Thermal Modeling in the Powertrain Analysis and Computational Environment (PACE)

John Gabriel Monroe, Matthew Doude, Tomasz Haupt, Gregory Henley, Angela Card, and Michael Mazzola
Mississippi State University, USA

Scott Shurin
US Army TARDEC, USA

Christopher Goodin
US Army ERDC, USA

John.G.Monroe@erdc.dren.mil

ABSTRACT

Ready for a High-Performance Computing (HPC) environment, the Powertrain Analysis and Computational Environment (PACE) is a forward-looking, C++ simulation tool that provides advanced behavioral modeling of the powertrain subsystem of a hybrid-electric or conventional vehicle. More specifically, PACE allows a power train simulation created in Matlab/Simulink to be ported into a high-performance cluster computing environment. Previously results with PACE involved advanced hardware and control features simulation of modern vehicle architectures, but it did not include comprehensive thermal modeling of power train components. Thus, a lumped-mass thermal model has been developed in Simulink separately from the original power train model used in PACE. The thermal model is integrated into PACE using PACE's built-in automated code generation functionality and taking advantage of the PACE powertrain model's modular structure. The integrated thermal model's output data demonstrates accurate results as compared to the standalone thermal model developed in Simulink. This paper reports on the workflow and validation of a new capability added to PACE that allows third-party (meaning non-Autonomie derived) external models to be integrated into PACE.

1.0 INTRODUCTION

The advancement of military ground vehicle technology necessitates state-of-the art simulation tools to design and model systems that may not be physically realized; this is especially true for advanced fighting vehicles. To this end, the U.S. Department of Defense's CREATE-GV program is developing a comprehensive, multi-physics simulation environment called MERCURY to facilitate high-resolution exploration of ground-vehicle designs. MERCURY implements independent modules called 'federates' in parallel to model all aspects of military ground vehicle mobility. An overview of the MERCURY Co-simulation Framework is given in Figure 1. Because the MERCURY federates are co-dependent, solving the high-resolution model requires a high-performance cluster computer.

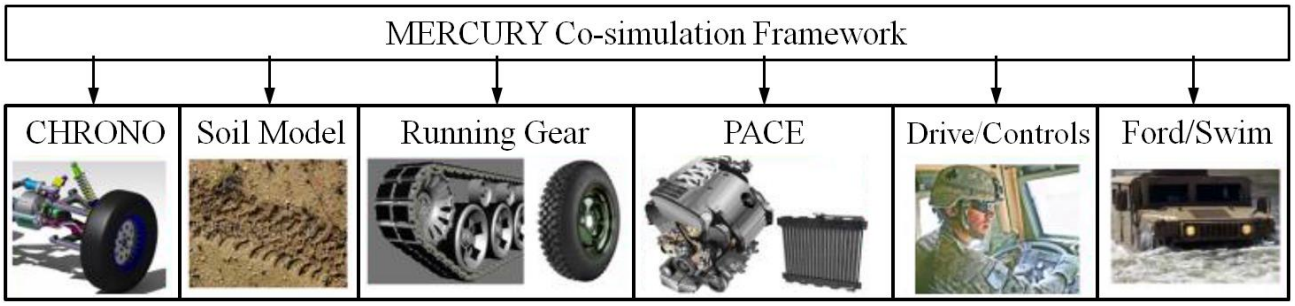

Figure 1. The Mercury Co-simulation Framework.

The powertrain federate in MERCURY is the Powertrain Analysis and Computing Environment (PACE) [1]. PACE provides MERCURY with a comprehensive powertrain behavioral-model that includes all components typical in a hybrid vehicle (engine, energy storage, transmission, electric motor, etc.). In the original report [1] the powertrain model was built in the Autonomie modeling environment, which is a forward-looking modeling tool developed by the Argonne National Laboratory [2]. Autonomie is based in Matlab/Simulink and was created for the design and simulation of civilian vehicle powertrains. Autonomie is a validated [3–5] design tool that provides accurate simulation of transient powertrain hardware and control features. However, PACE does not replicate Autonomie models. Instead, it can convert a specific powertrain architecture developed in Autonomie into C++ code that operates independent from Autonomie (a Microsoft Windows application) as a domain-specific federate (i.e., functional module) of MERCURY running on Linux operating systems typical of cluster computers. But it can also create or modify a powertrain federate for MERCURY from Matlab/Simulink simulation blocks independent of Autonomie. The thermal model described in this paper is one example.

