SIMULATION OF CAUSALLY DYNAMIC HYBRID BOND GRAPHS, WITH APPLICATION TO A POWER CONVERTER

R. Margetts
University of Lincoln,
School of Engineering,
Brayford Pool,
Lincoln, LN6 7TS, UK
email: rmargetts@lincoln.ac.uk

B. Boudon
Université Aix-Marseille,
CNRS,
ISM UMR 7287,
13288 Marseille, France
email: benjamin.boudon@univ-amu.fr

R. F. Ngwompo
University of Bath,
Dept. Mechanical Engineering,
Claverton Down,
Bath BA2 7AY, UK
email: r.f.ngwompo@bath.ac.uk

ABSTRACT
Causally dynamic hybrid bond graphs are generally considered unsuitable for simulation, and causality is therefore often constrained in hybrid bond graph models. This paper demonstrates how a causally dynamic model can be simulated, using a buck converter as a case study. A causally dynamic hybrid bond graph (utilizing controlled junctions) is used to derive a mixed-Boolean state equation. This state equation is transferred to MATLAB®, where a simple routine assigns values to the Boolean parameters and then solves the model. Where storage elements are in dynamic causality, the model takes descriptor state form and an implicit solver is used. Solver choice and event detection are discussed. MATLAB® was selected as an accessible environment which allows this type of model to be coded and solved, but the technique could be used in an environment of the practitioners choice. The power converter is successfully modeled with a fast simulation time, demonstrating that simulating causally dynamic hybrid bond graphs is possible and merits further refinement.

KEY WORDS
Bond graph; hybrid model; switched model; system dynamics; power converter.

1 Introduction

A causally dynamic hybrid bond graph has been proposed which is designed to be suitable for both graphical/qualitative analysis and simulation of nonsmooth dynamical systems [1, 2, 3]. Qualitative analysis has already been established for this type of model, and this paper demonstrates how causally dynamic models can be simulated.

Hybrid bond graphs are those containing elements enabling to describe both continuous and discontinuous behaviour. Different types of discontinuities can be encountered relating to different physical phenomena, such as structural discontinuities for contact or switches, or parametric discontinuities for elements with piecewise continuous functions. Switches are modelled in the bond graph framework using controlled junctions (X0 or X1), which are associated with a Boolean switching parameter. During the simulation, the causality assignment of the model can change with the state of the switch: a phenomenon known as dynamic causality.

Prior to this work, hybrid bond graph simulation generally relied on *causality resistance* or parasitic compliance e.g. [4, 5, 6], which can be incorporated in bond graph environments such as 20Sim. There is also work in the literature suggesting the use of petri-nets [7], DEVS (Discrete Event System Specification) [8] or alternative causality assignment procedures [9]. A full review is presented in [2]. However, the causally dynamic method was proposed to allow stiff and variable structure systems to be modelled without the addition of compliance or resistance. This is because adding parasitic elements may introduce undesirable high-frequency dynamics, complicating the model and slowing the simulation [10]. The causally dynamic bond graph can also reveal the essential properties of variable structure models, which may not otherwise be easily visible [11].

A feature of causally dynamic hybrid models is that, where storage elements are in dynamic causality, there is inevitably derivative causality in some modes of operation. Bond graph modellers typical avoid derivative causality for the purposes of simulation, as it signifies the presence of DAEs (Differential Algebraic Equations). However, these can reflect the physics of the system (for example, when two bodies are rigidly connected during a contact problem) and can be easily solved used modern solvers such as Backwards Differentiation Formula (BDF).

For ease of accessibility, the simulation was conducted in MATLAB®. However, the principles can be transferred to a language and environment of the reader's choice. It is the authors' hope that this technique can be incorporated into bond graph modelling environments, as well as impacting the study of hybrid and variable-structure systems in general.

