Global Powertrain Congress 2001

Validation of a Hybrid Modeling Software (PSAT) Using Its Extension for Prototyping (PSAT-PRO)

Aymeric Rousseau and Maxime Pasquier
Argonne National Laboratory

.

Abstract

Hybrid electric vehicles (HEVs) combine two sources of energy and offer a wide variety of component and drivetrain configurations. Moreover, optimizing the blending of these two energy sources is complex. Argonne National Laboratory (ANL), working in the Partnership for a New Generation of Vehicles (PNGV), maintains a forward-looking hybrid vehicle simulation software: The PNGV System Analysis Toolkit (PSAT). PSAT allows users to choose the best configuration and to optimize the control strategy in simulations. The importance of component models and the complexity of setting up optimized control laws require validation of models and control strategies developed in PSAT. In order to validate PSAT work, ANL developed PSAT-PRO, its extension for prototyping. In this paper, we describe the methodology used to develop the best control strategy using PSAT and to incorporate it into a vehicle controller using PSAT-PRO. We will then explain the process used to validate each component model with an actual component test using test stand facilities. Once each component model has been validated, ANL can perform some tests on a whole HEV using a chassis dynamometer. By comparing the simulated and measured data, we can demonstrate the PSAT component models and control strategy validation.

Introduction

The Partnership for a New Generation of Vehicles (PNGV), an historic public/private partnership between the U.S. government and the car manufacturers, was established to develop an environmentally friendly car to triple the efficiency of today's midsize cars. In order to respond to the needs of the System Analysis Team of the PNGV and the industry, Argonne National Laboratory (ANL) maintains the PNGV System Analysis Toolkit (PSAT), a forward-looking hybrid vehicle simulation software package.

One of the main challenges of hybrids is to choose the configuration and the components that are best suited to achieve PNGV goals. PSAT already includes many of the possible drivetrain configurations (about 180) to help PNGV and car manufacturers make the right decisions. However, in order to verify PSAT's usefulness, the component and the drivetrain models must be validated.

Validation is a very important aspect of software development, as it provides users the degree of accuracy of the software. Modeling tools can be validated using different data sources:

- From vehicle testing
- From component testing
- From drivetrain testing

ANL used all these methodologies to validate PSAT. However, if ANL Advanced Powertrain Test Facilities (APTF) was sufficient in the two first cases, the development of a specific tool dedicated to prototyping was necessary for drivetrain testing. In order to meet U.S. Department Of Energy (DOE) and PNGV needs, ANL developed PSAT-PRO, the extension of PSAT for prototyping.

In this article, we describe both PSAT and PSAT-PRO, the link between each software, and the methodology used to validate PSAT.

PSAT Introduction

PSAT was developed under the direction and with the contribution of Ford, General Motors, and Daimler-

Chrysler for the Partnership for a New Generation of Vehicle (PNGV).

Forward-Looking Model

PSAT is a powerful modeling tool that allows users to realistically evaluate not only fuel consumption and exhaust emissions for more than 20 different cycles, but also vehicle performances. Moreover, one of the most important characteristics of PSAT is that it is a forward-looking model, meaning PSAT allows users to model reality with real commands. PSAT is called a command-based model. In fact, we estimate the necessary wheel torque to achieve the desired speed by sending real commands to the different components, such as throttle for engine, displacement for clutch, gear number for transmission, or mechanical braking for wheels. In a way, we model a driver who follows a pre-defined speed cycle. Moreover, as components react to the commands as in reality, we can implement advanced component models, take into account transient effects (such as engine starting, clutch engagement/disengagement, or shifting) or develop realistic control strategies.

Conversely, in a backward-looking model, components cannot be controlled as in reality. The desired vehicle speed goes from the vehicle model back to the engine to finally determine how each component should be used to follow the speed cycle. Because of this model organization, quasi-steady models can only be used and, consequently, transient effects cannot be taken into account. Therefore, an accurate control application is not possible with a backward-looking model.

Looking toward the future, to be able to study transient effects and the interaction between components with accurate control commands, ANL developed a forward-looking model: PSAT.

Flexibility And Reusability

In a world of growing competitiveness, the role of simulation in vehicle development is constantly increasing. Because of the number of possible hybrid architectures, the development of this new generation of vehicles will require accurate, flexible simulation tools. Such a simulation program is necessary to quickly narrow the technology focus of the PNGV to those configurations and components that are best suited for achieving these goals. Therefore, the simulation should be flexible enough to encompass the wide variety of components and drivetrain configurations.

PSAT includes more than 150 predefined configurations, including conventional vehicles, parallel hybrids, series hybrids, fuel cell hybrids and power split hybrids. Users also have the possibility to choose two, four, and two times two-wheel drive. Such a capability is only possible by building all these drivetrain configurations according to user's inputs and component models from the libraries allowing users to choose the most appropriate configuration related to their requirements.

PSAT flexibility and reusability is based upon several characteristics, as described below.

