Torrey D.A., Selamogullari U.S.:
A Behavioral Model for DC-DC Converter using Modelica
2nd International Modelica Conference, Proceedings, pp. 167-172

Paper presented at the 2nd International Modelica Conference, March 18-19, 2002,
Deutsches Zentrum für Luft- und Raumfahrt e.V. (DLR), Oberpfaffenhofen, Germany.

All papers of this workshop can be downloaded from
http://www.Modelica.org/Conference2002/papers.shtml

A Behavioral Model for DC-DC Converters using Modelica

David A. Torrey Ugur Savas Selamogullari
Department of Electrical, Computer and Systems Engineering
Rensselaer Polytechnic Institute
Troy, NY 12180-3590 USA
Email: torred@rpi.edu, selamu@rpi.edu

Abstract

This paper describes the development of a behavioral model of a dc/dc converter. The focus is on developing a model that simulates quickly, yet retains the behavioral features of a physical converter. Based on a physically motivated behavioral circuit model, the model is then implemented in the Modelica modeling language and simulated using the Dymola simulation environment. Simulation results are given. A theoretical method for tuning the simulation model to a physical converter is presented.

1 Introduction

Dc/dc converters are power electronic circuits that convert a dc voltage to a different regulated dc voltage level. In this respect, ideally a dc/dc converter can be considered as a dc transformer that provides lossless transfer of energy between circuits at different voltage or current levels. There are several topological variations of dc converters; the converter is generally described in terms of its voltage conversion characteristics. For example, a Buck Converter generally produces an output voltage that is lower than the input voltage.

The main objective of the dc/dc converter is to control one or more power semiconductor switches to transfrom dc input from one level to another. This is usually accomplished by controlling the on and off durations of the semiconductors; filters are then used to remove the associated ac components of the input current and the output voltage [1, 2, 3, 4].

Our objective here is in describing the macroscopic dynamic behavior of the dc/dc converter without getting caught up in the control and switching operations taking place within a physical converter. We are trying to create a simulation model that faithfully emulates the behavior of a commercial dc/dc converter. The next section motivates the behavioral model.

2 The Behavioral Model

To model the true behavior of a dc/dc converter, a detailed model of the converter and its controller has to be built where every switching cycle is taken into account. However, the goal of our modeling effort is to emulate the behavior of a commercial dc/dc converter where the dynamics at the switching frequency are barely perceptible at the two ports of the converter. Modeling the cycle to cycle operation within the converter merely adds to the execution time of the model and potentially introduces numerical issues.

A more efficient way to simulate the behavior of a dc/dc converter is to use a circuit model that produces dynamic voltages and currents, but without considering internal converter quantities on an instantaneous or averaged basis [4]. The equivalent circuit contains no switching or switching ripple, and only the important macroscopic components of the waveforms are modeled [2]. With this approach, a behavioral model (input current, output voltage changes) of a dc/dc converter is obtained. The circuit and Modelica implementation are shown in Fig. 1 and Fig. 2, respectively.

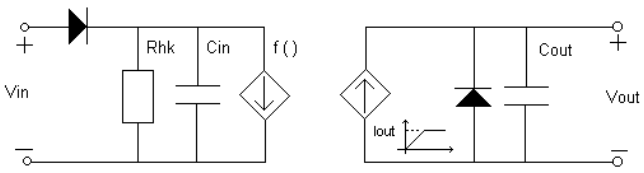

Figure 1: A behavioral model for a dc/dc converter.

A reflected load current is used on the input side of the converter. Any change in output voltage, load, and input voltage is reflected in this current since these are

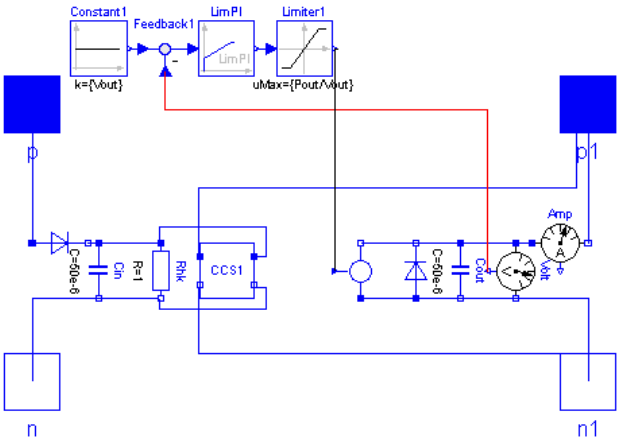

Figure 2: A schematic of the dc/dc converter behavioral model.

variables of the current equation $f(v_{\text{in}}, v_{\text{out}}, i_{\text{out}})$. At the output stage, a controlled current source is used to simulate the output current supplied by the converter. This current is controlled to regulate the output voltage. The model emulates the current limiting function of a practical dc/dc converter design that uses current-mode control. In order to control the output current, output voltage feedback is used. This is exactly what would be done in a physical converter. The output voltage is compared with the nominal voltage of the converter (V_{ref}) and an error is fed to a proportional plus integral (PI) controller. The output of this controller is used to drive a limiter with a limit value of $I_{\text{out}} = P_{\text{out}}/V_{\text{ref}}$. The output of the limiter drives a signal current source and defines the current drawn by the load.

