Missouri University of Science and Technology
Scholars' Mine

Electrical and Computer Engineering Faculty Research & Creative Works

Electrical and Computer Engineering

01 Jul 2006

An Efficient Discontinuous-Mode Model of a Switch Pole

Jonathan W. Kimball
Missouri University of Science and Technology, kimballjw@mst.edu

Follow this and additional works at: https://scholarsmine.mst.edu/ele_comeng_facwork

 Part of the Electrical and Computer Engineering Commons

Recommended Citation

J. W. Kimball, "An Efficient Discontinuous-Mode Model of a Switch Pole," *Proceedings of the IEEE Workshop on Computers in Power Electronics (2006, Troy, NY)*, pp. 260-263, Institute of Electrical and Electronics Engineers (IEEE), Jul 2006.
The definitive version is available at https://doi.org/10.1109/COMPEL.2006.305641

This Article - Conference proceedings is brought to you for free and open access by Scholars' Mine. It has been accepted for inclusion in Electrical and Computer Engineering Faculty Research & Creative Works by an authorized administrator of Scholars' Mine. This work is protected by U. S. Copyright Law. Unauthorized use including reproduction for redistribution requires the permission of the copyright holder. For more information, please contact scholarsmine@mst.edu.

An Efficient Discontinuous-Mode Model of a Switch Pole

Jonathan W. Kimball, *Senior Member*

Grainger Center for Electric Machinery and Electromechanics
University of Illinois at Urbana-Champaign
1406 W. Green St., Urbana, IL 61801 USA

***Abstract* – Simulating switching power supplies presents many challenges. A variety of switch pole models is available, from the physics-based to the behavioral. The present work develops and demonstrates a behavioral model that works well in discontinuous mode. The new model eliminates the extremely fast time constants normally associated with switches in a high impedance state. Simulation time is improved and fixed-time-step algorithms are now stable with reasonable step size.**

I. Introduction

Switching power converters are stiff systems that present special challenges for simulation. In many stiff systems, the fast dynamics decay quickly and then the remainder of the simulation involves only slow dynamics. In a switching power converter, though, the fast dynamics are re-initialized every switching cycle. Simulating a few seconds may result in hundreds of thousands, or even millions, of switching cycles.

Fast dynamics are particularly prevalent when the converter enters discontinuous conduction mode (DCM). In this situation, the inductor is first connected to a voltage source through a resistance on the order of Ω or mΩ (a switch or diode that is "on"). When the current reaches zero, the diode turns "off." When using a bilinear resistor model, which is typical in commercial packages like Dymola [1] or SIMPLORER, a diode that is "off" is modeled as a resistor whose value is in the MΩ range. The effective L/R time constant decreases by up to ten orders of magnitude. If the integration algorithm used by the simulator is not properly designed, the enormous change can result in an unstable simulation.

The simulation may not appropriately capture real-world dynamics. For example, the simulation may be unstable while the real system is well behaved. Current overshoot or undershoot may be present in the simulation but not in a practical circuit. Expertise in simulation and in the laboratory is needed to sort the real phenomena from those created by poor modeling or simulation.

Many behavioral models have been proposed. In [2, 3], a capacitor (possibly nonlinear) is added in parallel with the controlled switch or switches. This captures some important dynamics associated with switching edges, but often at a level of detail that does not interest the average designer. In [4, 5], extra voltage sources and current sources are switched into the circuit to model energy loss. These approaches are extremely useful if power dissipation needs to be estimated, but they can still include fast dynamics if the system ever enters DCM.

The present work proposes a new behavioral model in which fast dynamics are completely eliminated. Some of the capacitive effects of a real system are neglected, but the simulation results are still useful for exploring overall performance of the converter. Simulation results are greatly improved for fixed-time-step algorithms and are somewhat improved for variable-time-step algorithms. The equivalent four-terminal switching element is shown in Fig. 1. One terminal is Boolean-valued and determines the state of the controlled switch. An inductor with non-zero resistance is included within the switch pole model. As discussed in [6, 7], most converters can be constructed with such a switch pole or its inverse (swapping the controlled switch and diode locations).