2 Method

The causally dynamic bond graph generates a mixed-Boolean state equation. In general, this mixed-Boolean

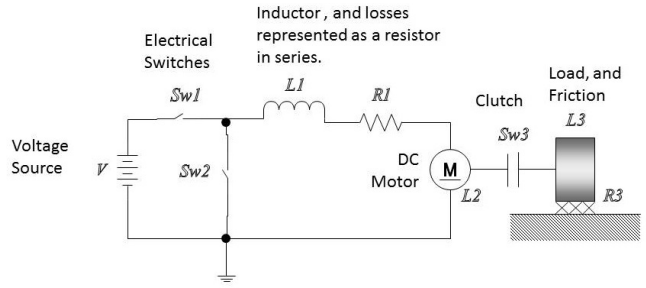

Figure 1. Schematic Diagram of a Buck Converter Supplying a D.C. Motor with Load [12].

state equation takes the form:

$$\mathbf{\Lambda}\dot{\mathbf{x}} = f_1(\mathbf{x}, \mathbf{z}, \mathbf{u}, \lambda, t) \tag{1}$$

$$\mathbf{0} = f_2(\mathbf{x}, \mathbf{z}, \mathbf{u}, \lambda, t) \tag{2}$$

This is an implicit model or DAE i.e. equations 1 and 2 are differential and algebraic equations respectively. If the system is LTI, a *descriptor state model* is obtained. Implicit models are typically avoided in the field of hybrid bond graphs, since they are considered unsuitable for simulation. However, there has been a body of work on simulating implicit models arising from bond graphs in the field of multibody dynamics [13, 14]. This work uses the mathematical property that models with a DAE index of less than two can still be simulated using modern solvers such as BDF.

A power converter was previously modelled as a causally dynamic hybrid bond graph [1], for comparison with Buisson et al.'s switching bond graph of a power converter. This was selected as a case study which often manifests in the literature on hybrid models, and can be challenging to simulate. The schematic is shown in figure 1 and the hybrid bond graph in figure 2.

MATLAB® scripts were written to simulate this model under a range of conditions. The general script defined simulation time and initial conditions, and then solved the model and plotted the results.

The mathematical model for this buck converter system is presented in Figure 3 without derivation. Some roughly representative figures are taken for the converter: inductance of $10mH$, resistance of 1.73Ω, motor shaft inertia of $2.25 \times 10^{-7} kg.m^2$, load inertia of $1 \times 10^{-5} kg.m^2$ and a damping coefficient of $2 \times 10^{-5} Nm/\,(rad/s)$. The DC motor is modelled as a gyrator element with a modulation constant of 0.00902. The input *u* assumes a $28V$ input from the power source.

2.1 Study 1: Normal Operation, Load Disconnected (Explicit Model)

In normal operation, switches 1 and 2 alternate between 'ON' or 'OFF.' In this study, switch 3 remains 'OFF' (i.e. the load is disconnected) to yield an explicit model. The model can be simplified to a system of Ordinary Differential Equations (ODEs) and solved with any solver. In this case, ODE15s was selected as it is a potentially stiff model.

$$\begin{bmatrix} \dot{p}_1 \\ \dot{p}_2 \\ \dot{p}_3 \end{bmatrix} = \begin{bmatrix} \frac{1}{L_1 R_1} & \frac{-a}{L_2} & 0 \\ \frac{a}{L_1} & 0 & 0 \\ 0 & 0 & \frac{1}{L_3 R_2} \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} + \begin{bmatrix} \lambda_1 & \lambda_2 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} V \\ G \end{bmatrix} \tag{3}$$

The switches operate at a set frequency, in this case $100kHz$. Appendix 1 shows an example MATLAB® code for this model. Essentially, the Boolean parameters are defined symbolically, and a short loop assigns them a '1' or '0' depending on the time. Note that the final results for this case were gathered using the same script as the other [implicit] models for consistency, but using a stiff solver to accommodate the different form of equation and high-frequency switching, as shown in Appendix 2.

2.2 Study 2: Normal Operation, Load Connected (Implicit Model)

In normal operation, the buck converter rapidly alternates between two modes: Sw1 'ON' Sw2 'OFF' and Sw1 'OFF' Sw2 'ON,' as already illustrated in Study 1. In this study, the load is connected (i.e. Sw3 remaining 'ON' throughout). This model is implicit, due to a causal path (representing a real kinematic constraint) between the rigidly-connected motor and load. The implicit solver ODE15i is used, as it is the only implicit solver available in Matlab.