On the input side there is a resistor (R_{hk}) to account for internal power needs of the converter. The power consumed by this resistor models the internal power consumed by the control circuitry; the power lost to switching, conduction and other parasitic losses within the converter are reflected in the efficiency used to determine the load current reflected to the input. The input capacitor is to provide energy storage for the input side and to simulate the input capacitance of a typical dc/dc converter. The output of the converter is only active if the input voltage is in within a valid range of input voltages $[V_{\text{in}_{\text{min}}}, V_{\text{in}_{\text{max}}}]$. Ideal diodes are used to restrict the flow of energy to be from the input to the output.

The model uses existing Modelica Library components such as resistors, ideal diodes, capacitors and two new components: a controlled current source (CCS) and a PI controller with limiting.

2.1 Controlled Current Source (CCS)

This component is the redefinition of existing Voltage Controlled Current Source within the Modelica Electrical Library with a new current definition equation and the addition of extra parameters. The relationship between input and output power of any dc-dc converter can be written as

$$P_{\text{in}} = \frac{P_{\text{out}}}{\eta} \quad ; \tag{1}$$

$$V_{\text{in}} \cdot I_{\text{in}} = \frac{V_{\text{out}} \cdot I_{\text{out}}}{\eta} \quad . \tag{2}$$

The load current reflected to the input is

$$I_{\text{in}} = \frac{V_{\text{out}} \cdot I_{\text{out}}}{\eta \cdot V_{\text{in}}} \quad . \tag{3}$$

In the model, V_{out} is the output voltage, I_{out} is the load current measured through the current sensor, η is efficiency of the converter and V_{in} is the input voltage.

The Modelica code of this component is given below. Since the output voltage is already used in the current equation and there will be no output voltage when input voltage is outside of range, parameters $V_{\text{in}_{\text{min}}}$ and $V_{\text{in}_{\text{max}}}$ are not used in this component. The current equation i_2 is used under the complete model equation section due to dependency on outside variables, which are not defined inside the class itself. The Modelica code given below is complete with i_2 in it. When i_2 is used outside of the *class CCS*, the class becomes a partial class.

```
class CCS

extends Electrical.Analog.Interfaces.TwoPort;
parameter Real Vout;
parameter Real Pout;
parameter Real eff;

equation

i1 = 0;
i2 = v1*(CurrentSensor.i)/((v2 + 1e-10)*eff);
end CCS;
```

2.2 PI Controller

This component is the combination of two standard Modelica library components: Gain and LimIntegrator. These two components are connected in parallel to form the proportional plus integral (PI) controller. The error signal is fed to the input of the PI controller and controller output is one that is proportional to both

magnitude and the integral of the input signal. Proportional control increases the speed of the response while using a term proportional to the integral of the error signal eliminates the steady state error.

Component LimIntegrator provides the option of turning the integrator off when the integral reaches a given upper or lower limit. This is used to prevent integrator wind-up. This way, the overall system has less overshoot and uses less control effort. The Modelica implementation of this component follows.

```
class LimPICont

parameter Real Pout;
parameter Real Vref;
Blocks.Continuous.LimIntegrator
LimIntegrator1(outMax[:]={Pout/Vref});
Blocks.Math.Add Add1;
Blocks.Math.Gain Gain1;
Blocks.Interfaces.InPort inPort;
Blocks.Interfaces.OutPort outPort;

equation

connect(LimIntegrator1.inPort,inPort);
connect(Gain1.inPort,inPort);
connect(Gain1.outPort, Add1.inPort1);
connect(LimIntegrator1.outPort,Add1.
        inPort2);
connect(Add1.outPort,outPort);

end LimPICont;
```

A theoretical method for determining the PI controller parameters is explained in the next section.

3 Determining PI Controller Parameters

The following method can be used to determine the PI controller parameters from a physical converter. A ripple is introduced on the output voltage by using an offset sinusiodal voltage source in series with a resistor as shown in Fig. 3. Using the sinusoidal voltage source provides control over the ripple frequency.