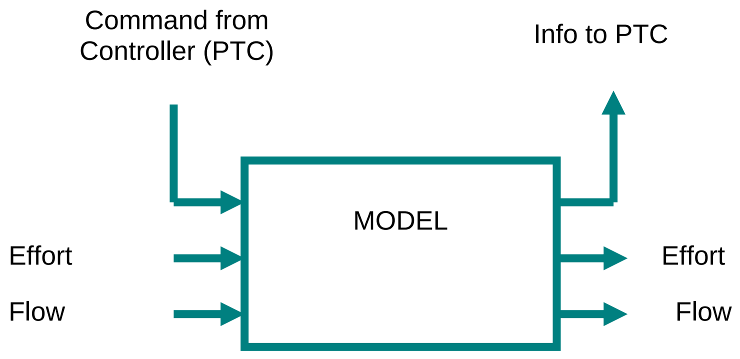
Components Models

Organization Format

In order to easily exchange the models and implement new ones, a common format, based on Bond Graph, is used between the input/output of the power ports, as shown in Figure 1. The first ports are used for the information:

- Input: components commands (i.e., on/off engine, gear number, etc…)
- Output (sensors): simulated measures (i.e., torque, rotational speed, current, voltage, etc.)

The second ports carry the effort (i.e., voltage, torque), and the last ones the flow (i.e., current, speed).



Effort = Torque, Voltage…
Flow = Speed, Current…

Fig. 1: Global Formalism for the Input/Output of the Models Using Bond Graph

Use of Library –

To ensure that the models we are using are the last ones changed or are not modified, we decided to use a library in which all the models are saved. Libraries enable users to copy blocks into their models from external libraries and automatically update the copied blocks when the source blocks change.

Use of Masks –

Hybrid electric drivetrain configurations can be very different from one another and also be rapidly complex. Often, one component can be used several times, such as the electric motor for the Precept from General Motors or the Prius from Toyota. In order to solve the problem of versioning, we decided to mask the different component models, allowing us to reuse the same model several times, as shown in Figure 2.

Use of GOTO-FROM Format –

To simplify the model, we decided to use the GOTO-FROM format. As far as the models are concerned, all the GOTO-FROM blocks are local and located at the upper level of the model (no blocks are located in the subsystems). To facilitate the work for Hardware In the Loop (HIL) (Control Desk access to the parameters and variables by using the Tags), the name of the Tags are defined using specific rules.

Use of common nomenclature for variable names –

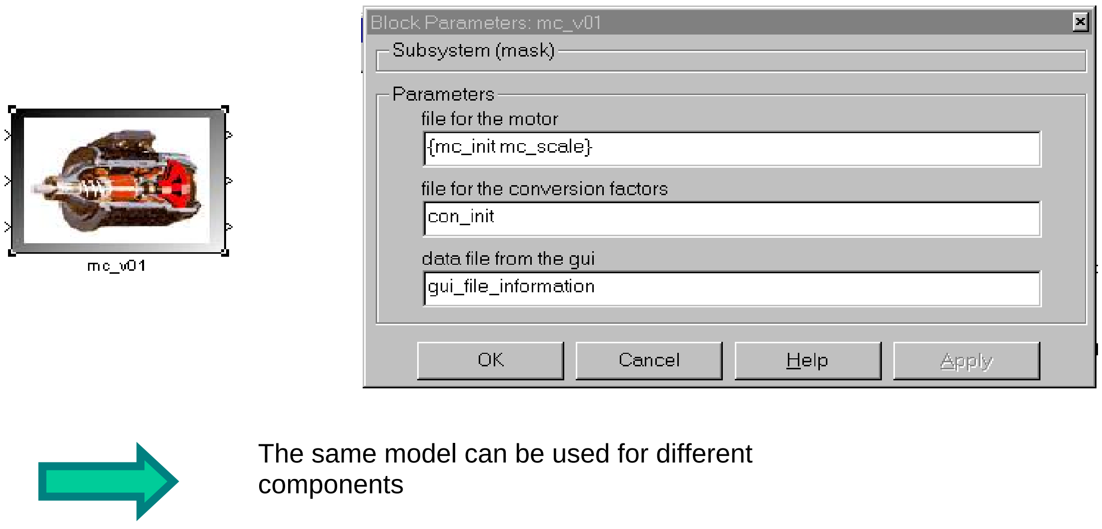
PSAT names have been parameterized and follow specific rules. At the component level, all the variables and parameters also follow established rules and are named according to the component and the type of data they represent. At the software level, the names are based upon the component (e.g., mc for motor/controller). In fact:

- The component model name is defined as ‘compo’_cm (e.g., mc_cm)
- The initialization file is ‘compo’_init,
- The scaling file is ‘compo’_scale,
- The calculation file is ‘compo’_calc,
- The parameter used to choose whether we scale is gui_scale_’compo’,
- The parameter used to scale the component is gui_’compo’,

Example of Drivetrain Configuration - Parallel Configuration Example –

As an example, it is interesting to look at a PSAT parallel configuration model to understand the benefit of using a library and standard format (Figure 3).

