2017 NDIA GROUND VEHICLE SYSTEMS ENGINEERING AND TECHNOLOGY SYMPOSIUM
MODELING & SIMULATION, TESTING AND VALIDATION (MSTV) TECHNICAL SESSION
AUGUST 8-10, 2017 - NOVI, MICHIGAN

INTEGRATION OF AN EXTERNALLY-DEVELOPED THERMAL MODEL IN MERCURY'S POWERTRAIN ANALYSIS AND COMPUTATIONAL ENVIRONMENT (PACE)

Gabriel Monroe, PhD
Christopher Goodin, PhD
U.S. Army Engineer Research and Development Center (ERDC)
Vicksburg, MS

Angela Card
Matthew Doude
Tomasz Haupt, PhD
Gregory Henley
Michael Mazzola, PhD
Center for Advanced Vehicular Systems (CAVS)
Mississippi State University
Starkville, MS

Scott Shurin
U.S. Army Tank Automotive Research, Development and Engineering Center (TARDEC)
Warren, MI

ABSTRACT

Part of CREATE-GV's Mercury, the Powertrain Analysis and Computational Environment (PACE) is a simulation tool that provides advanced behavioral modeling of the powertrain subsystem of conventional or hybrid-electric vehicles. PACE performs its task by converting an existing powertrain architecture created in Autonomie or Matlab/Simulink into HPC-ready C++ code using an automated code generation capability, which parses the powertrain model's Simulink XML files. Utilizing PACE's modular powertrain model structure, a Simulink lumped-mass thermal model has been developed separately to augment the original functionality of the powertrain model. The augmented powertrain model was then subjected to a high-fidelity max speed test in Mercury's simulation environment to demonstrate the successful integration of a '3rd party' component via the PACE module. The Mercury Driver Client was also modified to accept calculated temperatures as an input. Including thermal analysis in powertrain modeling is crucial for vehicle development, especially hybrid-electrics. The lumped-mass thermal model provides a first-order estimation of the thermal behavior, but the modular structure of PACE and (ultimately) Mercury allows more sophisticated models to be easily integrated.

INTRODUCTION

High-fidelity simulation tools are critical for modern military ground vehicle design. Time and resource requirements can prohibit creating multiple prototypes of advanced military vehicles, whereas state-of-the-art simulation software can explore a complete design space for an optimal solution. Therefore, the U.S. Department of Defense High-Performance Computing (HPC) Modernization Program Office initialized the Computational Research and Engineering Acquisition Tools and Environments (CREATE) Program. The military ground vehicle element of the program is the CREATE-GV project, which focuses on end-to-end mobility analysis and providing data for interfacing with trade space tools. Addressing the end-to-end mobility analysis focus area, CREATE-GV's Mercury [1] is a software application written in C++ that provides capabilities for concept design and analysis of wheeled and tracked vehicles. As shown in Figure 1, Mercury consists of independent physics-based models, or 'clients', that run in parallel to model the interactions of vehicle dynamics, powertrain performance, vehicle-terrain interaction (VTI), and the overarching driver controls; these are the main clients currently used by Mercury, but due to its modular framework, other modules are under development, *e.g.*, a ford/swim client for river crossing simulations.

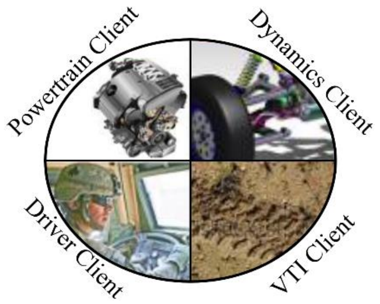

Figure 1: The Mercury Co-simulation Framework.

Certain Mercury simulations do not require all four clients, *e.g.*, a 'tilt-table' test determines the ground angle that will cause a vehicle to roll and only needs the Vehicle Dynamics and VTI clients. Therefore, certain simulations with narrow parameter spaces can be run on an individual computer with no speed loss compared to an HPC environment, but trade space studies involving many parameters or simulations using multiple clients require HPC for timely solutions (*e.g.*, hours compared to days). Mercury's modular framework allows the addition or modification of physics modules within a client without impacting other clients or Mercury's core code. Intra-client modification is possible so long as the base-class of the client being modified both inherits from the corresponding Mercury base-class and has an update method that receives and returns certain data structures specified by Mercury's core code.