The capacitor voltage and current can be calculated from the circuit equations once the resistor voltage and current are known. These two currents add up to the dc/dc converter output current:

$$i_{\text{out}} = i_{\text{r}} + i_{\text{c}} \quad . \tag{4}$$

The output current can also be written as:

$$i_{\text{out}} = H(s)(V_{\text{ref}} - V_{\text{out}}) \quad , \tag{5}$$

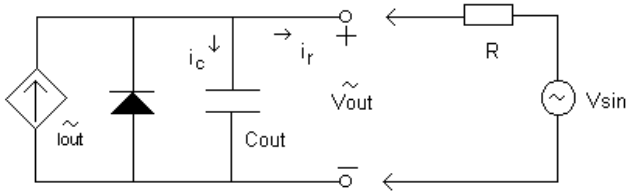

Figure 3: Circuit for PI parameter extraction.

$$H(s) = K_{\text{p}} + \frac{K_{\text{i}}}{s} \quad , \tag{6}$$

where $H(s)$ is the transfer function of the PI controller, V_{ref} is nominal output voltage of the converter and V_{out} is the output voltage. If only ripple quantities are considered, then

$$\tilde{i}_{\text{out}_\text{d}} = \tilde{i}_{\text{r}_\text{d}} + \tilde{i}_{\text{c}_\text{d}} \quad , \tag{7}$$

and

$$\tilde{i}_{\text{out}_\text{d}} = H(s) \cdot \tilde{v}_{\text{out}_\text{d}} \quad . \tag{8}$$

$\tilde{v}_{\text{out}_\text{d}}$ is the ripple voltage across the output capacitor. Since both $\tilde{i}_{\text{out}_\text{d}}$ and $\tilde{v}_{\text{out}_\text{d}}$ are known, PI controller parameters K_{p} and K_{i} can be calculated.

For high frequency values, the integrator effect of the PI controller is minimized and the proportional effect will be dominant. Under these conditions

$$\tilde{i}_{\text{out}_\text{d}} = K_{\text{p}} \cdot \tilde{v}_{\text{out}_\text{d}} \quad , \tag{9}$$

so

$$K_{\text{p}} = \frac{\tilde{i}_{\text{out}_\text{d}}}{\tilde{v}_{\text{out}_\text{d}}} \quad . \tag{10}$$

For low frequencies, the intgerator will be donimant, so

$$\tilde{i}_{\text{out}_\text{d}} = \frac{K_{\text{i}}}{s} \cdot \tilde{v}_{\text{out}_\text{d}} \quad , \tag{11}$$

$$= K_{\text{i}} \int \tilde{v}_{\text{out}_\text{d}}\, dt \quad . \tag{12}$$

The values found from this method are an approximation and give a starting point for final values. Test results using a physical converter can be used for fine-tuning of the simulation model.

4 Simulation Results

Simulations have been completed for $10\,\Omega$ and $20\,\Omega$ loading. For each simulation the following waveforms are plotted:

- Input current.
- Output voltage and current.

Simulation parameters and their values are:

- $V_{\text{in}_{\text{min}}} = 145\,\text{V}$
- $V_{\text{in}_{\text{max}}} = 208\,\text{V}$
- $V_{\text{out}} = 300\,\text{V}$
- $P_{\text{out}} = 5000\,\text{W}$
- $\eta = 0.95$
- LimIntegrator gain = 1 (actual gain would be determined from the test explained in Section 3)
- Gain = 1 (actual gain would be determined from the test explained in Section 3)
- $C_{\text{out}} = C_{\text{in}} = 50\mu\text{F}$

In order to show the effect of input voltage range on converter operation, a sinusoidal voltage source is placed in series with a constant voltage source to form the input voltage (see Fig. 4). Both $10\,\Omega$ and $20\,\Omega$ loading simulations are run for this case as well. For

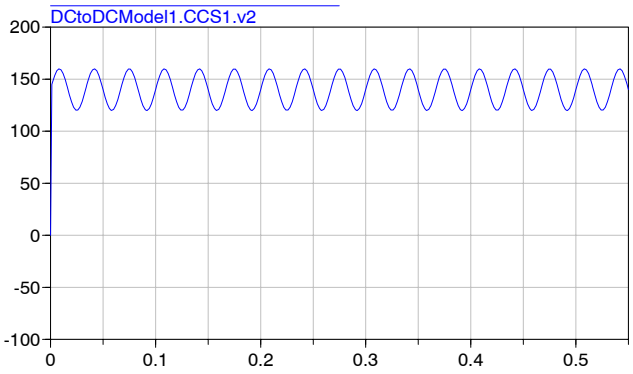

Figure 4: The time-varying input voltage.

the constant input voltage case, the simulation results for $10\,\Omega$ loading are shown in Fig. 5 and Fig. 6. The output current limit can be seen in Fig. 5.