The driver output is a torque demand at the wheels, which is proportional to an accelerator or brake pedal command. This demand is sent to the powertrain controller, which decides how each component of the drivetrain should operate. Indeed, we make a choice concerning the blending between the different energy sources, such as when and how to start the engine or shift a gear.



file for the motor: {mc_init mc_scale}

file for the conversion factors: con_init

data file from the gui: gui_file_information

Fig. 2: Use of Mask Allows Reuse of the Same Models

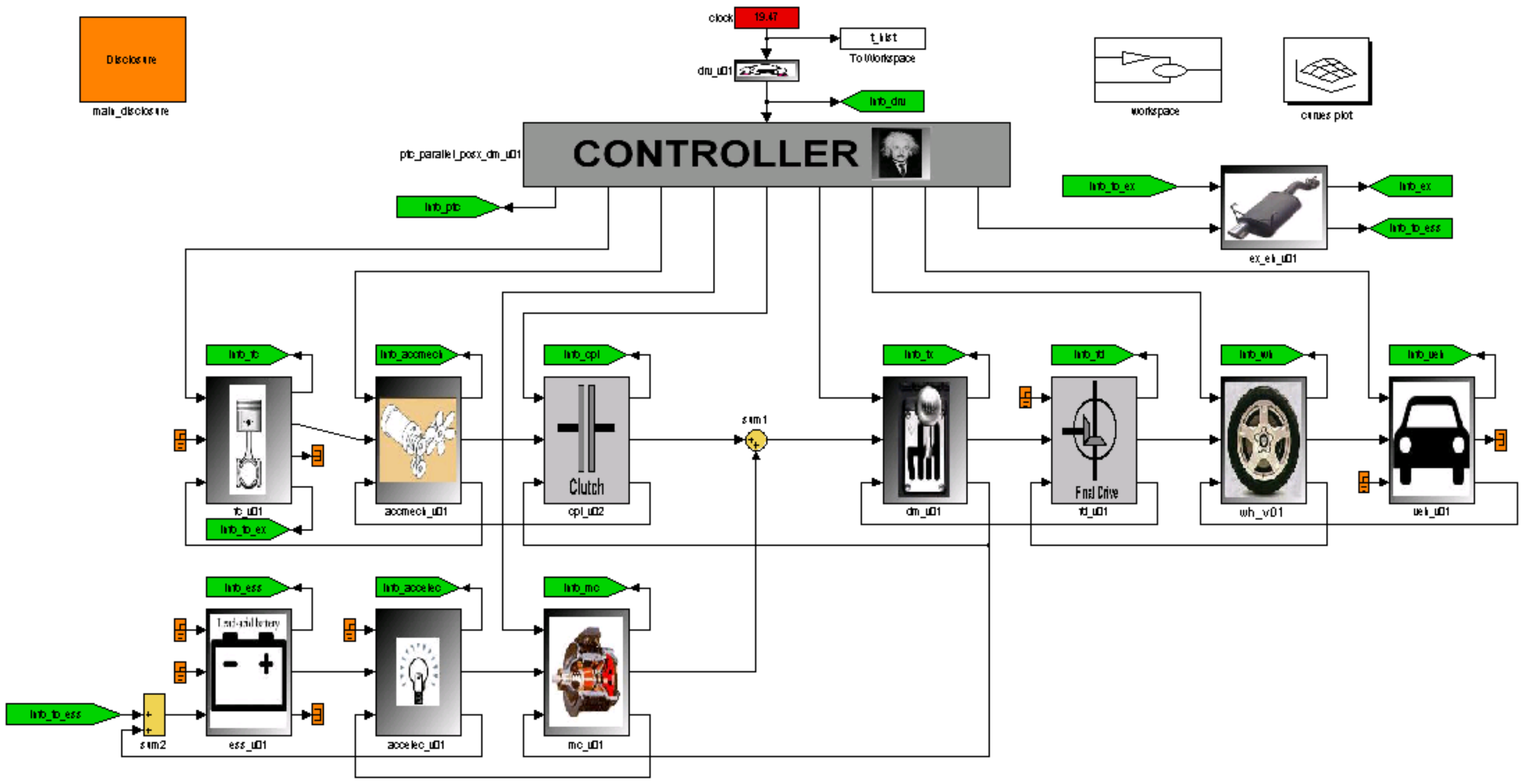


Figure 3: PSAT single shaft parallel hybrid

The powertrain controller will send specific commands to each drivetrain component: torque to the electric motor, throttle to the engine, displacement to the clutch, gear number to the transmission, and mechanical brake to the wheel.

Then, the mechanical power from the engine and the electrical power from the motor via the battery are summed. And in fact, both mechanical and electrical powers are used to propel the vehicle.As already mentioned, each component block has three inputs and three outputs and is linked to the library. These components models are reusable and perform as in reality. We can also use these components for control development: the first input will be the command, and the first output will be the information from the sensors.

Powertrain Controller Models

PSAT powertrain controllers, in charge of commanding the different components, have a generic structure common to all configurations. By using the accelerator pedal and the information (sensors) coming from the component models, we evaluate the constraints of the system, such as the maximum available torque of the engine. We then take those limits into account to define the optimized control strategy, which allows us to use the powertrain components to minimize fuel consumption and emissions. Finally, we take the transients into account by defining the actions necessary to satisfy the control strategy demands. For instance, if the control strategy decided to shift gear with a manual transmission, we must cut off the engine ignition, declutch, engage the neutral gear, engage the new gear, clutch, and inject once again. These steps must happen successively and lead to a modification of the demands previously sent by the control strategy.

