LOGIC-BASED, PERFORMANCE-DRIVEN ELECTRIC VEHICLE SOFTWARE DESIGN TOOL

FINAL REPORT

FEBRUARY 2001

Report Budget Number KLK305
Report Number N01-10

Prepared for

OFFICE OF UNIVERSITY RESEARCH AND EDUCATION

U.S. DEPARTMENT OF TRANSPORTATION

Prepared by

NATIONAL INSTITUTE FOR ADVANCED TRANSPORTATION TECHNOLOGY

UNIVERSITY OF IDAHO

Donald M. Blackketter, PhD, PE

David G. Alexander, Graduate Student

TABLE OF CONTENTS

EXECUTIVE SUMMARY

Two goals were pursued concurrently during this research. The first goal was to develop a performance-driven, steady state, hybrid electric vehicle (HEV) software design tool that would provide design information to the University of Idaho FutureTruck 2000 Suburban. The second goal of this research effort was to develop logic-based, computer algorithms that could be used to outperform numerical solvers currently available.

The HEV design software, SmartHEV, is a flexible and robust model of steady state HEV operation. The power flow through the vehicle components is modeled using the road load power equation. The components that are modeled include the battery pack, alternator, alternative power unit (APU), electric motor, transmission, differential, and wheels. SmartHEV integrates a user-friendly interface with logic-based algorithms that allow the selection of combinations of known and unknown variables. This flexibility in selecting known and unknown variables is a unique feature of SmartHEV. Known variables can also be used to step through a range of vehicle operation in order to calculate optimum performance levels. Parametric results can be plotted and compared against other design configurations. Through the process of selecting known and unknown variables, a user can gain tremendous insight into the relationships of the vehicle components and the variables that describe them.

Several logic-based algorithms were developed and implemented in SmartHEV to improve convergence success and rate of convergence for general systems of equations. The algorithms developed include

1 Rewriter
2 VarSelect
3 Solution Path
4 Best Solution Path
5 Swap-and-Solve

The equation Rewriter algebraically manipulates variables within equations to eliminate potential solving problems. The VarSelect routine guarantees that the user specify a non-singular set of unknown variables; the routine also provides tremendous flexibility and insight during the selection of known and unknown variables. The Solution Path and Best Solution Path algorithms determine the best strategy for solving a system of equations; this often results in faster computational speeds when compared to traditional solving strategies. The Swap-and-Solve routine determines alternative solution strategies that can be used to find a valid solution when convergence problems exist. These algorithms are integrated in a Windows environment that manages known and unknown variables and their solutions, making variable identification and selection easier for the user. The engineering design analyst will find that using these algorithms will reduce computational time and improve convergence success when solving linear and nonlinear systems of equations.

DESCRIPTION OF PROBLEM

Electric Vehicle Simulation and Modeling

Vehicle simulation software is essential to vehicle design and development. All current vehicle software simulations, however, require that components be specified prior to running the simulations. This works well if the user's goal is to gain insight into a particular design. Unfortunately, when the design has not yet been established, running multiple simulations on proposed vehicles can be time-consuming. SmartHEV is a vehicle software design tool that does not require a vehicle to be completely specified before performing a steady state simulation.

SmartHEV is a vehicle component design tool written in VisualBasic 6.0 that uses algorithms currently under development at the University of Idaho [1, 2]. SmartHEV can accept performance goals as well as component specifications as input based on steady state vehicle operation. The user-friendly graphical interface provides a platform allowing engineers to mix-and-match vehicle components and performance goals. The components that are modeled include the wheels, driveshaft/differential, transmission, electric motor, battery pack, alternator, and APU. The road load power includes effects due to aerodynamic drag, rolling resistance, uphill climbing and component efficiencies.

The algorithms provide valuable information to the user. Because component specifications and performance parameters are linked to the variables, the user learns how a change in one variable affects other variables. This way the user can focus on the correct set of variables in order to modify a component design to get a desired performance.