For the time-varying input voltage case, the simulation results for $20\,\Omega$ loading are shown in Fig. 7 and Fig. 8.

The input current waveform in Fig. 8 for the time-varying input voltage case can be explained by considering its equation:

$$I_{\text{in}} = \frac{V_{\text{out}} \cdot I_{\text{out}}}{\eta \cdot V_{\text{in}}} \quad . \tag{13}$$

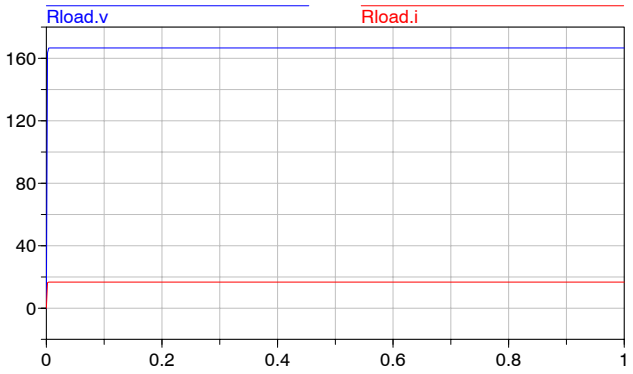

Figure 5: The output voltage and current waveforms for constant input voltage.

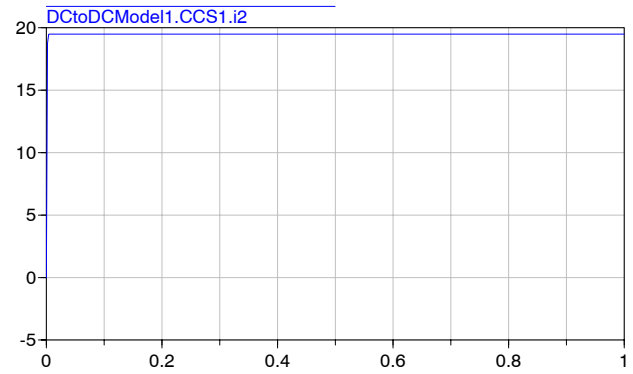

Figure 6: Input current waveform for the constant input voltage.

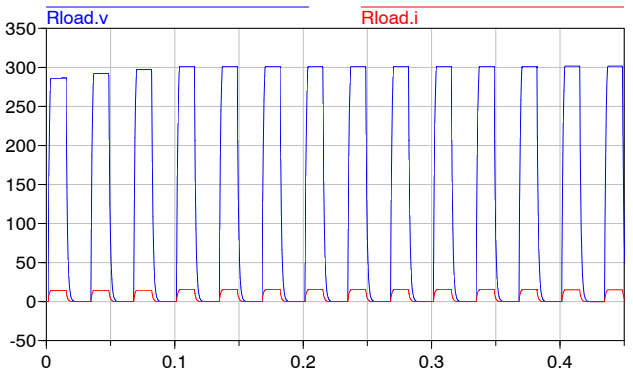

Figure 7: The output voltage and current waveforms for time-varying input voltage.

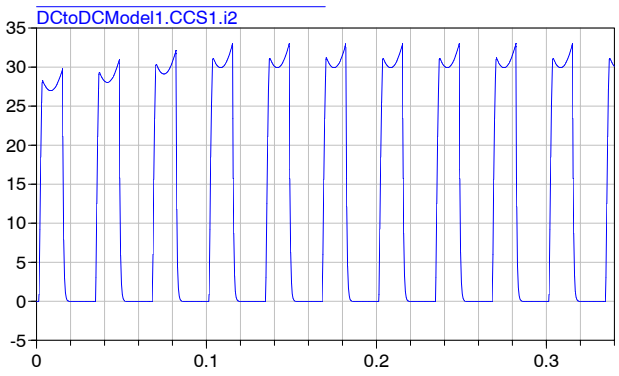

Figure 8: The input current waveform for the time-varying input voltage.

The output power has a constant value and the input voltage varies sinusoidally. Thus, the current value is found by division of a constant with a sinusoid. This explains why the input current decreases as the input voltage increases and vice versa.

5 Conclusion

This paper has described the development of a behavioral model for dc/dc converters. The model has been developed by using the modeling language Modelica. Two new Modelica components are developed from the existing libary components. Model itself is also a new addition to Modelica Electrical Library. Simulation results for 10 Ω and 20 Ω loading are given. Information about the tuning of PI controller parameters to a physical converter is explained.

