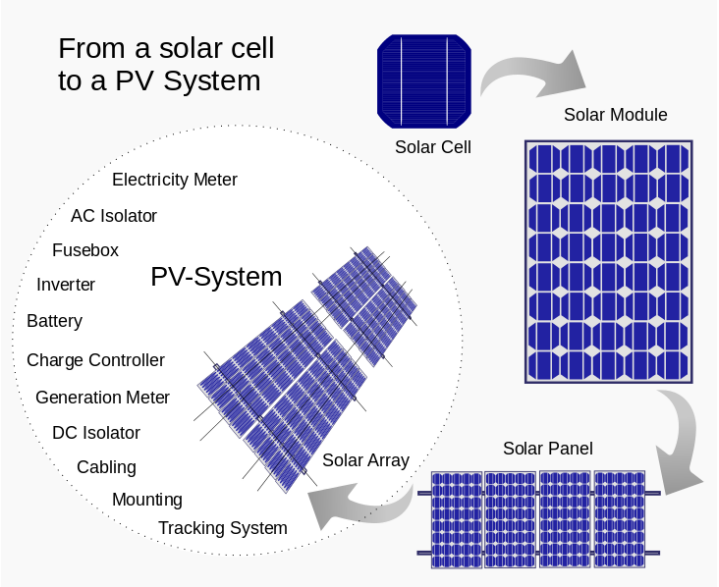


Figure 2.3: From a solar cell to a PV system [Rfa14]

Modern PV modules often contain 60, 72 or even 96 solar cells that are usually all connected in series in order to minimize resistive losses and to enable high voltages that are required for an efficient operation of the inverter [Sme+16].

PV modules have so-called bypass diodes integrated. These diodes are necessary, because in real-life conditions, PV modules can be partially shaded. The shade can be from an object nearby, like a tree, a chimney or a neighbouring building. It also can be caused by a leaf that has fallen onto the module. Partial shading can have significant consequences for the output of the solar module. As mentioned above, in a string of cells, the current is limited by the cell that generates the lowest current; a shaded cell thus dictates the maximum current flowing through the module.

Partial shading and bypass diode

[Sme+16] provides a very good example of this, as shown in Figure 2.4. In this case, 6 solar cells are connected in a string and one of them is shaded. The five unshaded solar cells act like a reverse bias source on the shaded solar cell, which can be graphically represented by reflecting their I-V curve through the $V = 0$ axis (see dashed line in Figure 2.4 (b)). Hence, the shaded solar cell is operated at the intersection of its I-V curve and the reflected curve. As this operating point is in its reverse-bias area, it does not generate energy, but starts to dissipate energy and heats up. The temperature can increase to such a critical level that the encapsulation material cracks, or other materials wear out. Further, high temperatures generally lead to a decrease of the PV output. In addition, a large reverse bias applied to the cell may induce junction breakdown, which can potentially damage the cell.

These problems occurring from partial shading can be prevented by including bypass diodes in the module, as illustrated in Figure 2.4 (c). If no cell is shaded, no current is flowing through the bypass diodes. However, if one cell is (partially) shaded, the bypass diode starts to pass current through because of the biasing from the other cells. As a

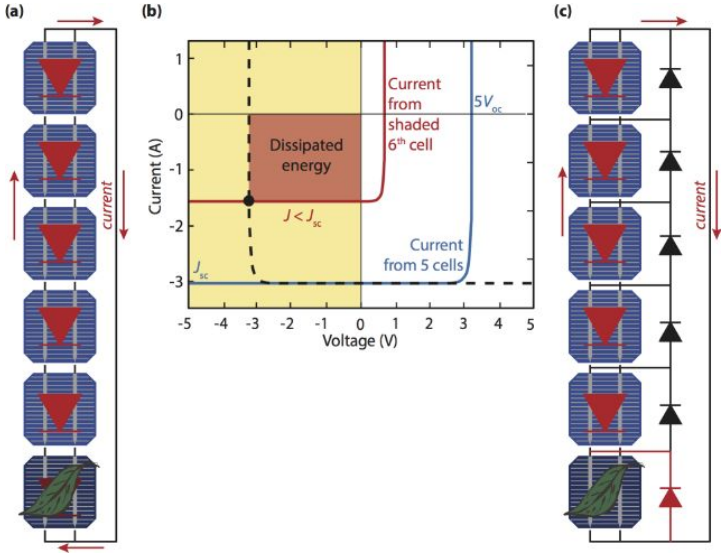


Figure 2.4: Bypass diodes and shading in PV modules [Sme+16]

result, current can flow around the shaded cell and the module can still produce the current equal to that of an unshaded single solar cell.

In real PV modules, not every solar cell is equipped with a bypass diode, but groups of cells share one diode. For example, a module of 60 cells, connected in series forming one string, can contain three bypass diodes, where each diode is shared by a group of 20 cells.

[Sme+16] also provides a great overview of PV systems. PV systems can be small and very simple, consisting of just a PV module and load, as in the direct powering of a water pump motor which only needs to operate when the Sun shines. On the other hand, PV systems can also be built as large power plants with a peak power of several MW; these are connected to the electricity grid. Many systems are placed on residential homes. When a whole house needs to be powered and is not connected to the electricity grid, the PV system must be operational day and night. It may also have to feed both AC and DC loads, have reserve power, and may even include a backup generator.

PV system categories

Depending on the configuration, [Sme+16] establishes the following three categories por PV systems:

- *Stand-alone systems*: also called *off-grid*, rely on solar power only. They can consist of the PV modules and a load only or they can include batteries for energy storage. In that case, they also typically include a charge controller. They can also include an inverter if the system needs to power Alternating Current (AC) loads.

- *Grid-connected systems*: these are connected to the grid through inverters. Depending on the size, the systems in this category could range between a residential rooftop system and a big PV power plant. They don't require batteries since they interface with the grid, but they increasingly do include energy storage for its benefits.

- *Hybrid systems*: which combine PV modules with a complementary method of electricity generation such as a diesel, gas or wind generator.

PV system components

Although the solar panels are the heart of a PV system, many other components are required for a working system, as we discussed very briefly above. Together, these components are called the Balance of System (BOS). Which components are required depends on whether the system is connected to the electricity grid or whether it is designed as a stand-alone system. The most important components belonging to the BOS are [Sme+16]:

- A *mounting structure* in order to place the modules. This mounting structure is sometimes static and sometimes includes moving elements so that the panels can track the movement of the Sun.

- *Cables* to connect the different components of the PV system with each other and to the electrical load. Thicker cables minimize resistive losses but increase the cost.

- *Energy storage* is not technically required but might be needed in order to increase the reliability of the power supply. It's increasingly present in modern PV systems and usually in the form of batteries.

- *Power converters* including Direct Current (DC)-DC converters to regulate the voltage output of the PV array as well as DC-AC converters, also known as inverters, to interface with the grid or to feed AC loads.

- *Charge controllers* that are used in stand-alone systems to control charging and often also discharging of the battery. They prevent the batteries from being overcharged and also from being discharged via the PV array during night. High end charge controllers also contain DC-DC converters together with a Maximum Power Point Tracker (MPPT) in order to make the PV voltage and current independent from the battery voltage and current.

2.3 Modelling and simulation

M&S is the discipline of developing models of systems and using those models to analyse and study those systems through the computation of key features. The process of computing these features is called simulation. A model is a simplified representation of a system, focused on the aspects of the system that are of interest. In the context of this thesis, the models will be of mathematical nature and focused on capturing the evolution of the system with time. The simulation of these models will consist in the numerical solution of the system of differential equations that arise with this modelling.

Brief history of continuous-time modelling

[ÅEM98] provides an overview of the history of continuous-time modelling and simulation. The first simulations, at the beginning of the 20th century, were analog. The idea was to model a system in terms of ordinary differential equations and then make a physical

device that obeyed the equations. The physical system was initialized with proper initial values and its development over time then mimicked the differential equations.

Later in the century, with the advent of digital computers, digital simulation was made possible. When this happened, it was natural that the first efforts emulated the systems created for analog simulation. Eventually, this led to languages and applications based on this paradigm of which Simulink is a current prime example.

The disadvantage of this paradigm is that it requires the manual derivation of explicit state models, an Ordinary Differential Equations (ODEs) description of a system, whereas the natural models for dynamical systems are Differential Algebraic Equations (DAEs), i.e. a mixture of differential and algebraic equations. This manipulation also means that it is cumbersome to build physics based model libraries in these block diagram languages. A general solution to this problem required a paradigm shift.

To address this shortcoming and improve usability, many domain-specific languages and tools were created. These focus on a specific domain like electrical or mechanical systems and provide a better user experience by constraining things in this way.

Another solution that is more recent is physical modelling languages and tools. A typical procedure for physical modeling is to cut a system into subsystems and to account for the behavior at the interfaces. Each subsystem is modelled by balances of mass, energy and momentum and material equations. The complete model is obtained by combining the descriptions of the subsystems and the interfaces. A model is considered as a constraint between system variables. This approach leads naturally to DAE descriptions and is very convenient for building reusable model libraries.

Modelica

Modelica is one of these languages and has gained wide support and adoption in the last decade. It is intended for modelling within many application domains such as electrical circuits, multi-body systems, drive trains, hydraulics, thermodynamical systems, and chemical processes etc. It supports several formalisms: ODEs, DAEs, bond graphs, finite state automata, and Petri nets etc. Modelica is intended to serve as a standard format so that models arising in different domains can be exchanged between tools and users.

M&S tools

The variety and scope of modelling languages and tools is immense. Several reviews can be found in the literature. For example, [MT16] presents a review of tools for modeling electric vehicle energy requirements and their impact on power distribution networks, [All+15] presents a review of modelling approaches and tools for the simulation of district-scale energy systems, and [SC14] presents a review of software tools for hybrid renewable energy systems.

2.4 Related Modelica libraries

In order to keep the scope of this text manageable, this section will just go over other Modelica libraries already developed, relevant to the area of PV systems.

PhotoVoltaics

The library most close to the vision of this library is the `PhotoVoltaics` library [Kra17]. As of this writing, it's under active development and provides models for PV cells, modules and arrays, Modelica record classes with commercial values for quick parametrization of PV panel models as well as a couple of simple converter models. It

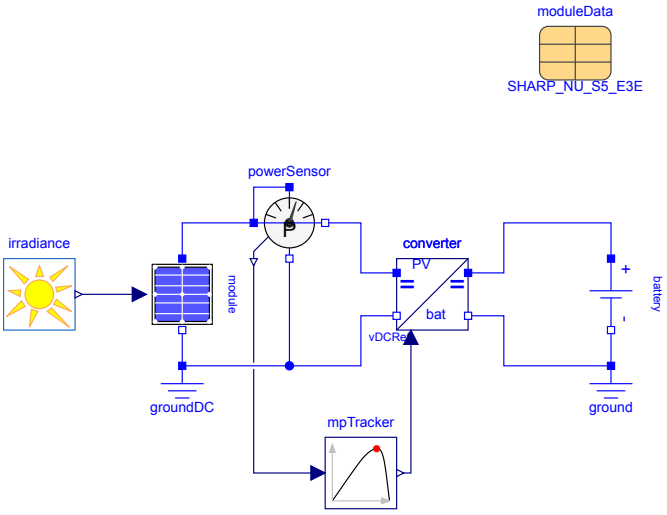


Figure 2.5: Example system included in `PhotoVoltaics`

also enables modelling of partially shadowed cells. Some of the blocks lack documentation, but the documentation that is there is good and it has a healthy examples package.

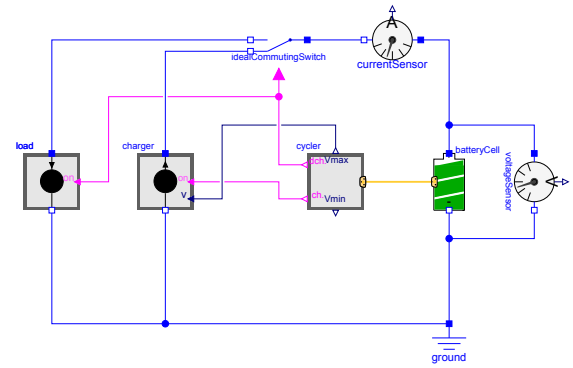
In several ways, `PhotoVoltaics` is superior to `PVSystems`. It includes a similar set of models: PV cells, modules and arrays, power converters, battery energy storage and some relevant control blocks. Figure 2.5 presents one of the examples included in this library. A similar system could be constructed with `PVSystems`.

The basic modelling of PV cells, modules and arrays is based on the same 1-diode model, but the approach used in `PhotoVoltaics` relies more heavily on component-based modelling. One advantage of this approach is that it reuses blocks and electrical models from Modelica Standard Library (MSL). This provides, out of the box, a conditional heat port that is used to model the thermal aspect of PV devices. An additional nice feature is that irradiance is also implemented conditionally - the user can select a constant irradiance value and disable the irradiance input.

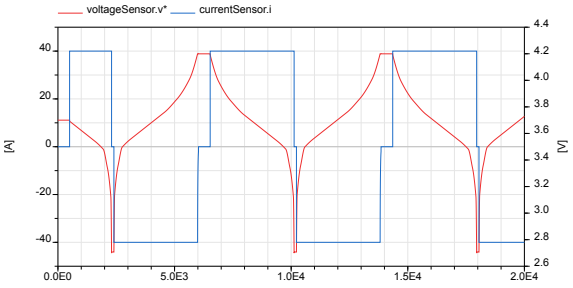
Irradiance is also best modelled in `PhotoVoltaics`, including blocks that model the change in irradiance with the changing position of the Sun. No such model is included with `PVSystems`.

Regarding models for power converters and control blocks, the library is a bit more lacking. This can't really be held against it, since the library aims at modelling PV components and these could seem reasonably out of its scope. The AC components are limited to quasi-static characteristics, ignoring transients and switching components. These are better modelled in `PVSystems` because its scope was conceived with power electronics designers in mind. The control blocks included with `PhotoVoltaics` are limited to an MPPT controller and some ancillary blocks, whereas `PVSystems` includes a more comprehensive collection of blocks.

Finally, `PhotoVoltaics` also includes 39 records of parameter values for commercially available PV modules. This records package constitutes an easy addition to the library



(a)



(b)

Figure 2.6: Example system included in `ElectricalEnergyStorage`: (a) diagram and (b) simulation results.

and one that would likely provide much value to users. As mentioned in Section 7.2, such a package is considered a worthy addition to `PVSystems` in the future.

ElectricalEnergyStorage

`ElectricalEnergyStorage` [Ein+11; Mod17a] is free library that contains models with different complexity for simulation of electric energy storage devices like batteries (single cells as well as stacks) interacting with battery management systems, loads and charging devices.

The package with the battery models is divided into `Cells` and `Stacks`, which are just arrays of cells connected in series and parallel. The cell models can be either a simple cell model with just a static ohmic impedance or a more complex cell model with basic self discharge, a variable ohmic impedance and a variable number of variable RC elements. They are also equipped with a bus interface based on expandable connectors. A `CellBus` is available for simple monitoring of a battery and a `ControlBus` is available for a more sophisticated interface with a battery management sytem.

The `CellRecords` are included for appropriate grouping of cell parameters and additional models and blocks exist in the categories of sensors, loads and chargers, battery management. Figure 2.6 presents one of the examples included with the library and simulation results.

PowerConverters

The MSL includes many of the elements needed to create models similar to those that can be built with `PVSystems`. The latest version, 3.2.2 at the time of this writing,

includes models for power converters, a comprehensive set of basic electric and electronic components, as well as multiphase, quasi-stationary and electrical machine models, all under the `Electrical` package.

Of special interest is the `PowerConverters` package, as it compares with the power electronics models included in `PVSystems`. It provides 16 different converter models, constructed from `IdealDiode`, `IdealThyristor` and `IdealGTOThyristor`. The last one used as a switching transistor model. All of these blocks have an `Ron`, `Goff` and `Vknee` parameters, modelling conduction losses, and the heat ports of components are propagated to the higher levels.

On the other hand, this package doesn't provide averaged version of the converter models, which `PVSystems` does. Additionally, the structure and architecture of the package could be improved. Most of the converters follow a very similar topology. In fact, all converters in the library, except for the centre-tapped rectifiers, are constructed with an H-bridge topology. It seems that grouping and reusing models around topology instead of conversion function would make a more compact, simple and user-friendly package.

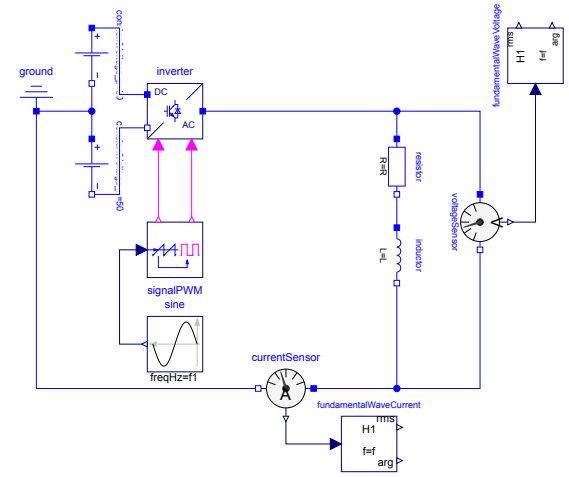
The control blocks included are limited, but that could be due to intentional constraint of the scope of the package. The interfaces could also be simplified, though they do provide both regular and multiphase versions of electrical ports. Figure 2.7 presents one of the examples included in with the library and simulation results. A similar example is included in `PVSystems` and discussed in Section 6.3.

PowerSystems

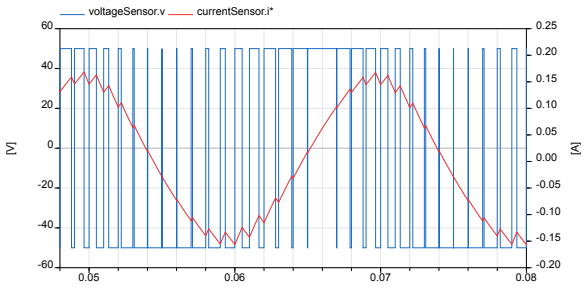
`PowerSystems` [FW14; FW16] is a free (standard conform) library that is intended to model electrical power systems at different levels of detail both in transient and steady-state mode. It's mainly aimed at modelling of grid level power systems and it's scope is that of bigger systems and longer simulation times than `PVSystems`. It's a very comprehensive library and features support for models in different reference systems (i.e. abc, dq0).

EDrives

The `EDrives` library [HK14] presents models for electric motor drives. This provides models for inverters at three levels: quasi static (neglecting electrical transients), averaging (neglecting switching effects) and switching, which is similar to the goal of `PVSystems`. It's focus is on the application of this model for driving of electric machines. This library is only available commercially.



(a)



(b)

Figure 2.7: Example system included in `PowerConverters`: (a) diagram and (b) simulation results.

3 Photovoltaic systems modelling

This chapter will cover the fundamental concepts regarding the modelling of the components outlined in Chapter 2. For each of these elements, a corresponding subsection will go over the equations to model it.

3.1 PV source

The term PV source is used here to refer to the device where the PV effect is taking place, from the PV cell to the PV array. As it will be shown, a single model is sufficient to model any of these devices, hence the use of this umbrella term.

One of the traditional ways to model a PV source defined in this way is to use the equivalent circuit presented in Figure 3.1. This is known as the single-diode circuit model of the solar cell. A two-diode model also exists, but the the single-diode version provides a decent approximation and is simpler [VGF09].

The relationship between the voltages and currents in Figure 3.1 can be established in the form of the following equation:

$$I = I_{pv} - I_0 \left[\exp\left(\frac{V + R_s I}{V_t\, a} \right) - 1 \right] - \frac{V + R_s I}{R_p} \tag{3.1}$$

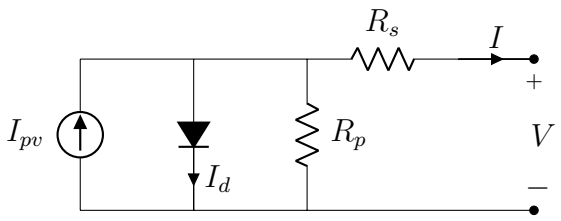


Figure 3.1: Equivalent circuit of a PV source

In this equation, each of the terms take the following form:

$$I_{pv} = \left(I_{pv,n} + K_I\,\Delta_T\right)\frac{G}{G_n} \tag{3.2a}$$

$$I_{pv,n} = \frac{R_p + R_s}{R_p}\,I_{sc,n} \tag{3.2b}$$

$$I_0 = \frac{I_{sc,n} + K_I\,\Delta_T}{\exp\left(\frac{V_{oc,n}+K_V\,\Delta_T}{a\,V_t}\right) - 1} \tag{3.2c}$$

$$V_t = \frac{k\,T}{q}\,N_s \tag{3.2d}$$

$$\Delta_T = T - T_n \tag{3.2e}$$

Where the values of $I_{sc,n}$, K_I, K_V and $V_{oc,n}$ can be established from the data-sheet, as presented in Table 3.1. Additionally, the values of the following terms are known: $G_n = 1000\,\mathrm{W/m^2}$ and $T_n = 298.15\,\mathrm{K}$ are the Standard Testing Conditions (STC) values of solar irradiation and temperature, respectively; $k = 1.380\,650\,3 \times 10^{-23}\,\mathrm{J/K}$ is the Boltzmann constant and $q = 1.602\,176\,46 \times 10^{-19}\,\mathrm{C}$ is the electric charge of the electron; N_s and N_p are the number of cells in series and in parallel, respectively; finally, G, T are the actual solar irradiation and ambient temperature, normally considered inputs to the model, and I and V are the actual panel/array current and voltage.

Table 3.1: Typical PV panel data-sheet parameters and values for Kyocera KC200GT [Kyo17]

Parameter	Symbol	Value
Nominal open-circuit voltage	$V_{oc,n}$	32.9 V
Nominal short-circuit current	$I_{sc,n}$	8.21 A
Voltage at MPP	V_{mp}	26.3 V
Current at MPP	I_{mp}	7.61 A
Open-circuit voltage/temperature coefficient	K_V	$-0.1230\,\mathrm{V/K}$
Short-circuit current/temperature coefficient	K_I	$0.0032\,\mathrm{A/K}$
Maximum experimental peak output power	$P_{max,e}$	200.143 W

After all this, we are still left with three unresolved symbols: a is the *diode ideality factor*, is determined experimentally and is not normally supplied in data-sheets, and R_s and R_p are the series and parallel resistances. These are also not supplied and need to be figured out from the available data-sheet information.

This is where [VGF09] provides a convenient solution, proposing the algorithm displayed in Figure 3.2 to reach values of R_s and R_p that provide a nice fit of the model to the data-sheet data by guaranteeing that the fit goes through the salient points of the I-V curve $(0, I_{sc,n})$, (V_{mp}, I_{mp}) and $(V_{oc,n}, 0)$, as well as coinciding with the maximum power point of the P-V curve $(V_{mp}, P_{max,e})$. To kick the algorithm off, the starting value for R_p

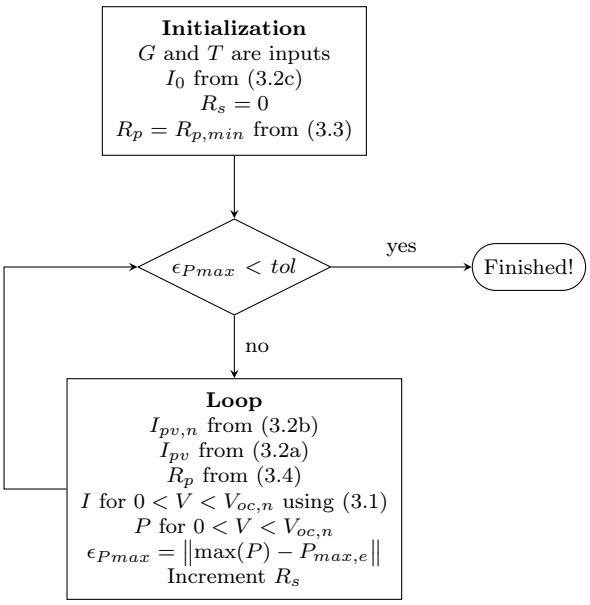


Figure 3.2: Algorithm to determine R_s and R_p

is specified by:

$$R_{p,min} = \frac{V_{mp}}{I_{sc,n} - I_{mp}} - \frac{V_{oc,n} - V_{mp}}{I_{mp}} \tag{3.3}$$

Subsequent value of R_p are computed by equating the Maximum Power Point (MPP) predicted by the model, $P_{max,m}$, with the MPP provided in the data-sheet, $P_{max,e}$. An expression for R_p can be achieved through algebraic manipulation of (3.1), reaching the following expression:

$$R_p = \frac{V_{mp} + R_s\, I_{mp}}{V_{mp}\, I_{mp} - V_{mp}\, I_{d,mp} - P_{max,e}} \tag{3.4}$$

where $I_{d,mp}$ is just the diode current when in the MPP,

$$I_{d,mp} = I_0 \left[\exp\left(\frac{V_{mp} + I_{mp}\, R_s}{N_s\, V_t\, a} \right) - 1 \right] \tag{3.5}$$

Applying this algorithm to the Kyocera KC200GT referenced in [VGF09], a very good fit is obtained with the values of $R_s = 0.221\,\Omega$ and $R_p = 415.405\,\Omega$.

3.2 Power electronics

The discipline of power electronics comprises the study of the technology and devices aimed at processing power. These devices are mainly made up of switches, magnetics like coils and transformers, and capacitors. This section will focus on the switching devices and the discussion will be based upon the ideas presented in [EM01] and [EMA16].

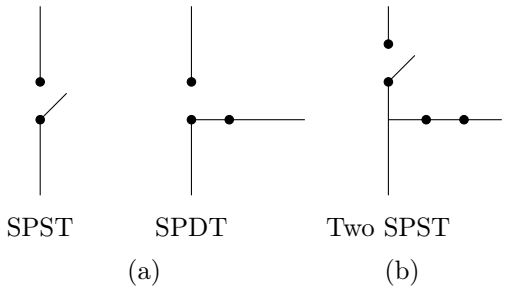


Figure 3.3: Poles and throws: (a) two types of switches and (b) SPST implementation of a SPDT switch.

3.2.1 Switch realization

In the applications explored in this text, all the switches that will appear will be, in abstract terms, of two types: Single-pole single-throw (SPST) and Single-pole double-throw (SPDT). Figure 3.3a depicts both types of switches. Since SPDT switches can be implemented with a pair of coordinated SPST switches, as shown in Figure 3.3b, these last ones will be the single subject of the following discussion.

SPST switches can be further classified depending on their current and voltage characteristics into four groups: single-quadrant, current-bidirectional two-quadrant, voltage-bidirectional two-quadrant and four-quadrant switches. An additional distinction will be made between *active* and *passive* switches. In the former, the switch state is controlled exclusively by a third terminal (control terminal), in the latter, the switch state is controlled by the applied current and/or voltage at the switch terminals. This taxonomy of switches is not comprehensive but will suffice for the purpose of this work.

In practical terms, these switches are implemented with semiconductor devices. Again, from the point of view of this discussion, we will consider only a few of these devices. The goal of this work is not to provide a comprehensive library of all of the semiconductor devices but to provide as few generic models as possible to capture their relevant features, and to make these models configurable in a way that accommodates the modelling of this wide range of possible devices. Figure 3.4 displays some examples of switches and their realizations with semiconductor devices.

3.2.2 Switch network concept

Modelling these switches will be done by constructing assemblies of semiconductor devices models from MSL, as explained in Section 4.2.

Additionally, average switching models will be provided for a general switch network that will enable the construction of different power converters. The advantage of this approach is that, in the cases were the goal of the simulation is to provide validation for a control algorithm, for example, faster simulations can be performed by creating a model that ignores everything happening at high frequencies like the switching frequencies. Further details will be presented in Section 4.2.

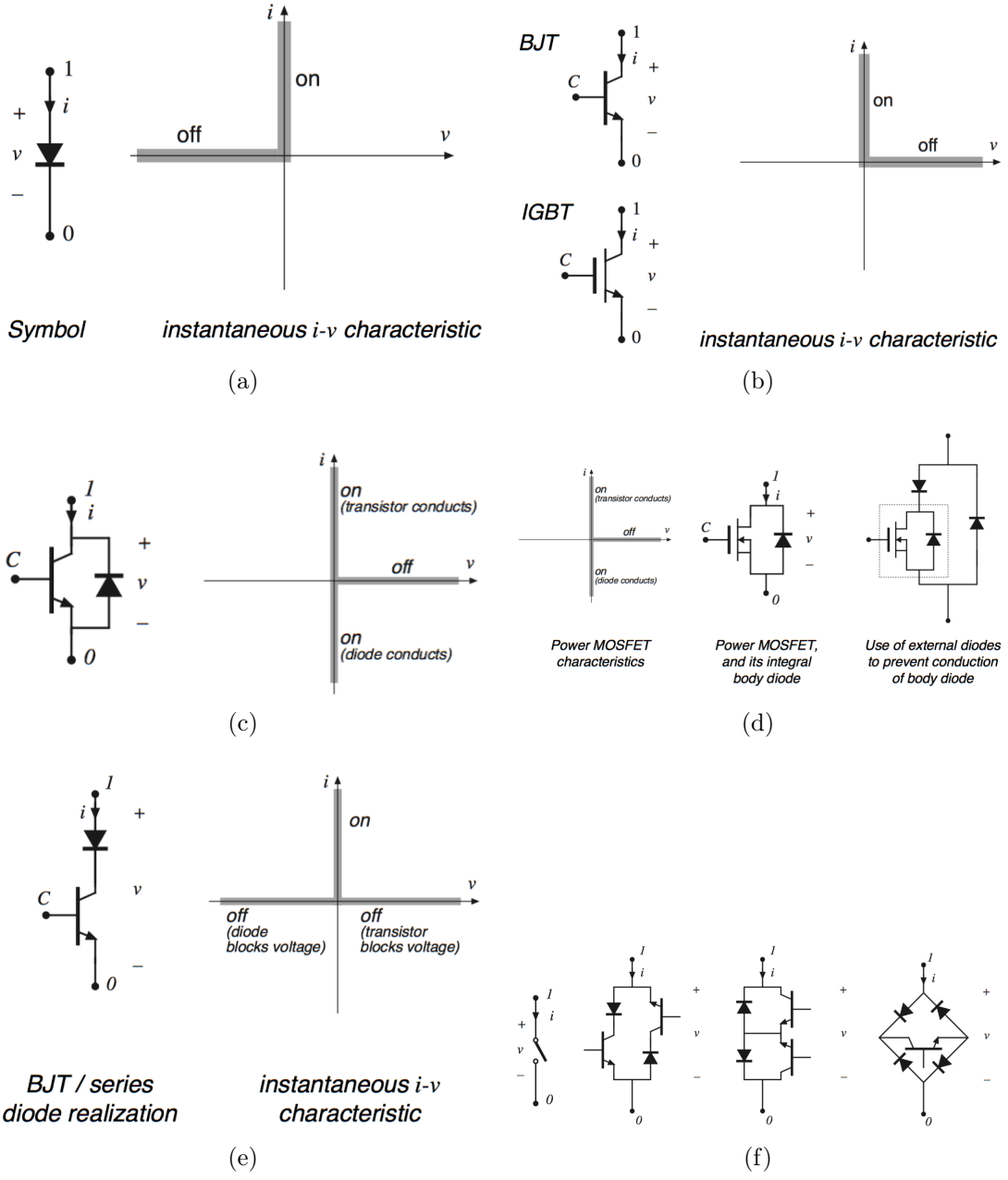


Figure 3.4: Examples of single-quadrant switches [EM01]

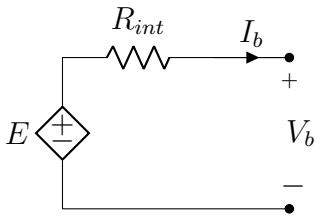


Figure 3.5: Equivalent circuit for a battery model

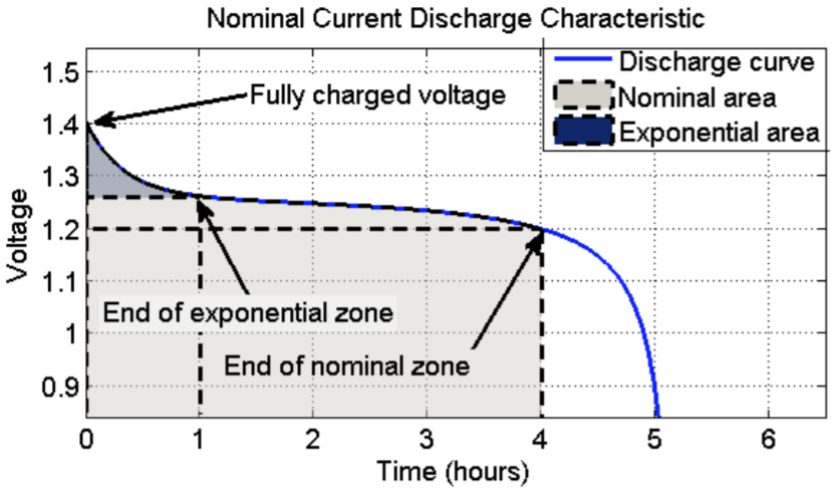


Figure 3.6: Generic battery discharge curve [TDD07]

3.3 Energy storage

Energy storage in PV systems can take many forms. In this work, the focus will stay on Li-ion batteries. Although the models presented here are often adequate to represent other battery chemistries. The following description is based on the work presented in [TDD07].

A decent approximation of a battery for system level studies can be created based on the circuit presented in Figure 3.5, where the value of the controlled voltage source, E, takes the following expression:

$$E = E_0 - K \frac{Q}{Q - i_t} + A\, e^{-B\, i_t} \tag{3.6}$$

where E_0 is the battery constant voltage in V, K is the battery polarization voltage in V, Q is the battery capacity in A h, i_t is the actual depth of discharge also in A h, A is the exponential zone amplitude in V and B is the inverse of the exponential zone equivalent time constant, in $\mathrm{A}^{-1}\,\mathrm{h}^{-1}$.

These parameters are obtained from the battery discharge curve, typically available in manufacturer's datasheets. A generic discharge curve is presented in Figure 3.6.

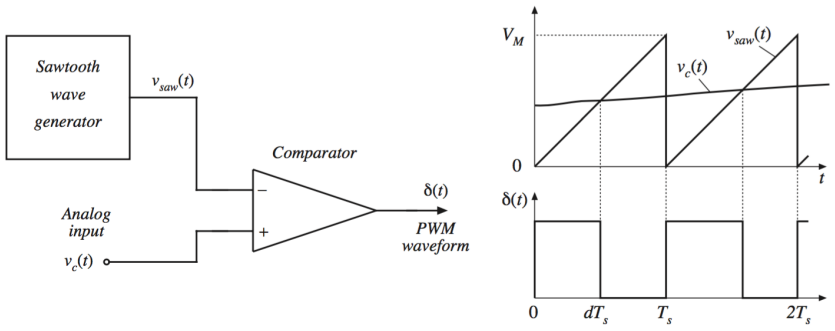


Figure 3.7: PWM generation [EMA16]

3.4 Control blocks

3.4.1 Switching waveforms generation

The switching waveforms are the signals that drive the opening and closing of the power switches. In this library, two switching waveforms generation schemes are considered: Pulse Width Modulation (PWM) and Current Programmed Mode (CPM) modulation. In the first, a control signal is compared with a sawtooth waveform to generate the PWM signal (Figure 3.7).

In the second, the switching signal is generated as shown in Figure 3.8. From exploring the details of this scheme, it is established that an artificial ramp signal more stability and better behaviour across a wider spectrum of values of the control signal.

This artificial ramp can either be added to the measured current (as in Figure 3.8b) or can be subtracted from the control signal (as in Figure 3.8a), to equal effect. Some digital logic is added so that, at the beginning of every switching period, the switching signal is set. When the measured current (plus the artificial ramp) reaches the control signal (minus the artificial ramp), the switching signal is reset.

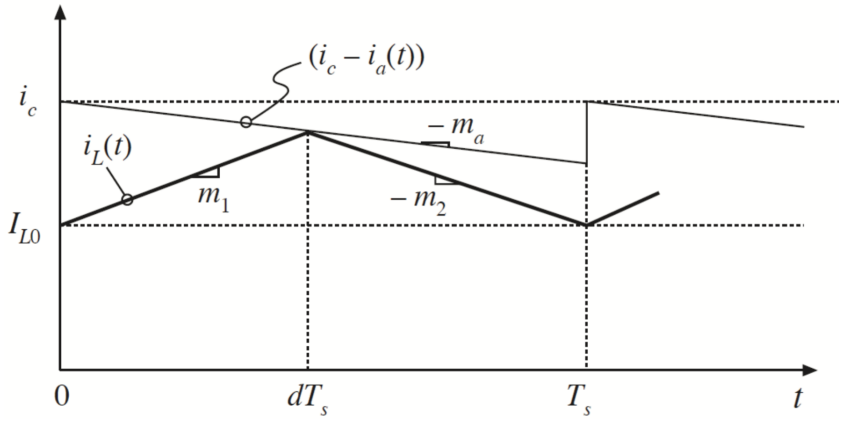
3.4.2 Coordinate transforms

In the control of AC power converters, two coordinate transformations are useful and popular. Power systems are three-phase in many high-power grid related applications.

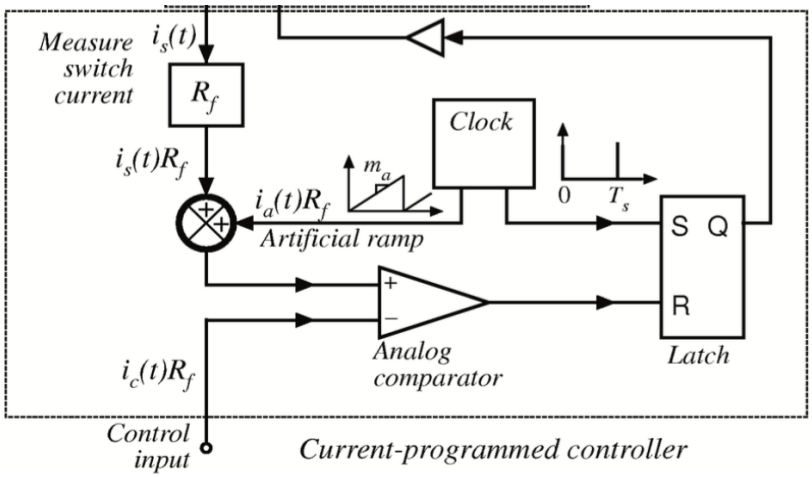
For a three-phase system, the Clarke transform is defined with the following equation [TLR11]:

$$\begin{pmatrix} v_\alpha \\ v_\beta \\ v_0 \end{pmatrix} = \frac{2}{3} \begin{pmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} v_a \\ v_b \\ v_c \end{pmatrix} \tag{3.7}$$

The $\alpha\beta 0$ coordinate system is called the *static reference frame*, because it remains static. The Park transform takes things a bit further by creating a rotating coordinate system. The system rotates at the frequency of the power system, which is why it also receives the name of *synchronous reference frame*.



(a)



(b)

Figure 3.8: CPM generation [EMA16]: (a) waveforms and (b) circuit.

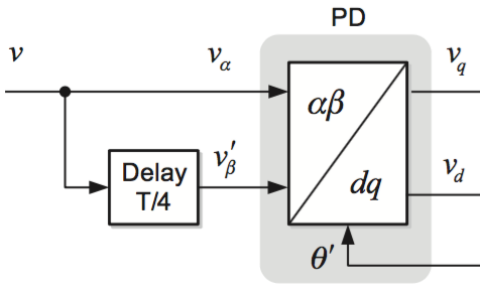


Figure 3.9: $T/4$ delay quadrature signal generator used in the phase detection part of a Phased-Locked Loop (PLL) [TLR11]

The transformation matrix to translate a voltage vector from the $\alpha\beta 0$ stationary reference frame to the $dq0$ synchronous reference frame is given by [TLR11]:

$$\begin{pmatrix} v_d \\ v_q \\ v_0 \end{pmatrix} = \begin{pmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} v_\alpha \\ v_\beta \\ v_0 \end{pmatrix} \tag{3.8}$$

The advantage of the $dq0$ reference frame is that the magnitudes for a balanced steady-state system will be DC quantities instead of AC time-varying quantities.

In the case of single-phase systems, a mathematical trick used to enable the application of synchronous reference frame control is to create a second signal by effecting a 90° shift on the original single-phase voltage or current signal. This is called a $T/4$ quadrature signal generator (Figure 3.9).

3.4.3 Controllers

A popular control strategy for grid-tied PV inverters is based on using Proportional Integral (PI) controllers in the synchronous reference frame, as depicted in Figure 3.10. This diagram presents the complete general controller for this kind of application.

From the DC voltage and current, the MPPT block establishes the DC voltage setpoint, which feeds a PI controller structure that generates the setpoint for the internal current controller on the d axis, i_d. The setpoint on the q axis is typically set to 0 or established by some other means, depending on the grid regulations.

A PLL block is applied to the measured grid voltage to establish the phase, which is then used in the Park and inverse Park transformations. These transformations are applied to the measured AC current before and after the PI controllers. Lastly, the output, which represents the control effort, is scaled and offset to create the appropriate levels for the control circuitry controlling the switching semiconductors.

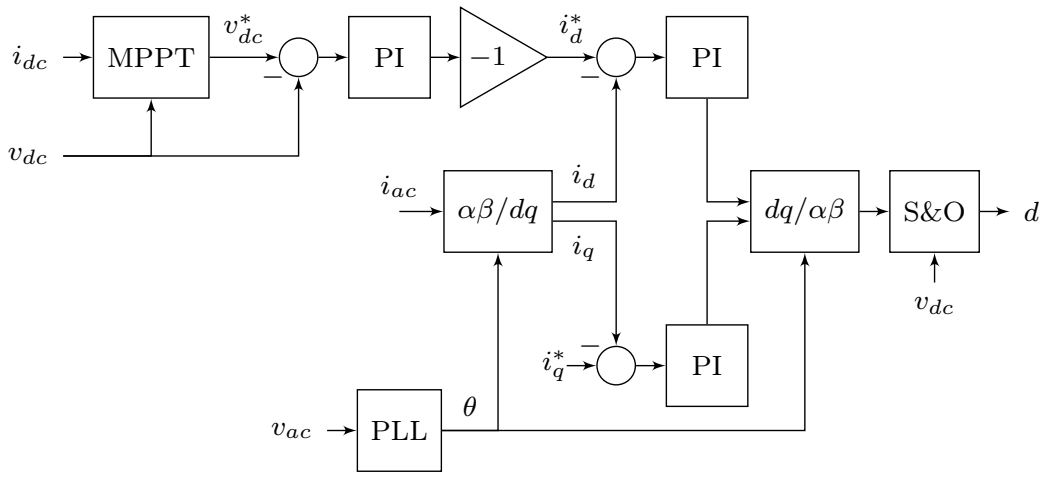


Figure 3.10: General synchronous reference frame inverter controller

4 PVSystems library

4.1 Overview and architecture

This chapter will provide a description of the PVSystems library. Figure 4.1 presents an overview of the library structure and contents, which comprises the following main packages:

- UsersGuide: includes documentation providing an overview of the library, a list of references and the license and contact information.
- Examples: contains two sub-packages, Application, with system models that are aimed at showcasing the use of the library, and Verification, with system models providing some form of unit testing for the component models from the Electrical and Control sub-packages.
- Electrical: contains models of electrical components and subsystems, mainly different variants (switched and averaged) of the switch network concept.
- Control: contains basic control blocks, like PWM, CPM modulator, coordinate transformations, PLL, MPPT controller and a couple of inverter controller assemblies.

For the sake of clarity, the full contents is only shown for Electrical and Control. The Examples package will be explored further in Chapters 5 and 6 since they contain models relevant to the validation and application of the library.

The Modelica language provides a range of possible classes, from records and types to models and packages. With regards to models, a useful classification can be made, as suggested in [Til17]:

- *Components*: models representing atomic components. They typically can't be simulated on their own, they preferably represent only one physical effect and are intended to be included in assemblies to form subsystems or systems.
- *Subsystems*: models created by assembling other models. These models are also not meant for simulation, but as convenient reusable subsystems.
- *Systems*: these are models that completely represent a system and are aimed at simulation. In PVSystems, this kind of models will be included in the Examples package. All other models are component or subsystem models.

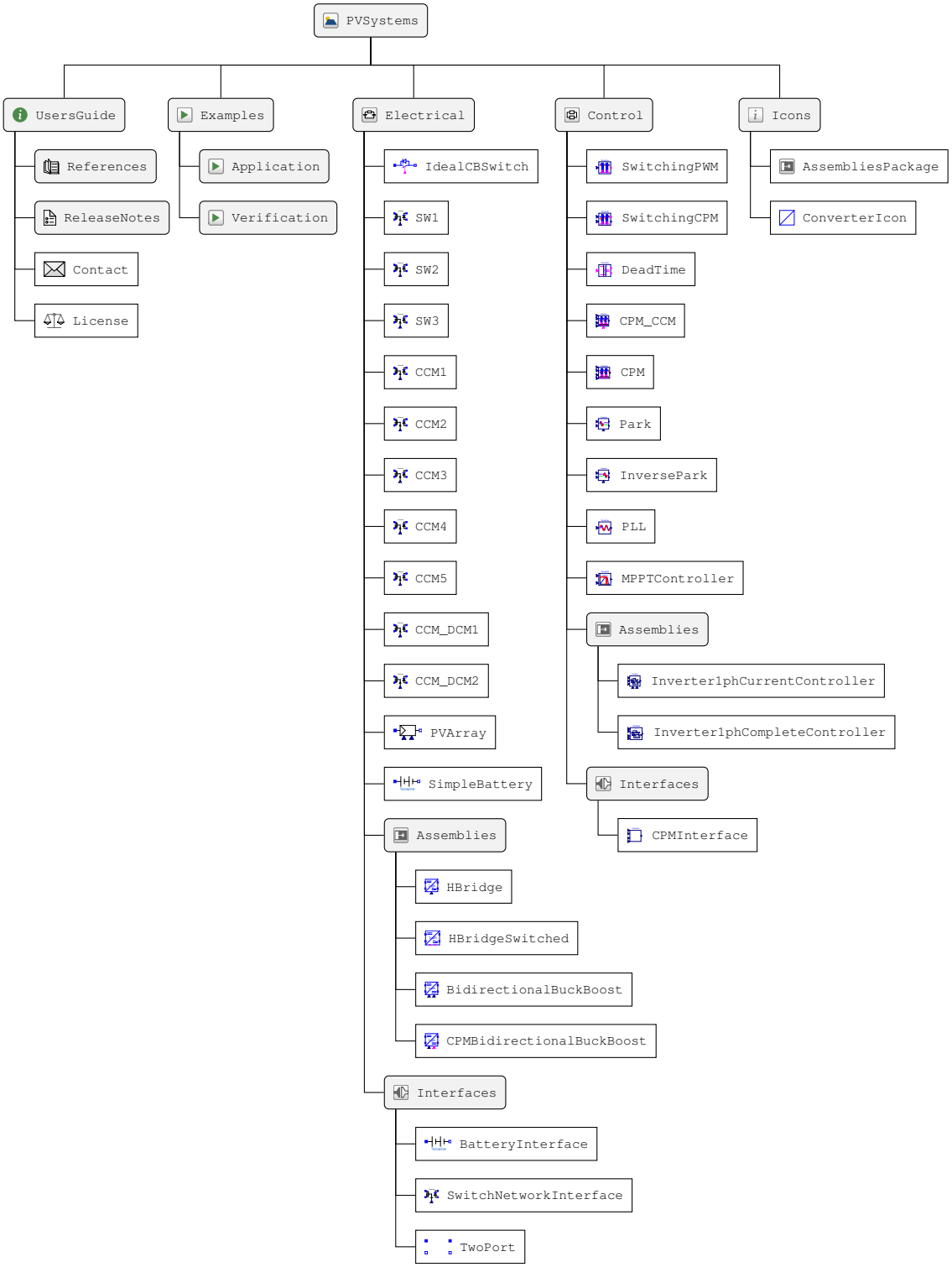


Figure 4.1: Overview of the PVSystems library

This chapter will describe each of the models in the Electrical and Control sub-packages. Listings are included for those models created directly using Modelica code and figures are included for those created using the block diagram in Dymola. The listings included in this chapter have had their annotations removed to make them easier to follow. For a complete version of the listings, see the Source code appendix.

4.2 Electrical models

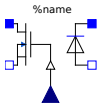
4.2.1 Interfaces

The Interfaces package in Electrical holds three classes. The **BatteryInterface** class inherits from the OnePort interface from the MSL and also provides a battery icon. This class is included to accommodate other battery implementations, apart from the SimpleBattery model included.

Listing 4.1: Electrical/Interfaces/BatteryInterface.mo

```
1 partial model BatteryInterface "Partial model for battery"
2   extends Modelica.Electrical.Analog.Interfaces.OnePort;
3 end BatteryInterface;
```

The **SwitchNetworkInterface** is included with the same intention. All of the switch network models inherit from this class, both the switched versions and the averaged ones. Since the switch network concept can be used to build converters, this results in a very convenient architecture that enables the user to instantiate a converter model and modify it to use any switched or averaged variant right from the diagram. This greatly improves the user experience.



Listing 4.2: Electrical/Interfaces/SwitchNetworkInterface.mo

```
1 partial model SwitchNetworkInterface "Interface for the averaged switch network models"
2   extends TwoPort;
3   parameter Real dmin(final unit = "1") = 1e-3 "Minimum duty cycle";
4   parameter Real dmax(final unit = "1") = 1 "Maximum duty cycle";
5   Modelica.Blocks.Interfaces.RealInput d "Duty cycle";
6 protected
7   Real dsat(final unit = "1") = smooth(0, if d > dmax then dmax else if d < dmin then dmin
        else d) "Saturated duty cycle";
8 end SwitchNetworkInterface;
```

The **TwoPort** class provides a common interface for several two-port components, including the converters in the Assemblies package as well the SwitchNetworkInterface itself. The reason that the equivalent two-port interface included in MSL wasn't used is because that one includes a current conservation equation for each port. When that class is used to build a model through composition using diagram blocks, this results in redundant equations and an overdetermined system. Except for that, the TwoPort class included in this library is equal to the one from the MSL.

Listing 4.3: Electrical/Interfaces/TwoPort.mo

```
1  partial model TwoPort "Common interface for power converters with two ports"
2    Modelica.SIunits.Voltage v1 "Voltage drop over the left port";
3    Modelica.SIunits.Voltage v2 "Voltage drop over the right port";
4    Modelica.SIunits.Current i1 "Current flowing from pos. to neg. pin of the left port";
5    Modelica.SIunits.Current i2 "Current flowing from pos. to neg. pin of the right port";
6    Modelica.Electrical.Analog.Interfaces.PositivePin p1 "Positive pin of the left port (
         potential p1.v > n1.v for positive voltage drop v1)";
7    Modelica.Electrical.Analog.Interfaces.NegativePin n1 "Negative pin of the left port";
8    Modelica.Electrical.Analog.Interfaces.PositivePin p2 "Positive pin of the right port (
         potential p2.v > n2.v for positive voltage drop v2)";
9    Modelica.Electrical.Analog.Interfaces.NegativePin n2 "Negative pin of the right port";
10 equation
11   v1 = p1.v - n1.v;
12   v2 = p2.v - n2.v;
13   i1 = p1.i;
14   i2 = p2.i;
15 end TwoPort;
```

4.2.2 Switching components

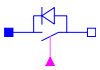
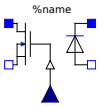


Figure 4.2 presents the four switching components included in `Electrical`. The first one, Figure 4.2a is the diagram of `IdealCBSwitch`, an ideal current-bidirectional switch. This is one of the most basic switch realizations. It's typically used in inverters, which is why it's included in the library.



The other three figures present the diagrams of the three available switching realizations of the switch network concept. They all extend from `SwitchNetworkInterface`, for plug in compatibility with the averaged implementations. 4.2b is the most basic implementation, using an closing switch on port 1 and a diode on port 2. 4.2c provides a synchronous version, with two complementary closing switches with optional dead-time. 4.2d is the most complex version, providing a current-bidirectional switch realization for each port.

4.2.3 Averaged components

The averaged switch network implementations are created directly with Modelica code, codifying the appropriate equations for each version. Table 4.1 lists the different variants and provides a short description of each. For an in-depth review of the process of arriving at those equations, see Section 3.2.2.