The present work outlines a lumped-mass thermal model of a hybrid vehicle powertrain built in Simulink outside of the Autonomie environment. This model was then integrated into PACE by automatically generating HPC-ready C++ code from the Simulink SLX file, a development reported in a companion paper. Although PACE is only one federate in the MERCURY environment, the integration of the thermal model demonstrates the ability of the automatic SLX parser to convert 'third party' Simulink modules into HPC-ready C++ code for integration into MERCURY in general.

2.0 DEVELOPMENT OF INDEPENDENT POWERTRAIN COMPONENT MODEL

The current PACE powertrain model consists of approximately 20 modules such as engine, gearbox, battery, etc. The physical parameters of each component module may be adjusted as part of the typical PACE workflow. However, within the PACE architecture, any of these modules can also be exchanged for externally developed models, provided the replacement models provide the same inputs and outputs. The existing modules can also be modified and augmented, such as with the thermal model functionality demonstrated in this paper. The thermal model outlined below was implemented as an addition to an existing Simulink model already used in PACE.

2.1 Model Description

The thermal model contains two coolant loops, each with its own pump: one for electrical devices (i.e. battery, power converter, and electric motor) and one for the engine. The four system components are modelled as transient lumped mass systems, i.e. only temporal temperature gradients are considered. Two coolant loops are used due the disparity between the engine and electrical operating temperatures (e.g. 100°C vs. 50°C), but the two radiators are placed in series with a single fan. The temperatures of the pumps themselves are not

tracked, but the heat loss from the pumps is calculated and added to the appropriate coolant flow. A general schematic of the Simulink thermal model is given in Figure 2.

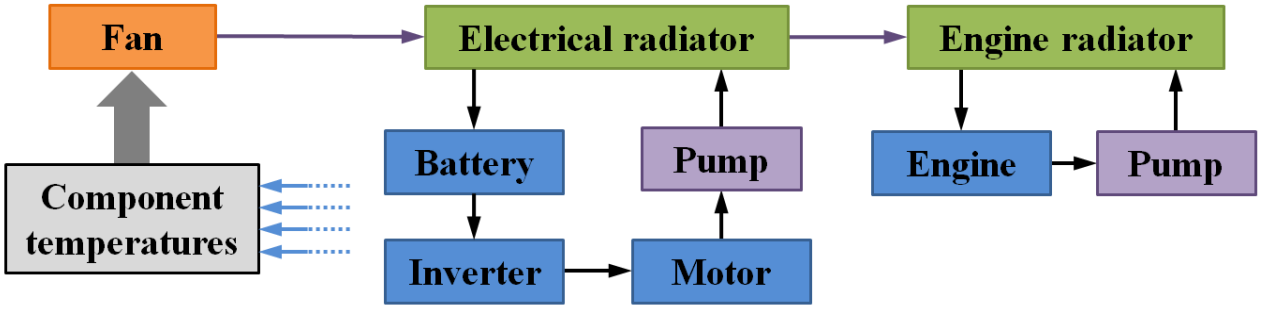

Figure 2. Schematic of Simulink thermal model.

The governing equations for the transient subsystems are given in Figure 3 [7]. By energy conservation, thermal energy produced by a component (q_{loss}) is either stored in the component (q_{str}) or transferred to the coolant (q_c). All of the thermal model's thermodynamic properties are assumed constant, but the overall heat transfer coefficient, U, in Figure 3 is a function of coolant flow rate (which is assumed constant for a given simulation).