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{p}_1 \\ \dot{p}_2 \\ \dot{p}_3 d \end{bmatrix} = \begin{bmatrix} \frac{1}{L_1 R_1} & \frac{-a}{L_2} & 0 \\ \frac{a}{L_1} & \frac{1}{L_2 R_2} & 0 \\ 0 & \frac{1}{L_2} & \frac{1}{L_3} \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 d \end{bmatrix} + \begin{bmatrix} \lambda_1 & \lambda_2 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} V \\ G \end{bmatrix} \tag{4}$$

As with Study 1, a switching frequency of $100kHz$ is used. The first few cycles are simulated assuming zero initial conditions.

Appendix 2 shows the code for this study. This routine includes codes for automatically deleting the unnecessary rows and columns of zeros (which would result in the model being identified as nonsingular).

2.3 Study 3: Constant Input, Load Disconnected during Operation (Implicit Model)

This study gives the case where a constant input is provided (i.e. Sw1 'ON' and Sw2 'OFF' throughout) and the load

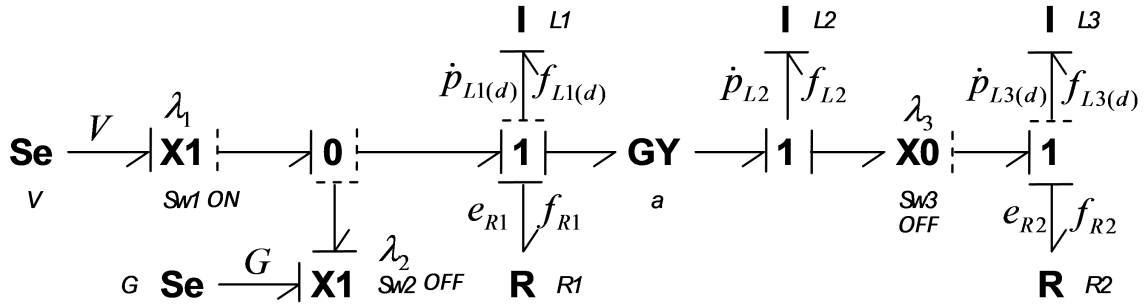

Figure 2. Hybrid Bond Graph Model of the System.

$$\begin{bmatrix} (\lambda_1 \oplus \lambda_2) & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & \lambda_3 \\ 0 & 0 & \bar{\lambda_3} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{p}_1 \\ \dot{p}_2 \\ \dot{p}_3 \\ \dot{p}_1 d \\ \dot{p}_3 d \end{bmatrix} =$$

$$\begin{bmatrix} \frac{-(\lambda_1 \oplus \lambda_2)}{L_1 R_1} & \frac{-a(\lambda_1 \oplus \lambda_2)}{L_2} & 0 & 0 & 0 \\ \frac{a(\lambda_1 \oplus \lambda_2)}{L_1} & \frac{\lambda_3}{L_2 R_2} & 0 & 0 & \\ 0 & 0 & \frac{\bar{\lambda_3}}{L_3 R_2} & 0 & 0 \\ 0 & 0 & 0 & \frac{(\lambda_1 \oplus \bar{\lambda_2})}{L_1} & 0 \\ 0 & \frac{\lambda_3}{L_2} & 0 & 0 & -\frac{\lambda_3}{L_3} \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_1 d \\ p_3 d \end{bmatrix} + \begin{bmatrix} \lambda_1 & \lambda_2 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} V \\ G \end{bmatrix} \quad (5)$$

Figure 3. Mathematical Model of the System as a Mixed-Boolean State Equation

is disconnected (Sw3 switches from 'ON' to 'OFF') at 5s. The simulation is stopped at the event time and some 'new' initial conditions (equal to the last state values prior to the event) defined. This prevents the model from becoming unstable at the event time. It is worth noting that there is no state reinitialisation or estimation: the state values are simply carried over from immediately before the event.

3 Results

3.1 Study 1: Normal Operation, Load Disconnected (Explicit Model)

The results are shown in figures 4 and 5. After an initial transition, a steady-state torque is applied to the motor. Note the small quantities: a zero voltage overall and around $3 \times 10^{-4} Nm$. No torque is applied to the [disconnected] load, which is consistent with expectation.