II. Continuous Conduction Mode

Synchronous converters are extremely easy to simulate. In a synchronous converter, DCM is impossible. When the main controlled switch is turned off, a diode commutates the current, but then another controlled switch in parallel with the diode is turned on. The inductor is always connected to a known potential through a low resistance. This is also termed "synchronous rectification."

The easiest way to model a synchronous switch pole is with

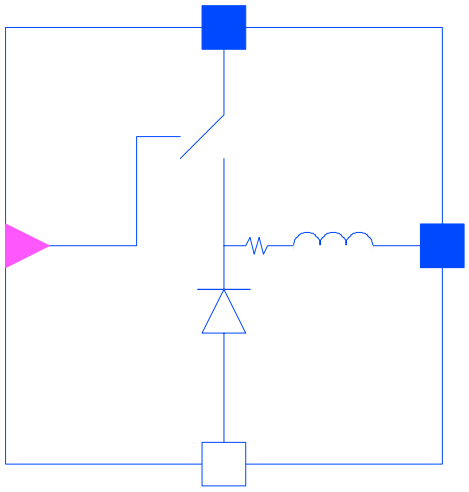


Fig. 1. Equivalent circuit element.

0-7803-9725-8/06/$20.00 ©2006 IEEE.

a single-pole double-throw (SPDT) switch. Modelica [8] code for a simple lossless model is given in Appendix A. Annotations have been removed. Some straightforward external logic can be used to model deadtime in, for example, an ac inverter.

Most converters, though, do not include synchronous rectification. As long as the current is continuous, a standard converter will behave the same as a synchronous converter. Both bilinear resistor models and the new model proposed below behave similarly. An example simulation of the new model is shown in Fig. 2.

III. Discontinuous Conduction Mode

If converters always operated in continuous conduction mode, bilinear resistor models would be sufficient, despite some technical drawbacks. The primary difficulty with a bilinear model comes when inductor current goes to zero. For a diode model to approximate a real diode, the off-state resistance must be between 100 kΩ and 10 MΩ, depending on the voltages involved. With inductance values on the order of 100 µH, resulting time constants are at or below 1 ns. Fixed-time-step algorithms, such as the typical implementations of Euler's method or Runge-Kutta methods, would need a time step on the order of 100 ps to adequately capture the resulting dynamics.

Clearly, using a fixed time step of 100 ps for a 1 s total simulation time would require an excessive amount of computation time and data memory. More importantly, a short time step does not really add any useful information. Suppose the converter involved is switching at 50 kHz. A 100 ps time step would result in 200,000 simulation intervals per switching cycle. In most situations, about 20 properly-chosen points per switching interval give enough information. This observation motivates variable-time-step algorithms, in which the time step is adjusted depending on the derivative of the state variables. Near each switching edge, a time step of 100 ps is still necessary, but after the fast transients decay, a more appropriate time step like 100 ns can be used.

What if the 1 ns time constants could be eliminated altogether? Then the slower time step could be used throughout the simulation, with one or two extra points at each switching edge to eliminate time quantization effects. The resulting improvement in simulation speed could be substantial.

To eliminate fast time constants, resistances on the order of MΩ cannot be included. In the new model, resistance is allowed to go to infinity (conductance goes to zero). The basic principle is that the converter has three possible states:

1. Controlled switch "on" (order of mΩ).
2. Diode "on" (some voltage drop plus resistance on order of mΩ).
3. Both switches off (zero conductance), inductor terminals shorted together.

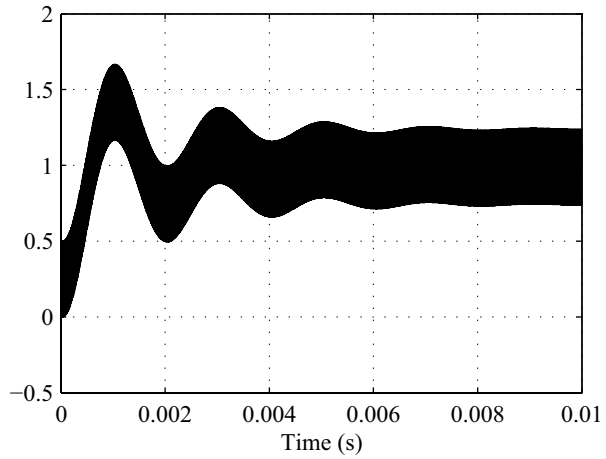


Fig. 2. Continuous conduction mode simulation with new model (inductor current shown).

