Revising Engineering Models: Combining Computational Discovery with Knowledge

Stephen D. Bay, Daniel G. Shapiro, and Pat Langley

Institute for the Study of Learning and Expertise
2164 Staunton Court, Palo Alto, CA 94306
sbay@apres.stanford.edu, dgs@stanford.edu, langley@isle.org

Abstract. Developing mathematical models that represent physical devices is a difficult and time consuming task. In this paper, we present a hybrid approach to modeling that combines machine learning methods with knowledge from a human domain expert. Specifically, we propose a system for automatically revising an initial model provided by an expert with an equation discovery program that is tightly constrained by domain knowledge. We apply our system to learning an improved model of a battery on the International Space Station from telemetry data. Our results suggest that this hybrid approach can reduce model development time and improve model quality.

1 Introduction

Building accurate mathematical models of physical devices is an important engineering task. For example, engineers at NASA have developed detailed models that describe the electrical power system on the International Space Station (ISS). The engineers use these models for many tasks, including mission planing, monitoring, and fault diagnosis [3, 4]. Because the components on the space station are run close to operating limits, the models must be very accurate, as there is little room for error.

However, accurately modeling a physical device is a difficult problem for several reasons. First and foremost, device modeling is an inverse problem that involves reasoning backward from observations on a device's behavior to possible equations that may have generated the data. Second, our knowledge of most devices is incomplete. For instance, engineers commonly assume constant operating conditions for variables whose affect is not fully understood. Finally, device modeling involves many practical difficulties. For example, data for model development is often available only for a limited range of conditions and may not cover the deployed situation. This is especially true for ISS components, whose operating conditions cannot be easily duplicated. Additionally, testing a component on a lab bench will not account for interactions with nearby devices or changes as the device ages.

If the structure of the model (i.e. the forms of the equations) is known, but not the specific values for parameters, many techniques can learn the missing parameter values from data. However, a more likely situation is that the structure of the equations, and perhaps even the set of relevant variables, are not

completely known. This leaves the engineer with the difficult task of building an appropriate model manually from domain principles and her intuitions.

Building models manually is an iterative and time consuming process whereby an engineer may specify an initial model, tune its parameters, and then test it against data. If the model's performance is inadequate, the engineer will revise the model and repeat the process until she is sure that it is accurate enough for the intended task. This trial and error approach is cumbersome, especially with many parameters or possible model structures.

An alternative is to rely on computational methods to automatically discover a model. For example, equation discovery programs, such as Bacon [5] and Lagramge [8], take data in the form of observations and attempt to find equations that govern the relationship between independent and dependent variables. This approach is appealing because it automates much of the modeling process. However, equation discovery methods can suffer from very large search spaces and require strong constraints to limit the search space [8].

In this paper, we propose and formalize a hybrid modeling technique that combines the engineer's knowledge about a device with machine learning methods. In particular, we use engineering knowledge to constrain the search for better models and we use computational discovery programs to manage search, parameter fitting, and model scoring. We believe this approach has several advantages. From the engineer's perspective, a hybrid approach would let them focus on identifying possible refinements and explore a wider set than could be done manually. From a computational perspective, domain knowledge massively constrains the search space and makes equation discovery feasible.

We demonstrate this hybrid approach by revising battery models to better explain real-world behavior. In the next section, we begin by describing a simple battery model and showing how an engineer might revise it to explain complex non-linear behavior. In Section 3, we present our method for combining equation discovery and background knowledge. In Section 4, we evaluate our method on revising the battery model and show that much of the non-linearity can be recovered. In Section 5, we test our approach on improving battery models for the International Space Station from telemetry data. We then discuss limitations and related work, and conclude with a discussion of future research.

2 An Engineering Approach to Model Revision

Iterative refinement is a common engineering approach for modeling a device. An engineer starts with an initial model that is not perfect, but that explains much of the known behavior. Next, the engineer makes successive changes to the model to improve its predictive power. In this section, we give an example of this process from battery modeling. Although battery models have existed for many years, they are complex electro-chemical devices that are not well understood. Battery modeling is an active research area and new models are continually being published.

