ABSTRACT

This paper describes the development of advanced control and modeling algorithms for the various types of motor drives considered for hybrid electric vehicles (HEVs). The algorithms are given in the high-level language of MATLAB/Simulink. The algorithms consist of Simulink blocks that can be easily implemented in a real-time test environment for induction, switched reluctance, and permanent magnet synchronous machines. This eliminates the need for specialized programming in C or assembly languages providing investigators a much simpler way to study proposed control algorithms for various motor drives.

INTRODUCTION

This paper describes the development of a Simulink library of control and modeling algorithms for the various types of motor drives considered for hybrid electric vehicles (HEVs). The algorithms are given in the high-level language Simulink from Mathworks Inc. The algorithms consist of Simulink blocks that can be easily implemented in a real-time test environment for induction, switched reluctance, and permanent magnet synchronous machines. This eliminates the need for specialized programming in C or assembly languages providing investigators a much simpler way to study proposed control algorithms for various motor drives.

The components of a HEV include an internal combustion engine, an AC motor/generator, a battery and power electronics. From the electrical point of view, a major concern is the challenge of using an inverter (power electronics) to convert the battery power to the appropriate time-varying voltages required by the electric motor to produce torque for the vehicle's propulsion.

Presently, researchers have considered several motor types including the DC motor, induction motor, permanent magnet (PM) synchronous motor, and the switched reluctance (SR) motor. Each of these motors are quite different in their operation, and each requires a specialized computer controller (software program) to determine when the electronic switches of the power inverter should switch to produce the appropriate voltage for the motor. The issue of which motor to use for the propulsion of electric vehicles is still unresolved because the cost, performance, reliability, size, and efficiency all play a large role in the decision and no one motor can presently be said to be the "best" choice (The dc motor has been ruled out due to its higher cost and higher maintenance requirements.). Consequently, having a library of various motor models and controllers is advantageous for the development of HEVs.

The approach here is to do the control system design software using MATLAB/Simulink which allows one not only to design a controller for the system using block diagrams (on the design PC), but it also provides the capability to convert the block diagram to executable code which is then downloaded to be run on a real-time PC. That is, the software program RTW® (real-time workshop) converts the Simulink diagram to C code and then this can be converted to real-time executable code. The I/O (input/output) interface boards read in the information (voltages, currents, speed, position, temperature, etc.) from the motor needed by the control program and sends out the commands to the power electronics as determined by the control program. This approach eliminates the need for specialized programming in C or Assembly Languages, and instead allows investigators to spend their time intensively studying the proposed control algorithms.

Basic Simulink Libraries are developed for the switched reluctance, PM synchronous and induction motors.

REAL-TIME TEST BED

A real-time computing platform has been developed as a test bed to efficiently and accurately carry out the real-time implementation of sophisticated computer algorithms for the control of any of the possible alternatives for the propulsion of the electric vehicle. As shown in the Figure below, this platform consists of three

1 An appropriate analogy is the typewriter versus the word processor. The ability to do work with a word processor (correspondingly, a real-time setup) greatly decreases the turn around time and increases the possibilities compared with using a typewriter (C or Assembly Language Programming).
personal computers (PCs) in which one serves as “host" PC for the development/design using Simulink and the other two “target" PCs are used to run in real-time to serve as the controller for the motor/inverter. There are various 3rd party vendors who provide the capability to convert the Simulink generated C-code to real-time executable code. We chose the real-time implementation platform using RTLAB® from Opal RT Technologies 1 since it provides a high performance solution from off the shelf equipment making it relatively a low cost approach. The two target PCs allow one to different parts of the controller at different rates. In particular, one target PC can run with a 50-microsec step size for the power electronic inverter controller and the other with a 200-microsec step size for the motor controller.

As a first example, it is shown how the evaluation platform can be a powerful tool to automatically characterize (model) and control a SR motor drive system for HEVs. The basic library consists of computer algorithms (Simulink programs) provide.

Automatic identification of the $P$th phase torque function $\tau_P(n_R, \theta, i)$.

Automatic identification of $R_s, J$.