Within PSAT powertrain controller, different strategies can be selected within a particular powertrain model. Indeed, as the strategy has an important impact on the fuel consumption, it is interesting to switch between different control strategies for comparison. For instance, the engine start can be based on different parameters, such as vehicle speed, power demand, torque demand, level of State Of Charge (SOC) or a combination of these. In order to evaluate the impact of these different strategies, we can select and compare them using the same components models. This also allows us to easily implement new strategies.

PSAT-PRO Introduction

PSAT-PRO: generic and reusable

PSAT-PRO must be generic and reusable enough to control any kind of configurations (conventional or hybrids). The structure of PSAT-PRO has been developed to be the same with any configuration. Three main blocks compose PSAT-PRO model:

- Test procedure,
- Control command system,
- Physical system.

Users can define commands to apply to the system in the test procedure. Then, the test procedure sends the command bus to the control command system, which is in charge to effectively apply those commands to each component in accordance with the system constraints. The physical system represents the vehicle model composed by the components models, either Digital to Analog (DAC) and Analog to Digital Converters (ADC), depending on the process phase (simulation or rapid prototyping). In fact, the vehicle model should react exactly as the actual system as we send the same commands. Differences come from the converters: when the user controls the actual system, the commands must be converted to communicate with the actuators controllers. The data sent by the sensors must be translated in a digital format to be able to use the measures. Figure 4 shows the three main blocks of the PSAT-PRO model.

The command bus goes from the test procedure, through the control command system, to the physical system. The physical system reacts by sending the measure bus to the control command system through the test procedure. Using ControlDesk, we have access to the commands and the measures in only one block: the test procedure.

The test procedure

The command bus is created in the test procedure, and the measure bus is received. For each actuator, we have a command, or several, depending on the mode used. For instance, a motor can be controlled in torque or in speed mode if a speed regulation exists in the motor controller. So in this case, we create a torque command, a speed command, and a mode command in the bus command. This bus command will be modified later in the control command system to process the information.

The control command system

The control command system is the brain of the system, as each component control is decided here. It includes several functions:

1) As the command and the measure buses go through the control command system, a function to continuously check if measures reach commands has been developed. If not, a warning is sent. With this function, we also check if the commands are possible (i.e., if the user asks commands in accordance with the components constraints). We can also check if the measures are not above the constraints (i.e., over speed, over heat, etc.). In each possible case, we send a warning to the emergency function, which modifies the commands in order to follow the right emergency procedure corresponding to the warning.

2) The components constraints are calculated in the constraints function, which takes into account each component of the power train and their interaction. For instance, the engine constraints can affect the motor range depending of the gear ratio.

3) A parameter function has been developed in order to calculate the values needed but not measured or not measurable, such as the SOC.

4) The control command system also includes a function, which is in charge of defining the system status, such as the motor's direction.

5) To decide how to satisfy the users' demands in an optimal way, we include the PSAT control strategy and transient block as a function. The details on how to incorporate the control strategy explained later in this paper.

6) Before sending the commands, a saturation function is in charge to limit the commands with the appropriate constraints to ensure we always send acceptable commands.

The physical system

The physical system receives the commands from the control command system. The physical system model is built using the PSAT structure and components models described previously. When users want to use Rapid Prototyping, the components models are replaced by the actual components, and a DAC and ADC are added. The physical system is shown in Figure 5. The bus command is converted to send analog values to the actuators controllers through the dSPACE board outputs. The converted analog values coming from the dSPACE board inputs create the measures bus.

From PSAT Modeling to PSAT_PRO Prototyping – a Generic Process

As PSAT is a forward-looking model, it is well suited to accurately control the powertrain components. Moreover, because PSAT includes many configurations, users can develop their own control strategies for the specific powertrain studied. Then, PSAT-PRO control software allows us to download it in the vehicle controller. In this section, we describe in detail the three steps necessary to reach this goal:

- Simulation
- Hardware In the Loop (HIL)
- Rapid prototyping

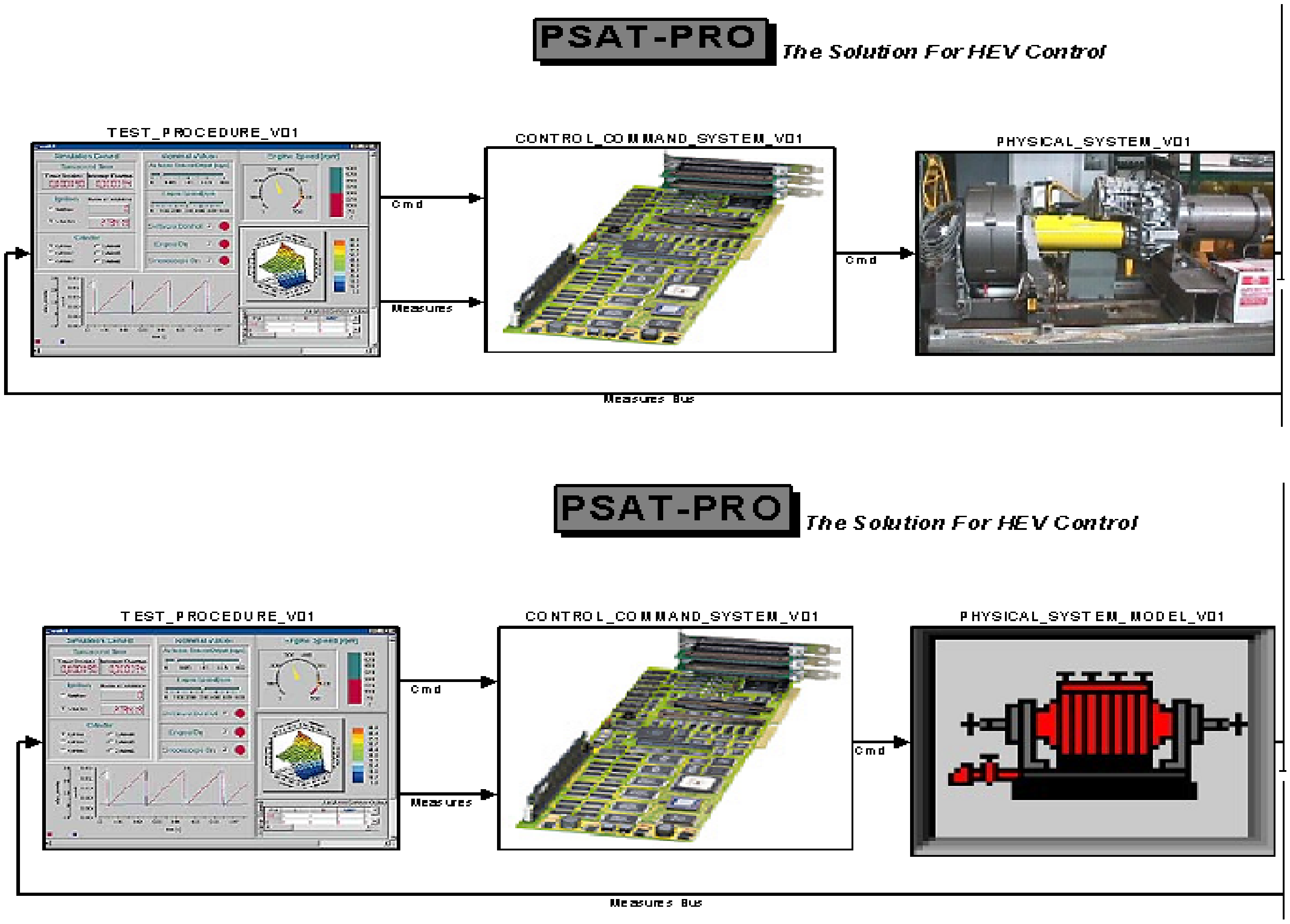

PSAT-PRO
The Solution For HEV Control
TEST_PROCEDURE_V01
CONTROL_COMMAND_SYSTEM_V01
PHYSICAL_SYSTEM_V01
Cmd
Measures
Cmd
Measures Bus
PSAT-PRO
The Solution For HEV Control
TEST_PROCEDURE_V01
CONTROL_COMMAND_SYSTEM_V01
PHYSICAL_SYSTEM_MODEL_V01
Cmd
Measures
Cmd
Measures Bus