Need for Advanced Numerical Techniques and Algorithms

Because of the large number and complexity of components in an electric vehicle, modeling and simulation results in a large system of equations that can be difficult to

solve. Difficulties arise when solving large sets of equations for a variety of reasons. Depending on the known and unknown variables, some systems require initial guesses to be close to their actual value. Other systems produce multiple answers, some of which would be physically impossible to use in a design. Still other systems produce answers that are near singular or were improperly derived resulting in numerical instability. These problems can be daunting and at times, even insurmountable.

Not only are the systems of equations difficult to solve, but they are also difficult to manage, simply because of the limits of human cognitive ability. The human mind can successfully manipulate approximately only eight chunks of information at one time [3]. Given that each variable and each operator in an equation are chunks, two simple equations can challenge the limits of human cognitive ability. Algorithms that help manage variables and their values can be of tremendous help when solving large systems of equations. Several of the design algorithms that were developed during this project provide structure to a user for selecting known and unknown variables and assigning values, often without the user needing to understand how this is accomplished.

APPROACH AND METHODOLOGY

SmartHEV Program Development

SmartHEV simulates the steady state operation of an HEV based on the relationships derived from the road load power equation. The road load power accounts for the rate of energy required to power a vehicle and is based on the first law of thermodynamics. The power required at the wheels to maintain a vehicle at a prescribed velocity under various driving conditions is calculated using the following equation, adapted from *SIMPLEV*, an electric vehicle simulation program [4]:

$$P(t) = P_{aero} + P_{rolling} + P_{grade} + P_{acc} + P_{bearing} . \quad (1)$$

P_{aero} is the power demand as a result of aerodynamic drag; $P_{rolling}$ is the load due to the resistance of the road on the wheels; P_{grade} is the power required to climb a hill; P_{acc} is the power necessary for acceleration; and $P_{bearing}$ is the power required in overcoming the resistance of the bearings and the final drive shaft.

The total power *P(t)* necessary to meet the velocity is transmitted to the drive shaft. This power request is then translated into a specified torque and speed at the drive shaft. Using the torque and speed at the drive shaft, the demand on the transmission is determined. With the transmission gear ratio, the requested torque and speed from the electric motor are determined. The losses through the inverter are calculated using a constant inverter efficiency coefficient.

The power requested at the high voltage bus is used to determine the total power available for discharging and charging the battery pack. The total battery power is defined as

$$P_{bat} = P_{alt} + P_{aux} + P_{bus} , \quad (2)$$

where, P_{alt} is the power from the alternator used to charge the pack, P_{bus} is the BUS power demand, and P_{aux} is the power required for auxiliary loads. Power discharging from the batteries is positive; power is negative when recharging. Equation (2) balances the power from each vehicle component. This configuration enables the user to determine the power demand from each component while holding the power from the remaining components at zero and thus determine the steady state power demand from each component necessary for continuous operation.

The power demand necessary to maintain a vehicle at a prescribed velocity is split between the battery pack and the alternator. The power through the alternator is adjusted using a constant alternator efficiency, and the alternator then requests power from the APU. The battery pack discharge efficiency is considered in relation to the heat losses due to ohmic heating and an approximate constant efficiency.

The battery pack component was modeled with an open circuit battery voltage, internal resistance and load voltage. A voltage loop equation was derived in order to calculate the current at the battery pack. The current was found using the positive root of the quadratic equation which resulted from the voltage loop, with internal resistance and open circuit voltage held constant.

The equations derived from modeling the power flow through each component between the road and the battery pack were compiled and were used as the system equations in SmartHEV. These equations accurately model a series HEV. The system of equations can be altered by the user, but proper care must be taken to maintain numerical stability. The user is not presented with the equations upon starting SmartHEV. Vehicle configurations are explored by the user in the graphical interface where only the components and their variables are presented.

The interface for SmartHEV was designed to be flexible, easy to use, and dynamic. A user can indicate any variable as known by selecting its name or using the up/down arrow next to the variable value. All variables appear either on the main screen or on the

pulldown menu. The variable value box turns yellow when it is selected as known or orange if selected as unknown. When the user clicks a variable label that is unknown, a dialog box appears. The dialog box lists all possible variables that can be exchanged with the selected unknown variable. If a variable is selected from the list, it becomes unknown and the variable that was originally selected becomes known. This interchange maintains a non-singular system of equations.