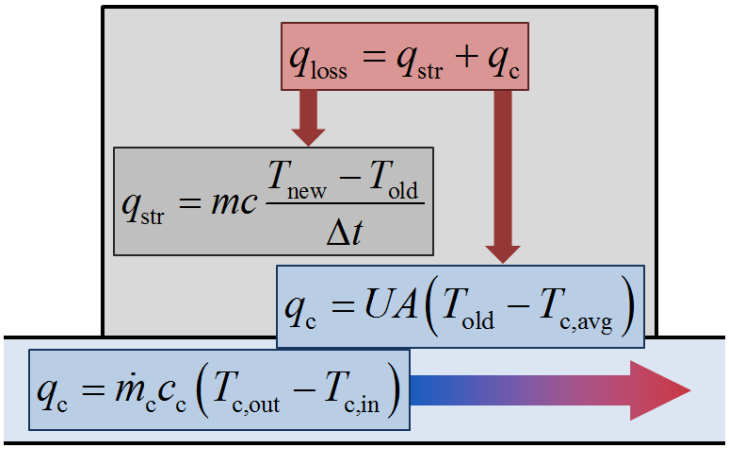

Figure 3. System of equations for transient subsystems

In the context of Figure 2, the important outputs of Figure 3 are the new component temperature, T_{new}, and the outlet temperature of the coolant, $T_{c,out}$; the former is tracked to determine if the component's operational or maximum temperature has been reached, while the latter is an input for the next component in the coolant loop. The equations in Figure 3 can be combined to explicitly solve for T_{new} and $T_{c,out}$ in terms of known values and the intermediary variable q_c (i.e. heat added to the coolant).

At each simulation time step the temperatures of the battery, power converter, electric motor, and engine are checked against specified operating and maximum temperatures (T_{op} and T_{max}, respectively). Simple logic gates are used to cause the fan speed to increase if a component temperature is above its T_{op}; this acceleration continues until $T_c < T_{op}$ or a specified maximum fan speed is reached. The cooling performance of the two

radiators varies as a function of air flow rate, and either cooling loop can trigger an increase in fan speed. If the cooling load decreases, or the component temperatures decrease for some other reason (e.g. lower ambient temperature), the fan speed will slowly reduce to conserve energy until a component's operating temperature is again exceeded.

2.2 Model Implementation

Once finished, the thermal model was integrated with the Simulink representation of the Autonomie model as a 'secondary plant' in the "mechanical accessory" module. Although a second plant is unique within the PACE model architecture, this implementation was beneficial from the perspective of the overall PACE architecture because it provided easy access to the data required by the thermal model and it conforms to the logic of the automated Simulink-to-C++ process – i.e. the definition as a plant well identifies the XML branch to be processed (as is described below). Note that the "integration" of the thermal module was done at the level of the PACE driver routine. To this end all buses were removed, and a direct map was created between output and input ports of the subsystems. Currently, the thermal model is not integrated with other power controllers to initiate component performance truncation when component begin overheating.

Although rudimentary, the thermal model demonstrates the original functionality of PACE 1.0 being augmented with an independently-developed model. Furthermore, this development occurred entirely in Simulink. It will be shown that adding the thermal model is almost trivial from the perspective of generating C++ code since no distinction is made between the thermal model and any another (i.e. original) module.

3.0 INTEGRATION OF INDEPENDENT MODEL INTO PACE

From the PACE vantage point, the thermal model is just another powertrain component, a peer to engine plant, gearbox controller, and others, implemented as a Simulink block of type *Subsystem*. Consequently, it is integrated into PACE in the same way as the other components. The details of this process are described in related papers focused on the PACE implementation [1], [8]. Here, for the sake of completeness, a brief summary is given.

The integration is done in two steps: first, the C++ implementation of the Thermal Model subsystem is generated, which results in a standalone C++ class, and then, the PACE driving routine is modified: the Thermal Model object is instantiated, initialized, and added to the PACE's workflow, so that it is executed at each time step of the powertrain simulation fed by the current values of the Thermal Model input variables resulting from execution of the other components.