Fig. 4: PSAT-PRO model and real physical systems

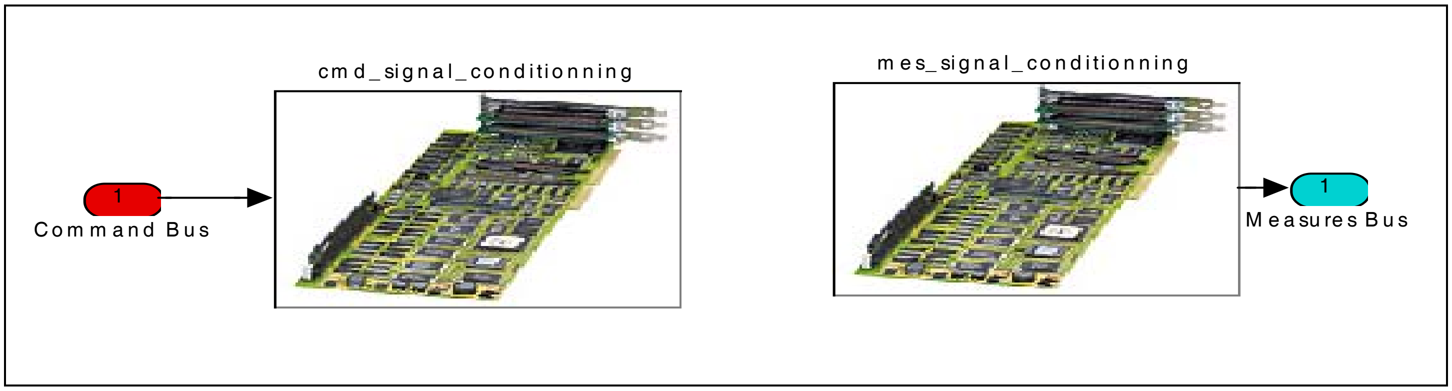


Fig. 5: PSAT-PRO Physical System for Rapid Prototyping

The purpose of PSAT-PRO is to allow users to go from modeling, using PSAT, to prototyping, for any kind of vehicle or configuration. As with PSAT, PSAT-PRO must also be generic and reusable. Moreover, both softwares must be linked to directly integrate the control strategy optimized in PSAT. Finally, we explain how the comparison between the test data and the simulation results can provide modeling validation and improvement.

A Process In Three Steps

Simulation

Using PSAT, we simulate the vehicle performance in order to define the best drivetrain, select the appropriate component technology and size, and develop the control strategy in accordance with the specifications. In PSAT-PRO, we model the physical system exactly as in real life, including signal conditioning or a dynamometer model for testing the powertrain on a test stand. The goal is here to minimize the modifications of PSAT component models to later facilitate the validation of those models. To verify the performance of the physical system model that will be later be controlled on a bench or in a vehicle, a simulation phase must be performed, following the test procedure decided upon earlier.

Hardware In the Loop

Once the PSAT-PRO model has been verified, Real-Time Workshop is used to automatically generate C code from Simulink block diagrams. Generated code is applied in a variety of applications such as real-time embedded control, DSP design, HIL, and stand-alone simulation. The generated C code is portable and runs on many floating-point processors. Once the PSAT-PRO model is downloaded on the processor, we can apply commands, observe in real time, and capture data from sensors using ControlDesk. ControlDesk is dSPACE's new experiment software, which provides tools for controlling, monitoring, and automating real-time experiments. The development of controllers is then more effective.

In this phase, we verify the test procedure in real time in order to finish the control checking. In fact, the control is downloaded in a microcontroller running at 10 Hz and the physical system in another microcontroller running faster. The two systems communicate together exactly as in reality using the appropriate signals. The faster microcontroller has to exactly represent, in real time, the actual system behavior. The control strategy can then be checked in a safe way.

Rapid Prototyping

After control is checked both in simulation and in real time, the last phase consists of replacing the modeled component with the real ones from the test stand or the car. Using ControlDesk, PSAT-PRO proposes a graphical user interface to control each component of a hybrid powertrain. Figure 6 provides an example of the PSAT-PRO ControlDesk layout.

PSAT-PRO is linked to PSAT

The Physical System

The physical system model is built using the PSAT components models library. In fact, libraries enable users to copy blocks into their models from external libraries and automatically update the copied blocks when the source blocks change. Libraries allow users to access models provided by others (such as PSAT) to ensure that their models automatically include the most recent versions of these blocks. By using similar component models as in PSAT, we then facilitate the validation process of the different models.

The Control Strategy

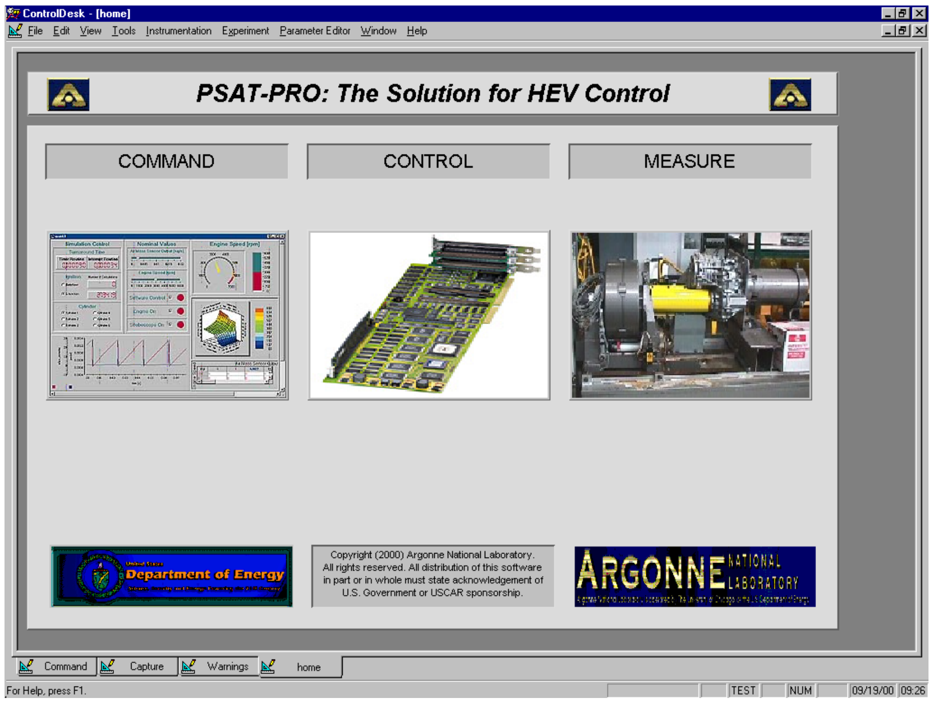
As described before, the PSAT-PRO control command system includes several functions. One function is the control strategy developed in PSAT. As both softwares use the same structure, the PSAT control strategy is “plug and play”. We simply need to copy the control strategy and the transient block previously developed for simulation purposes into the specific control that will later be used for HIL or rapid prototyping. It is very interesting to develop a control strategy based on simulation and to test its efficiency in real life. Once again, using this methodology, we facilitate the validation process.