The main powertrain client in Mercury is the Powertrain Analysis and Computing Environment (PACE) [2]. PACE allows for the modelling or modification of all components typical in a conventional or hybrid vehicle (engine, energy storage, transmission, electric motor, etc.) as assembled in the U.S. Department of Energy's Autonomie software. Autonomie is an experimentally validated [3–5] modeling environment developed by the Argonne National Laboratory that accurately simulates transient powertrain hardware and control features. Dependent on Mathworks's Simulink, Autonomie was originally created for the design and simulation of civilian vehicle powertrains but is also applicable to military vehicles. However, PACE does not replicate the Autonomie software. Instead, PACE is intended to bridge the gap between powertrain engineers familiar with Autonomie or Simulink and HPC specialists. To this end, PACE will convert a specific powertrain architecture developed with Autonomie into C++ code, which then operates independently from Autonomie (a Microsoft Windows application) as a Mercury client in an HPC environment. Prior to their conversion into HPC-ready C++ code, PACE powertrain models exist as Matlab/Simulink blocks, either created by Autonomie or directly in

Simulink such as the thermal model presented here. Thus, the base powertrain architecture can be modified and new sub-models can be added independent of Autonomie. A graphical overview of the Autonomie, PACE, and Mercury interactions is provided in Figure 2.

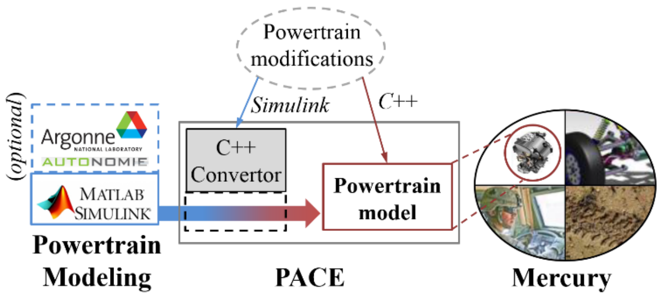

Figure 2: Overview of Autonomie, PACE, and Mercury interactions.

In general, the modular nature of PACE and Mercury would allow any number or type of modifications or additions to PACE, so long as the resulting inputs and outputs matched the powertrain client requirements in Mercury. The first demonstration of this capability was the implementation of a lumped-mass thermal sub-model into a hybrid powertrain model at the PACE level [6]. C++ code was automatically generated from an augmented Simulink model [7], and the resulting simulated thermal behavior was shown to match the unconverted model when tested within the PACE client. This work outlines the integration of a thermal-augmented PACE powertrain with Mercury and demonstrates the impact of thermal considerations in high-fidelity vehicle simulations by comparing a vehicle's top speeds with and without the powertrain thermal model.

THERMAL MODEL INTEGRATION

The lump-mass thermal model and the conversion of the PACE Simulink modules into HPC-ready C++ code has been discussed in detail elsewhere [6,7], but for the sake of completeness a brief summary is given here. The current PACE thermal model is designed to match the cooling needs of a notional light tactical vehicle with a parallel micro-hybrid architecture. As such, the thermal model contains two coolant (water) loops, one for electrical devices and one for the engine, each with its own pump. The disparity between the engine and electrical operating temperatures, *e.g.*, 100 °C vs. 50 °C, necessitates two coolant loops. However, the loops' radiators are placed in series with a single fan, and the cooling performances of the two radiators vary as a function of air flow rate. Although the fan is variable speed, the liquid coolant pumps are not; instead, in the present simulations the pumps provide constant flow rates of 0.378 kg/s and 4.54 kg/s for the electrical and engine loops, respectively. A general schematic of the Simulink thermal model is given in Figure 3.

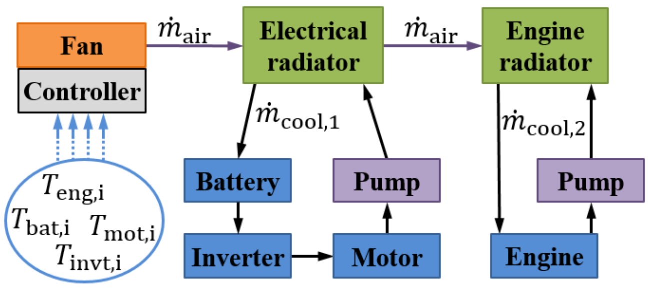

Figure 3: Schematic of thermal sub-model for LTV.

At each simulation time step, the temperatures of the powertrain components are checked against specified operating and maximum temperatures, *i.e.*, T_{op} and T_{max}, respectively. If any component temperature (*i.e.*, T_{com}), is above its T_{op}, the fan speed increases until $T_{com} < T_{op}$ or the maximum fan speed is reached. If $T_{com} < T_{op}$, the fan speed will decrease toward a minimum fan speed until $T_{com} > T_{op}$ again.