Units for all variables are displayed when the pointer (mouse arrow) hovers over a variable label. The user can change the variable units by right-clicking on the variable label. A list of possible common units in both the International System (SI) and the English System is provided for each variable.

SmartHEV continuously verifies whether or not sufficient variables have been selected as known and/or unknown to make a valid set of solvable equations. When a valid set exists, SmartHEV will solve the set and display the results. A change in a known value will cause changes in unknown variable values, but only those unknown variable values that change will appear in yellow text with an orange background. Unknown variable values that do not change remain in black text on an orange background. The changes in text and background colors make it easy to determine which unknown variable values are dependent on known values. Whenever a known value is selected or changed, the results are automatically updated on the screen. Tremendous insight can be gained by observing the different variable interactions caused by the various component relationships.

Advanced Numerical Techniques and Algorithms

The design algorithms presented in this research were combined into a general, linear and nonlinear equation solver. The solver is general in the sense that the user can input any equation or system of equations. The unique design algorithms pre-process the variables and equations and then send the optimized set of equations to the solving engine, which utilizes a standard Newton-Raphson method for systems of equations. The design

algorithms—the Rewriter, VarSelect, Solution Path, Best Solution Path, and Swap-and-Solve—are invisible to the user and, except for a few initialization parameters, require little user input.

The five design algorithms that were developed perform one of two distinct functions. First, the algorithms manipulate system equations and variables in order to improve convergence success and speed. Second, the algorithms prevent the modeling analyst from making mistakes during variable selection and definition.

Rewriter Algorithm

The Rewriter algorithm analyzes and rewrites equations that are input by the user in an equation editor screen. If possible, it rewrites equations algebraically to remove unknown variables in the denominator of fractions. While simple, this approach eliminates a potential a “divide-by-zero” error. Because the Newton-Raphson method is an open method, it is possible that, if the unknown variable were in the denominator, the denominator could go to zero. The Rewriter eliminates this potential error whenever possible (see Fig. [1]). A flow chart for the Rewriter is illustrated in Fig. (2).

Figure 1. Two equations as rewritten by the Rewriter algorithm

VarSelect Algorithm

The VarSelect algorithm contains two logic routines called the U_Select and the K_Select. Once equations are input in the equation editor and rewritten by the ReWriter routine, all variables are parsed from the equations and displayed as labels and variable value boxes on the screen. Initially, no variable is known or unknown. The user selects the known variables and those that are unknown and therefore to be solved. Once a variable is selected as known, the K_Select routine identifies any other variable(s) that must be unknown based on the first variable as known. If K_Select determines that a variable must be unknown, its background

