Optimization of Dynamic Battery Parameter Characterization Experiments via Differential Evolution*

Joel Forman1, Jeffrey Stein2, and Hosam Fathy3

Abstract—Characterization is important for making models match reality and allowing for quick and accurate measurements of parameters. In this paper we present a method for designing dynamic battery experiments using an evolutionary algorithm that directly generates Pareto fronts via differential evolution. This optimization creates current trajectories for multiple objectives, namely, maximizing Fisher information gathered while minimizing battery damage. An estimator is used on simulated battery experiments to verify the improvements associated with these trajectories. This exercise illustrates the experimental trade-offs between gathering parameter information and causing battery degradation. The procedure in this paper is widely applicable as both the battery model and parameter’s of interest can be substituted as needed.

I. INTRODUCTION

Understanding, estimating, and managing battery health for ever increasing fleets of electrified vehicles is clearly of great financial and engineering importance. Financially, the quantity of electric vehicles continues to increase. For many of these vehicles, the battery will be a significant portion of their manufacturing cost. From an engineering perspective, battery health affects usability, including vehicle performance and range. This paper improves on understanding, estimating, and managing battery health by creating a systematic method for designing battery experiments and diagnostics based on tools from optimal experimental design and evolutionary optimization. These experiments and diagnostics gather the greatest amount of battery parameter estimation information possible while minimizing damage to the battery. We demonstrate this method by using it to create Pareto fronts of current trajectories that maximize Fisher information (in the D-optimal sense) while minimizing battery damage. The generated trajectories are simulated with artificial measurement noise which is fed to an estimator for validation. This creates sets of parameter estimations whose statistics can be related to Fisher information. Since this method improves parameter estimation accuracy while minimizing battery damage, it can improve both battery State-of-Health (SoH) estimation and battery health modeling experiments.

The trajectories designed by this method have a variety of practical applications. Firstly, the diagnostics based on these trajectories can be implemented to run while the vehicle charges overnight. The estimation algorithm can then be computed on-site with a micro processor or off-site via cloud computing. After this computation the vehicle’s estimate of SoH would be updated, allowing the vehicle controllers to make better decisions. This is different from traditional SoH estimation methods in that it is both offline and the trajectory the battery undergoes is prescribed [1]–[5]. We believe conducting estimation offline is not only appropriate but actually beneficial. It is appropriate, as the SoH changes very slowly and beneficial because it does not increase the online computational burden. Additionally, the trajectory is custom designed rather than imposed by the driver and vehicle which helps to increase information content. Another interesting application is the improvement of long term battery health experiments. An ideal battery health test for long term experiments would be short in duration (to minimize the time the battery does not spend cycling), cause very little damage (so that observing health has a minimal impact on the measured results), and maximally informative about the parameters of interest. The trajectories created by this method help with all three of these goals: one can specify the duration and then use the Pareto front to choose an appropriate trade-off between information and damage.

The major difference between our approach and that typically used for battery experiments is the scope of possible trajectories considered. Typically one works with a small finite set of trajectories and the problem is treated as a subset selection problem. This is the case in the health experiments in [6] as well as by Schmidt et al.’s work fitting a single particle model [7]. Rather than limiting ourselves to preselected trajectories, we use an algorithm that explores the function space (up to a sampling rate of 5 Hz). This has a variety of advantages. Firstly, by being dramatically less restrictive, it enables the optimization to achieve greater improvements. Second, it enables us to gain insight into what types of trajectories are best for this identification work. Finally, it allows one to approach situations where it is unclear which trajectories might be appropriate. The cost of these advantages is a dramatic increase in the computational complexity of the optimization. As computers continue to become faster this will continue to become less disadvantageous. Presently the authors overcome this difficulty by making use of parallel computing clusters. This allows us to simulate 10 minutes of the Doyle-Fuller-Newman (DFN) model roughly 1.5 million times per optimization.

For this work we focus on estimating parameter’s of the DFN model. Intuitively, ease of estimating parameters...
depends on how slight perturbations of parameters affect the measured outputs of the model. This work designs input current trajectories that maximize the parameter estimation information carried by the voltage output. This idea of estimation information is formalized by Fisher information, which uses the output parameter sensitivities to compute information content. In the linear case (the DFN model is nonlinear) Fisher information provides a bound on best possible parameter estimation behavior [8]. In the nonlinear case it is a useful guide for improving estimator performance. Each current trajectory has an associated Fisher information value, and this value is used as one of the two optimization multi-objectives.