Figure 2 shows a simple battery model drawn as an equivalent electric circuit. In the model, V_{cb} represents the battery voltage of an ideal cell. The term

R_s represents a resistor connected in series to the battery cell and models the battery's internal resistance to current flow when the circuit is completed. The term R_p is a resistor connected in parallel to the battery cell and represents resistance to self-discharge. In this model, V_{cb}, R_p, and R_s are constants and cannot be directly observed. The state of charge (soc) is a measure of the total electric charge stored in the battery.

To complete the electric circuit, the battery must be connected to another device, which we will call a controller. For this paper, we assume that the controller is an active device that regulates the charging and discharging of the battery. It charges the battery at constant current and discharges at constant resistive load. The battery interacts with the controller through i and V_t, which are the current into (or out of) the battery and the voltage at the battery terminals, respectively. The variables i, V_t, and soc are observable.[1]

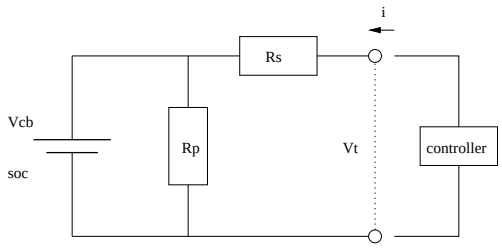


Fig. 1. A battery model.

Although this component model appears simple, it maps onto a complex set of equations that govern the input/output relationships of the battery. The terminal voltage, V_t, is determined by Equation 1 during charge and Equation 2 during discharge. The battery's state of charge is modeled by Equation 3, which is a differential equation that states the rate of change is equal to the current flow minus loss through the resistor R_p.

$$V_{t\_charge} = V_{cb} + i \times R_s \tag{1}$$

$$V_{t\_discharge} = \frac{V_{cb} \times R_{load}}{R_s + R_{load}} \tag{2}$$

$$\frac{dsoc}{dt} = i - \frac{V_{cb}}{R_p} \tag{3}$$

This model can explain much of a battery's behavior, but it is not adequate for many applications. Chan and Sutanto [2] point out several deficiencies and suggest modifications to improve its fidelity.[2] First, the model fails to explain

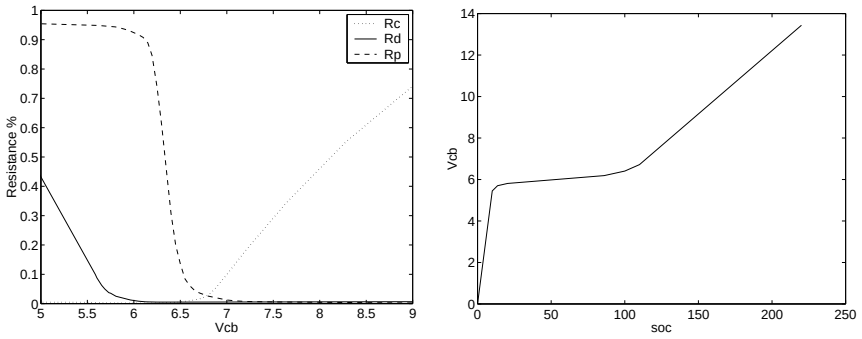
[1] State of charge may not be observable in some batteries. For our work modeling components on the space station, the batteries are Nickel-Hydrogen pressure cells and soc can be observed indirectly through the battery's temperature and pressure.

[2] In their paper, Chan and Sutanto examine five historical models and point out their deficiencies before suggesting an improved version. The model in Figure 2 is not identical to any of the five models but has many common elements with them.

changes such as the apparent series resistance, R_s, depending on whether the battery is charging or discharging. They suggest an improvement where R_s is equivalent to a resistor R_c during charge and a resistor R_d during discharge. Second, the model ignores dependence of battery properties on the state of charge. For example, real batteries become much more difficult to charge when they are nearly full compared to when they are empty. This could be represented in the model by making R_c a monotonically increasing function of *soc*. In general, all of the terms V_{cb}, R_p, R_c, and R_d will depend on battery properties and are not constants.

Chan and Sutanto focused their paper by modeling a specific battery from a given manufacturer. They made R_c, R_d, and R_p functions of V_{cb}, which in turn depended on *soc*. Although there is some expectation about the general shape of these functions, the exact forms were not known and they resorted to the manufacturer's test data to determine the functions empirically. Figure 2 shows the functional forms, with the dependent variable on the y axis.