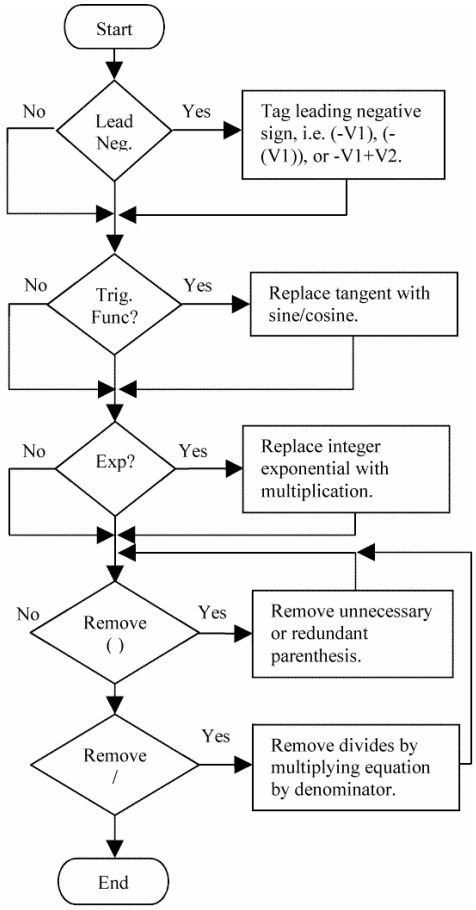

Figure 2. Rewriter flow chart

color is changed to orange, in which case the variable can no longer be selected as known. However, if at a later time the user decides that it would be advantageous to have the unknown variable known, the VarSelect routine will identify all known variables that could be swapped or exchanged with that unknown variable. These two operations give tremendous insight into the interactions between variables and equations and provide the user with valuable information for making informed design decisions. A non-singular set of variables and equations is maintained throughout the variable selection process, thereby increasing the chances of convergence.

The logic used in the VarSelect routine has evolved over the past several years. The VarSelect routine determines a valid set of known and unknown variables based on the number of variables present in each equation and the frequency of variables in all

equations. The selection of known and unknown variables is independent of any mathematical operator. Earlier applications developed at the University of Idaho, such as SmartSolve 1.2 [1], SmartSolve 2.2 [2] and Kinematics and Dynamics [5] have used similar routines with great success.

Solution Path/Best Solution Path Algorithms

A significant amount of work to determine the most efficient strategy to solve systems of equations was conducted in the 1960s and 1970s. Ramirez and Vestal outline two algorithms that help select a set of unknown variables in order to decouple a system of equations [6]. A system of equations that is decoupled has the fewest number of equations that must be solved simultaneously. In a decoupled system, each unknown variable is solved using one equation. This is known as a sequential solution path. The algorithms outlined by Ramirez and Vestal are designed to help structure and organize a large set of equations to facilitate calculating by hand. Christensen, Lee, and Rudd have published similar work [7, 8]. Much of this work has never been implemented into a numerical solver or coded into a computer program.

The main function of the Solution Path is to determine a solution path that minimizes the number of simultaneous equations that must be sent to the numerical solver. Once a variable is determined as unknown, either by the user or the VarSelect routine, the Solution Path checks to see if enough variables have been specified as known in order to solve for a specific unknown. The Solution Path routine also determines the order of solving each unknown variable.

The Solution Path uses information from the structure of the equations and variables to make decisions, much like the VarSelect routine. Ramirez and Vestal outlined two algorithms used for determining the best-known variables and the best solution path [6]. The Solution Path routine diverges from their work by allowing the user to select any set of known and unknown variables, on which the solution path is then based. SmartHEV uses the Solution Path routine to determine whether or not the set of equations to be

solved contains coupled equations (multiple equations used to solve multiple unknown variables at the same time) or sequential equations (one unknown variable used to solve one equation at a time).

The Solution Path algorithm is a powerful tool for solving systems of equations. Simultaneous solution paths require guess values to be close to their solved value. However, a sequential solution path does not necessarily require careful guesses. This is particularly true with systems of equations where linear equations can be solved early in the sequential solution path and nonlinear equations can be solved later. Additionally, sequential solution paths tend to iterate to a solution faster than simultaneous solution paths.

The Best Solution Path routine, also based on the work of Ramirez and Vestal [6], is similar to the Solution Path routine, except that it determines the best solution path without using the known and unknown variables selected by the user. However, the Best Solution Path also determines the degree to which each variable is nonlinear. The variables with higher degrees of nonlinearity are given more weight for being known variables. This information is then used to determine the best-unknown variables. Best-unknown variables are typically linear and result in the least number of coupled equations. A list of the best-unknown variables is generated and used in the Swap-and-Solve routine, which is outlined below.