An excellent evolutionary algorithm for multi-objective optimization is the Nondominated Sorting Genetic Algorithm II (NSGAIIt) which evolves populations to directly create Pareto fronts [9]. Another useful evolutionary algorithm is Differential Evolution (DE), which works directly on real encoded problems and has been shown to be very effective in a variety of applications [10][12]. The advantages of each of these evolutionary methods are married by Kwan, Yang, and Chen who created the NSGAI-DE algorithm [13]. This algorithm combines the performance of DE with the advantage of direct Pareto front creation, and thus it is ideal for our problem.

By pairing the health metric with Fisher information we create an optimization problem for which NSGAI-DE generates a Pareto front. The Fisher information is based on the simultaneous estimation of two parameters: $d_2$, solution diffusivity and $R_{SEI}$, the resistance due to anode film thickness. Each of these relates to battery health, and $d_2$ is chosen specifically due to previous difficulties in its estimation [14].

The optimization results in a Pareto front. At high damage ranges there is little to no improvement in Fisher information. However, at mid to low damage ranges the Fisher information improves, frequently by orders of magnitude. The practical nature of these improvements are validated through the repeated use of an estimator on simulated data. This validation finds that Fisher information is an excellent qualitative guide for the estimator’s performance. However, it is a poor quantitative predictor as the estimator dramatically out performs the predicted Fisher information. This is likely due to Fisher information being based on linear assumptions whereas the DFN model is highly nonlinear - this idea is further considered in the discussion section.

The remainder of this paper focuses on the specifics of optimal trajectory generation, estimation and results. Specifically, §III presents the problem in a mathematical context and discusses how the application is optimized. §IV presents the results of the optimization along with estimator results based on simulated noise (as a validation). A discussion follows in §V where assumptions, computational advantages, and extensions to the method are discussed. Finally §VI contains concluding remarks and a summary of this paper’s contributions.

II. BATTERYS AND ELECTROCHEMICAL MODEL
Lithium Iron Phosphate

We focus on cells $26650$ cells LiFePO$_4$ cathodes. They are capable of very high rates, including a continuous 30 C-rate discharge. They have a nominal voltage of 3.3V and an operating region of 2.0V to 3.6V. The name plate capacity of these batteries is 2.3 Amp-Hrs. Of course, the methods within this paper could be applied to other battery types, provided the correct parameters are available for accurate simulation of the DFN model.

The Doyle-Fuller-Newman Model

The DFN model originates from work by Doyle, Fuller, and Newman published in 1993 and 1994 [15], [16]. Much has been written about this model and it has been used in a variety of applications regarding batteries. Of specific interest is [17], which fits many parameters of the DFN model to experimental data for the LiFePO$_4$ cells that are used within this work. The salient features of the DFN model are diffusion in both the solution and the solid, where the distribution in the solid is distributed in a pseudo 2D manner. These are connected to the electrical equations within the battery by two highly coupled nonlinear boundary value problems that control the flow of current and voltage distributions within side the battery. This creates a nonlinear DAE which can be very computationally intensive to solve. We use a combination of model reduction methods to simulate the model, see [18] and article [17]. The DFN model excels in areas where there are high rates that are highly transient as the model stems from first principles (it is based on binary concentrated solution theory).

The importance of the model in this application is two-fold. A model is obviously needed for simulation. However, the model also plays an important role in how the Fisher information is defined. Fisher information is related to the ease of estimating various parameters, but these parameters must be from a model. In fact, different Fisher information values can result for a physical parameter if the model used in calculation is changed. In addition, for Fisher information to be useful the model used must be reasonably accurate. For all of these reasons we use the DFN model solely throughout this paper.

Parameters of Interest

In this work we focus on two parameters of interest in the DFN model. The first, $R_{SEI}$, is the resistance of the solid electrolyte interface layer on the anode and it is related to battery health in two ways. First, creation of this film consumes the batteries cyclable Li-ions which decreases the maximum capacity. Second, the film resistance can cause power fade (at least within the DFN model) as the resistance decreases the batteries ability to provide power. The second parameter $d_2$, the diffusivity of the electrolyte, is slightly related to battery health as its decrease represents the electrolyte clogging within the battery. However our major interest in this parameter is as a challenge to the method - this parameter has been shown to be very difficult to
identify in a previous work by the authors [17]. Of course any combination of model parameters could have been used, and these two were chosen for demonstration. It is worth noting the that method is not limited to the two parameter case - Fisher information can be readily computed for larger sets of parameters.

III. TRAJECTORY OPTIMIZATION METHOD

Our goal is to design dynamic experiments to maximize the Fisher information gained regarding relevant parameters while minimizing battery damage. To this end we design open-loop current trajectories and measure the voltage response of the cell. These current and voltage trajectories are then fed to an offline estimator which attempts to minimize voltage error ($L^2$) by assigning the model appropriate parameter values. This section explains the mathematical and computational specifics of this procedure. The following section discusses the results for the trajectory optimization and the estimator’s performance.