The hybrid architecture in PACE currently includes ~20 modules such as engine, gearbox, battery, etc. The parameters of each component may be adjusted as needed as part of the typical PACE workflow. Furthermore, any module within the PACE architecture can be replaced with externally developed models if the new module has identical inputs and outputs as the replaced module. Because the thermal model is a new addition, not replacement, to the existing PACE architecture, integrating the thermal calculations into other PACE modules requires a few additional steps be taken prior to the conversion into HPC-ready C++ code. The additional preprocessing mainly consisted of updating the initialization files to include the information required by the thermal model, but some restructuring of the original thermal Simulink code was necessary to conform with the PACE conversion process.

After the thermal model was added to the powertrain architecture developed in Autonomie, the entire augmented PACE model was automatically converted from Simulink into HPC-ready C++ code as is described in [7]. The capability of developing new modules, such as the thermal model, in Simulink before automatic translation to C++ is crucial as it makes PACE (and, by extension, Mercury) more accessible to automotive engineers. The lumped-mass thermal model provides a first-order estimation of the thermal behavior. However, as with higher level Mercury components, the PACE structure only requires that thermal model inputs/outputs be kept consistent, which allows the model to be easily replaced or modified, *e.g.*, switching from a hybrid to a conventional powertrain architecture.

RESULTS AND DISCUSSION

Two types of powertrain tests were simulated in Mercury to demonstrate the impact of the thermal model addition. All simulations were performed using the high-performance, hybrid powertrain architecture shown schematically in Figure 3. First, a simple max speed test was simulated in an unmodified version of Mercury with the augmented PACE Client at ambient temperatures of either 20 ^{o}C or 40 ^{o}C. Secondly, the Mercury Driver Client was modified by adding a simple controller with the goal of preventing the engine from overheating by limiting vehicle speed. With the modified Driver Client in place, the 40 ^{o}C ambient speed test was again simulated. Because a max speed test with only three variations was being simulated, tests were run on a Mac Pro desktop with a 1/2048 s time step. Simulation constants for the powertrain components from Figure 3 are given in Table 1.

Table 1. Components' simulation constants

	T_{init} (^{o}C)	$T_{\text{op}}/T_{\text{max}}$ (^{o}C)	m (kg)	c_{p} (J/kg·K)
Battery	25	45/50	130	910
Inverter	25	45/50	15	910
Motor	25	45/50	50	910
Engine	25	90/100	200	910

Max Speed Test

To approach a quasi-thermal-equilibrium process in the powertrain components, the Mercury driver module maintained a relatively low acceleration (5.0 mph/min), until the vehicle speed stopped increasing. Figure 4 shows all component temperatures and the fan air flow rate for both ambient temperatures. During both tests, only the engine temperature surpassed its preset maximum temperature as the ratio of heat loss to cooling capacity was small for the electrical components, which results in their temperatures being within a 1 ^{o}C or 3 ^{o}C band for the 20 ^{o}C or 40 ^{o}C tests, respectively. Thus, Figure 5 only shows the engine temperature with the vehicle speed.

In Figure 4, T_{eng} surpasses $T_{\text{eng,op}}$ sooner for T_{amb} = 40 ^{o}C, as would be expected. This behavior is clearly seen in the response of the cooling fan, which begins ramping up 4 s earlier at 40 ^{o}C ambient than at 20 ^{o}C. Although the faster cooling flow slows the rate at which temperatures increase (as evident by the difference in the slope of T_{eng} before and after the fan speed increases), the engine temperature does not drop below 90 ^{o}C again. The electrical component temperatures do not reach steady state before the end of the simulations, but were tracking to slightly above T_{amb} in both cases.

Including $T_{\text{eng,max}}$ in Figure 5 shows that the engine overheats 14.2 seconds sooner in 40^{o}C ambient than 20 ^{o}C (251.1 s and 265.3 s, respectively), which corresponds to 24.5 mph at 40 ^{o}C and 25.6 mph at 20 ^{o}C ambient. These speeds are in comparison with the vehicle's maximum speed of 44.4 mph with no thermal considerations.