The most basic version, **CCM1**, assumes no losses and no transformer, so it ends up in the simple equivalent DC transformer equations. Notice that d_{sat} is just a saturated version of the duty cycle, d, input. The equation to establish d_{sat} is provided in `SwitchNetworkInterface` (Listing 4.2), from which the switch network implemen-

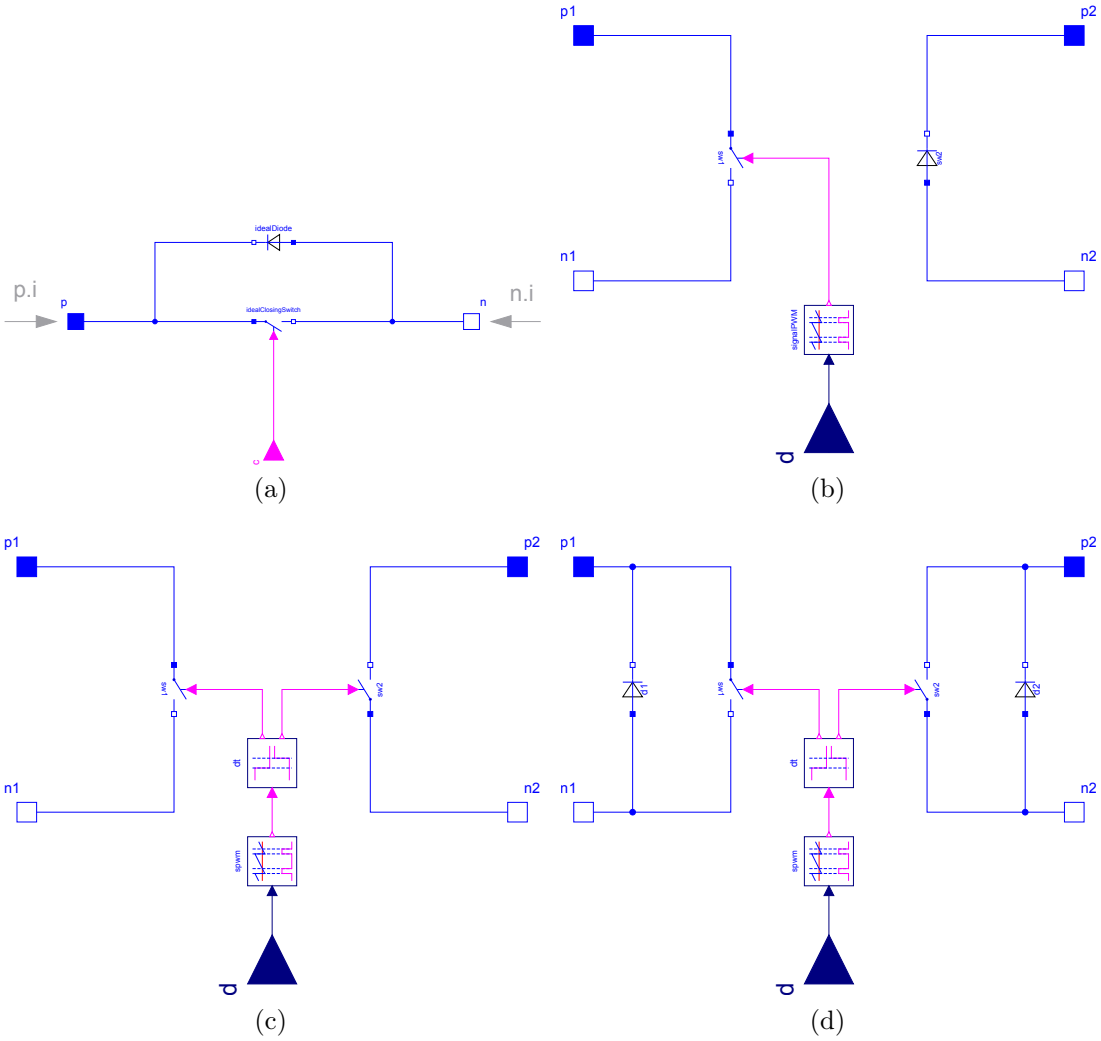


Figure 4.2: Basic switching components in Electrical

Table 4.1: Averaged switch network models [EMA16]

Model	Mode	Parameters	Limitations
CCM1	CCM		Ideal switches, no transformer
CCM2	CCM	R_{on}, V_D, R_D	No switching losses, no transformer
CCM3	CCM	n	Ideal switches
CCM4	CCM	R_{on}, V_D, R_D, n	No switching losses
CCM5	CCM	R_{on}, V_D, Q_r, t_r, f_s	No transformer
CCM_DCM1	CCM or DCM	L, f_s	Ideal switches, no transformer
CCM_DCM2	CCM or DCM	L, f_s, n	Ideal switches

tations extend.

$$v_1 = \frac{1 - d_{sat}}{d_{sat}} v_2 \tag{4.1a}$$

$$-i_2 = \frac{1 - d_{sat}}{d_{sat}} i_1 \tag{4.1b}$$

Listing 4.4: Electrical/CCM1.mo

```
model CCM1 "Average CCM model with no losses"
  extends Interfaces.SwitchNetworkInterface;
equation
  0 = p1.i + n1.i;
  0 = p2.i + n2.i;
  v1 = (1 - dsat) / dsat * v2;
  -i2 = (1 - dsat) / dsat * i1;
end CCM1;
```

The equations for **CCM2** account for conduction losses, with the transistor on resistance, R_{on}, and the diode on resistance, R_D, and forward voltage, V_D.

$$v_1 = \left(\frac{1}{d_{sat}} R_{on} + \frac{1 - d_{sat}}{d_{sat}^2} R_D \right) i_1 + \frac{1 - d_{sat}}{d_{sat}} (v_2 + V_D) \tag{4.2a}$$

$$-i_2 = \frac{1 - d_{sat}}{d_{sat}} i_1 \tag{4.2b}$$

Listing 4.5: Electrical/CCM2.mo

```
model CCM2 "Average CCM model with conduction losses"
  extends Interfaces.SwitchNetworkInterface;
  parameter Modelica.SIunits.Resistance Ron = 0 "Transistor on resistance";
  parameter Modelica.SIunits.Resistance RD = 0 "Diode on resistance";
  parameter Modelica.SIunits.Voltage VD = 0 "Diode forward voltage drop";
```

```
equation
  0 = p1.i + n1.i;
  0 = p2.i + n2.i;
  v1 = i1 * (Ron / dsat + (1 - dsat) * RD / dsat ^ 2) + (1 - dsat) / dsat * (v2 + VD);
  -i2 = i1 * (1 - dsat) / dsat;
end CCM2;
```

CCM3 provides the same lossless equations but with an added optional transformer ratio, n, for converters that include a transformer.

$$v_1 = \frac{1 - d_{sat}}{n\, d_{sat}}\, v_2 \tag{4.3a}$$

$$-i_2 = \frac{1 - d_{sat}}{n\, d_{sat}}\, i_1 \tag{4.3b}$$

Listing 4.6: Electrical/CCM3.mo

```
model CCM3 "Average CCM model with no losses and tranformer"
  extends Interfaces.SwitchNetworkInterface;
  parameter Real n(final unit = "1") = 1 "Transformer turns ratio 1:n (primary:secondary)"
      ;
equation
  0 = p1.i + n1.i;
  0 = p2.i + n2.i;
  v1 = (1 - dsat) * v2 / dsat / n;
  -i2 = (1 - dsat) * i1 / dsat / n;
end CCM3;
```

In **CCM4**, both the transformer ratio and the conduction losses are included.

$$v_1 = \left(\frac{1}{d_{sat}}\, R_{on} + \frac{1 - d_{sat}}{n^2\, d_{sat}^2}\, R_D\right) i_1 + \frac{1 - d_{sat}}{n\, d_{sat}}\, (v_2 + V_D) \tag{4.4a}$$

$$-i_2 = \frac{1 - d_{sat}}{n\, d_{sat}}\, i_1 \tag{4.4b}$$

Listing 4.7: Electrical/CCM4.mo

```
model CCM4 "Average CCM model with conduction losses and tranformer"
  extends Interfaces.SwitchNetworkInterface;
  parameter Modelica.SIunits.Resistance Ron = 0 "Transistor on resistance";
  parameter Modelica.SIunits.Resistance RD = 0 "Diode on resistance";
  parameter Modelica.SIunits.Voltage VD = 0 "Diode forward voltage drop";
  parameter Real n(final unit = "1") = 1 "Transformer turns ratio 1:n (primary:secondary)"
      ;
equation
  0 = p1.i + n1.i;
  0 = p2.i + n2.i;
  v1 = i1 * (Ron / dsat + (1 - dsat) * RD / n ^ 2 / dsat ^ 2) + (1 - dsat) / dsat / n * (
      v2 + VD);
  -i2 = i1 * (1 - dsat) / dsat / n;
end CCM4;
```

In the case of **CCM5**, conduction losses are taken into account with the transistor on resistance, R_{on}, and the diode forward voltage drop, V_D. Switching losses are also estimated by providing the switching frequency, f_s, and the reverse diode recovery time, t_r, and charge, Q_r.

$$v_1 = \frac{i_1 - f_s\, Q_r}{d_{sat} + f_s\, t_r}\, R_{on} + \frac{1 - d_{sat}}{d_{sat}}\, (v_2 + V_D) \tag{4.5a}$$

$$-i_2 = \frac{1 - d_{sat} - f_s\, t_r}{d_{sat} + f_s\, t_r}\, i_1 - \frac{f_s\, Q_r}{d_{sat} + f_s\, t_r} \tag{4.5b}$$

Listing 4.8: Electrical/CCM5.mo

```
model CCM5 "Average CCM model with conduction losses and diode reverse recovery"
  extends Interfaces.SwitchNetworkInterface;
  parameter Modelica.SIunits.Resistance Ron = 0 "Transistor on resistance";
  parameter Modelica.SIunits.Voltage VD = 0 "Diode forward voltage drop";
  parameter Modelica.SIunits.Charge Qr "Diode reverse recovery charge";
  parameter Modelica.SIunits.Time tr "Diode reverse recovery time";
  parameter Modelica.SIunits.Frequency fs "Switching frequency";
equation
  0 = p1.i + n1.i;
  0 = p2.i + n2.i;
  v1 = (i1 - fs * Qr) * Ron / (dsat + fs * tr) + (1 - dsat) / dsat * (v2 + VD);
  -i2 = i1 * (1 - dsat - fs * tr) / (dsat + fs * tr) - fs * Qr / (dsat + fs * tr);
end CCM5;
```

The first of the Continuous Conduction Mode (CCM)-Discontinuous Conduction Mode (DCM) models assumes no losses and no transformer. As explained in Section 3.2.2, the averaged model valid for DCM obeys the following equations:

$$R_e = \frac{2\, L_e\, f_s}{d_{sat}^2} \tag{4.6a}$$

$$\mu = \max\left(d_{sat}, \frac{1}{1 + R_e\, \frac{i_1}{v_2}}\right) \tag{4.6b}$$

$$v_1 = \frac{1 - \mu}{\mu}\, v_2 \tag{4.6c}$$

$$-i_2 = \frac{1 - \mu}{\mu}\, i_1 \tag{4.6d}$$

Listing 4.9: Electrical/CCM_DCM1.mo

```
model CCM_DCM1 "Average CCM-DCM model with no losses"
  extends Interfaces.SwitchNetworkInterface;
  parameter Modelica.SIunits.Inductance Le "Equivalent DCM inductance";
  parameter Modelica.SIunits.Frequency fs "Switching frequency";
protected
  Real mu "Effective switch conversion ratio";
  Real Re "Equivalent DCM port 1 resistance";
```

```
equation
  0 = p1.i + n1.i;
  0 = p2.i + n2.i;
  Re = 2 * Le * fs / dsat ^ 2;
  mu = max(dsat, 1 / (1 + Re * max(0, i1) / v2));
  v1 = (1 - mu) / mu * v2;
  -i2 = (1 - mu) / mu * i1;
end CCM_DCM1;
```

In CCM_DCM2, an optional transformer ratio, n, is added.

$$R_e = \frac{2\,L_e\,f_s}{d_{sat}^2} \tag{4.7a}$$

$$\mu = \max\left(d_{sat}, \frac{1}{1 + R_e\,\frac{i_1}{v_2}}\right) \tag{4.7b}$$

$$v_1 = \frac{1-\mu}{n\,\mu}\,v_2 \tag{4.7c}$$

$$-i_2 = \frac{1-\mu}{n\,\mu}\,i_1 \tag{4.7d}$$

Listing 4.10: Electrical/CCM_DCM2.mo

```
model CCM_DCM2 "Average CCM-DCM model with no losses and transformer"
  extends Interfaces.SwitchNetworkInterface;
  parameter Modelica.SIunits.Inductance Le "Equivalent DCM inductance";
  parameter Modelica.SIunits.Frequency fs "Switching frequency";
  parameter Real n(final unit = "1") = 1 "Transformer turns ratio 1:n (primary:secondary)"
      ;
protected
  Real mu "Effective switch conversion ratio";
  Real Re "Equivalent DCM port 1 resistance";
equation
  0 = p1.i + n1.i;
  0 = p2.i + n2.i;
  Re = 2 * Le * n * fs / dsat ^ 2;
  mu = max(dsat, 1 / (1 + Re * max(0, i1) / v2));
  v1 = (1 - mu) * v2 / mu / n;
  -i2 = (1 - mu) * i1 / mu / n;
end CCM_DCM2;
```

4.2.4 Photovoltaic arrays

The model included in PVSystems for the modelling of general PV arrays is PVArray. It extends from the OnePort interface, from the Modelica.Electrical.Analog library. It includes two Real inputs, the solar irradiance, G, and the ambient temperature, T.

The parameters correspond to the ones previously presented in Table 3.1 and in the discussion of Section 3.1. Additionally, the if statement at the end of the model provides some conditions to manage the behaviour in the boundaries.

Listing 4.11: Electrical/PVArray.mo

```
model PVArray "Flexible PV array model"
  extends Modelica.Electrical.Analog.Interfaces.OnePort;
  Modelica.Blocks.Interfaces.RealInput G "Solar irradiation";
  Modelica.Blocks.Interfaces.RealInput T "Panel temperature";
  constant Modelica.SIunits.Charge q = 1.60217646e-19 "Electron charge";
  constant Real Gn = 1000 "STC irradiation";
  constant Modelica.SIunits.Temperature Tn = 298.15 "STC temperature";
  parameter Modelica.SIunits.Current Imp = 7.61 "Maximum power current";
  parameter Modelica.SIunits.Voltage Vmp = 26.3 "Maximum power voltage";
  parameter Modelica.SIunits.Current Iscn = 8.21 "Short circuit current";
  parameter Modelica.SIunits.Voltage Vocn = 32.9 "Open circuit voltage";
  parameter Real Kv = -0.123 "Voc temperature coefficient";
  parameter Real Ki = 3.18e-3 "Isc temperature coefficient";
  parameter Real Ns = 54 "Number of cells in series";
  parameter Real Np = 1 "Number of cells in parallel";
  parameter Modelica.SIunits.Resistance Rs = 0.221 "Equivalent series resistance of array"
      ;
  parameter Modelica.SIunits.Resistance Rp = 415.405 "Equivalent parallel resistance of
      array";
  parameter Real a = 1.3 "Diode ideality constant";
  parameter Modelica.SIunits.Current Ipvn = Iscn "Photovoltaic current at STC";
  Modelica.SIunits.Voltage Vt "Thermal voltage of the array";
  Modelica.SIunits.Current Ipv "Photovoltaic current of the cell";
  Modelica.SIunits.Current I0 "Saturation current of the cell";
  Modelica.SIunits.Current Id "Diode current";
  Modelica.SIunits.Current Ir "Rp current";
equation
  Vt = Ns * Modelica.Constants.k * T / q;
  Ipv = (Ipvn + Ki * (T - Tn)) * G / Gn;
  I0 = (Iscn + Ki * (T - Tn)) / (exp((Vocn + Kv * (T - Tn)) / a / Vt) - 1);
  Id = I0 * (exp((v - Rs * i) / a / Vt) - 1);
  Ir = (v - Rs * i) / Rp;
  if v < 0 then
    i = v / ((Rs + Rp) / Np);
  elseif v > Vocn then
    i = 0;
  else
    i = -Np * (Ipv - Id - Ir);
  end if;
end PVArray;
```

4.2.5 Energy storage

The `SimpleBattery` model extends from the `BatteryInterface` model, which itself extends from `OnePort` from the MSL (Listing 4.1). In this version, this battery interface model is currently not adding much. It's included to provide a common class from which other battery implementations can extend in the future.

It includes the parameters discussed in Section 3.3, and the output voltage is calculated as a function of the state of charge, according to the equation presented in that section:

$$E = E_0 - K\,\frac{Q}{Q - i_t} + A\,e^{-B\,i_t} \tag{4.8}$$

Listing 4.12: Electrical/SimpleBattery.mo

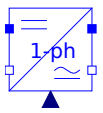
```
model SimpleBattery "Simple battery model"
```

```
  extends Interfaces.BatteryInterface;
  type BatteryCapacity = Real(final quantity = "Energy", final unit = "A.h");
  parameter .Modelica.SIunits.Resistance Rint = 0.09 "Internal resistance";
  parameter .Modelica.SIunits.Voltage E0 = 3.7348 "Constant battery voltage";
  parameter .Modelica.SIunits.Voltage K = 0.00876 "Polarization voltage";
  parameter BatteryCapacity Q = 1 "Rated battery capacity";
  parameter .Modelica.SIunits.Voltage A = 0.468 "Exponential region amplitude";
  parameter Real B = 3.5294 "Exponential zone time constant inverse";
  parameter BatteryCapacity DoDini = 0 "Initial Depth of Discharge";
  .Modelica.SIunits.Voltage E;
  BatteryCapacity it(start = DoDini, fixed = true) "Actual depth of discharge";
equation
  v = E + i * Rint;
  der(it) = -i / 3600;
  E = max(0, E0 - K * Q / (Q - it) + A * exp(-B * it));
end SimpleBattery;
```

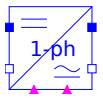
4.2.6 Power converters

Under the `Assemblies` sub-package, four models for two different converters are included. Both converters are topologically quite similar, both use 4 switches that can be modelled using 2 switch network models.

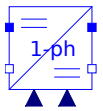
The first model, **HBridge** (Figure 4.3a), is an implementation of the H-bridge topology with replaceable switch network models. This means that this model can be used to create both switching and averaged models and the implementation can be changed very easily through a dialogue available from the block diagram.



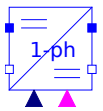
The second model, **HBridgeSwitched** (Figure 4.3b), is a switched implementation of the H-bridge topology. This model is not as versatile as the previous one and is included mainly for comparison and validation purposed. This model might be removed in future releases of the library.



The third model, **BidirectionalBuckBoost** (Figure 4.4a), is a non-inverting bidirectional buck boost that is also implemented with replaceable switch network models. Apart from the switch elements, it also includes the input and output ripple filtering capacitors and the power inductor, as well as Equivalent Series Resistance (ESR) for all three.



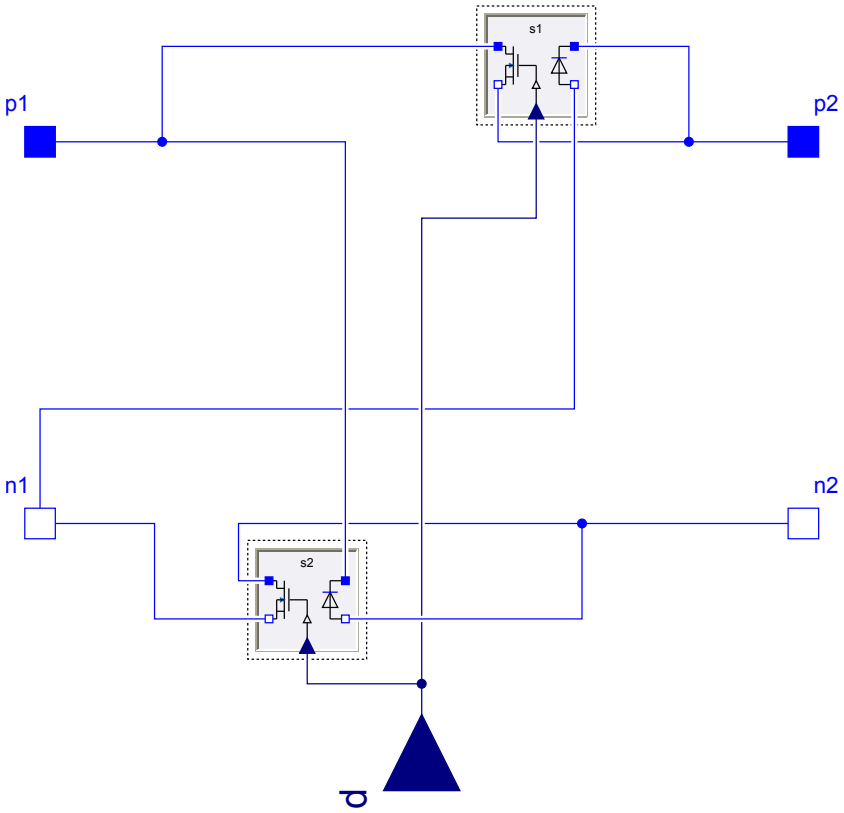
The fourth and final model, **CPMBidirectionalBuckBoost** (Figure 4.4b), is just an instantiation of the previous buck boost model with the requisite control blocks around it.



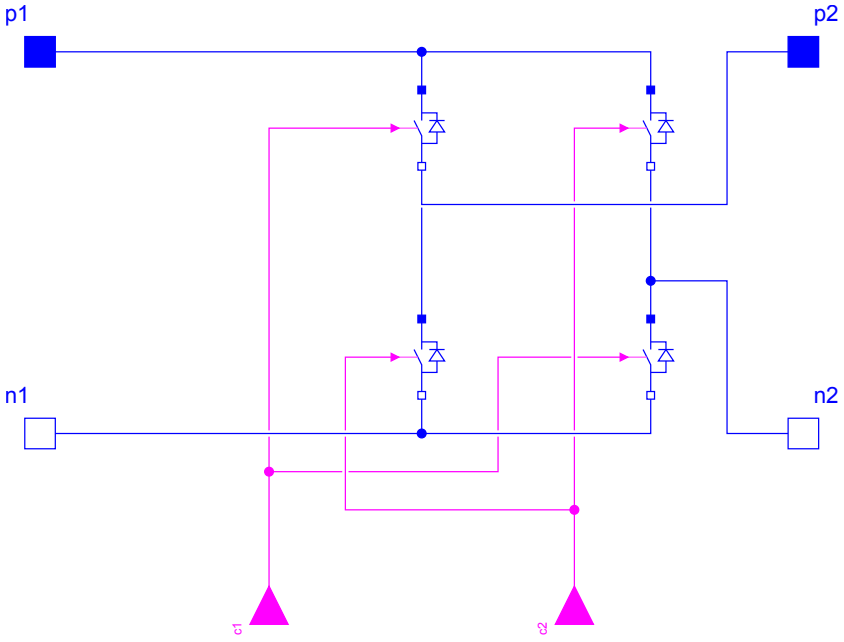
4.3 Control models

The control blocks included in this library are not meant to be comprehensive. The vision for `PVSystems` is to provide models of physical components of PV systems to build models to validate converter or supervisory control software.

This control software is not typically developed with Modelica or within a Modelica Integrated Development Environment (IDE). It could be some C source code, code

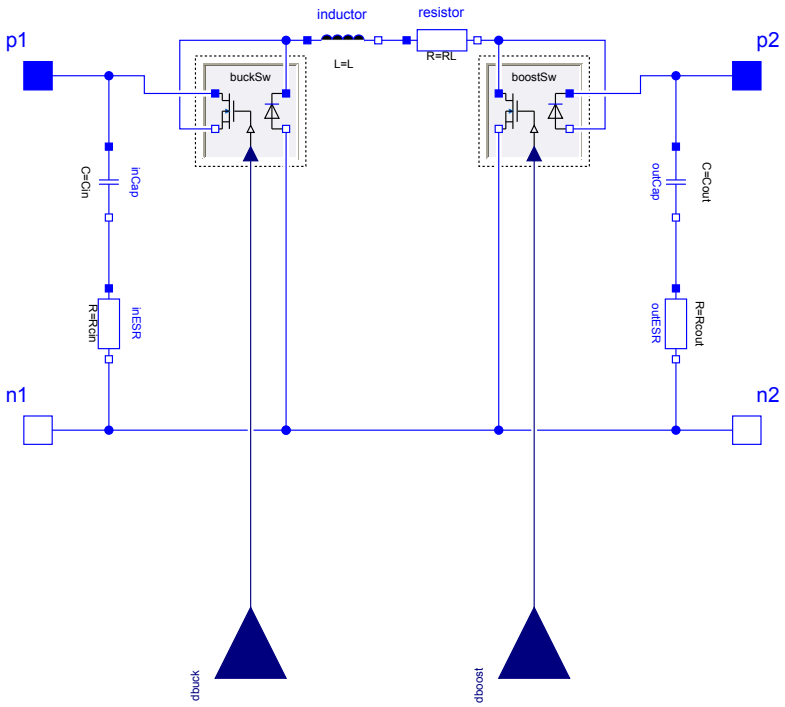


(a)

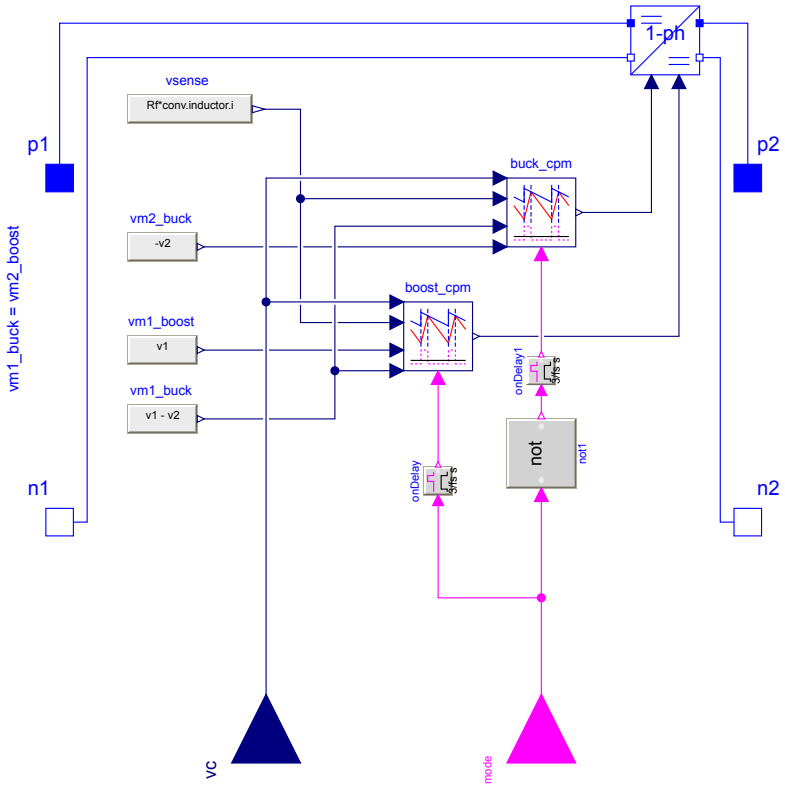


(b)

Figure 4.3: H-bridge converters in `Electrical.Assemblies`



(a)



(b)

Figure 4.4: Bidirectional buck-boost converters in `Electrical.Assemblies`

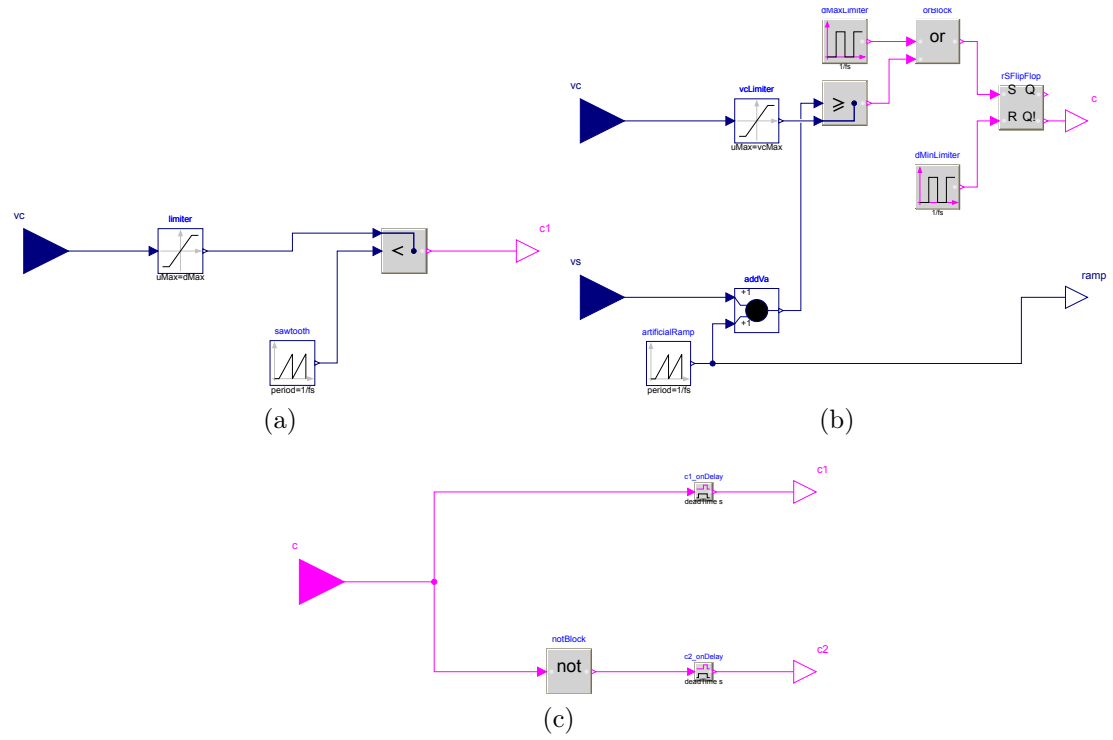


Figure 4.5: Basic switching control blocks in `Control`

written in Simulink or a LabVIEW program. In all of these cases, developers don't want to reproduce their work in Modelica, they want to test it directly. This can be achieved through model exchange and co-simulation, using the FMI, for example.

The reason to include a few control blocks is to be able to construct basic examples to showcase the use of the library.

4.3.1 Basic components

The first three blocks are shown in Figure 4.5. **`SwitchingPWM`** (Figure 4.5a), is a very basic implementation of a PWM generator. The input is optionally saturated and compared with a sawtooth signal with fixed frequency. The output is the PWM signal.

`SwitchingCPM` (Figure 4.5b) is a regular switching version of the CPM modulator. As explained in Section 3.4.1, this works by measuring the current over a switch or inductor of interest and creating the PWM signal by comparing that with a control input value. This implementation also features saturation of the control input and artificial ramp. This block, as many others, is taken from [EMA16].

`DeadTime` (Figure 4.5c) is a block that prevents two switches in series from being closed at the same time. From a single PWM firing signal, it creates two complementary signals and introduces an on-delay to both to perform this functionality.

CPM_CCM (Listing 4.13) is an averaged version of the CPM modulator, valid only in CCM operation. As explained in Section 3.4.3, it obeys the following equation:

$$d = \frac{2\,(v_c - v_s)}{\frac{R_f}{L\,f_s}\,(v_{m1} + v_{m2})\,(1-d) + 2\,V_a} \tag{4.9}$$

Notice also that an `if` statement is used to disable the block. This is useful in some applications like the non-inverting bidirectional buck boost, which can be operated as a buck in series with a boost converter by disabling a pair of switches. It also removes the non-linear equation from the list of equations to be solved, which prevents convergence issues and speeds up simulation.

Listing 4.13: Control/CPM_CCM.mo

```
model CPM_CCM "Current Peak Mode modulator for averaged CCM models"
  extends Interfaces.CPMInterface;
  parameter Real d_disabled(final unit = "1") "Value of duty cycle when disabled";
  Modelica.Blocks.Interfaces.BooleanInput enable "Block enable/disable";
equation
  if enable then
    d = 2 * (vc - vs) / (Rf / L / fs * (vm1 + vm2) * (1 - d) + 2 * Va);
  else
    d = d_disabled;
  end if;
end CPM_CCM;
```

CPM (Listing 4.14) is also an averaged version of the CPM modulator, but valid both in CCM and CPM. The equations determining its behaviour are now the following:

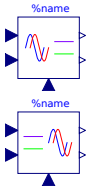
$$d_2 = \min\left(\frac{L\,f_s\,(v_c - V_a\,d)}{R_f\,v_{m2}}, 1-d\right) \tag{4.10a}$$

$$d = 2\,\frac{v_c\,(d+d_2) - v_s}{v_{m1} + v_{m2}}\,d_2\,(d+d_2) + 2\,V_a\,(d+d_2) \tag{4.10b}$$

Listing 4.14: Control/CPM.mo

```
model CPM "Current Peak Mode modulator for averaged models"
  extends Interfaces.CPMInterface;
protected
  Real d2;
equation
  d2 = min(L * fs * (vc - Va * d) / Rf / vm2, 1 - d);
  d = 2 * (vc * (d + d2) - vs) / (Rf / L / fs * (vm1 + vm2) * d2 * (d + d2) + 2 * Va * (d
      + d2));
end CPM;
```

Both the **Park** (Listing 4.15) and **InversePark** (Listing 4.16) blocks are built around the same equations, the only difference being which signals are defined as inputs and which are defined as outputs. The solver will take care



of working with whatever causality is defined. As explained in Section 3.4.2, they obey the following equation:

$$\begin{pmatrix} d \\ q \end{pmatrix} = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \tag{4.11}$$

Listing 4.15: Control/Park.mo

```
block Park "Park transformation"
  extends Modelica.Blocks.Icons.Block;
  Modelica.Blocks.Interfaces.RealInput alpha;
  Modelica.Blocks.Interfaces.RealInput beta;
  Modelica.Blocks.Interfaces.RealOutput d;
  Modelica.Blocks.Interfaces.RealOutput q;
  Modelica.Blocks.Interfaces.RealInput theta;
equation
  d = alpha * cos(theta) + beta * sin(theta);
  q = (-alpha * sin(theta)) + beta * cos(theta);
end Park;
```

Listing 4.16: Control/InversePark.mo

```
block InversePark "Inverse Park transformation"
  extends Modelica.Blocks.Icons.Block;
  Modelica.Blocks.Interfaces.RealInput d;
  Modelica.Blocks.Interfaces.RealInput q;
  Modelica.Blocks.Interfaces.RealOutput alpha;
  Modelica.Blocks.Interfaces.RealOutput beta;
  Modelica.Blocks.Interfaces.RealInput theta;
equation
  d = alpha * cos(theta) + beta * sin(theta);
  q = (-alpha * sin(theta)) + beta * cos(theta);
end InversePark;
```

The last of the control basic building blocks is the **PLL** (Figure 4.6). The purpose of the block is to extract the phase of a given signal with which we want to be synchronised. This is done by a very simple Quadrature Signal Generator (QSG), delaying the input signal by a quarter of a cycle. These two signals are used as input for a `Park` block, which is closed to make the q coordinate equal to zero, using a Proportional Integral Derivative (PID) controller.

4.3.2 Controllers

Three controllers are available in the library. **MPPTController** (Listing 4.17) provides a simple implementation of the Perturb and Observe (P&O) algorithm for MPP tracking. The power output from the PV array is measured and an increase or decrease of the DC bus voltage is effected. If the power decreases, the opposite action is effected in the next step. If the power increases, the same action is repeated. If the power doesn't change (i.e. it's below a certain threshold), voltage is not changed. This algorithm is not the most efficient or effective, but it has reasonable performance and is easy to implement.

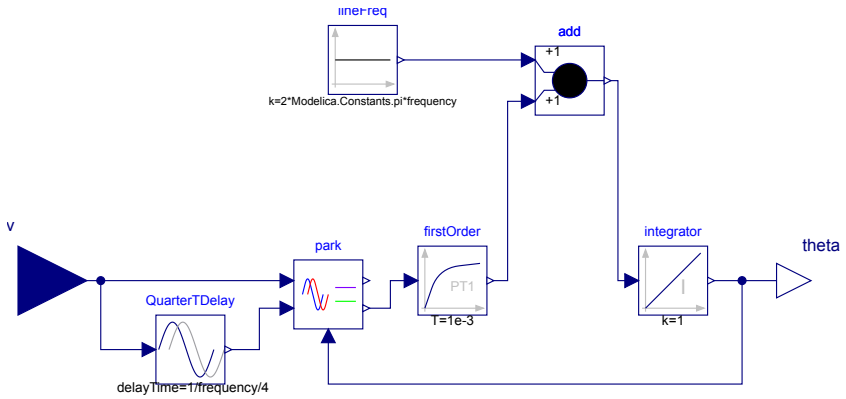


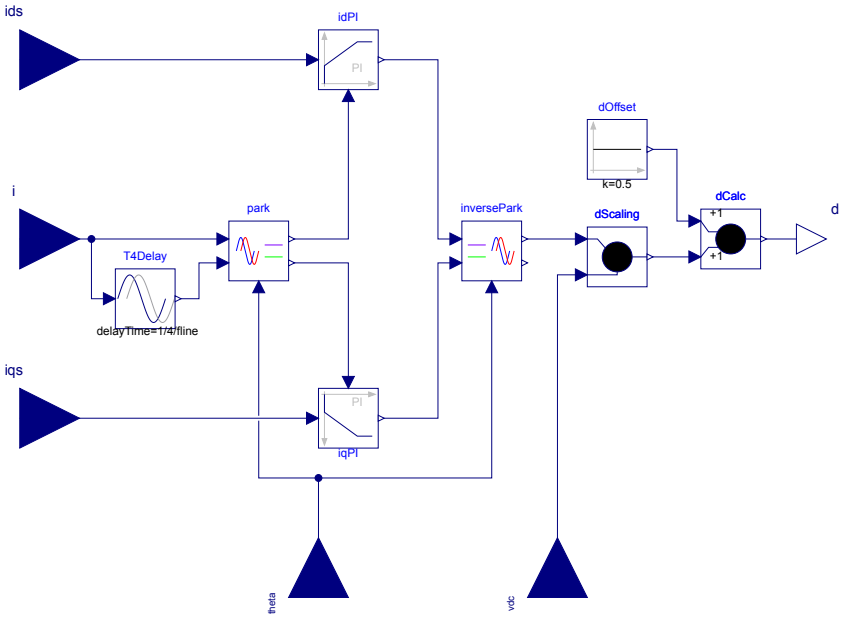
Figure 4.6: Phase-locked loop in Control

Listing 4.17: Control/MPPTController.mo

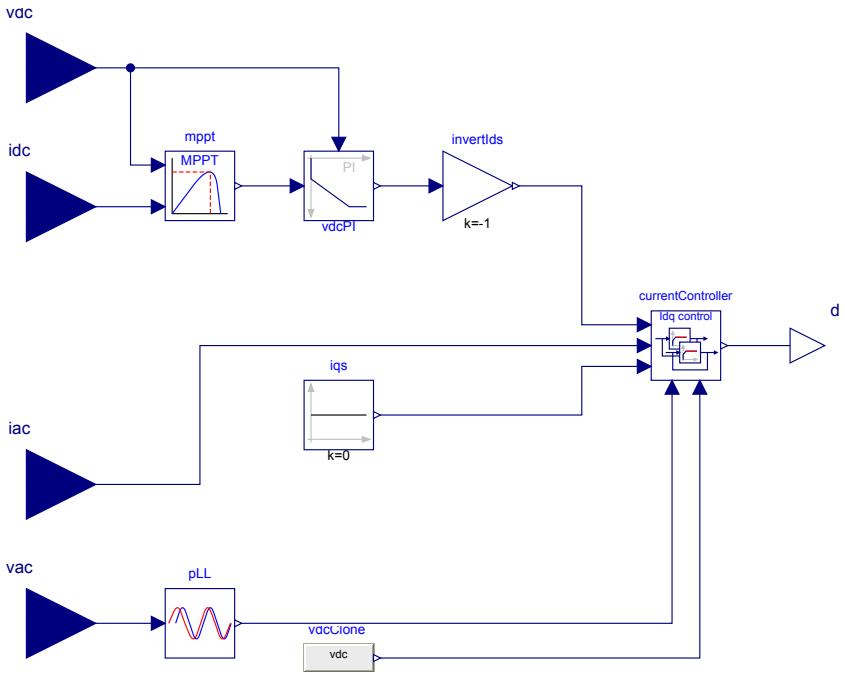
```
block MPPTController "Maximum Power Point Tracking Controller"
  extends Modelica.Blocks.Interfaces.SI2SO;
  parameter Modelica.SIunits.Time sampleTime = 1 "Sample time of control block";
  parameter Modelica.SIunits.Voltage vrefStep = 5 "Step of change for vref";
  parameter Modelica.SIunits.Power pkThreshold = 1 "Power threshold below which no change
      is considered";
  parameter Modelica.SIunits.Voltage vrefStart = 10 "Voltage reference initial value";
protected
  discrete Modelica.SIunits.Voltage vk;
  discrete Modelica.SIunits.Current ik;
  discrete Modelica.SIunits.Power pk;
  discrete Modelica.SIunits.Voltage vref(start = vrefStart);
equation
  when sample(sampleTime, sampleTime) then
    vk = pre(u1);
    ik = pre(u2);
    pk = vk * ik;
    if abs(pk - pre(pk)) < pkThreshold then
      vref = pre(vref);
    elseif pk - pre(pk) > 0 then
      vref = pre(vref) + vrefStep * sign(vk - pre(vk));
    else
      vref = pre(vref) - vrefStep * sign(vk - pre(vk));
    end if;
  end when;
  y = vref;
end MPPTController;
```

Inverter1phCurrentController (Figure 4.7a) provides current control in the synchronous reference frame for a 1-phase inverter. This is achieved by calculating the dq coordinates of the output current, using a PI controller for each coordinate. With this setup, the d coordinate can be used to control the active power and the q coordinate to control the reactive power. The output of this block is the duty cycle, which can then drive an inverter.

Inverter1phCompleteController (Figure 4.7b) build upon the previous by adding an outer voltage control loop to the active power control (i_d), using an MPPTController block. It also adds a PLL block for synchronization with the grid.



(a)



(b)

Figure 4.7: Inverter controllers in `Control.Assemblies`

5 Development and verification

5.1 Introduction

A quick and rough measure for the size of software projects is the Lines Of Code (LOC). By that measure, `PVSystems` is a small project. It has also been developed by one programmer alone, which obviates the need for communication and coordination among team members.

For such a project, not a lot of software engineering considerations have to be made. The development and maintenance workload remains small enough that no specific organizing strategy needs to be followed to make it manageable.

On the other hand, this is also a learning project. With the goal of exploring and showcasing good practices, this chapter will explore some software engineering topics. Two aspects are of interest:

- Library architecture: the way software is organized has a big impact on its quality, maintainability and extensibility. A good architecture can be created through thoughtful design. From an Agile perspective, a good architecture can also be evolved in time, as new considerations and requirements arise. The library architecture was discussed in Section 4.1.

- Development tools and practices: these will be covered in this chapter. A special emphasis will be placed on verification and validation, in Section 5.1.2.

5.1.1 Development tools and practices

In it's simplest form, the development of `PVSystems` could be carried out with a text editor and a Modelica compiler. This could be quite painful, though. Modelica specific tools and IDEs exist to provide convenience for the developer. Convenience is typically translated into fewer developer mistakes and software bugs, as well as saved time.

The main IDE used in the development of `PVSystems` is Dymola [Das17], from Dassault Systèmes. It was used almost exclusively except for a couple of features available in OpenModelica [Ope17], another popular Modelica IDE, which is open source and is maintained by the Open Source Modelica Consortium.

In addition to these, Emacs has been used for some source code editing tasks and Git has been used for version control. The source has also been hosted and made publicly available on GitHub [Rod17a], together with the Dymola exported documentation [Rod17b]. Even for a one-developer project, source code version control is key. This is probably by now an uncontroversial remark, but it wasn't always like this. Source code version control still has some footing to gain within professionals not trained as computer scientists.

5.1.2 Verification and validation

Verification and validation are normally used interchangeably, but the IEEE Standard for System and Software Verification and Validation [IEE+12], provides the following definitions:

- Verification: (A) The process of evaluating a system or component to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase. (B) The process of providing objective evidence that the system, software, or hardware and its associated products conform to requirements (e.g., for correctness, completeness, consistency, and accuracy) for all life cycle activities during each life cycle process (acquisition, supply, development, operation, and maintenance); satisfy standards, practices, and conventions during life cycle processes; and successfully complete each life cycle activity and satisfy all the criteria for initiating succeeding life cycle activities. Verification of interim work products is essential for proper understanding and assessment of the life cycle phase product(s).

- Validation: (A) The process of evaluating a system or component during or at the end of the development process to determine whether it satisfies specified requirements. (B) The process of providing evidence that the system, software, or hardware and its associated products satisfy requirements allocated to it at the end of each life cycle activity, solve the right problem (e.g., correctly model physical laws, implement business rules, and use the proper system assumptions), and satisfy intended use and user needs.

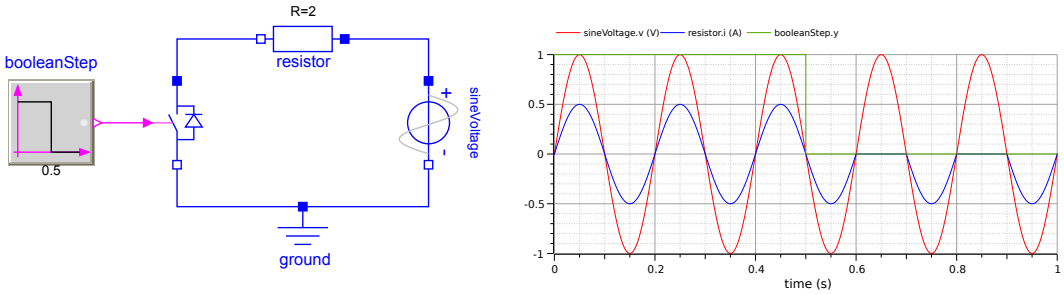
In simpler terms, verification is checking that we're *doing the thing right*, validation is checking that we're *doing the right thing*.

It is assumed, from the experience of the author, that the models provided in `PVSystems` are indeed useful for simulation and verification of PV systems and converter control software. Validation of the models developed will be explored in the following section.

5.2 Validation models and results

5.2.1 `IdealCBSwitch`

The model `IdealCBSwitch` provides an ideal current bidirectional switch, implemented with an ideal diode in anti-parallel with an ideal closing switch, both taken from MSL. The switch should provide bidirectional current flow when the boolean control signal is high and negative current flow when the boolean control signal is low. The validation diagram and results are shown in Figure 5.1.



(a) (b)

Figure 5.1: Validation of `IdealCBSwitch`

5.2.2 Switching switch network models

The validation of the three switching variants of the switch network concept can be done with a similar setup. The validation diagrams and results are displayed on Figure 5.2. `SW1` includes an ideal closing switch in port 1 and an anti-parallel diode in port 2. Connecting them in parallel and providing the right setup, switching frequency and duty cycle should produce the same result as the one obtained with `IdealCBSwitch`. This can be confirmed by comparing Figure 5.1b with Figure 5.2b.

`SW2` is formed with a single ideal closing switch in each port. The firing signals for those switches are complementary and include a dead time so that both switches are not closed at the same time. `SW3` is the same but includes anti-parallel diodes with each switch.

Setting up both validation diagrams equally (Figure 5.2c and Figure 5.2e), the results are similar and as expected (Figure 5.2d and Figure 5.2f). The difference lies in the fact that `SW2` blocks both positive and negative current, whereas `SW3` allows negative current through the diodes.

5.2.3 CCM averaged switch network models

For convenience, validation of the CCM averaged switch networks is done in a single model, `CCMXVerification`, displayed in Figure 5.3. This model includes one simple circuit for each of the 5 versions, parametrized with the values presented in Table 5.1. An input source with constant voltage and a resistive load are provided. Duty cycle is provided as a ramp starting at 0.1 for the first 0.1 s and ending at 0.9 for the last 0.1 s.

Figure 5.4 presents the output voltage and input current for each switch network. These are the variables of interest, since the model works as a transformer and the input voltage and output current are being established with the voltage source and the load, respectively.

An equivalent circuit is created with the use of LTspice, using the original library of switch network models [EMA16]. The results from LTspice and Dymola are both imported into MATLAB and compared. Figures 5.5 and 5.6 confirm that the relative differences between applications are small.

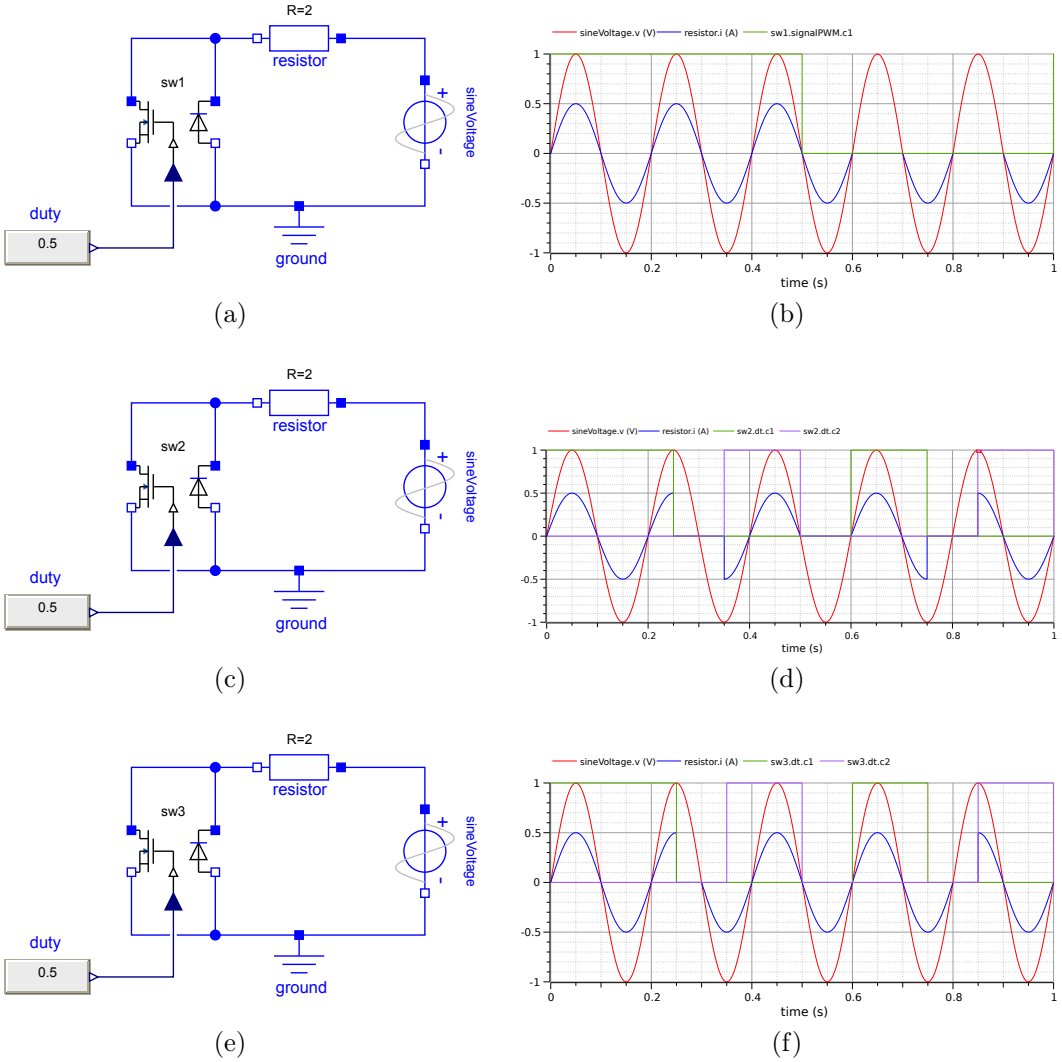


Figure 5.2: Validation of switching switch network models

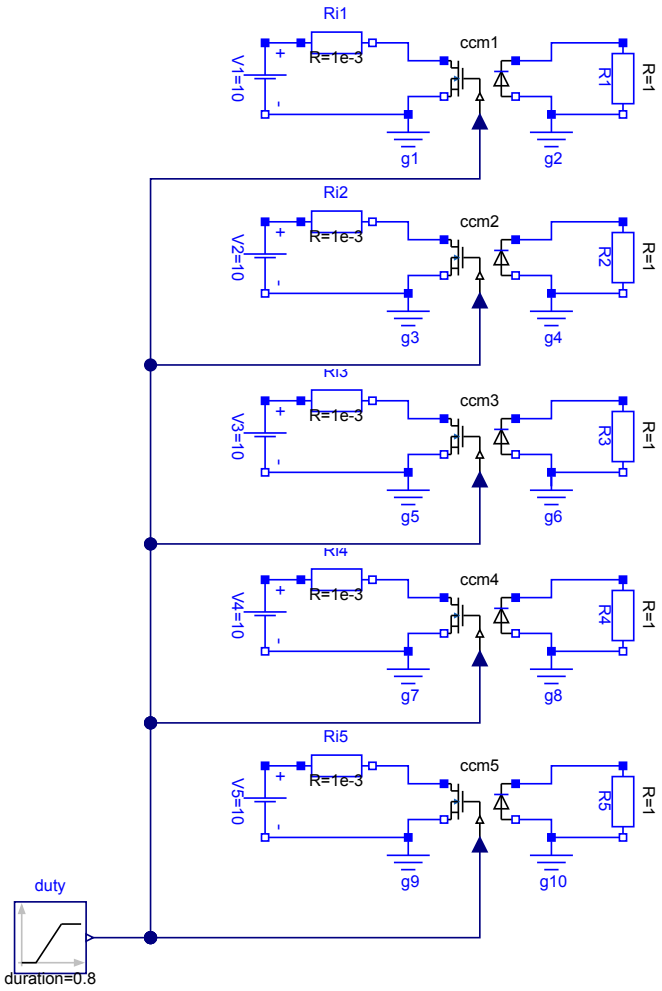
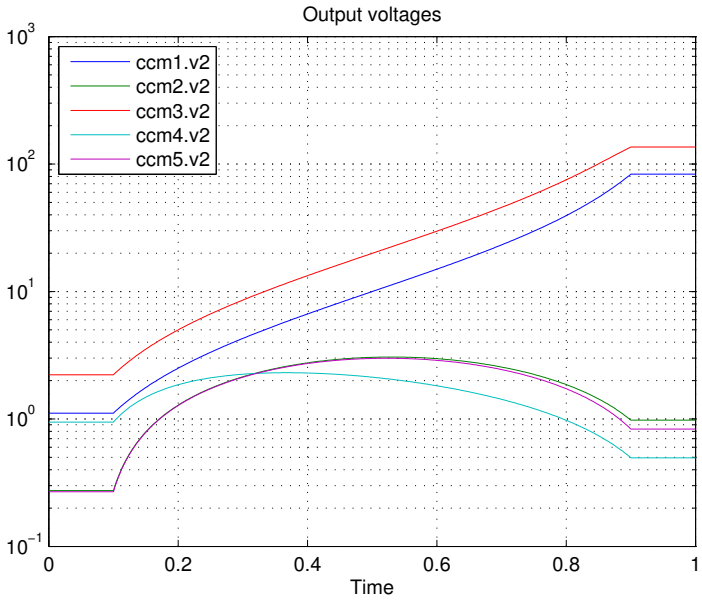
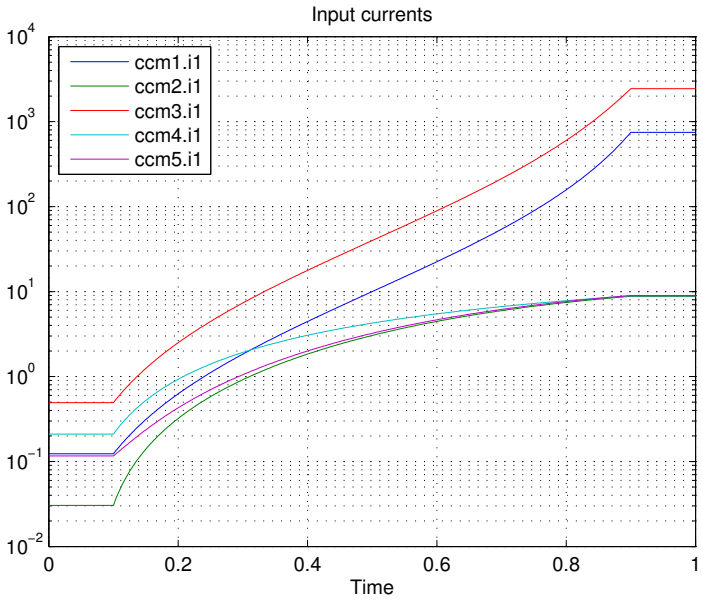


Figure 5.3: `CCMXVerification` diagram



(a)



(b)

Figure 5.4: `CCMXVerification` results: (a) output voltages and (b) input currents.

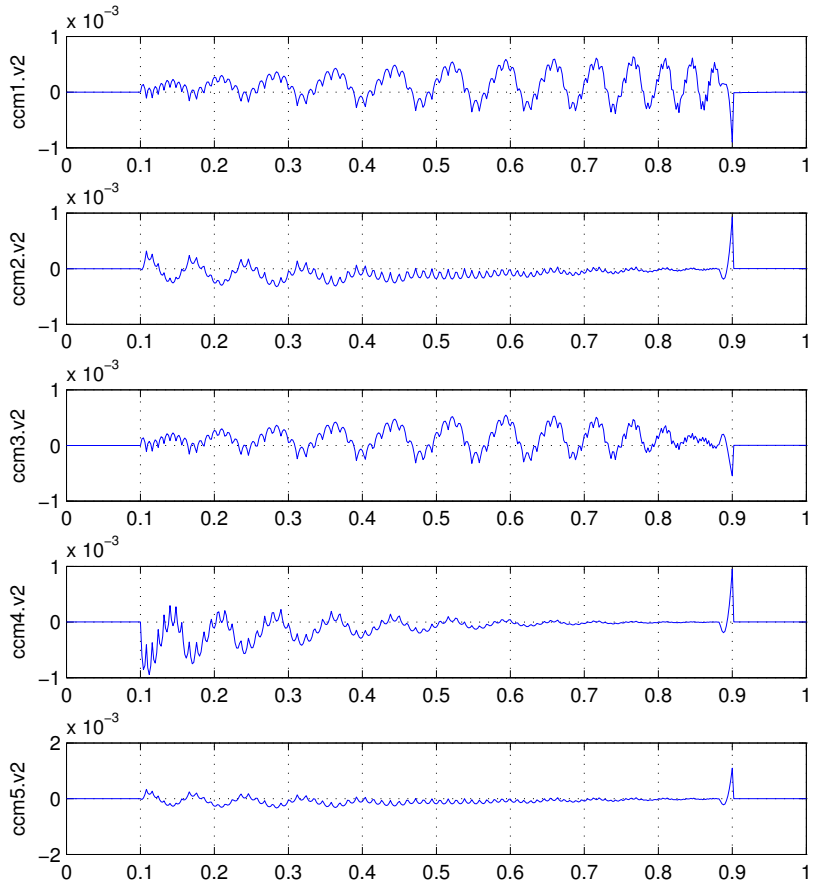


Figure 5.5: LTspice and `PVSystems` output voltages differences

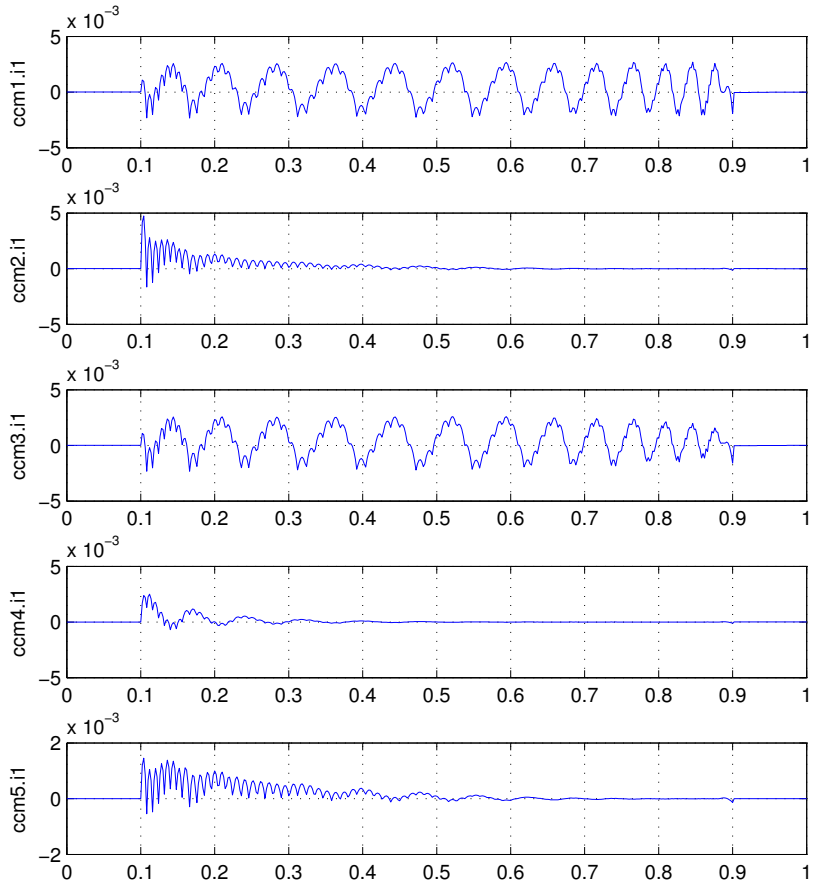


Figure 5.6: LTspice and `PVSystems` input currents differences

Table 5.1: Parametrization in `CCMXVerification`

Parameter	Value
R_{on}	$1\,\Omega$
R_D	$0.01\,\Omega$
V_D	$0.8\,\mathrm{V}$
n	2
Q_r	$0.75\,\mu\mathrm{C}$
t_r	$75\,\mathrm{ns}$

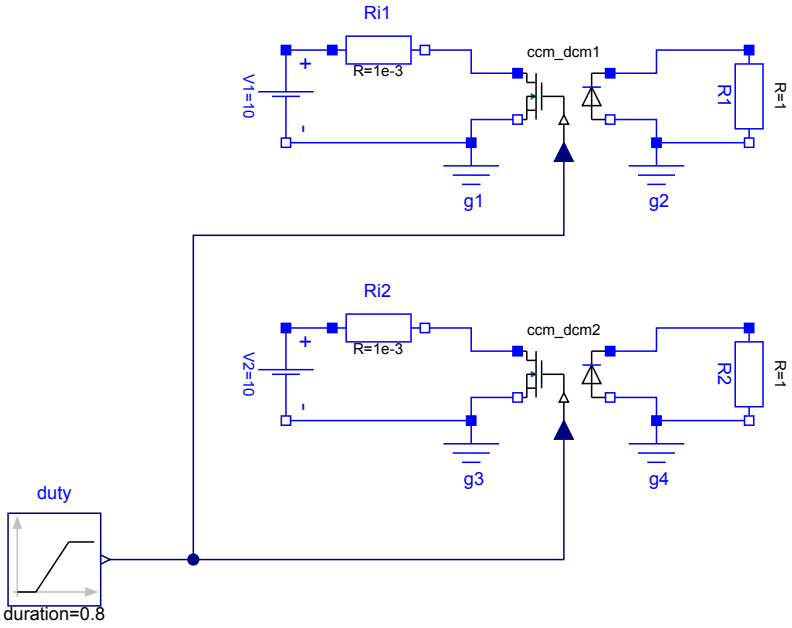


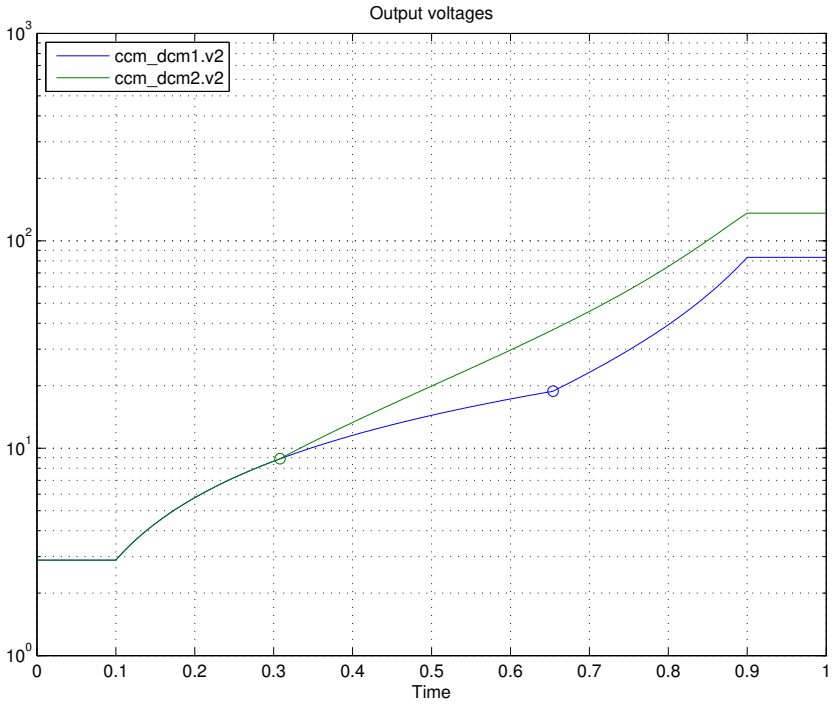
Figure 5.7: `CCM_DCMXVerification` diagram

5.2.4 CCM-DCM averaged switch network models

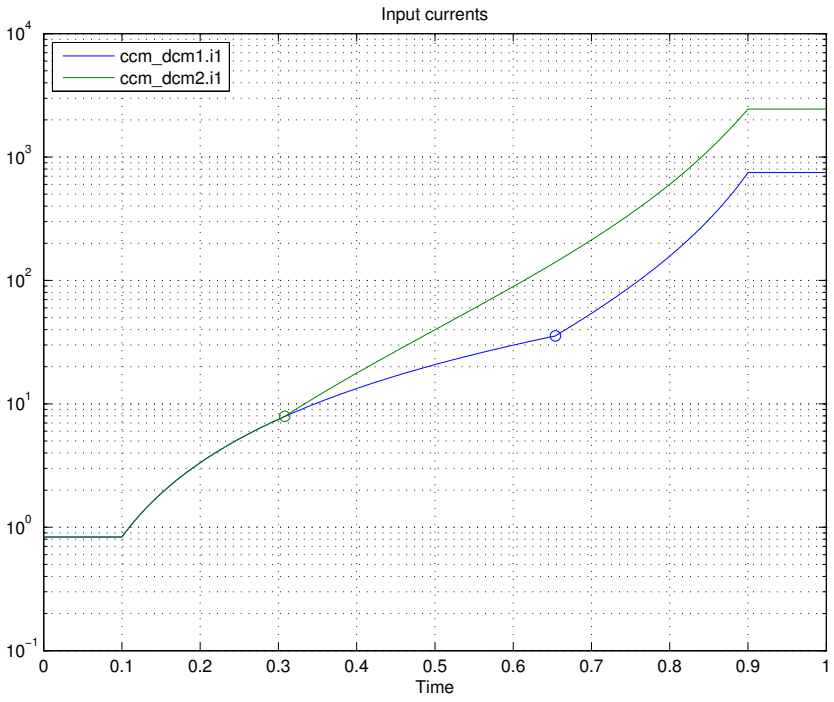
The CCM-DCM models can be tested with the same setup, using the diagram from Figure 5.7. Setting the parameters according to Table 5.2, it can trigger a CCM-DCM transition. This transition generates a sudden change in the slope of the equivalent transformer ratio and is marked with a circle for each curve in Figure 5.8. The differences with LTspice simulation of equivalent circuits are shown in Figures 5.9a and 5.9b, and are also reasonably small.