As PSAT-PRO is linked to PSAT, it becomes easy to integrate any PSAT control strategy in the PSAT-PRO control command system. PSAT-PRO allows users to directly integrate the control strategy optimized in PSAT for any kind of configurations.

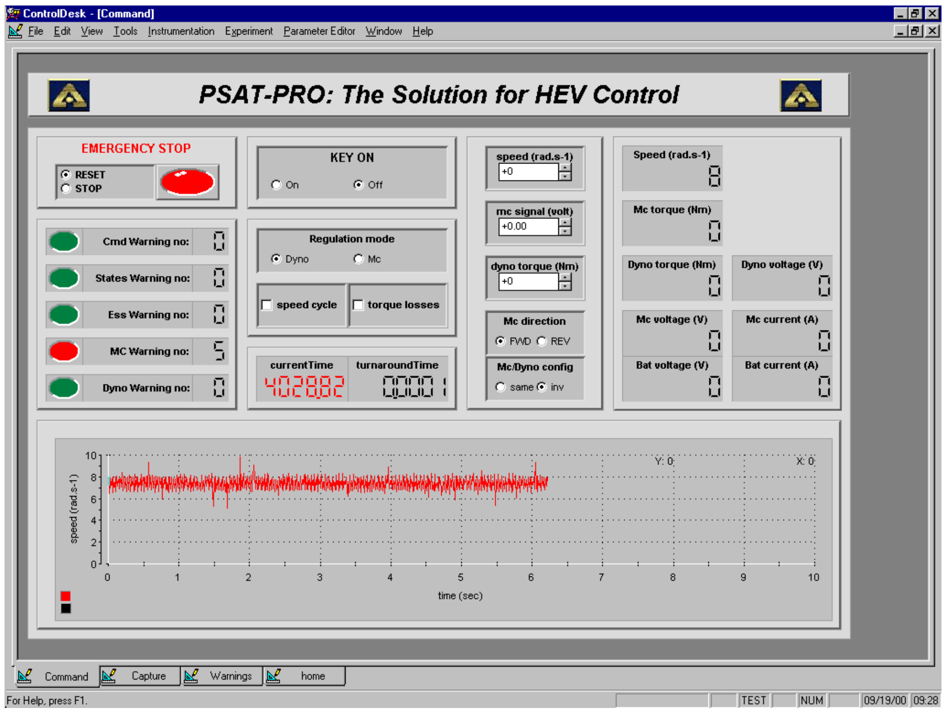
Calibration Process

Simulation Results And Measures

The calibration process is used to improve the physical system model accuracy. The principle is simple: as we use the same test procedure for each step (Simulation, HIL, rapid prototyping), the physical system, real or simulated, should react exactly the same way. In one hand we have the simulation results, and in the other hand, we have the measures. In order to compare the results and to analyze the differences, the measures collected are transformed in the right format and then using graphical tools, differences are analyzed and explained.


ControlDesk - [home]
File Edit View Tools Instrumentation Experiment Parameter Editor Window Help
PSAT-PRO: The Solution for HEV Control
COMMAND
CONTROL
MEASURE
Department of Energy
ARGONNE NATIONAL LABORATORY
Command Capture Warnings home
For Help, press F1.
TEST NUM 09/19/00 09:26

Copyright (2000) Argonne National Laboratory. All rights reserved. All distribution of this software in part or in whole must state acknowledgement of U.S. Government or USCAR sponsorship.



[✓ RESET] STOP

Cmd Warning no: 0
States Warning no: 0
Ess Warning no: 0
MC Warning no: 5
Dyno Warning no: 0

KEY ON: On [✓ Off]

Regulation mode: [✓ Dyno] Mc

speed cycle ☐ torque losses ☐

currentTime: 402882 turnaroundTime: 00001

speed (rad.s-1): +0
mc signal (volt): +0.00
dyno torque (Nm): +0

Mc direction: [✓ FWD] REV

Mc/Dyno config: same [✓ inv]

Speed (rad.s-1): 8
Mc torque (Nm): 0
Dyno torque (Nm): 0
Dyno voltage (V): 0
Mc voltage (V): 0
Mc current (A): 0
Bat voltage (V): 0
Bat current (A): 0