The third state is the key new contribution. In DCM, inductor current goes to zero and remains at zero until the controlled switch turns back on. This translates into $di/dt = 0$, or equivalently, the inductor voltage is zero. Zero voltage is equivalent to a short circuit. So, the new model simply shorts the inductor during the time when the current is zero. The presence of non-zero resistance prevents a reduction in the order of the system. Modelica code (without annotations) is given in Appendix B.

The Modelica model was simulated in Dymola. The simulated circuit is shown in Fig. 3. The inductor is 100 µH with a parasitic resistance of 0.1 Ω. Fig. 4 compares a bilinear resistor model to the new model. The results are effectively the same because the integration method was DASSL [9]. DASSL is an extremely robust method for simulating stiff systems and can be used for differential algebraic equations (DAEs).

IV. Comparisons Between Methods

There are two goals for the new model. First, the simulation

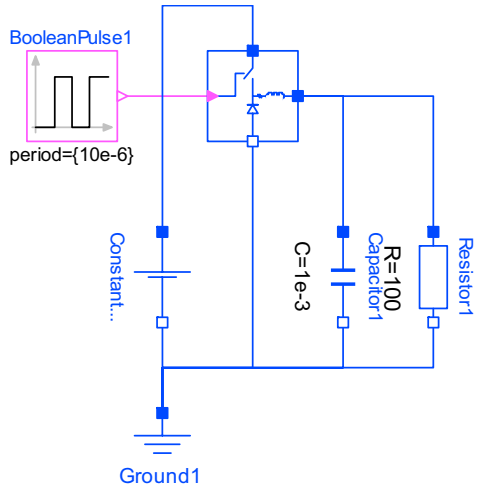


Fig. 3. Circuit for simulation.

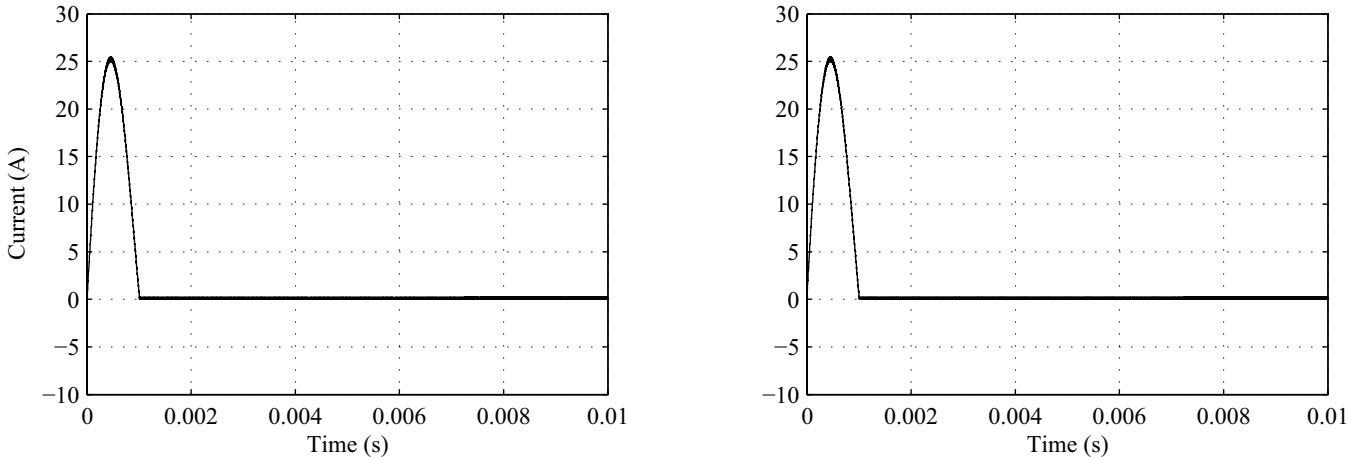


Fig. 4. Discontinuous mode simulation using DASSL. (a) Model using bilinear resistors. (b) New model.

results should be more reliable even when using inferior integration algorithms. Second, simulation speed should be greatly improved.

Dymola includes several fixed-time-step integration algorithms. Fig. 5 shows the new model in the circuit of Fig. 3 simulated with Euler's method. The time step was set at 300 ns. Obviously, the results are not as accurate as with DASSL, but the waveform adequately captures large-signal behavior.