The curves in Figure 2 can be obtained by performing in-depth battery testing, ideally for each specific physical device. However, some tests can be destructive and shorten the lifespan of the battery, such as those involving deep discharge. Manufacturers often provide these curves for a typical battery, but they are not specific to an individual physical device and may not cover the relevant operating conditions or external effects. This provides a perfect opportunity for machine learning techniques to improve existing models by allowing adaptation in response to observational data.



(a) (b)

Fig. 2. Dependence of battery parameters on other variables. (a) R_c, R_d, and R_p versus V_{cb}. Resistance is scaled by the maximum observed value. (b) V_{cb} versus *soc*.

3 Combining Equation Discovery with Knowledge

Our goal is to help the engineer with the revision process and to support the types of refinements described in the previous section. We envision a system where the user can input information about her modeling problem, including data on the

specific device she is modeling, and the system would suggest several revisions to the model that better explain the observed data.

The key insight of our work is that engineers will not suggest arbitrary changes to a model. Although they may not know the exact changes needed, they have a good idea of where their model is wrong because they know the approximations and assumptions made in the model's development. We feel this knowledge can be leveraged by computational tools.

3.1 Problem Definition

We can state the problem of revising an engineering model as follows:

- Given: an initial set of equations that describe the system's behavior;
- Given: data on the observable variables in the equations;
- Given: knowledge about the equations and how they might be modified;
- Find: an improved model that better explains the data.

Knowledge about the equations takes two forms in our current system. First, the user can specify plausible values for parameters, such as a valid range or an initial guess. For example, in the battery model in Figure 2 the user can state that R_s is between 0 and 10 ohms with an initial guess of 0.1. Second, the user can specify that a term which is a parameter in the initial model may depend functionally on other variables in the analysis. The user can also specify a set of plausible independent variables and possible functional forms. For example, she may believe that V_{cb} is not a constant and is possibly a quadratic or sigmoidal function of other variables such as *soc* or *temperature*.

Our problem definition is stated as a "single shot" process that is solved once, but clearly refinement can be iterative. Often the errors from one stage of revision will suggest new refinements that can further improve the model. This leads to a set of relevant models, each progressively explaining more of the data.

3.2 Transformation into Equation Discovery

We transform our problem into an equation discovery task. We use Lagramge [8], which is a program for equation discovery that can find both ordinary differential equations and regular algebraic equations that describe the data. The system uses a context-free grammar to define a space of possible equations that may explain the observed data. Lagramge searches through the space of equations defined by the grammar, evaluates each candidate model on the data, and returns the best models according to a score function.

Our system takes the knowledge specified by the engineer and compiles a highly constrained grammar to search for revisions of the initial model. The knowledge is transformed according to three rules:

- the initial equation becomes the starting state of the grammar;
- variable dependencies are encoded as symbol expansions in the grammar;
- knowledge about the values of constant parameters are passed to Lagramge to be used in parameter fitting.

For example, consider trying to revise the model of V_t, the voltage at the battery terminals. We first transform Equations 1 and 2 into an initial starting sentence,

$$V_t \rightarrow \delta(i)(V_{cb} + i \times R_s) + \delta(-i)\frac{V_{cb} \times R_{load}}{R_s + R_{load}}$$

where $\delta(x)$ is an indicator function that is one when x is positive and zero otherwise. Note that i is defined as positive when current flow is into the battery, and the above production covers both charge and discharge conditions. Next, if we believe that V_{cb} may depend on the variables *time*, *soc*, and *temperature*, with possible forms that are sigmoidal or linear (in one or two variables), we obtain the following productions for the grammar:

$$\begin{aligned} V_{cb} &\rightarrow const_1 + const_2/(1 + e^{(X - const_3)const_4}) \quad | \\ &\quad const_1 + const_2 X + const_3 X \quad | \\ &\quad const_1 + const_2 X \\ X &\rightarrow temperature \mid soc \mid time \end{aligned}$$

Finally, any information about the constants is passed through the grammar to Lagramge, for instance,

$$\begin{aligned} R_{load} &\rightarrow const[0:10:2] \\ R_s &\rightarrow const \end{aligned}$$

The constant R_{load} is given an allowable range from 0 to 10 with an initial guess of 2, and R_s is left unspecified.