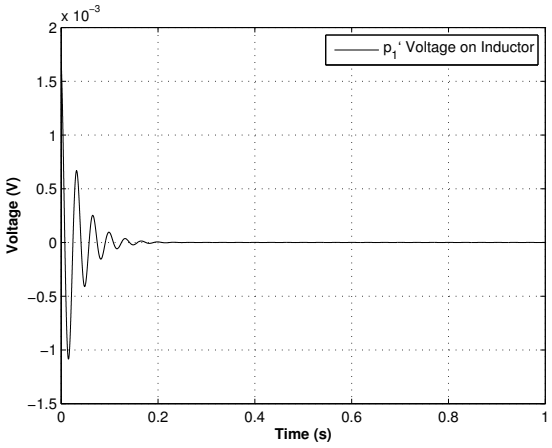

Figure 4. Study 1 Voltage: Normal Operation, Load Disconnected.

3.2 Study 2: Normal Operation, Load Connected (Implicit Model)

The results are shown in figures 6 and 7. In this case the voltage and torque on the motor are reduced compared to study 1, and a small torque is evident on the load. The graphs appear 'noisy,' but enlarging the signal (as in Figure 8) reveals that it is periodic consistent with the fast switching frequency of the electrical switches. Using more time steps might yield a smoother time signal, but at the expense

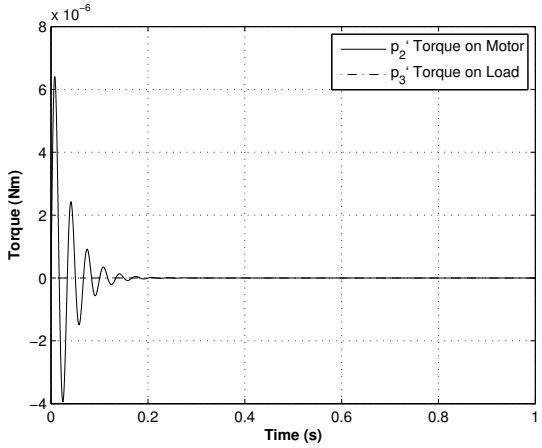

Figure 5. Study 1 Torque: Normal Operation, Load Disconnected.

of simulation time.

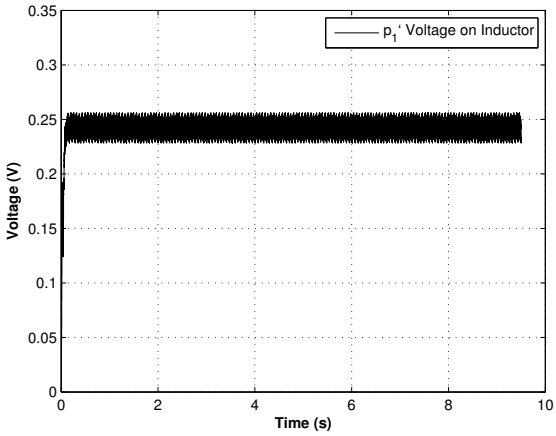

Figure 6. Study 2 Voltage: Normal Operation, Load Connected.

3.3 Study 3: Constant Input, Load Disconnected during Operation (Implicit Model)

The results are shown in figures 9 and 10. The load is disconnected 5s into the simulation. The model runs quickly, and shows a 'spike' in voltage immediately after the event. The torque on the motor is increased after the event, consistent with the absence of the load.

4 Conclusion

This work demonstrates how a mixed-Boolean state model, derived from a hybrid bond graph, can be solved using MATLAB®.

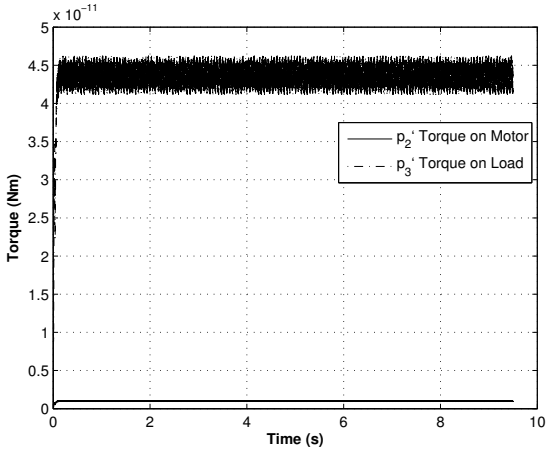

Figure 7. Study 2 Torque: Normal Operation, Load Connected.