A motor controller (software algorithm) that determines the currents $i_a, i_b, i_c$ needed at each instant in time so that the motor produces the requisite torque.

The basic mathematical model of a three-phase switched reluctance motor is given by

$$\frac{d}{dt} \lambda_1(n_R, \theta, i_1) = -R_s i_1 + u_1$$ (1)

$$\frac{d}{dt} \lambda_2(n_R, \theta, i_2) = -R_s i_2 + u_2$$ (2)

$$\frac{d}{dt} \lambda_3(n_R, \theta, i_3) = -R_s i_3 + u_3$$ (3)

$$J \frac{d\omega}{dt} = \tau - \tau_L$$ (4)

$$\tau = \tau_1(n_R, \theta, i_1) + \tau_2(n_R, \theta, i_2) + \tau_3(n_R, \theta, i_3)$$ (5)

$$= \frac{\partial}{\partial \theta} \left[ \int_0^{i_1} \lambda_1(n_R, \theta, i) \, di + \int_0^{i_2} \lambda_2(n_R, \theta, i) \, di + \int_0^{i_3} \lambda_3(n_R, \theta, i) \, di \right]$$ (6)

Here $\lambda_a, \lambda_b, u_a, i = 1,2,3$ are, respectively, the fluxes, currents and voltages in the three stator phases. This is written in nonlinear state-space form as

$$\frac{di_1}{dt} = \left[ -R_s i_1 - \frac{\partial \lambda_1(n_R, \theta, i_1)}{\partial \theta} n_R \omega + u_1 \right] \frac{\partial \lambda_1(n_R, \theta, i)}{\partial i}$$ (7)

$$\frac{di_2}{dt} = \left[ -R_s i_2 - \frac{\partial \lambda_2(n_R, \theta, i_2)}{\partial \theta} n_R \omega + u_2 \right] \frac{\partial \lambda_2(n_R, \theta, i)}{\partial i}$$ (8)

$$\frac{di_3}{dt} = \left[ -R_s i_3 - \frac{\partial \lambda_3(n_R, \theta, i_3)}{\partial \theta} n_R \omega + u_3 \right] \frac{\partial \lambda_3(n_R, \theta, i)}{\partial i}$$ (9)

$$J \frac{d\omega}{dt} = \tau(n_R, \theta, i_1, i_2, i_3) - \tau_L$$ (10)
The basic controller consists in using high-gain PI current controllers of the form

\[ u_i = K_P (i_{ir} - i_i) + K_I \int_0^t (i_{ir} - i_i) dt \]  

where it is assumed that \( i_i \to i_{ir} \) fast enough that the reference currents \( i_{ir} \) can be considered as new inputs, the model reduces to

\[ J \frac{d\omega}{dt} = \tau_1 (n_R \theta, i_{ir}) + \tau_2 (n_R \theta, i_{2r}) + \tau_3 (n_R \theta, i_{3r}) - \tau_L \]  

The problem remains is, given a requested torque in the machine \( \tau_r \), determine as a function of position \( \theta \) the amount of torque each phase contributes so that

\[ \tau_r = \tau_{ir} + \tau_{2r} + \tau_{3r} \]  

and determine the amount of current required in each phase to obtain these desired torques, i.e.,

\[ i_{ir} = \tau_1 (n_R \theta, \tau_{ir}) \]  
\[ i_{2r} = \tau_2 (n_R \theta, \tau_{2r}) \]  
\[ i_{3r} = \tau_3 (n_R \theta, \tau_{3r}) \]  

These currents are then the commands to the current controller.

The algorithms used the in SIMULINK Library used to specify torque and current are those described in [7][8].
Clicking on the **Phase Current Command** and **their Torques** block gives the fourth level shown in Figure 6.

**Figure 6: Fourth Level**

Finally, clicking on the **Phase 1 Torque** block gives the fifth level of the model shown in Figure 7. The point here is to show that complexity involved in such controllers.