Optimization Formulation

The result of this optimization is a set of current trajectories that form a Pareto front for identifying these two specific parameters, although the problem could easily be reformulated for different parameters or more parameters. Solving this problem enables more efficient experiments for battery characterization by better designing input trajectories.

This problem is formulated as a multi-objective open-loop trajectory optimization. Our first objective is to optimize Fisher information. For multiple parameters different versions of Fisher information exist. We use D-optimal as this minimizes the area (or volume/content for dimensions greater than two) of the the estimation ellipses. This makes for more accurate estimations, provided one is using an appropriate estimator. The second objective is to minimize the damage caused by the trajectory (invasiveness). We use the health model from [6]. For the signal we allow any currents even of magnitude less than 2.5 C-rate. Current trajectories that result in the voltage being less than 2.0V are then fed to an offline estimator which attempts to minimize battery damage. To this end we design open-loop current trajectories and measure the voltage response of the cell. These current and voltage trajectories can be treated as a differential equation for this section [26].

Consider the standard model formulation:

\[
\dot{x} = f(x; \theta) \\
y = g(x; \theta) + \text{error} \\
\text{error} \sim \text{Normal}(0, \sigma^2).
\]

Where $x$ is the state and $y$ is the output and $f$ and $g$ are the appropriate functions. $\theta$ is the parameter vector, that we are computing the Fisher information for. The astute reader may notice that this formulation is a differential equation rather than DAE - the DFN model has a low enough index that it can be treated as a differential equation for this section [26].

Of particular interest is the sensitivities at a set of instances in time, one can form a sensitivity matrix as follows [25]:

\[
S_{i,j} = \left[ \frac{\partial y(t)}{\partial \theta_i} \right]_{t=t_j}.
\]

Where the entry $S_{i,j}$ is the output sensitivity to parameter $\theta_i$ sampled at time instant $t_j$. The Fisher information is then given by [7], [8], [20]–[24]:

\[
\text{Fisher} = S^T S / \sigma^2.
\]

Where $\text{Fisher} \in \mathbb{R}^{p \times p}$. For optimization we use the D-optimal metric which is as follows [27]:

\[
F_D = \det(\text{Fisher})
\]

We compute the Fisher information by perturbing $\theta$ in each of its direction and simulating the DFN model. We use forward differencing (instead of central differencing) to decrease the number of necessary simulation calls. This allows us to compute the two sensitivities that we need with only three runs of the simulations (or $p+1$ simulations for $p$ parameters).

We note that the Fisher information is a local measure of information. This can be seen as it solely depends on the partial derivatives related to the output. For linear systems this can be used to accurately bound an estimator’s performance via the Cramé-Rao bound. For nonlinear systems this is only an approximation that we use as a guide for our optimizations. We continue further with this guide notion in §V, where we discuss the results of the optimization and simulated estimator.
### Table I

**Estimated Parameters for the Symmetric Model (Repeated)**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\beta_1$</td>
<td>$1.1484 \times 10^{-7}$</td>
<td>Amp $\times$ Hour $\times$ Sec$^{-1}$</td>
</tr>
<tr>
<td>$\beta_2$</td>
<td>$-3.9984 \times 10^{-8}$</td>
<td>Hour $\times$ Sec$^{-1}$</td>
</tr>
<tr>
<td>$\beta_3$</td>
<td>$-1.3158 \times 10^{-7}$</td>
<td>Amp $\times$ Hour $\times$ Sec$^{-1} \times$ Volt$^{-1}$</td>
</tr>
<tr>
<td>$\beta_4$</td>
<td>$-5.5487 \times 10^{-10}$</td>
<td>Amp $\times$ Hour $\times$ Sec$^{-1} \times$ Amp$^{-2}$</td>
</tr>
<tr>
<td>$\beta_5$</td>
<td>$4.9680 \times 10^{-8}$</td>
<td>Amp $\times$ Hour $\times$ Sec$^{-1} \times$ Volt$^{-2}$</td>
</tr>
<tr>
<td>$\beta_6$</td>
<td>$1.1660 \times 10^{-8}$</td>
<td>Hour $\times$ Sec$^{-1} \times$ Volt$^{-1}$</td>
</tr>
<tr>
<td>$\beta_7$</td>
<td>$-6.1065 \times 10^{-9}$</td>
<td>Amp $\times$ Hour $\times$ Sec$^{-1} \times$ Volt$^{-3}$</td>
</tr>
</tbody>
</table>

**Health**

As our measure of invasiveness we use the model from [6]. The battery health model is as follows:

$$\dot{h}(I, V) = \beta_1 + \beta_2 \parallel I \parallel + \beta_3 V + \beta_4 \parallel I \parallel^2 + \beta_5 V^2 + \beta_6 \parallel I \parallel + \beta_7 V^3$$

(5)

Where the $\beta$ parameters are given in Table I. The objective then is to minimize:

$$\text{damage} = - \int_{t=0}^{T_f} \dot{h}(t) dt.$$  

(6)

Where this model has been developed with long term cycling of multiple batteries undergoing different CCCV cycles.