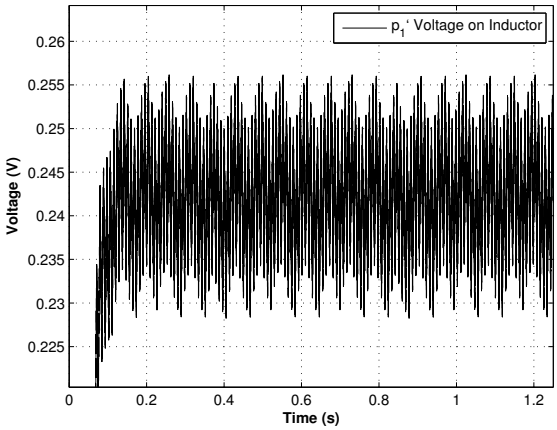

Figure 8. Study 2 Voltage: Normal Operation, Load Connected, Signal Enlarged.

Once the Boolean parameters have been assigned a numeric value, the model can be solved using standard ODE solvers. Since hybrid models are often implicit (where commutation places a storage element in derivative causality) the ode15i solver is suggested for use.

Where switching occurs during a simulation, a simple routine assigns Boolean parameters at given times and simplifies the model (to prevent nonsingularities due to unused rows and columns).

In these studies, switching occurred at known times. Matlab can also handle event-driven commutation using the 'event' command, as demonstrated using the example of a bouncing ball [15].

A programme of future work is proposed including more in-depth studies of power converters.

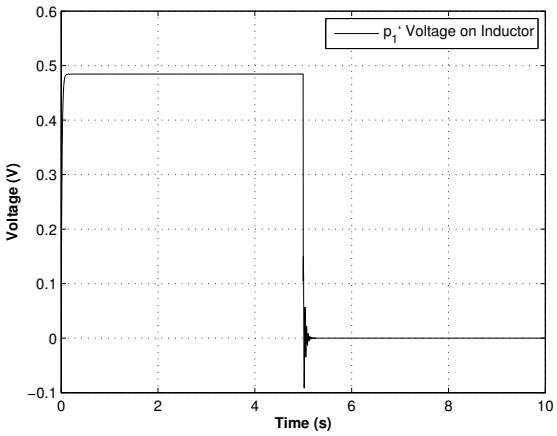

Figure 9. Study 3 Voltage: Constant Input, Load Disconnected during Operation.

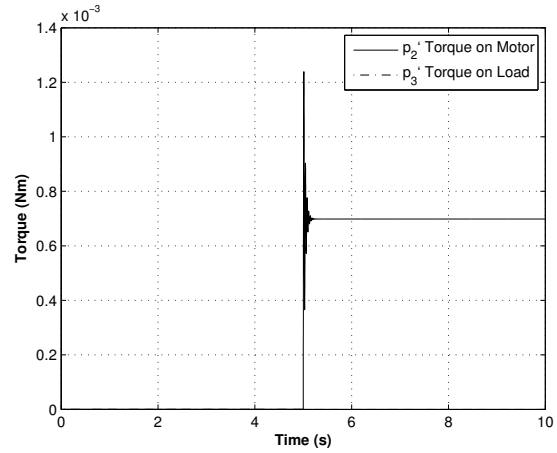

Figure 10. Study 3 Torque: Constant Input, Load Disconnected during Operation.

Acknowledgements

This work was a consequence of a collaborative visit by engineering researchers at Université Aix-Marseille to the University of Lincoln. The authors would like to acknowledge the help of Dr. Thu-Thuy Dang of Université Aix-Marseille for providing representative data for the case study.

References

[1] R. Margetts, R. F. Ngwompo, and M. Fortes da Cruz. Construction and Analysis of Causally Dynamic Hybrid Bond Graphs. *Proc. IMechE Part I - J. Syst. and Control Eng.*, 227:329–346, 2013.

[2] Rebecca Margetts. *Modelling & Analysis of Hybrid Dynamic Systems using a Bond Graph Approach*. PhD thesis, University of Bath, Bath, UK, 2013.

[3] R. Margetts and R.F. Ngwompo. Mode switching in causally dynamic hybrid bond graphs. *Mechatronics*, May 2015.