5.2.5 `PVArray`

The diagram shown in Figure 5.10a is created to validate the `PVArray` model. Irradiance and temperature are set at constant values (STC) and the voltage source connected to



(a)



(b)

Figure 5.8: `CCM_DCMXVerification` results: (a) output voltages and (b) input currents.

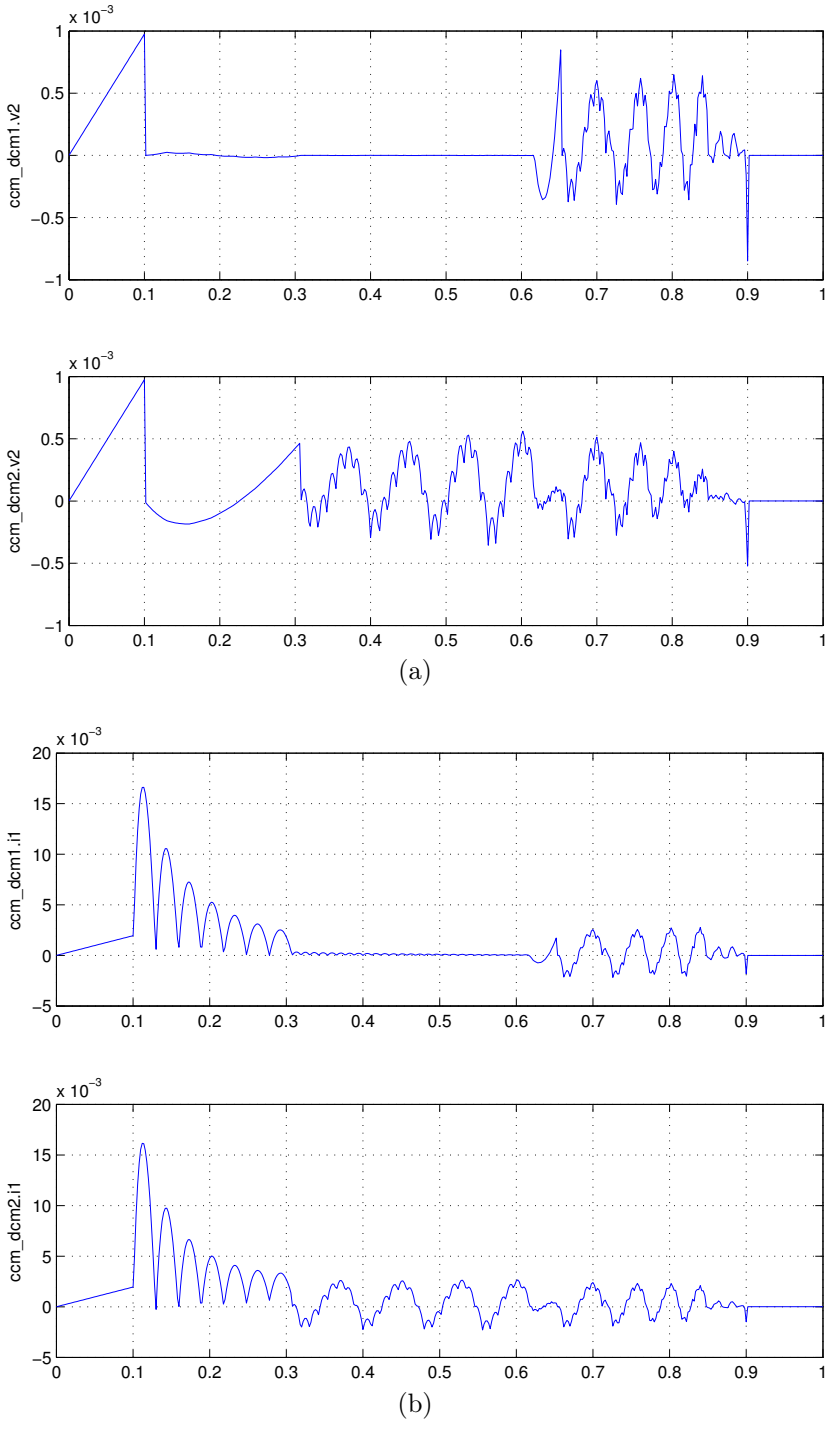


Figure 5.9: LTspice and `PVSystems` differences for `CCM_DCMXVerification`: (a) output voltages and (b) input currents.

Table 5.2: Parametrization in `CCM_DCMXVerification`

Parameter	Value
L_e	0.6 µH
f_s	100 kHz
n	2

the PV array provides a voltage sweep. Plotting current and power against voltage yields Figure 5.10b. The waveforms and notable points, which have been tagged, agree with those presented in [VGF09].

5.2.6 SimpleBattery

The diagram for `SimpleBatteryVerification` is shown in Figure 5.11a. A current source is set at ±2 A for charging/discharging. A simple feedback mechanism is included to put the battery through full charge and discharge cycles. The waveforms and values correspond to those presented in [TDD07].

5.2.7 SwitchingPWM

The validation of `SwitchingPWM` can be done by exciting the block with a varying duty cycle and measuring the actual duty cycle of the output control signal. The diagram to perform this operation and the result are displayed in Figure 5.12.

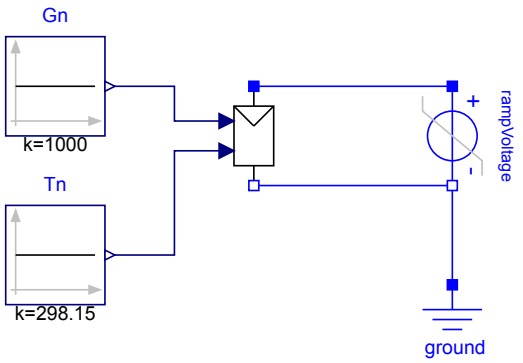
5.2.8 SwitchingCPM

The validation of `SwitchingCPM` requires a slightly more sophisticated setup, as displayed in Figure 5.13a. The output of the CPM block is used to switch between two different constant values as the input to an integrator block whose output is fed back as an measurement signal. The constants `vdT` and `vdpT` correspond to the voltage values that an inductor would be subjected to in each of the `pwm` periods. In a real application, these values would not be constant, but this setup provides a simple and decent approximation. The constant of the integrator block represents to the inverse of the emulated inductance value. In this example, $L = 400\,\mu H$.

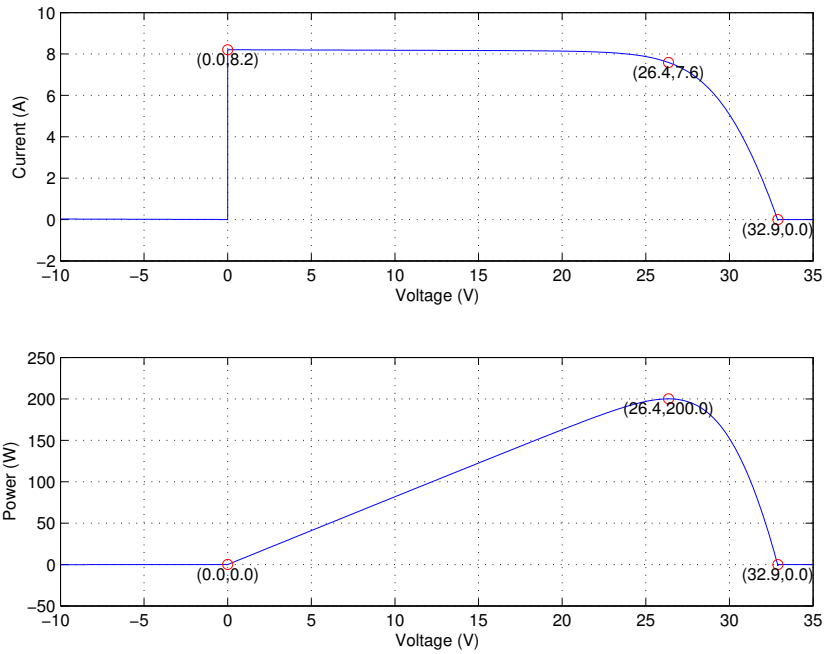
The results displayed in Figure 5.13b illustrate how the PWM output signal is conditioned by the rising and falling of the measured signal (what would normally be an inductor current), resetting the signal when the measured current reaches the control signal minus the artificial ramp.

5.2.9 DeadTime

The `DeadTime` block is also quite simple. An appropriate dead time interval is set with the block's corresponding parameter and the input pulse signal is compared with the

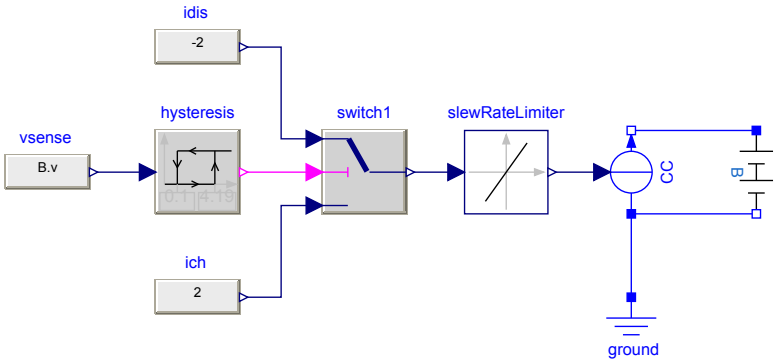


(a)

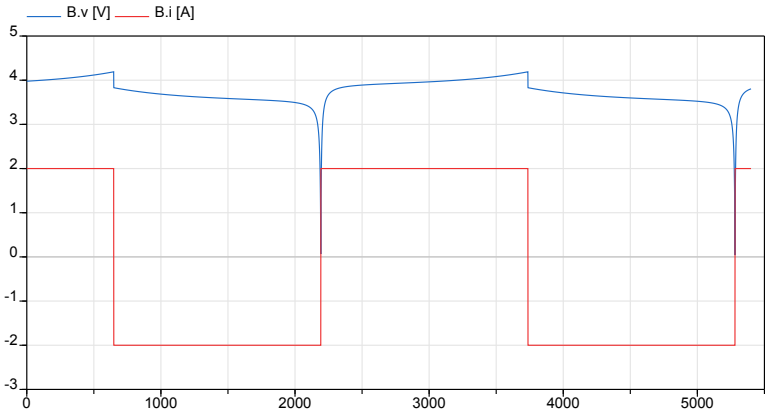


(b)

Figure 5.10: `PVArrayVerification`: (a) diagram and (b) results.

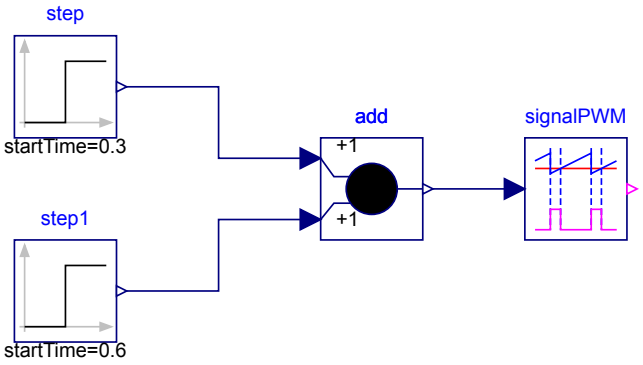


(a)

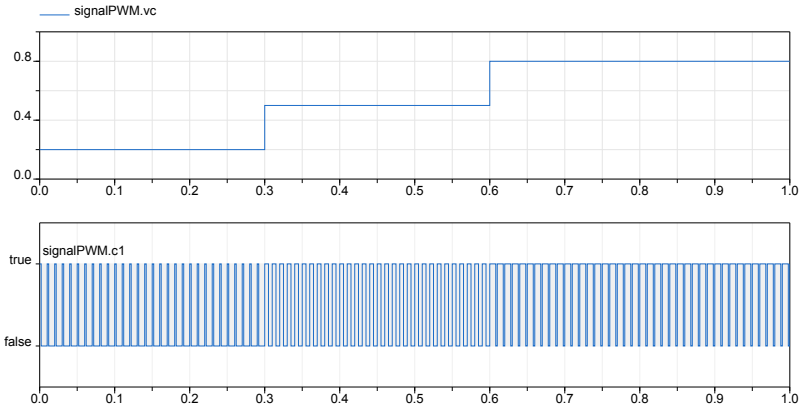


(b)

Figure 5.11: `SimpleBatteryVerification`: (a) diagram and (b) results.

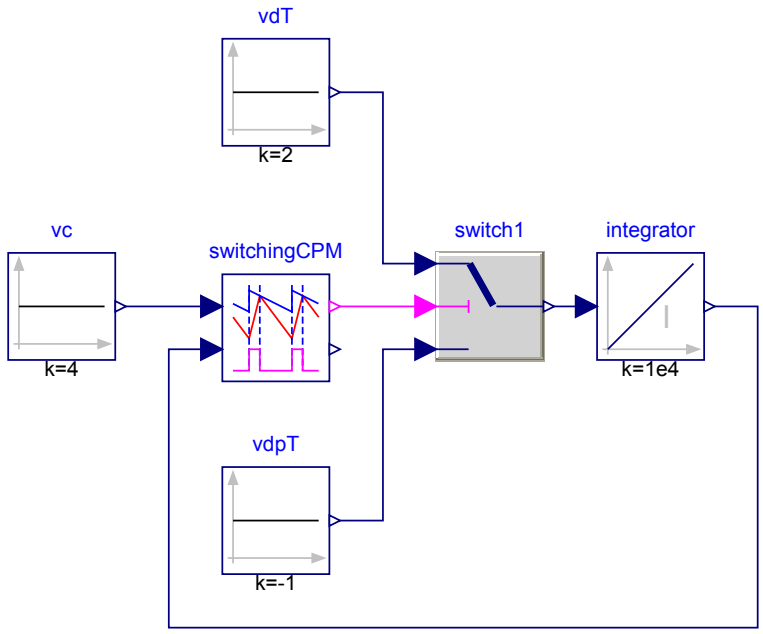


(a)

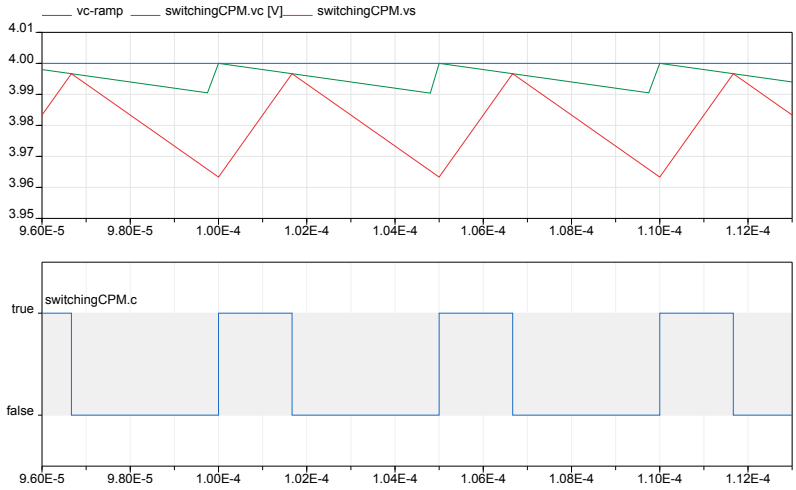


(b)

Figure 5.12: `SwitchingPWMVerification`: (a) diagram and (b) results.

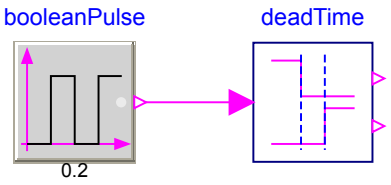


(a)

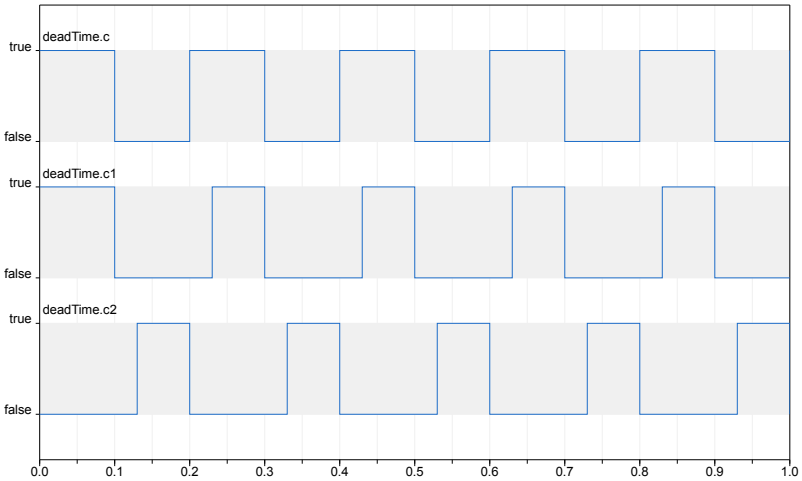


(b)

Figure 5.13: `SwitchingCPMVerification`: (a) diagram and (b) results.



(a)



(b)

Figure 5.14: `DeadTimeVerification`: (a) diagram and (b) results.

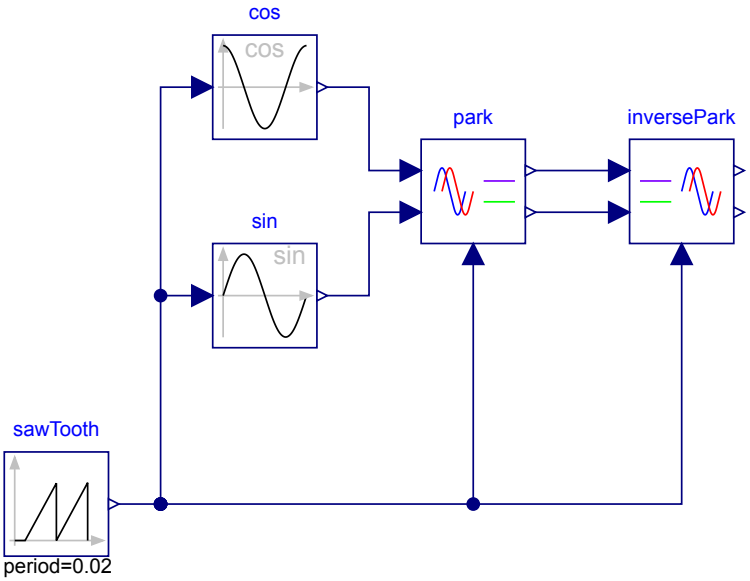
output signals. These signals are the complement of each other with an added dead time in between that, in practical applications, will prevent series switches from being closed at the same time causing a short-circuit.

5.2.10 Park transforms

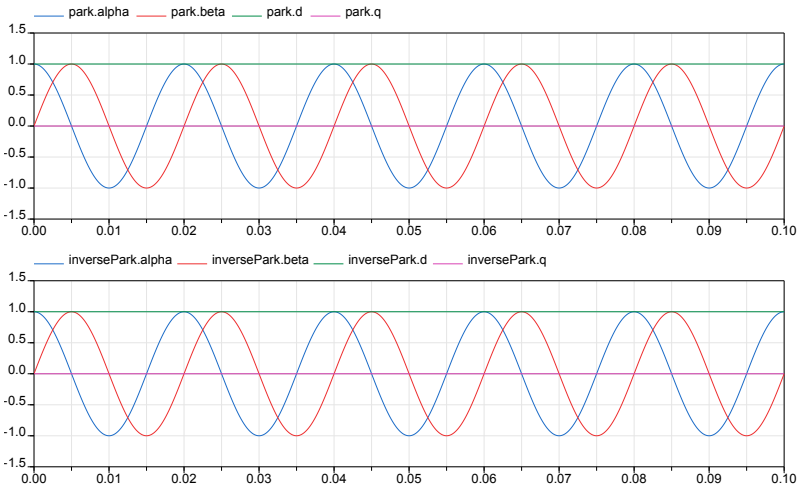
In order to validate the Park transforms blocks, the block diagram from Figure 5.15a is developed. A reference θ is generated with a sawtooth generator, driving sine and cosine blocks to provide the input for `Park`. The output of this block is used as the input for `InversePark`. As displayed in Figure 5.15b, the results fit what's to be expected from these transforms. Since no initial phase is added to the sine and cosine signals and the angle between them is 90°, d matches the amplitude of the signals and q is equal to 0.

5.2.11 `PLL`

The verification of this block is done with the diagram displayed in Figure 5.16a. A sinusoidal input is provided to the `PLL` block which, parametrized correctly, synchronizes with that input signal. The output of the `PLL` block is the phase, so it's input into a cosine block to compare results, as displayed in Figure 5.16b.

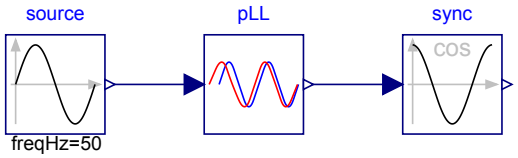


(a)

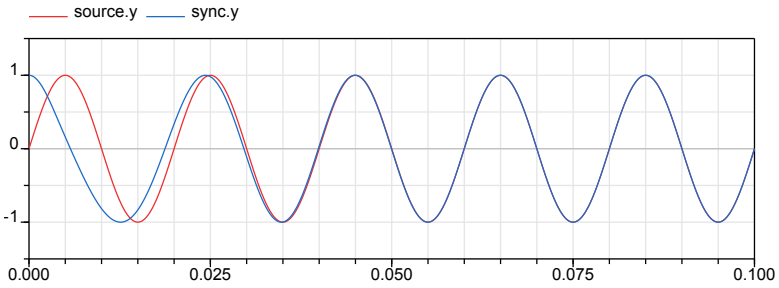


(b)

Figure 5.15: `ParkTransformsVerification`: (a) diagram and (b) results.



(a)



(b)

Figure 5.16: `PLLVerification`: (a) diagram and (b) results.

5.2.12 `MPPTController`

The validation of `MPPTController` requires a more complex setup, as displayed in Figure 5.17. For comparison, two `PVArray` blocks are included. They both are exposed to the same varying conditions of irradiance and temperature, shown at the top of Figure 5.18a. The values start at STC and gradually deviate from that.

The voltage point of one of the `PVArray` blocks is set at a constant value of 26 V, which at STC temperature and irradiance is very close to the MPP. The other `PVArray` block is set by the `MPPTController`, which is also subjected to a perturbation in the final stage of the simulation, as can be seen at the bottom of Figure 5.18a.

Finally, Figure 5.18b displays voltage, current and power values for both circuits. The static `PVArray` starts at the MPP but is no longer at it, once the temperature and irradiance conditions change by the end of the simulation. The `PVArray` block controlled by the `MPPTController` adapts to changing conditions.

5.3 Regression tests

A goal regarding verification and validation has been to provide a somewhat automatic process that can be run frequently and comfortably, in order to raise an early warning and insure that bugs don't remain unnoticed and unfixed for a long time.

This is achieved with the use of *regression tests*. Regression testing is a type of software testing which verifies that software which was previously developed and tested still performs correctly after it was changed or interfaced with other software. In contrast, non-regression testing aims to verify whether, after introducing or updating a given software application, the change has had the intended effect [Wik17].

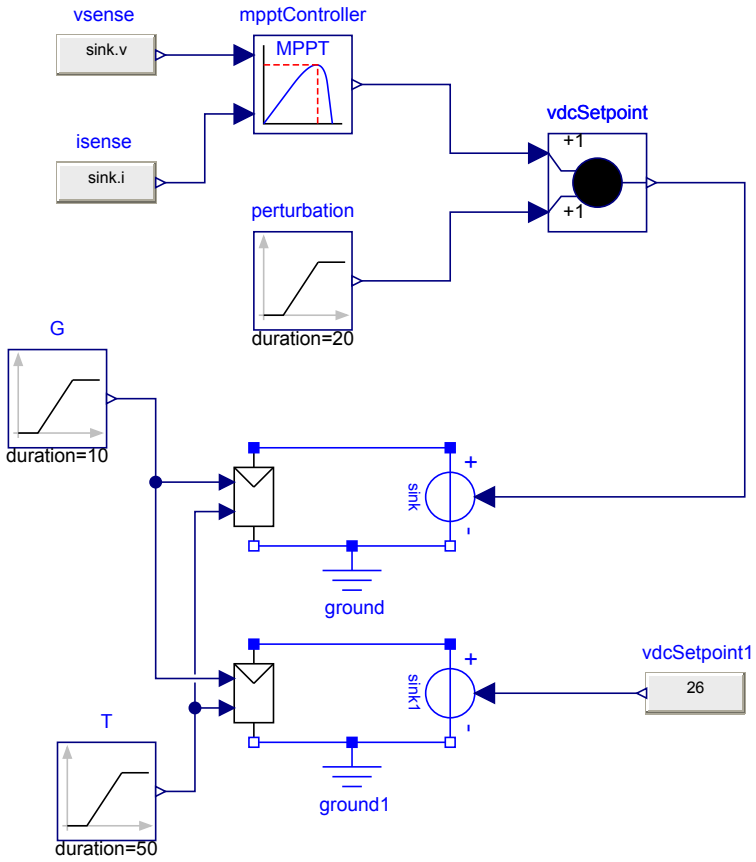
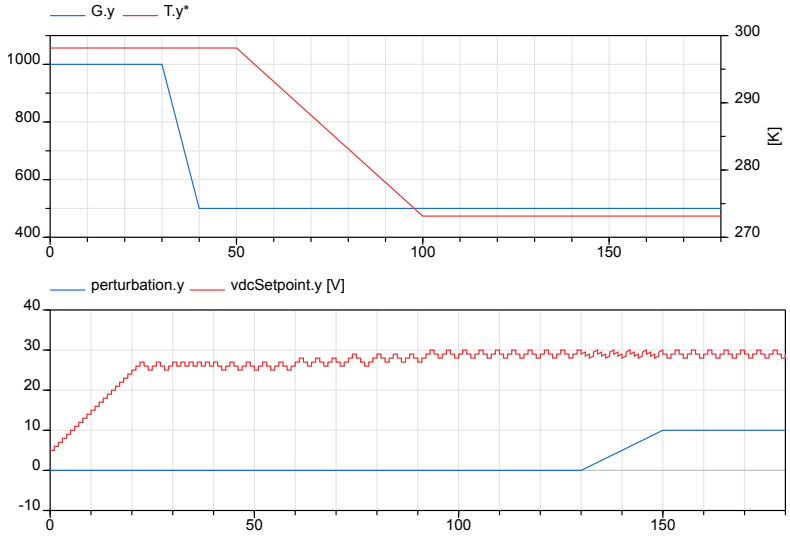
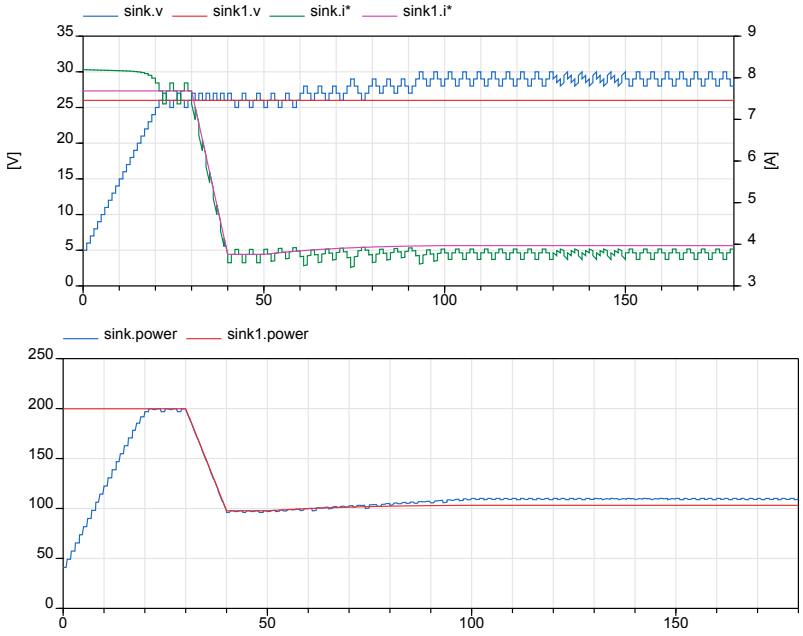


Figure 5.17: `MPPTControllerVerification` diagram



(a)



(b)

Figure 5.18: `MPPTControllerVerification` results

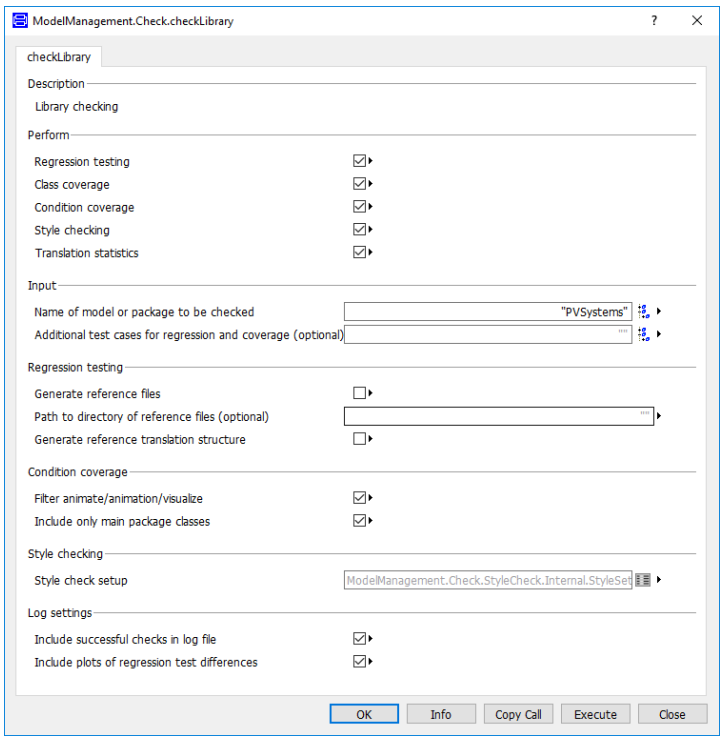


Figure 5.19: Dialog for the `checkLibrary` function

Regression testing is a very convenient way to provide verification for Modelica code. Test models are created (included in the `Verification` sub-package in `PVSystems`) and reference simulation results are generated and manually *validated* by the developer. Once those reference results have been obtained, future verifications can be performed automatically by the verification program, by comparing the new results with the reference ones.

One approach is to create Modelica models and scripts (`.mos` files) to perform these checks. This is indeed a popular approach and many solutions have already been implemented. One of those solutions is the `ModelManagement` library, included with the appropriate Dymola license.

This library provides the function `checkLibrary`, which can be called directly from the Graphical User Interface (GUI) in Dymola, by right-clicking and selecting Call Function..., spawning the dialog shown in Figure 5.19.

This function runs some checks on all or part of a selected library. It also runs regression tests on *system* models by simulating them and comparing the results with references. It assumes models having a `StopTime` annotation to be *system* models.

Apart from regression testing, this function provides some other features:

- Class coverage: providing information about the classes that were exercised during regression tests. This points out to the developer classes that might not have been tested.
- Condition coverage: at a lower level, some branches of `if` statements or other

conditional structures might have not been exercised.

- Style checking: providing checks pertaining the style of the Modelica code. This will include acceptable class names, appropriate documentation, etc.
- Translation statistics: using this option, the statistics of the translated model can be included in the regression testing to detect changes in, for example; the number and sizes of linear and nonlinear system of equations and the number of state variables.

Once the reference results have been generated, running the test on an intermediate version of `PVSystems` yielded the following report:

Library check log

TASKS

- Regression testing
- Model structure testing
- Class coverage analysis
- Condition coverage analysis

REGRESSION TEST RESULTS

- PVSystems.Examples.Application.BuckOpen: Validation ok. Structural validation ok. Translation time validation ok.
- PVSystems.Examples.Application.Inverter1phOpen: Validation ok. Structural validation ok. Translation time validation ok.
- PVSystems.Examples.Application.Inverter1phOpenSynch: Validation ok. Structural validation ok. Translation time validation ok.
- PVSystems.Examples.Application.Inverter1phClosedSynch: Validation ok. Structural validation ok. Translation time validation ok.
- PVSystems.Examples.Application.PVInverter1ph: Validation ok. Structural validation ok. Translation time validation ok.
- PVSystems.Examples.Application.PVInverter1phSynch: Validation ok. Structural validation ok. Translation time validation ok.
- PVSystems.Examples.Application.USBBatteryConverter: Validation ok. Structural validation ok. Translation time validation ok.
- PVSystems.Examples.Verification.IdealCBSwitchValidation: Validation ok. Structural validation ok. Translation time validation ok.
- PVSystems.Examples.Verification.MPPTControllerValidation: Validation ok. Structural validation ok. Translation time validation ok.
- PVSystems.Examples.Verification.PLLValidation: Validation ok. Structural validation ok. Translation time validation ok.
- PVSystems.Examples.Verification.PVArrayValidation: Validation ok. Structural validation ok. Translation time validation ok.
- PVSystems.Examples.Verification.SignalPWMValidation: Validation ok. Structural validation ok. Translation time validation ok.

- PVSystems.Examples.Verification.SwitchingCPMValidation: Validation ok. Structural validation ok. Translation time validation ok.

STATISTICS
26 tests performed on 13 test cases.
All tests ok, validation of PVSystems, successful.

CLASS COVERAGE
36 of 50 classes are covered by the test suite. 14 classes are not covered.
Of the not covered classes
14 are models
Listing classes in PVSystems

- package UsersGuide
- package References
- package ReleaseNotes
- package Examples
- package Application
  - model BuckOpen, used 1 time.
  - model Inverter1phOpen, used 1 time.
  - model Inverter1phOpenSynch, used 1 time.
  - model Inverter1phClosed, not used.
  - model Inverter1phClosedSynch, used 1 time.
  - model PVInverter1ph, used 1 time.
  - model PVInverter1phSynch, used 1 time.
  - model USBBatteryConverter, used 1 time.
- package Verification
  - model IdealCBSwitchValidation, used 1 time.
  - model MPPTControllerValidation, used 1 time.
  - model ParkValidation, not used.
  - model PLLValidation, used 1 time.
  - model PVArrayValidation, used 1 time.
  - model SignalPWMValidation, used 1 time.
  - model SwitchingCPMValidation, used 1 time.
  - model SimpleBatteryValidation, not used.
- package Electrical
  - model IdealCBSwitch, used 9 times.
  - model SW1, not used.
  - model SW2, not used.
  - model SW3, not used.
  - model CCM1, used 12 times.
  - model CCM2, not used.
  - model CCM3, not used.
  - model CCM4, not used.
  - model CCM5, not used.
  - model CCM_DCM1, used 1 time.

  - model CCM_DCM2, not used.
  - model PVArray, used 5 times.
  - model SimpleBattery, not used.
- package Assemblies
  - model HBridge, used 5 times.
    - model SwitchModel, used 10 times.
  - model HBridgeSwitched, used 2 times.
  - model BidirectionalBuckBoost, used 1 time.
    - model SwitchModel, used 2 times.
    - model CPMBidirectionalBuckBoost, used 1 time.
- package Interfaces
  - model BatteryInterface, not used.
  - model SwitchNetworkInterface, used 13 times.
    - model TwoPort, used 22 times.
- package Control
  - block SwitchingPWM, used 4 times.
  - block SwitchingCPM, used 1 time.
  - block DeadTime, used 2 times.
  - model CPM_CCM, used 2 times.
  - model CPM, not used.
  - block Park, used 8 times.
  - block InversePark, used 3 times.
  - block PLL, used 5 times.
  - block MPPTController, used 3 times.
  - package Assemblies
    - block Inverter1phCurrentController, used 3 times.
    - block Inverter1phCompleteController, used 2 times.
  - package Interfaces
    - model CPMInterface, used 2 times.
- package Icons

CONDITION COVERAGE

- `not (PV.v < 0) and PV.v > PV.Vocn` is always **false**
- `not (pVArray1.v < 0) and pVArray1.v > pVArray1.Vocn` is always **false**
- `not HBsw.idealCBSwitch.idealClosingSwitch.useHeatPort` is always **true**
- `not HBsw.idealCBSwitch.idealDiode.useHeatPort` is always **true**
- `not HBsw.idealCBSwitch1.idealClosingSwitch.useHeatPort` is always **true**
- `not HBsw.idealCBSwitch1.idealDiode.useHeatPort` is always **true**
- `not HBsw.idealCBSwitch2.idealClosingSwitch.useHeatPort` is always **true**
- `not HBsw.idealCBSwitch2.idealDiode.useHeatPort` is always **true**

- not HBsw.idealCBSwitch3.idealClosingSwitch.useHeatPort is always **true**
- not HBsw.idealCBSwitch3.idealDiode.useHeatPort is always **true**
- not R.useHeatPort is always **true**
- not Rav.useHeatPort is always **true**
- not Rbatt.useHeatPort is always **true**
- not Rdc.useHeatPort is always **true**
- not Rload.useHeatPort is always **true**
- not Rsw.useHeatPort is always **true**
- not conv.conv.inESR.useHeatPort is always **true**
- ...

The first part of the report lists the classes on which regression tests are performed, as well as their results. In this run, all regression tests turn out to be successful.

The next section reports on class coverage. Things are not going so well, since only 36 out of the 50 classes are being exercised by the tests. It turns out that even several models from the `Application` and `Verification` packages are not run, which is not what was intended. This is due to these models not including the `StopTime` annotation, since the default time of 1 s was used. This needs to be fixed.

Additionally, many of the models from `Electrical` and some from `Control` are not tested either since they are not called by the models that were run. This is not ideal, since bugs in these classes will go unnoticed.

The final section goes deeper and lists the conditional branches that remained inactive in classes that *were* tested. This will also provide a hiding place for bugs.

The library style check log, checking for missing documentation, invalid class names and other style issues, is the following:

Library style check log

- PVSystems.UsersGuide
  - Documentation missing
- PVSystems.UsersGuide.References
  - Documentation missing
- PVSystems.UsersGuide.References.EM01
  - Documentation missing
- PVSystems.UsersGuide.References.EMA16
  - Documentation missing
- PVSystems.UsersGuide.References.TDD07

  - Documentation missing
- PVSystems.UsersGuide.References.VGF09
  - Documentation missing
- PVSystems.UsersGuide.ReleaseNotes, Check ok
- PVSystems.UsersGuide.ReleaseNotes.Version_0_6_1
  - Bad class name Name shall not contain '_'.
- PVSystems.UsersGuide.ReleaseNotes.Version_0_6_0
  - Bad class name Name shall not contain '_'.
- PVSystems.UsersGuide.Contact, Check ok
- PVSystems.UsersGuide.License, Check ok
- PVSystems.Examples
  - Documentation missing
- PVSystems.Examples.Application
  - Documentation missing
- PVSystems.Examples.Application.BuckOpen, Check ok
- PVSystems.Examples.Application.Inverter1phOpen, Check ok
- PVSystems.Examples.Application.Inverter1phOpenSynch, Check ok
- PVSystems.Examples.Application.Inverter1phClosed, Check ok
- PVSystems.Examples.Application.Inverter1phClosedSynch
  - Documentation missing
- PVSystems.Examples.Application.PVInverter1ph
  - Documentation missing
- PVSystems.Examples.Application.PVInverter1phSynch
  - Documentation missing
- PVSystems.Examples.Application.USBBatteryConverter
  - Documentation missing
- PVSystems.Examples.Verification
  - Documentation missing
- PVSystems.Examples.Verification.IdealCBSwitchValidation, Check ok
- PVSystems.Examples.Verification.MPPTControllerValidation, Check ok
- PVSystems.Examples.Verification.ParkValidation, Check ok
- PVSystems.Examples.Verification.PLLValidation, Check ok
- PVSystems.Examples.Verification.PVArrayValidation, Check ok
- PVSystems.Examples.Verification.SignalPWMValidation, Check ok
- PVSystems.Examples.Verification.SwitchingCPMValidation, Check ok
- PVSystems.Examples.Verification.SimpleBatteryValidation
  - Documentation missing
- PVSystems.Electrical
  - Documentation missing
- PVSystems.Electrical.IdealCBSwitch, Check ok
- PVSystems.Electrical.SW1
  - Documentation missing
- PVSystems.Electrical.SW2

  - Documentation missing
- PVSystems.Electrical.SW3
  - Documentation missing
- PVSystems.Electrical.CCM1, Check ok
- PVSystems.Electrical.CCM2, Check ok
- PVSystems.Electrical.CCM3, Check ok
- PVSystems.Electrical.CCM4, Check ok
- PVSystems.Electrical.CCM5, Check ok
- PVSystems.Electrical.CCM_DCM1
  - Bad class name Name shall not contain '_'.
- PVSystems.Electrical.CCM_DCM2
  - Bad class name Name shall not contain '_'.
- PVSystems.Electrical.PVArray, Check ok
- PVSystems.Electrical.SimpleBattery
  - Documentation missing
- PVSystems.Electrical.SimpleBattery.BatteryCapacity, Check ok
- PVSystems.Electrical.Assemblies
  - Documentation missing
- PVSystems.Electrical.Assemblies.HBridge, Check ok
- PVSystems.Electrical.Assemblies.HBridge.SwitchModel
  - Documentation missing
  - Class description string missing
- PVSystems.Electrical.Assemblies.HBridgeSwitched, Check ok
- PVSystems.Electrical.Assemblies.BidirectionalBuckBoost
  - (Possibly) insufficient documentation
- PVSystems.Electrical.Assemblies.BidirectionalBuckBoost.SwitchModel
  - Documentation missing
  - Class description string missing
- PVSystems.Electrical.Assemblies.CPMBidirectionalBuckBoost
  - (Possibly) insufficient documentation
- PVSystems.Electrical.Interfaces
  - Documentation missing
- PVSystems.Electrical.Interfaces.BatteryInterface
  - (Possibly) insufficient documentation
- PVSystems.Electrical.Interfaces.SwitchNetworkInterface
  - Documentation missing
- PVSystems.Electrical.Interfaces.TwoPort
  - Documentation missing
- PVSystems.Control
  - Documentation missing
- PVSystems.Control.SwitchingPWM, Check ok
- PVSystems.Control.SwitchingCPM, Check ok
- PVSystems.Control.DeadTime, Check ok

- PVSystems.Control.CPM_CCM
  - Bad class name Name shall not contain '_'.
- PVSystems.Control.CPM, Check ok
- PVSystems.Control.Park, Check ok
- PVSystems.Control.InversePark, Check ok
- PVSystems.Control.PLL
  - Description string missing for
    - frequency
- PVSystems.Control.MPPTController, Check ok
- PVSystems.Control.Assemblies, Check ok
- PVSystems.Control.Assemblies.Inverter1phCurrentController, Check ok
- PVSystems.Control.Assemblies.Inverter1phCompleteController, Check ok
- PVSystems.Control.Interfaces
  - (Possibly) insufficient documentation
- PVSystems.Control.Interfaces.CPMInterface
  - (Possibly) insufficient documentation
- PVSystems.Icons, Check ok
- PVSystems.Icons.AssembliesPackage, Check ok
- PVSystems.Icons.ConverterIcon
  - Documentation missing

In order to get closer to the one-click testing ideal, a small caller script is created (Listing 5.1) and made accessible through the GUI using a Dymola annotation in the root package of the library that makes the script executable from the Commands dialog.

Listing 5.1: Resources/Scripts/Dymola/callCheckLibrary.mos

```
1 cd("Resources/Tests");
2 ModelManagement.Check.checkLibrary(name="PVSystems", referenceFileDirectory="RefData")
```

Making it easier to run tests will increase the probability that they will be run. Maximizing coverage and minimizing run time are two other important aspects to focus on.

6 Application

6.1 Introduction

The `PVSystems` library has a healthy amount of example models (Figure 6.1) that showcase application of the library component and assembly models. Some of the examples are also specifically created for verification purposes through the use of regression tests (Section 5.3).

The examples are ordered in increasing complexity. This chapter will present an in-depth discussion of three notable examples.

6.2 Open-loop buck converter

The buck converter is a switched converter used to produce a reduced DC output voltage from a given DC input voltage. Figure 6.2a presents the circuit diagram of a typical buck converter implementation. The transistor will be governed by a PWM signal with a duty cycle D and a period T_s, as shown in Figure 6.2b. Lets assume that the signal logic is such that during the first interval, $D\,T_s$, the transistor is closed and the diode is open. This results in a circuit like the one displayed in Figure 6.2c. During the second interval, $D'\,T_s$, the positions are switched, having the transistor open and the diode closed, resulting in the circuit displayed in Figure 6.2d.

In general terms, the equations for the state variables (inductor voltage and capacitor current) will be the following:

$$\begin{cases} v_L = V_g - v_o \\ i_c = i_L - \dfrac{v_o}{R} \end{cases} \quad \text{during } D\,T_s \tag{6.1}$$

$$\begin{cases} v_L = -v_o \\ i_c = i_L - \dfrac{v_o}{R} \end{cases} \quad \text{during } D'\,T_s \tag{6.2}$$

In order to perform the DC analysis of the converter, lets assume CCM and apply the small-ripple approximation. In steady-state, volt-seconds balance in the inductor and charge balance in the capacitor yield the following equations:

$$0 = \langle v_L \rangle = D\,V_g - V_o \tag{6.3}$$

$$0 = \langle i_c \rangle = I_L - \frac{V_o}{R} \tag{6.4}$$

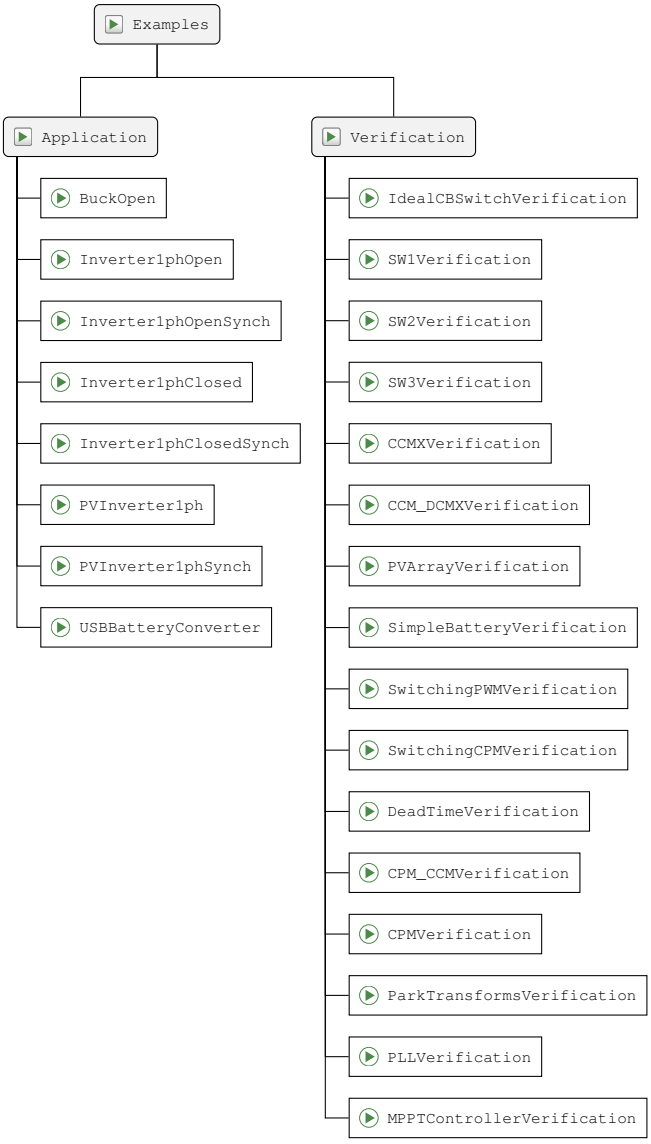


Figure 6.1: Examples in `PVSystems` library

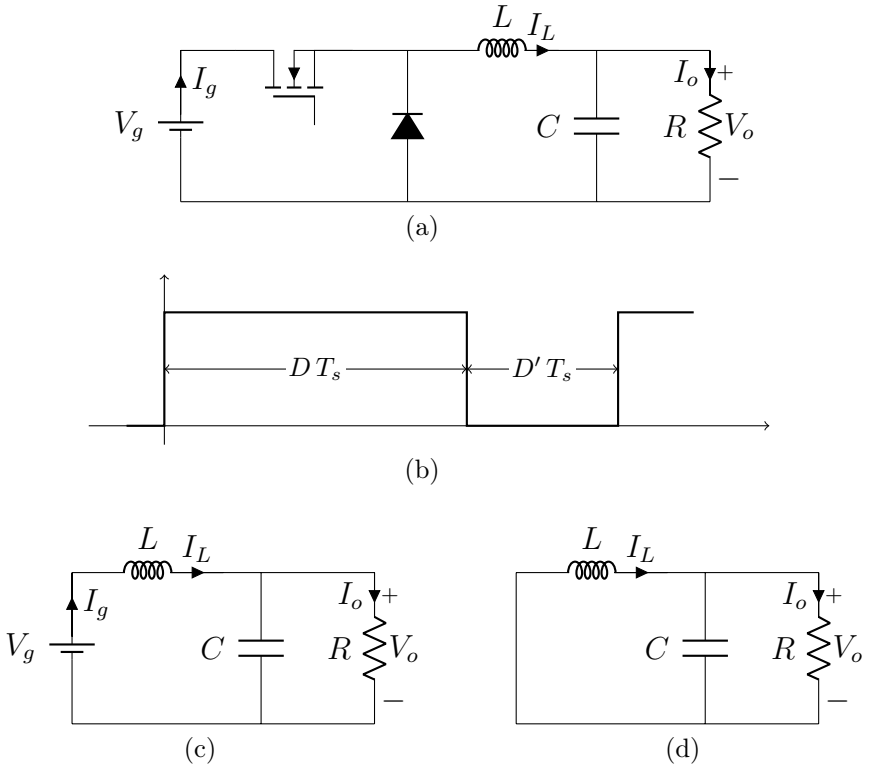


Figure 6.2: Buck converter diagrams: (a) typical implementation, (b) PWM signal, (c) circuit state during $D\,T_s$ interval, (d) circuit state during $D'\,T_s$.

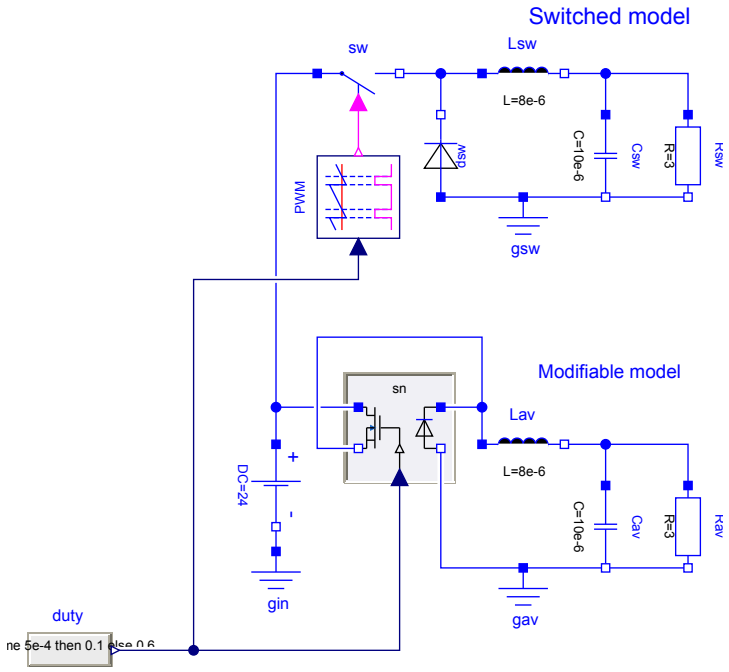


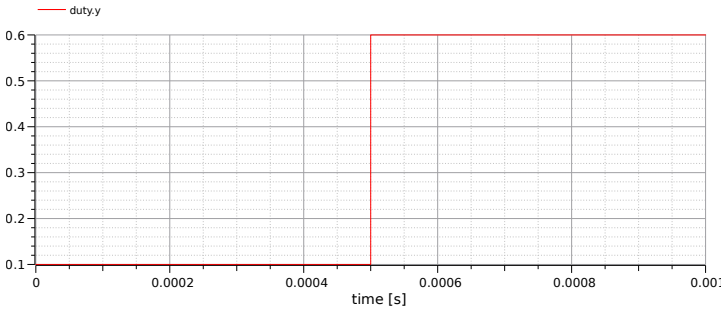
Figure 6.3: `BuckOpen` diagram

An interesting feature is that DC analysis results in CCM don't depend on the inductance or capacitance values. These will play a role in determining the dynamic properties of the system. In the case of DCM, the conversion ratio can be obtained as [EM01]:

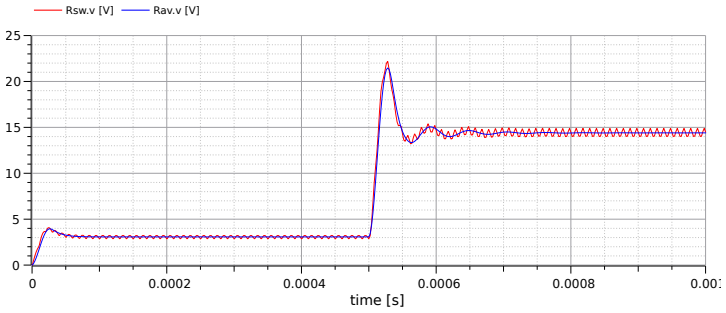
$$\frac{V_o}{V_g} = \frac{2}{1 + \sqrt{1 + \frac{8\,L}{R\,D_1^2\,T_s}}} \tag{6.5}$$

The diagram for `BuckOpen` is presented in Figure 6.3. It includes a switched and a modifiable version of the converter. The modifiable version can be made to instantiate a switched or an averaged model. Out of the box, the switch network is configured with `CCM1`, that is, averaged and no losses. Running the simulation and plotting duty cycle, output voltage and inductor current produces Figure 6.3.