Generation of the C++ code (step one) is automated - that is, preformed programmatically - by transforming its XML description: (hierarchical) list of Simulink blocks, their parameters, and connectivity. The transformation involves "translation" of an XML t<block> tags into executable code: an assignment, a simple expression (e.g, summation or multiplication), and/or a call to a function from an independently developed C++ library. The library is a part of the PACE system, and it comprises implementation of the basic Simulink blocks (such as integrator, switch, and lookup_nD). The library has been developed applying the common knowledge of numerical methods as guided by the Simulink documentation.

The process of the generating C++ code from a Simulink subsystem is very generic, and nothing that is specific to the thermal modeling was needed to be added in order to incorporate the Thermal Model described in this paper into PACE. The successful integration of it shows the flexibility of our approach: any valid component

developed as Simulink subsystem can be easily assimilated with PACE. Of course, the models can be developed directly in C++, skipping the "translation phase" (step one) altogether. However, this requires the model developer be not only fluent with C++, but also be familiar with the PACE software architecture. The capability of developing new models in Simulink makes PACE more accessible to automotive engineers.

The Simulink implementation of the Thermal Model has been integrated with the rest of the Simulink implementation of the Powertrain model. Although it is not necessary, it is advantageous. Firstly, it uniquely defines the integration of the Thermal Model with the rest of the system – thus guiding the step two of the integration process. Secondly, it serves as the baseline for our PACE verification and validation process, as described below.

4.0 FUNCTIONAL VALIDATION

4.1 Thermal Model Results

To demonstrate the model's behavior, the component temperatures and the air flow rate (as set by the fan) are plotted for four combinations of engine power loss and ambient temperature. The electrical loop components' heat losses are kept constant across the four simulations and are set sufficiently low as to not trigger an increase in fan speed. This allows the clear demonstration of the fan logic in direct response to engine temperature alone. The simulation constants for the system components (including the engine) are given in Table 1. The coolant (water) flow rate for the two loops is kept constant at 0.378 kg/s and 4.54 kg/s for the electrical and engine loops, respectively [8].

Table 1. Simulation constants for components

	T_{init} (°C)	T_{op}/T_{max} (°C)	m (kg)	c_p (J/kg·K)	q_{loss} (W)
Battery	25	45/50	130	910	450
Inverter	25	45/50	15	910	30
Motor	25	45/50	50	910	125
Engine	25	90/100	200	910	

The transient component temperatures and the resulting air flow rate are given for $q_{loss,eng}$ = (80, 120) kW in Figure 4 and Figure 5 at T_{amb} = 20°C and 40°C, respectively. In the model, the air flow rate (i.e. fan speed) increases and decreases at different rates. In the cases shown in Figures 4 and 5, the flow rates change at 0.756 m^3/s^2 and 0.0378 m^3/s^2 ascending and descending, respectively (i.e. 0.5 s and 10 s for a full range change). However, these rates (m^3/s^2) could be changed based on the simulation time step and the specifications of the fan being modelled.

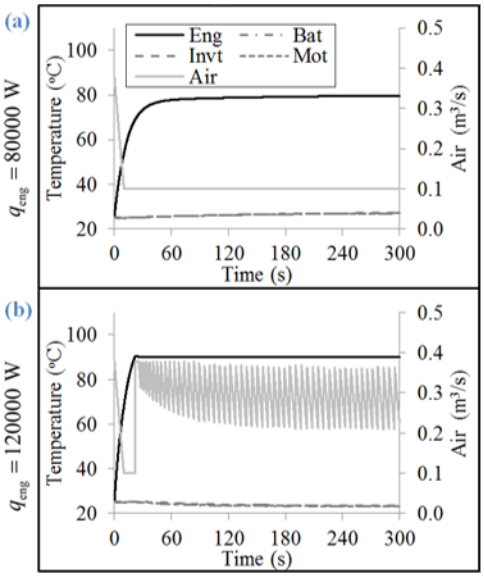

Figure 4. Simulink simulation results of component temperatures and air flow rate for engine heat loss of (a) 80 kW and (b) 120 kW at T_{amb} = 20°C