[4] G. M. Asher. The Robust Modelling of Variable Topology Circuits Using Bond Graphs. In *Proc. 1993 Western Simulation Multiconference - Int. Conf. on Bond Graph Modeling ICBGM'93*, pages 126–131, La Jolla, CA, 1993.

[5] P. C. Breedveld. An alternative Model for Static and Dynamic Friction in Dynamic System Simulation. In *1st IFAC Conf. Mechatronic Systems*, pages 717–722, Darmstadt, Germany, 2000.

[6] W. Borutzky. Bond-graph-based fault detection and isolation for hybrid system models. *Proc. IMechE. Part I, J. Syst. and Control Eng.*, 226:742–760, 2012.

[7] W. Borutzky. Discontinuities in a bond graph framework. *J. Franklin Institute - Eng. and Appl. Math.*, 332B:141–154, 1995.

[8] E. Kofman and S. Junco. Quantized Bond Graphs: An Approach for Discrete Event Simulation of Physical Systems. In *Proc. 2001 Int. Conf. on Bond Graph Modeling ICBGM'01*, pages 369–374, Phoenix, AZ, 2001.

[9] C. B. Low, D. Wang, S. Arogeti, and J. B. Zhang. Causality assignment and model approximation for hybrid bond graph: Fault diagnosis perspectives. *IEEE Trans. Automation Sci. and Eng.*, 7:570–580, 2010.

[10] J. Buisson. Analysis and Characterization of Hybrid Systems With Bond-Graphs. In *Proc. Int. Conf. on Systems, Man and Cybernetics: Systems Engineering In the Service of Humans*, volume 1, pages 264–269, Le Touquet, France, 1993. IEEE.

[11] F E Cellier, M Otter, and H Elmqvist. Bond Graph Modeling of Variable Structure Systems. *Simulation Series*, 27:49, 1994.

[12] J. Buisson, H. Cormerais, and P. Y. Richard. Analysis of the bond graph model of hybrid physical systems with ideal switches. *Proc. IMechE Part I - J. Syst. and Control Eng.*, 216:47–63, 2002.

[13] B. Boudon, F. Malburet, and J. Carmona. Simulation of a helicopter's main gearbox semi-active suspension with bond graphs. *Multibody System Dynamics Journal*, 2017.

[14] W Borutzky. *Bond Graph Methodology: Development and Analysis of Multi-disciplinary Dynamic System Models*. Springer-Verlag London Ltd., London, UK, 1st edition, 2010.

[15] R. Margetts and R. F. Ngwompo. Hybrid bond graphs for contact, using controlled junctions and dynamic causality. In *Proc. SCS SummerSim'14*, Monterey, CA, 2014. SCS.

APPENDICES

Appendix 1: Example MATLAB Script for Explicit Model

Example Matlab code is provided here with the intention that it can illustrate how hybrid systems may be approached, and be used as a pseudocode for future programming activities in other languages.

```
tspan = [0:(1/1000000):1];
iniCon = [0,0,0];
[t, y] = ode15s(@hbg_explicit, tspan,
iniCon);

figure(1);
set(0,'DefaultAxesColorOrder',[0 0 0],...
'DefaultAxesLineStyleOrder','-|-.|--|:')
plot(t,y(:,1));
grid on;
legend('p_1` Voltage on Inductor');
ylabel('\bfVoltage (V)');
xlabel('\bfTime (s)');

figure(2)
set(0,'DefaultAxesColorOrder',[0 0 0],...
'DefaultAxesLineStyleOrder','-|-.|--|:')
pt = plot(t,y(:,2),t,y(:,3));  %plot(t,y,
'b-','LineWidth',2);
grid on;
legend('p_2` Torque on Motor','p_3`
Torque on Load')
xlabel('\bfTime (s)');
ylabel('\bfTorque (Nm)');
```