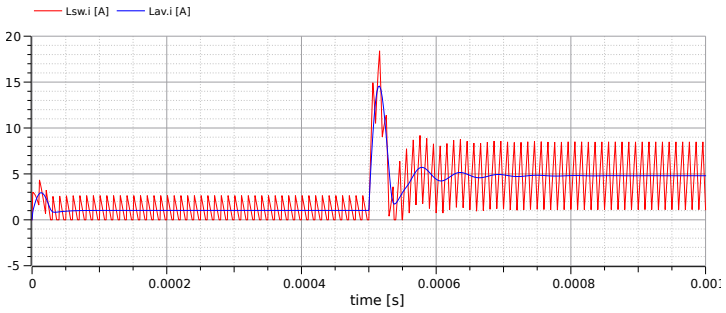
With either Dymola or OpenModelica, one can measure the value of the average voltage output as 3.07 V in DCM operation and 14.4 V in CCM operation, which is exactly what can be obtained from the analytical results using (6.5) and (6.3), respectively. The average inductor current corresponds with the average load current, which can be obtained by dividing those output voltage values by the resistor value. This gives 1.02 A and 4.8 A, respectively, which also agrees exactly with simulation results.



(a)



(b)



(c)

Figure 6.4: `BuckOpen` simulation results: (a) duty cycle, (b) output voltage, (c) inductor current.

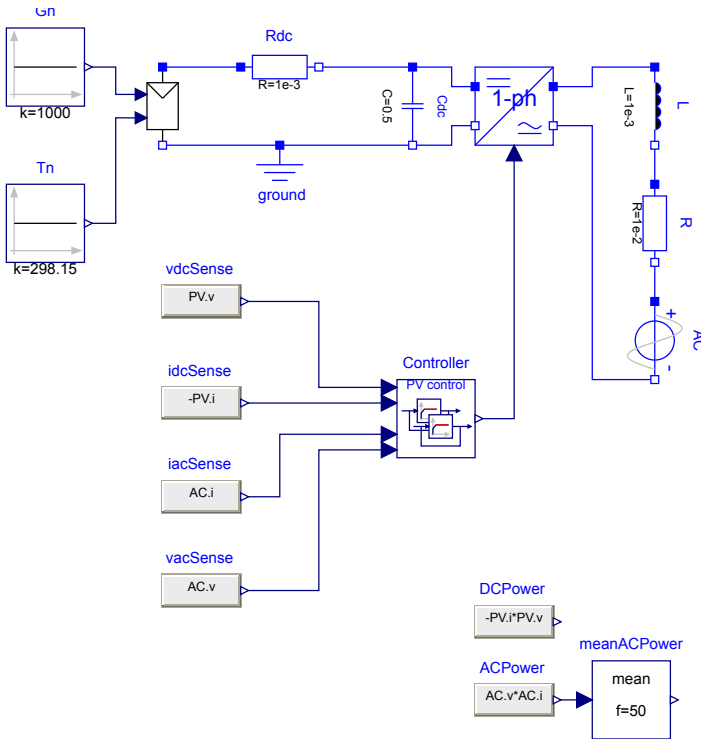


Figure 6.5: `PVInverter1phSynch` diagram

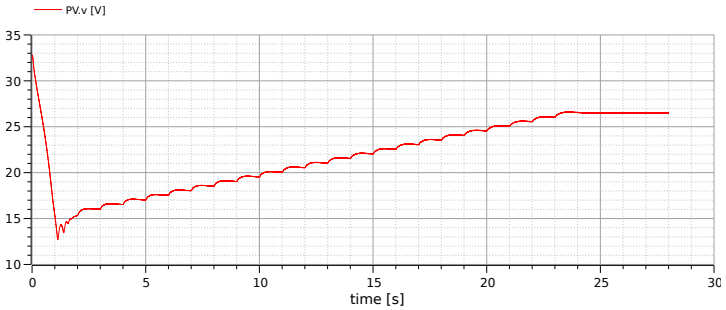
6.3 Grid-tied PV inverter

This example provides a model of a very simple single-phase grid-tied photovoltaic system (Figure 6.5). On the DC side, a single PV panel is modelled with `PVArray` placed in series with a small resistor. A capacitor is used to model the capacitance placed in the DC bus of the inverter. On the AC side, the output filter is modelled with an inductor in series with a small resistor. The grid is modelled with an ideal voltage source.

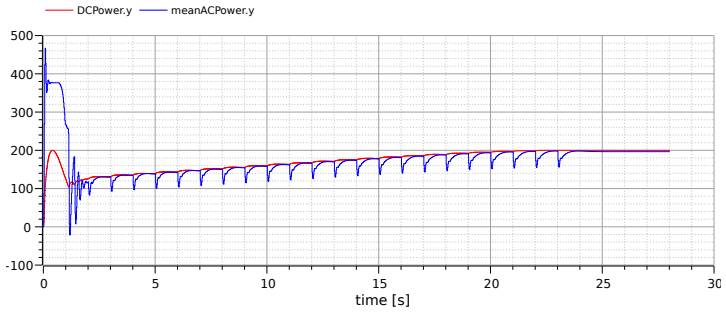
The control is implemented with an instance of the `Inverter1phCompleteController`, available as an assembly in the `Control` package. For more details on this block, see Section 4.3.2.

The simulation is configured to run for 28 s. The grid frequency is set at 50 Hz, which makes this time quite long compared with the time-scale of the grid dynamics. On the other hand, the MPPT control and corresponding power changes are on the scale of seconds, which is why the stop time is set at that value, in order to observe some interesting dynamics at that level.

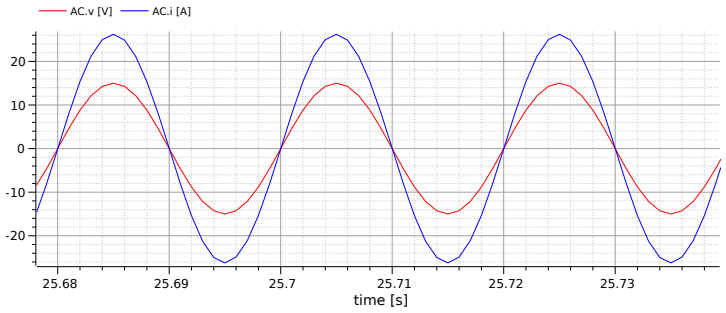
Figure 6.6a displays the variation of the DC voltage imposed by the inverter, controlled by the MPPT and current controller loop. The steps in voltage correspond to the adjustments that the MPPT controller is making (following the P&O algorithm). These steps are translated into the power steps shown in Figure 6.6b. Eventually, since the irradiance and ambient temperature conditions are not changing, the controller finds the MPP. This is close to the 200 W of the default `PVArray` parameter values. Finally, Figure 6.6c displays a close-up of the output voltage and current, to emphasize the fact



(a)



(b)



(c)

Figure 6.6: `PVInverter1phSynch` simulation results: (a) PV array voltage, (b) input and output power, (c) grid voltage and output current.

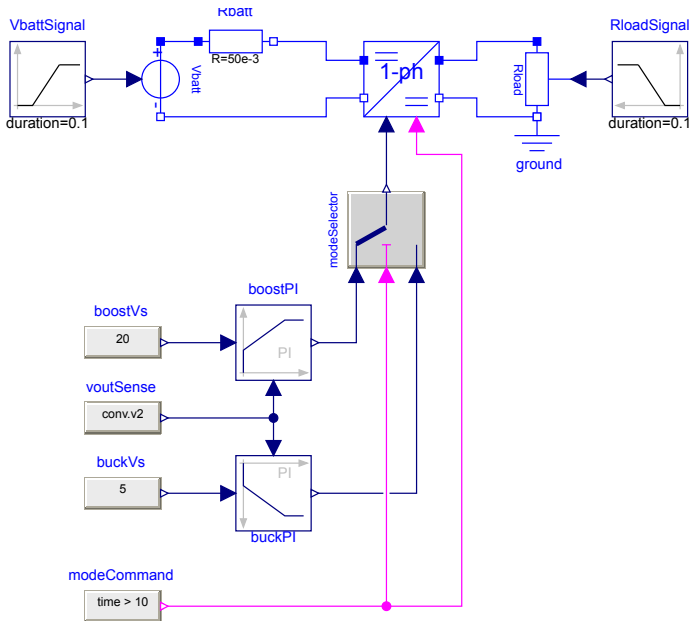


Figure 6.7: `USBBatteryConverter` diagram

that the output current is controlled to stay in phase with the grid voltage, providing only active power to the grid.

6.4 USB battery converter

A battery, simulated with a controlled voltage source in series with a small resistance, is interfaced with a USB device, simulated with a resistive load. The converter is a component included in the `Electrical.Assemblies` package and is discussed in Section 4.2.6. Figure 6.7 presents the block diagram.

This example is borrowed from [EMA16]. The application is not that related with photovoltaics, but provides a good showcase of the power electronics models in this library. The converter is specified to have three operating modes:

1. Battery voltage 12.6 V, USB voltage 5 ± 0.1 V at 2 A, converter supplies bus.
2. Battery voltage 9.6 V, USB voltage 20 ± 0.1 V at 3 A, converter supplies bus.
3. Battery voltage 11.1 V, USB voltage 20 V, bus supplies 60 W to charge battery.

An efficient solution to these step-down and bidirectional step-up requirements is a non-inverting buck-boost converter with bi-directional switches operated in a buck/boost modal fashion (i.e. the boost switches are disabled while in buck mode and vice versa). A possible solution to these requirements using this topology is expressed through the parametrization of `CPMBidirectionalBuckBoost` (Figure 6.8).

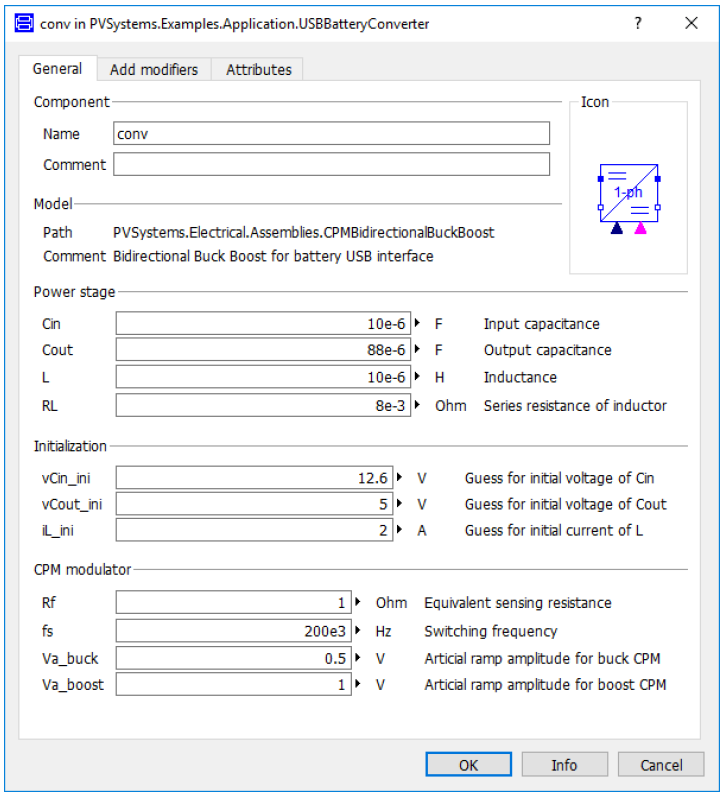
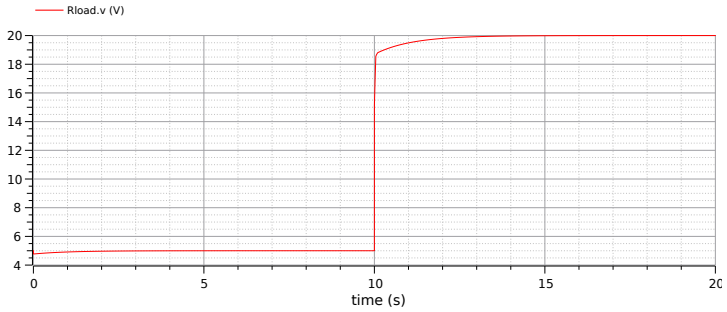
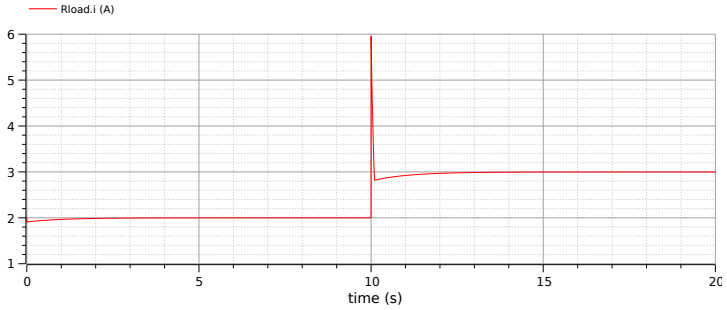


Figure 6.8: Converter parametrization in `USBBatteryConverter`



(a)



(b)

Figure 6.9: `USBBatteryConverter` simulation results: (a) output voltage, (b) output current.

This converter model includes both the electrical and control components of a CPM controlled modal non-inverting buck-boost. The default stop time is set at 20 s. Running the simulation and plotting the output voltage and current produces Figure 6.9.

7 Conclusions and future work

7.1 Conclusions

Technology creates abundance. This point was made in the opening chapter, with the story of how aluminium was scarce before the right technology made it abundant. The decision to create a Modelica library for photovoltaic systems with a focus on power electronics and with a model for battery energy storage was based on the belief that these technologies will be at the centre of an energy abundant future. Providing tools for scientists and engineers to develop these technologies is one strategy to accelerate the advent of this future.

Modelica supports this purpose perfectly. With it's open non-propietary nature, it's readily available to users and tool developers without costly fees or complicated license agreements. This kind of openness can drive a thriving ecosystem that eventually creates solutions that surpass the commercially available traditional ones.

Modelica is also developed with composability and reusability in mind, with the inclusion of object orientation and acausal equation based modelling and connection of components. This makes it easy to create models integrating several physical domains.

List of tasks accomplished

The following list presents a comprehensive list of the tasks accomplished in this work:

- A brief review of the current PV energy market and trends as well as of the technology was performed and an overview provided in Section 2.2.
- A brief review of the current M&S technology, practices and tools was performed and an overview provided in Section 2.3.
- An overview of the Modelica language and ecosystem was provided in Section 2.4. This includes a review of existing Modelica libraries relevant to the PV domain.
- A discussion of the generic PV source model from [VGF09] was presented in Section 3.1.
- Some basic concepts related to power electronics switches and topologies were presented in Section 3.2. Notably, this includes the switch network concept from [EMA16], presented in Section 3.2.2, which provides a versatile way of creating averaged power converter models.
- A discussion of the generic Battery Energy Storage (BES) model from [TDD07] was presented in Section 3.3.

- A basic collection of control elements relevant to the PV and power electronics domains is described in Section 3.4.
- All these models are implemented in the Modelica language, to conform the `PVSystems` library.
- The source code of the library is made available under the MIT license in a GitHub repository [Rod17a].
- An HTML version of the documentation of the library is exported using Dymola and made available online at [Rod17b].
- A thorough description of the library is compiled and presented in Chapter 4.
- The fitness of these models to represent PV systems is demonstrated with a collection of application and verification examples, presented in Chapter 6 and Section 5.2, respectively.
- Throughout the development, an effort was made to explore and apply best software and model development practices. This resulted in the use of Git for version control, the creation of a fairly automated test strategy and the application of ideas from [Til17] for the design of the library structure and architecture. A brief discussion of this topic is presented in Section 5.1.1 and the use of regression tests is discussed in Section 5.3.

List of conclusions

Following is a list of conclusions that were drawn from the work performed in this thesis:

- All of the relevant behaviours of the elements included in the original scope of the library (photovoltaic array, power electronics and battery energy storage) could be modelled with the use of equivalent circuits. Many physical processes where left out, but the goal of providing models for converter control development is achieved with this level of modelling.
- The switch network concept becomes really valuable when modelling the power electronics components, since it provides a generalized and composable structure for the modelling of power converters. Coupled with Modelica language features like `redeclare`, it also provides a way of creating very user-friendly models that can be instantiated as switched or averaged variants.
- Taking into account the implicit size and time scales that were assumed as a consequence of the goal of the library (supporting converter control development), it seems that the modelling of battery energy storage might not even be required. The reason for this is that BES is normally designed with capacities that take several hours to charge or discharge. Typically, this is a time range that a converter control software developer wouldn't want to simulate, even when using the faster averaged models. For the time range relevant for this application, a constant voltage source might be good enough.

- Regression testing in Dymola can provide a very quick and convenient way to verify the correctness of models as changes are made. To be fair, this is true when considering changes that don't substantially change the functionality of a given model and assuming the reference data generated represents a run of a correct model. Regression tests will be most helpful in mature projects in which changes mainly correspond to optimizations of the code.

- From experience and from observing other Modelica libraries [HG14] and [Mod17b], it seems that packaging the collection of Verification and Validation (V&V) examples in a separate Modelica library would be better than the approach taken in this work (creating a `Verification` package inside the `Examples` package). The package quickly becomes crowded and increases the size of the library. These are not typically models that the end user will be interested in. Additionally, by using a separate package, verification can be performed against models created with other Modelica libraries without introducing extra dependencies in the main Modelica library.

- The development of the library has made evident the power of some of the features of the Modelica language like object-orientation and acausal equation based modelling. These features together with the use of the `redeclare` directive enabled a nice extensible and user-friendly architecture of power electronics modelling based on the switch network concept.

7.2 Future work

The list of possible extensions and enhancements that could be made to `PVSystems` is too big to fit in this document. Some of the most interesting explorations are listed here:

- Some of the features available in Modelica have not been explored and the quality of the library could probably improve with their use. For example, records could be used to group parameters and provide an easy way to parametrize models with values corresponding to commercially available products. This approach would probably be very productive in `PVArray` and `SimpleBattery`.

- Models should be extended from single-phase to multi-phase. Many of the photovoltaic systems are three-phase. This extension should probably be made taking into account the multi-phase interfaces included already in MSL.

- New models would probably be demanded by users wanting to model typical photovoltaic systems which, in many occasions, form part of micro-grids that also include a line transformer and an induction machine and synchronous generator.

- The available models could be made more sophisticated by making losses explicit in a thermal port.

- The library would benefit from feedback from users. Some attempts have been made to promote the library with not a lot of success. The feedback from users could be used to validate that the library is in fact accomplishing its goal and to improve and extend it.

- The averaged models of power electronics could be used to perform linear analysis, using Dymola 2017, to support the use of techniques from classical control. Examples should be provided where Bode plots, transfer functions and poles and zeros are obtained.

- The use of the Functional Mock-up Interface (FMI) standard was explored by trying to generate exported Functional Mock-up Unit (FMU) versions of some of the models, for use in other tools. This effort was not successful, but having this kind of application example would be quite powerful, since it would showcase a use case that would be quite popular - having the model developed in Modelica but used in some other tool (MATLAB, LabVIEW) where the control software is developed.

- V&V would greatly benefit from extending two approaches to more models in the library. Models of the same systems could be developed in another tool like LTspice or with other Modelica libraries like [Kra17] and [FW16], to gain a stronger conviction that the results are correct. The library could also be used to model real-life systems that are in operation, as opposed to examples from the literature, to compare simulation and experimental results. As mentioned in Section 7.1, the library would also benefit from using a separate V&V library.

- The averaged models based on the switch network concept would probably be helpful in projects outside of the PV domain. Since they would probably have wide appeal, inclusion in the MSL could be explored.

Bibliography

Chapter 1: Introduction, goals and structure

[16] *Modelica*. en. Page Version ID: 747136840. Oct. 2016. (Visited on 01/02/2017).

[DK14] Peter H. Diamandis and Steven Kotler. *Abundance: The Future Is Better Than You Think*. English. Reprint edition. New York: Free Press, Sept. 2014. ISBN: 978-1-4516-1683-5.

[EAB14] Omar Ellabban, Haitham Abu-Rub, and Frede Blaabjerg. "Renewable Energy Resources: Current Status, Future Prospects and Their Enabling Technology". en. In: *Renewable and Sustainable Energy Reviews* 39 (Nov. 2014), pp. 748–764. ISSN: 13640321. DOI: 10.1016/j.rser.2014.07.113. (Visited on 12/28/2016).

[Lie+14] Grischa Liebel et al. "Assessing the State-of-Practice of Model-Based Engineering in the Embedded Systems Domain". In: *International Conference on Model Driven Engineering Languages and Systems*. Springer, 2014, pp. 166–182.

[Mus+14] Gunter Mussbacher et al. "The Relevance of Model-Driven Engineering Thirty Years from Now". In: *International Conference on Model Driven Engineering Languages and Systems*. Springer, 2014, pp. 183–200.

[Tod16] Benjamin Todd. *80,000 Hours: Find a Fulfilling Career That Does Good*. English. Nov. 2016.

[Urb15] Tim Urban. *How Tesla Will Change The World*. http://waitbutwhy.com/2015/06/how-tesla-will-change-your-life.html. June 2015. (Visited on 01/03/2017).

[Vic15] Bret Victor. *What Can a Technologist Do about Climate Change? A Personal View*. http://worrydream.com/ClimateChange/. Nov. 2015. (Visited on 12/03/2016).

Chapter 2: Technology review

[ÅEM98] Karl Johan Åström, Hilding Elmqvist, and Sven Erik Mattsson. "Evolution of Continuous-Time Modeling and Simulation." In: *ESM*. 1998, pp. 9–18.

[All+15] Jonas Allegrini et al. "A Review of Modelling Approaches and Tools for the Simulation of District-Scale Energy Systems". en. In: *Renewable and Sustainable Energy Reviews* 52 (Dec. 2015), pp. 1391–1404. ISSN: 13640321. DOI: 10.1016/j.rser.2015.07.123. (Visited on 12/27/2016).

[Ein+11] M. Einhorn et al. "A Modelica Library for Simulation of Electric Energy Storages". In: *Proceedings of the 8th International Modelica Conference; March 20th-22nd; Technical Univeristy; Dresden; Germany*. Linköping University Electronic Press, 2011, pp. 436–445. DOI: 10.3384/ecp11063436.

[FW14] Rüdiger Franke and Hansjürg Wiesmann. "Flexible Modeling of Electrical Power Systems – the Modelica PowerSystems Library". In: Mar. 2014, pp. 515–522. DOI: 10.3384/ecp14096515. (Visited on 12/16/2016).

[FW16] Rüdiger Franke and Hansjürg Wiesmann. *Modelica/PowerSystems*. 2016. (Visited on 12/16/2016).

[HK14] Anton Haumer and Christian Kral. "The New EDrives Library: A Modular Tool for Engineering of Electric Drives". In: Mar. 2014, pp. 155–163. DOI: 10.3384/ecp14096155. (Visited on 01/07/2017).

[Kop16] R. H. E. M. Koppelaar. "Solar-PV Energy Payback and Net Energy: Meta-Assessment of Study Quality, Reproducibility, and Results Harmonization". In: *Renewable and Sustainable Energy Reviews* (2016). ISSN: 1364-0321. DOI: 10.1016/j.rser.2016.10.077. (Visited on 01/04/2017).

[Kra17] Christian Kral. *PhotoVoltaics - Modelica Library for the Simulation of Photo Voltaic Cells and Modules*. 2017. (Visited on 12/20/2016).

[Mad] Morten Vesterager Madsen. *Solar Cells - the Three Generations*. http://plasticphotovoltaics.org/lc/lc-solarcells/lc-introduction.html. (Visited on 01/04/2017).

[Mod17a] Modelica Association. *ElectricalEnergyStorage: Free Library That Contains Models with Different Complexity for Simulating of Electric Energy Storages like Batteries (Single Cells as Well as Stacks) Interacting with Loa.*. Apr. 2017. (Visited on 09/04/2017).

[MT16] Khizir Mahmud and Graham E. Town. "A Review of Computer Tools for Modeling Electric Vehicle Energy Requirements and Their Impact on Power Distribution Networks". en. In: *Applied Energy* 172 (June 2016), pp. 337–359. ISSN: 03062619. DOI: 10.1016/j.apenergy.2016.03.100. (Visited on 12/27/2016).

[Rfa14] Rfassbind. *From a Solar Cell to a PV System. Diagram of the Possible Components of a Photovoltaic System*. By Rfassbind - Own work., Public Domain, https://commons.wikimedia.org/w/index.php?curid=34961018. Aug. 2014. (Visited on 01/05/2017).

[SC14] Sunanda Sinha and S.S. Chandel. "Review of Software Tools for Hybrid Renewable Energy Systems". en. In: *Renewable and Sustainable Energy Reviews* 32 (Apr. 2014), pp. 192–205. ISSN: 13640321. DOI: 10.1016/j.rser.2014.01.035. (Visited on 12/27/2016).

[Sme+16] Arno Smets et al. *Solar Energy: The Physics and Engineering of Photovoltaic Conversion, Technologies and Systems*. English. UIT Cambridge, Jan. 2016.

Chapter 3: Photovoltaic systems modelling

[EM01] Robert W. Erickson and Dragan Maksimović. *Fundamentals of Power Electronics*. English. Google-Books-ID: S91G_Nz_KjEC. Springer Science & Business Media, Jan. 2001. ISBN: 978-0-7923-7270-7.

[EMA16] Robert W. Erickson, Dragan Maksimović, and Khurram Afridi. *Power Electronics Specialization*. https://www.coursera.org/specializations/power-electronics. 2016. (Visited on 01/13/2017).

[Kyo17] Kyocera. *Kyocera KC200GT Data-Sheet*. 2017. (Visited on 01/10/2017).

[TDD07] O. Tremblay, L. A. Dessaint, and A. I. Dekkiche. "A Generic Battery Model for the Dynamic Simulation of Hybrid Electric Vehicles". In: *2007 IEEE Vehicle Power and Propulsion Conference*. Sept. 2007, pp. 284–289. DOI: 10.1109/VPPC.2007.4544139.

[TLR11] Remus Teodorescu, Marco Liserre, and Pedro Rodriguez. *Grid Converters for Photovoltaic and Wind Power Systems*. en. John Wiley & Sons, July 2011. ISBN: 978-1-119-95720-1.

[VGF09] M. G. Villalva, J. R. Gazoli, and E. R. Filho. "Comprehensive Approach to Modeling and Simulation of Photovoltaic Arrays". In: *IEEE Transactions on Power Electronics* 24.5 (May 2009), pp. 1198–1208. ISSN: 0885-8993. DOI: 10.1109/TPEL.2009.2013862.

Chapter 4: `PVSystems` library

[EMA16] Robert W. Erickson, Dragan Maksimović, and Khurram Afridi. *Power Electronics Specialization*. https://www.coursera.org/specializations/power-electronics. 2016. (Visited on 01/13/2017).

[Til17] Michael M. Tiller. *Modelica by Example*. 2017. (Visited on 01/26/2017).

Chapter 5: Development and verification

[Das17] Dassault Systèmes AB. *Dymola*. 2017. (Visited on 05/24/2017).

[EMA16] Robert W. Erickson, Dragan Maksimović, and Khurram Afridi. *Power Electronics Specialization*. https://www.coursera.org/specializations/power-electronics. 2016. (Visited on 01/13/2017).

[IEE+12] IEEE Computer Society et al. *IEEE Standard for System and Software Verification and Validation*. English. OCLC: 806965114. New York: Institute of Electrical and Electronics Engineers, 2012. ISBN: 978-0-7381-7268-2. (Visited on 05/25/2017).

[Ope17] Open Source Modelica Consortium. *OpenModelica*. 2017. (Visited on 05/24/2017).

[Rod17a] Raúl Rodríguez Pearson. *PVSystems: A Modelica Library for Photovoltaic System and Power Converter Design*. Apr. 2017. (Visited on 05/24/2017).

[Rod17b] Raúl Rodríguez Pearson. *PVSystems Documentation*. https://raulrpearson.github.io/PVSystems/. 2017. (Visited on 05/24/2017).

[TDD07] O. Tremblay, L. A. Dessaint, and A. I. Dekkiche. "A Generic Battery Model for the Dynamic Simulation of Hybrid Electric Vehicles". In: *2007 IEEE Vehicle Power and Propulsion Conference*. Sept. 2007, pp. 284–289. DOI: 10.1109/VPPC.2007.4544139.

[VGF09] M. G. Villalva, J. R. Gazoli, and E. R. Filho. "Comprehensive Approach to Modeling and Simulation of Photovoltaic Arrays". In: *IEEE Transactions on Power Electronics* 24.5 (May 2009), pp. 1198–1208. ISSN: 0885-8993. DOI: 10.1109/TPEL.2009.2013862.

[Wik17] Wikimedia Foundation. *Regression Testing*. en. Page Version ID: 778065098. Apr. 2017. (Visited on 05/25/2017).

Chapter 6: Application

[EM01] Robert W. Erickson and Dragan Maksimović. *Fundamentals of Power Electronics*. English. Google-Books-ID: S91G_Nz_KjEC. Springer Science & Business Media, Jan. 2001. ISBN: 978-0-7923-7270-7.

[EMA16] Robert W. Erickson, Dragan Maksimović, and Khurram Afridi. *Power Electronics Specialization*. https://www.coursera.org/specializations/power-electronics. 2016. (Visited on 01/13/2017).

Chapter 7: Conclusions and future work

[EMA16] Robert W. Erickson, Dragan Maksimović, and Khurram Afridi. *Power Electronics Specialization*. https://www.coursera.org/specializations/power-electronics. 2016. (Visited on 01/13/2017).

[FW16] Rüdiger Franke and Hansjürg Wiesmann. *Modelica/PowerSystems*. 2016. (Visited on 12/16/2016).

[HG14] Hawaii Natural Energy Institute and Georgia Tech Research Corporation. *FCSys - Modelica Library of Fuel Cell Models*. 2014. (Visited on 12/20/2016).

[Kra17] Christian Kral. *PhotoVoltaics - Modelica Library for the Simulation of Photo Voltaic Cells and Modules*. 2017. (Visited on 12/20/2016).

[Mod17b] Modelica Association. *Modelica Standard Library*. 2017. (Visited on 12/16/2016).

[Rod17a] Raúl Rodríguez Pearson. *PVSystems: A Modelica Library for Photovoltaic System and Power Converter Design*. Apr. 2017. (Visited on 05/24/2017).

[Rod17b] Raúl Rodríguez Pearson. *PVSystems Documentation*. `https://raulrpearson.github.io/PVSystems/`. 2017. (Visited on 05/24/2017).

[TDD07] O. Tremblay, L. A. Dessaint, and A. I. Dekkiche. "A Generic Battery Model for the Dynamic Simulation of Hybrid Electric Vehicles". In: *2007 IEEE Vehicle Power and Propulsion Conference*. Sept. 2007, pp. 284–289. DOI: 10.1109/VPPC.2007.4544139.

[Til17] Michael M. Tiller. *Modelica by Example*. 2017. (Visited on 01/26/2017).

[VGF09] M. G. Villalva, J. R. Gazoli, and E. R. Filho. "Comprehensive Approach to Modeling and Simulation of Photovoltaic Arrays". In: *IEEE Transactions on Power Electronics* 24.5 (May 2009), pp. 1198–1208. ISSN: 0885-8993. DOI: 10.1109/TPEL.2009.2013862.

Acronyms and abbreviations

AC Alternating Current.

BES Battery Energy Storage.

BOS Balance of System.

CCM Continuous Conduction Mode.

CPM Current Programmed Mode.

DAE Differential Algebraic Equation.

DC Direct Current.

DCM Discontinuous Conduction Mode.

EPT Energy Payback Time.

ESR Equivalent Series Resistance.

FMI Functional Mock-up Interface.

FMU Functional Mock-up Unit.

GUI Graphical User Interface.

IDE Integrated Development Environment.

LOC Lines Of Code.

M&S modelling and simulation.

MIT Massachusetts Institute of Technology.

MPP Maximum Power Point.

MPPT Maximum Power Point Tracker.

MSL Modelica Standard Library.

NER Net Energy Ratio.

ODE Ordinary Differential Equation.

P&O Perturb and Observe.

PI Proportional Integral.

PID Proportional Integral Derivative.

PLL Phased-Locked Loop.

PV photovoltaic.

PWM Pulse Width Modulation.

QSG Quadrature Signal Generator.

SPDT Single-pole double-throw.

SPST Single-pole single-throw.

STC Standard Testing Conditions.

V&V Verification and Validation.

Source code

Control/Assemblies/Inverter1phCompleteController.mo

```
within PVSystems.Control.Assemblies;
block Inverter1phCompleteController
  "Complete synchronous reference frame inverter controller"
  extends Modelica.Blocks.Icons.Block;
  // Parameters
  parameter Real ik=0.1 "Current PI gain";
  parameter Modelica.SIunits.Time iT=0.01 "Current PI time constant";
  parameter Real idMax=Modelica.Constants.inf "Maximum effort for id loop";
  parameter Real iqMax=Modelica.Constants.inf "Maximum effort for iq loop";
  parameter Real vk=0.1 "Voltage PI gain";
  parameter Modelica.SIunits.Time vT=0.01 "Voltage PI time constant";
  parameter Real vdcMax=Modelica.Constants.inf "Maximum effort for vdc loop";
  parameter Modelica.SIunits.Frequency fline=50 "Line frequency";
  // Interface
  Modelica.Blocks.Interfaces.RealInput iac "AC current sense" annotation (
     Placement(transformation(extent={{-140,-60},{-100,-20}}, rotation=0)));
  Modelica.Blocks.Interfaces.RealInput vac "AC voltage sense" annotation (
     Placement(transformation(extent={{-140,-100},{-100,-60}}, rotation=0)));
  Modelica.Blocks.Interfaces.RealInput idc "DC current sense" annotation (
     Placement(transformation(extent={{-140,20},{-100,60}}, rotation=0)));
  Modelica.Blocks.Interfaces.RealInput vdc "DC voltage sense" annotation (
     Placement(transformation(extent={{-140,60},{-100,100}}, rotation=0)));
  Modelica.Blocks.Interfaces.RealOutput d "Duty cycle" annotation (Placement(
      transformation(extent={{100,-10},{120,10}}, rotation=0)));
  // Components
  Modelica.Blocks.Sources.Constant iqs(k=0) annotation (Placement(
      transformation(extent={{-40,-30},{-20,-10}}, rotation=0)));
  PVSystems.Control.MPPTController mppt(
    sampleTime=1,
    vrefStep=0.5,
    pkThreshold=0.5,
    vrefStart=15) annotation (Placement(transformation(extent={{-80,36},{-60,56}},
        rotation=0)));
  Modelica.Blocks.Continuous.LimPID vdcPI(
    k=vk,
    controllerType=Modelica.Blocks.Types.SimpleController.PI,
    Ti=vT,
    yMax=vdcMax) annotation (Placement(transformation(extent={{-40,56},{-20,36}},
        rotation=0)));
  Inverter1phCurrentController currentController(
    k=ik,
    T=iT,
    fline=fline,
    idMax=idMax,
    iqMax=iqMax)
    annotation (Placement(transformation(extent={{60,-10},{80,10}})));
  PLL pLL(frequency=fline)
    annotation (Placement(transformation(extent={{-80,-90},{-60,-70}})));
  Modelica.Blocks.Sources.RealExpression vdcClone(y=vdc)
    annotation (Placement(transformation(extent={{-40,-100},{-20,-80}})));
  Modelica.Blocks.Math.Gain invertIds(k=-1)
    annotation (Placement(transformation(extent={{0,36},{20,56}})));
```

```
53 equation
54   connect(currentController.d, d)
55     annotation (Line(points={{81,0},{110,0}}, color={0,0,127}));
56   connect(idc, mppt.u2)
57     annotation (Line(points={{-120,40},{-120,40},{-82,40}},color={0,0,127}));
58   connect(vdc, mppt.u1) annotation (Line(points={{-120,80},{-90,80},{-90,52},{-82,
59         52}}, color={0,0,127}));
60   connect(iac, currentController.i) annotation (Line(points={{-120,-40},{-70,-40},
61         {-70,0},{58,0}}, color={0,0,127}));
62   connect(iqs.y, currentController.iqs) annotation (Line(points={{-19,-20},{40,
63         -20},{40,-6},{58,-6}}, color={0,0,127}));
64   connect(vac, pLL.v) annotation (Line(points={{-120,-80},{-120,-80},{-82,-80}},
65       color={0,0,127}));
66   connect(pLL.theta, currentController.theta)
67     annotation (Line(points={{-59,-80},{66,-80},{66,-12}}, color={0,0,127}));
68   connect(vdcClone.y, currentController.vdc) annotation (Line(points={{-19,-90},
69         {-19,-90},{74,-90},{74,-12}}, color={0,0,127}));
70   connect(mppt.y, vdcPI.u_s)
71     annotation (Line(points={{-59,46},{-42,46}}, color={0,0,127}));
72   connect(vdc, vdcPI.u_m)
73     annotation (Line(points={{-120,80},{-30,80},{-30,58}}, color={0,0,127}));
74   connect(vdcPI.y, invertIds.u)
75     annotation (Line(points={{-19,46},{-10,46},{-2,46}}, color={0,0,127}));
76   connect(invertIds.y, currentController.ids)
77     annotation (Line(points={{21,46},{40,46},{40,6},{58,6}}, color={0,0,127}));
78   annotation (Icon(coordinateSystem(preserveAspectRatio=true, extent={{-100,-100},
79           {100,100}}), graphics={
80       Rectangle(
81         extent={{-48,50},{12,-10}},
82         lineColor={0,0,127},
83         fillColor={255,255,255},
84         fillPattern=FillPattern.Solid),
85       Line(points={{-38,40},{-38,-4}}, color={192,192,192}),
86       Polygon(
87         points={{-38,40},{-42,32},{-34,32},{-38,40}},
88         lineColor={192,192,192},
89         fillColor={192,192,192},
90         fillPattern=FillPattern.Solid),
91       Line(points={{-42,0},{2,0}}, color={192,192,192}),
92       Line(points={{-38,0},{-38,14},{-30,24},{2,24}}, color={0,0,127}),
93       Line(
94         visible=strict,
95         points={{-30,24},{2,24}},
96         color={255,0,0}),
97       Polygon(
98         points={{0,4},{-4,-4},{4,-4},{0,4}},
99         lineColor={192,192,192},
100        fillColor={192,192,192},
101        fillPattern=FillPattern.Solid,
102        origin={-2,0},
103        rotation=270),
104      Line(points={{12,20},{52,20}}, color={0,0,127}),
105      Line(points={{-88,20},{-48,20}}, color={0,0,127}),
106      Line(points={{-68,20},{-68,-30},{32,-30},{32,20}}, color={0,0,127}),
107      Polygon(
108        points={{0,4},{-4,-4},{4,-4},{0,4}},
109        lineColor={0,0,127},
110        fillColor={0,0,127},
111        fillPattern=FillPattern.Solid,
112        origin={56,20},
113        rotation=270),
114      Polygon(
115        points={{0,4},{-4,-4},{4,-4},{0,4}},
116        lineColor={0,0,127},
117        fillColor={0,0,127},
118        fillPattern=FillPattern.Solid,
```

```
        origin={-52,20},
        rotation=270),
      Rectangle(
        extent={{-18,10},{42,-50}},
        lineColor={0,0,127},
        fillColor={255,255,255},
        fillPattern=FillPattern.Solid),
      Line(points={{-8,0},{-8,-44}}, color={192,192,192}),
      Polygon(
        points={{-8,0},{-12,-8},{-4,-8},{-8,0}},
        lineColor={192,192,192},
        fillColor={192,192,192},
        fillPattern=FillPattern.Solid),
      Line(points={{-12,-40},{32,-40}}, color={192,192,192}),
      Line(points={{-8,-40},{-8,-26},{0,-16},{32,-16}}, color={0,0,127}),
      Line(
        visible=strict,
        points={{0,-16},{32,-16}},
        color={255,0,0}),
      Polygon(
        points={{0,4},{-4,-4},{4,-4},{0,4}},
        lineColor={192,192,192},
        fillColor={192,192,192},
        fillPattern=FillPattern.Solid,
        origin={28,-40},
        rotation=270),
      Line(points={{42,-20},{82,-20}}, color={0,0,127}),
      Line(points={{-58,-20},{-18,-20}}, color={0,0,127}),
      Line(points={{-38,-20},{-38,-70},{62,-70},{62,-20}}, color={0,0,127}),
      Polygon(
        points={{0,4},{-4,-4},{4,-4},{0,4}},
        lineColor={0,0,127},
        fillColor={0,0,127},
        fillPattern=FillPattern.Solid,
        origin={86,-20},
        rotation=270),
      Polygon(
        points={{0,4},{-4,-4},{4,-4},{0,4}},
        lineColor={0,0,127},
        fillColor={0,0,127},
        fillPattern=FillPattern.Solid,
        origin={-22,-20},
        rotation=270),
      Text(
        extent={{-100,80},{100,70}},
        lineColor={0,0,255},
        textString="PV control")}), Documentation(info="<html>
          <p>
            An
            additional <a href=\"modelica://Modelica.Blocks.Continuous.LimPID\">LimPID</a
                >
            block is used to closed the DC voltage loop around the <i>d</i>
            component of the AC current,
            using <a href=\"modelica://
                PVSystems.Control.Assemblies.Inverter1phCurrentController\">
                Inverter1phCurrentController</a>.
          </p>

          <p>
            Currently, this block doesn't provide control of the <i>q</i>
            component, which is set to 0.</p>
        </html>"));
end Inverter1phCompleteController;
```

Control/Assemblies/Inverter1phCurrentController.mo

```
within PVSystems.Control.Assemblies;
block Inverter1phCurrentController
  "Simple synchronous reference frame PI current controller"
  extends Modelica.Blocks.Icons.Block;
  parameter Real k(final unit="1") = 0.1 "PI controllers gain";
  parameter Modelica.SIunits.Time T(final min=Modelica.Constants.small) = 0.01
    "PI controllers time constant (T>0 required)";
  parameter Modelica.SIunits.Frequency fline=50 "AC line frequency";
  parameter Real idMax=Modelica.Constants.inf "Maximum effort for id loop";
  parameter Real iqMax=Modelica.Constants.inf "Maximum effort for iq loop";
  Park park annotation (Placement(transformation(extent={{-70,-14},{-50,6}},
        rotation=0)));
  Modelica.Blocks.Nonlinear.FixedDelay T4Delay(delayTime=1/4/fline) annotation (
     Placement(transformation(extent={{-108,-30},{-88,-10}}, rotation=0)));
  Modelica.Blocks.Continuous.LimPID idPI(
    k=k,
    controllerType=Modelica.Blocks.Types.SimpleController.PI,
    Ti=T,
    yMax=idMax) annotation (Placement(transformation(extent={{-40,50},{-20,70}},
        rotation=0)));
  Modelica.Blocks.Continuous.LimPID iqPI(
    k=k,
    controllerType=Modelica.Blocks.Types.SimpleController.PI,
    Ti=T,
    yMax=iqMax) annotation (Placement(transformation(extent={{-40,-50},{-20,-70}},
        rotation=0)));
  InversePark inversePark
    annotation (Placement(transformation(extent={{8,-14},{28,6}}, rotation=0)));
  Modelica.Blocks.Sources.Constant dOffset(k=0.5) annotation (Placement(
      transformation(extent={{50,20},{70,40}}, rotation=0)));
  Modelica.Blocks.Interfaces.RealInput i "Sensed current" annotation (Placement(
      transformation(extent={{-160,-20},{-120,20}}, rotation=0),
      iconTransformation(extent={{-140,-20},{-100,20}})));
  Modelica.Blocks.Interfaces.RealInput ids "Current d component setpoint"
    annotation (Placement(transformation(extent={{-160,40},{-120,80}}, rotation=
        0), iconTransformation(extent={{-140,40},{-100,80}})));
  Modelica.Blocks.Interfaces.RealInput iqs "Current q component setpoint"
    annotation (Placement(transformation(extent={{-160,-80},{-120,-40}},
        rotation=0), iconTransformation(extent={{-140,-80},{-100,-40}})));
  Modelica.Blocks.Interfaces.RealInput theta "Sensed AC voltage phase"
    annotation (Placement(transformation(
      origin={-40,-120},
      extent={{-20,-20},{20,20}},
      rotation=90)));
  Modelica.Blocks.Interfaces.RealInput vdc "Sensed DC voltage" annotation (
     Placement(transformation(
      origin={40,-120},
      extent={{-20,-20},{20,20}},
      rotation=90)));
  Modelica.Blocks.Interfaces.RealOutput d "Duty cycle output" annotation (
     Placement(transformation(extent={{120,-10},{140,10}}, rotation=0),
      iconTransformation(extent={{100,-10},{120,10}})));
  Modelica.Blocks.Math.Division dScaling annotation (Placement(transformation(
        extent={{50,-16},{70,4}}, rotation=0)));
  Modelica.Blocks.Math.Add dCalc annotation (Placement(transformation(extent={{
         88,-10},{108,10}}, rotation=0)));
equation
  // Connections
  connect(park.beta, T4Delay.y) annotation (Line(points={{-72,-8},{-80,-8},{-80,
        -20},{-87,-20}}, color={0,0,127}));
  connect(iqPI.y, inversePark.q) annotation (Line(points={{-19,-60},{0,-60},{0,
        -8},{6,-8}}, color={0,0,127}));
  connect(idPI.y, inversePark.d)
```

```
  annotation (Line(points={{-19,60},{0,60},{0,0},{6,0}},color={0,0,127}));
connect(i, park.alpha)
  annotation (Line(points={{-140,0},{-140,0},{-72,0}}, color={0,0,127}));
connect(i, T4Delay.u) annotation (Line(points={{-140,0},{-116,0},{-116,-20},{
      -110,-20}}, color={0,0,127}));
connect(inversePark.theta, theta) annotation (Line(points={{18,-16},{18,-80},
      {-40,-80},{-40,-120}},color={0,0,127}));
connect(inversePark.alpha, dScaling.u1)
  annotation (Line(points={{29,0},{48,0}}, color={0,0,127}));
connect(vdc, dScaling.u2)
  annotation (Line(points={{40,-120},{40,-12},{48,-12}}, color={0,0,127}));
connect(dScaling.y, dCalc.u2)
  annotation (Line(points={{71,-6},{86,-6}}, color={0,0,127}));
connect(dCalc.y, d)
  annotation (Line(points={{109,0},{109,0},{130,0}}, color={0,0,127}));
connect(theta, park.theta) annotation (Line(points={{-40,-120},{-40,-80},{-60,
      -80},{-60,-16}}, color={0,0,127}));
connect(dOffset.y, dCalc.u1)
  annotation (Line(points={{71,30},{80,30},{80,6},{86,6}}, color={0,0,127}));
connect(idPI.u_s, ids)
  annotation (Line(points={{-42,60},{-140,60}}, color={0,0,127}));
connect(idPI.u_m, park.d)
  annotation (Line(points={{-30,48},{-30,0},{-49,0}}, color={0,0,127}));
connect(park.q, iqPI.u_m)
  annotation (Line(points={{-49,-8},{-30,-8},{-30,-48}}, color={0,0,127}));
connect(iqPI.u_s, iqs) annotation (Line(points={{-42,-60},{-86,-60},{-140,-60}},
                color={0,0,127}));
annotation (
  Icon(coordinateSystem(preserveAspectRatio=true, extent={{-100,-100},{100,
        100}}), graphics={
      Rectangle(
        extent={{-48,50},{12,-10}},
        lineColor={0,0,127},
        fillColor={255,255,255},
        fillPattern=FillPattern.Solid),
      Line(points={{-38,40},{-38,-4}}, color={192,192,192}),
      Polygon(
        points={{-38,40},{-42,32},{-34,32},{-38,40}},
        lineColor={192,192,192},
        fillColor={192,192,192},
        fillPattern=FillPattern.Solid),
      Line(points={{-42,0},{2,0}}, color={192,192,192}),
      Line(points={{-38,0},{-38,14},{-30,24},{2,24}}, color={0,0,127}),
      Line(
        visible=strict,
        points={{-30,24},{2,24}},
        color={255,0,0}),
      Polygon(
        points={{0,4},{-4,-4},{4,-4},{0,4}},
        lineColor={192,192,192},
        fillColor={192,192,192},
        fillPattern=FillPattern.Solid,
        origin={-2,0},
        rotation=270),
      Line(points={{12,20},{52,20}}, color={0,0,127}),
      Line(points={{-88,20},{-48,20}}, color={0,0,127}),
      Line(points={{-68,20},{-68,-30},{32,-30},{32,20}}, color={0,0,127}),
      Polygon(
        points={{0,4},{-4,-4},{4,-4},{0,4}},
        lineColor={0,0,127},
        fillColor={0,0,127},
        fillPattern=FillPattern.Solid,
        origin={56,20},
        rotation=270),
      Polygon(
        points={{0,4},{-4,-4},{4,-4},{0,4}},
```

```
130         lineColor={0,0,127},
131         fillColor={0,0,127},
132         fillPattern=FillPattern.Solid,
133         origin={-52,20},
134         rotation=270),
135       Rectangle(
136         extent={{-18,10},{42,-50}},
137         lineColor={0,0,127},
138         fillColor={255,255,255},
139         fillPattern=FillPattern.Solid),
140       Line(points={{-8,0},{-8,-44}}, color={192,192,192}),
141       Polygon(
142         points={{-8,0},{-12,-8},{-4,-8},{-8,0}},
143         lineColor={192,192,192},
144         fillColor={192,192,192},
145         fillPattern=FillPattern.Solid),
146       Line(points={{-12,-40},{32,-40}}, color={192,192,192}),
147       Line(points={{-8,-40},{-8,-26},{0,-16},{32,-16}}, color={0,0,127}),
148       Line(
149         visible=strict,
150         points={{0,-16},{32,-16}},
151         color={255,0,0}),
152       Polygon(
153         points={{0,4},{-4,-4},{4,-4},{0,4}},
154         lineColor={192,192,192},
155         fillColor={192,192,192},
156         fillPattern=FillPattern.Solid,
157         origin={28,-40},
158         rotation=270),
159       Line(points={{42,-20},{82,-20}}, color={0,0,127}),
160       Line(points={{-58,-20},{-18,-20}}, color={0,0,127}),
161       Line(points={{-38,-20},{-38,-70},{62,-70},{62,-20}}, color={0,0,127}),
162       Polygon(
163         points={{0,4},{-4,-4},{4,-4},{0,4}},
164         lineColor={0,0,127},
165         fillColor={0,0,127},
166         fillPattern=FillPattern.Solid,
167         origin={86,-20},
168         rotation=270),
169       Polygon(
170         points={{0,4},{-4,-4},{4,-4},{0,4}},
171         lineColor={0,0,127},
172         fillColor={0,0,127},
173         fillPattern=FillPattern.Solid,
174         origin={-22,-20},
175         rotation=270),
176       Text(
177         extent={{-100,80},{100,70}},
178         lineColor={0,0,255},
179         textString="Idq control")}),
180     Documentation(info="<html>
181       <p>
182         Synchronous reference frame current controller for a 1-phase
183         inverter. It takes the measured and the dq setpoints and
184         calculates the duty cycle, which can be then used as the input to
185         the <a href=\"modelica://PVSystems.Control.SignalPWM\">SignalPWM</a>
186         block in switching models or directly as the input of the switch
187         or converter in averaged models.
188       </p>
189
190       <p>
191         The control is performed with
192         two <a href=\"modelica://Modelica.Blocks.Continuous.LimPID\">LimPID</a>
193         blocks (one per component) configured as a PI controller.</p>
194      </html>"),
195     Diagram(coordinateSystem(extent={{-120,-100},{120,100}}, initialScale=0.1)));
```

```
196 end Inverter1phCurrentController;
```

Control/Assemblies/package.mo

```
1  within PVSystems.Control;
2  package Assemblies "Block assemblies useful in PV and power electronics"
3  extends Icons.AssembliesPackage;
4
5
6    annotation(Documentation(info="
7      <html>
8        <p>
9          Block assemblies useful in PV and power electronics</p>
10     </html>"));
11 end Assemblies;
```

Control/Assemblies/package.order

```
1 Inverter1phCurrentController
2 Inverter1phCompleteController
```

Control/CPM.mo

```
1  within PVSystems.Control;
2  model CPM "Current Peak Mode modulator for averaged models"
3    extends Interfaces.CPMInterface;
4  protected
5    Real d2;
6  equation
7    d2 = min(L*fs*(vc - Va*d)/Rf/vm2, 1 - d);
8    d = 2*(vc*(d + d2) - vs)/(Rf/L/fs*(vm1 + vm2)*d2*(d + d2) + 2*Va*(d + d2));
9    annotation (Icon(graphics={
10       Line(points={{-80,20},{-70,0},{-50,0},{-30,60},{10,0},{30,0},{50,60},{
11           80,20}}, color={255,0,0}),
12       Line(points={{-52,-140}}, color={0,0,255}),
13       Line(points={{-80.1563,45.078},{-50,30},{-50,70},{30,30},{30,70},{
14           79.531,45.234}}, color={0,0,255}),
15       Line(
16         points={{-50,80},{-50,-30}},
17         color={0,0,255},
18         pattern=LinePattern.Dash),
19       Line(
20         points={{-30,80},{-30,-30}},
21         color={0,0,255},
22         pattern=LinePattern.Dash),
23       Line(
24         points={{30,80},{30,-30}},
25         color={0,0,255},
26         pattern=LinePattern.Dash),
27       Line(
28         points={{50,80},{50,-30}},
29         color={0,0,255},
30         pattern=LinePattern.Dash),
```

```
      Line(
        points={{-80,-80},{-50,-80},{-50,-40},{-30,-40},{-30,-80},{30,-80},{
           30,-40},{50,-40},{50,-80},{80,-80}},
        color={255,0,255},
        pattern=LinePattern.Dot),
      Line(points={{-80,-70},{80,-70}}, color={0,0,0})}),
   Documentation(info="<html>
      <p>
        Current-Programmed-Mode controller model. Computes duty ratio
        based on averaged inductor current, voltages applied to the
        inductor, and amplitude of the artificial ramp. The CPM controller
        model is valid <b>for CCM and DCM operation</b> of the power
        converter. All parameters and inputs are referred to the primary
        side.
      </p>

      <p>
        <i>Limitation</i>: does not include sampling effects or preditions
        of period-doubling instability.
      </p>

      <p>
        Model taken
        from <a href=\"modelica://PVSystems.UsersGuide.References.EM01\">EM01</a>
        and <a href=\"modelica://PVSystems.UsersGuide.References.EMA16\">EMA16</a>.</p>
    </html>"));
end CPM;
```

Control/CPM_CCM.mo

```
within PVSystems.Control;
model CPM_CCM "Current Peak Mode modulator for averaged CCM models"
  extends Interfaces.CPMInterface;
  parameter Real d_disabled(final unit="1") "Value of duty cycle when disabled";
  Modelica.Blocks.Interfaces.BooleanInput enable
    "Block enable/disable" annotation (Placement(transformation(
      extent={{-20,-20},{20,20}},
      rotation=90,
      origin={0,-120}), iconTransformation(
      extent={{-20,-20},{20,20}},
      rotation=90,
      origin={0,-120})));
equation
  if enable then
    d = 2*(vc - vs)/(Rf/L/fs*(vm1 + vm2)*(1 - d) + 2*Va);
  else
    d = d_disabled;
  end if;
  annotation (Icon(graphics={
      Line(points={{-80,20},{-50,-20},{-30,60},{30,-20},{50,60},{80,20}},
         color={255,0,0}),
      Line(points={{-52,-140}}, color={0,0,255}),
      Line(points={{-80.1563,45.078},{-50,30},{-50,70},{30,30},{30,70},{
          79.531,45.234}}, color={0,0,255}),
      Line(
        points={{-50,80},{-50,-30}},
        color={0,0,255},
        pattern=LinePattern.Dash),
      Line(
        points={{-30,80},{-30,-30}},
        color={0,0,255},
        pattern=LinePattern.Dash),
```

```
      Line(
        points={{30,80},{30,-30}},
        color={0,0,255},
        pattern=LinePattern.Dash),
      Line(
        points={{50,80},{50,-30}},
        color={0,0,255},
        pattern=LinePattern.Dash),
      Line(
        points={{-80,-80},{-50,-80},{-50,-40},{-30,-40},{-30,-80},{30,-80},{
           30,-40},{50,-40},{50,-80},{80,-80}},
        color={255,0,255},
        pattern=LinePattern.Dot),
      Line(points={{-80,-70},{80,-70}}, color={0,0,0})}),
   Documentation(info="<html>
      <p>
       Current-Programmed-Mode controller model. Computes duty ratio
       based on averaged inductor current, voltages applied to the
       inductor, and amplitude of the artificial ramp. This CPM
       controller model is valid <b>only for CCM operation</b> of the
       powerconverter. All parameters and inputs are referred to the
       primary side.
      </p>

      <p>
       <i>Limitation</i>: does not include sampling effects or preditions
       of period-doubling instability.
      </p>

      <p>
       Model taken
       from <a href=\"modelica://PVSystems.UsersGuide.References.EM01\">EM01</a>
       and <a href=\"modelica://PVSystems.UsersGuide.References.EMA16\">EMA16</a>.</p>
    </html>"));
end CPM_CCM;
```