To select the best revision produced with the grammar, we use Lagramge's minimum description length (MDL) score function, which evaluates a candidate model by taking into consideration both the sum of squared error on the training set and the model's complexity, measured as the size of its parse tree.

4 Revising a Battery Model with Synthetic Data

To demonstrate the feasibility of our revision approach, we used synthetic data to test our system's ability to refine initial models. Synthetic data lets us compare the discovered changes with the true structure.

We used Equations 1 to 3 in conjunction with the battery parameters in Figure 2 to generate synthetic data by simulating it in Matlab with an ordinary differential equation solver (`ode113`). We assumed that the controller cycles and charges the battery at constant current followed by discharge at constant resistive load ($R_{load} = 2\Omega$). We examined two cases: (1) Charging occurs with current $i = 3A$, which results in steady cycling of *soc* from about 98% to 73%; (2) charging occurs with current $i = 2A$, which results in a gradual loss of *soc* from 92% to 30%. For each case, we generated data for 1000 time points (eight cycles). We added an irrelevant variable, *temperature*, that varied sinusoidally with a period matched to the charge-discharge cycles.

For the initial model, we used Equations 1 to 3 with all parameters considered constants. We tried a simple scenario in which an engineer might believe that R_s and R_p are well modeled as constants with respective ranges and initial values of [0:100:1] and [0:200:100]. The engineer may also believe that V_{cb} is not a constant and could depend on other variables such as *temperature*, *soc*, or *time*, with a functional form that is a polynomial (up to third degree) or a sigmoid.

The above statements were automatically compiled into a grammar from a file specification and then used as an input to Lagramge. During execution, the program expands the grammar and examines 13 different revisions. The best revision according to Lagramge's MDL score involves expanding V_{cb} to be a linear function of *soc*. Figure 4a shows the target signal, and Figure 4b and c shows the reconstruction error for the initial and revised models. The results indicate that the revised model was better able to reconstruct the signal.

Although the revised model reduced the error for case 1, the error was still sizeable. We performed another refinement iteration in which we let R_s depend the variables *time*, *temperature*, or *soc* with a polynomial form. We recompiled the grammar file and reran Lagramge, which explored 240 possible revisions and suggested expanding R_c as a quadratic function of *soc*. Figure 4d shows the reduced error of this new revision compared with the first refinement.

Equations 4 and 5 show the final results and the revisions have moved the initial model closer to the curves in Figure 2. The linear expansion of V_{cb} on *soc* partially reconstructs the curve in Figure 2b, and the quadratic expansion of R_c attempts to model the sharp increase in R_c with increases in *soc*.

$$V_{t\_charge} = (5.84 + 0.00451soc) + i \times (0.145 - 0.00527soc + 4.90E{-}5soc^2) \quad (4)$$

$$V_{t\_discharge} = (5.41 + 0.00876soc) \times 2/(2 + 0.00495) \quad (5)$$

Finally, we note that the parameters in the linear equations that represent V_{cb} differ slightly in the case of charge and discharge. This is caused by a limitation of Lagramge, which its authors are addressing.

5 Modeling Batteries on the Space Station

Our experiments with revising battery models on synthetic data showed that our system can refine initial models to explain complex, non-linear behavior. In this section, we apply our approach to battery models for the International Space Station with real telemetry data to show that it can develop accurate models and is robust to problems in data quality.

We modeled the batteries for a single power channel on the Space Station. Within a power channel there are three battery units that each contain two sets of 36 nickel-hydrogen cells. We treated the entire collection of 216 cells as a single battery, and here we focus on modeling the battery's terminal voltage, V_t.

We have telemetry data for 24 hours with samples approximately every ten seconds. Only a fraction of the cells are instrumented with sensors, so we averaged readings from six cells to obtain the battery's temperature and pressure.