The Solution Path and Best Solution Path algorithms use a similar sequence of steps to determine a solution strategy based on the number of variables and the number of times a variable is present. The algorithms generate a matrix of zeros and ones (the occurrence matrix) that represents the existence of a variable in an equation. For example, Table 1 shows the occurrence matrix for the following equations:

$$a^2 + b = c \qquad (3\text{ a})$$

$$a + d = \sin(c) \qquad (3\text{ b})$$

$$a * b = c / e \qquad (3\text{ c})$$

Table 1 Occurrence Matrix

Variables	***a***	***b***	***c***	***d***	***e***
Eq. (3 a)	1	1	1	0	0
Eq. (3 b)	1	0	1	1	0
Eq. (3 c)	1	1	1	0	1

The solution path can be determined by reviewing the occurrence matrix. Our example has three equations; therefore, there can only be three unknown variables. Careful selection of the unknown variables will result in a sequential solution path; in other words, all equations can be decoupled. If the sum of the rows for any variable is equal to one, that variable is present in only one equation. The equation that contains the one occurrence of that variable must be used to solve for that variable. In Table 1, variables *d* and *e* have sums equal to one. Therefore, *d* and *e* would be best identified as unknown variables. These variables are not coupled with any other equation. In fact, any other variable can be selected as unknown and the system would be decoupled.

This would not be the case if *d* or *e* were known. If both *d* and *e* were known, no variable would have a row that summed to one. This would indicate that a simultaneous set of equations existed. In that case, all three equations would have to be solved simultaneously to solve this system.

If *b* and *e* were known and *a*, *c*, and *d* unknown, the system would be partially coupled. Equations (3 a) (3 c) could be solved simultaneously for variables *a* and *c*. With these results, *d* could be solved sequentially.

The Solution Path and Best Solution Path evaluate the occurrence matrix in a similar way as described above. The rows containing ones are summed for each variable. Variables with a sum equal to one are solved first. For some systems of equations, this will be how all the variables are solved. When there is no variable with a sum equal to one, the Solution Path and Best Solution Path determine the smallest set of coupled equations that can be solved. Once an equation is identified as solvable, it is removed from the occurrence matrix. The columns for each variable in the occurrence matrix are added

again and searched for a value of one. The algorithm proceeds until all equations have been removed from the occurrence matrix.

Swap-and-Solve Algorithm

The Swap-and-Solve routine provides an additional solving strategy when the system of equations does not converge to a solution. The Swap-and-Solve routine uses the VarSelect and the Best Solution Path routines to iterate on a solution in the event that the Newton-Raphson method fails to converge based on the known and unknown variables selected by the user. When the Swap-and-Solve routine runs, it compares the unknown variables selected by the user with the best-unknown variables that were determined by the Best Solution Path. If a best-unknown variable is missing from the list of unknowns specified by the user, that variable is added to a list of potential swap variables. Variables that the user selects as known but that are best-unknown variables are called swap-unknown variables. Variables that are unknown but are better as known variables are called swap-known variables.

All of the swap-unknown variables are exchanged with originally unknown variables and a solution path is generated. The solution path with the most decoupled equations is selected as the Best Solution Path. The swap-unknown variable is then seeded with a guess that is its actual known value, and the swap-known variable is seeded with a guess of unity. The Best Solution Path is then sent to the solver. If the system converges to a solution, the swap-unknown value is compared with the variables known value. If the values are nearly the same, within a specified tolerance, the system has solved correctly; otherwise the Best Solution Path is sent to the false-position routine
The false-position routine uses the false-position bracketing method to iterate on the swap-known value in order to bring the swap-unknown variable within a specified tolerance of its known value. The swap-known variable was set as a known variable in the set of equations, but it was originally an unknown variable. Its swap-known value is only a guess. The upper and lower limits for the swap-known variable are first determined. This guarantees that the root exists within the bounds. Once the swap-known

variable is successfully bracketed, the false-position method iterates on the swap-known variable until the difference in the swap-unknown value and its known value are with the tolerance limit.

The Swap-and-Solve routine can find a solution for problems that other solvers cannot because it eliminates the need for more accurate guesses. If the user-specified known and unknown variables result in a coupled solution path, good guesses—sometimes even within tenths of a percent of the solved value—are often necessary in order to converge to a solution. Guesses this accurate are often difficult to determine.