References

[1] J. G. Kassakian, G. C. Verghese, and M. F. Schlecht, *Principles of Power Electronics*, Addison-Wesley, 1991.

[2] P. T. Krein, *Elements of Power Electronics*, Oxford University Press, 1998.

[3] N. Mohan, T. M. Undeland, W. P. Robbins, *Power Electronics*, 2nd Edition, John Wiley and Sons Inc., 1995.

[4] D. W. Hart, *Introduction to Power Electronics*, Prentice Hall Inc., 1997.

[5] Modelica Web Page (www.modelica.org)

[6] Dymola User's Manual

[7] Modelica Standard Electrical Library

A Modelica Code of DC-DC Converter Model

```
package DCModel

model DCtoDCModel "DC/DC Converter Model"

parameter Real Pout;
parameter Real Vref;
parameter Real Vinmin;
parameter Real Vinmax;
parameter Real eff;
Real Iout;

model SignalCurrent
"Generic current source using the input signal as
source current"

extends Electrical.Analog.Interfaces.OnePort;
Blocks.Interfaces.InPort inPort(final n=1) ;

end SignalCurrent;

class LimPICont

parameter Real Pout;
parameter Real Vref;
Blocks.Continuous.LimIntegrator
LimIntegrator1(outMax[:]={Pout/Vref}) ;
Blocks.Math.Add Add1 ;
Blocks.Math.Gain Gain1 ;
Blocks.Interfaces.InPort inPort ;
Blocks.Interfaces.OutPort outPort ;

equation

connect(LimIntegrator1.inPort, inPort) ;
connect(Gain1.inPort, inPort) ;
connect(Gain1.outPort, Add1.inPort1) ;
connect(LimIntegrator1.outPort, Add1.inPort2) ;
connect(Add1.outPort, outPort) ;

end LimPICont;

partial class CCS

extends Electrical.Analog.Interfaces.TwoPort;

equation

i1 = 0;

end CCS;

CCS CCS1 ;
Electrical.Analog.Basic.Resistor Rhk ;
Electrical.Analog.Basic.Capacitor Cout(C=50e-6) ;
Electrical.Analog.Ideal.IdealDiode Din ;
Electrical.Analog.Basic.Capacitor Cin(C=50e-6) ;
Electrical.Analog.Interfaces.PositivePin p ;
Electrical.Analog.Interfaces.NegativePin n ;
Electrical.Analog.Interfaces.PositivePin p1 ;
Electrical.Analog.Interfaces.NegativePin n1 ;
Electrical.Analog.Sensors.CurrentSensor Amp ;
Blocks.Math.Feedback Feedback1 ;
Blocks.Sources.Constant Constant1(k={Vout}) ;
Electrical.Analog.Sensors.VoltageSensor Volt ;
Blocks.Nonlinear.Limiter Limiter1(uMax={Pout/Vout}, uMin={0});
LimPICont LimPI(Pout=Pout, Vref=Vref) ;
Electrical.Analog.Ideal.IdealDiode Dout ;
SignalCurrent SignalCurrent1 ;

equation

connect(Volt.p, Amp.p) ;
connect(Din.n, Cin.p) ;
connect(Din.p, p) ;
connect(Cin.n, n) ;
connect(Rhk.p, Cin.p) ;
connect(Cin.n, Rhk.n) ;
connect(CCS1.p2, Rhk.p) ;
connect(CCS1.n2, Rhk.n) ;
connect(CCS1.p1, p1) ;
connect(CCS1.n1, n1) ;
connect(Cout.p, Volt.p) ;
connect(Cout.n, Volt.n) ;
connect(Volt.n, n1) ;
connect(Amp.n, p1) ;
connect(Constant1.outPort, Feedback1.inPort1) ;
connect(Feedback1.outPort, LimPI.inPort) ;
connect(LimPI.outPort, Limiter1.inPort) ;
connect(Volt.outPort, Feedback1.inPort2) ;
connect(Limiter1.outPort, SignalCurrent1.inPort) ;
connect(Dout.n, SignalCurrent1.n) ;
connect(SignalCurrent1.p, Dout.p) ;
connect(Dout.p, Cout.n) ;
connect(Dout.n, Cout.p) ;

Iout = Pout/Vref;

CCS1.i2 = (CCS1.v1)*(Amp.i)/((CCS1.v2 + 1e-10)*eff);

SignalCurrent1.i = if p.v > Vinmin and p.v < Vinmax then
SignalCurrent1.inPort.signal[1]
                  else 0;

end DCtoDCModel;

end DCModel;
```