A variety of integration algorithms are compared in Table I. Detecting DCM involves root-finding, a feature available in a minority of integration methods that are built into Dymola. Table I compares all of the relevant algorithms. LSODAR [10] is generally competitive with DASSL, although without support for DAEs. ODASSL is a variant of DASSL that supports overdetermined DAEs. The implementation of the Runge-Kutta method in Dymola does not include root-finding, but will simulate event-based systems anyway and act on events on the next fixed time step. The simulations were performed on a 2.5 GHz Pentium 4 with 1 GB of RAM running Windows XP.

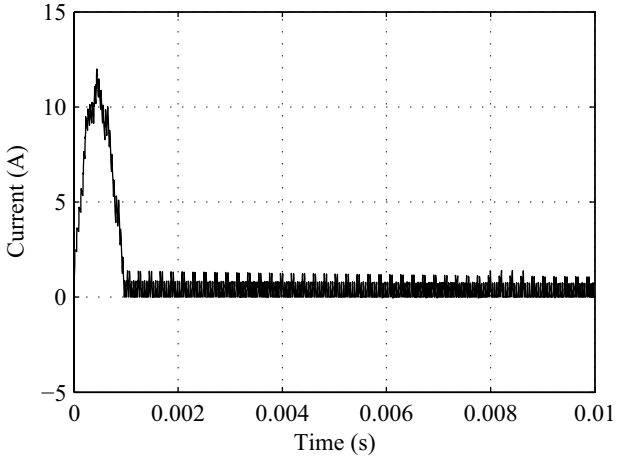


Fig. 5. Inductor current in new model using Euler's method (time step of 300 ns).

The new model has an apparently tremendous advantage over a bilinear model. The new model accomplishes with one element the functionality of four elements from the Modelica Standard Library connected appropriately. The result is a decrease in the number of variables by about a factor of two. The CPU must spend some time computing the value of each variable and storing the result to memory. To make a more fair comparison, a reduced-variable bilinear model was constructed. The reduced-variable model has the same total number of variables as the new model, so all of the simulation time differences should relate directly to model efficiency. The computation times for Euler's method are related entirely to the number of variables in the system.

Using a 300 ns time step, the fixed-time-step algorithms were unstable for the bilinear models. Some simulations did not complete; others completed but with unusable results. This is not surprising given the discussion above that would indicate a need for a sub-ns time step. With the new model, though, results similar to Fig. 5 were attained for all fixed-time-step algorithms.

The smallest improvement in simulation time (3%) was for

TABLE I
COMPARISON OF SIMULATION TIMES
(SECONDS OF CPU TIME FOR 10 MS OF RESULTS)

Method	Bilinear Resistors	Bilinear (Reduced Variables)	New Model
LSODAR	0.813	0.422	0.297
DASSL	1.060	0.484	0.469
ODASSL	1.080	0.485	0.453
Euler	0.406*	0.203*	0.203
Runge-Kutta (2nd order)	*	1.020*	0.203
Runge-Kutta (3rd order)	0.594*	0.219*	0.219
Runge-Kutta (4th order)	*	1.880*	0.250

* - simulation unstable

DASSL. The DASSL algorithm is extremely robust for stiff systems, DAEs, and so forth. Using LSODAR, another algorithm designed for stiff systems, the improvement was much more dramatic, 29.6%. LSODAR is generally faster than DASSL (up to 36.6% in this case), particularly for well-designed models.

V. Conclusions

A new model for a switch pole, including its associated inductor, has been shown. The new model provides much more accurate results when fixed-time-step algorithms are used. The new model also simulates up to 29.6% faster than a simple bilinear resistor model when advanced stiff-system solvers are used. The order of the system (the number of state variables) remains constant, so standard integration algorithms can be used. Modelica code is provided for use in Dymola, but other modeling languages and simulation environments can be used as well.

References

[1] Dynasim, *Dymola User Manual (version 5.1b)*. Lund, Sweden: Dynasim AB, 2003.

[2] J. L. Tichenor, S. D. Sudhoff, and J. L. Drewniak, "Behavioral IGBT modeling for predicting high frequency effects in motor drives," *IEEE Trans. Power Electronics*, vol. 15, pp. 354-360, Mar. 2000.