The complete controller is developed off line in simulation. Then, one removes the motor model and replaces them with the RTLAB (Opal RT) I/O blocks icons (or a similar 3rd party vendor’s software icons) to communicate with the A/Ds, D/As, encoder, and digital I/O. Then one compiles the SIMULINK model into C code using RTW that is then converted into executable code using RTLAB (Opal RT, Inc.) software. The look up table in the above figure shows the torque $\tau_1(n_R, i_1)$ as a function of position for various current levels. Such a table is found experimentally. A SIMULINK model was developed to automatically identify the function $\tau_1(n_R, i_1)$ and the experimental results for an eight-pole rotor switched reluctance motor are shown in the figure below.

**Figure 7: Fifth Level**

This was an automatic identification procedure in which the position of load motor is incremented in steps of $360\degree/400 = .09\degree$ starting at zero going to a full $360\degree$ turn of the motor. With a constant current in phase 1 of the SR motor, the torque is measured using a torque sensor placed between the test motor and load motor. This procedure is automatically repeated for 10 current levels in steps of $2.5A/10 = .25A$ resulting in Figure 8. The data of Figure 8 for just one positive torque cycle is given Figure 8b below. Figure 8 is the data that goes in the lookup table shown in Figure 7.

**Figure 8a**
*Experimentally measured torque as a function of position and current*

The data of Figure 8 is inverted to obtain the phase current as a function of position and torque which is needed by the controller. That is, given the rotor position and the torque, the current in the phase required to get the torque is given in Figure 8c. The Simulink feedback controller outlined above was then converted for real-time implementation using RTLAB (Opal RT, Inc.) software. A trajectory was chosen to have the motor go up to 50 rad/sec and then come back down to rest. The actual data of the speed tracking is shown in Figure 8d.

**Figure 8b**
*Torque as function of position and current (one cycle from Figure 8a)*
In this figure, the speed was computed by backward differentiation of the encoder position measurement and the granularity in the speed is simply due to the resolution of the encoder. The position was also tracked and is shown in Figure 8e along with its reference trajectory. The error in the position tracking is small enough that it is not discernible in this figure. The current tracking is also shown below for one of the phases. This particular implementation used current command amplifiers and the figure indicates that the amplifier was not able to track the commanded currents with zero steady-state error. In fact, further investigation showed that the amplifiers low bandwidth prevented the controller from tracking high speed trajectories. The authors are now developing methods to get around this difficulty.

\[ \frac{d\theta_{pmSS}}{dt} - M \frac{d^2\theta_{pmSS}}{dt^2} = v_{S1} - Ri_{S1} + K_w \omega \sin(n_p \theta) \]

\[ -M \frac{d^3i_{S1}}{dt^3} + L_S \frac{di_{S2}}{dt} = v_{S2} - Ri_{S2} + K_w \omega \sin\left(n_p \theta - \frac{2\pi}{3}\right) \]
The power preserving three-phase to two-phase transformation is defined by

\[
\begin{bmatrix}
i_a \\
i_b \\
i_0
\end{bmatrix} = \frac{2}{\sqrt{3}}
\begin{bmatrix}
1 & -1/2 & -1/2 \\
0 & \sqrt{3}/2 & -\sqrt{3}/2 \\
1/\sqrt{2} & 1/\sqrt{2} & 1/\sqrt{2}
\end{bmatrix}
\begin{bmatrix}
i_{S1} \\
i_{S2} \\
i_{S3}
\end{bmatrix}
\]

(22)

and transforms the original model into the equivalent model

\[
(L_S + M) \frac{di_a}{dt} = v_a - R_i a + \frac{3}{\sqrt{2}} K_m \omega \sin(n_p \theta)
\]

(24)

\[
(L_S + M) \frac{di_b}{dt} = v_b - R_i b + \frac{3}{\sqrt{2}} K_m \omega \sin(n_p \theta)
\]

(24)

\[
(L_S + M) \frac{di_0}{dt} = \frac{1}{\sqrt{3}} v_0 - \frac{1}{\sqrt{3}} R_i 0
\]