The Swap-and-Solve routine decouples the solution path by swapping known and unknown variables. The resulting solution path may be sequential, and therefore may not require accurate guesses in order to converge. If the first swap does not result in a convergent solution, the Swap-and-Solve routine will find the next variable to swap that minimizes the number of coupled equations in the solution path. Once a solution is found, even though it is based on a guessed known variable, the method of false-position will bring the swap-known value close to its true value on successive iterations. When the Swap-and-Solve routine is called it is typically not the set of equations that is faulty, but rather the solution path. Using the Swap-and-Solve technique to find a better solution path can make the difference between success or failure when solving systems of equations.

FINDINGS; CONCLUSIONS; RECOMMENDATIONS

SmartHEV

Figure 3 shows the graphical user interface (GUI) of SmartHEV. The variable value boxes have adjustments that change the value by 10 percent for every click of the up or down arrow. The variable value with the dark background and light numbers in Fig. 3 indicate that the variable has changed value because of a change that was made to a known variable value. All major components are easily visible, each containing a set of variables that govern the performance of the HEV.

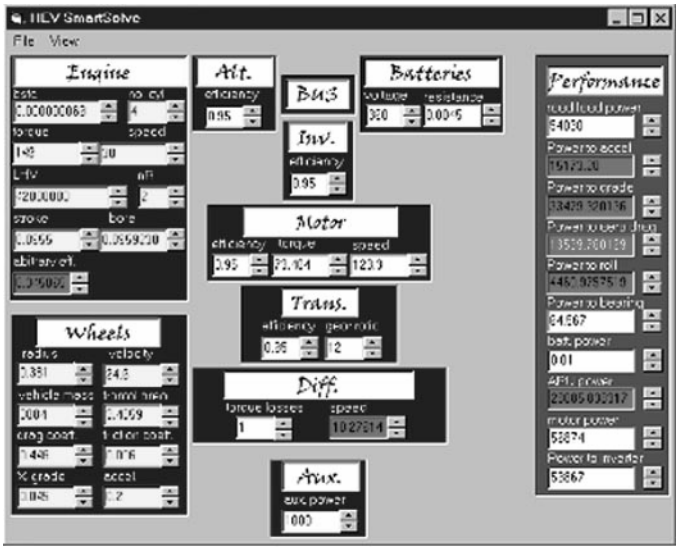

Figure 3. SmartHEV GUI

Table 2 lists the results from four different design iterations of SmartHEV based on the input parameters and performance goals listed in Table 3. In Table 2, the first column indicates the results with no power provided by the battery pack. Column two lists the results with no power provided by the APU. The third column, Hybrid I, lists the model results based on a hybrid operation. Neither the APU nor the battery pack output power was specified. Hybrid II, in the fourth column, was designed with a known APU power output of 30 kW. As a result, the APU speed and torque were adjusted to accommodate the constant APU power output.

Table 2 Design Results from Four Different Scenarios

	No Battery Pack	No APU	Hybrid I	Hybrid II	Units
Road Load	33,835	33,835	33,835	33,835	watts
Accel. Power	3,478	3,478	3,478	3,478	watts
Grade	20,463	20,463	20,463	20,463	watts
Aero.	5,989	5,989	5,989	5,989	watts
Rolling	3,065	3,065	3,065	3,065	watts
Bearing	838	838	838	838	watts
Motor Power	35,616	35,616	35,616	35,616	watts
Motor Speed	92.7	92.7	92.7	92.7	rev/s
Motor Torque	61.2	61.2	61.2	61.2	Nm
APU Power	51,400	0.0	34,557	30,000	watts
APU Speed	110	0.0	100	80	rev/s
APU Torque	73.9	0.0	55	60	Nm
APU Eff.	0.25	0.0	0.19	0.20	
Batt. Power	0.00	48,830	16,000	20,330	watts
Batt. Current	0.0	245	80	102	amps
Alt. Power	48,830	0.0	32,829	28,500	watts
Alt. Current	244	0.0	164	143	amps
Gear ratio	6.67	6.67	6.67	6.67	