[3] C. A. Murphy and P. T. Krein, "High level Simulink switch model for investigating capacitive effects," in *Proc. Computers in Power Electronics Workshop*, 2000, pp. 241-246.

[4] J. W. Kimball, "Modeling controlled switches and diodes for electro-thermal simulation," in *Proc. Power Electronics Specialists Conf.*, 2005, pp. 2175-2179.

[5] B. Cassimere, S. D. Sudhoff, D. Aliprantis, and M. D. Swinney, "IGBT and PN junction diode loss modeling for system simulations," in *Proc. IEEE Intl. Conf. Electric Machines and Drives*, 2005, pp. 941-949.

[6] E. Van Dijk, J. N. Spruijt, D. M. O'Sullivan, and J. B. Klaassens, "PWM-switch modeling of dc-dc converters," *IEEE Trans. Power Electronics*, vol. 10, pp. 659-665, Nov. 1995.

[7] R. Tymerski, V. Vorperian, F. C. Y. Lee, and W. T. Baumann, "Nonlinear modeling of the PWM switch," *IEEE Trans. Power Electronics*, vol. 4, pp. 225-233, April 1989.

[8] M. Otter, "Modelica Portal -- Modelica," May 9, 2005, HTTP://WWW.MODELICA.ORG/LIBRARY/MODELICA.

[9] L. R. Petzold, "A description of DASSL: A differential/algebraic system solver," in *Proc. 10th IMACS World Congress*, 1982, pp. 430-432.

[10] A. C. Hindmarsh, "ODEPACK, a systematized collection of ODE solvers," in *Scientific Computing*, R. S. Stepleman et al., Ed. Amsterdam: North-Holland, 1983.

Appendix A: SPDT Switch Pole (Modelica)

```
model Synchronous "Synchronous Switch Pole with Lossy Inductor"
  parameter Modelica.SIunits.Inductance L=100e-6 "Inductor Value";
  parameter Modelica.SIunits.Resistance R=0.1 "Parasitic Resistance";

  Boolean poscurrent;
  Modelica.SIunits.Voltage Vsw "Switch Node Voltage";
  Modelica.SIunits.Current Il "Inductor Current";
  Modelica.Electrical.Analog.Interfaces.PositivePin p …;
  Modelica.Electrical.Analog.Interfaces.NegativePin n …;
  Modelica.Electrical.Analog.Interfaces.Pin s …;
  Modelica.Blocks.Interfaces.BooleanInPort Q …;
equation
  poscurrent = Q.signal[1] or (pre(poscurrent) and (Il > 0));
  L*der(Il) = Vsw - s.v - R*Il;
  Il = -s.i;
  Vsw = if Q.signal[1] then p.v else n.v;
  p.i = if Q.signal[1] then Il else 0;
  n.i = if Q.signal[1] then 0 else -Il;
end Synchronous;
```

Appendix B: DCM Switch Pole (Modelica)

```
model BuckDerived "Buck Derived Switch Pole with Lossy Inductor"
  parameter Modelica.SIunits.Inductance L=100e-6 "Inductor Value";
  parameter Modelica.SIunits.Resistance R=0.1 "Parasitic Resistance";
  parameter Modelica.SIunits.Voltage Vf=0.7 "Diode Forward Voltage";

  Boolean poscurrent;
  Modelica.SIunits.Voltage Vsw "Switch Node Voltage";
  Modelica.SIunits.Current Il "Inductor Current";
  Modelica.Electrical.Analog.Interfaces.PositivePin p …;
  Modelica.Electrical.Analog.Interfaces.NegativePin n …;
  Modelica.Electrical.Analog.Interfaces.Pin s …;
  Modelica.Blocks.Interfaces.BooleanInPort Q …;
equation
  poscurrent = Q.signal[1] or (pre(poscurrent) and (Il > 0));
  L*der(Il) = Vsw - s.v - R*Il;
  Il = -s.i;
  Vsw = if Q.signal[1] then p.v else if poscurrent then (n.v - Vf) else s.v;
  p.i = if Q.signal[1] then Il else 0;
  n.i = if (poscurrent and not Q.signal[1]) then -Il else 0;
end BuckDerived;
```