**Trajectory Constraints**

The current trajectories in these specific optimizations are constrained to be exactly 10 minutes long. The sampling rate is at 5 Hz so each trajectory has 3001 control points. One could use splines and other methods to decrease the number of control points (indeed, the authors has in the past - cubic and Gauss-Legendre-Lobatto, see [28]). However decreasing the number of control points introduces filtering due to the choice of splines. Typically, when one expects a smooth trajectory as a result of an optimization this is not a problem. However, OED trajectories tend to have very high frequency characteristics. Due to this we opt to make every point in the trajectory a control point. As can be seen in §IV the optimized trajectories do have a high frequency component.

Additionally the voltage and current are both bound. The current is bound between $\pm5.75A$ ($\pm2.5C$-rate). This bounding is easy to implement - the optimization algorithm will not try trajectories outside of this. The voltage is bound between 2.0V and 3.6V which are the design specifications for these specific cells. This voltage constraint is enforced by rejecting population members that violate it.

**NSGAI with DE**

There are a multitude of optimization methods available in the literature. Evolutionary algorithms are typically very useful in environments with large numbers of variables and cases where models are noisy. In this case we happen to be dealing with both of these so these methods are ideal. Additionally there are a variety of ways to generate Pareto fronts. One can change the weights on each objective and reoptimize repeatedly. Or one can use an evolutionary method where one searches for the entire Pareto front at once by optimizing an entire population. NSGAI is such an algorithm [9].

NSGAI is the Nondominated Sorting Genetic Algorithm II and was created by Deb, Pratap et al. [9]. This algorithm starts as a typical genetic algorithm. However, rather than using a single objective related to fitness, it can use several. It does this by sorting the population into fronts, where members of each front are dominated by exactly $i$ members. Here, a member is dominated if another member has better values for all of the multi-objective functions. The best front is the front of members which are not dominated by any member of the population and this front forms the Pareto front. As the population evolves NSGAI selects members from the least dominated fronts and has a mechanism to avoid overcrowding sections of the fronts. Because of this, it can compute a Pareto front in one optimization, rather than having to reoptimize with different weighting factors on each of the objectives. This frequently saves large amounts of computational time, as the evolutionary algorithm does not have to repeat similar optimizations.

Originally NSGAI was designed to use a genetic algorithm for cross breeding and mutations - the mechanisms that create a new population from an old one. However, one can use other mechanisms to build new populations. We choose to use DE because it is naturally real encoded (as is our problem) and has been shown to be very effective on a variety of difficult optimization problems [12]. DE was introduced by Storn and Price in [10] and has performed very well as an optimization/search algorithm. The basic DE algorithm works as follows. An initial population is generated randomly. Each population member is then perturbed based on vector directions with in the population. Not all of the elements are written to the new vector, this is controlled by a weighted coin flip (with probability $C_R$). Fitness values for the new population are then computed, and if the perturbed members improve on their associated nonperturbed member they are used to replace them. One can refer to a variety of the literature on the subject for further specifics on the algorithm [10], [12], [29]. One of the major advantages of DE is it changes step size and direction based on the spread of the population members. When the spread is large DE creates large perturbations for searching the space. When the population members are close, the step size is proportionally reduced. Additionally, the optimization variable sensitivity is automatically taken into account with the population spread. All of this happens with a very low number of tuning parameters for the algorithm, making it easier to robustly apply to a variety of optimization problems.

Ideally one would like to combine the benefits of both NSGAI and DE. NSGAI-DE is such an algorithm - it uses NSGAI to directly evolve Pareto fronts and DE to improve its search and optimization [13]. NSGAI-DE uses an unmodified version NSGAI’s selection mechanism and a slightly modified form of DE’s member generation. This modification creates new members based solely off perturbations about the members of the nondominated front.

---

872
Each member of the nondominated front is equally likely to be used in each perturbation. These perturbations are then combined with the base population and the population is culled (thinned) by NSGAII to produce the next generation. This combination of NSGAII with DE is highly effective, as can be seen in the results section. The next subsection discusses the computational specifics of running NSGAII-DE for the trajectory optimization and plain DE for estimation.