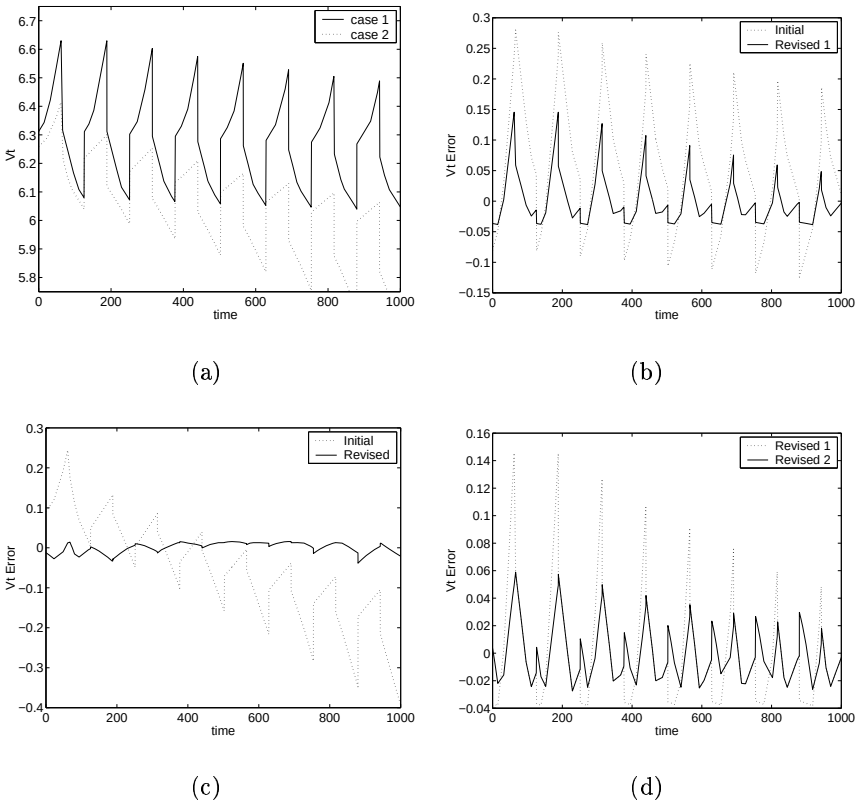


Fig. 3. Original signal and reconstruction error for (a) target V_t. (b) case 1 for the initial model and first refinement, (c) case 2 for the initial model and first refinement, and (d) case 1 for the first and second refinements.

We estimated the state of charge by the ratio of *pressure* to *temperature*. The current and voltage were available for each group of 36 cells (six total) and we summed and averaged them to get total current and terminal battery voltage.

The data are very poor quality and suffer from several problems. First, the signals for the observed variables have many dropouts for long time periods and this affects approximately 1/4 of all time points. Second, because of bandwidth limitations, the signals are encoded at low resolution. For example, the sensors can only report current flow to the nearest Ampere. Finally, the data show evidence of non-Gaussian noise that manifests itself as large spikes in the signal.

We linearly interpolated the data to register time points at ten second intervals and to impute missing values. Figure 5a shows the target variable V_t after this processing. We divided the data into a training set, of approximately three quarters of the data (before the dashed line in Figure 5a), and a test set consisting of the remaining data. We used Equations 1, 2, and 3 for our initial model.

As possible refinements, we let V_{cb} be a function of the variables *temperature*, *pressure*, or *soc* with possible functional forms that are polynomial (up to third degree), sigmoidal, or linear in two variables. We let R_s depend on the same variables with a polynomial form.

Lagramge explores 6859 revisions and takes approximately nine hours of computation time on an 1.5 Ghz Pentium 4.[3]The top ranked revision, shown in Equations 6 and 7, modifies the initial model by representing V_{cb} as a linear function of *soc*. Figure 5b shows the prediction error on the test data, which is much smaller than the error of the initial model.

$$V_{t\_charge} = (36.2 + 76.2 \times soc) - i \times 0.214 \tag{6}$$

$$V_{t\_discharge} = (20.3 + 36.2 \times soc) \times 5.77/(2.60 + 0.408) \tag{7}$$

Table 1 shows summary statistics for the initial model and the top three revisions returned by Lagramge. These results indicate that the revised models greatly improved the test error compared with the initial model. The mean squared error (MSE) for the revised models are approximately one third that of the initial model. However, MSE is sensitive to outliers, so we also report mean absolute error, which is more robust. On this measure the revised models all obtained an average error of about one volt. This is surprisingly good, considering that the individual sensors only resolve to one volt. Finally, the difference in predictive performance between the revised models is not substantial. Because the second and third models add extra complexity but do not significantly improve the models, they are rated worse with Lagramge's MDL score function.