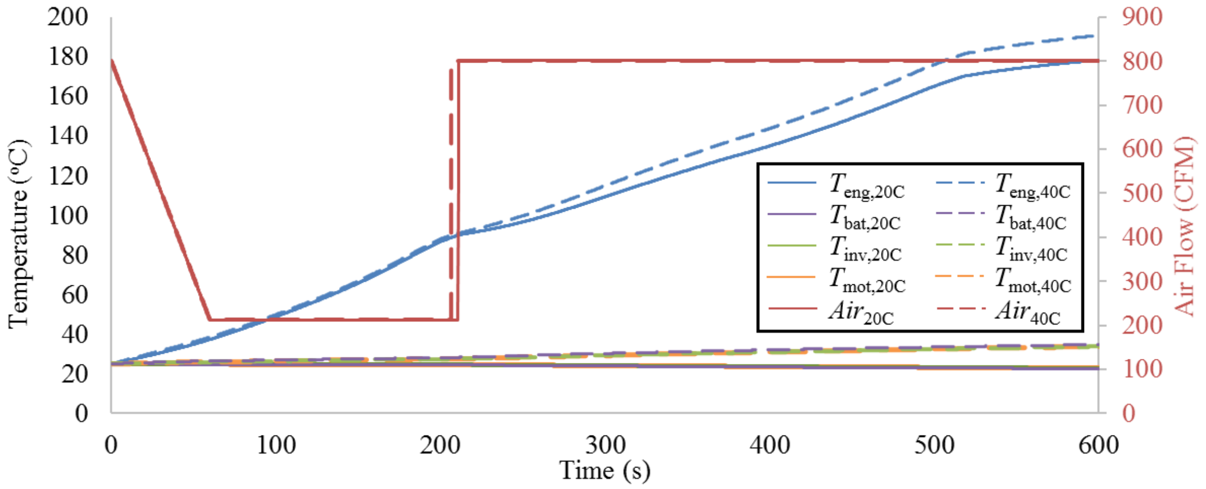

Figure 5: Powertrain component temperatures and air flow vs. time

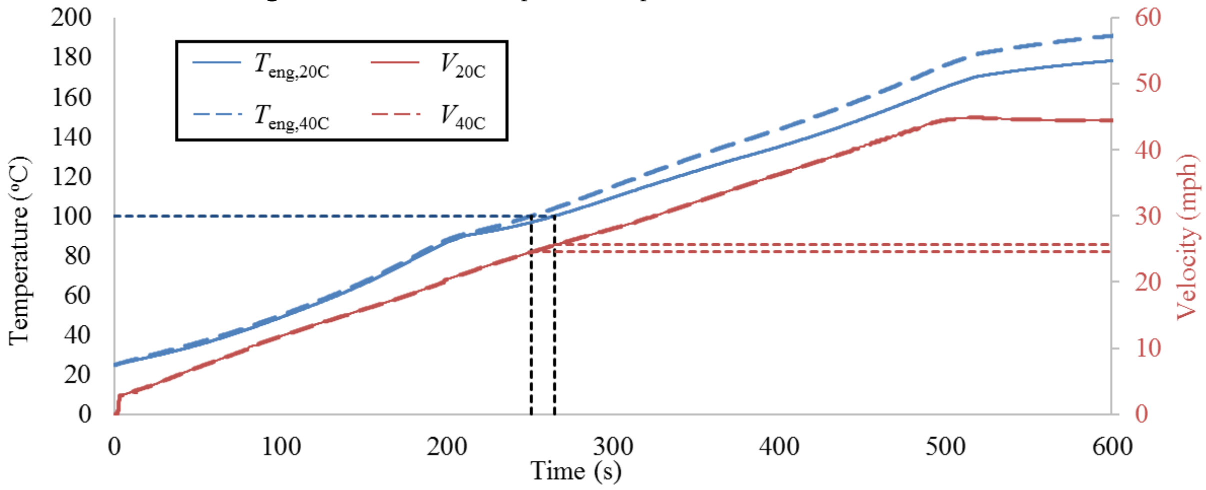

Figure 4: Engine temperature and vehicle speed vs. time.

However, the actual top speed as regulated by temperature is actually lower than 25 mph, since the powertrain's thermal state lags the vehicle speed.

Modified Driver Client

For the modified Driver Client speed test, a controller logic similar to that of the cooling fan was added to the throttle. Once $T_{eng} > 95^{c}$C, the throttle reduced until $T_{eng} < 95$ ^{c}C, and it begins to increase again at 5 mph/min. Besides the integration of engine temperature into the Driver Client, the simulation was identical to the max speed tests. Figure 6 gives T_{eng} and vehicle speed vs time for the modified Driver Client simulation for $T_{amb} = 40^{o}$C.