The system is represented in a MATLAB® function *hbg* as follows:

```
function  out = hbg(t, y)
index = t*100000;
ts = round(index);
n=0;
even = n(rem(ts,2)==0);
if even == 0;
l1=0;
l2=1;
l3=0;
else
l1=1;
l2=0;
l3=0;
end
L1 = 10e-3;
L2 = 2.25e-7;
L3 = 1e-5;
R1 = 1.73;
R2 = 2e-5;
a = 0.00902;

A=[-1*xor(l1,l2)/(L1*R1) -a*xor(l1,l2)/L2 0
a*xor(l1,l2)/L1 -l3/(L2*R2) 0
0 0 ~l3/(L3*R2)];
B=[l1 l2
0 0
0 0];
u = [28;0];

out = A*y + B*u;

end
```

Appendix 2: Example MATLAB® Script for Implicit Models

The implicit model is set up in much the same way as the explicit one, calling a function.

```
tspan = [0:(1/1000000):9.5];
y0 = zeros(1,3)';
yp0 = zeros(1,3)';
[t, y] = ode15i(@hbg_implicit, tspan,
y0, yp0);
```

In the third case study, the simulation was halted at the event time and then run again. This prevented the model from becoming unstable and failing as it became unable to meet integration tolerances. The states are not reinitialised at the event time: they are simply taken to be the same as at the last time step (which is reasonable since they have reached steady state). N.b. it is possible to simplify this code by specifying event times as an integrator option.

```
% Run simulation until event
tspan1 = [0:1e-3:(5-(1e-3))];
y01 = [0 0 0]';
yp01 = zeros(1,3)';
[t1, y1] = ode15i(@hbg_implicit, tspan1,
y01, yp01);

% Run simulation from event
tspan2 = [5:1e-3:10];
y02 = [0.104 0 0]';
yp02 = zeros(1,3)';
[t2, y2] = ode15i(@hbg_implicit, tspan2,
y02, yp02);

% Concatenate results vectors
t = [t1; t2];
y = [y1; y2];
```

The system is represented in a MATLAB® function as follows. The values of l1, l2 and l3 are altered to reflect the state of the corresponding switches at a given time, specified using t.

In order to utilise the ode15i [implicit] solver, the implicit (or 'descriptor') state model is rearranged into $0 = f(y, y', t)$ form. The model must be simplified for each mode of operation since, if rows and columns of zeros are left in the matrices, the model is identified as nonsingular and cannot be solved.

```
function  out = hbg(t, y, yp)

index = t*100000;
ts = round(index);
n=0;
even = n(rem(ts,2)==0);
if even == 0;  %Change to 100kHz
l1=0;
l2=1;
l3=1;
hsm = hss(l1, l2, l3);
else
l1=1;
l2=0;
l3=1;
hsm = hss(l1, l2, l3);
end

u = [28;0]; % Input V
d = length(hsm(:,1));
w = length(u(:,1));
A = hsm(:,1:d);
B = hsm(:,d+1:d+w);
E = hsm(:,d+w+1:d+w+d);
out = A*y + B*u - E*yp;

end
---------------------------------
function hsm = hss(l1, l2, l3)

L1 = 10e-3;
L2 = 2.25e-7;
L3 = 1e-5;
R1 = 1.73;
R2 = 2e-5;
a = 0.00902;
A = [-1*xor(l1,l2)/(L1*R1) -a*xor(l1,l2)/L2
0  0 0
a*xor(l1,l2)/L1 -l3/(L2*R2) 0 0 0
0 0 ~l3/(L3*R2) 0 0
0 0 0 ~xor(l1,l2)/L1 0
0 l3/L2 0 0 -l3/L3];
B = [l1 l2
0 0
0 0
0 0
0 0];
E = [xor(l1,l2) 0 0 0 0
0 1 0 0 l3
0 0 ~l3 0 0
0 0 0 0 0
0 0 0 0 0];

% Reduce size of model
order = length(A(1,:));
z = zeros(1,order);
i = 1;
for n = (1:order)
% Delete rows of zeros
if sum(A(n,:)) == 0
else
Ared1(i,:) = A(n,:);
Bred1(i,:) = B(n,:);
Ered1(i,:) = E(n,:);
i = i+1;
end;
end;

i = 1;
for n = (1:order)
% Delete Columns of zeros
if sum(Ared1(:,n)) == 0
else
Ared2(:,i) = Ared1(:,n);
Bred2 = Bred1;
Ered2(:,i) = Ered1(:,n);
i = i+1;
end;
end;

hsm = [Ared2 Bred2 Ered2];
end
```