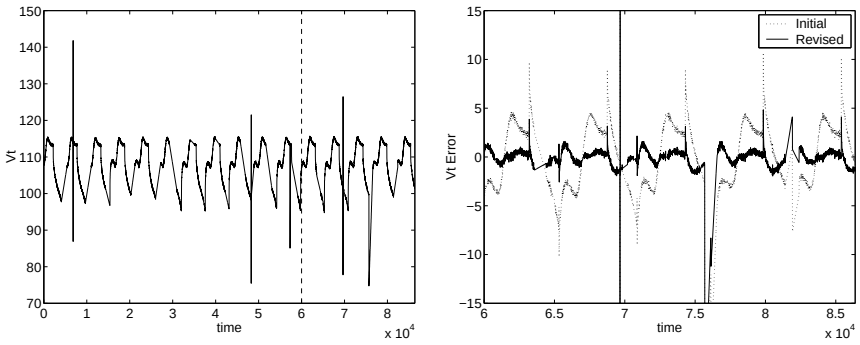
Table 1. Error statistics on the training and test data: Lagramge's MDL score, mean squared error (MSE), and mean absolute error.

	Training		Test	
	MDL	MSE	MSE	Mean Abs.
Initial Model				
V_{cb}, R_c, and R_d are constants	n.a.	12.3	20.5	2.75
Best Revised Models				
1. V_{cb} is a linear function of *soc*.	2.65	2.14	6.99	1.01
2. V_{cb} and R_d are linear functions of *soc*.	2.67	2.12	6.95	1.00
3. V_{cb} is a linear function of *soc*; R_d is a linear function of *temperature*.	2.68	2.13	6.99	1.00

6 Limitations

We demonstrated with experiments that our system can successfully refine initial models to better explain data. However, our revision approach has four important limitations that we discuss here.

[3] The number of revisions is much greater than for the synthetic example in Section 4 because the number of sentences that can be produced by grammar can expand exponentially with additional productions.



(a) (b)

Fig. 4. Battery terminal voltage V_t and prediction error. (a) Training and test data for V_t. (b) Error predicting V_t.

First, our system focuses the search for better models by exploring revisions that are near an initial model. This provides tremendous power if the true model is close to the initial model. However, if the true model is structurally very different, then searching near the initial model will not find the necessary revisions.

Second, our system depends on an expert to suggest plausible functional forms that may explain the values of dependent variables in the initial model. Again, this provides tremendous power if the expert provides specific forms that closely match behavior in the real physical device. However, our experiments suggest that our system may be robust to mis-specification. On synthetic data, it was still able to significantly improve the initial model even though none of the functional forms exactly matched the relationships in Figure 2. On real data, even though we have a limited knowledge of battery dynamics, the forms we suggested were capable of greatly lowering the prediction error.

Third, the suggested revisions are conditional on the data seen and may not generalize well to new operating conditions. For example, the models in Section 4 were revised on data that represented a battery whose state of charge varied from 98% to 30%. The revised model may not perform well outside this region, such as at very low charge levels. This limitation is not unique to our system, but applies to all induction algorithms.

Finally, Lagramge took over nine hours to revise the battery model for the Space Station. This is clearly too slow to support iterative and interactive refinement with an engineer. We are examining methods to speed up Lagramge with techniques such as error bounds to eliminate poor candidates quickly.

7 Related Work

Our approach builds on recent work by Todorovski and Dzeroski [9] who proposed revising a mathematical model by providing Lagramge with a grammar

that encodes a specific set of changes. They let Lagramge refit the value of known constants based on the data, and they supported replacing a polynomial in the original equation with a polynomial of arbitrary degree (on the same variables). In their application, a major goal was minimal change with the initial model, so their implementation examines each revision separately and then considers only a few combinations.

Our application of revising models of engineering devices, specifically the devices on the space station, has driven our work in a slightly different direction. We start with the assumption that the model is wrong, because of approximations and different operating conditions in orbit, and that the engineers can (mostly) identify the parts of the model that need to be revised. Because of these assumptions we allow many more changes to the model. Specifically, we allow revisions involving a variety of functional forms, we allow these forms to depend on different sets of variables, and we consider all changes at once to catch interactions.