Although not shown explicitly in Figure 6, the fan speed increased when $T_{eng} = 90$ ^{c}C at the exact same time (206.6 s) as in Figure 4, as shown by the slope change of T_{eng}. However, the vehicle speed reduces at 232.2 s when $T_{eng} > 95$ ^{c}C, which also corresponds to the maximum velocity of 22.3 mph. During the last 10 s of the simulation, the average velocity and engine temperature are 17.0 mph and 94.9 ^{o}C, respectively. For the 20 ^{o}C ambient test, the average velocity was 21.1 mph. These speeds are starkly contrasted by the powertrain mechanical limit of 44.4 mph shown in Figure 5. In a design scenario, these simulations would demonstrate the need for a more robust engine cooling solution.

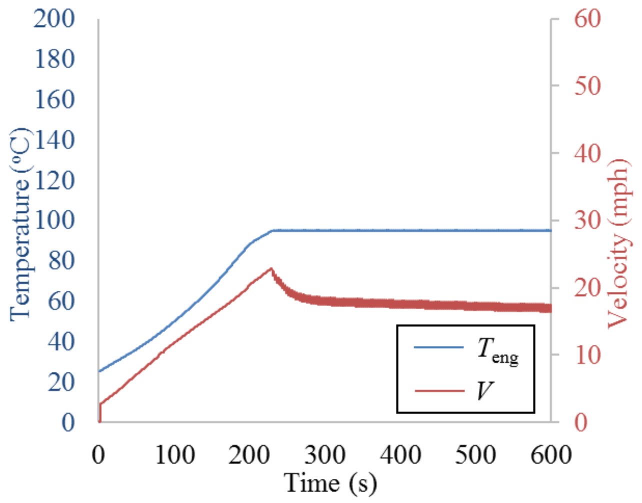

Figure 6: Engine temperature and vehicle speed vs. time at 40°C

SUMMARY

The integration of a standalone lumped-mass thermal model with an existing PACE powertrain model has been described, and the impact of thermal considerations in military vehicle design and simulations was shown by performing high-fidelity simulations in Mercury. Furthermore, the output of the thermal model was integrated into the Mercury Driver Client, which reduced vehicle speed when the engine started overheating. Future efforts will add fan and pump electrical loads to the vehicle power systems to impact fuel economy and available engine torque. Also, smoother controllers, *e.g.*, PID, will be added to the cooling fan and Driver Client. Although the thermal sub-model was the focus of this work, the method of sub-model integration can be generalized to other topics. PACE and Mercury are designed to be easily extended with new or improved physics modules to adequately quantify the mobility performance of military ground vehicles.

ACKNOWLEDGMENTS

Permission to publish was granted by the Director of the Geotechnical and Structures Laboratory, U.S. Army Engineer Research and Development Center (ERDC).

Mercury is a collaborative work between the ERDC and the U.S. Army Tank Automotive Research, Development and Engineering Center (TARDEC).

Autonomie is a product of the Argonne National Laboratory and was created by Aymeric Rousseau, Philip Sharer, Sylvain Pagerit, Ram Vijayagopal, Shane Halbach, Neeraj Shidore, and Dominik Karbowski. For more information, visit www.autonomie.net

REFERENCES

[1] Goodin, C., Mange, J., Pace, S., Skorupa, T., Kedziorek, D., Priddy, J., and Lynch, L., 2017, “Simulating the Mobility of Wheeled Ground Vehicles with Mercury,” SAE Int. J. Commer. Veh., **10**(2), pp. 2017-01–0273.

[2] Haupt, T. A., Card, A. E., Doude, M., Mazzola, M. S., Shurin, S., and Hufnagel, A., 2016, “Powertrain Analysis and Computational Environment (PACE) for Multi-Physics Simulations Using High Performance Computing,” SAE Tech. Pap., (2016-01–0308).

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

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

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

[6] Monroe, J., Doude, M., Haupt, T., Henley, G., Card, A., Mazzola, M., Shurin, S., and Goodin, C., 2017, “Thermal Modeling in the Powertrain Analysis and Computational Environment (PACE),” NATO AVT-265.

[7] Haupt, T., Henley, G., Card, A., Mazzola, M. S., Doude, M., Shurin, S., and Goodin, C., 2017, "Near Automatic Translation of Autonomie-Based Power Train Architectures for Multi-Physics Simulations Using High Performance Computing," SAE Int. J. Commer. Veh., **10**(2), pp. 2017-01–0267.