Use of Graphical Simulation Tools in the Design of Electric Drive Systems

Scott Ellerthorpe
Electric Propulsion Branch
Naval Undersea Warfare Center
Newport, RI 02871
(401) 841-3049
(401) 841-1131 [FAX]
scott@kraken.npt.nuwc.navy.mil

Dr. Duane Hanselman
Dept. of Electrical and Computer Engineering
University of Maine - Orono
Orono, ME
(207) 581-2246
(207) 581-2220 [FAX]
duaneh@eece.maine.edu

Abstract
Simulation has always been an integral part of engineering design. The recent advent of graphical tools has led to new applications in the design of electric drive systems. This paper will present a hierarchical approach to modeling. Examples will be presented illustrating application to system modeling, circuit design and software emulation. By using the same simulation platform for each of these tasks, highly-integrated models can be made, giving greater insight in subsystem performance and interaction.

1. Purpose

There is an ongoing trend in the design of electric drive systems towards simulation studies instead of prototyping. Drive systems can vary widely in size, and, in some situations, it can take several weeks to setup and perform a single test. Simulation studies are preferred to initially determine feasible architectures before the investment in hardware is made. After the system is fabricated, simulation can be used to reduce the number of tests that need to be performed.

This paper shows how hierarchy-based simulation models can be used in the design and evaluation of electrical drive systems. The advantage of a hierarchy is that the subsystem components can be changed at a macro level without extensive reworking of the system model. This allows new subsystem concepts to be easily evaluated against previous designs in the context of the overall system. Since the previous designs can be verified with experimental data, the engineer will have greater confidence in the new simulation results.

2. System Modeling

One of the traditional applications of simulation tools has been the modeling of system level performance. This is usually done at a macro-level using lumped, low-order models of system components. Overall system performance can than be predicted.

One advantage in this approach is that it easily lends itself to hierarchical modeling. As more detailed component models are developed, they can replace the lumped models while maintaining the reset of the system model. Thus, the value of higher complexity can easily be evaluated.

The first example is of a propulsion system (Figure 1), which was implemented using Simulink[1]. The load is primarily a propulsor, which requires a torque proportional to the square of the shaft speed. Adding inertia and damping, the dynamics are thus described by,

\[ \dot{\omega} = \frac{1}{J}(T_e - B\omega - A\omega^2). \] (1)

The motor model itself was greatly simplified. In fact, it is described simply as a generated torque proportional to the commanded current output of a PI regulator, as shown in (2). However, the simple gain block can be replaced by a more complex motor representation. Notice that the PI regulator is simply a block; the full model is one level down in the hierarchy.

\[ T_e = K_c(K_p(\omega^* - \omega)dt - K_i(\omega^* - \omega))(2) \]
The model in Figure 1 was used for two different purposes. Initially, the range of acceptable control gains was studied to determine a robust operating point. After the actual system was operated, the data was used to determine the accuracy of the load coefficients.

A comparison of the simulated and experimental step responses is given in Figure 2. Even with this greatly simplified model containing uncertain parameters, the general characteristics, such as rise time and overshoot, are still preserved. Two different systems were tested to obtain the experimental data. The simulation appears to match one motor more closely than the second, suggesting that the load on the second motor was different.

3. Circuit Simulation

A second area of application is circuit design. Both analog and digital circuits can be directly simulated. Analog circuits are identical to analog systems; the issue is how to correctly model digital equations. Digital equations have two different implementations. For combinatorial logic, the Boolean expressions can be directly implemented using zero for logic low and non-zero for logic high. Registered systems are handled the same as software simulations, discussed in the next section.

Consider the pulse-width modulation (PWM) circuit given in Figure 3, which contains both analog and digital circuitry. The circuit could be easily modeled in SPICE; however, integration into another model would be difficult. By using the same platform as for the system simulation, the circuit effects can be included in a detailed performance analysis.

The simulation model is given in Figure 4. The analog portion tries to track a current command using the PI control loop described by (3)-(4). The control output $U$ is then used in a ramp comparison scheme[2] to determine switching instances, which are put into a programmable array logic device(PAL) to determine the actual PWM signal.

$$I_{\text{error}} = I_{\text{ref}} - I_{\text{actual}}$$  \hspace{1cm} (3)

$$U = K_s (K_p I_{\text{error}} + K_i \int I_{\text{error}} dt)$$  \hspace{1cm} (4)

A comparison of the simulated circuit to the actual one is given in Figure 5. Features such as the phase lag and the zero crossing characteristic are identical. Since the control gains are set in hardware, the system performance can be examined without reworking the circuit board, saving time and expense.