(25)

\[
J \frac{d\omega}{dt} = -\frac{3}{\sqrt{2}} K_m i_a \sin(n_p \theta) + \frac{3}{\sqrt{2}} K_m i_b \cos(n_p \theta) - \tau_L
\]

(26)

Under balanced operation in which \(v_0 = v_{S1} + v_{S2} + v_{S3} = 0, i_0 = i_{S1} + i_{S2} + i_{S3} = 0\), one obtains the two-phase equivalent model given by

\[
L \frac{di_d}{dt} = v_d - R_i d + K \omega \sin(n_p \theta)
\]

(27)

\[
L \frac{di_q}{dt} = v_q - R_i q + K \omega \sin(n_p \theta)
\]

(28)

\[
\sqrt{3}(L_S - 2M) \frac{dv_0}{dt} = v_0 - R_i 0
\]

(29)

\[
J \frac{d\omega}{dt} = -Ki_d \omega \sin(n_p \theta) + Ki \omega \cos(n_p \theta) - \tau_L
\]

(30)

where \(L = L_S + M\) (approximately \(K = \sqrt{3}/2 K_m \)), \(3L_S/2\). \(i_b\) and \(i_b\) are the equivalent currents in phases \(a\) and \(b\), respectively.

Letting \(V_{bus}\) denote the DC bus voltage of a 3-phase inverter, the maximum voltage out of the inverter is obtained in six step mode, resulting in a peak of the fundamental waveform equal to \(v_{max} = (2/\pi) V_{bus}\), which is the maximum phase voltage. With \(i_{max}\) and \(v_{max}\) denoting the limits of the 3-phase currents and voltages, respectively, the corresponding limits \(I_{max}, V_{max}\) for the equivalent 2-phase motor are \(I_{max} = \frac{3}{2} I_{max}, V_{max} = \sqrt{3} V_{max}\).

The direct-quadrature or \(dq\) transformation is given by

\[
\begin{bmatrix}
i_d \\
i_q
\end{bmatrix} = \begin{bmatrix}
\cos(n_p \theta) & \sin(n_p \theta) \\
-\sin(n_p \theta) & \cos(n_p \theta)
\end{bmatrix}
\begin{bmatrix}
i_a \\
i_b
\end{bmatrix}
\]

(31)

\[
\begin{bmatrix}
v_d \\
v_q
\end{bmatrix} = \begin{bmatrix}
\cos(n_p \theta) & \sin(n_p \theta) \\
-\sin(n_p \theta) & \cos(n_p \theta)
\end{bmatrix}
\begin{bmatrix}
v_a \\
v_b
\end{bmatrix}
\]

(32)

where \(i_d, i_q\) and \(v_d, v_q\) are the transformed currents and voltages respectively, in the \(dq\) (for direct and quadrature) reference frame. The state-space model in the \(dq\) coordinates is

\[
L \frac{di_d}{dt} = v_d - R_i d + n_p \omega i_d
\]

(33)

\[
L \frac{di_q}{dt} = v_q - R_i q - n_p \omega i_q - K \omega
\]

(34)

\[
J \frac{d\omega}{dt} = Ki_q - \tau_L
\]

(35)

\[
\frac{d\theta}{dt} = \omega
\]

(36)

A standard controller is then

\[
v_q = K_p (i_{dref} - i_d), i_{dref} = 0
\]

(37)

\[
v_d = K_p (i_{qref} - i_q), i_{qref} = \tau_{ref}/K
\]

(38)

Such a model is shown in the figure below using standard field-oriented implementation. The simulation of
the motor is shown on the right side while the simulation of the controller is shown on the left side.

To illustrate the how such a standard off line Simulink simulation is converted to real-time, the first step is shown in the figure below. The block labeled **SM_Master** contains the controller blocks from the above figure.

In more detail, clicking on this block gives the block diagram below. This should be compared with first figure in this section as appropriate I/O blocks have replaced the Simulink blocks of the motor/amplifier.

An experimental run in which the speed was brought up from zero to 600 rads/sec and back to zero is shown in the figure 12 below. The speed was computed by differentiating the feedback from the optical encoder. Figure 13 shows the speed computed using a speed observer. Figure 14 shows a plot of the quadrature current that was need to make the speed runs shown above.