Control/DeadTime.mo

```
within PVSystems.Control;
block DeadTime "Introduces a dead time in complementary PWM firing signals"
  extends Modelica.Blocks.Icons.Block;
  parameter Modelica.SIunits.Time deadTime=0 "Dead time";
  Modelica.Blocks.Interfaces.BooleanInput c "PWM input"
    annotation (Placement(transformation(extent={{-140,-20},{-100,20}})));
  Modelica.Blocks.Interfaces.BooleanOutput c1 "PWM output" annotation (
     Placement(transformation(
       extent={{-10,-10},{10,10}},
       rotation=0,
       origin={110,40})));
  Modelica.Blocks.Interfaces.BooleanOutput c2 "PWM complement" annotation (
     Placement(transformation(
       extent={{-10,-10},{10,10}},
       rotation=0,
       origin={110,-40})));
  Modelica.Blocks.MathBoolean.OnDelay c1_onDelay(delayTime=deadTime)
    annotation (Placement(transformation(extent={{56,36},{64,44}})));
  Modelica.Blocks.MathBoolean.OnDelay c2_onDelay(delayTime=deadTime)
    annotation (Placement(transformation(extent={{56,-44},{64,-36}})));
  Modelica.Blocks.Logical.Not notBlock
    annotation (Placement(transformation(extent={{-10,-50},{10,-30}})));
equation
  connect(c1_onDelay.y, c1)
```

```
  annotation (Line(points={{64.8,40},{110,40}}, color={255,0,255}));
 connect(c2_onDelay.y, c2)
  annotation (Line(points={{64.8,-40},{110,-40}}, color={255,0,255}));
 connect(notBlock.y, c2_onDelay.u)
  annotation (Line(points={{11,-40},{54.4,-40}}, color={255,0,255}));
 connect(c, c1_onDelay.u) annotation (Line(points={{-120,0},{-60,0},{-60,40},{
       54.4,40}}, color={255,0,255}));
 connect(c, notBlock.u) annotation (Line(points={{-120,0},{-60,0},{-60,-40},{-12,
       -40}}, color={255,0,255}));
 annotation (Icon(coordinateSystem(preserveAspectRatio=false, extent={{-100,-100},
        {100,100}}),graphics={
     Rectangle(
       extent={{-20,80},{20,-80}},
       pattern=LinePattern.None,
       fillColor={255,255,255},
       fillPattern=FillPattern.Backward,
       lineColor={0,0,255}),
     Line(points={{-70,70},{-20,70},{-20,10},{70,10}}, color={255,0,255}),
     Line(points={{-70,-70},{20,-70},{20,-10},{70,-10}}, color={255,0,255}),
     Line(
       points={{20,80},{20,-80}},
       color={0,0,255},
       pattern=LinePattern.Dash),
     Line(
       points={{-20,80},{-20,-80}},
       color={0,0,255},
       pattern=LinePattern.Dash)}),
   Documentation(info="<html>
      <p>
       Given an input boolean firing signal, output that signal and it's
       complement with <i>deadTime</i> seconds of dead time.</p>
    </html>"));
end DeadTime;
```

Control/Interfaces/CPMInterface.mo

```
within PVSystems.Control.Interfaces;
partial model CPMInterface "Common interface for averaged CPM block"
 extends Modelica.Blocks.Icons.Block;
 parameter Modelica.SIunits.Inductance L
   "Equivalent inductance, referred to primary";
 parameter Modelica.SIunits.Frequency fs
   "Switching frequency";
 parameter Modelica.SIunits.Voltage Va
   "Amplitude of the artificial ramp, Va=Rf*ma/fs";
 parameter Modelica.SIunits.Resistance Rf
   "Equivalent current-sense resistance";
 Modelica.Blocks.Interfaces.RealInput vc
   "Control input, vc=Rf*ic"
   annotation (
     Placement(transformation(extent={{-140,80},{-100,120}}, rotation=0),
       iconTransformation(extent={{-140,80},{-100,120}})));
 Modelica.Blocks.Interfaces.RealInput vs
   "Sensed average inductor current vs=Rf*iL"
   annotation (
     Placement(transformation(extent={{-140,20},{-100,60}}, rotation=0),
       iconTransformation(extent={{-140,20},{-100,60}})));
 Modelica.Blocks.Interfaces.RealInput vm1
   "Voltage across L in interval 1, slope m1=vm1/L"
   annotation (Placement(transformation(extent={{-140,-60},{-100,-20}},
       rotation=0), iconTransformation(extent={{-140,-60},{-100,-20}})));
 Modelica.Blocks.Interfaces.RealInput vm2
```

```
    "(-) Voltage across L in interval 2, slope m2=vm2/L"
    annotation (Placement(transformation(extent={{-140,-120},{-100,-80}},
        rotation=0), iconTransformation(extent={{-140,-120},{-100,-80}})));
  Modelica.Blocks.Interfaces.RealOutput d
    "Duty cycle"
    annotation (Placement(
       transformation(extent={{100,-10},{120,10}}, rotation=0)));
  annotation(Documentation(info="
    <html>
      <p>
        Common interface for averaged CPM block</p>
    </html>"));
end CPMInterface;
```

Control/Interfaces/package.mo

```
within PVSystems.Control;
package Interfaces "Common interfaces for control blocks"
extends Modelica.Icons.InterfacesPackage;

  annotation(Documentation(info="
    <html>
      <p>
        Common interfaces for control blocks</p>
    </html>"));
end Interfaces;
```

Control/Interfaces/package.order

```
CPMInterface
```

Control/InversePark.mo

```
within PVSystems.Control;
block InversePark "Inverse Park transformation"
  extends Modelica.Blocks.Icons.Block;
  Modelica.Blocks.Interfaces.RealInput d annotation (Placement(transformation(
        extent={{-140,20},{-100,60}}, rotation=0)));
  Modelica.Blocks.Interfaces.RealInput q annotation (Placement(transformation(
        extent={{-140,-60},{-100,-20}}, rotation=0)));
  Modelica.Blocks.Interfaces.RealOutput alpha annotation (Placement(
       transformation(extent={{100,30},{120,50}}, rotation=0)));
  Modelica.Blocks.Interfaces.RealOutput beta annotation (Placement(
       transformation(extent={{100,-50},{120,-30}}, rotation=0)));
  Modelica.Blocks.Interfaces.RealInput theta annotation (Placement(
       transformation(
       origin={0,-120},
       extent={{-20,-20},{20,20}},
       rotation=90)));
equation
  d = alpha*cos(theta) + beta*sin(theta);
  q = -alpha*sin(theta) + beta*cos(theta);
  annotation (
```

```
21    Diagram(graphics),
22    Icon(graphics={
23       Line(
24         points={{0,0},{15,60},{30,0},{45,-60},{60,0}},
25         color={0,0,255},
26         smooth=Smooth.Bezier),
27       Line(
28         points={{-96,-234}},
29         color={255,0,0},
30         smooth=Smooth.Bezier),
31       Line(points={{-80,20},{-20,20}}, color={128,0,255}),
32       Line(points={{-80,-20},{-20,-20}}, color={0,255,0}),
33       Line(
34         points={{15,0},{30,60},{45,0},{60,-60},{75,0}},
35         color={255,0,0},
36         smooth=Smooth.Bezier)}),
37    Documentation(info="<html>
38       <p>
39         Perform inverse Park transformation. This transformation translates
40         from the synchronous reference frame (d-q) to the static reference
41         frame (alfa-beta).
42       </p>
43     </html>"));
44 end InversePark;
```

Control/MPPTController.mo

```
1  within PVSystems.Control;
2  block MPPTController "Maximum Power Point Tracking Controller"
3    extends Modelica.Blocks.Interfaces.SI2SO;
4    parameter Modelica.SIunits.Time sampleTime=1 "Sample time of control block";
5    parameter Modelica.SIunits.Voltage vrefStep=5 "Step of change for vref";
6    parameter Modelica.SIunits.Power pkThreshold=1
7      "Power threshold below which no change is considered";
8    parameter Modelica.SIunits.Voltage vrefStart=10
9      "Voltage reference initial value";
10 protected
11   discrete Modelica.SIunits.Voltage vk;
12   discrete Modelica.SIunits.Current ik;
13   discrete Modelica.SIunits.Power pk;
14   discrete Modelica.SIunits.Voltage vref(start=vrefStart);
15 equation
16   when sample(sampleTime, sampleTime) then
17     vk = pre(u1);
18     ik = pre(u2);
19     pk = vk*ik;
20     if abs(pk - pre(pk)) < pkThreshold then
21       // power unchanged => don't change vref
22       vref = pre(vref);
23     elseif pk - pre(pk) > 0 then
24       // power increased => repeat last action
25       vref = pre(vref) + vrefStep*sign(vk - pre(vk));
26     else
27       // power decreased => change last action
28       vref = pre(vref) - vrefStep*sign(vk - pre(vk));
29     end if;
30   end when;
31   y = vref;
32   annotation (
33     Diagram(graphics),
34     Documentation(info="<html>
35       <p>
```

```
        Maximum power-point tracking controller. Given the DC voltage and
        current, this controller will output a moving reference for a DC
        voltage control loop in order to maximize the power extracted from a
        PV array for a given (unknown) solar irradiation and junction
        temperature.
      </p>

      <p>
        The operation of the block can be customized by setting the
        following parameters:
      </p>

      <ul class=\"org-ul\">
        <li><i>sampleTime</i>: sample time of the control block. The control
          output will be updated with this period.
        </li>
        <li><i>vrefStep</i>: amount of change to vref when updated.
        </li>
        <li><i>pkThreshold</i>: amount of power below which it is considered
          that no change in power has occurred.
        </li>
      </ul>
    </html>"),
   Icon(graphics={
      Line(points={{-80,80},{-80,-80},{80,-80}}, color={0,0,0}),
      Line(
        points={{-80,-80},{40,80},{60,-80}},
        color={0,0,255},
        smooth=Smooth.Bezier),
      Line(
        points={{-80,40},{30,40},{30,-80}},
        color={255,0,0},
        pattern=LinePattern.Dash),
      Text(
        extent={{-60,80},{60,40}},
        lineColor={0,0,255},
        pattern=LinePattern.Dash,
        fillColor={95,95,95},
        fillPattern=FillPattern.Solid,
        textString="MPPT")}));
end MPPTController;
```

Control/package.mo

```
within PVSystems;
package Control "Control elements for power converters"
extends Modelica.Icons.Package;









annotation (Icon(coordinateSystem(preserveAspectRatio=true, extent={{-100.0,-100.0},
        {100.0,100.0}}), graphics={
```

```
    Rectangle(
      origin={0.0,35.1488},
      fillColor={255,255,255},
      extent={{-30.0,-20.1488},{30.0,20.1488}}),
    Rectangle(
      origin={0.0,-34.8512},
      fillColor={255,255,255},
      extent={{-30.0,-20.1488},{30.0,20.1488}}),
    Line(origin={-51.25,0.0}, points={{21.25,-35.0},{-13.75,-35.0},{-13.75,
        35.0},{6.25,35.0}}),
    Polygon(
      origin={-40.0,35.0},
      pattern=LinePattern.None,
      fillPattern=FillPattern.Solid,
      points={{10.0,0.0},{-5.0,5.0},{-5.0,-5.0}}),
    Line(origin={51.25,0.0}, points={{-21.25,35.0},{13.75,35.0},{13.75,-35.0},
        {-6.25,-35.0}}),
    Polygon(
      origin={40.0,-35.0},
      pattern=LinePattern.None,
      fillPattern=FillPattern.Solid,
      points={{-10.0,0.0},{5.0,5.0},{5.0,-5.0}})}));
end Control;
```

Control/package.order

```
SwitchingPWM
SwitchingCPM
DeadTime
CPM_CCM
CPM
Park
InversePark
PLL
MPPTController
Assemblies
Interfaces
```

Control/Park.mo

```
within PVSystems.Control;
block Park "Park transformation"
  extends Modelica.Blocks.Icons.Block;
  Modelica.Blocks.Interfaces.RealInput alpha annotation (Placement(
      transformation(extent={{-140,20},{-100,60}}, rotation=0)));
  Modelica.Blocks.Interfaces.RealInput beta annotation (Placement(
      transformation(extent={{-140,-60},{-100,-20}}, rotation=0)));
  Modelica.Blocks.Interfaces.RealOutput d annotation (Placement(transformation(
       extent={{100,30},{120,50}}, rotation=0)));
  Modelica.Blocks.Interfaces.RealOutput q annotation (Placement(transformation(
       extent={{100,-50},{120,-30}}, rotation=0)));
  Modelica.Blocks.Interfaces.RealInput theta annotation (Placement(
      transformation(
      origin={0,-120},
      extent={{-20,-20},{20,20}},
      rotation=90)));
equation
```

```
  d = alpha*cos(theta) + beta*sin(theta);
  q = -alpha*sin(theta) + beta*cos(theta);
  annotation (
    Diagram(graphics),
    Icon(graphics={
        Line(
          points={{-75,0},{-60,60},{-45,0},{-30,-60},{-15,0}},
          color={0,0,255},
          smooth=Smooth.Bezier),
        Line(
          points={{-96,-234}},
          color={255,0,0},
          smooth=Smooth.Bezier),
        Line(points={{20,20},{80,20}}, color={128,0,255}),
        Line(points={{20,-20},{80,-20}}, color={0,255,0}),
        Line(
          points={{-60,0},{-45,60},{-30,0},{-15,-60},{0,0}},
          color={255,0,0},
          smooth=Smooth.Bezier)}),
    Documentation(info="<html>
        <p>
          Perform Park transformation. This transformation translates from the
          static reference frame (alfa-beta) to the synchronous reference
          frame (d-q).
        </p>
      </html>"));
end Park;
```

Control/PLL.mo

```
within PVSystems.Control;
block PLL "Phase-locked loop"
  extends Modelica.Blocks.Icons.Block;
  parameter Modelica.SIunits.Frequency frequency=50;
  Modelica.Blocks.Continuous.Integrator integrator annotation (Placement(
        transformation(extent={{60,-10},{80,10}}, rotation=0)));
  Modelica.Blocks.Continuous.FirstOrder firstOrder(T=1e-3, k=100) annotation (
      Placement(transformation(extent={{-4,-10},{16,10}},rotation=0)));
  Modelica.Blocks.Nonlinear.FixedDelay QuarterTDelay(delayTime=1/frequency/4)
    annotation (Placement(transformation(extent={{-80,-30},{-60,-10}}, rotation=
          0)));
  Modelica.Blocks.Interfaces.RealInput v annotation (Placement(transformation(
        extent={{-140,-20},{-100,20}}, rotation=0)));
  Modelica.Blocks.Interfaces.RealOutput theta annotation (Placement(
        transformation(extent={{100,-10},{120,10}}, rotation=0)));
  Park park annotation (Placement(transformation(extent={{-40,-14},{-20,6}},
        rotation=0)));
  Modelica.Blocks.Math.Add add annotation (Placement(transformation(extent={{30,
          48},{50,68}}, rotation=0)));
  Modelica.Blocks.Sources.Constant lineFreq(k=2*Modelica.Constants.pi*frequency)
    annotation (Placement(transformation(extent={{-30,54},{-10,74}}, rotation=0)));
equation
  connect(v, park.alpha)
    annotation (Line(points={{-120,0},{-42,0}}, color={0,0,127}));
  connect(QuarterTDelay.y, park.beta) annotation (Line(points={{-59,-20},{-50,-20},
        {-50,-8},{-42,-8}}, color={0,0,127}));
  connect(QuarterTDelay.u, v) annotation (Line(points={{-82,-20},{-96,-20},{-96,
        0},{-120,0}}, color={0,0,127}));
  connect(integrator.y, theta)
    annotation (Line(points={{81,0},{110,0}}, color={0,0,127}));
  connect(park.theta, integrator.y) annotation (Line(points={{-30,-16},{-30,-30},
        {90,-30},{90,0},{81,0}}, color={0,0,127}));
```

```
  connect(add.y, integrator.u)
    annotation (Line(points={{51,58},{54,58},{54,0},{58,0}}, color={0,0,127}));
  connect(firstOrder.y, add.u2)
    annotation (Line(points={{17,0},{22,0},{22,52},{28,52}}, color={0,0,127}));
  connect(lineFreq.y, add.u1)
    annotation (Line(points={{-9,64},{28,64}}, color={0,0,127}));
  connect(park.q, firstOrder.u) annotation (Line(points={{-19,-8},{-12,-8},{-12,
        0},{-6,0}}, color={0,0,127}));
  annotation (
    Diagram(graphics),
    Icon(graphics={Line(
        points={{-70,0},{-50,60},{-30,0},{-10,-60},{10,0},{30,60},{50,0},{70,
           -60},{90,0}},
        color={0,0,255},
        smooth=Smooth.Bezier), Line(
        points={{-90,0},{-64,60},{-44,0},{-18,-60},{2,0},{22,60},{44,0},{64,-60},
           {88,0}},
        color={255,0,0},
        smooth=Smooth.Bezier)}),
    Documentation(info="<html>
       <p>
        Phase-locked loop. Given a sinusoidal input, extract the phase.
       </p>
     </html>"));
end PLL;
```

Control/SwitchingCPM.mo

```
within PVSystems.Control;
block SwitchingCPM "Current Peak Mode modulator for switching models"
  extends Modelica.Blocks.Icons.Block;
  parameter Real dMin(
    final min=Modelica.Constants.small,
    final max=1) = 0 "Minimum duty cycle";
  parameter Real dMax(
    final min=Modelica.Constants.small,
    final max=1) = 1 "Maximum duty cycle";
  // TODO: assert dMax > dMin
  parameter Modelica.SIunits.Frequency fs(final min=Modelica.Constants.small)
    "Switching frequency";
  parameter Modelica.SIunits.Time startTime(final min=0) = 0
    "Time instant of first pulse";
  parameter Modelica.SIunits.Voltage Va(final min=0)
    "Amplitude of artificial ramp";
  parameter Modelica.SIunits.Voltage vcMax "Maximum control voltage";
  Modelica.Blocks.Interfaces.RealInput vc "Control voltage"
    annotation (Placement(transformation(
        extent={{-140,20},{-100,60}}, rotation=0)));
  Modelica.Blocks.Interfaces.RealInput vs "Sensed voltage"
    annotation (Placement(transformation(
        extent={{-140,-60},{-100,-20}}, rotation=0)));
  Modelica.Blocks.Interfaces.BooleanOutput c "Boolean firing signal"
    annotation (Placement(
       transformation(extent={{100,30},{120,50}}, rotation=0)));
  Modelica.Blocks.Interfaces.RealOutput ramp "Artificial ramp signal"
    annotation (Placement(
       transformation(extent={{100,-50},{120,-30}}, rotation=0)));
  Modelica.Blocks.Logical.GreaterEqual greaterEqual
    annotation (Placement(transformation(extent={{-10,38},{10,58}})));
  Modelica.Blocks.Logical.RSFlipFlop rSFlipFlop
    annotation (Placement(transformation(extent={{70,36},{90,56}})));
  Modelica.Blocks.Math.Add addVa
```

```
    annotation (Placement(transformation(extent={{-50,-56},{-30,-36}})));
  Modelica.Blocks.Sources.SawTooth artificialRamp(
    amplitude=Va,
    period=1/fs,
    nperiod=-1,
    offset=0,
    startTime=startTime)
    annotation (Placement(transformation(extent={{-90,-80},{-70,-60}})));
  Modelica.Blocks.Nonlinear.Limiter vcLimiter(uMax=vcMax, uMin=0)
    annotation (Placement(transformation(extent={{-50,30},{-30,50}})));
  Modelica.Blocks.Sources.BooleanPulse dMinLimiter(
    period=1/fs,
    startTime=startTime,
    width=100*dMin)
    annotation (Placement(transformation(extent={{32,0},{52,20}})));
  Modelica.Blocks.Sources.BooleanPulse dMaxLimiter(
    period=1/fs,
    startTime=startTime + dMax/fs,
    width=100*(1 - dMax))
    annotation (Placement(transformation(extent={{-10,66},{10,86}})));
  Modelica.Blocks.Logical.Or orBlock
    annotation (Placement(transformation(extent={{32,66},{52,86}})));
equation
  connect(addVa.u1, vs)
    annotation (Line(points={{-52,-40},{-120,-40}}, color={0,0,127}));
  connect(artificialRamp.y, addVa.u2) annotation (Line(points={{-69,-70},{-60,-70},
        {-60,-52},{-52,-52}}, color={0,0,127}));
  connect(ramp, artificialRamp.y) annotation (Line(points={{110,-40},{80,-40},{
        80,-70},{-69,-70}}, color={0,0,127}));
  connect(rSFlipFlop.QI, c)
    annotation (Line(points={{91,40},{110,40}}, color={255,0,255}));
  connect(addVa.y, greaterEqual.u1) annotation (Line(points={{-29,-46},{-20,-46},
        {-20,48},{-12,48}}, color={0,0,127}));
  connect(vc, vcLimiter.u)
    annotation (Line(points={{-120,40},{-52,40}}, color={0,0,127}));
  connect(vcLimiter.y, greaterEqual.u2)
    annotation (Line(points={{-29,40},{-12,40}}, color={0,0,127}));
  connect(dMinLimiter.y, rSFlipFlop.R) annotation (Line(points={{53,10},{60,10},
        {60,40},{68,40}}, color={255,0,255}));
  connect(greaterEqual.y, orBlock.u2) annotation (Line(points={{11,48},{20,48},
        {20,68},{30,68}},color={255,0,255}));
  connect(orBlock.y, rSFlipFlop.S) annotation (Line(points={{53,76},{60,76},{60,
        52},{68,52}}, color={255,0,255}));
  connect(dMaxLimiter.y, orBlock.u1)
    annotation (Line(points={{11,76},{30,76}}, color={255,0,255}));
  annotation (Icon(graphics={
      Line(points={{-80,20},{-50,-20},{-30,60},{30,-20},{50,60},{80,20}},
         color={255,0,0}),
      Line(points={{-52,-140}}, color={0,0,255}),
      Line(points={{-80.1563,45.078},{-50,30},{-50,70},{30,30},{30,70},{
          79.531,45.234}}, color={0,0,255}),
      Line(
        points={{-50,80},{-50,-80}},
        color={0,0,255},
        pattern=LinePattern.Dash),
      Line(
        points={{-30,80},{-30,-80}},
        color={0,0,255},
        pattern=LinePattern.Dash),
      Line(
        points={{30,80},{30,-80}},
        color={0,0,255},
        pattern=LinePattern.Dash),
      Line(
        points={{50,80},{50,-80}},
        color={0,0,255},
```

```
        pattern=LinePattern.Dash),
      Line(points={{-80,-80},{-50,-80},{-50,-40},{-30,-40},{-30,-80},{30,-80},
          {30,-40},{50,-40},{50,-80},{80,-80}}, color={255,0,255})}),
   Documentation(info="<html>
      <p>
       Current-programmed mode (CPM), i.e. Peak Current Mode modulator
       switching model. Generates PWM signal based on sensed current
       signal <i>vs</i> and control current signal <i>vc</i>. Also
       outputs the artificial ramp signal.
      </p>

      <p>
       Model taken
       from <a href=\"modelica://PVSystems.UsersGuide.References.EM01\">EM01</a>
       and <a href=\"modelica://PVSystems.UsersGuide.References.EMA16\">EMA16</a>.</p>
    </html>"));
end SwitchingCPM;
```

Control/SwitchingPWM.mo

```
within PVSystems.Control;
block SwitchingPWM
 "Generates a pulse width modulated (PWM) boolean fire signal"
 extends Modelica.Blocks.Icons.Block;
 parameter Real dMax=1 "Maximum duty cycle";
 parameter Real dMin=0 "Minimum duty cycle";
 parameter Modelica.SIunits.Frequency fs "Switching frequency";
 parameter Modelica.SIunits.Time startTime=0 "Start time";
 Modelica.Blocks.Interfaces.RealInput vc "Control voltage"
   annotation (Placement(transformation(extent={{-140,-20},{-100,20}})));
 Modelica.Blocks.Interfaces.BooleanOutput c1 "Firing PWM signal" annotation (
    Placement(transformation(
      extent={{-10,-10},{10,10}},
      rotation=0,
      origin={110,0})));
 Modelica.Blocks.Nonlinear.Limiter limiter(uMax=dMax, uMin=dMin)
   annotation (Placement(transformation(extent={{-60,-10},{-40,10}})));
 Modelica.Blocks.Logical.Less greaterEqual annotation (Placement(
     transformation(extent={{-10,10},{10,-10}}, origin={50,0})));
 Modelica.Blocks.Sources.SawTooth sawtooth(
   final period=1/fs,
   final amplitude=1,
   final nperiod=-1,
   final offset=0,
   final startTime=startTime) annotation (Placement(transformation(origin={0,-50},
       extent={{-10,-10},{10,10}})));
equation
 connect(vc, limiter.u)
   annotation (Line(points={{-120,0},{-94,0},{-62,0}}, color={0,0,127}));
 connect(sawtooth.y, greaterEqual.u1) annotation (Line(points={{11,-50},{20,-50},
       {20,0},{38,0}}, color={0,0,127}));
 connect(limiter.y, greaterEqual.u2)
   annotation (Line(points={{-39,0},{0,0},{0,8},{38,8}}, color={0,0,127}));
 connect(greaterEqual.y, c1)
   annotation (Line(points={{61,0},{110,0}}, color={255,0,255}));
 annotation (Icon(graphics={
     Line(points={{-80,40},{80,40}},
       color={255,0,0}),
     Line(points={{-52,-140}}, color={0,0,255}),
     Line(points={{-80.1563,54.922},{-50,70},{-50,30},{30,70},{30,30},{79.531,
         54.766}}, color={0,0,255}),
     Line(
```

```
        points={{-50,80},{-50,-80}},
        color={0,0,255},
        pattern=LinePattern.Dash),
      Line(
        points={{-30,80},{-30,-80}},
        color={0,0,255},
        pattern=LinePattern.Dash),
      Line(
        points={{30,80},{30,-80}},
        color={0,0,255},
        pattern=LinePattern.Dash),
      Line(
        points={{50,80},{50,-80}},
        color={0,0,255},
        pattern=LinePattern.Dash),
      Line(points={{-80,-80},{-50,-80},{-50,-40},{-30,-40},{-30,-80},{30,-80},
          {30,-40},{50,-40},{50,-80},{80,-80}}, color={255,0,255})}),
    Documentation(info="<html>
      <p>
        Generate boolean firing signal from duty cycle input. Adapted
        from <a href=\"modelica://
            Modelica.Electrical.PowerConverters.DCDC.Control.SignalPWM\">SignalPWM</a>.</
            p>
    </html>"));
end SwitchingPWM;
```

Electrical/Assemblies/BidirectionalBuckBoost.mo

```
within PVSystems.Electrical.Assemblies;
model BidirectionalBuckBoost "Bidirectional Buck Boost converter"
  extends Interfaces.TwoPort;
  extends PVSystems.Icons.ConverterIcon;
  parameter Modelica.SIunits.Capacitance Cin "Input capacitance"
    annotation (Dialog(group="Power stage"));
  parameter Modelica.SIunits.Resistance Rcin
    "Series resistance of input capacitor"
    annotation (Dialog(group="Power stage"));
  parameter Modelica.SIunits.Capacitance Cout "Output capacitance"
    annotation (Dialog(group="Power stage"));
  parameter Modelica.SIunits.Resistance Rcout
    "Series resistance of output capacitor"
    annotation (Dialog(group="Power stage"));
  parameter Modelica.SIunits.Inductance L "Inductance"
    annotation (Dialog(group="Power stage"));
  parameter Modelica.SIunits.Resistance RL "Series resistance of inductor"
    annotation (Dialog(group="Power stage"));
  parameter Modelica.SIunits.Voltage vCin_ini=0
    "Guess for initial voltage of Cin"
    annotation (Dialog(group="Initialization"));
  parameter Modelica.SIunits.Voltage vCout_ini=0
    "Guess for initial voltage of Cout"
    annotation (Dialog(group="Initialization"));
  parameter Modelica.SIunits.Current iL_ini=0 "Guess for initial current of L"
    annotation (Dialog(group="Initialization"));
  parameter Real dmax(final unit="1") = 1 "Maximum duty cycle"
    annotation (Dialog(group="Switches"));
  parameter Real dmin(final unit="1") = 1e-3 "Minimum duty cycle"
    annotation (Dialog(group="Switches"));
  Modelica.Electrical.Analog.Basic.Capacitor outCap(C=Cout, v(start=vCout_ini))
    annotation (Placement(transformation(
      extent={{-10,10},{10,-10}},
      rotation=270,
```

```
35        origin={80,20})));
36    Modelica.Electrical.Analog.Basic.Capacitor inCap(C=Cin, v(start=vCin_ini))
37      annotation (Placement(transformation(
38        extent={{-10,-10},{10,10}},
39        rotation=270,
40        origin={-80,20})));
41    Modelica.Electrical.Analog.Basic.Inductor inductor(L=L, i(start=iL_ini))
42      annotation (Placement(transformation(extent={{-24,50},{-4,70}})));
43    replaceable model SwitchModel = CCM1 constrainedby
44      Interfaces.SwitchNetworkInterface
45      annotation(choicesAllMatching=true);
46    SwitchModel buckSw(dmin=dmin, dmax=dmax)
47      annotation (Placement(transformation(extent={{-50,30},{-30,50}})));
48    SwitchModel boostSw(dmin=dmin, dmax=dmax)
49      annotation (Placement(transformation(extent={{30,30},{50,50}})));
50    Modelica.Blocks.Interfaces.RealInput dbuck "Buck control voltage" annotation (
51       Placement(transformation(
52        extent={{-20,-20},{20,20}},
53        rotation=90,
54        origin={-40,-120})));
55    Modelica.Blocks.Interfaces.RealInput dboost "Boost control voltage"
56      annotation (Placement(transformation(
57        extent={{-20,-20},{20,20}},
58        rotation=90,
59        origin={40,-120})));
60    Modelica.Electrical.Analog.Basic.Resistor resistor(R=RL)
61      annotation (Placement(transformation(extent={{4,50},{24,70}})));
62    Modelica.Electrical.Analog.Basic.Resistor inESR(R=Rcin) annotation (Placement(
63        transformation(
64        extent={{-10,-10},{10,10}},
65        rotation=270,
66        origin={-80,-20})));
67    Modelica.Electrical.Analog.Basic.Resistor outESR(R=Rcout) annotation (
68       Placement(transformation(
69        extent={{-10,10},{10,-10}},
70        rotation=270,
71        origin={80,-20})));
72  equation
73    connect(buckSw.n1, inductor.p) annotation (Line(points={{-50,35},{-60,35},{-60,
74          60},{-24,60}}, color={0,0,255}));
75    connect(buckSw.p2, inductor.p)
76      annotation (Line(points={{-30,45},{-30,60},{-24,60}}, color={0,0,255}));
77    connect(boostSw.n1, buckSw.n2) annotation (Line(points={{30,35},{30,-50},{-30,
78          -50},{-30,35}},color={0,0,255}));
79    connect(boostSw.n2, boostSw.p1) annotation (Line(points={{50,35},{60,35},{60,
80          60},{30,60},{30,45}}, color={0,0,255}));
81    connect(p1, buckSw.p1) annotation (Line(points={{-100,50},{-100,50},{-70,50},
82          {-70,45},{-50,45}},color={0,0,255}));
83    connect(outCap.p, boostSw.p2) annotation (Line(points={{80,30},{80,50},{70,50},
84          {70,45},{50,45}}, color={0,0,255}));
85    connect(inCap.p, buckSw.p1) annotation (Line(points={{-80,30},{-80,50},{-70,
86          50},{-70,45},{-50,45}}, color={0,0,255}));
87    connect(resistor.n, boostSw.p1)
88      annotation (Line(points={{24,60},{30,60},{30,45}}, color={0,0,255}));
89    connect(inductor.n, resistor.p)
90      annotation (Line(points={{-4,60},{4,60}}, color={0,0,255}));
91    connect(p2, boostSw.p2) annotation (Line(points={{100,50},{100,50},{70,50},{
92          70,45},{50,45}}, color={0,0,255}));
93    connect(inCap.n, inESR.p)
94      annotation (Line(points={{-80,10},{-80,-10}},color={0,0,255}));
95    connect(outCap.n, outESR.p)
96      annotation (Line(points={{80,10},{80,10},{80,-10}}, color={0,0,255}));
97    connect(outESR.n, n2)
98      annotation (Line(points={{80,-30},{80,-50},{100,-50}}, color={0,0,255}));
99    connect(inESR.n, n1) annotation (Line(points={{-80,-30},{-80,-50},{-100,-50}},
100       color={0,0,255}));
```

```
  connect(inESR.n, buckSw.n2) annotation (Line(points={{-80,-30},{-80,-50},{-30,
        -50},{-30,35}}, color={0,0,255}));
  connect(outESR.n, buckSw.n2) annotation (Line(points={{80,-30},{80,-50},{-30,
        -50},{-30,35}}, color={0,0,255}));
  connect(dbuck, buckSw.d)
    annotation (Line(points={{-40,-120},{-40,28}}, color={0,0,127}));
  connect(dboost, boostSw.d)
    annotation (Line(points={{40,-120},{40,-46},{40,28}}, color={0,0,127}));
  annotation (Icon(graphics={
      Text(
        extent={{-60,30},{60,-30}},
        lineColor={0,0,255},
        fillColor={255,255,255},
        fillPattern=FillPattern.Solid,
        textString="1-ph"),
      Line(points={{-70,50},{-10,50}}, color={0,0,255}),
      Line(points={{-70,70},{-10,70}}, color={0,0,255}),
      Line(points={{10,-70},{70,-70}}, color={0,0,255}),
      Line(points={{10,-50},{70,-50}}, color={0,0,255})}), Documentation(info=
       "<html><p>Bidirectional buck boost converter</p></html>"));
end BidirectionalBuckBoost;
```

Electrical/Assemblies/CPMBidirectionalBuckBoost.mo

```
within PVSystems.Electrical.Assemblies;
model CPMBidirectionalBuckBoost
  "Bidirectional Buck Boost for battery USB interface"
  extends Interfaces.TwoPort;
  extends PVSystems.Icons.ConverterIcon;
  parameter Modelica.SIunits.Capacitance Cin "Input capacitance"
    annotation (Dialog(group="Power stage"));
  parameter Modelica.SIunits.Voltage vCin_ini=0
    "Guess for initial voltage of Cin"
    annotation (Dialog(group="Initialization"));
  parameter Modelica.SIunits.Capacitance Cout "Output capacitance"
    annotation (Dialog(group="Power stage"));
  parameter Modelica.SIunits.Voltage vCout_ini=0
    "Guess for initial voltage of Cout"
    annotation (Dialog(group="Initialization"));
  parameter Modelica.SIunits.Inductance L "Inductance"
    annotation (Dialog(group="Power stage"));
  parameter Modelica.SIunits.Current iL_ini=0 "Guess for initial current of L"
    annotation (Dialog(group="Initialization"));
  parameter Modelica.SIunits.Resistance RL "Series resistance of inductor"
    annotation (Dialog(group="Power stage"));
  parameter Modelica.SIunits.Resistance Rf "Equivalent sensing resistance"
    annotation (Dialog(group="CPM modulator"));
  parameter Modelica.SIunits.Frequency fs "Switching frequency"
    annotation (Dialog(group="CPM modulator"));
  parameter Modelica.SIunits.Voltage Va_buck
    "Articial ramp amplitude for buck CPM"
    annotation (Dialog(group="CPM modulator"));
  parameter Modelica.SIunits.Voltage Va_boost
    "Articial ramp amplitude for boost CPM"
    annotation (Dialog(group="CPM modulator"));
  Control.CPM_CCM buck_cpm(
    L=L,
    Rf=1,
    fs=fs,
    Va=Va_buck,
    d_disabled=1) annotation (Placement(transformation(
       extent={{-10,-10},{10,10}},
```

```
39        rotation=0,
40        origin={40,40})));
41    Control.CPM_CCM boost_cpm(
42      L=L,
43      Rf=1,
44      fs=fs,
45      Va=Va_boost,
46      d_disabled=0) annotation (Placement(transformation(
47        extent={{-10,-10},{10,10}},
48        rotation=0,
49        origin={10,4})));
50    Modelica.Blocks.Sources.RealExpression vsense(y=Rf*conv.inductor.i)
51      annotation (Placement(transformation(extent={{-80,60},{-44,80}})));
52    Modelica.Blocks.Sources.RealExpression vm1_buck(y=v1 - v2)
53      annotation (Placement(transformation(extent={{-80,-30},{-60,-10}})));
54    Modelica.Blocks.Sources.RealExpression vm2_buck(y=-v2)
55      annotation (Placement(transformation(extent={{-80,20},{-60,40}})));
56    Modelica.Blocks.Sources.RealExpression vm1_boost(y=v1)
57      annotation (Placement(transformation(extent={{-80,-10},{-60,10}})));
58    Modelica.Blocks.Interfaces.RealInput vc "Buck control voltage" annotation (
59       Placement(transformation(
60        extent={{-20,-20},{20,20}},
61        rotation=90,
62        origin={-40,-120})));
63    Modelica.Blocks.Interfaces.BooleanInput mode "Boost control voltage"
64      annotation (Placement(transformation(
65        extent={{-20,-20},{20,20}},
66        rotation=90,
67        origin={40,-120})));
68    BidirectionalBuckBoost conv(
69      Cin=Cin,
70      Cout=Cout,
71      L=L,
72      RL=RL,
73      vCin_ini=vCin_ini,
74      vCout_ini=vCout_ini,
75      iL_ini=iL_ini,
76      Rcin=1e-3,
77      Rcout=1e-3,
78      dmax=1,
79      dmin=1e-3) annotation (Placement(transformation(extent={{66,80},{86,100}})));
80    Modelica.Blocks.Logical.Not not1 annotation (Placement(transformation(
81        extent={{10,-10},{-10,10}},
82        rotation=270,
83        origin={40,-30})));
84    Modelica.Blocks.MathBoolean.OnDelay onDelay(delayTime=3/fs) annotation (
85       Placement(transformation(
86        extent={{-4,-4},{4,4}},
87        rotation=90,
88        origin={10,-38})));
89    Modelica.Blocks.MathBoolean.OnDelay onDelay1(delayTime=3/fs) annotation (
90       Placement(transformation(
91        extent={{-4,-4},{4,4}},
92        rotation=90,
93        origin={40,-6})));
94  equation
95    connect(conv.p1, p1)
96      annotation (Line(points={{66,95},{-100,95},{-100,50}}, color={0,0,255}));
97    connect(conv.p2, p2)
98      annotation (Line(points={{86,95},{100,95},{100,50}}, color={0,0,255}));
99    connect(conv.n2, n2) annotation (Line(points={{86,85},{92,85},{92,-50},{100,-50}},
100       color={0,0,255}));
101   connect(conv.n1, n1) annotation (Line(points={{66,85},{-92,85},{-92,-50},{-100,
102         -50}}, color={0,0,255}));
103   connect(vm2_buck.y, buck_cpm.vm2)
104     annotation (Line(points={{-59,30},{-59,30},{28,30}}, color={0,0,127}));
```

```
  connect(vsense.y, boost_cpm.vs) annotation (Line(points={{-42.2,70},{-30,70},
        {-30,8},{-2,8}}, color={0,0,127}));
  connect(vm1_buck.y, boost_cpm.vm2) annotation (Line(points={{-59,-20},{-20,
        -20},{-20,-6},{-2,-6}}, color={0,0,127}));
  connect(vm1_boost.y, boost_cpm.vm1)
    annotation (Line(points={{-59,0},{-59,0},{-2,0}}, color={0,0,127}));
  connect(vm1_buck.y, buck_cpm.vm1) annotation (Line(points={{-59,-20},{-20,-20},
        {-20,36},{28,36}}, color={0,0,127}));
  connect(vsense.y, buck_cpm.vs) annotation (Line(points={{-42.2,70},{-30,70},{
        -30,44},{28,44}}, color={0,0,127}));
  connect(vc, buck_cpm.vc)
    annotation (Line(points={{-40,-120},{-40,50},{28,50}}, color={0,0,127}));
  connect(vc, boost_cpm.vc)
    annotation (Line(points={{-40,-120},{-40,14},{-2,14}}, color={0,0,127}));
  connect(mode, not1.u) annotation (Line(points={{40,-120},{40,-120},{40,-72},{
        40,-42}}, color={255,0,255}));
  connect(boost_cpm.d, conv.dboost)
    annotation (Line(points={{21,4},{80,4},{80,78}}, color={0,0,127}));
  connect(buck_cpm.d, conv.dbuck)
    annotation (Line(points={{51,40},{72,40},{72,78}}, color={0,0,127}));
  connect(mode, onDelay.u) annotation (Line(points={{40,-120},{40,-72},{10,-72},
        {10,-43.6}}, color={255,0,255}));
  connect(onDelay.y, boost_cpm.enable)
    annotation (Line(points={{10,-33.2},{10,-8}}, color={255,0,255}));
  connect(not1.y, onDelay1.u)
    annotation (Line(points={{40,-19},{40,-11.6}}, color={255,0,255}));
  connect(onDelay1.y, buck_cpm.enable)
    annotation (Line(points={{40,-1.2},{40,28}}, color={255,0,255}));
  annotation (
    Diagram(graphics={Text(
        extent={{-22,2},{22,-2}},
        lineColor={0,0,255},
        textString="vm1_buck = vm2_boost",
        origin={-112,12},
        rotation=90)}),
    Icon(graphics={
       Text(
        extent={{-60,30},{60,-30}},
        lineColor={0,0,255},
        fillColor={255,255,255},
        fillPattern=FillPattern.Solid,
        textString="1-ph"),
       Line(points={{-70,50},{-10,50}}, color={0,0,255}),
       Line(points={{-70,70},{-10,70}}, color={0,0,255}),
       Line(points={{10,-70},{70,-70}}, color={0,0,255}),
       Line(points={{10,-50},{70,-50}}, color={0,0,255})}),
    Documentation(info="<html><p>Bidirectional buck boost converter</p></html>"));
end CPMBidirectionalBuckBoost;
```

Electrical/Assemblies/HBridge.mo

```
within PVSystems.Electrical.Assemblies;
model HBridge "Basic ideal H-bridge topology (averaged)"
  extends Interfaces.TwoPort;
  extends PVSystems.Icons.ConverterIcon;
  Modelica.Blocks.Interfaces.RealInput d annotation (Placement(transformation(
      origin={0,-120},
      extent={{-20,-20},{20,20}},
      rotation=90)));
  replaceable model SwitchModel = CCM1 constrainedby
    Interfaces.SwitchNetworkInterface
    annotation(choicesAllMatching=true);
```

```
  SwitchModel s1 annotation (Placement(transformation(extent={{20,60},{40,80}},
        rotation=0)));
  SwitchModel s2 annotation (Placement(transformation(extent={{-40,-80},{-20,-60}},
        rotation=0)));
equation
  connect(s1.p1, p1) annotation (Line(points={{20,75},{-68,75},{-68,50},{-100,
        50}}, color={0,0,255}));
  connect(s1.n1, p2)
    annotation (Line(points={{20,65},{20,50},{100,50}}, color={0,0,255}));
  connect(s2.n1, n1) annotation (Line(points={{-40,-75},{-70,-75},{-70,-50},{-100,
        -50}}, color={0,0,255}));
  connect(s2.p1, n2) annotation (Line(points={{-40,-65},{-48,-65},{-48,-50},{
        100,-50}}, color={0,0,255}));
  connect(s1.n2, n1) annotation (Line(points={{40,65},{40,-20},{-100,-20},{-100,
        -50}}, color={0,0,255}));
  connect(d, s2.d) annotation (Line(points={{0,-120},{0,-92},{-30,-92},{-30,-82}},
      color={0,0,127}));
  connect(d, s1.d) annotation (Line(points={{0,-120},{0,30},{30,30},{30,58}},
      color={0,0,127}));
  connect(s2.p2, p1)
    annotation (Line(points={{-20,-65},{-20,50},{-100,50}}, color={0,0,255}));
  connect(s1.p2, p2) annotation (Line(points={{40,75},{70,75},{70,50},{100,50}},
      color={0,0,255}));
  connect(s2.n2, n2) annotation (Line(points={{-20,-75},{42,-75},{42,-50},{100,
        -50}}, color={0,0,255}));
  annotation (
    Diagram(graphics),
    Icon(graphics={Text(
        extent={{-60,30},{60,-30}},
        lineColor={0,0,255},
        fillColor={255,255,255},
        fillPattern=FillPattern.Solid,
        textString="1-ph"),Line(points={{-70,50},{-10,50}}, color={0,0,255}),
        Line(points={{-70,70},{-10,70}}, color={0,0,255}),Line(points={{10,-70},
        {70,-70}}, color={0,0,255}),Line(
        points={{10,-50},{24,-40},{40,-50},{56,-60},{70,-50}},
        color={0,0,255},
        smooth=Smooth.Bezier)}),
    Documentation(info="<html><p>This model further
    composes IdealAverageCCMSwitch to form a typical H-bridge
    configuration from which a 1-phase inverter can be constructed.
    This model is based in averaged switch models.</p></html>"));
end HBridge;
```

Electrical/Assemblies/HBridgeSwitched.mo

```
within PVSystems.Electrical.Assemblies;
model HBridgeSwitched "Basic ideal H-bridge topology (switched)"
  extends Interfaces.TwoPort;
  extends PVSystems.Icons.ConverterIcon;
  Modelica.Blocks.Interfaces.BooleanInput c1 annotation (Placement(
      transformation(
      origin={-40,-100},
      extent={{-10,-10},{10,10}},
      rotation=90)));
  Modelica.Blocks.Interfaces.BooleanInput c2 annotation (Placement(
      transformation(
      origin={40,-100},
      extent={{-10,-10},{10,10}},
      rotation=90)));
  IdealCBSwitch idealCBSwitch annotation (Placement(transformation(
      extent={{-10,-10},{10,10}},
```

```
      rotation=270,
      origin={0,30})));
  IdealCBSwitch idealCBSwitch1 annotation (Placement(transformation(
      extent={{-10,-10},{10,10}},
      rotation=270,
      origin={0,-30})));
  IdealCBSwitch idealCBSwitch2 annotation (Placement(transformation(
      extent={{-10,-10},{10,10}},
      rotation=270,
      origin={60,30})));
  IdealCBSwitch idealCBSwitch3 annotation (Placement(transformation(
      extent={{-10,-10},{10,10}},
      rotation=270,
      origin={60,-30})));
equation
  connect(c1, idealCBSwitch.c) annotation (Line(points={{-40,-100},{-40,-100},{
      -40,24},{-40,30},{-7,30}}, color={255,0,255}));
  connect(c1, idealCBSwitch3.c) annotation (Line(points={{-40,-100},{-40,-60},{
      20,-60},{20,-30},{53,-30}}, color={255,0,255}));
  connect(c2, idealCBSwitch2.c) annotation (Line(points={{40,-100},{40,-50},{40,
      30},{53,30}}, color={255,0,255}));
  connect(c2, idealCBSwitch1.c) annotation (Line(points={{40,-100},{40,-70},{-20,
      -70},{-20,-30},{-7,-30}}, color={255,0,255}));
  connect(p1, idealCBSwitch2.p) annotation (Line(points={{-100,50},{-40,50},{60,
      50},{60,40}}, color={0,0,255}));
  connect(idealCBSwitch.p, idealCBSwitch2.p)
    annotation (Line(points={{0,40},{0,50},{60,50},{60,40}}, color={0,0,255}));
  connect(idealCBSwitch1.p, idealCBSwitch.n)
    annotation (Line(points={{0,-20},{0,0},{0,20}}, color={0,0,255}));
  connect(idealCBSwitch2.n, idealCBSwitch3.p)
    annotation (Line(points={{60,20},{60,0},{60,-20}}, color={0,0,255}));
  connect(n1, idealCBSwitch1.n) annotation (Line(points={{-100,-50},{0,-50},{0,
      -40},{-1.77636e-015,-40}}, color={0,0,255}));
  connect(idealCBSwitch3.n, idealCBSwitch1.n) annotation (Line(points={{60,-40},
      {60,-50},{0,-50},{0,-40},{-1.77636e-015,-40}}, color={0,0,255}));
  connect(n2, idealCBSwitch3.p) annotation (Line(points={{100,-50},{80,-50},{80,
      -10},{60,-10},{60,-20}}, color={0,0,255}));
  connect(p2, idealCBSwitch.n) annotation (Line(points={{100,50},{80,50},{80,10},
      {0,10},{0,20},{-1.77636e-015,20}}, color={0,0,255}));
  annotation (Icon(graphics={Text(
      extent={{-60,30},{60,-30}},
      lineColor={0,0,255},
      fillColor={255,255,255},
      fillPattern=FillPattern.Solid,
      textString="1-ph"),Line(points={{-70,50},{-10,50}}, color={0,0,255}),
      Line(points={{-70,70},{-10,70}}, color={0,0,255}),Line(points={{10,-70},
      {70,-70}}, color={0,0,255}),Line(
      points={{10,-50},{24,-40},{40,-50},{56,-60},{70,-50}},
      color={0,0,255},
      smooth=Smooth.Bezier)}), Documentation(info="<html><p>This model further
    composes IdealTwoLevelBranch to form a typical H-bridge
    configuration from which a 1-phase inverter can be constructed.
    This model is based on discrete switch models.</p></html>"));
end HBridgeSwitched;
```

Electrical/Assemblies/package.mo

```
within PVSystems.Electrical;
package Assemblies "Electrical assemblies useful in PV and power electronics"
extends PVSystems.Icons.AssembliesPackage;


```

```
6
7
8 end Assemblies;
```

Electrical/Assemblies/package.order

```
1 HBridge
2 HBridgeSwitched
3 BidirectionalBuckBoost
4 CPMBidirectionalBuckBoost
```

Electrical/CCM1.mo

```
1 within PVSystems.Electrical;
2 model CCM1 "Average CCM model with no losses"
3   extends Interfaces.SwitchNetworkInterface;
4 equation
5   0 = p1.i + n1.i;
6   0 = p2.i + n2.i;
7   v1 = (1 - dsat)/dsat*v2;
8   -i2 = (1 - dsat)/dsat*i1;
9   annotation(Documentation(info="<html>
10      <p>
11        <em>Application</em>: two-switch PWM converters.
12      </p>
13
14      <p>
15        <em>Limitations</em>: ideal switches, CCM only, no transformer.
16      </p>
17
18      <p>
19        Model taken
20        from <a href=\"modelica://PVSystems.UsersGuide.References.EM01\">EM01</a>
21        and <a href=\"modelica://PVSystems.UsersGuide.References.EMA16\">EMA16</a>.</p>
22    </html>"));
23 end CCM1;
```

Electrical/CCM2.mo

```
1 within PVSystems.Electrical;
2 model CCM2 "Average CCM model with conduction losses"
3   extends Interfaces.SwitchNetworkInterface;
4   parameter Modelica.SIunits.Resistance Ron=0 "Transistor on resistance";
5   parameter Modelica.SIunits.Resistance RD=0 "Diode on resistance";
6   parameter Modelica.SIunits.Voltage VD=0 "Diode forward voltage drop";
7 equation
8   0 = p1.i + n1.i;
9   0 = p2.i + n2.i;
10  v1 = i1*(Ron/dsat + (1 - dsat)*RD/dsat^2) + (1 - dsat)/dsat*(v2 + VD);
11  -i2 = i1*(1 - dsat)/dsat;
12  annotation(Documentation(info="<html>
13      <p>
14        <em>Application</em>: two-switch PWM converters, includes
```

```
      conduction losses due to Ron, VD, Rd.
    </p>

    <p>
      <em>Limitations</em>: CCM only, no transformer.
    </p>

    <p>
      Model taken
      from <a href=\"modelica://PVSystems.UsersGuide.References.EM01\">EM01</a>
      and <a href=\"modelica://PVSystems.UsersGuide.References.EMA16\">EMA16</a>.</p>
   </html>"));
end CCM2;
```

Electrical/CCM3.mo

```
within PVSystems.Electrical;
model CCM3 "Average CCM model with no losses and tranformer"
  extends Interfaces.SwitchNetworkInterface;
  parameter Real n(final unit="1") = 1
    "Transformer turns ratio 1:n (primary:secondary)";
equation
  0 = p1.i + n1.i;
  0 = p2.i + n2.i;
  v1 = (1 - dsat)*v2/dsat/n;
  -i2 = (1 - dsat)*i1/dsat/n;
  annotation(Documentation(info="<html>
    <p>
      <em>Application</em>: two-switch PWM converters, with (possibly)
      transformer.
    </p>

    <p>
      <em>Limitations</em>: ideal switches, CCM only.
    </p>

    <p>
      Model taken
      from <a href=\"modelica://PVSystems.UsersGuide.References.EM01\">EM01</a>
      and <a href=\"modelica://PVSystems.UsersGuide.References.EMA16\">EMA16</a>.</p>
   </html>"));
end CCM3;
```

Electrical/CCM4.mo

```
within PVSystems.Electrical;
model CCM4 "Average CCM model with conduction losses and tranformer"
  extends Interfaces.SwitchNetworkInterface;
  parameter Modelica.SIunits.Resistance Ron=0 "Transistor on resistance";
  parameter Modelica.SIunits.Resistance RD=0 "Diode on resistance";
  parameter Modelica.SIunits.Voltage VD=0 "Diode forward voltage drop";
  parameter Real n(final unit="1") = 1
    "Transformer turns ratio 1:n (primary:secondary)";
equation
  0 = p1.i + n1.i;
  0 = p2.i + n2.i;
  v1 = i1*(Ron/dsat + (1 - dsat)*RD/n^2/dsat^2) + (1 - dsat)/dsat/n*(v2+VD);
```

```
  -i2 = i1*(1 - dsat)/dsat/n;
  annotation(Documentation(info="<html>
     <p>
       <em>Application</em>: two-switch PWM converters, includes
       conduction losses due to Ron, VD, RD and (possibly) transformer.
     </p>

     <p>
       <em>Limitations</em>: CCM only.
     </p>

     <p>
       Model taken
       from <a href=\"modelica://PVSystems.UsersGuide.References.EM01\">EM01</a>
       and <a href=\"modelica://PVSystems.UsersGuide.References.EMA16\">EMA16</a>.</p>
    </html>"));
end CCM4;
```

Electrical/CCM5.mo

```
within PVSystems.Electrical;
model CCM5
  "Average CCM model with conduction losses and diode reverse recovery"
  extends Interfaces.SwitchNetworkInterface;
  parameter Modelica.SIunits.Resistance Ron=0 "Transistor on resistance";
  parameter Modelica.SIunits.Voltage VD=0 "Diode forward voltage drop";
  parameter Modelica.SIunits.Charge Qr "Diode reverse recovery charge";
  parameter Modelica.SIunits.Time tr "Diode reverse recovery time";
  parameter Modelica.SIunits.Frequency fs "Switching frequency";
equation
  0 = p1.i + n1.i;
  0 = p2.i + n2.i;
  v1 = (i1 - fs*Qr)*Ron/(dsat + fs*tr) + (1 - dsat)/dsat*(v2 + VD);
  -i2 = i1*(1 - dsat - fs*tr)/(dsat + fs*tr) - fs*Qr/(dsat + fs*tr);
  annotation(Documentation(info="<html>
     <p>
       <em>Application</em>: two-switch PWM converters, includes
       conduction losses due to Ron, VD and diode reverse recovery
       losses.
     </p>

     <p>
       <em>Limitations</em>: CCM only, d'&gt;tr/Ts, &lt;i1&gt; &gt;
       Qr/Ts.
     </p>

     <p>
       Model taken
       from <a href=\"modelica://PVSystems.UsersGuide.References.EM01\">EM01</a>
       and <a href=\"modelica://PVSystems.UsersGuide.References.EMA16\">EMA16</a>.</p>
    </html>"));
end CCM5;
```