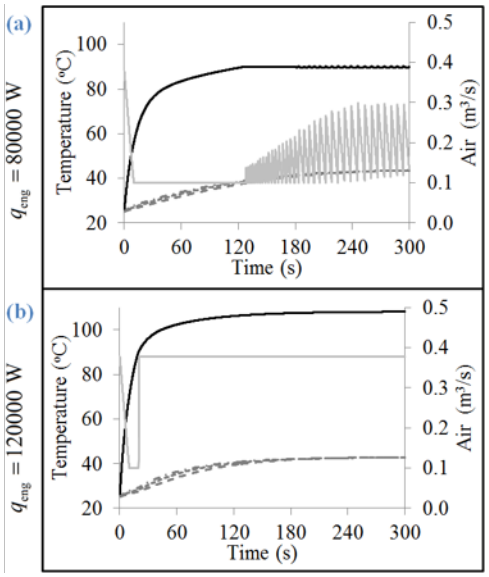

Figure 5. Simulink simulation results of component temperatures and air flow rate for engine heat loss of (a) 80 kW and (b) 120 kW at T_{amb} = 40°C.

For all simulations, the air flow rate is initialized at its maximum value of 0.378 m^3/s. Because the initial component temperatures are all below their respective operating temperatures, the air flow decreases for the first 10 s before reaching the set minimum air flow rate of 0.1 m^3/s. In Figure 4(a) the low q_{eng} and cool ambient temperature results in the air flow rate remaining at 0.1 m^3/s for the duration of the simulation since the steady state engine temperature is below $T_{op,eng}$. However, for both Figure 4(b) and Figure 5(a), the engine temperature surpasses $T_{op,eng}$. causing the air flow rate to increase. Once $T_{eng} < T_{op,eng}$ due to the increased cooling, the fan speed begins to drop again. This cycle is repeated for the remainder of the simulation. In Figure 5(b), the maximum air flow rate is insufficient to keep $T_{eng} < T_{op,eng}$ (or even $T_{max,eng}$) due to the high q_{eng} and hot ambient temperature. Therefore the air flow rate increases to a constant 0.378 m^3/s. The variation in the electrical components' transient responses (most noticeable in Figure 5(a) and (b)) is due to the different heat losses and masses.

4.2 PACE Validation

To verify the automatic parsing of the Simulink SLX file into C++ code, the conditions in [8] were replicated with the generated C++ code and the results compared with the original Simulink model. As an example, Figure 6 shows the result of the comparison for battery and power connector temperatures. On the left hand side, a scatter plot Matlab/Simulink vs C++ model results are shown, and on the right, the C++ results are superimposed on those generated by Simulink/Matlab. As seen in 6, the generated C++ code accurately replicates the results seen in the original Simulink model.