The point here is to show how easy it is to take the SIMULINK blocks from the Library and turn them into a simulation and then into a real-time implementation.
INDUCTION MOTOR

The third motor for which a Library of simulation models have been developed is the induction motor. A standard two-phase equivalent model of the induction motor is given by [32]

\[
\frac{d\theta}{dt} = \omega \tag{49}
\]

\[
\frac{d\omega}{dt} = \mu \psi_d i_q - (D / J) \omega - \tau_L / J \tag{50}
\]

\[
\frac{dv_d}{dt} = -\eta \psi_d + \eta M_i \omega \tag{51}
\]

\[
\frac{di_d}{dt} = -\lambda i_d + (\eta M / \sigma L_R L_S) \psi_d + n_p \omega i_q + \eta M_i i_d / \psi_d + u_d / \sigma L_S \tag{52}
\]

\[
\frac{di_q}{dt} = -\eta \psi_q - (M / \sigma L_R L_S) n_p \omega \psi_d - n_p \omega i_q - \eta M_i i_q / \psi_d + u_q / \sigma L_S \tag{53}
\]

\[
\frac{dp}{dt} = n_p \omega + \eta M_i / \psi_d \tag{54}
\]

A standard approach is to use PI current controllers of the form

\[
v_d = K_d \int_{0}^{t} (i_{dref} - i_d) dt + K_{dp} (i_{dref} - i_d) \tag{55}
\]

\[
v_q = K_q \int_{0}^{t} (i_{qref} - i_q) dt + K_{qp} (i_{qref} - i_q) \tag{56}
\]

resulting in the reduced order model

\[
\frac{d\theta}{dt} = \omega \tag{57}
\]

\[
\frac{d\omega}{dt} = \mu \psi_d i_q - (D / J) \omega - \tau_L / J \tag{58}
\]

\[
\frac{dv_d}{dt} = -\eta \psi_d + \eta M_i \omega \tag{59}
\]

\[
\frac{di_d}{dt} = -\lambda i_d + (\eta M / \sigma L_R L_S) \psi_d + n_p \omega i_q + \eta M_i i_d / \psi_d + u_d / \sigma L_S \tag{60}
\]

\[
\frac{di_q}{dt} = -\eta \psi_q - (M / \sigma L_R L_S) n_p \omega \psi_d - n_p \omega i_q - \eta M_i i_q / \psi_d + u_q / \sigma L_S \tag{61}
\]

\[
\frac{dp}{dt} = n_p \omega + \eta M_i / \psi_d \tag{62}
\]

The dq current references are chosen as

\[
i_{qref} = \frac{\tau_{\text{ref}}}{\mu \psi_d} \tag{63}
\]

\[
i_{dref} = K_d \int_{0}^{t} (\psi_{dref} - \psi_d) dt + K_{dp} (\psi_{dref} - \psi_d) \tag{64}
\]

and flux magnitude \( \dot{\psi}_d \) and angle \( \dot{\psi} \) are estimated using

\[
\dot{\psi}_d = \sqrt{\dot{\psi}_{d_d}^2 + \dot{\psi}_{d_q}^2} \tag{65}
\]

\[
\dot{\psi} = \frac{\dot{\psi}_d}{\psi_d} \tag{66}
\]
CONCLUSIONS AND FURTHER WORK

This paper outlines the development of a library of SIMULINK blocks that can be used to implement state of the art controllers for switched reluctance, PM synchronous and Induction motors. The objective of this work is to provide developer of HEV controllers a readily accessible set of basic control algorithms for real-time implementation. Future work will concentrate on encoderless control schemes for the SR and Induction motors, parameter identification schemes for the induction and PM synchronous motors and power electronic control schemes including Carrier-based pulse width modulation (PWM), Space vector PWM and Harmonic elimination based PWM.

REFERENCES

2. Vas, Peter, Sensorless Vector and Direct Torque Control, Oxford Press, 1998. (See chapter 5)