Electrical/CCM_DCM1.mo

```
within PVSystems.Electrical;
model CCM_DCM1 "Average CCM-DCM model with no losses"
```

```
  extends Interfaces.SwitchNetworkInterface;
  parameter Modelica.SIunits.Inductance Le "Equivalent DCM inductance";
  parameter Modelica.SIunits.Frequency fs "Switching frequency";
protected
  Real mu "Effective switch conversion ratio";
  Real Re "Equivalent DCM port 1 resistance";
equation
  0 = p1.i + n1.i;
  0 = p2.i + n2.i;
  Re = 2*Le*fs/dsat^2;
  mu = max(dsat, 1/(1 + Re*max(0,i1)/v2));
  v1 = (1 - mu)/mu*v2;
  -i2 = (1 - mu)/mu*i1;
  annotation(Documentation(info="<html>
     <p>
       <em>Application</em>: two-switch PWM converters, CCM or DCM.
     </p>

     <p>
       <em>Limitations</em>: ideal switches, no transformer.
     </p>

     <p>
       Model taken
       from <a href=\"modelica://PVSystems.UsersGuide.References.EM01\">EM01</a>
       and <a href=\"modelica://PVSystems.UsersGuide.References.EMA16\">EMA16</a>.</p>
    </html>"));
end CCM_DCM1;
```

Electrical/CCM_DCM2.mo

```
within PVSystems.Electrical;
model CCM_DCM2 "Average CCM-DCM model with no losses and transformer"
  extends Interfaces.SwitchNetworkInterface;
  parameter Modelica.SIunits.Inductance Le "Equivalent DCM inductance";
  parameter Modelica.SIunits.Frequency fs "Switching frequency";
  parameter Real n(final unit="1") = 1
    "Transformer turns ratio 1:n (primary:secondary)";
protected
  Real mu "Effective switch conversion ratio";
  Real Re "Equivalent DCM port 1 resistance";
equation
  0 = p1.i + n1.i;
  0 = p2.i + n2.i;
  Re = 2*Le*n*fs/dsat^2;
  mu = max(dsat, 1/(1 + Re*max(0,i1)/v2));
  v1 = (1 - mu)*v2/mu/n;
  -i2 = (1 - mu)*i1/mu/n;
  annotation(Documentation(info="<html>
     <p>
       <em>Application</em>: two-switch PWM converters, CCM or DCM with
       (possibly) transformer.
     </p>

     <p>
       <em>Limitations</em>: ideal switches.
     </p>

     <p>
       Model taken
       from <a href=\"modelica://PVSystems.UsersGuide.References.EM01\">EM01</a>
       and <a href=\"modelica://PVSystems.UsersGuide.References.EMA16\">EMA16</a>.</p>
```

```
    </html>"));
end CCM_DCM2;
```

Electrical/IdealCBSwitch.mo

```
within PVSystems.Electrical;
model IdealCBSwitch "Basic two-cuadrant current bidirectional switch"
  extends Modelica.Electrical.Analog.Interfaces.TwoPin;
  // Components
  Modelica.Electrical.Analog.Ideal.IdealClosingSwitch idealClosingSwitch
    annotation (Placement(transformation(extent={{-10,10},{10,-10}},rotation=0)));
  Modelica.Electrical.Analog.Ideal.IdealDiode idealDiode annotation (Placement(
      transformation(
      origin={0,40},
      extent={{-10,-10},{10,10}},
      rotation=180)));
  Modelica.Blocks.Interfaces.BooleanInput c annotation (Placement(
      transformation(
      origin={0,-70},
      extent={{-10,-10},{10,10}},
      rotation=90)));
equation
  connect(p, idealClosingSwitch.p)
    annotation (Line(points={{-100,0},{-10,0}}, color={0,0,255}));
  connect(idealClosingSwitch.n, n)
    annotation (Line(points={{10,0},{100,0}}, color={0,0,255}));
  connect(idealDiode.p, n) annotation (Line(points={{10,40},{60,40},{60,0},{100,
      0}}, color={0,0,255}));
  connect(idealDiode.n, p) annotation (Line(points={{-10,40},{-60,40},{-60,0},{
      -100,0}}, color={0,0,255}));
  connect(c, idealClosingSwitch.control)
    annotation (Line(points={{0,-70},{0,-7}}, color={255,0,255}));
  annotation (Icon(graphics={Line(points={{-98,0},{-20,0}}, color={0,0,255}),
      Line(points={{-20,-20},{20,0},{100,0}}, color={0,0,255}),Line(points=
      {{-40,0},{-40,40},{-20,40}}, color={0,0,255}),Line(points={{-20,40},{
      10,60},{10,20},{-20,40}}, color={0,0,255}),Line(points={{10,40},{40,
      40},{40,0}}, color={0,0,255}),Line(points={{-20,60},{-20,20}}, color=
      {0,0,255}),Line(points={{0,-78},{0,-10}}, color={255,85,255})}),
    Documentation(info="<html>
    <p>This model represents and idealized current bi-directional
    switch. This is the typical IGBT in anti-parallel with a diode from
    which many converters are built.</p>
    </html>"));
end IdealCBSwitch;
```

Electrical/Interfaces/BatteryInterface.mo

```
within PVSystems.Electrical.Interfaces;
partial model BatteryInterface "Partial model for battery"
  extends Modelica.Electrical.Analog.Interfaces.OnePort;
  annotation (Documentation(info=
      "<html><p>Partial model for battery</p></html>"), Icon(
      coordinateSystem(
      preserveAspectRatio=false,
      extent={{-100,-100},{100,100}},
      grid={2,2}), graphics={Line(points={{-90,0},{-50,0}}, color={0,0,0}),
        Line(points={{50,0},{90,0}}, color={0,0,0}),Line(points={{-50,40},{-50,
```

```
        -40}}, color={0,0,0}),Line(points={{-20,20},{-20,-20}}, color={0,0,0}),
        Line(points={{-20,0},{20,0}}, color={0,0,0}),Line(points={{20,40},{20,
        -40}}, color={0,0,0}),Line(points={{50,20},{50,-20}}, color={0,0,0}),
        Text(
        extent={{-80,-40},{80,-80}},
        lineColor={28,108,200},
        textString="%name")}));
end BatteryInterface;
```

Electrical/Interfaces/package.mo

```
within PVSystems.Electrical;
package Interfaces "Interfaces"
extends Modelica.Icons.InterfacesPackage;
end Interfaces;
```

Electrical/Interfaces/package.order

```
BatteryInterface
SwitchNetworkInterface
TwoPort
```

Electrical/Interfaces/SwitchNetworkInterface.mo

```
within PVSystems.Electrical.Interfaces;
partial model SwitchNetworkInterface
  "Interface for the averaged switch network models"
  extends TwoPort;
  parameter Real dmin(final unit="1") = 1e-3 "Minimum duty cycle";
  parameter Real dmax(final unit="1") = 1 "Maximum duty cycle";
  Modelica.Blocks.Interfaces.RealInput d "Duty cycle" annotation (Placement(
      transformation(
      origin={0,-120},
      extent={{-20,-20},{20,20}},
      rotation=90)));
protected
  Real dsat(final unit="1") = smooth(0, if d > dmax then dmax else if d < dmin
    then dmin else d) "Saturated duty cycle";
  annotation (Icon(graphics={Polygon(
        points={{60,20},{40,-20},{80,-20},{60,20}},
        lineColor={0,0,0},
        fillColor={255,255,255}),Line(points={{60,50},{60,-50}}),Line(points=
        {{60,50},{90,50}}),Line(points={{80,20},{40,20}}, color={0,0,255}),
        Text(extent={{-100,100},{100,70}}, textString="%name"),Line(points={{
        60,-50},{90,-50}}),Line(points={{-60,-50},{-60,50}}),Line(points={{-60,
        30},{-80,30}}),Line(points={{-48,30},{-48,-30}}),Line(points={{-80,0},
        {-80,-50}}),Line(points={{-90,-50},{-80,-50}}),Line(points={{-80,30},
        {-80,50}}),Line(points={{-90,50},{-80,50}}),Polygon(
        points={{-60,0},{-70,5},{-70,-5},{-60,0}},
        lineColor={28,108,200},
        fillColor={0,0,255},
        fillPattern=FillPattern.Solid),Line(points={{-60,-30},{-80,-30}}),
        Line(points={{-60,0},{-80,0}}),Polygon(
```

```
30         points={{0,-40},{-10,-60},{10,-60},{0,-40}},
31         lineColor={0,0,0},
32         fillColor={255,255,255}),Line(points={{0,-60},{0,-100}}),Line(points=
33         {{0,0},{0,-40}}),Line(points={{-46,0},{0,0}})));
34 end SwitchNetworkInterface;
```

Electrical/Interfaces/TwoPort.mo

```
1  within PVSystems.Electrical.Interfaces;
2  partial model TwoPort "Common interface for power converters with two ports"
3    Modelica.SIunits.Voltage v1 "Voltage drop over the left port";
4    Modelica.SIunits.Voltage v2 "Voltage drop over the right port";
5    Modelica.SIunits.Current i1
6      "Current flowing from pos. to neg. pin of the left port";
7    Modelica.SIunits.Current i2
8      "Current flowing from pos. to neg. pin of the right port";
9    Modelica.Electrical.Analog.Interfaces.PositivePin p1
10     "Positive pin of the left port (potential p1.v > n1.v for positive voltage drop v1)"
11     annotation (Placement(transformation(extent={{-110,40},{-90,60}})));
12   Modelica.Electrical.Analog.Interfaces.NegativePin n1
13     "Negative pin of the left port"
14     annotation (Placement(transformation(extent={{-90,-60},{-110,-40}})));
15   Modelica.Electrical.Analog.Interfaces.PositivePin p2
16     "Positive pin of the right port (potential p2.v > n2.v for positive voltage drop v2)"
17     annotation (Placement(transformation(extent={{110,40},{90,60}})));
18   Modelica.Electrical.Analog.Interfaces.NegativePin n2
19     "Negative pin of the right port"
20     annotation (Placement(transformation(extent={{90,-60},{110,-40}})));
21 equation
22   v1 = p1.v - n1.v;
23   v2 = p2.v - n2.v;
24   i1 = p1.i;
25   i2 = p2.i;
26 end TwoPort;
```

Electrical/package.mo

```
1  within PVSystems;
2  package Electrical "Library for electrical models"
3  extends Modelica.Icons.Package;
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
```

```
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37 annotation (Icon(coordinateSystem(preserveAspectRatio=true, extent={{-100.0,-100.0},
38        {100.0,100.0}}), graphics={
39     Rectangle(origin={20.3125,82.8571}, extent={{-45.3125,-57.8571},{4.6875,-27.8571}}),
40     Line(origin={8.0,48.0}, points={{32.0,-58.0},{72.0,-58.0}}),
41     Line(origin={9.0,54.0}, points={{31.0,-49.0},{71.0,-49.0}}),
42     Line(origin={-2.0,55.0}, points={{-83.0,-50.0},{-33.0,-50.0}}),
43     Line(origin={-3.0,45.0}, points={{-72.0,-55.0},{-42.0,-55.0}}),
44     Line(origin={1.0,50.0}, points={{-61.0,-45.0},{-61.0,-10.0},{-26.0,-10.0}}),
45     Line(origin={7.0,50.0}, points={{18.0,-10.0},{53.0,-10.0},{53.0,-45.0}}),
46     Line(origin={6.2593,48.0}, points={{53.7407,-58.0},{53.7407,-93.0},{-66.2593,
47         -93.0},{-66.2593,-58.0}}})));
48 end Electrical;
```

Electrical/package.order

```
1 IdealCBSwitch
2 SW1
3 SW2
4 SW3
5 CCM1
6 CCM2
7 CCM3
8 CCM4
9 CCM5
10 CCM_DCM1
11 CCM_DCM2
12 PVArray
13 SimpleBattery
14 Assemblies
15 Interfaces
```

Electrical/PVArray.mo

```
1 within PVSystems.Electrical;
2 model PVArray "Flexible PV array model"
3   extends Modelica.Electrical.Analog.Interfaces.OnePort;
4   // Interface
5   Modelica.Blocks.Interfaces.RealInput G "Solar irradiation" annotation (
6     Placement(transformation(
7       origin={-30,-55},
8       extent={{-15,-15},{15,15}},
```

```
      rotation=90)));
  Modelica.Blocks.Interfaces.RealInput T "Panel temperature" annotation (
     Placement(transformation(
      origin={30,-55},
      extent={{-15,-15},{15,15}},
      rotation=90)));
  // Constants
  constant Modelica.SIunits.Charge q=1.60217646e-19 "Electron charge";
  constant Real Gn=1000 "STC irradiation";
  constant Modelica.SIunits.Temperature Tn=298.15 "STC temperature";
  // Basic datasheet parameters
  parameter Modelica.SIunits.Current Imp=7.61 "Maximum power current";
  parameter Modelica.SIunits.Voltage Vmp=26.3 "Maximum power voltage";
  parameter Modelica.SIunits.Current Iscn=8.21 "Short circuit current";
  parameter Modelica.SIunits.Voltage Vocn=32.9 "Open circuit voltage";
  parameter Real Kv=-0.123 "Voc temperature coefficient";
  parameter Real Ki=3.18e-3 "Isc temperature coefficient";
  // Basic model parameters
  parameter Real Ns=54 "Number of cells in series";
  parameter Real Np=1 "Number of cells in parallel";
  parameter Modelica.SIunits.Resistance Rs=0.221
    "Equivalent series resistance of array";
  parameter Modelica.SIunits.Resistance Rp=415.405
    "Equivalent parallel resistance of array";
  parameter Real a=1.3 "Diode ideality constant";
  // Derived model parameters
  parameter Modelica.SIunits.Current Ipvn=Iscn "Photovoltaic current at STC";
  // Variables
  Modelica.SIunits.Voltage Vt "Thermal voltage of the array";
  Modelica.SIunits.Current Ipv "Photovoltaic current of the cell";
  Modelica.SIunits.Current I0 "Saturation current of the cell";
  Modelica.SIunits.Current Id "Diode current";
  Modelica.SIunits.Current Ir "Rp current";
equation
  // Auxiliary variables
  Vt = Ns*Modelica.Constants.k*T/q;
  Ipv = (Ipvn + Ki*(T - Tn))*G/Gn;
  I0 = (Iscn + Ki*(T - Tn))/(exp((Vocn + Kv*(T - Tn))/a/Vt) - 1);
  Id = I0*(exp((v - Rs*i)/a/Vt) - 1);
  Ir = (v - Rs*i)/Rp;
  if v < 0 then
    i = v/((Rs + Rp)/Np);
  elseif v > Vocn then
    i = 0;
  else
    i = -Np*(Ipv - Id - Ir);
  end if;
  annotation (
    Documentation(info="<html><p>Flexible PV array model. The model can be
  parametrized with the use of PV module datasheets. As a default, the
  data from the Kyocera KC200GT is provided. The model is presented in
  \"Comprehensive Approach to Modeling and Simulation of Photovoltaic
  Arrays\" by M.G. Villalva et al.</p></html>"),
    Icon(coordinateSystem(
       preserveAspectRatio=false,
       extent={{-100,-100},{100,100}},
       grid={2,2}), graphics={Line(points={{-90,0},{-60,0}}, color={0,0,0}),
         Rectangle(
         extent={{-60,-40},{60,40}},
         lineColor={0,0,0},
         fillColor={255,255,255},
         fillPattern=FillPattern.Solid),Line(points={{-60,-40},{-20,0}}, color=
          {0,0,0}),Line(points={{-20,0},{-60,40}}, color={0,0,0}),Line(points=
         {{60,0},{90,0}}, color={0,0,0})})),
    Diagram(coordinateSystem(
       preserveAspectRatio=false,
```

```
75        extent={{-100,-100},{100,100}},
76        grid={2,2})));
77 end PVArray;
```

Electrical/SimpleBattery.mo

```
1  within PVSystems.Electrical;
2  model SimpleBattery "Simple battery model"
3    extends Interfaces.BatteryInterface;
4    import Modelica.SIunits.Resistance;
5    import Modelica.SIunits.Voltage;
6    import Modelica.SIunits.Current;
7    type BatteryCapacity = Real (final quantity="Energy", final unit="A.h");
8    // Parameters (Li-ion values as defaults)
9    parameter Resistance Rint=0.09 "Internal resistance";
10   parameter Voltage E0=3.7348 "Constant battery voltage";
11   parameter Voltage K=0.00876 "Polarization voltage";
12   parameter BatteryCapacity Q=1 "Rated battery capacity";
13   parameter Voltage A=0.468 "Exponential region amplitude";
14   parameter Real B=3.5294 "Exponential zone time constant inverse";
15   parameter BatteryCapacity DoDini=0 "Initial Depth of Discharge";
16   // Variables
17   Voltage E;
18   BatteryCapacity it(start=DoDini, fixed=true) "Actual depth of discharge";
19 equation
20   v = E + i*Rint;
21   der(it) = -i/3600;
22   E = max(0, E0 - K*Q/(Q - it) + A*exp(-B*it));
23 end SimpleBattery;
```

Electrical/SW1.mo

```
1  within PVSystems.Electrical;
2  model SW1 "Switched model implemented with switch + diode"
3    extends Interfaces.SwitchNetworkInterface;
4    Modelica.Electrical.Analog.Ideal.IdealClosingSwitch sw1(Ron=Ron) annotation (
5       Placement(transformation(
6         extent={{-10,-10},{10,10}},
7         rotation=270,
8         origin={-40,0})));
9    Modelica.Electrical.Analog.Ideal.IdealDiode sw2(Ron=RD, Vknee=VD) annotation (
10     Placement(transformation(
11        extent={{-10,-10},{10,10}},
12        rotation=90,
13        origin={40,0})));
14   Control.SwitchingPWM signalPWM(
15     dMax=dMax,
16     dMin=dMin,
17     fs=fs,
18     startTime=startTime) annotation (Placement(transformation(
19        extent={{10,10},{-10,-10}},
20        rotation=270,
21        origin={0,-70})));
22   parameter Real dMax=1 "Maximum duty cycle";
23   parameter Real dMin=0 "Minimum duty cycle";
24   parameter Modelica.SIunits.Frequency fs "Switching frequency";
25   parameter Modelica.SIunits.Time startTime=0 "Start time";
```

```
  parameter Modelica.SIunits.Resistance RD=1.E-5
    "Forward state-on differential resistance (closed resistance)";
  parameter Modelica.SIunits.Voltage VD=0 "Forward threshold voltage";
  parameter Modelica.SIunits.Resistance Ron=1.E-5 "Closed switch resistance";
equation
  connect(p1, sw1.p)
    annotation (Line(points={{-100,50},{-40,50},{-40,10}}, color={0,0,255}));
  connect(n1, sw1.n) annotation (Line(points={{-100,-50},{-40,-50},{-40,-10}},
      color={0,0,255}));
  connect(sw2.n, p2) annotation (Line(points={{40,10},{40,10},{40,50},{100,50}},
      color={0,0,255}));
  connect(n2, sw2.p)
    annotation (Line(points={{100,-50},{40,-50},{40,-10}}, color={0,0,255}));
  connect(signalPWM.vc, d)
    annotation (Line(points={{0,-82},{0,-120},{0,-120}}, color={0,0,127}));
  connect(signalPWM.c1, sw1.control) annotation (Line(points={{2.22045e-015,-59},
        {0,-59},{0,-1.33227e-015},{-33,-1.33227e-015}}, color={255,0,255}));
  annotation (Icon(coordinateSystem(preserveAspectRatio=false)), Diagram(
      coordinateSystem(preserveAspectRatio=false)));
end SW1;
```

Electrical/SW2.mo

```
within PVSystems.Electrical;
model SW2 "Switched model implemented with switch x 2"
  extends Interfaces.SwitchNetworkInterface;
  Modelica.Electrical.Analog.Ideal.IdealClosingSwitch sw1(Ron=Ron) annotation (
     Placement(transformation(
       extent={{-10,-10},{10,10}},
       rotation=270,
       origin={-40,0})));
  Control.SwitchingPWM spwm(
    dMax=dMax,
    dMin=dMin,
    fs=fs,
    startTime=startTime) annotation (Placement(transformation(
       extent={{10,10},{-10,-10}},
       rotation=270,
       origin={0,-70})));
  parameter Real dMax=1 "Maximum duty cycle";
  parameter Real dMin=0 "Minimum duty cycle";
  parameter Modelica.SIunits.Frequency fs "Switching frequency";
  parameter Modelica.SIunits.Time startTime=0 "Start time";
  parameter Modelica.SIunits.Resistance RD=1.E-5
    "Forward state-on differential resistance (closed resistance)";
  parameter Modelica.SIunits.Voltage VD=0 "Forward threshold voltage";
  parameter Modelica.SIunits.Resistance Ron=1.E-5 "Closed switch resistance";
  Modelica.Electrical.Analog.Ideal.IdealClosingSwitch sw2(Ron=Ron) annotation (
     Placement(transformation(
       extent={{10,10},{-10,-10}},
       rotation=270,
       origin={40,0})));
  Control.DeadTime dt(deadTime=deadTime) annotation (Placement(transformation(
       extent={{-10,-10},{10,10}},
       rotation=90,
       origin={0,-30})));
  parameter Modelica.SIunits.Time deadTime=0 "Dead time";
equation
  connect(p1, sw1.p)
    annotation (Line(points={{-100,50},{-40,50},{-40,10}}, color={0,0,255}));
  connect(n1, sw1.n) annotation (Line(points={{-100,-50},{-40,-50},{-40,-10}},
      color={0,0,255}));
```

```
  connect(spwm.vc, d)
    annotation (Line(points={{0,-82},{0,-120}}, color={0,0,127}));
  connect(dt.c, spwm.c1)
    annotation (Line(points={{0,-42},{0,-59}}, color={255,0,255}));
  connect(dt.c1, sw1.control) annotation (Line(points={{-4,-19},{-4,-19},{-4,0},
        {-33,0}}, color={255,0,255}));
  connect(sw2.n, p2) annotation (Line(points={{40,10},{40,10},{40,50},{100,50}},
      color={0,0,255}));
  connect(n2, sw2.p)
    annotation (Line(points={{100,-50},{40,-50},{40,-10}}, color={0,0,255}));
  connect(dt.c2, sw2.control) annotation (Line(points={{4,-19},{4,-19},{4,0},{
        33,0}}, color={255,0,255}));
  annotation (Icon(coordinateSystem(preserveAspectRatio=false)), Diagram(
      coordinateSystem(preserveAspectRatio=false)));
end SW2;
```

Electrical/SW3.mo

```
within PVSystems.Electrical;
model SW3 "Switched model implemented with switch + anti-parallel diode x 2"
  extends Interfaces.SwitchNetworkInterface;
  Modelica.Electrical.Analog.Ideal.IdealClosingSwitch sw1(Ron=Ron) annotation (
     Placement(transformation(
       extent={{-10,-10},{10,10}},
       rotation=270,
       origin={-40,0})));
  Control.SwitchingPWM spwm(
    dMax=dMax,
    dMin=dMin,
    fs=fs,
    startTime=startTime) annotation (Placement(transformation(
       extent={{10,10},{-10,-10}},
       rotation=270,
       origin={0,-70})));
  parameter Real dMax=1 "Maximum duty cycle";
  parameter Real dMin=0 "Minimum duty cycle";
  parameter Modelica.SIunits.Frequency fs "Switching frequency";
  parameter Modelica.SIunits.Time startTime=0 "Start time";
  parameter Modelica.SIunits.Resistance RD=1.E-5
    "Forward state-on differential resistance (closed resistance)";
  parameter Modelica.SIunits.Voltage VD=0 "Forward threshold voltage";
  parameter Modelica.SIunits.Resistance Ron=1.E-5 "Closed switch resistance";
  Modelica.Electrical.Analog.Ideal.IdealClosingSwitch sw2(Ron=Ron) annotation (
     Placement(transformation(
       extent={{10,10},{-10,-10}},
       rotation=270,
       origin={40,0})));
  Control.DeadTime dt(deadTime=deadTime) annotation (Placement(transformation(
       extent={{-10,-10},{10,10}},
       rotation=90,
       origin={0,-30})));
  Modelica.Electrical.Analog.Ideal.IdealDiode d1 annotation (Placement(
       transformation(
       extent={{-10,-10},{10,10}},
       rotation=90,
       origin={-80,0})));
  Modelica.Electrical.Analog.Ideal.IdealDiode d2 annotation (Placement(
       transformation(
       extent={{-10,-10},{10,10}},
       rotation=90,
       origin={80,0})));
  parameter Modelica.SIunits.Time deadTime=0 "Dead time";
```

```
equation
  connect(p1, sw1.p)
    annotation (Line(points={{-100,50},{-40,50},{-40,10}}, color={0,0,255}));
  connect(n1, sw1.n) annotation (Line(points={{-100,-50},{-40,-50},{-40,-10}},
      color={0,0,255}));
  connect(spwm.vc, d)
    annotation (Line(points={{0,-82},{0,-120}}, color={0,0,127}));
  connect(dt.c, spwm.c1)
    annotation (Line(points={{0,-42},{0,-59}}, color={255,0,255}));
  connect(dt.c1, sw1.control) annotation (Line(points={{-4,-19},{-4,-19},{-4,0},
      {-33,0}}, color={255,0,255}));
  connect(sw2.n, p2) annotation (Line(points={{40,10},{40,10},{40,50},{100,50}},
      color={0,0,255}));
  connect(n2, sw2.p)
    annotation (Line(points={{100,-50},{40,-50},{40,-10}}, color={0,0,255}));
  connect(dt.c2, sw2.control) annotation (Line(points={{4,-19},{4,-19},{4,0},{
      33,0}}, color={255,0,255}));
  connect(d1.n, sw1.p) annotation (Line(points={{-80,10},{-80,50},{-40,50},{-40,
      10}}, color={0,0,255}));
  connect(d1.p, sw1.n) annotation (Line(points={{-80,-10},{-80,-50},{-40,-50},{
      -40,-10}}, color={0,0,255}));
  connect(d2.n, p2) annotation (Line(points={{80,10},{80,10},{80,50},{100,50}},
      color={0,0,255}));
  connect(d2.p, n2) annotation (Line(points={{80,-10},{80,-10},{80,-50},{100,
      -50}}, color={0,0,255}));
  annotation (Icon(coordinateSystem(preserveAspectRatio=false)), Diagram(
      coordinateSystem(preserveAspectRatio=false)));
end SW3;
```

Examples/Application/BuckOpen.mo

```
within PVSystems.Examples.Application;
model BuckOpen "Ideal open-loop buck converter"
  extends Modelica.Icons.Example;
  Modelica.Electrical.Analog.Sources.ConstantVoltage DC(V=24)
                                                 annotation (
     Placement(transformation(
       origin={-30,-40},
       extent={{-10,-10},{10,10}},
       rotation=270)));
  Modelica.Electrical.Analog.Basic.Resistor Rav(R=3) annotation (Placement(
       transformation(
       origin={70,-50},
       extent={{-10,-10},{10,10}},
       rotation=270)));
  Modelica.Electrical.Analog.Basic.Inductor Lav(L=8e-6) annotation (Placement(
       transformation(extent={{20,-40},{40,-20}}, rotation=0)));
  Modelica.Electrical.Analog.Basic.Capacitor Cav(C=10e-6) annotation (
     Placement(transformation(
       origin={50,-50},
       extent={{-10,-10},{10,10}},
       rotation=270)));
  replaceable Electrical.CCM_DCM1 sn(Le=Lav.L, fs=PWM.fs)
                                          constrainedby
    PVSystems.Electrical.Interfaces.SwitchNetworkInterface annotation (
      Placement(transformation(extent={{-10,-36},{10,-16}},
                                            rotation=0)),
     choicesAllMatching=true);
  Modelica.Electrical.Analog.Ideal.IdealClosingSwitch sw annotation (Placement(
       transformation(extent={{-20,70},{0,50}}, rotation=0)));
  Modelica.Electrical.Analog.Ideal.IdealDiode dsw annotation (Placement(
       transformation(
```

```
      origin={10,40},
      extent={{-10,-10},{10,10}},
      rotation=90)));
  Control.SwitchingPWM PWM(fs=1e5) annotation (Placement(transformation(extent={{-10,-10},
        {10,10}}, rotation=90,
      origin={-10,30})));
  Modelica.Electrical.Analog.Basic.Resistor Rsw(R=3) annotation (Placement(
      transformation(
      origin={70,40},
      extent={{-10,-10},{10,10}},
      rotation=270)));
  Modelica.Electrical.Analog.Basic.Inductor Lsw(L=8e-6) annotation (Placement(
      transformation(extent={{20,50},{40,70}}, rotation=0)));
  Modelica.Electrical.Analog.Basic.Capacitor Csw(C=10e-6) annotation (
     Placement(transformation(
      origin={50,40},
      extent={{-10,-10},{10,10}},
      rotation=270)));
  Modelica.Electrical.Analog.Basic.Ground gin annotation (Placement(
      transformation(extent={{-40,-76},{-20,-56}}, rotation=0)));
  Modelica.Electrical.Analog.Basic.Ground gsw annotation (Placement(
      transformation(extent={{20,10},{40,30}}, rotation=0)));
  Modelica.Electrical.Analog.Basic.Ground gav annotation (Placement(
      transformation(extent={{20,-80},{40,-60}}, rotation=0)));
  Modelica.Blocks.Sources.RealExpression duty(y=if time < 5e-4 then 0.1 else 0.6)
    annotation (Placement(transformation(extent={{-90,-90},{-70,-70}})));
equation
  connect(Cav.n, gav.p)
    annotation (Line(points={{50,-60},{30,-60}}, color={0,0,255}));
  connect(Rav.n, gav.p)
    annotation (Line(points={{70,-60},{30,-60}}, color={0,0,255}));
  connect(Lav.n, Rav.p)
    annotation (Line(points={{40,-30},{70,-30},{70,-40}}, color={0,0,255}));
  connect(Cav.p, Lav.n)
    annotation (Line(points={{50,-40},{50,-30},{40,-30}}, color={0,0,255}));
  connect(DC.p, sn.p1)
    annotation (Line(points={{-30,-30},{-30,-21},{-10,-21}}, color={0,0,255}));
  connect(sn.p2, Lav.p)
    annotation (Line(points={{10,-21},{20,-21},{20,-30}}, color={0,0,255}));
  connect(sn.n2, gav.p)
    annotation (Line(points={{10,-31},{10,-60},{30,-60}}, color={0,0,255}));
  connect(sw.p, DC.p)
    annotation (Line(points={{-20,60},{-30,60},{-30,-30}}, color={0,0,255}));
  connect(sw.n, dsw.n)
    annotation (Line(points={{0,60},{10,60},{10,50}}, color={0,0,255}));
  connect(Lsw.n, Rsw.p)
    annotation (Line(points={{40,60},{70,60},{70,50}}, color={0,0,255}));
  connect(Csw.p, Lsw.n)
    annotation (Line(points={{50,50},{50,60},{40,60}}, color={0,0,255}));
  connect(Lsw.p, dsw.n)
    annotation (Line(points={{20,60},{10,60},{10,50}}, color={0,0,255}));
  connect(Csw.n, Rsw.n)
    annotation (Line(points={{50,30},{70,30}}, color={0,0,255}));
  connect(dsw.p, gsw.p)
    annotation (Line(points={{10,30},{30,30}}, color={0,0,255}));
  connect(gsw.p, Csw.n)
    annotation (Line(points={{30,30},{50,30}}, color={0,0,255}));
  connect(gin.p, DC.n)
    annotation (Line(points={{-30,-56},{-30,-53},{-30,-50}}, color={0,0,255}));
  connect(PWM.c1, sw.control)
    annotation (Line(points={{-10,41},{-10,41},{-10,53}}, color={255,0,255}));
  connect(PWM.vc, sn.d) annotation (Line(points={{-10,18},{-10,10},{-50,10},{-50,
        -80},{0,-80},{0,-38}}, color={0,0,127}));
  connect(sn.n1, Lav.p) annotation (Line(points={{-10,-31},{-20,-31},{-20,-4},{20,
        -4},{20,-30}}, color={0,0,255}));
  connect(duty.y, sn.d)
```

```
98    annotation (Line(points={{-69,-80},{0,-80},{0,-38}}, color={0,0,127}));
99  annotation (
100   Diagram(graphics={Text(
101       extent={{32,-8},{70,-20}},
102       lineColor={0,0,255},
103       textString="Modifiable model"),
104                 Text(
105       extent={{32,78},{70,66}},
106       lineColor={0,0,255},
107       textString="Switched model")}),
108   experiment(StopTime=0.001, __Dymola_NumberOfIntervals=1000),
109   Documentation(info="<html>
110      <p>
111       This example compares two implementations of a buck
112       DC-DC converter. The switched version is built using
113       mostly blocks
114       from <a href=\"Modelica://Modelica.Electrical.Analog\">Modelica's
115       electrical library</a> but also includes
116       the <a href=\"Modelica://PVSystems.Control.SwitchingPWM\">SwitchingPWM</a>
117       model. The averaged version is built around a
118       replaceable instance of the average switch model for CCM
119       (continuous conduction mode) and DCM (discontinuous
120       conduction mode) considering no losses.
121      </p>
122
123      <p>
124       This example showcases how components from PVSystems can
125       be mixed with components from the Modelica Standard
126       Library to build systems that might be of
127       interest. Additionally, it aims validating the average
128       switch model performance by comparison with the more
129       accurate/detailed switched model.
130      </p>
131
132      <p>
133       This is still an open-loop system. A duty cycle value is
134       fed to the SwitchingPWM block to drive the ideal closing
135       switch or to the averaged switch network model. The duty
136       cycle value begins at 0.1 and changes to 0.6 in the
137       middle of the simulation. The effect of this change can
138       be observed by plotting the output voltages:
139      </p>
140
141
142      <div class=\"figure\">
143       <p><img src=\"modelica://PVSystems/Resources/Images/BuckOpenResultsA.png\"
144             alt=\"BuckOpenResultsA.png\" />
145       </p>
146      </div>
147
148      <p>
149       The output voltage for both implementations is not
150       exactly the same but it can be seen that the averaged
151       model provides a very decent approximation. This is the
152       case because both the switching and the averaged
153       implementations are neglecting losses and because they
154       are both correctly modelling CCM and DCM modes. The
155       converter is operating in DCM in the first interval and
156       in CCM in the second:
157      </p>
158
159
160      <div class=\"figure\">
161       <p><img src=\"modelica://PVSystems/Resources/Images/BuckOpenResultsB.png\"
162             alt=\"BuckOpenResultsB.png\" />
163       </p>
```

```
164       </div>
165
166       <p>
167         An interesting exercise to complete this example would
168         be to build a controller to close the loop and study the
169         system's behaviour.</p>
170     </html>"),
171    __Dymola_experimentSetupOutput);
172 end BuckOpen;
```

Examples/Application/Inverter1phClosed.mo

```
1  within PVSystems.Examples.Application;
2  model Inverter1phClosed
3    "Stand-alone 1-phase closed-loop inverter with constant DC source"
4    extends Modelica.Icons.Example;
5    Modelica.Electrical.Analog.Sources.ConstantVoltage DC(V=580) annotation (
6       Placement(transformation(
7         origin={-20,50},
8         extent={{-10,-10},{10,10}},
9         rotation=270)));
10   Modelica.Electrical.Analog.Basic.Ground ground annotation (Placement(
11        transformation(extent={{-30,14},{-10,34}}, rotation=0)));
12   PVSystems.Electrical.Assemblies.HBridge HB annotation (Placement(
13        transformation(extent={{20,40},{40,60}}, rotation=0)));
14   Modelica.Electrical.Analog.Basic.Resistor R(R=1e-2) annotation (Placement(
15        transformation(
16        origin={70,30},
17        extent={{-10,-10},{10,10}},
18        rotation=270)));
19   Modelica.Electrical.Analog.Basic.Inductor L(L=1e-3) annotation (Placement(
20        transformation(
21        origin={70,70},
22        extent={{-10,-10},{10,10}},
23        rotation=270)));
24   Modelica.Blocks.Sources.Step iqSetpoint(height=141.4, startTime=0.3)
25     annotation (Placement(transformation(extent={{-70,0},{-50,20}}, rotation=0)));
26   Modelica.Blocks.Sources.Step idSetpoint(
27     height=141.4 - 200,
28     offset=200,
29     startTime=0.3) annotation (Placement(transformation(extent={{-70,-40},{
30           -50,-20}}, rotation=0)));
31   Modelica.Blocks.Sources.SawTooth sawTooth(amplitude=2*Modelica.Constants.pi,
32      period=0.02) annotation (Placement(transformation(extent={{-40,-60},{-20,
33           -40}},rotation=0)));
34   Control.Assemblies.Inverter1phCurrentController control annotation (Placement(
35        transformation(
36        origin={10,-10},
37        extent={{-10,-10},{10,10}},
38        rotation=0)));
39   Modelica.Blocks.Sources.RealExpression iacSense(y=L.i)
40     annotation (Placement(transformation(extent={{-40,-20},{-20,0}})));
41   Modelica.Blocks.Sources.RealExpression vdcSense(y=DC.v)
42     annotation (Placement(transformation(extent={{-70,-80},{-50,-60}})));
43 equation
44   connect(DC.n, ground.p)
45     annotation (Line(points={{-20,40},{-20,37},{-20,34}}, color={0,0,255}));
46   connect(R.p, L.n)
47     annotation (Line(points={{70,40},{70,60}}, color={0,0,255}));
48   connect(HB.p1, DC.p) annotation (Line(points={{20,55},{0,55},{0,60},{-20,60}},
49        color={0,0,255}));
50   connect(HB.n1, DC.n) annotation (Line(points={{20,45},{0,45},{0,40},{-20,40}},
```

```
51        color={0,0,255}));
52    connect(HB.p2, L.p) annotation (Line(points={{40,55},{46,55},{46,80},{70,80}},
53        color={0,0,255}));
54    connect(HB.n2, R.n) annotation (Line(points={{40,45},{46,45},{46,20},{70,20}},
55        color={0,0,255}));
56    connect(sawTooth.y, control.theta) annotation (Line(points={{-19,-50},{-19,-50},
57          {6,-50},{6,-22}}, color={0,0,127}));
58    connect(control.d, HB.d)
59      annotation (Line(points={{21,-10},{30,-10},{30,38}}, color={0,0,127}));
60    connect(iqSetpoint.y, control.ids) annotation (Line(points={{-49,10},{-10,10},
61          {-10,-4},{-2,-4}}, color={0,0,127}));
62    connect(idSetpoint.y, control.iqs) annotation (Line(points={{-49,-30},{-10.5,
63          -30},{-10.5,-16},{-2,-16}}, color={0,0,127}));
64    connect(iacSense.y, control.i)
65      annotation (Line(points={{-19,-10},{-2,-10}}, color={0,0,127}));
66    connect(vdcSense.y, control.vdc) annotation (Line(points={{-49,-70},{-49,-70},
67          {14,-70},{14,-22}}, color={0,0,127}));
68    annotation (experiment(StopTime=0.6, __Dymola_NumberOfIntervals=3000),
69                  Documentation(info="<html>
70        <p>
71          This example explores a closed-loop inverter. No grid is
72          present, which simplifies things. But, since the
73          controller is implemented in the synchronous (dq)
74          reference frame, a synchronization source needs to
75          exist. This is implemented with the saw tooth generator,
76          which emulates the output of the PLL.
77        </p>
78
79        <p>
80          As can be seen in the following figure, one can now
81          comfortably specify the setpoint for the output current
82          of the inverter:
83        </p>
84
85
86        <div class=\"figure\">
87          <p><img src=\"modelica://PVSystems/Resources/Images/Inverter1phClosedResults.png\"
88                alt=\"Inverter1phClosedResults.png\" />
89          </p>
90        </div>
91
92        <p>
93          Having the posibility to separately control the current
94          in each dq axis enables one to control the power factor
95          (i.e. the phase lag between the voltage and the current)
96          as well as the amplitude of the current.
97        </p>
98
99        <p>
100         In this example, the equivalent synchronization signal
101         is plotted to see this phase shift as the setpoints
102         change. Notice how, when the q component of the current
103         is 0, the d component is equal to the peak current.</p>
104     </html>"),
105   __Dymola_experimentSetupOutput);
106 end Inverter1phClosed;
```

Examples/Application/Inverter1phClosedSynch.mo

```
1 within PVSystems.Examples.Application;
2 model Inverter1phClosedSynch
3   "Grid-tied 1-phase closed-loop inverter with constant DC source"
```

```
  extends Modelica.Icons.Example;
  Modelica.Electrical.Analog.Sources.ConstantVoltage DC(V=580) annotation (
     Placement(transformation(
       origin={-18,70},
       extent={{-10,-10},{10,10}},
       rotation=270)));
  Modelica.Electrical.Analog.Sources.SineVoltage AC(freqHz=50, V=480)
    annotation (Placement(transformation(
       origin={80,70},
       extent={{-10,-10},{10,10}},
       rotation=270)));
  Control.PLL pll annotation (Placement(transformation(
       extent={{10,-10},{-10,10}},
       rotation=180,
       origin={-40,-50})));
  PVSystems.Electrical.Assemblies.HBridge HB(d(start=0.5))
    annotation (Placement(transformation(extent={{2,60},{22,80}}, rotation=0)));
  Modelica.Electrical.Analog.Basic.Inductor L(L=1e-3) annotation (Placement(
       transformation(extent={{34,80},{54,100}}, rotation=0)));
  Modelica.Electrical.Analog.Basic.Resistor R(R=1e-2)
                                       annotation (Placement(
       transformation(extent={{60,80},{80,100}}, rotation=0)));
  Control.Assemblies.Inverter1phCurrentController control(d(start=0.5))
    annotation (Placement(transformation(
       origin={-10,10},
       extent={{-10,-10},{10,10}},
       rotation=0)));
  Modelica.Blocks.Sources.Constant idSetpoint(k=400)
    annotation (Placement(transformation(extent={{-90,30},{-70,50}})));
  Modelica.Blocks.Sources.Constant iqSetpoint(k=0)
    annotation (Placement(transformation(extent={{-90,-30},{-70,-10}})));
  Modelica.Electrical.Analog.Basic.Ground ground
    annotation (Placement(transformation(extent={{-28,40},{-8,60}})));
  Modelica.Blocks.Sources.RealExpression vacSense(y=AC.v)
    annotation (Placement(transformation(extent={{-90,-60},{-70,-40}})));
  Modelica.Blocks.Sources.RealExpression iacSense(y=AC.i)
    annotation (Placement(transformation(extent={{-90,0},{-70,20}})));
  Modelica.Blocks.Sources.RealExpression vdcSense(y=DC.v)
    annotation (Placement(transformation(extent={{-90,-90},{-70,-70}})));
equation
  connect(L.n, R.p)
    annotation (Line(points={{54,90},{60,90}}, color={0,0,255}));
  connect(HB.p2, L.p) annotation (Line(points={{22,75},{28,75},{28,90},{34,90}},
       color={0,0,255}));
  connect(R.n, AC.p)
    annotation (Line(points={{80,90},{80,80}}, color={0,0,255}));
  connect(DC.p, HB.p1) annotation (Line(points={{-18,80},{-2,80},{-2,75},{2,75}},
       color={0,0,255}));
  connect(DC.n, HB.n1) annotation (Line(points={{-18,60},{-2,60},{-2,65},{2,65}},
       color={0,0,255}));
  connect(iqSetpoint.y, control.iqs) annotation (Line(points={{-69,-20},{-40,-20},
        {-40,4},{-22,4}}, color={0,0,127}));
  connect(idSetpoint.y, control.ids) annotation (Line(points={{-69,40},{-40,40},
        {-40,16},{-22,16}}, color={0,0,127}));
  connect(DC.n, ground.p)
    annotation (Line(points={{-18,60},{-18,60}}, color={0,0,255}));
  connect(AC.n, HB.n2) annotation (Line(points={{80,60},{80,50},{28,50},{28,65},
        {22,65}}, color={0,0,255}));
  connect(control.d, HB.d)
    annotation (Line(points={{1,10},{12,10},{12,58}}, color={0,0,127}));
  connect(iacSense.y, control.i)
    annotation (Line(points={{-69,10},{-22,10}}, color={0,0,127}));
  connect(pll.theta, control.theta) annotation (Line(points={{-29,-50},{-29,-50},
        {-14,-50},{-14,-2}}, color={0,0,127}));
  connect(vacSense.y, pll.v)
    annotation (Line(points={{-69,-50},{-52,-50},{-52,-50}}, color={0,0,127}));
```

```
70  connect(vdcSense.y, control.vdc)
71    annotation (Line(points={{-69,-80},{-6,-80},{-6,-2}}, color={0,0,127}));
72  annotation (experiment(StopTime=0.3, __Dymola_NumberOfIntervals=3000),
73     __Dymola_experimentSetupOutput,
74     Documentation(info="<html>
75        <p>
76         This example includes a voltage source on the AC
77         side. This will add the synchronization challenge for
78         the controller: in order to provide adequate control of
79         the output current, the duty cycle needs to be carefully
80         in synch with the AC grid voltage.
81        </p>
82
83        <p>
84         Plotting the current through the load, the dq setpoints,
85         the grid voltage and the actual computed d value of the
86         current yields the following graph:
87        </p>
88
89
90        <div class=\"figure\">
91          <p><img src=\"modelica://PVSystems/Resources/Images/
                Inverter1phClosedSynchResults.png\"
92              alt=\"Inverter1phClosedSynchResults.png\" />
93          </p>
94        </div>
95
96        <p>
97         After an initial period where the signals are reaching
98         their steady-state values, the current successfully
99         reaches the setpoint value. Since the q setpoint is
100        equal to zero, the output current stays in phase with
101        the grid voltage and the d setpoint value equals the
102        peak current value.</p>
103      </html>"));
104 end Inverter1phClosedSynch;
```

Examples/Application/Inverter1phOpen.mo

```
1  within PVSystems.Examples.Application;
2  model Inverter1phOpen
3   "Stand-alone 1-phase open-loop inverter with constant DC source"
4   extends Modelica.Icons.Example;
5   Electrical.Assemblies.HBridgeSwitched
6                         HBsw annotation (Placement(
7       transformation(extent={{20,40},{40,60}}, rotation=0)));
8   Modelica.Electrical.Analog.Sources.ConstantVoltage DCsw(V=580) annotation (
9      Placement(transformation(
10       origin={-10,50},
11       extent={{-10,-10},{10,10}},
12       rotation=270)));
13  Modelica.Electrical.Analog.Basic.Ground gsw annotation (Placement(
14       transformation(extent={{-20,14},{0,34}}, rotation=0)));
15  Modelica.Electrical.Analog.Basic.Resistor Rsw(R=1e-2)
16                                        annotation (Placement(
17       transformation(
18       origin={90,30},
19       extent={{-10,-10},{10,10}},
20       rotation=270)));
21  Modelica.Electrical.Analog.Basic.Inductor Lsw(L=1e-3) annotation (Placement(
22       transformation(
23       origin={90,70},
```

```
      extent={{-10,-10},{10,10}},
      rotation=270)));
  Modelica.Blocks.Sources.Sine duty(
    offset=0.5,
    freqHz=50,
    amplitude=0.05)
            annotation (Placement(transformation(extent={{-90,-90},{-70,-70}},
        rotation=0)));
  PVSystems.Electrical.Assemblies.HBridge HBav annotation (Placement(
      transformation(extent={{20,-40},{40,-20}}, rotation=0)));
  Modelica.Electrical.Analog.Basic.Resistor Rav(R=1e-2)
                                        annotation (Placement(
      transformation(
      origin={90,-50},
      extent={{-10,-10},{10,10}},
      rotation=270)));
  Modelica.Electrical.Analog.Basic.Inductor Lav(L=1e-3) annotation (Placement(
      transformation(
      origin={90,-10},
      extent={{-10,-10},{10,10}},
      rotation=270)));
  Modelica.Electrical.Analog.Sources.ConstantVoltage DCav(V=580) annotation (
     Placement(transformation(
      origin={-10,-30},
      extent={{-10,-10},{10,10}},
      rotation=270)));
  Modelica.Electrical.Analog.Basic.Ground gav annotation (Placement(
      transformation(extent={{-20,-66},{0,-46}}, rotation=0)));
  Control.SwitchingPWM switchingPWM(fs=3125)
    annotation (Placement(transformation(extent={{-40,-10},{-20,10}})));
  Control.DeadTime deadTime annotation (Placement(transformation(
      extent={{-10,-10},{10,10}},
      rotation=90,
      origin={30,20})));
equation
  connect(DCsw.n, gsw.p)
    annotation (Line(points={{-10,40},{-10,34}}, color={0,0,255}));
  connect(HBsw.n1, DCsw.n) annotation (Line(points={{20,45},{10,45},{10,40},{-10,
        40}}, color={0,0,255}));
  connect(HBsw.p1, DCsw.p) annotation (Line(points={{20,55},{10,55},{10,60},{-10,
        60}}, color={0,0,255}));
  connect(HBsw.p2, Lsw.p) annotation (Line(points={{40,55},{60,55},{60,80},{90,
        80}}, color={0,0,255}));
  connect(HBsw.n2, Rsw.n) annotation (Line(points={{40,45},{60,45},{60,20},{90,
        20}}, color={0,0,255}));
  connect(Rsw.p, Lsw.n) annotation (Line(points={{90,40},{90,46},{90,50},{90,60}},
      color={0,0,255}));
  connect(Rav.p, Lav.n) annotation (Line(points={{90,-40},{90,-36},{90,-30},{90,
        -20}}, color={0,0,255}));
  connect(HBav.p2, Lav.p) annotation (Line(points={{40,-25},{60,-25},{60,0},{90,
        0}}, color={0,0,255}));
  connect(Rav.n, HBav.n2) annotation (Line(points={{90,-60},{60,-60},{60,-35},{
        40,-35}}, color={0,0,255}));
  connect(HBav.d, duty.y)
    annotation (Line(points={{30,-42},{30,-80},{-69,-80}}, color={0,0,127}));
  connect(DCav.n, gav.p)
    annotation (Line(points={{-10,-40},{-10,-43},{-10,-46}}, color={0,0,255}));
  connect(DCav.p, HBav.p1) annotation (Line(points={{-10,-20},{10,-20},{10,-25},
        {20,-25}}, color={0,0,255}));
  connect(DCav.n, HBav.n1) annotation (Line(points={{-10,-40},{10,-40},{10,-35},
        {20,-35}}, color={0,0,255}));
  connect(deadTime.c1, HBsw.c1)
    annotation (Line(points={{26,31},{26,35.5},{26,40}}, color={255,0,255}));
  connect(deadTime.c2, HBsw.c2)
    annotation (Line(points={{34,31},{34,40}}, color={255,0,255}));
  connect(switchingPWM.c1, deadTime.c)
```

```
  annotation (Line(points={{-19,0},{30,0},{30,8}}, color={255,0,255}));
connect(switchingPWM.vc, duty.y) annotation (Line(points={{-42,0},{-50,0},{-50,
      -80},{-69,-80}}, color={0,0,127}));
annotation (
  Diagram(graphics={Text(
       extent={{10,74},{48,62}},
       lineColor={0,0,255},
       textString="Switched model"),Text(
       extent={{12,-6},{50,-18}},
       lineColor={0,0,255},
       textString="Modifiable model")}),
  experiment(StopTime=0.5, __Dymola_NumberOfIntervals=5000),
  Documentation(info="<html>
     <p>
      This example presents two implementations of an open
      loop 1-phase inverter. The function of the inverter is
      to convert DC voltage and current into AC voltage and
      current. To keep things simple, a constant DC source is
      included on the DC side and an RL load is included on
      the AC side. Typically, inverters are placed inside a
      more complicated setup, which might require MPPT
      (Maximum Power Point Tracking) on the DC side when
      connected to a PV array and AC synchronization when
      connected to a grid on the AC side instead of just a
      simple passive load.
     </p>

     <p>
      Nevertheless, the example still showcases an interesting
      application. Upon running the simulation with the
      provided values, plotting the resistor voltages yields
      the following figure:
     </p>


     <div class=\"figure\">
      <p><img src=\"modelica://PVSystems/Resources/Images/Inverter1phOpenResults.png\"
            alt=\"Inverter1phOpenResults.png\" />
      </p>
     </div>

     <p>
      The AC is achieved with the inverter topology (called an
      H-bridge) as well as with the duty cycle sinusoidal
      modulation. Have a look at the duty cycle driving the
      SwitchingPWM block and compare it with the voltage drop
      in the resistor.
     </p>

     <p>
      Compare it with the voltage drop in the inductor. The
      voltage coming out of the inverter is actually a square
      wave and the inductor is providing some crude (but
      enough for some applications) filtering. Play around
      with the value of the inductor to see how it provides a
      better or worse filtering performance (decreasing or
      increasing the voltage and current ripple in the
      resistor, which in this example is assumed to be the
      load being fed). Since this is an open loop
      configuration, it will also change the peak value of the
      voltage drop in the resistor, as well as its phase.
     </p>

     <p>
      Importantly, see how the the average model provides a
      very good approximation for low frequencies. This kind
```

```
        of model won't be useful to study ripples and to
        evaluate the performance of different PWM modulations
        (sinusoidal modulation is being used in this example) or
        of different filter configurations, since those are
        concerned with the high frequencies in the system. On
        the other hand, the average models will be very useful
        to study controllers and to perform longer simulations
        since the simulation step doesn't need to be so small as
        to accurately represent the switching dynamics.</p>
    </html>"));
end Inverter1phOpen;
```