Other work in equation discovery has also incorporated domain knowledge, but in different ways. Washio and Motoda [10] developed SDS, a program that uses dimensional analysis to constrain the possible equations. Bradley, Easley, and Stolle [1] developed PRET a program that automatically tries to find an ordinary differential equation model of a physical system. PRET uses automated reasoning about modeling techniques to select from a set of traditional system identification methods.

Finally, we have focused on developing interpretable and transparent models that can be examined by the engineer. An alternative approach is black box techniques, such as neural networks (e.g., [6]), which model a device's input/output behavior without attempting to find a concise mathematical description. Neural networks are not always applicable because they are not transparent and are difficult to verify. However, recently Saito et al. [7] have started to address this drawback in the context of revision by examining methods that use neural networks to learn interpretable structures.

8 Conclusions and Future Work

We presented an approach for combining machine learning methods with an engineers' knowledge to revise models of physical devices. The engineer specifies an initial model and possible revisions to that model and we combine this with an equation discovery program to manage the search process. Our experiments showed that this method can successfully revise models of physical devices from noisy sensor data and substantially improve their accuracy.

Our work represents a first step toward a computer assisted environment for revising models of physical devices. This approach is promising and may speed model development by relieving the engineer of tedious computational tasks. We also believe this approach may lead to better models by allowing exploration of a wide set of refinements and adaptation to observed data.

There are many directions for future work and we highlight three areas: First, we intend to apply our approach to improving models of other components on the

space station. Second, we intend to expand the types of qualitative knowledge that an engineer can specify to constrain the search space. For example, in addition to specifying a set of relevant variables, the engineer can also specify the general effect of those variables. For instance, in our battery model R_s should increase with *soc* as it becomes progressively more difficult to charge a battery as it nears maximum capacity. We can eliminate models without this behavior. Finally, we intend to explore how the engineer can interact with the search process, possibly by specifying a search order or viewing intermediate results and selecting particular paths to follow.

Acknowledgments

This work was supported by grant NCC 2-1220 from NASA Ames Research Center. We thank Rick Alena and Daryl Fletcher for providing access to the data, Ljupco Todorovski and Saso Dzeroski for their help with Lagramge, and Javier Sanchez for assistance with simulation tools.

References

1. E. Bradley, M. Easley, and R. Stolle. Reasoning about nonlinear system identification. *Artificial Intelligence*, 133:139–188, 2001.
2. H.L. Chan and D. Sutanto. A new battery model for use with battery energy storage systems and electric vehicle power systems. In *Proceedings of the IEEE Power Engineering Society Winter Meeting Conference*, 2000.
3. J. S. Hojnicki, R. D. Green, T. W. Kerslake, D. B. McKissock, and J. J. Trudell. Space station freedom electrical performance model. In *Proceedings of the 28th Intersociety Energy Conversion Engineering Conference*, 1993.
4. T. W. Kerslake, J. S. Hojnicki, R. D. Green, and J. C. Follo. System performance predictions for space station freedom's electrical power system. In *Proceedings of the 28th Intersociety Energy Conversion Engineering Conference*, 1993.
5. P. Langley, H. Simon, G. Bradshaw, and J. M. Zytkow. *Scientific Discovery: Computational Explorations of the Creative Process.* The MIT Press, 1987.
6. J. Peng, Y. Chen, and R. Eberhart. Battery pack state of charge estimator design using computational intelligence approaches. In *Proceedings of the Fifteenth Annual Battery Conference on Applications and Advances*, 2000.
7. K. Saito, P. Langley, T. Grenager, C. Potter, A. Torregrosa, and S. A. Klooster. Computational revision of quantitative scientific models. In *Proceedings of the Fourth International Conference on Discovery Science*, pages 336–349, 2001.
8. L. Todorovski and S. Dzeroski. Declarative bias in equation discovery. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pages 376–384, 1997.
9. L. Todorovski and S. Dzeroski. Theory revision in equation discovery. In *Proceedings of the Fourth International Conference on Discovery Science*, 2001.
10. T. Washio and H. Motoda. Discovering admissible models of complex systems based on scale-types and identity constraints. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, pages 810–817, 1997.