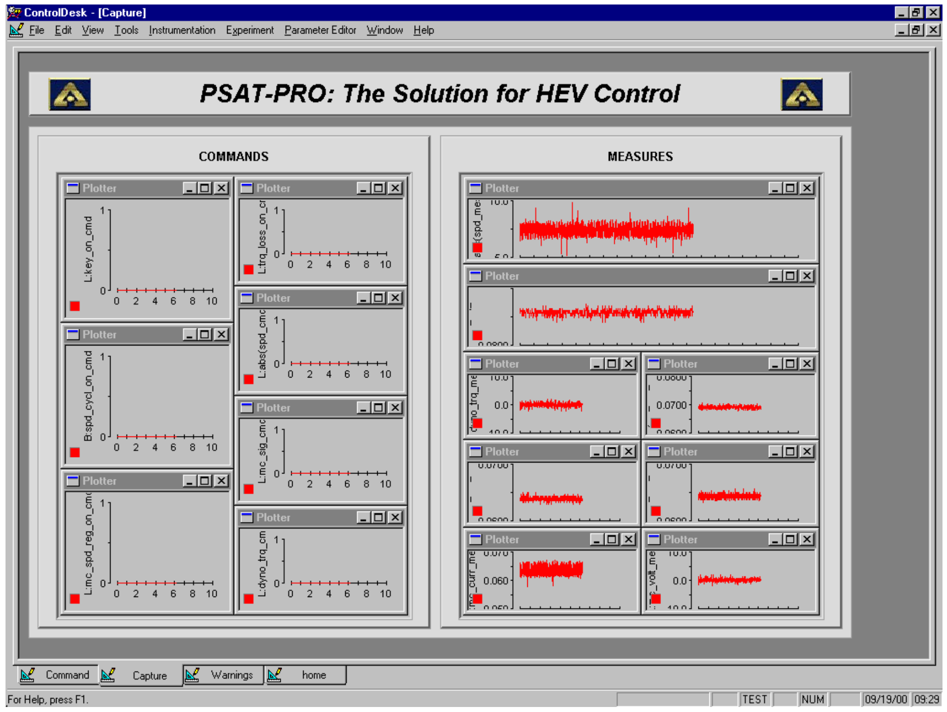


Figure 6: Example of PSAT-PRO ControlDesk's Layout

A Closed Loop Process

If necessary, the model is modified in order to improve its accuracy and to be closer to reality. After the modifications, we rerun the test procedure in simulation and compare again the results and the measures. This loop can be performed several times depending on the precision level required.

Physical System Model Calibration

When we perform a test in the rapid prototyping phase, we save the commands from the test procedure. So, in simulation, we are able to apply these commands to the physical system model. In this way, we are sure the differences can come only from the physical system model, as we use the same control command system.

Data Acquisition

With the ControlDesk data management tool, we capture all the data used on the model. This means we have access to the commands bus and the measures bus. For each test performed, we are able to save a file in a Matlab format containing the measures and the commands. We then integrate the data in our Simulink model to perform the same test in Simulation. Then, we use a program based on graphical analysis to compare the results. So, we are able to compare the differences between the simulation results and the rapid prototyping results (measures).

Conclusion

Because of the number of possible hybrid architectures, the development of this new generation of vehicles will require simulation tools. Moreover, the complexity of this new generation of vehicles imposes the use of an intelligent control. The PNGV System Analysis Toolkit (PSAT) allows users to choose the appropriate configuration, component sizes, and technology and to optimize the control strategy in simulation for a wide range of HEVs. But to control the real vehicle using this optimized control strategy, a more complex control command system is needed. The number of components to control increases the control complexity. That's why three steps are necessary to complete the integration of the control in the vehicle. PSAT and PSAT-PRO tools offer users the unique possibility to develop their own control strategy, optimize it, integrate it in a prototype, perform tests, calibrate the model, and evaluate the efficiency of their control strategy. Moreover, as PSAT and PSAT-PRO share the same organization and models, this process is reusable and can be applied to any kind of vehicles.

Acknowledgments

The authors are grateful for the support given by the U.S. Department of Energy (DOE) for funding our work, the Partnership for a New Generation of Vehicles (PNGV) and USCAR for their support and guidance.

References

ControlDesk 2.0. Experiment Guide. DSPACE GmbH.

DeCharentenay, F.; Delhom, M.; and Rault, A.. 1996. Seamless Mechatronic Design of an Electric Vehicle Powertrain Convergence 96, pp 413-421.

Karnopp, D.; Margolis, D.; and Rosenberg R. 1990. *System Dynamics: A Unified Approach*, 2nd ed. John Wiley & Sons, Inc., New York.

Matlab V5.3.1/Simulink V3.0 Users Guide. The Mathworks, Inc.

Rimaux, S., Delhom, M., Combes, E., and Rault, A. 1998. Hybrid Vehicle Powertrain: Modeling and Control.

Rousseau, A., and Pasquier, M., "Validation Process of a System Analysis Model: PSAT". SAE Paper 01-P183.

Rousseau, A., and Larsen, R., **2000,** Simulation and Validation of Hybrid Electric Vehicles Using PSAT. GPC Conference, Detroit, Mich., June 6-8.

Contacts

Aymeric Rousseau
Research Engineer
Center for Transportation Research
(630) 252-7261
E-mail: arousseau@anl.gov

Maxime Pasquier
Research Engineer
Energy Systems Division
(630) 252-9717
E-mail: mpasquier@anl.gov