Examples/Application/Inverter1phOpenSynch.mo

```
within PVSystems.Examples.Application;
model Inverter1phOpenSynch
  "Grid-tied 1-phase open-loop inverter with constant DC source"
  extends Modelica.Icons.Example;
  Electrical.Assemblies.HBridgeSwitched
                        HBsw
    annotation (Placement(transformation(extent={{0,40},{20,60}}, rotation=0)));
  Modelica.Electrical.Analog.Sources.ConstantVoltage DCsw(V=580) annotation (
     Placement(transformation(
       origin={-30,50},
       extent={{-10,-10},{10,10}},
       rotation=270)));
  Modelica.Electrical.Analog.Sources.SineVoltage ACsw(freqHz=50, V=480)
    annotation (Placement(transformation(
       origin={50,30},
       extent={{-10,-10},{10,10}},
       rotation=270)));
  Modelica.Electrical.Analog.Basic.Inductor Lsw(L=1e-3) annotation (Placement(
       transformation(
       origin={50,70},
       extent={{-10,-10},{10,10}},
       rotation=270)));
  Control.PLL pLL annotation (Placement(transformation(
       origin={-80,-56},
       extent={{10,-10},{-10,10}},
       rotation=180)));
  Modelica.Blocks.Math.Cos sin annotation (Placement(transformation(
       origin={-46,-56},
       extent={{10,-10},{-10,10}},
       rotation=180)));
  Modelica.Blocks.Math.Add add(k2=1, k1=580/580/2) annotation (Placement(
       transformation(
       origin={-10,-50},
       extent={{10,-10},{-10,10}},
       rotation=180)));
  Modelica.Blocks.Sources.Constant const(k=0.5) annotation (Placement(
       transformation(
       origin={-80,-24},
       extent={{10,-10},{-10,10}},
       rotation=180)));
  Modelica.Electrical.Analog.Basic.Resistor Rsw(R=1e-2)
                                        annotation (Placement(
       transformation(
       origin={50,50},
       extent={{-10,-10},{10,10}},
       rotation=270)));
  PVSystems.Electrical.Assemblies.HBridge HBav annotation (Placement(
       transformation(extent={{70,-20},{90,0}}, rotation=0)));
```

```
49   Modelica.Electrical.Analog.Basic.Inductor Lav(L=1e-3) annotation (Placement(
50       transformation(
51       origin={120,10},
52       extent={{-10,-10},{10,10}},
53       rotation=270)));
54   Modelica.Electrical.Analog.Basic.Resistor Rav(R=1e-2)
55                                           annotation (Placement(
56       transformation(
57       origin={120,-10},
58       extent={{-10,-10},{10,10}},
59       rotation=270)));
60   Modelica.Electrical.Analog.Basic.Ground gsw annotation (Placement(
61       transformation(extent={{-40,20},{-20,40}},
62                                       rotation=0)));
63   Modelica.Blocks.Sources.RealExpression VacSense(y=ACsw.v)
64     annotation (Placement(transformation(extent={{-120,-66},{-100,-46}})));
65   Modelica.Electrical.Analog.Sources.SineVoltage ACav(freqHz=50, V=480)
66     annotation (Placement(transformation(
67       origin={120,-30},
68       extent={{-10,-10},{10,10}},
69       rotation=270)));
70   Modelica.Electrical.Analog.Sources.ConstantVoltage DCav(V=580) annotation (
71      Placement(transformation(
72       origin={40,-10},
73       extent={{-10,-10},{10,10}},
74       rotation=270)));
75   Modelica.Electrical.Analog.Basic.Ground gav annotation (Placement(
76       transformation(extent={{30,-40},{50,-20}}, rotation=0)));
77   Control.SwitchingPWM switchingPWM(fs=3125) annotation (Placement(
78       transformation(
79       extent={{-10,-10},{10,10}},
80       rotation=90,
81       origin={10,-10})));
82   Control.DeadTime deadTime annotation (Placement(transformation(
83       extent={{-10,-10},{10,10}},
84       rotation=90,
85       origin={10,20})));
86 equation
87   connect(pLL.theta, sin.u)
88     annotation (Line(points={{-69,-56},{-58,-56}}, color={0,0,127}));
89   connect(const.y, add.u2) annotation (Line(points={{-69,-24},{-30,-24},{-30,
90         -44},{-22,-44}},
91                 color={0,0,127}));
92   connect(sin.y, add.u1)
93     annotation (Line(points={{-35,-56},{-22,-56}}, color={0,0,127}));
94   connect(VacSense.y, pLL.v) annotation (Line(points={{-99,-56},{-99,-56},{-92,
95         -56}}, color={0,0,127}));
96   connect(ACav.p, Rav.n)
97     annotation (Line(points={{120,-20},{120,-20}}, color={0,0,255}));
98   connect(Rav.p, Lav.n)
99     annotation (Line(points={{120,0},{120,0}}, color={0,0,255}));
100  connect(HBav.p2, Lav.p) annotation (Line(points={{90,-5},{100,-5},{100,20},{
101        120,20}}, color={0,0,255}));
102  connect(HBav.n2, ACav.n) annotation (Line(points={{90,-15},{100,-15},{100,-40},
103        {120,-40}}, color={0,0,255}));
104  connect(ACsw.p, Rsw.n)
105    annotation (Line(points={{50,40},{50,40}}, color={0,0,255}));
106  connect(Rsw.p, Lsw.n)
107    annotation (Line(points={{50,60},{50,60}}, color={0,0,255}));
108  connect(HBsw.n2, ACsw.n) annotation (Line(points={{20,45},{30,45},{30,20},{50,
109        20}}, color={0,0,255}));
110  connect(HBsw.p2, Lsw.p) annotation (Line(points={{20,55},{30,55},{30,80},{50,
111        80}}, color={0,0,255}));
112  connect(DCav.n, gav.p)
113    annotation (Line(points={{40,-20},{40,-20}}, color={0,0,255}));
114  connect(DCsw.n, gsw.p)
```

```
  annotation (Line(points={{-30,40},{-30,40}}, color={0,0,255}));
connect(DCsw.p, HBsw.p1) annotation (Line(points={{-30,60},{-14,60},{-14,55},
      {0,55}}, color={0,0,255}));
connect(DCsw.n, HBsw.n1) annotation (Line(points={{-30,40},{-14,40},{-14,45},
      {0,45}}, color={0,0,255}));
connect(DCav.n, HBav.n1) annotation (Line(points={{40,-20},{56,-20},{56,-15},
      {70,-15}}, color={0,0,255}));
connect(DCav.p, HBav.p1)
  annotation (Line(points={{40,0},{56,0},{56,-5},{70,-5}}, color={0,0,255}));
connect(add.y, HBav.d)
  annotation (Line(points={{1,-50},{80,-50},{80,-22}}, color={0,0,127}));
connect(deadTime.c1, HBsw.c1)
  annotation (Line(points={{6,31},{6,40}}, color={255,0,255}));
connect(deadTime.c2, HBsw.c2)
  annotation (Line(points={{14,31},{14,35.5},{14,40}}, color={255,0,255}));
connect(switchingPWM.c1, deadTime.c)
  annotation (Line(points={{10,1},{10,4.5},{10,8}}, color={255,0,255}));
connect(switchingPWM.vc, HBav.d) annotation (Line(points={{10,-22},{10,-50},{
      80,-50},{80,-22}}, color={0,0,127}));
annotation (
  experiment(StopTime=0.5, __Dymola_NumberOfIntervals=5000),
  Documentation(info="<html>
    <p>
      This example goes a step further
      than <a href=\"Modelica://PVSystems.Examples.Application.Inverter1phOpen\">
          Inverter1phOpen</a>
      and includes grid synchronization. Typically this is the condition
      for inverters in real-life situations. Both switched and averaged
      implementations are presented for comparison purposes and it can be
      seen that they both provide very similar results (excluding the fact
      that high frequencies are left out in the averaged model).
    </p>


    <div class=\"figure\">
      <p><img src=\"modelica://PVSystems/Resources/Images/Inverter1phOpenSynchResults.png
          \"
            alt=\"Inverter1phOpenSynch_Plot.png\" />
      </p>
    </div>

    <p>
      Since this is still open-loop and there's no in-quadrature
      separation, the value of the current can't comfortably be specified
      to be of a certain value. Since the RL load has almost equal real
      and imaginary parts, the current that is drawn from the inverter has
      a power factor different than one.
    </p>

    <p>
      A key value to pay attention to in this example is the gain that is
      placed in the <i>Add</i> block.
    </p>


    <div class=\"figure\">
      <p><img src=\"modelica://PVSystems/Resources/Images/Inverter1phOpenSynchDialog.png
          \"
            alt=\"Inverter1phOpenSynchDialog.png\" />
      </p>
    </div>

    <p>
      It's initially set at 0.5. The value is expressed as 580/580/2 to
      highlight the fact that this gain should be normalized to the DC
      voltage value. Above that, over-modulation will occur and the output
```

```
178       current of the inverter will become quite ugly. Play around with
179       this value (using values between 0 and 0.5) to see how the output
180       current of the inverter changes.
181     </p>
182     </html>"),
183    Diagram(coordinateSystem(extent={{-140,-100},{140,100}}, initialScale=0.1),
184       graphics={ Text(
185         extent={{56,14},{94,2}},
186         lineColor={0,0,255},
187         textString="Modifiable model"),
188                  Text(
189         extent={{-12,74},{26,62}},
190         lineColor={0,0,255},
191         textString="Switched model")}),
192    Icon(coordinateSystem(initialScale=0.1)),
193    __Dymola_Commands(file="Resources/Scripts/Dymola/
           Inverter1phOpenSynch_RunPlotAndSave.mos"
194       "RunPlotAndSave"));
195 end Inverter1phOpenSynch;
```

Examples/Application/package.mo

```
1  within PVSystems.Examples;
2  package Application "More complete application examples"
3  extends Modelica.Icons.ExamplesPackage;
4
5
6
7
8
9
10 end Application;
```

Examples/Application/package.order

```
1 BuckOpen
2 Inverter1phOpen
3 Inverter1phOpenSynch
4 Inverter1phClosed
5 Inverter1phClosedSynch
6 PVInverter1ph
7 PVInverter1phSynch
8 USBBatteryConverter
```

Examples/Application/PVInverter1ph.mo

```
1 within PVSystems.Examples.Application;
2 model PVInverter1ph "Stand-alone 1-phase closed-loop inverter with PV source"
3   extends Modelica.Icons.Example;
4   Electrical.PVArray PV(v(start=450)) annotation (Placement(transformation(
5       origin={-40,60},
6       extent={{-10,-10},{10,10}},
7       rotation=270)));
```

```
  Modelica.Blocks.Sources.Constant Gn(k=1000) annotation (Placement(
      transformation(extent={{-80,70},{-60,90}}, rotation=0)));
  Modelica.Blocks.Sources.Constant Tn(k=298.15) annotation (Placement(
      transformation(extent={{-80,30},{-60,50}}, rotation=0)));
  PVSystems.Electrical.Assemblies.HBridge Inverter annotation (Placement(
      transformation(extent={{40,50},{60,70}}, rotation=0)));
  Modelica.Electrical.Analog.Basic.Inductor L(L=1e-3) annotation (Placement(
      transformation(
      origin={90,74},
      extent={{-10,-10},{10,10}},
      rotation=270)));
  Modelica.Electrical.Analog.Basic.Resistor R(R=1e-2)
                                       annotation (Placement(
      transformation(
      origin={90,48},
      extent={{-10,-10},{10,10}},
      rotation=270)));
  Modelica.Electrical.Analog.Basic.Capacitor Cdc( C=5e-1, v(start=
        10))
    annotation (Placement(transformation(
      origin={20,60},
      extent={{-10,-10},{10,10}},
      rotation=270)));
  Modelica.Electrical.Analog.Basic.Resistor Rdc(R=1e-3, v(start=30))
    annotation (Placement(transformation(extent={{-20,70},{0,90}}, rotation=0)));
  Modelica.Electrical.Analog.Basic.Ground ground annotation (Placement(
      transformation(extent={{-20,20},{0,40}}, rotation=0)));
  Modelica.Blocks.Sources.Cosine vacEmulation(freqHz=50) annotation (Placement(
      transformation(extent={{-40,-70},{-20,-50}}, rotation=0)));
  Control.Assemblies.Inverter1phCompleteController controller(
    fline=50,
    ik=0.1,
    iT=0.01,
    vk=10,
    vT=0.5,
    iqMax=10,
    vdcMax=71,
    idMax=10)
          annotation (Placement(transformation(
      origin={30,-30},
      extent={{-10,-10},{10,10}},
      rotation=0)));
  Modelica.Blocks.Sources.RealExpression iacSense(y=L.i)
    annotation (Placement(transformation(extent={{-40,-44},{-20,-24}})));
  Modelica.Blocks.Sources.RealExpression idcSense(y=-PV.i)
    annotation (Placement(transformation(extent={{-40,-20},{-20,0}})));
  Modelica.Blocks.Sources.RealExpression vdcSense(y=PV.v)
    annotation (Placement(transformation(extent={{-40,0},{-20,20}})));
  Modelica.Blocks.Sources.RealExpression DCPower(y=-PV.i*PV.v)
    annotation (Placement(transformation(extent={{40,-72},{60,-52}})));
  Modelica.Blocks.Sources.RealExpression ACPower(y=R.i*R.v)
    annotation (Placement(transformation(extent={{40,-92},{60,-72}})));
  Modelica.Blocks.Math.Mean meanACPower(f=50)
    annotation (Placement(transformation(extent={{70,-92},{90,-72}})));
equation
  connect(Gn.y, PV.G) annotation (Line(points={{-59,80},{-54,80},{-54,63},{-45.5,
        63}}, color={0,0,127}));
  connect(Tn.y, PV.T) annotation (Line(points={{-59,40},{-54,40},{-54,57},{-45.5,
        57}}, color={0,0,127}));
  connect(Cdc.p, Inverter.p1) annotation (Line(points={{20,70},{34,70},{34,65},
        {40,65}}, color={0,0,255}));
  connect(L.n, R.p)
    annotation (Line(points={{90,64},{90,58}}, color={0,0,255}));
  connect(Cdc.n, Inverter.n1) annotation (Line(points={{20,50},{34,50},{34,55},
        {40,55}}, color={0,0,255}));
  connect(PV.p, Rdc.p) annotation (Line(points={{-40,70},{-40,70},{-40,80},{-20,
```

```
74        80}}, color={0,0,255}));
75  connect(Cdc.n, ground.p) annotation (Line(points={{20,50},{20,48},{20,40},{-10,
76        40}}, color={0,0,255}));
77  connect(PV.n, ground.p)
78    annotation (Line(points={{-40,50},{-40,40},{-10,40}}, color={0,0,255}));
79  connect(Rdc.n, Cdc.p) annotation (Line(points={{0,80},{10,80},{20,80},{20,70}},
80      color={0,0,255}));
81  connect(Inverter.p2, L.p) annotation (Line(points={{60,65},{70,65},{70,90},{
82        90,90},{90,84}}, color={0,0,255}));
83  connect(Inverter.n2, R.n) annotation (Line(points={{60,55},{70,55},{70,30},{
84        90,30},{90,38}}, color={0,0,255}));
85  connect(idcSense.y, controller.idc) annotation (Line(points={{-19,-10},{0,-10},
86        {0,-26},{18,-26}}, color={0,0,127}));
87  connect(vdcSense.y, controller.vdc) annotation (Line(points={{-19,10},{10,10},
88        {10,-22},{18,-22}}, color={0,0,127}));
89  connect(vacEmulation.y, controller.vac) annotation (Line(points={{-19,-60},{0,
90        -60},{0,-38},{18,-38}}, color={0,0,127}));
91  connect(iacSense.y, controller.iac)
92    annotation (Line(points={{-19,-34},{-0.5,-34},{18,-34}}, color={0,0,127}));
93  connect(ACPower.y, meanACPower.u)
94    annotation (Line(points={{61,-82},{64.5,-82},{68,-82}}, color={0,0,127}));
95  connect(controller.d, Inverter.d)
96    annotation (Line(points={{41,-30},{50,-30},{50,48}}, color={0,0,127}));
97  annotation (Diagram(coordinateSystem(initialScale=0.1)), experiment(StopTime=
98        3, Interval=0.001),
99        Documentation(info="<html>
100          <p>
101            This example adds a PV array to the DC side. To start as
102            simple as possible, the AC side is just a passive RL
103            load. A general controller for this kind of setup is
104            devised and packaged
105            as <a href=\"Modelica://
                   PVSystems.Control.Assemblies.Inverter1phCompleteController\">
                   Inverter1phCompleteController</a>. This
106            block accepts no input because it's assumed that the
107            controller will try to extract the maximum active power
108            from the PV array. Internally, the q current setpoint is
109            set to zero.
110          </p>
111
112          <p>
113            Plotting the DC bus voltage and the output current
114            confirms shows that this is in fact how the controller
115            is behaving:
116          </p>
117
118
119          <div class=\"figure\">
120            <p><img src=\"modelica://PVSystems/Resources/Images/PVInverter1phResultsA.png
                     \"
121                  alt=\"PVInverter1phResultsA.png\" />
122            </p>
123          </div>
124
125          <p>
126            The maximum power point is achieved by indirectly
127            balancing the difference between the power delivered by
128            the PV array and the power dumped on to the grid. As the
129            maximum power point is being reached, the difference
130            tends to zero:
131          </p>
132
133
134          <div class=\"figure\">
135            <p><img src=\"modelica://PVSystems/Resources/Images/PVInverter1phResultsB.png
                     \"
```

```
                    alt=\"PVInverter1phResultsB.png\" /></p>
          </div>
        </html>"));
end PVInverter1ph;
```

Examples/Application/PVInverter1phSynch.mo

```
within PVSystems.Examples.Application;
model PVInverter1phSynch
  "Grid-tied 1-phase closed-loop inverter with PV source"
  extends Modelica.Icons.Example;
  Electrical.PVArray PV(v(start=450)) annotation (Placement(transformation(
      origin={-40,70},
      extent={{-10,-10},{10,10}},
      rotation=270)));
  Modelica.Blocks.Sources.Constant Gn(k=1000) annotation (Placement(
      transformation(extent={{-80,70},{-60,90}}, rotation=0)));
  Modelica.Blocks.Sources.Constant Tn(k=298.15) annotation (Placement(
      transformation(extent={{-80,30},{-60,50}}, rotation=0)));
  PVSystems.Electrical.Assemblies.HBridge Inverter annotation (
     Placement(transformation(extent={{40,60},{60,80}}, rotation=0)));
  Modelica.Electrical.Analog.Sources.SineVoltage AC(freqHz=50, V=15)
    annotation (Placement(transformation(
      origin={86,10},
      extent={{-10,-10},{10,10}},
      rotation=270)));
  Modelica.Electrical.Analog.Basic.Inductor L(L=1e-3) annotation (Placement(
      transformation(
      origin={86,70},
      extent={{-10,-10},{10,10}},
      rotation=270)));
  Modelica.Electrical.Analog.Basic.Resistor R(R=1e-2) annotation (Placement(
      transformation(
      origin={86,40},
      extent={{-10,-10},{10,10}},
      rotation=270)));
  Modelica.Electrical.Analog.Basic.Capacitor Cdc(v(start=32.8), C=0.5)
    annotation (Placement(transformation(
      origin={24,70},
      extent={{-10,-10},{10,10}},
      rotation=270)));
  Control.Assemblies.Inverter1phCompleteController Controller(
    ik=0.1,
    iT=0.01,
    fline=50,
    vk=10,
    vT=0.5,
    idMax=20,
    iqMax=20,
    vdcMax=50)
          annotation (Placement(transformation(
      origin={30,-10},
      extent={{-10,-10},{10,10}},
      rotation=0)));
  Modelica.Electrical.Analog.Basic.Resistor Rdc(R=1e-3, v(start=30))
    annotation (Placement(transformation(extent={{-20,70},{0,90}}, rotation=0)));
  Modelica.Electrical.Analog.Basic.Ground ground annotation (Placement(
      transformation(extent={{-20,40},{0,60}}, rotation=0)));
  Modelica.Blocks.Sources.RealExpression vdcSense(y=PV.v)
    annotation (Placement(transformation(extent={{-40,10},{-20,30}})));
  Modelica.Blocks.Sources.RealExpression idcSense(y=-PV.i)
    annotation (Placement(transformation(extent={{-40,-16},{-20,4}})));
```

```
56   Modelica.Blocks.Sources.RealExpression iacSense(y=AC.i)
57     annotation (Placement(transformation(extent={{-40,-40},{-20,-20}})));
58   Modelica.Blocks.Sources.RealExpression vacSense(y=AC.v)
59     annotation (Placement(transformation(extent={{-40,-64},{-20,-44}})));
60   Modelica.Blocks.Sources.RealExpression DCPower(y=-PV.i*PV.v)
61     annotation (Placement(transformation(extent={{40,-72},{60,-52}})));
62   Modelica.Blocks.Sources.RealExpression ACPower(y=AC.v*AC.i)
63     annotation (Placement(transformation(extent={{40,-92},{60,-72}})));
64   Modelica.Blocks.Math.Mean meanACPower(f=50)
65     annotation (Placement(transformation(extent={{70,-92},{90,-72}})));
66 equation
67   connect(Gn.y, PV.G) annotation (Line(points={{-59,80},{-52,80},{-52,73},{-45.5,
68         73}}, color={0,0,127}));
69   connect(Tn.y, PV.T) annotation (Line(points={{-59,40},{-52,40},{-52,67},{-45.5,
70         67}}, color={0,0,127}));
71   connect(Cdc.p, Inverter.p1) annotation (Line(points={{24,80},{34,80},{34,75},
72         {40,75}}, color={0,0,255}));
73   connect(R.n, AC.p)
74     annotation (Line(points={{86,30},{86,20}}, color={0,0,255}));
75   connect(L.n, R.p)
76     annotation (Line(points={{86,60},{86,56},{86,50}}, color={0,0,255}));
77   connect(Cdc.n, Inverter.n1) annotation (Line(points={{24,60},{34,60},{34,65},
78         {40,65}}, color={0,0,255}));
79   connect(Inverter.p2, L.p) annotation (Line(points={{60,75},{70,75},{70,80},{
80         86,80}}, color={0,0,255}));
81   connect(Inverter.n2, AC.n)
82     annotation (Line(points={{60,65},{70,65},{70,0},{86,0}}, color={0,0,255}));
83   connect(Rdc.n, Cdc.p)
84     annotation (Line(points={{0,80},{24,80}}, color={0,0,255}));
85   connect(PV.p, Rdc.p)
86     annotation (Line(points={{-40,80},{-30,80},{-20,80}}, color={0,0,255}));
87   connect(PV.n, Cdc.n)
88     annotation (Line(points={{-40,60},{24,60}}, color={0,0,255}));
89   connect(PV.n, ground.p)
90     annotation (Line(points={{-40,60},{-10,60}}, color={0,0,255}));
91   connect(Controller.d, Inverter.d)
92     annotation (Line(points={{41,-10},{50,-10},{50,58}}, color={0,0,127}));
93   connect(vdcSense.y, Controller.vdc) annotation (Line(points={{-19,20},{0,20},
94         {0,-2},{18,-2}},color={0,0,127}));
95   connect(idcSense.y, Controller.idc)
96     annotation (Line(points={{-19,-6},{-10,-6},{18,-6}}, color={0,0,127}));
97   connect(iacSense.y, Controller.iac) annotation (Line(points={{-19,-30},{-10,-30},
98         {-10,-14},{18,-14}}, color={0,0,127}));
99   connect(vacSense.y, Controller.vac) annotation (Line(points={{-19,-54},{0,-54},
100        {0,-18},{18,-18}}, color={0,0,127}));
101  connect(ACPower.y, meanACPower.u)
102    annotation (Line(points={{61,-82},{64.5,-82},{68,-82}}, color={0,0,127}));
103  annotation (Icon(graphics), experiment(StopTime=28, Interval=0.001),
104    __Dymola_experimentSetupOutput,
105    Documentation(info="<html>
106       <p>
107         This example represents a simple yet complete grid-tied
108         PV inverter system. A long simulation is performed so as
109         to visualize the time evolution of the MPPT control,
110         which is necessarily much slower than the output current
111         control. This long simulation time is manageable because
112         an averaged switch model is being used, which means that
113         the simulation can have longer time steps.
114       </p>
115
116       <p>
117         This evolution can be observed by plotting the DC bus
118         voltage as well as the input and output power to the
119         inverter:
120       </p>
121
```

```
122
123       <div class=\"figure\">
124         <p><img src=\"modelica://PVSystems/Resources/Images/PVInverter1phSynchResultsA.png
              \"
125               alt=\"PVInverter1phSynchResultsA.png\" />
126         </p>
127       </div>
128
129       <p>
130         As expected, the power factor of the output power is 1
131         (all active power), having the output current in synch
132         with the grid voltage:
133       </p>
134
135
136       <div class=\"figure\">
137         <p><img src=\"modelica://PVSystems/Resources/Images/PVInverter1phSynchResultsB.png
              \"
138               alt=\"PVInverter1phSynchResultsB.png\" /></p>
139       </div>
140     </html>"));
141 end PVInverter1phSynch;
```

Examples/Application/USBBatteryConverter.mo

```
1  within PVSystems.Examples.Application;
2  model USBBatteryConverter "Bidirectional converter for USB battery interface"
3    extends Modelica.Icons.Example;
4    Electrical.Assemblies.CPMBidirectionalBuckBoost conv(
5      Cin=10e-6,
6      Cout=88e-6,
7      L=10e-6,
8      Rf=1,
9      fs=200e3,
10     RL=8e-3,
11     Va_buck=0.5,
12     Va_boost=1,
13     vCin_ini=12.6,
14     vCout_ini=5,
15     iL_ini=2) annotation (Placement(transformation(extent={{4,60},{24,80}})));
16   Modelica.Blocks.Sources.RealExpression boostVs(y=20)
17     annotation (Placement(transformation(extent={{-70,-10},{-50,10}})));
18   Modelica.Blocks.Sources.RealExpression buckVs(y=5)
19     annotation (Placement(transformation(extent={{-70,-50},{-50,-30}})));
20   Modelica.Electrical.Analog.Basic.Ground ground
21     annotation (Placement(transformation(extent={{40,40},{60,60}})));
22   Modelica.Electrical.Analog.Basic.Resistor Rbatt(R=50e-3)
23     annotation (Placement(transformation(extent={{-40,70},{-20,90}})));
24   Modelica.Blocks.Continuous.LimPID buckPI(
25     k=10,
26     controllerType=Modelica.Blocks.Types.SimpleController.PI,
27     Ti=1,
28     yMin=0,
29     yMax=8) annotation (Placement(transformation(extent={{-30,-30},{-10,-50}})));
30   Modelica.Blocks.Sources.RealExpression voutSense(y=conv.v2)
31     annotation (Placement(transformation(extent={{-70,-30},{-50,-10}})));
32   Modelica.Blocks.Continuous.LimPID boostPI(
33     k=10,
34     controllerType=Modelica.Blocks.Types.SimpleController.PI,
35     Ti=1,
36     yMin=0,
37     yMax=8) annotation (Placement(transformation(extent={{-30,-10},{-10,10}})));
```

```
38   Modelica.Blocks.Logical.Switch modeSelector annotation (Placement(
39       transformation(
40       extent={{-10,-10},{10,10}},
41       rotation=90,
42       origin={10,30})));
43   Modelica.Electrical.Analog.Basic.VariableResistor Rload annotation (Placement(
44       transformation(
45       extent={{-10,-10},{10,10}},
46       rotation=270,
47       origin={50,70})));
48   Modelica.Electrical.Analog.Sources.SignalVoltage Vbatt annotation (Placement(
49       transformation(
50       extent={{-10,10},{10,-10}},
51       rotation=270,
52       origin={-50,70})));
53   Modelica.Blocks.Sources.Ramp VbattSignal(
54     duration=0.1,
55     startTime=10,
56     offset=12.6,
57     height=9.6 - 12.6)
58     annotation (Placement(transformation(extent={{-90,60},{-70,80}})));
59   Modelica.Blocks.Sources.Ramp RloadSignal(
60     duration=0.1,
61     startTime=10,
62     offset=2.5,
63     height=6.67 - 2.5)
64     annotation (Placement(transformation(extent={{90,60},{70,80}})));
65   Modelica.Blocks.Sources.BooleanExpression modeCommand(y=time > 10)
66     annotation (Placement(transformation(extent={{-70,-80},{-50,-60}})));
67 equation
68   connect(Rbatt.n, conv.p1) annotation (Line(points={{-20,80},{-6,80},{-6,75},{
69         4,75}}, color={0,0,255}));
70   connect(buckVs.y, buckPI.u_s)
71     annotation (Line(points={{-49,-40},{-32,-40}}, color={0,0,127}));
72   connect(voutSense.y, buckPI.u_m)
73     annotation (Line(points={{-49,-20},{-20,-20},{-20,-28}}, color={0,0,127}));
74   connect(boostVs.y, boostPI.u_s)
75     annotation (Line(points={{-49,0},{-32,0}}, color={0,0,127}));
76   connect(conv.n2, ground.p) annotation (Line(points={{24,65},{34,65},{34,60},{
77         50,60}}, color={0,0,255}));
78   connect(voutSense.y, boostPI.u_m) annotation (Line(points={{-49,-20},{-34,-20},
79         {-20,-20},{-20,-12}},color={0,0,127}));
80   connect(modeSelector.y, conv.vc)
81     annotation (Line(points={{10,41},{10,41},{10,58}}, color={0,0,127}));
82   connect(boostPI.y, modeSelector.u1)
83     annotation (Line(points={{-9,0},{2,0},{2,18}}, color={0,0,127}));
84   connect(buckPI.y, modeSelector.u3)
85     annotation (Line(points={{-9,-40},{18,-40},{18,18}}, color={0,0,127}));
86   connect(Vbatt.p, Rbatt.p)
87     annotation (Line(points={{-50,80},{-46,80},{-40,80}}, color={0,0,255}));
88   connect(Vbatt.n, conv.n1) annotation (Line(points={{-50,60},{-50,60},{-6,60},
89         {-6,65},{4,65}}, color={0,0,255}));
90   connect(ground.p, Rload.n)
91     annotation (Line(points={{50,60},{50,60}}, color={0,0,255}));
92   connect(Rload.p, conv.p2) annotation (Line(points={{50,80},{34,80},{34,75},{
93         24,75}}, color={0,0,255}));
94   connect(VbattSignal.y, Vbatt.v)
95     annotation (Line(points={{-69,70},{-57,70}}, color={0,0,127}));
96   connect(RloadSignal.y, Rload.R)
97     annotation (Line(points={{69,70},{61,70}}, color={0,0,127}));
98   connect(modeCommand.y, modeSelector.u2) annotation (Line(points={{-49,-70},{10,
99         -70},{10,18}}, color={255,0,255}));
100  connect(modeCommand.y, conv.mode) annotation (Line(points={{-49,-70},{30,-70},
101        {30,50},{18,50},{18,58}}, color={255,0,255}));
102  annotation (experiment(StopTime=20, Interval=0.001),
103    __Dymola_experimentSetupOutput,
```

```
104     Documentation(info="<html>
105        <p>
106         A battery, simulated with a controlled voltage source in
107         series with a small resistance, is interfaced with a USB
108         device, simulated with a resistive load. The converter
109         is a component included in
110         the <a href=\"Modelica://PVSystems.Electrical.Assemblies\">Electrical.Assemblies
                </a>
111         package.
112        </p>
113
114        <p>
115         This example is borrowed
116         from <a href=\"modelica://PVSystems.UsersGuide.References.EMA16\">EMA16</a>. The
117         application is not that related with photovoltaics, but
118         provides a good showcase of the power electronics models
119         in this library. The converter is specified to have
120         three operating modes:
121        </p>
122
123        <ul class=\"org-ul\">
124         <li>Battery voltage 12.6V, USB voltage 5+/-0.1V at 2A,
125           converter supplies bus.
126         </li>
127         <li>Battery voltage 9.6V, USB voltage 20+/-0.1V at 3A,
128           converter supplies bus.
129         </li>
130         <li>Battery voltage 11.1V, USB voltage 20V, bus supplies
131           60W to charge battery.
132         </li>
133        </ul>
134
135        <p>
136         An efficient solution to these step-down and
137         bidirectional step-up requirements is a non-inverting
138         buck-boost converter with bi-directional switches
139         operated in a buck/boost modal fashion (i.e. the boost
140         switches are disabled while in buck mode and vice
141         versa). A possible solution to these requirements using
142         this topology is expressed through the parametrization
143         of <a href=\"modelica://
                PVSystems.Electrical.Assemblies.CPMBidirectionalBuckBoost\">
                CPMBidirectionalBuckBoost</a>:
144        </p>
145
146
147        <div class=\"figure\">
148         <p><img src=\"modelica://PVSystems/Resources/Images/
                USBBatteryConverterParameters.png\"
149               alt=\"USBBatteryConverterParameters.png\" />
150         </p>
151        </div>
152
153        <p>
154         This converter model includes both the electrical and
155         control components of a Current-Peak Mode controlled
156         modal non-inverting buck-boost. The default stop time is
157         set at 20 seconds. Running the simulation and plotting
158         the output voltage and current produces the following
159         result:
160        </p>
161
162
163        <div class=\"figure\">
164         <p><img src=\"modelica://PVSystems/Resources/Images/
                USBBatteryConverterResults.png\"
```

```
                alt=\"USBBatteryConverterResults.png\" /></p>
        </div>
      </html>"));
end USBBatteryConverter;
```

Examples/package.mo

```
within PVSystems;
package Examples "Application and validation examples"
extends Modelica.Icons.ExamplesPackage;

end Examples;
```

Examples/package.order

```
Application
Verification
```

Examples/Verification/CCM_DCMXVerification.mo

```
within PVSystems.Examples.Verification;
model CCM_DCMXVerification "Averaged CCM_DCM models verification"
 extends Modelica.Icons.Example;
 Modelica.Blocks.Sources.Ramp duty(
   duration=0.8,
   startTime=0.1,
   height=0.8,
   offset=0.1) annotation (Placement(transformation(
      extent={{-10,-10},{10,10}},
      rotation=0,
      origin={-70,-30})));
 Electrical.CCM_DCM1 ccm_dcm1(fs=100e3, Le=0.6e-6)
   annotation (Placement(transformation(extent={{30,60},{50,80}})));
 Modelica.Electrical.Analog.Sources.ConstantVoltage V1(V=10) annotation (
    Placement(transformation(
      extent={{-10,-10},{10,10}},
      rotation=270,
      origin={-20,70})));
 Modelica.Electrical.Analog.Basic.Resistor R1(R=1) annotation (Placement(
      transformation(
      extent={{-10,10},{10,-10}},
      rotation=270,
      origin={80,70})));
 Modelica.Electrical.Analog.Basic.Ground g1
   annotation (Placement(transformation(extent={{10,40},{30,60}})));
 Modelica.Electrical.Analog.Basic.Ground g2
   annotation (Placement(transformation(extent={{50,40},{70,60}})));
 Modelica.Electrical.Analog.Basic.Resistor Ri1(R=1e-3)
   annotation (Placement(transformation(extent={{-10,70},{10,90}})));
 Electrical.CCM_DCM2 ccm_dcm2(
   fs=100e3,
   n=2,
   Le=0.6e-6)
```

```
       annotation (Placement(transformation(extent={{30,0},{50,20}})));
  Modelica.Electrical.Analog.Sources.ConstantVoltage V2(V=10) annotation (
     Placement(transformation(
       extent={{-10,-10},{10,10}},
       rotation=270,
       origin={-20,10})));
  Modelica.Electrical.Analog.Basic.Resistor R2(R=1) annotation (Placement(
       transformation(
       extent={{-10,10},{10,-10}},
       rotation=270,
       origin={80,10})));
  Modelica.Electrical.Analog.Basic.Ground g3
    annotation (Placement(transformation(extent={{10,-20},{30,0}})));
  Modelica.Electrical.Analog.Basic.Ground g4
    annotation (Placement(transformation(extent={{50,-20},{70,0}})));
  Modelica.Electrical.Analog.Basic.Resistor Ri2(R=1e-3)
    annotation (Placement(transformation(extent={{-10,10},{10,30}})));
equation
  connect(V1.p,Ri1. p)
    annotation (Line(points={{-20,80},{-15,80},{-10,80}}, color={0,0,255}));
  connect(Ri1.n, ccm_dcm1.p1) annotation (Line(points={{10,80},{20,80},{20,75},{
        30,75}}, color={0,0,255}));
  connect(g1.p, ccm_dcm1.n1) annotation (Line(points={{20,60},{20,60},{20,65},{30,
        65}}, color={0,0,255}));
  connect(R1.p, ccm_dcm1.p2) annotation (Line(points={{80,80},{60,80},{60,75},{50,
        75}}, color={0,0,255}));
  connect(g2.p, ccm_dcm1.n2) annotation (Line(points={{60,60},{60,60},{60,65},{50,
        65}}, color={0,0,255}));
  connect(R1.n, g2.p)
    annotation (Line(points={{80,60},{60,60}}, color={0,0,255}));
  connect(V1.n,g1. p)
    annotation (Line(points={{-20,60},{0,60},{20,60}}, color={0,0,255}));
  connect(V2.p,Ri2. p)
    annotation (Line(points={{-20,20},{-20,20},{-10,20}}, color={0,0,255}));
  connect(V2.n,g3. p)
    annotation (Line(points={{-20,0},{-10,0},{20,0}}, color={0,0,255}));
  connect(Ri2.n, ccm_dcm2.p1) annotation (Line(points={{10,20},{20,20},{20,15},{
        30,15}}, color={0,0,255}));
  connect(g3.p, ccm_dcm2.n1)
    annotation (Line(points={{20,0},{20,5},{30,5}}, color={0,0,255}));
  connect(g4.p, ccm_dcm2.n2)
    annotation (Line(points={{60,0},{60,5},{50,5}}, color={0,0,255}));
  connect(R2.n, g4.p)
    annotation (Line(points={{80,0},{70,0},{60,0}}, color={0,0,255}));
  connect(R2.p, ccm_dcm2.p2) annotation (Line(points={{80,20},{60,20},{60,15},{50,
        15}}, color={0,0,255}));
  connect(duty.y, ccm_dcm2.d)
    annotation (Line(points={{-59,-30},{40,-30},{40,-2}}, color={0,0,127}));
  connect(duty.y, ccm_dcm1.d) annotation (Line(points={{-59,-30},{-40,-30},{-40,
        40},{40,40},{40,58}}, color={0,0,127}));
  annotation (
    experiment(StartTime=0, StopTime=1, Tolerance=1e-3), Diagram(graphics={
        Rectangle(extent={{-90,90},{90,-50}}, lineColor={255,255,255})}));
end CCM_DCMXVerification;
```

Examples/Verification/CCMXVerification.mo

```
within PVSystems.Examples.Verification;
model CCMXVerification "CCMX models verification"
  extends Modelica.Icons.Example;
  Electrical.CCM1 ccm1
    annotation (Placement(transformation(extent={{30,110},{50,130}})));
```

```
6  Modelica.Blocks.Sources.Ramp duty(
7    duration=0.8,
8    startTime=0.1,
9    height=0.8,
10   offset=0.1) annotation (Placement(transformation(
11      extent={{-10,-10},{10,10}},
12      rotation=0,
13      origin={-80,-120})));
14 Modelica.Electrical.Analog.Sources.ConstantVoltage V1(V=10) annotation (
15     Placement(transformation(
16      extent={{-10,-10},{10,10}},
17      rotation=270,
18      origin={-20,120})));
19 Modelica.Electrical.Analog.Basic.Resistor R1(R=1) annotation (Placement(
20      transformation(
21      extent={{-10,10},{10,-10}},
22      rotation=270,
23      origin={80,120})));
24 Modelica.Electrical.Analog.Basic.Ground g1
25   annotation (Placement(transformation(extent={{10,90},{30,110}})));
26 Modelica.Electrical.Analog.Basic.Ground g2
27   annotation (Placement(transformation(extent={{50,90},{70,110}})));
28 Modelica.Electrical.Analog.Basic.Resistor Ri1(R=1e-3)
29   annotation (Placement(transformation(extent={{-10,120},{10,140}})));
30 Electrical.CCM2 ccm2(
31   Ron=1,
32   RD=0.01,
33   VD=0.8)
34   annotation (Placement(transformation(extent={{30,60},{50,80}})));
35 Modelica.Electrical.Analog.Sources.ConstantVoltage V2(V=10) annotation (
36     Placement(transformation(
37      extent={{-10,-10},{10,10}},
38      rotation=270,
39      origin={-20,70})));
40 Modelica.Electrical.Analog.Basic.Resistor R2(R=1) annotation (Placement(
41      transformation(
42      extent={{-10,10},{10,-10}},
43      rotation=270,
44      origin={80,70})));
45 Modelica.Electrical.Analog.Basic.Ground g3
46   annotation (Placement(transformation(extent={{10,40},{30,60}})));
47 Modelica.Electrical.Analog.Basic.Ground g4
48   annotation (Placement(transformation(extent={{50,40},{70,60}})));
49 Modelica.Electrical.Analog.Basic.Resistor Ri2(R=1e-3)
50   annotation (Placement(transformation(extent={{-10,70},{10,90}})));
51 Electrical.CCM3 ccm3(n=2)
52   annotation (Placement(transformation(extent={{30,10},{50,30}})));
53 Modelica.Electrical.Analog.Sources.ConstantVoltage V3(V=10) annotation (
54     Placement(transformation(
55      extent={{-10,-10},{10,10}},
56      rotation=270,
57      origin={-20,20})));
58 Modelica.Electrical.Analog.Basic.Resistor R3(R=1) annotation (Placement(
59      transformation(
60      extent={{-10,10},{10,-10}},
61      rotation=270,
62      origin={80,20})));
63 Modelica.Electrical.Analog.Basic.Ground g5
64   annotation (Placement(transformation(extent={{10,-10},{30,10}})));
65 Modelica.Electrical.Analog.Basic.Ground g6
66   annotation (Placement(transformation(extent={{50,-10},{70,10}})));
67 Modelica.Electrical.Analog.Basic.Resistor Ri3(R=1e-3)
68   annotation (Placement(transformation(extent={{-10,20},{10,40}})));
69 Electrical.CCM4 ccm4(
70   Ron=1,
71   RD=0.01,
```

```
    n=2,
    VD=0.8)
    annotation (Placement(transformation(extent={{30,-40},{50,-20}})));
  Modelica.Electrical.Analog.Sources.ConstantVoltage V4(V=10) annotation (
      Placement(transformation(
        extent={{-10,-10},{10,10}},
        rotation=270,
        origin={-20,-30})));
  Modelica.Electrical.Analog.Basic.Resistor R4(R=1) annotation (Placement(
        transformation(
        extent={{-10,10},{10,-10}},
        rotation=270,
        origin={80,-30})));
  Modelica.Electrical.Analog.Basic.Ground g7
    annotation (Placement(transformation(extent={{10,-60},{30,-40}})));
  Modelica.Electrical.Analog.Basic.Ground g8
    annotation (Placement(transformation(extent={{50,-60},{70,-40}})));
  Modelica.Electrical.Analog.Basic.Resistor Ri4(R=1e-3)
    annotation (Placement(transformation(extent={{-10,-30},{10,-10}})));
  Electrical.CCM5 ccm5(
    Ron=1,
    VD=0.8,
    fs=100e3,
    Qr=0.75e-6,
    tr=75e-9)
    annotation (Placement(transformation(extent={{30,-92},{50,-72}})));
  Modelica.Electrical.Analog.Sources.ConstantVoltage V5(V=10) annotation (
      Placement(transformation(
        extent={{-10,-10},{10,10}},
        rotation=270,
        origin={-20,-82})));
  Modelica.Electrical.Analog.Basic.Resistor R5(R=1) annotation (Placement(
        transformation(
        extent={{-10,10},{10,-10}},
        rotation=270,
        origin={80,-82})));
  Modelica.Electrical.Analog.Basic.Ground g9
    annotation (Placement(transformation(extent={{10,-112},{30,-92}})));
  Modelica.Electrical.Analog.Basic.Ground g10
    annotation (Placement(transformation(extent={{50,-112},{70,-92}})));
  Modelica.Electrical.Analog.Basic.Resistor Ri5(R=1e-3)
    annotation (Placement(transformation(extent={{-10,-82},{10,-62}})));
equation
  connect(R1.p, ccm1.p2) annotation (Line(points={{80,130},{60,130},{60,125},{
        50,125}},
            color={0,0,255}));
  connect(ccm1.n2,R1. n)
    annotation (Line(points={{50,115},{60,115},{60,110},{80,110}},
                                              color={0,0,255}));
  connect(g1.p, ccm1.n1) annotation (Line(points={{20,110},{20,110},{20,115},{
        30,115}},
            color={0,0,255}));
  connect(V1.n, g1.p)
    annotation (Line(points={{-20,110},{20,110}},
                                    color={0,0,255}),
      experiment(
        StartTime=0,
        StopTime=1,
        Tolerance=1e-3));
  connect(V1.p, Ri1.p)
    annotation (Line(points={{-20,130},{-10,130}}, color={0,0,255}));
  connect(Ri1.n, ccm1.p1) annotation (Line(points={{10,130},{20,130},{20,125},{
        30,125}}, color={0,0,255}));
  connect(duty.y, ccm5.d) annotation (Line(points={{-69,-120},{-14,-120},{40,
        -120},{40,-94}}, color={0,0,127}));
  connect(ccm4.d, ccm5.d) annotation (Line(points={{40,-42},{40,-60},{-52,-60},
```

```
138         {-52,-120},{40,-120},{40,-94}}, color={0,0,127}));
139   connect(ccm3.d, ccm5.d) annotation (Line(points={{40,8},{40,-10},{-52,-10},{
140         -52,-120},{40,-120},{40,-94}}, color={0,0,127}));
141   connect(ccm2.d, ccm5.d) annotation (Line(points={{40,58},{40,40},{-52,40},{
142         -52,-120},{40,-120},{40,-94}}, color={0,0,127}));
143   connect(ccm1.d, ccm5.d) annotation (Line(points={{40,108},{40,92},{-52,92},{
144         -52,-120},{40,-120},{40,-94}}, color={0,0,127}));
145   connect(V2.p, Ri2.p)
146     annotation (Line(points={{-20,80},{-10,80},{-10,80}}, color={0,0,255}));
147   connect(Ri2.n, ccm2.p1) annotation (Line(points={{10,80},{20,80},{20,75},{30,
148         75}}, color={0,0,255}));
149   connect(g3.p, ccm2.n1) annotation (Line(points={{20,60},{20,60},{20,65},{30,
150         65}}, color={0,0,255}));
151   connect(R2.p, ccm2.p2) annotation (Line(points={{80,80},{60,80},{60,75},{50,
152         75}}, color={0,0,255}));
153   connect(g4.p, ccm2.n2) annotation (Line(points={{60,60},{60,60},{60,65},{50,
154         65}}, color={0,0,255}));
155   connect(V3.p, Ri3.p)
156     annotation (Line(points={{-20,30},{-15,30},{-10,30}}, color={0,0,255}));
157   connect(Ri3.n, ccm3.p1) annotation (Line(points={{10,30},{20,30},{20,25},{30,
158         25}}, color={0,0,255}));
159   connect(g5.p, ccm3.n1) annotation (Line(points={{20,10},{20,10},{20,15},{30,
160         15}}, color={0,0,255}));
161   connect(R3.p, ccm3.p2) annotation (Line(points={{80,30},{72,30},{60,30},{60,
162         25},{50,25}}, color={0,0,255}));
163   connect(g6.p, ccm3.n2) annotation (Line(points={{60,10},{60,6},{60,15},{50,15}},
164       color={0,0,255}));
165   connect(V4.p, Ri4.p)
166     annotation (Line(points={{-20,-20},{-15,-20},{-10,-20}}, color={0,0,255}));
167   connect(Ri4.n, ccm4.p1) annotation (Line(points={{10,-20},{20,-20},{20,-25},{
168         30,-25}}, color={0,0,255}));
169   connect(g7.p, ccm4.n1) annotation (Line(points={{20,-40},{20,-40},{20,-35},{
170         30,-35}}, color={0,0,255}));
171   connect(V5.p, Ri5.p)
172     annotation (Line(points={{-20,-72},{-15,-72},{-10,-72}}, color={0,0,255}));
173   connect(Ri5.n, ccm5.p1) annotation (Line(points={{10,-72},{20,-72},{20,-77},{
174         30,-77}}, color={0,0,255}));
175   connect(g9.p, ccm5.n1) annotation (Line(points={{20,-92},{20,-92},{20,-87},{
176         30,-87}}, color={0,0,255}));
177   connect(R5.p, ccm5.p2) annotation (Line(points={{80,-72},{60,-72},{60,-77},{
178         50,-77}}, color={0,0,255}));
179   connect(g10.p, ccm5.n2) annotation (Line(points={{60,-92},{60,-92},{60,-87},{
180         50,-87}}, color={0,0,255}));
181   connect(R4.p, ccm4.p2) annotation (Line(points={{80,-20},{60,-20},{60,-25},{
182         50,-25}}, color={0,0,255}));
183   connect(g8.p, ccm4.n2) annotation (Line(points={{60,-40},{60,-40},{60,-35},{
184         50,-35}}, color={0,0,255}));
185   connect(ccm1.n2, g2.p)
186     annotation (Line(points={{50,115},{60,115},{60,110}}, color={0,0,255}));
187   connect(R2.n, g4.p)
188     annotation (Line(points={{80,60},{70,60},{60,60}}, color={0,0,255}));
189   connect(R3.n, g6.p)
190     annotation (Line(points={{80,10},{78,10},{60,10}}, color={0,0,255}));
191   connect(R4.n, g8.p)
192     annotation (Line(points={{80,-40},{60,-40}}, color={0,0,255}));
193   connect(R5.n, g10.p)
194     annotation (Line(points={{80,-92},{60,-92}}, color={0,0,255}));
195   connect(V2.n, g3.p)
196     annotation (Line(points={{-20,60},{0,60},{20,60}}, color={0,0,255}));
197   connect(V3.n, g5.p)
198     annotation (Line(points={{-20,10},{0,10},{20,10}}, color={0,0,255}));
199   connect(V4.n, g7.p)
200     annotation (Line(points={{-20,-40},{0,-40},{20,-40}}, color={0,0,255}));
201   connect(V5.n, g9.p)
202     annotation (Line(points={{-20,-92},{0,-92},{20,-92}}, color={0,0,255}));
203   annotation (
```

```
204     experiment(
205       StartTime=0, StopTime=1,Tolerance=1e-3),
206     Diagram(
207       coordinateSystem(extent={{-100,-140},{100,140}}, initialScale=0.1),
208         graphics={Rectangle(extent={{-100,140},{100,-140}}, lineColor={255,255,
209             255})})),
210     Icon(
211       coordinateSystem(extent={{-100,-100},{100,100}}, initialScale=0.1)));
212 end CCMXVerification;
```

Examples/Verification/CPM_CCMVerification.mo

```
1  within PVSystems.Examples.Verification;
2  model CPM_CCMVerification "Averaged CPM_CCM verification"
3    extends Modelica.Icons.Example;
4    Control.SwitchingCPM switchingCPM(
5      dMin=0.05,
6      dMax=0.95,
7      fs=200e3,
8      vcMax=10,
9      Va=0.01) annotation (Placement(transformation(extent={{-40,36},{-20,56}})));
10   Modelica.Blocks.Continuous.Integrator integrator(initType=
         Modelica.Blocks.Types.Init.InitialState,
11     k=1e4,
12     y_start=3.99)
13     annotation (Placement(transformation(extent={{50,40},{70,60}})));
14   Modelica.Blocks.Sources.Constant vdT(k=2)
15     annotation (Placement(transformation(extent={{-40,70},{-20,90}})));
16   Modelica.Blocks.Logical.Switch switch1
17     annotation (Placement(transformation(extent={{20,40},{40,60}})));
18   Modelica.Blocks.Sources.Constant vdpT(k=-1)
19     annotation (Placement(transformation(extent={{-40,4},{-20,24}})));
20   Modelica.Blocks.Sources.Constant vc(k=4)
21     annotation (Placement(transformation(extent={{-90,40},{-70,60}})));
22   Control.CPM_CCM CPM_CCM(
23     L=1e-4,
24     fs=200e3,
25     Rf=1,
26     d_disabled=0.05,
27     Va=0.01) annotation (Placement(transformation(extent={{40,-40},{60,-20}})));
28   Modelica.Blocks.Math.Abs abs1
29     annotation (Placement(transformation(extent={{0,-70},{20,-50}})));
30   Modelica.Blocks.Sources.BooleanStep enable(startTime=1e-5) annotation (
31       Placement(transformation(
32         extent={{-10,-10},{10,10}},
33         rotation=90,
34         origin={50,-60})));
35   Modelica.Blocks.Math.Mean mean(f=200e3)
36     annotation (Placement(transformation(extent={{-40,-36},{-20,-16}})));
37 equation
38   connect(vdT.y, switch1.u1)
39     annotation (Line(points={{-19,80},{0,80},{0,58},{18,58}},
40                                               color={0,0,127}));
41   connect(switch1.y, integrator.u)
42     annotation (Line(points={{41,50},{41,50},{48,50}},
43                                         color={0,0,127}));
44   connect(vdpT.y, switch1.u3) annotation (Line(points={{-19,14},{10,14},{10,42},
45         {18,42}}, color={0,0,127}));
46   connect(switchingCPM.c, switch1.u2)
47     annotation (Line(points={{-19,50},{18,50}},
48                                    color={255,0,255}));
49   connect(integrator.y, switchingCPM.vs) annotation (Line(points={{71,50},{80,50},
```

```
      {80,-80},{-50,-80},{-50,42},{-42,42}}, color={0,0,127}));
  connect(vc.y, switchingCPM.vc)
    annotation (Line(points={{-69,50},{-42,50}},
                                  color={0,0,127}));
  connect(CPM_CCM.vc, switchingCPM.vc) annotation (Line(points={{38,-20},{20,-20},
      {20,-8},{-60,-8},{-60,50},{-42,50}}, color={0,0,127}));
  connect(abs1.u, vdpT.y) annotation (Line(points={{-2,-60},{-10,-60},{-10,14},{
      -19,14}}, color={0,0,127}));
  connect(abs1.y, CPM_CCM.vm2) annotation (Line(points={{21,-60},{28,-60},{28,-40},
      {38,-40}}, color={0,0,127}));
  connect(CPM_CCM.vm1, vdT.y) annotation (Line(points={{38,-34},{0,-34},{0,80},{
      -19,80}}, color={0,0,127}));
  connect(mean.y, CPM_CCM.vs)
    annotation (Line(points={{-19,-26},{10,-26},{38,-26}}, color={0,0,127}));
  connect(mean.u, switchingCPM.vs) annotation (Line(points={{-42,-26},{-50,-26},
      {-50,42},{-42,42}}, color={0,0,127}));
  connect(enable.y, CPM_CCM.enable)
    annotation (Line(points={{50,-49},{50,-42}}, color={255,0,255}));
  annotation (experiment(StopTime=2e-4), Diagram(graphics={Rectangle(extent={{
          -100,100},{90,-90}}, lineColor={255,255,255})}));
end CPM_CCMVerification;
```

Examples/Verification/CPMVerification.mo

```
within PVSystems.Examples.Verification;
model CPMVerification "Averaged CPM verification"
  extends Modelica.Icons.Example;
  extends Modelica.Icons.UnderConstruction;
  Control.SwitchingCPM switchingCPM(
    dMin=0.05,
    dMax=0.95,
    fs=200e3,
    vcMax=10,
    Va=0.1) annotation (Placement(transformation(extent={{-40,36},{-20,56}})));
  Modelica.Blocks.Continuous.LimIntegrator
                               integrator(initType=Modelica.Blocks.Types.Init.InitialState,

    k=1e4,
    outMin=0,
    outMax=Modelica.Constants.inf)
    annotation (Placement(transformation(extent={{50,40},{70,60}})));
  Modelica.Blocks.Sources.Constant vdT(k=6)
    annotation (Placement(transformation(extent={{-40,70},{-20,90}})));
  Modelica.Blocks.Logical.Switch switch1
    annotation (Placement(transformation(extent={{20,40},{40,60}})));
  Modelica.Blocks.Sources.Constant vdpT(k=-3)
    annotation (Placement(transformation(extent={{-40,4},{-20,24}})));
  Modelica.Blocks.Sources.Constant vc(k=0.5)
    annotation (Placement(transformation(extent={{-90,40},{-70,60}})));
  Control.CPM CPM(
    fs=200e3,
    Rf=1,
    L=1e-4,
    Va=0.1) annotation (Placement(transformation(extent={{40,-40},{60,-20}})));
  Modelica.Blocks.Math.Abs abs1
    annotation (Placement(transformation(extent={{0,-70},{20,-50}})));
  Modelica.Blocks.Math.Mean mean(f=200e3)
    annotation (Placement(transformation(extent={{-40,-36},{-20,-16}})));
equation
  connect(vdT.y, switch1.u1)
    annotation (Line(points={{-19,80},{0,80},{0,58},{18,58}},
                                           color={0,0,127}));
```

```
  connect(switch1.y, integrator.u)
    annotation (Line(points={{41,50},{41,50},{48,50}},
                                        color={0,0,127}));
  connect(vdpT.y, switch1.u3) annotation (Line(points={{-19,14},{10,14},{10,42},
        {18,42}}, color={0,0,127}));
  connect(switchingCPM.c, switch1.u2)
    annotation (Line(points={{-19,50},{18,50}},
                                    color={255,0,255}));
  connect(integrator.y, switchingCPM.vs) annotation (Line(points={{71,50},{80,50},
        {80,-80},{-50,-80},{-50,42},{-42,42}}, color={0,0,127}));
  connect(vc.y, switchingCPM.vc)
    annotation (Line(points={{-69,50},{-42,50}},
                                      color={0,0,127}));
  connect(CPM.vc, switchingCPM.vc) annotation (Line(points={{38,-20},{20,-20},{
        20,-8},{-60,-8},{-60,50},{-42,50}}, color={0,0,127}));
  connect(abs1.u, vdpT.y) annotation (Line(points={{-2,-60},{-10,-60},{-10,14},{
        -19,14}}, color={0,0,127}));
  connect(abs1.y, CPM.vm2) annotation (Line(points={{21,-60},{28,-60},{28,-40},
        {38,-40}}, color={0,0,127}));
  connect(CPM.vm1, vdT.y) annotation (Line(points={{38,-34},{0,-34},{0,80},{-19,
        80}}, color={0,0,127}));
  connect(mean.y, CPM.vs)
    annotation (Line(points={{-19,-26},{10,-26},{38,-26}}, color={0,0,127}));
  connect(mean.u, switchingCPM.vs) annotation (Line(points={{-42,-26},{-50,-26},
        {-50,42},{-42,42}}, color={0,0,127}));
  annotation (experiment(StopTime=2e-4), Diagram(graphics={Rectangle(extent={{
          -100,100},{90,-90}}, lineColor={255,255,255})}));
end CPMVerification;
```

Examples/Verification/DeadTimeVerification.mo

```
within PVSystems.Examples.Verification;
model DeadTimeVerification "DeadTime verification"
  extends Modelica.Icons.Example;
  Control.DeadTime deadTime(deadTime=0.03)
    annotation (Placement(transformation(extent={{0,-10},{20,10}})));
  Modelica.Blocks.Sources.BooleanPulse booleanPulse(period=0.2)
    annotation (Placement(transformation(extent={{-40,-10},{-20,10}})));
equation
  connect(booleanPulse.y, deadTime.c)
    annotation (Line(points={{-19,0},{-2,0}}, color={255,0,255}));
  annotation (
    experiment(StartTime=0, StopTime=1, Tolerance=1e-3), Diagram(graphics={
        Rectangle(extent={{-50,20},{30,-20}}, lineColor={255,255,255})}));
end DeadTimeVerification;
```

Examples/Verification/IdealCBSwitchVerification.mo

```
within PVSystems.Examples.Verification;
model IdealCBSwitchVerification
  "Ideal current bidirectional switch verification"
  extends Modelica.Icons.Example;
  Electrical.IdealCBSwitch idealCBSwitch annotation (Placement(transformation(
      origin={-30,10},
      extent={{-10,-10},{10,10}},
      rotation=270)));
  Modelica.Electrical.Analog.Sources.SineVoltage sineVoltage(freqHz=5, V=1)
```

```
10     annotation (Placement(transformation(
11        origin={30,10},
12        extent={{-10,-10},{10,10}},
13        rotation=270)));
14   Modelica.Blocks.Sources.BooleanStep booleanStep(startValue=true, startTime=
15        0.5) annotation (Placement(transformation(extent={{-70,0},{-50,20}},
16          rotation=0)));
17   Modelica.Electrical.Analog.Basic.Ground ground annotation (Placement(
18        transformation(extent={{-10,-30},{10,-10}},rotation=0)));
19   Modelica.Electrical.Analog.Basic.Resistor resistor(R=2) annotation (Placement(
20        transformation(
21        origin={0,30},
22        extent={{-10,-10},{10,10}},
23        rotation=180)));
24 equation
25   connect(booleanStep.y, idealCBSwitch.c)
26     annotation (Line(points={{-49,10},{-49,10},{-37,10}}, color={255,0,255}));
27   connect(idealCBSwitch.p, resistor.n)
28     annotation (Line(points={{-30,20},{-30,30},{-10,30}}, color={0,0,255}));
29   connect(resistor.p, sineVoltage.p)
30     annotation (Line(points={{10,30},{30,30},{30,20}}, color={0,0,255}));
31   connect(idealCBSwitch.n, ground.p)
32     annotation (Line(points={{-30,0},{-30,-10},{0,-10}}, color={0,0,255}));
33   connect(ground.p, sineVoltage.n) annotation (Line(points={{0,-10},{16,-10},{
34         30,-10},{30,0}}, color={0,0,255}));
35   annotation (
36     experiment(
37       StartTime=0,
38       StopTime=1,
39       Tolerance=1e-3),
40     Documentation(info="<html>
41        <p>
42         This example presents a circuit composed of a resistor
43         in series with a sinusoidal AC voltage source and the
44         ideal current bidirectional switch. The switch is
45         operated by a step block that changes from 0 to 1 in the
46         middle of the simulation. This changes the state of the
47         switch from open to closed.
48        </p>
49
50        <p>
51         To use the example, simulate the model as provided and
52         plot the source voltage as well as the switch voltage,
53         the plot should look like this:
54        </p>
55
56
57        <div class=\"figure\">
58         <p><img src=\"modelica://PVSystems/Resources/Images/
               IdealCBSwitchVerificationResults.png\"
59               alt=\"IdealCBSwitchVerificationResults.png\" />
60         </p>
61        </div>
62
63        <p>
64         Notice how at the begining of the simulation, when the
65         switch is not closed, it blocks all the positive
66         voltage, preventing current from flowing. On the other
67         hand, the negative voltage is not blocked, so the
68         current can flow (through the anti-parallel diode). When
69         the switch is closed using the firing signal, it never
70         blocks voltage, allowing bidirectional flow of current.
71        </p>
72
73        <p>
74         Plot the voltage drop in the resistor to confirm these
```

```
        results or play with the parameter values to see what
        effects they have.</p>
    </html>"),
   Diagram(graphics={Rectangle(extent={{-80,40},{48,-30}}, lineColor={255,255,
          255})}));
end IdealCBSwitchVerification;
```