4. Software Simulation

The final area of application to be discussed is software simulation. Earlier in the system simulation section, the software (PI Control) was represented as a block. This section will descend one level further into the hierarchy to examine how that block is implemented.

Software simulations are similar to analog ones, with the addition of the frame rate and quantization effects. The frame rate is the frequency at which the calculations are performed. Thus, continuous functions such as differentiation and integration require special consideration. For example, a backward-difference approximation[3] was used for integration(5) in this study. The variable $\frac{1}{H}$ represents the frame time.

$$\int \omega' dt = \frac{1}{H} \sum \omega_i$$  \hspace{1cm} (5)

Quantization effects occur since a micro-processor can only represent distinct values rather than a continuum. Small variation in analog signals may have the same digital value. For example, this software (Listing 1) uses a scale factor of 2 RPM/bit for speed. Thus, the actual speed can vary by one RPM from the internal value without any effect in the software calculations. The software in Listing 1 governs the speed of a pump. One of the design requirements was that the overshoot be minimized. This was accomplished by limiting the range and which the integration was performed to speed errors(6) below a threshold $\omega_{\text{int}}$. $C$ in (7) is one for errors within this range.

$$\omega' = \omega^* - \omega.$$  \hspace{1cm} (6)

$$C = u_i (\omega_{\text{int}} - \omega^*) - u_i (\omega^* - \omega_{\text{int}}).$$  \hspace{1cm} (7)
The speed is mainly governed by a PI regulator, with the integration range limited by (8). However, the load, in addition to inertia and damping, contains a term related to the pressure head. To compensate for this, a feedforward term proportional to the product of the speed and pressure head was included. The complete control algorithm is given by (8).

\[ I^* = CK_i \sum \omega_i^2 + K_p \omega_i + K_s P \omega_i. \]  

The simulation is given in Figure 6. Note that each input must be quantized and sampled. The calculations must also be quantized; however, the sampling is implicit as long as each of the inputs has been properly held.

```c
/**  Calculate the error. */
speed_error = PUMP_SPEED - motor_speed;

/**  The current integrator is only used in the neighborhood of the pump setpoint in order to minimize overshoot. This also allows a larger gain, which will lower the steady state error. **/
if((speed_error < WINT) && (speed_error > -WINT))
    /
    ***  Backward difference (Euler). ***
    current_integrator += Ki*H*speed_error;

    /
    ***  Bound the integrator. ***
    if(current_integrator > IMAX)
        current_integrator = IMAX;
    if(current_integrator < 0)
        current_integrator = 0;

    /
    ***  Zero the integrator for large speed errors. This should minimize overshoot. ***
    else current_integrator = 0;

    /
    ***  Sum the various terms to get the total commanded current. ***
    current_command = current_integrator +
        + Kp*speed_error + Kh*motor_speed*pressure_change;

    /
    ***  Bound the current command. ***
    if(current_command > IMAX)
        current_command = IMAX;
    else if(current_command < 0)
        current_command = 0;

Listing 1 - Software Control Law
```

5. Summary

Graphical simulation is a powerful tool for the design engineer. This paper illustrated by example some of the different methods by which simulation can be used in the design of an electric drive system, such as system modeling, circuit design and software emulation. Since these models are all done in the same platform, they can be easily integrated.

The hierarchical approach used with modular subsystems easily lends itself to comparative studies. For example, the software shown in Figure 6 can be formed into a single block with the gains entered as parameters. It can then be integrated into a larger system, such as a pump model. A simple PI block (see Figure 1) could also be used. Any performance differences would show the effect of the feedforward term in the controller.

Any form of simulation requires continual verification whenever experimental data is available. The verified models can then be used in future models with greater confidence.

References


Acknowledgments

The authors gratefully acknowledge the members of Code 8231 at the Naval Undersea Warfare Center for their support and assistance in preparing this paper.
Shaft Damping
Sensor Delay

Figure 1 - Propulsion System Macro Model

Figure 2 - Actual and Simulated System Response
Figure 3 - Circuit Schematic

Figure 4 - Hybrid Circuit Simulation Model

Figure 5 - Actual and Simulated Circuit Response
Figure 6 - Software Simulation Model