SmartHEV was bench-tested against the Advanced Vehicle Simulator (ADVISOR) developed at the National Renewable Energy Laboratory (NREL). ADVISOR is a recognized industry standard for simulating alternative fuel vehicles. The sizes of the vehicle components were based upon the following performance goals: constant grade of 6 percent, maximum speed of 40.2 m/s, and acceleration of 0-26.9 m/s in 12 sec. Because SmartHEV is a vehicle design tool rather than a vehicle simulation tool, the performance criteria had to be modified to make comparisons with ADVISOR valid. SmartHEV determined the vehicle component sizes for each performance criteria separately. The grade test was performed under constant speed at 24.6 m/s (55 mph). SmartHEV then calculated the component sizes based on the maximum speed at zero grade. The best effort acceleration was determined at zero grade as well. The results were compared and the maximum component size was determined. The design parameters shown in Table 3

were input to both SmartHEV and ADVISOR. Table 4 shows the comparison between the SmartHEV simulation and the one run in ADVISOR.

Table 3 Design Parameters

Mass	1413	kg
Acceleration	0.1	m/s^2
Velocity	24.6	m/s
Grade	0.06	
Drag Coefficient	0.335	
Frontal Area	2	m^2
Rolling Coefficient	0.006	
Bearing Loss	9.6	Nm
Wheel Radius	0.282	m
Battery Voltage	200	volts
Battery Resistance	0.0045	ohms

Table 4 Component Power Requirements for SmartHEV and ADVISOR

Component	**SmartHEV**	**ADVISOR**
APU	46 kW	49 kW
Battery Modules	29 @ 12 V	29 @ 12 V
Alternator	44 kW	56 kW

Numerical Techniques and Algorithms

The five logic-based algorithms, Rewriter, VarSelectSolution Path, Best Solution Path, and Swap-and-Solve, have been successfully implemented into SmartHEV. A variety of combinations of known and unknown variables can be selected with confidence, knowing that a solvable set of equations is maintained. If a particular arrangement of known and unknown variables is difficult to solve, SmartHEV will swap variables to find a viable solution path. Information provided to the user while selecting variables and component parameters is extremely helpful and insightful. Knowing which components are affected by changes in particular variables guides the user towards workable design solutions. The results are fast, accurate, and easy to follow.

Future Work

SmartHEV is still in the validation process. It will be calibrated with NIATT's FutureTruck 2000 Suburban data. SmartHEV will then be incorporated into an on-road vehicle simulation program called Clean Vehicle Energy Management, which was developed at the University of Idaho.

Further work on the logic-based algorithms will continue, specifically in the areas of tracking solved values and optimizing the Swap-and-Solve routine. We plan to add a Singular Value Decomposition (SVD) routine that will identify over-constrained systems of equations. These added features will lessen even more the dependency on providing accurate guess values for unknown variables and add power and flexibility to the design process.

REFERENCES

1. Glumbik, J. P., “Variable Interactions within Design Equations: A Methodology in Equation Management,” MS Thesis, University of Idaho, 1997.
2. Benson, J. L., “Simultaneous Equation Algorithms for Variable Interaction in the Solution Path,” MS Thesis, University of Idaho, 1998.
3. Sanders, M. S. *Human Factors in Engineering and Design*. 7th Ed. New York: McGraw-Hill, 1993.
4. US Department of Energy, “SIMPLEV: A Simple Electric Vehicle Simulation Program,” Version 2.0, DOE, Idaho Falls, ID
5. Blackketter, D. M., Kinematics and Dynamics Software, University of Idaho, 1996.
6. Ramirez, F. W. and C. R. Vestal, “Algorithms for Structuring Design Calculations,” *Chemical Engineering Science*, vol. 27, 1972, pp. 2243-2222.
7. Lee, W. and Rudd D. F., “On the Ordering of Recycle Calculations,” *American Institute of Chemical Engineers*, vol. 12, 1966, pp. 1184–1190.
8. Lee, W., J. H. Christensen, and D. F. Rudd, D. F., “Design Variable Selection to Simplify Process Calculations,” American Institute of Chemical Engineers, vol. 12, 1966, pp. 1104–1110.