Examples/Verification/MPPTControllerVerification.mo

```
within PVSystems.Examples.Verification;
model MPPTControllerVerification "MPPT controller verification"
  extends Modelica.Icons.Example;
  Modelica.Electrical.Analog.Basic.Ground ground annotation (Placement(
      transformation(extent={{-30,-40},{-10,-20}}, rotation=0)));
  Electrical.PVArray pVArray annotation (Placement(transformation(
      origin={-40,-10},
      extent={{-10,-10},{10,10}},
      rotation=270)));
  Modelica.Electrical.Analog.Sources.SignalVoltage sink annotation (Placement(
      transformation(
      origin={0,-10},
      extent={{-10,-10},{10,10}},
      rotation=270)));
  Control.MPPTController mpptController(
    sampleTime=1,
    pkThreshold=0.01,
    vrefStep=1,
    vrefStart=5) annotation (Placement(transformation(
      origin={-30,74},
      extent={{-10,-10},{10,10}},
      rotation=0)));
  Modelica.Blocks.Sources.Ramp G(
    offset=1000,
    height=-500,
    startTime=30,
    duration=10) annotation (Placement(transformation(extent={{-90,0},{-70,20}},
        rotation=0)));
  Modelica.Blocks.Sources.Ramp T(
    height=-25,
    offset=273.15 + 25,
    startTime=50,
    duration=50) annotation (Placement(transformation(extent={{-80,-80},{-60,-60}},
        rotation=0)));
  Modelica.Blocks.Math.Add vdcSetpoint annotation (Placement(transformation(
      origin={30,54},
      extent={{-10,-10},{10,10}},
      rotation=0)));
  Modelica.Blocks.Sources.Ramp perturbation(
    height=10,
    offset=0,
    duration=20,
    startTime=130) annotation (Placement(transformation(
      origin={-30,34},
      extent={{-10,-10},{10,10}},
      rotation=0)));
  Modelica.Blocks.Sources.RealExpression vsense(y=sink.v)
    annotation (Placement(transformation(extent={{-80,70},{-60,90}})));
  Modelica.Blocks.Sources.RealExpression isense(y=sink.i)
    annotation (Placement(transformation(extent={{-80,44},{-60,64}})));
  Modelica.Blocks.Sources.RealExpression vdcSetpoint1(y=26)
    annotation (Placement(transformation(extent={{60,-60},{40,-40}})));
  Modelica.Electrical.Analog.Basic.Ground ground1 annotation (Placement(
```

```
54        transformation(extent={{-30,-80},{-10,-60}}, rotation=0)));
55    Electrical.PVArray pVArray1 annotation (Placement(transformation(
56        origin={-40,-50},
57        extent={{-10,-10},{10,10}},
58        rotation=270)));
59    Modelica.Electrical.Analog.Sources.SignalVoltage sink1 annotation (Placement(
60        transformation(
61        origin={0,-50},
62        extent={{-10,-10},{10,10}},
63        rotation=270)));
64  equation
65    connect(G.y, pVArray.G) annotation (Line(points={{-69,10},{-60,10},{-60,-7},{
66          -45.5,-7}}, color={0,0,127}));
67    connect(vdcSetpoint.y, sink.v) annotation (Line(points={{41,54},{60,54},{60,-10},
68          {7,-10}}, color={0,0,127}));
69    connect(perturbation.y, vdcSetpoint.u2) annotation (Line(points={{-19,34},{0,34},
70          {0,48},{18,48}}, color={0,0,127}));
71    connect(pVArray.p, sink.p)
72      annotation (Line(points={{-40,0},{0,0}}, color={0,0,255}));
73    connect(vsense.y, mpptController.u1)
74      annotation (Line(points={{-59,80},{-59,80},{-42,80}}, color={0,0,127}));
75    connect(mpptController.y, vdcSetpoint.u1) annotation (Line(points={{-19,74},{0,
76          74},{0,60},{18,60}}, color={0,0,127}));
77    connect(isense.y, mpptController.u2) annotation (Line(points={{-59,54},{-50,54},
78          {-50,68},{-42,68}}, color={0,0,127}));
79    connect(pVArray1.p, sink1.p) annotation (Line(points={{-40,-40},{-28,-40},{-14,
80          -40},{0,-40}}, color={0,0,255}));
81    connect(sink1.v, vdcSetpoint1.y)
82      annotation (Line(points={{7,-50},{39,-50}}, color={0,0,127}));
83    connect(T.y, pVArray1.T) annotation (Line(points={{-59,-70},{-52,-70},{-52,-53},
84          {-45.5,-53}}, color={0,0,127}));
85    connect(T.y, pVArray.T) annotation (Line(points={{-59,-70},{-52,-70},{-52,-13},
86          {-45.5,-13}}, color={0,0,127}));
87    connect(G.y, pVArray1.G) annotation (Line(points={{-69,10},{-60,10},{-60,-47},
88          {-45.5,-47}}, color={0,0,127}));
89    connect(pVArray.n, ground.p)
90      annotation (Line(points={{-40,-20},{-30,-20},{-20,-20}}, color={0,0,255}));
91    connect(sink.n, ground.p)
92      annotation (Line(points={{0,-20},{-10,-20},{-20,-20}}, color={0,0,255}));
93    connect(pVArray1.n, ground1.p)
94      annotation (Line(points={{-40,-60},{-20,-60}}, color={0,0,255}));
95    connect(ground1.p, sink1.n)
96      annotation (Line(points={{-20,-60},{-1.77636e-015,-60}}, color={0,0,255}));
97    annotation (experiment(StopTime=180), Documentation(info="<html>
98        <p>
99          This examples places an MPPT controller closing the loop
100         for a voltage source connected to a PV array. The MPPT
101         controller senses the power coming out of the PV array
102         and provides a setpoint for the voltage source. This
103         changes the operation point of the PV array with the
104         goal of maximizing its output power for any given solar
105         irradiation and junction temperature conditions.
106       </p>
107
108       <p>
109         The model is designed to challenge the control by
110         ramping solar irradiation, temperature at different
111         times and by injecting a perturbation into the control
112         loop:
113       </p>
114
115
116       <div class=\"figure\">
117         <p><img src=\"modelica://PVSystems/Resources/Images/
              MPPTControllerVerificationResultsA.png\"
118             alt=\"MPPTControllerVerificationResultsA.png\"
```

```
119               />
120         </p>
121       </div>
122
123       <p>
124         The MPPT controller successfully deals with these
125         changing conditions as shown in the following plots,
126         which compares the static PV array control with the MPPT
127         control:
128       </p>
129
130
131       <div class=\"figure\">
132         <p><img src=\"modelica://PVSystems/Resources/Images/
              MPPTControllerVerificationResultsB.png\"
133               alt=\"MPPTControllerVerificationResultsB.png\"
134               /></p>
135       </div>
136     </html>"),
137    Diagram(graphics={Rectangle(extent={{-100,92},{70,-90}}, lineColor={255,255,
           255})}));
139 end MPPTControllerVerification;
```

Examples/Verification/package.mo

```
1  within PVSystems.Examples;
2  package Verification "Simple examples for verification of library's components"
3  extends Modelica.Icons.ExamplesPackage;
4
5
6
7
8
9
10
11
12 end Verification;
```

Examples/Verification/package.order

```
1  IdealCBSwitchVerification
2  SW1Verification
3  SW2Verification
4  SW3Verification
5  CCMXVerification
6  CCM_DCMXVerification
7  PVArrayVerification
8  SimpleBatteryVerification
9  SwitchingPWMVerification
10 SwitchingCPMVerification
11 DeadTimeVerification
12 CPM_CCMVerification
13 CPMVerification
14 ParkTransformsVerification
15 PLLVerification
16 MPPTControllerVerification
```

Examples/Verification/ParkTransformsVerification.mo

```
within PVSystems.Examples.Verification;
model ParkTransformsVerification "Park transforms verification"
  extends Modelica.Icons.Example;
  Control.Park park
    annotation (Placement(transformation(extent={{0,20},{20,40}}, rotation=0)));
  Control.InversePark inversePark annotation (Placement(transformation(extent={
          {40,20},{60,40}}, rotation=0)));
  Modelica.Blocks.Sources.SawTooth sawTooth(amplitude=2*Modelica.Constants.pi,
      period=0.02) annotation (Placement(transformation(extent={{-80,-40},{-60,
          -20}}, rotation=0)));
  Modelica.Blocks.Math.Sin sin annotation (Placement(transformation(extent={{-40,
          0},{-20,20}}, rotation=0)));
  Modelica.Blocks.Math.Cos cos annotation (Placement(transformation(extent={{-40,
          40},{-20,60}}, rotation=0)));
equation
  connect(park.d, inversePark.d)
    annotation (Line(points={{21,34},{38,34}}, color={0,0,127}));
  connect(park.q, inversePark.q)
    annotation (Line(points={{21,26},{38,26}}, color={0,0,127}));
  connect(cos.u, sawTooth.y) annotation (Line(points={{-42,50},{-50,50},{-50,-30},
        {-59,-30}}, color={0,0,127}));
  connect(sin.u, sawTooth.y) annotation (Line(points={{-42,10},{-50,10},{-50,-30},
        {-59,-30}}, color={0,0,127}));
  connect(cos.y, park.alpha) annotation (Line(points={{-19,50},{-10,50},{-10,34},
        {-2,34}}, color={0,0,127}));
  connect(sin.y, park.beta) annotation (Line(points={{-19,10},{-10,10},{-10,26},
        {-2,26}}, color={0,0,127}));
  connect(park.theta, sawTooth.y)
    annotation (Line(points={{10,18},{10,-30},{-59,-30}}, color={0,0,127}));
  connect(inversePark.theta, sawTooth.y)
    annotation (Line(points={{50,18},{50,-30},{-59,-30}}, color={0,0,127}));
  annotation (
    Diagram(graphics={Rectangle(extent={{-90,70},{70,-50}}, lineColor={255,255,
          255})}),
    experiment(StopTime=0.1),
    Documentation(info="<html>
       <p>
        This example provides some easy input for the Park
        transform blocks to check that calculations are being
        done as expected. Run the simulation and you should get
        something like the following figure:
       </p>


       <div class=\"figure\">
        <p><img src=\"modelica://PVSystems/Resources/Images/ParkVerificationResults.png\"
              alt=\"ParkVerificationResults.png\" />
        </p>
       </div>

       <p>
        As expected, <em>d</em> is equal to the peak amplitude
        of the input signal and <em>q</em> sets at zero. Feeding
        the signals back to the inverse transformation block
        recreates the original signals (which overlap them on
        the plot).</p>
     </html>"));
end ParkTransformsVerification;
```

Examples/Verification/PLLVerification.mo

```
within PVSystems.Examples.Verification;
model PLLVerification "PLL verification"
  extends Modelica.Icons.Example;
  Modelica.Blocks.Sources.Sine source(freqHz=50) annotation (Placement(
      transformation(extent={{-50,-10},{-30,10}}, rotation=0)));
  Control.PLL pLL annotation (Placement(transformation(extent={{-10,-10},{10,10}},
       rotation=0)));
  Modelica.Blocks.Math.Cos sync annotation (Placement(transformation(extent={{
        30,-10},{50,10}}, rotation=0)));
equation
  connect(source.y, pLL.v)
    annotation (Line(points={{-29,0},{-12,0}}, color={0,0,127}));
  connect(pLL.theta, sync.u)
    annotation (Line(points={{11,0},{28,0}}, color={0,0,127}));
  annotation (
    Diagram(graphics={Rectangle(extent={{-60,20},{60,-20}}, lineColor={255,255,
          255})}),
    experiment(
      StartTime=0,
      StopTime=0.1,
      Tolerance=1e-4),
    Documentation(info="<html>
       <p>
        This simple example provides a sinusoidal input to the
        PLL block and applies the output provided by the PLL,
        the calculated phase of the input sine, to drive a sine
        block so that the synchronization capabilities of the
        PLL can be visualized.
       </p>

       <p>
        Run the model and plot the output of the sinusoidal
        source and the output of the sine block to see how,
        after some short transient, the PLL successfully follows
        the reference:
       </p>


       <div class=\"figure\">
        <p><img src=\"modelica://PVSystems/Resources/Images/PLLVerificationResults.png\"
              alt=\"PLLVerificationResults.png\" /></p>
       </div>
     </html>"));
end PLLVerification;
```

Examples/Verification/PVArrayVerification.mo

```
within PVSystems.Examples.Verification;
model PVArrayVerification "PVArray verification"
  extends Modelica.Icons.Example;
  Modelica.Electrical.Analog.Sources.RampVoltage rampVoltage(
    duration=1,
    V=45,
    offset=-10) annotation (Placement(transformation(
       origin={40,10},
       extent={{-10,-10},{10,10}},
       rotation=270)));
  Modelica.Electrical.Analog.Basic.Ground ground annotation (Placement(
```

```
12        transformation(extent={{30,-40},{50,-20}}, rotation=0)));
13    Electrical.PVArray pVArray annotation (Placement(transformation(
14        origin={0,10},
15        extent={{-10,-10},{10,10}},
16        rotation=270)));
17    Modelica.Blocks.Sources.Constant Gn(k=1000) annotation (Placement(
18        transformation(extent={{-50,10},{-30,30}}, rotation=0)));
19    Modelica.Blocks.Sources.Constant Tn(k=298.15) annotation (Placement(
20        transformation(extent={{-50,-24},{-30,-4}}, rotation=0)));
21  equation
22    connect(Gn.y, pVArray.G) annotation (Line(points={{-29,20},{-16,20},{-16,13},
23          {-5.5,13}}, color={0,0,127}));
24    connect(Tn.y, pVArray.T) annotation (Line(points={{-29,-14},{-16,-14},{-16,7},
25          {-5.5,7}}, color={0,0,127}));
26    connect(pVArray.p, rampVoltage.p)
27      annotation (Line(points={{1.83691e-015,20},{40,20}}, color={0,0,255}));
28    connect(pVArray.n, rampVoltage.n)
29      annotation (Line(points={{-1.83691e-015,0},{40,0}}, color={0,0,255}));
30    connect(ground.p, rampVoltage.n)
31      annotation (Line(points={{40,-20},{40,0}}, color={0,0,255}));
32    annotation (
33      Diagram(graphics={Rectangle(extent={{-60,40},{60,-40}}, lineColor={255,255,
34            255})}),
35      Documentation(info="<html>
36         <p>
37          A ramp DC voltage source is applied in parallel to an
38          instance of the PVArray model. The voltage ramp is
39          configured to sweep from -10 volts to 35 volts in 1
40          second. This provides the enough voltage range to cover
41          all of the PV array's working range when initialized
42          with default values.
43         </p>
44
45         <p>
46          To use the example, simulate the model and start by
47          displaying both voltage and current of the ramp voltage
48          source. A figure like the following should be displayed:
49         </p>
50
51
52         <div class=\"figure\">
53          <p><img src=\"modelica://PVSystems/Resources/Images/PVArrayVerificationResults.png
                \"
54                alt=\"PVArrayVerificationResults.png\" />
55          </p>
56         </div>
57
58         <p>
59          Notice how the variation in the current delivered by the
60          PV array (sinked by the voltage source) reflects the
61          familiar PV module curve.
62         </p>
63
64         <p>
65          Modify the values for the irradiance and temperature
66          blocks and see how these changes are reflected in a
67          change in the PV curve, accurately reflecting the
68          effects of these variables in the PV module
69          performance. </p>
70       </html>"),
71      experiment(
72        StartTime=0,
73        StopTime=1,
74        Tolerance=1e-4));
75  end PVArrayVerification;
```

Examples/Verification/SimpleBatteryVerification.mo

```
within PVSystems.Examples.Verification;
model SimpleBatteryVerification "SimpleBattery verification"
  extends Modelica.Icons.Example;
  Modelica.Electrical.Analog.Sources.SignalCurrent CC annotation (Placement(
      transformation(
      extent={{-10,-10},{10,10}},
      rotation=90,
      origin={60,10})));
  Modelica.Electrical.Analog.Basic.Ground ground
    annotation (Placement(transformation(extent={{50,-40},{70,-20}})));
  Electrical.SimpleBattery B(Q=1, DoDini=0.5) annotation (Placement(
      transformation(
      extent={{-10,-10},{10,10}},
      rotation=270,
      origin={90,10})));
  Modelica.Blocks.Nonlinear.SlewRateLimiter slewRateLimiter(Rising=4)
    annotation (Placement(transformation(extent={{20,0},{40,20}})));
  Modelica.Blocks.Logical.Hysteresis hysteresis(uHigh=4.19, uLow=0.1)
    annotation (Placement(transformation(extent={{-54,0},{-34,20}})));
  Modelica.Blocks.Logical.Switch switch1
    annotation (Placement(transformation(extent={{-14,0},{6,20}})));
  Modelica.Blocks.Sources.RealExpression idis(y=-2)
    annotation (Placement(transformation(extent={{-54,30},{-34,50}})));
  Modelica.Blocks.Sources.RealExpression ich(y=2)
    annotation (Placement(transformation(extent={{-54,-30},{-34,-10}})));
  Modelica.Blocks.Sources.RealExpression vsense(y=B.v)
    annotation (Placement(transformation(extent={{-90,0},{-70,20}})));
equation
  connect(ground.p, CC.p)
    annotation (Line(points={{60,-20},{60,-20},{60,0}}, color={0,0,255}));
  connect(CC.p, B.n) annotation (Line(points={{60,0},{90,0}}, color={0,0,255}));
  connect(CC.n, B.p)
    annotation (Line(points={{60,20},{90,20}}, color={0,0,255}));
  connect(slewRateLimiter.y, CC.i)
    annotation (Line(points={{41,10},{53,10}}, color={0,0,127}));
  connect(switch1.y, slewRateLimiter.u)
    annotation (Line(points={{7,10},{18,10}}, color={0,0,127}));
  connect(hysteresis.y, switch1.u2)
    annotation (Line(points={{-33,10},{-24,10},{-16,10}}, color={255,0,255}));
  connect(idis.y, switch1.u1) annotation (Line(points={{-33,40},{-24,40},{-24,18},
        {-16,18}}, color={0,0,127}));
  connect(ich.y, switch1.u3) annotation (Line(points={{-33,-20},{-24,-20},{-24,2},
        {-16,2}}, color={0,0,127}));
  connect(vsense.y, hysteresis.u)
    annotation (Line(points={{-69,10},{-56,10}}, color={0,0,127}));
  annotation (Icon(coordinateSystem(preserveAspectRatio=false)), Diagram(
      coordinateSystem(preserveAspectRatio=false), graphics={Rectangle(extent
        ={{-100,50},{100,-40}}, lineColor={255,255,255})}),
    experiment(StopTime=5400, __Dymola_NumberOfIntervals=10000),
    __Dymola_experimentSetupOutput,
    Documentation(info="<html>
        <p>
         This example provides a charge/discharge control logic
         to a current source in parallel with the battery
         model. The control is configured to put the battery
         through charge/discharge cycles for as long as the
         simulation runs:
        </p>


        <div class=\"figure\">
         <p><img src=\"modelica://PVSystems/Resources/Images/
             SimpleBatteryVerificationResults.png\"
```

```
                alt=\"SimpleBatteryVerificationResults.png\" />
          </p>
        </div>

        <p>
          Notice how the charge and discharge cycles take about 30
          minutes, which is what was to be expected by
          charging/discharging a 1A.h battery with a 2A
          current.</p>
      </html>"));
end SimpleBatteryVerification;
```

Examples/Verification/SW1Verification.mo

```
within PVSystems.Examples.Verification;
model SW1Verification "SW1 verification"
  extends Modelica.Icons.Example;
  Electrical.SW1 sw1(fs=1)
    annotation (Placement(transformation(extent={{-40,0},{-20,20}})));
  Modelica.Electrical.Analog.Sources.SineVoltage sineVoltage(freqHz=5, V=1)
    annotation (Placement(transformation(
       origin={30,10},
       extent={{-10,-10},{10,10}},
       rotation=270)));
  Modelica.Electrical.Analog.Basic.Ground ground annotation (Placement(
       transformation(extent={{-10,-30},{10,-10}},rotation=0)));
  Modelica.Electrical.Analog.Basic.Resistor resistor(R=2) annotation (Placement(
       transformation(
       origin={0,30},
       extent={{-10,-10},{10,10}},
       rotation=180)));
  Modelica.Blocks.Sources.RealExpression duty(y=0.5)
    annotation (Placement(transformation(extent={{-70,-30},{-50,-10}})));
equation
  connect(resistor.p,sineVoltage. p)
    annotation (Line(points={{10,30},{30,30},{30,20}}, color={0,0,255}));
  connect(ground.p,sineVoltage. n)
    annotation (Line(points={{0,-10},{30,-10},{30,0}}, color={0,0,255}));
  connect(sw1.p2, resistor.n) annotation (Line(points={{-20,15},{-20,15},{-20,
        30},{-10,30}}, color={0,0,255}));
  connect(sw1.n2, ground.p)
    annotation (Line(points={{-20,5},{-20,-10},{0,-10}}, color={0,0,255}));
  connect(sw1.n1, ground.p)
    annotation (Line(points={{-40,5},{-40,-10},{0,-10}}, color={0,0,255}));
  connect(sw1.p1, resistor.n)
    annotation (Line(points={{-40,15},{-40,30},{-10,30}}, color={0,0,255}));
  connect(duty.y, sw1.d)
    annotation (Line(points={{-49,-20},{-30,-20},{-30,-2}}, color={0,0,127}));
  annotation (experiment(
     StartTime=0,
     StopTime=1,
     Tolerance=1e-3), Diagram(graphics={Rectangle(extent={{-80,40},{48,-30}},
          lineColor={255,255,255})}));
end SW1Verification;
```

Examples/Verification/SW2Verification.mo

```
within PVSystems.Examples.Verification;
model SW2Verification "SW2 verification"
  extends Modelica.Icons.Example;
  Electrical.SW2 sw2(deadTime=0.1, fs=2)
    annotation (Placement(transformation(extent={{-40,0},{-20,20}})));
  Modelica.Electrical.Analog.Sources.SineVoltage sineVoltage(freqHz=5, V=1)
    annotation (Placement(transformation(
       origin={30,10},
       extent={{-10,-10},{10,10}},
       rotation=270)));
  Modelica.Electrical.Analog.Basic.Ground ground annotation (Placement(
       transformation(extent={{-10,-30},{10,-10}},rotation=0)));
  Modelica.Electrical.Analog.Basic.Resistor resistor(R=2) annotation (Placement(
       transformation(
       origin={0,30},
       extent={{-10,-10},{10,10}},
       rotation=180)));
  Modelica.Blocks.Sources.RealExpression duty(y=0.5)
    annotation (Placement(transformation(extent={{-70,-30},{-50,-10}})));
equation
  connect(resistor.p,sineVoltage. p)
    annotation (Line(points={{10,30},{30,30},{30,20}}, color={0,0,255}));
  connect(ground.p,sineVoltage. n)
    annotation (Line(points={{0,-10},{30,-10},{30,0}}, color={0,0,255}));
  connect(sw2.p2, resistor.n) annotation (Line(points={{-20,15},{-20,15},{-20,
        30},{-10,30}}, color={0,0,255}));
  connect(sw2.n2, ground.p)
    annotation (Line(points={{-20,5},{-20,-10},{0,-10}}, color={0,0,255}));
  connect(sw2.n1, ground.p)
    annotation (Line(points={{-40,5},{-40,-10},{0,-10}}, color={0,0,255}));
  connect(sw2.p1, resistor.n)
    annotation (Line(points={{-40,15},{-40,30},{-10,30}}, color={0,0,255}));
  connect(duty.y, sw2.d)
    annotation (Line(points={{-49,-20},{-30,-20},{-30,-2}}, color={0,0,127}));
  annotation (experiment(
     StartTime=0,
     StopTime=1,
     Tolerance=1e-3), Diagram(graphics={Rectangle(extent={{-80,40},{48,-30}},
          lineColor={255,255,255})}));
end SW2Verification;
```

Examples/Verification/SW3Verification.mo

```
within PVSystems.Examples.Verification;
model SW3Verification "SW3 verification"
  extends Modelica.Icons.Example;
  Electrical.SW3 sw3(fs=2, deadTime=0.1)
    annotation (Placement(transformation(extent={{-40,0},{-20,20}})));
  Modelica.Electrical.Analog.Sources.SineVoltage sineVoltage(freqHz=5, V=1)
    annotation (Placement(transformation(
       origin={30,10},
       extent={{-10,-10},{10,10}},
       rotation=270)));
  Modelica.Electrical.Analog.Basic.Ground ground annotation (Placement(
       transformation(extent={{-10,-30},{10,-10}},rotation=0)));
  Modelica.Electrical.Analog.Basic.Resistor resistor(R=2) annotation (Placement(
       transformation(
       origin={0,30},
       extent={{-10,-10},{10,10}},
       rotation=180)));
  Modelica.Blocks.Sources.RealExpression duty(y=0.5)
    annotation (Placement(transformation(extent={{-70,-30},{-50,-10}})));
```

```
equation
  connect(resistor.p,sineVoltage. p)
    annotation (Line(points={{10,30},{30,30},{30,20}}, color={0,0,255}));
  connect(ground.p,sineVoltage. n)
    annotation (Line(points={{0,-10},{30,-10},{30,0}}, color={0,0,255}));
  connect(sw3.n1, ground.p)
    annotation (Line(points={{-40,5},{-40,-10},{0,-10}}, color={0,0,255}));
  connect(sw3.p1, resistor.n)
    annotation (Line(points={{-40,15},{-40,30},{-10,30}}, color={0,0,255}));
  connect(duty.y, sw3.d)
    annotation (Line(points={{-49,-20},{-30,-20},{-30,-2}}, color={0,0,127}));
  connect(sw3.p2, resistor.n)
    annotation (Line(points={{-20,15},{-20,30},{-10,30}}, color={0,0,255}));
  connect(sw3.n2, ground.p)
    annotation (Line(points={{-20,5},{-20,-10},{0,-10}}, color={0,0,255}));
  annotation (experiment(
      StartTime=0,
      StopTime=1,
      Tolerance=1e-3), Diagram(graphics={Rectangle(extent={{-80,40},{48,-30}},
          lineColor={255,255,255})}));
end SW3Verification;
```

Examples/Verification/SwitchingCPMVerification.mo

```
within PVSystems.Examples.Verification;
model SwitchingCPMVerification "SwitchingCPM verification"
  extends Modelica.Icons.Example;
  Control.SwitchingCPM switchingCPM(
    vcMax=5,
    dMin=0.05,
    dMax=0.95,
    fs=200e3,
    Va=0.01) annotation (Placement(transformation(extent={{-20,-14},{0,6}})));
  Modelica.Blocks.Continuous.Integrator integrator(initType=
      Modelica.Blocks.Types.Init.InitialState,
     y_start=3.99,
    k=1e4)
    annotation (Placement(transformation(extent={{50,-10},{70,10}})));
  Modelica.Blocks.Sources.Constant vdT(k=2)
    annotation (Placement(transformation(extent={{-20,30},{0,50}})));
  Modelica.Blocks.Logical.Switch switch1
    annotation (Placement(transformation(extent={{20,-10},{40,10}})));
  Modelica.Blocks.Sources.Constant vdpT(k=-1)
    annotation (Placement(transformation(extent={{-20,-50},{0,-30}})));
  Modelica.Blocks.Sources.Constant vc(k=4)
    annotation (Placement(transformation(extent={{-60,-10},{-40,10}})));
equation
  connect(vdT.y, switch1.u1)
    annotation (Line(points={{1,40},{10,40},{10,8},{18,8}}, color={0,0,127}));
  connect(switch1.y, integrator.u)
    annotation (Line(points={{41,0},{41,0},{48,0}}, color={0,0,127}));
  connect(vdpT.y, switch1.u3) annotation (Line(points={{1,-40},{10,-40},{10,-8},
        {18,-8}}, color={0,0,127}));
  connect(switchingCPM.c, switch1.u2)
    annotation (Line(points={{1,0},{18,0}}, color={255,0,255}));
  connect(integrator.y, switchingCPM.vs) annotation (Line(points={{71,0},{80,0},
        {80,-60},{-30,-60},{-30,-8},{-22,-8}}, color={0,0,127}));
  connect(vc.y, switchingCPM.vc)
    annotation (Line(points={{-39,0},{-22,0}}, color={0,0,127}));
  annotation (experiment(StopTime=0.0002),
    Documentation(info="<html>
      <p>
```

```
          The switching CPM block requires the <em>vs</em> input,
          corresponding to the voltage output of the current
          sensor. In order to simplify things, a switch with some
          constant sources and an integrator are used to emulate
          the behaviour of an inductor. This setup creates the
          conditions to exercise the CPM block, as can be seen in
          the following figure:
        </p>


        <div class=\"figure\">
          <p><img src=\"modelica://PVSystems/Resources/Images/
              SwitchingCPMVerificationresults.Png\"
                alt=\"SwitchingCPMVerificationresults.Png\"
                /></p>
        </div>
      </html>>"),
    Diagram(graphics={Rectangle(extent={{-68,58},{88,-68}}, lineColor={255,255,
          255})}));
end SwitchingCPMVerification;
```

Examples/Verification/SwitchingPWMVerification.mo

```
within PVSystems.Examples.Verification;
model SwitchingPWMVerification "SwitchingPWM verification"
  extends Modelica.Icons.Example;
  Control.SwitchingPWM signalPWM(fs=100)
    annotation (Placement(transformation(extent={{20,0},{40,20}}, rotation=0)));
  Modelica.Blocks.Sources.Step step(
    height=0.3,
    offset=0.2,
    startTime=0.3) annotation (Placement(transformation(extent={{-80,20},{-60,
          40}}, rotation=0)));
  Modelica.Blocks.Sources.Step step1(height=0.3, startTime=0.6) annotation (
      Placement(transformation(extent={{-80,-20},{-60,0}}, rotation=0)));
  Modelica.Blocks.Math.Add add
    annotation (Placement(transformation(extent={{-20,0},{0,20}}, rotation=0)));
equation
  connect(step.y, add.u1) annotation (Line(points={{-59,30},{-40,30},{-40,16},{
        -22,16}}, color={0,0,127}));
  connect(step1.y, add.u2) annotation (Line(points={{-59,-10},{-40,-10},{-40,4},
        {-22,4}}, color={0,0,127}));
  connect(add.y, signalPWM.vc)
    annotation (Line(points={{1,10},{18,10}}, color={0,0,127}));
  annotation (
    Diagram(graphics={Rectangle(extent={{-90,50},{50,-30}}, lineColor={255,255,
          255})}),
    experiment(
      StartTime=0,
      StopTime=1,
      Tolerance=1e-4),
    Documentation(info="<html>
      <p>
        This model provides a changing duty cycle with the use
        of two step blocks. When running the simulation with the
        provided values, plotting the fire output generates the
        following graph:
      </p>


      <div class=\"figure\">
```

```
        <p><img src=\"modelica://PVSystems/Resources/Images/
            SwitchingPWMVerificationResults.png\"
              alt=\"SwitchingPWMVerificationResults.png\" />
        </p>
       </div>

       <p>
        Through inspection of the plot, it can be seen how the
        signal constitutes a PWM signal with a duty cycle
        changing in steps through the values 0.2, 0.5 and
        0.8. Zoom into the signal to confirm this fact as well
        as the value of the period, set at 10 milliseconds.</p>
     </html>"));
end SwitchingPWMVerification;
```

Icons/AssembliesPackage.mo

```
within PVSystems.Icons;
partial class AssembliesPackage "Icon for packages of assemblies"
  extends Modelica.Icons.Package;
  annotation (
    Icon(
      graphics={
        Polygon(
          fillColor={255,255,255},
          fillPattern=FillPattern.Solid,
          points={{-80,60},{-30,60},{-30,60},{-30,10},{-30,10},{10,10},{10,10},
              {20,30},{40,30},{50,0},{40,-30},{20,-30},{10,-10},{10,-10},{-30,-10},
              {-30,-10},{-30,-60},{-30,-60},{-80,-60},{-80,-60},{-80,0},{-80,60},
              {-80,60}},
          lineColor={95,95,95},
          smooth=Smooth.Bezier),
        Polygon(
          points={{-20,60},{-20,60},{80,60},{80,60},{80,-60},{80,-60},{-20,-60},
              {-20,-60},{-20,-20},{-20,-20},{10,-20},{10,-40},{50,-40},{60,0},{
              50,40},{10,40},{10,20},{-20,20},{-20,20},{-20,60}},
          lineColor={95,95,95},
          smooth=Smooth.Bezier,
          fillColor={95,95,95},
          fillPattern=FillPattern.Solid)}),
    Documentation(info="
      <html>
       <p>
        This icon shall be used for a package that contains assemblies of
        components aimed at being used as subsystems of a system
        model.</p>
      </html>"));
end AssembliesPackage;
```

Icons/ConverterIcon.mo

```
within PVSystems.Icons;
partial class ConverterIcon "Icon for power converter models"
  annotation(Icon(graphics={Rectangle(
        extent={{-100,100},{100,-100}},
        lineColor={0,0,255},
        fillColor={255,255,255},
```

```
        fillPattern=FillPattern.Solid),Line(points={{-100,-100},{100,100}},
        color={0,0,255})}));
end ConverterIcon;
```

Icons/package.mo

```
within PVSystems;
package Icons "Library of icons"
extends Modelica.Icons.IconsPackage;


  annotation(Documentation(info="
    <html>
      <p>
        This package contains definitions for the graphical layout of
        components which may be used in different libraries. The icons can
        be utilized by inheriting them in the desired class using
        \"extends\" or by directly copying the \"icon\" layer.</p>
    </html>"));
end Icons;
```

Icons/package.order

```
AssembliesPackage
ConverterIcon
```

package.mo

```
package PVSystems "A Modelica library for photovoltaic system and power converter design"
extends Modelica.Icons.Package;









annotation (
  uses(Modelica(version="3.2.2")),
  preferredView="info",
  version="0.6.3",
  versionDate="2017-09-08",
  Documentation(info="<html>
      <p>
        <b>Overview</b>
      </p>

      <p>
        <b>PVSystems</b> is
        a <a href=\"https://www.modelica.org/\">Modelica</a> library
        providing models useful for the design and evaluation of
        photovoltaic systems and power converters as well as their
        associated control algorithms.
```

```
26        </p>
27
28
29        <div class=\"figure\">
30          <p><img src=\"modelica://PVSystems/Resources/Images/screenshot_small.png\"
31                alt=\"screenshot.png\" />
32          </p>
33        </div>
34
35        <p>
36          The library is the result of a research project carried out in the
37          form of a master's degree thesis. There are two intended audiences
38          for the library:
39        </p>
40
41        <ul class=\"org-ul\">
42          <li><b><b>Users</b></b>: the library is intended to be rich enough
43            in component and subsystem models that it proves useful for
44            those interested in designing and evaluating photovoltaic
45            systems, power converters and their associated control
46            algorithms. Check out the usage section to learn more.
47          </li>
48          <li><b><b>Developers</b></b>: the library is also intended to
49            explore and showcase best practices for the development of
50            Modelica libraries. Many of these best practices are inspired or
51            taken from
52            other <a href=\"https://github.com/raulrpearson?language=modelica&tab=stars\">
                  Modelica
53            libraries on GitHub</a> and from the
54            excellent <a href=\"http://book.xogeny.com/\">Modelica by
55            Example</a>.
56          </li>
57        </ul>
58
59        <p>
60          The library is currently in the early stages of development, so
61          the structure and contents will probably be updated regularly. The
62          intention is to provide models in the following categories:
63        </p>
64
65        <ul class=\"org-ul\">
66          <li><b><b><a href=\"modelica://PVSystems.Control\">Control</a></b></b>:
67            based on the interfaces provided
68            in <a href=\"modelica://Modelica.Blocks\">Modelica.Blocks</a>,
69            common blocks used in the control of power converters, including
70            Park and Clarke transforms, Space Vector Modulation and grid
71            synchronization blocks.
72          </li>
73          <li><b><b><a href=\"modelica://PVSystems.Electrical\">Electrical</a></b></b>:
74            based on the interfaces provided
75            in <a href=\"modelica://Modelica.Electrical.Analog\">Modelica.Electrical.Analog
                  </a>,
76            common electrical models including PV arrays, energy storage
77            devices, power converters, transformers, loads and other grid
78            elements. The library features both switched and averaged models
79            of power converters.
80          </li>
81          <li><b><b><a href=\"modelica://PVSystems.Examples\">Examples</a></b></b>:
82            a comprehensive set of examples will be provided to showcase the
83            capabilities and explain the use of the library.
84          </li>
85        </ul>
86
87        <p>
88          <b>Download and usage</b>
89        </p>
```

```
90
91        <p>
92          You can grab a copy of the library by clonning the repository or
93          downloading
94          a <a href=\"https://github.com/raulrpearson/PVSystems/archive/master.zip\">zip
95          of the latest commit</a>. Take into account that the library is
96          currently in the early stages of development, so some models might
97          not be stable and the structure and contents of the library will
98          probably be updated regularly. If you want to stay up to date with
99          development, you
100         can <a href=\"https://github.com/raulrpearson/PVSystems/subscription\">watch
101         the project</a> if you have a GitHub account or you can subscribe
102         to
103         the <a href=\"https://github.com/raulrpearson/PVSystems/commits/master.atom\">
              commits
104         feed</a>.
105       </p>
106
107       <p>
108         The library can be used inside tools
109         like <a href=\"http://www.3ds.com/products-services/catia/products/dymola/\">
              Dymola</a>
110         or <a href=\"https://openmodelica.org/\">OpenModelica</a> to
111         create models of PV systems. These same tools can be used in
112         conjuntion with other tools supporting
113         the <a href=\"https://fmi-standard.org\">FMI standard</a> for
114         model exchange and co-simulation. For example, a PV system model
115         developed in OpenModelica using this library could then be used to
116         validate a control algorithm developed in MATLAB/Simulink or
117         LabVIEW.
118       </p>
119
120       <p>
121         <b>Contributing</b>
122       </p>
123
124       <p>
125         If you have any <b><b>questions, comments, suggestions, ideas or
126           feature requests</b></b>, please do share those as well as
127           any <b><b>mistakes or bugs</b></b> you might discover. You can
128           open an issue in
129           the <a href=\"https://github.com/raulrpearson/PVSystems/issues\">Issues</a>
130           section of the repository or, if you prefer, contact me
131           by <a href=\"mailto:raul.rodriguez.pearson@gmail.com\">email</a>. Contributions
132           in the form
133           of <a href=\"https://github.com/raulrpearson/PVSystems/pulls\">Pull
134           Requests</a> are always welcome.
135       </p>
136
137       <p>
138         <b>License</b>
139       </p>
140
141       <p>
142         PVSystems is licensed under the MIT
143         License. See <a href=\"modelica://PVSystems.UsersGuide.License\">License</a>
144         for the full license text.
145       </p>
146     </html>"),
147   Icon(graphics={Ellipse(
148       extent={{-70,52},{-10,-6}},
149       pattern=LinePattern.None,
150       lineColor={0,0,0},
151       fillColor={229,184,0},
152       fillPattern=FillPattern.Solid), Polygon(
153       points={{-78,-60},{-42,14},{42,14},{86,-60},{-78,-60}},
```

```
154       fillColor={27,77,130},
155       fillPattern=FillPattern.Solid,
156       pattern=LinePattern.None)}),
157   __Dymola_Commands(file="Resources/Scripts/Dymola/callCheckLibrary.mos"
158      "Run regression tests"));
159 end PVSystems;
```

package.order

```
1 UsersGuide
2 Examples
3 Electrical
4 Control
5 Icons
```

UsersGuide/Contact.mo

```
1  within PVSystems.UsersGuide;
2  class Contact "Contact"
3   extends Modelica.Icons.Contact;
4   annotation (Documentation(info="<html>
5      <p>
6       Copyright (c)
7       2016-2017 <a href=\"mailto:raul.rodriguez.pearson@gmail.com\">úRalí
8       Rodrguez Pearson</a>
9      </p>
10
11     <p>
12      If you have any <strong><strong>questions, comments,
13         suggestions, ideas or feature
14         requests</strong></strong>, please do share those as
15         well as any <strong><strong>mistakes or
16         bugs</strong></strong> you might discover. You can
17         open an issue in
18         the <a href=\"https://github.com/raulrpearson/PVSystems/issues\">Issues</a>
19         section of the repository or, if you prefer, contact
20         the author
21         by <a href=\"mailto:raul.rodriguez.pearson@gmail.com\">email</a>. Contributions
22         in the form
23         of <a href=\"https://github.com/raulrpearson/PVSystems/pulls\">Pull
24         Requests</a> are always welcome.
25     </p>
26
27     <p>
28      The library is the result of a research project carried
29      out in the form of a master's degree thesis under the
30      supervision
31      of <a href=\"http://www.euclides.dia.uned.es/aurquia/index.html\">Dr. Alfonsoí
32      Urqua</a> from
33      the <a href=\"http://www.euclides.dia.uned.es/\">Research
34      group on Modelling &amp;amp; Simulation in Control
35      Engineering</a>, part of
36      the <a href=\"http://www.dia.uned.es/\">Department of
37      Computer Science &amp;amp; Automatic Control</a>
38      at <a href=\"http://www.uned.es/webuned/home.htm\">Universidad
39      Nacional de óEducacin a Distancia</a>. The master's
40      degree is organized in collaboration
```

```
      with <a href=\"http://www.ucm.es\">Universidad
      Complutense de Madrid</a>.</p>
   </html>"));
end Contact;
```

UsersGuide/License.mo

```
within PVSystems.UsersGuide;
class License "License"
 annotation (Documentation(info="<html>
    <p>
     MIT License
    </p>

    <p>
     Copyright (c) 2016-2017 úRal íRodrguez Pearson
    </p>

    <p>
     Permission is hereby granted, free of charge, to any person
     obtaining a copy of this software and associated documentation
     files (the \"Software\"), to deal in the Software without
     restriction, including without limitation the rights to use, copy,
     modify, merge, publish, distribute, sublicense, and/or sell copies
     of the Software, and to permit persons to whom the Software is
     furnished to do so, subject to the following conditions:
    </p>

    <p>
     The above copyright notice and this permission notice shall be
     included in all copies or substantial portions of the Software.
    </p>

    <p>
     THE SOFTWARE IS PROVIDED \"AS IS\", WITHOUT WARRANTY OF ANY KIND,
     EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
     MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
     NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT
     HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
     WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
     OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
     DEALINGS IN THE SOFTWARE.
    </p>
   </html>"), Icon(graphics={ Line(points = {{-60, 60}, {60, 60}}, color = {95, 95, 95}),
        Line(points = {{0, 60}, {0, -60}}, color = {95, 95, 95}), Line(points = {{-60,
       -60}, {60, -60}}, color = {95, 95, 95}), Ellipse(lineColor = {95, 95, 95},
       fillColor = {175, 175, 175},
         fillPattern = FillPattern.Solid, extent = {{-10, 70}, {10, 50}}, endAngle = 360),
             Line(points = {{-90, -10}, {-60, 60}, {-30, -10}}, color = {95, 95, 95}),
            Line(points = {{30, -10}, {60, 60}, {90, -10}}, color = {95, 95, 95}),
      Polygon(
       points={{-90,-10},{-30,-10},{-60,-30},{-90,-10}},
       lineColor={0,0,0},
       fillColor={175,175,175},
       fillPattern=FillPattern.Solid),
      Polygon(
       points={{30,-10},{90,-10},{60,-30},{30,-10}},
       lineColor={0,0,0},
       fillColor={175,175,175},
       fillPattern=FillPattern.Solid)}, coordinateSystem(initialScale = 0.1)));
end License;
```

UsersGuide/package.mo

```
within PVSystems;
package UsersGuide "User's Guide"
extends Modelica.Icons.Information;
annotation (DocumentationClass=true);
end UsersGuide;
```

UsersGuide/package.order

```
References
ReleaseNotes
Contact
License
```

UsersGuide/References/EM01.mo

```
within PVSystems.UsersGuide.References;
class EM01 "<html>R. W. Erickson and D. ćMaksimovi, Fundamentals of Power
       Electronics. Springer Science &amp; Business Media, 2001.</html>"
 extends Modelica.Icons.References;
 annotation (preferredView="info", DocumentationClass=false);
end EM01;
```

UsersGuide/References/EMA16.mo

```
within PVSystems.UsersGuide.References;
class EMA16 "<html>R. W. Erickson, D. ćMaksimovi and K. Afridi,
 <a href=\"https://www.coursera.org/specializations/power-electronics\">
 <i>Power Electronics Specialization</i></a> at Coursera.
 University of Colorado Boulder, 2016.</html>"
 extends Modelica.Icons.References;
 annotation (preferredView="info", DocumentationClass=false);
end EMA16;
```

UsersGuide/References/package.mo

```
within PVSystems.UsersGuide;
package References "References"
extends Modelica.Icons.References;

```

```
6
7
8
9  annotation (preferredView="info");
10 end References;
```

UsersGuide/References/package.order

```
1 EM01
2 EMA16
3 TDD07
4 VGF09
```

UsersGuide/References/TDD07.mo

```
1 within PVSystems.UsersGuide.References;
2 class TDD07 "<html>O. Tremblay, L. A. Dessaint, and A. I. Dekkiche, “A Generic
3        Battery Model for the Dynamic Simulation of Hybrid Electric”
4        Vehicles, in 2007 IEEE Vehicle Power and Propulsion Conference,
5        2007, pp. -284289.</html>"
6   extends Modelica.Icons.References;
7   annotation (preferredView="info", DocumentationClass=false);
8 end TDD07;
```

UsersGuide/References/VGF09.mo

```
1 within PVSystems.UsersGuide.References;
2 class VGF09 "<html>M. G. Villalva, J. R. Gazoli, and E. R. Filho, “Comprehensive
3        Approach to Modeling and Simulation of Photovoltaic ”Arrays,
4        IEEE Transactions on Power Electronics, vol. 24, no. 5,
5        pp. -11981208, May 2009.</html>"
6   extends Modelica.Icons.References;
7   annotation (preferredView="info", DocumentationClass=false);
8 end VGF09;
```

UsersGuide/ReleaseNotes/package.mo

```
1 within PVSystems.UsersGuide;
2 package ReleaseNotes "Release notes"
3 extends Modelica.Icons.ReleaseNotes;
4
5
6
7 annotation (Documentation(info="<html>
8     <p>
9       This section includes an item per release, indicating version
10      number and release date. Release notes are included under each
11      corresponding item.
```

```
12      </p>
13
14      <p>
15        <a href=\"http://semver.org/\">Semantic Versioning</a> is followed
16        to establish version numbers. Given a version number
17        MAJOR.MINOR.PATCH, an increment in the:
18      </p>
19      <ul class=\"org-ul\">
20        <li>MAJOR version indicates incompatible API changes.
21        </li>
22        <li>MINOR version indicates new functionality in a
23          backwards-compatible manner.
24        </li>
25        <li>PATCH version indicates backwards-compatible bug fixes.
26        </li>
27      </ul>
28
29      <p>
30        Notice, though, that major version zero (0.y.z) is for initial
31        development - anything may change at any time and the public API
32        should not be considered stable.</p>
33    </html>"));
34 end ReleaseNotes;
```

UsersGuide/ReleaseNotes/package.order

```
1 Version_0_6_3
2 Version_0_6_2
3 Version_0_6_1
4 Version_0_6_0
```

UsersGuide/ReleaseNotes/Version_0_6_0.mo

```
1  within PVSystems.UsersGuide.ReleaseNotes;
2  class Version_0_6_0 "Version 0.6.0 (April 3, 2017)"
3   extends Modelica.Icons.ReleaseNotes;
4   annotation (Documentation(info="<html>
5      <p>
6        <b>Changes</b>:
7      </p>
8      <ul class=\"org-ul\">
9        <li>The main change in this release is a very heavy refactoring of
10         files. Functionality wise, the library hasn't changed that much,
11         but every model has been split into it's own file.
12       </li>
13       <li>Updated the info text for the root class PVSystems with the
14         contents of the README.md file.
15       </li>
16     </ul>
17     <p>
18       <b>Additions</b>:
19     </p>
20     <ul class=\"org-ul\">
21       <li>Added battery model together with a validation example model.
22       </li>
23       <li>Added User's Guide package with References, Release Notes,
24         Contact and License information.
```

```
      </li>
    </ul>
  </html>"));
end Version_0_6_0;
```

UsersGuide/ReleaseNotes/Version_0_6_1.mo

```
within PVSystems.UsersGuide.ReleaseNotes;
class Version_0_6_1 "Version 0.6.1 (April 19, 2017)"
  extends Modelica.Icons.ReleaseNotes;
  annotation (Documentation(info="<html>
  <p>
    <strong>Modifications</strong>:
  </p>
  <ul class=\"org-ul\">
    <li>The averaged switch models have been expanded. As a start, a
      partial
      model <a href=\"modelica://PVSystems.Electrical.Interfaces.SwitchNetworkInterface\">
          SwitchNetworkInterface</a>
      has been added to provide the common interface.
    </li>
    <li>Many models and blocks (especially blocks) with no icons have
      been given an icon. Things look much better.
    </li>
    <li>The controller assemblies have been moved to
      an <a href=\"modelica://PVSystems.Control.Assemblies\">Assemblies</a>
      package and have been reviewed, cleaned up and some bugs have
      been resolved (in the previous commit, none of them really
      worked).
    </li>
  </ul>
  <p>
    <strong>Additions</strong>:
  </p>
  <ul class=\"org-ul\">
    <li>An <a href=\"modelica://PVSystems.Icons\">Icons</a> package
      has been added to hold icons that can be reused.
    </li>
    <li>Interfaces and Assemblies packages have been added
      to <a href=\"modelica://PVSystems.Electrical\">Electrical</a>
      and <a href=\"modelica://PVSystems.Control\">Control</a> to hold
      partial models and models, respectively, that can be reused.
    </li>
    <li>All of the averaged switch variants
      in <a href=\"modelica://PVSystems.UsersGuide.References.EM01\">EM01</a>
      and <a href=\"modelica://PVSystems.UsersGuide.References.EMA16\">EMA16</a>
      have been added
      (<a href=\"modelica://PVSystems.Electrical.CCM1\">CCM1</a>, <a href=\"modelica://
          PVSystems.Electrical.CCM2\">CCM2</a>, <a href=\"modelica://
          PVSystems.Electrical.CCM3\">CCM3</a>, <a href=\"modelica://
          PVSystems.Electrical.CCM4\">CCM4</a>, <a href=\"modelica://
          PVSystems.Electrical.CCM5\">CCM5</a>, <a href=\"modelica://
          PVSystems.Electrical.CCM_DCM1\">CCM-DCM1</a>
      and <a href=\"modelica://PVSystems.Electrical.CCM_DCM2\">CCM-DCM2</a>).
    </li>
    <li>Additionally some averaged and switched control blocks also
      in <a href=\"modelica://PVSystems.UsersGuide.References.EM01\">EM01</a>
      and <a href=\"modelica://PVSystems.UsersGuide.References.EMA16\">EMA16</a>
      have also been added
      (<a href=\"modelica://PVSystems.Control.SwitchingCPM\">SwitchingCPM</a>, <a href=\"
          modelica://PVSystems.Control.CPM_CCM\">CPM-CCM</a>, <a href=\"modelica://
          PVSystems.Control.CPM\">CPM</a>).
```

```
48     </li>
49     <li>A <a href=\"modelica://PVSystems.Control.DeadTime\">DeadTime</a>
50       block has been added to be used in conjunction with any of the
51       blocks producing switching signals
52       (currently <a href=\"modelica://PVSystems.Control.SignalPWM\">SignalPWM</a>
53       and <a href=\"modelica://PVSystems.Control.SwitchingCPM\">SwitchingCPM</a>),
54       to create a complement switching signal with an optional dead
55       time value.
56     </li>
57   </ul>
58   <p>
59     <strong>Deletions</strong>:
60   </p>
61   <ul class=\"org-ul\">
62     <li>The model Ideal2LevelLeg has been removed since it added
63       complexity and didn't seem to be that useful.
64     </li>
65   </ul>
66 </html>"));
67 end Version_0_6_1;
```

UsersGuide/ReleaseNotes/Version_0_6_2.mo

```
1  within PVSystems.UsersGuide.ReleaseNotes;
2  class Version_0_6_2 "Version 0.6.2 (July 6, 2017)"
3    extends Modelica.Icons.ReleaseNotes;
4    annotation(Documentation(info="<html>
5      <p>
6        <strong>Modifications</strong>:
7      </p>
8      <ul class=\"org-ul\">
9        <li>The main new feature of this release is the slight
10         restructuring of switch network models that now have
11         switching, as well as averaged, variants. This
12         provides generic converter blocks that can be changed
13         from switched to averaged with the click of a mouse.
14       </li>
15       <li>The SignalPWM block is renamed to SwitchingPWM and
16         its icon is changed.
17       </li>
18       <li>Documentation for examples is updated and
19         completed. The names of components used in some of the
20         examples are changed.
21       </li>
22       <li>The Validation package is renamed to
23         Verification. This name change has also been applied
24         to models inside this package and associated files
25         like results plots.
26       </li>
27       <li>MPPTController is taken out of Control.Assemblies
28         and placed directly in the Control package.
29       </li>
30       <li>Instances of switch networks are made replaceable to
31         allow the use of different flavours of converters and
32         circuits (switched and averaged).
33       </li>
34       <li>HBridgeAveraged is renamed to HBridge since it can
35         now be instantiated with switched or averaged models.
36       </li>
37       <li>The averaged models (CCM1, CCM2&#x2026;) include the
38         port current equations that were previously part of
39         the TwoPortConverter interface, since they now inherit
```

```
        from TwoPort, which doesn't include those equations.
      </li>
    </ul>
    <p>
      <strong>Additions</strong>:
    </p>
    <ul class=\"org-ul\">
      <li>Three variants of switched switch network models,
        SW1 (switch and diode), SW2 (2 switches) and SW3 (2
        switches + antiparallel diode).
      </li>
      <li>The script callCheckLibrary.mos has been added to
        provide a convenient way to run regression tests. To
        provide even more convenience, a
        _<sub>Dymola</sub><sub>Commands</sub> annotation has
        been added to the root package file so that this
        script can be run from the Commands menu when using
        Dymola. The reference regression tests should be
        updated in each release if necessary.
      </li>
      <li>Added a script to export listings for publishing.
      </li>
      <li>TwoPort interface and ConverterIcon added in place
        of TwoPortConverter.
      </li>
    </ul>
    <p>
      <strong>Deletions</strong>:
    </p>
    <ul class=\"org-ul\">
      <li>TwoPortConverter interface is removed and
        substituted by a TwoPort interface (copied from MSL
        without current equations) and a ConverterIcon.
      </li>
    </ul>
  </html>"));
end Version_0_6_2;
```

UsersGuide/ReleaseNotes/Version_0_6_3.mo

```
within PVSystems.UsersGuide.ReleaseNotes;
class Version_0_6_3 "Version 0.6.3 (September 8, 2017)"
  extends Modelica.Icons.ReleaseNotes;
  annotation(Documentation(info="<html>
    <p>
      <strong>Fixes and modifications</strong>:
    </p>
    <ul class=\"org-ul\">
      <li>USBBatteryConverter, which instantiates
        CPMBidirectionalBuckBoost couldn't run the simulation
        through a mode change (from buck to boost or
        viceversa). CPMBidirectionalBuckBoost now incorporates
        one on delay block for each mode enable to prevent
        them from being active at the same time. This allows
        for transitions to be simulated, without loss of
        accuracy.
      </li>
      <li>CCM4 was missing the term for forward voltage drop
        loss. This release adds that.
      </li>
      <li>CCM_DCM1 and CCM_DCM2 had an incorrect calculation
        of mu which could provide negative values. This was
```

```
23          fixed. Additionally, CCM_DCM2 was missing an 'n' term
24          in the calculation of Re.
25        </li>
26        <li>CPM had some missing terms in the equation for d.
27        </li>
28      </ul>
29      <p>
30        <strong>Additions</strong>:
31      </p>
32      <ul class=\"org-ul\">
33        <li>Verification models were added in the corresponding
34          examples package for SW1, SW2, SW3, CCM1, CCM2, CCM3,
35          CCM4, CCM5, CCM_DCM1, CCM_DCM2, CPM_CCM, CPM and
36          DeadTime.
37        </li>
38      </ul>
39      <p>
40        <strong>Deletions</strong>:
41      </p>
42      <ul class=\"org-ul\">
43        <li>No deletions.
44        </li>
45      </ul>
46    </html>"));
47  end Version_0_6_3;
```