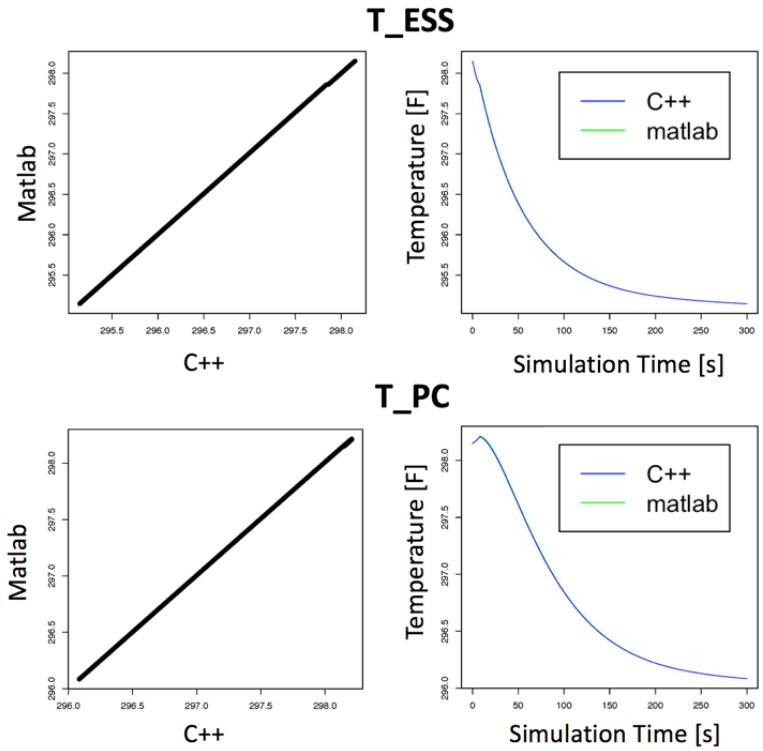

Figure 6. Comparison of battery (top row) and power converter (bottom row) in Simulink and C++.

5.0 SUMMARY

A Simulink thermal model of a hybrid-electric vehicle has been developed for integration into PACE by converting the Simulink SLX file into HPC-ready C++ code. The lumped-mass thermal model estimates the temperatures of components placed on two coolant loops and incorporates simple logic for the air flow rate through in-series radiators (one for each loop) based on the components temperature responses. Future versions of the thermal module will include more sophisticated control algorithms. The Simulink thermal model was developed outside of Autonomie before integration into PACE, which shows that independent Simulink models can be converted into HPC-ready C++ code, significantly increasing the generalization of the powertrain federate in MERCURY by supporting independent third-party developed Simulink modules.

6.0 ACKNOWLEDGMENTS

Permission to publish was granted by Director, Geotechnical & Structures Laboratory, U.S. Army Engineer Research and Development Center.

Material presented in this paper is a product of the CREATE-GV Element of the Computational Research and Engineering Acquisition Tools and Environments (CREATE) Program sponsored by the U.S. Department of Defense HPC Modernization Program Office. This effort was sponsored under contract number W912HZ-13-C-0037. Public Release, Distribution Unlimited.
Autonomie, is a product of the Argonne National Laboratory. Readers can go to http://www.autonomie.net/ for more information about Autonomie.

7.0 REFERENCES

[1] Haupt, T.A., Card, A.E., Doude, M., Mazzola, et al., "Powertrain Analysis and Computational Environment (PACE) for Multi-Physics Simulations Using High Performance Computing," SAE Tech. Pap. (2016-01–0308), 2016, doi:10.4271/2016-01-0308.

[2] Autonomie, www.autonomie.net.

[3] Kim, N., Rousseau, A., and Rask, E., "Autonomie Model Validation with Test Data for 2010 Toyota Prius," SAE Tech. Pap. (2012-01–1040), 2012, doi:10.4271/2012-01-1040.

[4] Kim, N., Rousseau, A., and Lohse-Busch, H., "Advanced Automatic Transmission Model Validation Using Dynamometer Test Data," SAE Tech. Pap. (2014-01–1778), 2014, doi:10.4271/2014-01-1778.

[5] Lee, D., Rousseau, A., and Rask, E., "Development and Validation of the Ford Focus Battery Electric Vehicle Model," SAE Tech. Pap. (2014-01–1809), 2014, doi:10.4271/2014-01-1809.

[6] Hodge, B.K. and Taylor, R.P., "Analysis and Design of Energy Systems," 3rd ed., Prentice Hall, Upper Saddle River, N.J, ISBN 978-0135-25973-3, 1999.

[7] Bergman, T.L., Lavine, A.S., Incropera, F., and Dewitt, D.P., "Fundamentals of Heat and Mass Transfer," 7th ed., John Wiley & Sons, Hoboken, NJ, ISBN 978-0470-50197-9, 2011.

[8] Haupt, T., Henley, G., Card, A., Mazzola, M., Doude, M., Shurin, S., and Goodin, C., "Near Automatic Translation of Autonomie-based Power Train Architectures for Multi-Physics Simulations using High Performance Computing," Accepted for publication 2017 SAE World Congress.