**Computation**

There are two main computational problems within this work. The first is the NSGAII-DE optimization which evolves the current trajectories. To this end we require the Fisher information to be computed for each trajectory considered. The second examines the effects of these created trajectories through the repeated use of DE to estimate parameters based on simulated data with measurement noise. These DE optimizations adjust the estimation parameter values by computing and comparing voltage responses. Much of the computation between these two problems is similar, we discuss these similarities first and then follow with their differences.

In both of these problems efficient computation of the DFN model is critical. To this end we use the modal Legendre form of the DFN model with quasi-linearization, see [18]. We use a 5th order polynomial representation across the width of each cell region: anode, separator, and cathode. In the radial direction we use a 10th order polynomial representation, where the polynomials of odd degree are omitted due to symmetry. By using Legendre modes we reduce the number of differential equations to 56 and the number of algebraic equations to 30. In addition, we use quasi-linearization nearly identical to that in [17] except it is applied to the modal coordinates. We implement our simulation code in MATLAB although other environments may provide better computational speed (MATLAB is excellent for agile development).

Even with these computational reductions we still need to make use of parallel computing. Specifically we use a server-client architecture to parallelize the optimization function calls. The NSGAII-DE and normal DE run on the server, and then evaluation requests are sent out to the clients. This communication uses custom Java code to connect the server and clients over TCP/IP. We link the DFN model simulations to this code by calling it within the MATLAB environment. A combination of our in lab cluster and the University of Michigan’s Center for Advanced Computing cluster were used to conduct these computations. Maximally we had 56 cores running to compute the trajectory optimizations.

Another commonality between these two problems is that they both feature bounds on their optimization variables. To gracefully handle perturbations that find members outside these bounds we only halve their distance to the boundary when they violate it [11]. This performs better than placing them on the boundary, which would discourage population diversity.

### Table II

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial Population Members</td>
<td>4000</td>
</tr>
<tr>
<td>Population Members</td>
<td>1000</td>
</tr>
<tr>
<td>( C_r )</td>
<td>0.9</td>
</tr>
<tr>
<td>Number of Generations</td>
<td>550</td>
</tr>
</tbody>
</table>

Despite all these similarities the two problems have a few important differences. For the trajectory optimization we have a 3001 point current trajectory for 10 minute runs. To compute Fisher information we repeat the runs with the parameters perturbed. This allows us to compute an optimization evaluation with three DFN model runs. The specifics of the trajectory optimization are summarized in Table II. Where \( C_r \) is the probability that a specific value mutates and the number of generations was decided upon by stopping the optimization after it appeared to have converged for a substantial number of steps.

It is important to note how the optimization population is initialized. We use stochastic Algorithm 1 to create the initial trajectories. In this algorithm \( r_{i,j} \) is the \( i^{th} \) trajectory’s \( j^{th} \) value. This gives us a set of trajectories similar to white noise, but with nonzero means that allow the State-of-Charge (SoC) to drift. Originally we had only used white noise, but we discovered this useful heuristic through experimentation with the population initializations.

### Algorithm 1 Initial Population Generation

**Require:** Number

```latex
\textbf{Require:} 
\textbf{NumberOfTrajectories;}
\textbf{NumberOfControlPoints; \( I_{\text{max}} \)}

\textbf{for} \( i = 1 \rightarrow \text{NumberOfTrajectories} \) \textbf{do}
\textbf{for} \( j = 1 \rightarrow \text{NumberOfControlPoints} \) \textbf{do}
\quad \textbf{randomReal} \leftarrow \text{RandomUniform}[0,1]
\quad \textbf{randomSign} \leftarrow \text{RandomUniform}[-1,1]
\quad \textbf{if} \( p \leq 0.5 \) \textbf{then}
\quad \quad \textbf{\( r_{i,j} = 2 \times I_{\text{max}} \times randomSign \times randomReal \)}
\quad \textbf{else}
\quad \quad \textbf{\( r_{i,j} = 2 \times I_{\text{max}} \times randomSign \times (1 - (2 - p) \times randomReal) \)}
\textbf{end if}
\textbf{end for}
\textbf{end for}
```

While the estimation work is very similar there are some key differences. The estimator was run 50 times for each of the 12 cases (6 optimized and 6 initial). For the estimator we only need to run the DFN model once to get an evaluation (the trajectory optimization has to be run three times to compute Fisher information). There are only two inputs to the estimator’s optimization: the values of \( R_{SEI} \) and \( d_2 \). The optimizations are cut off after 500 generations - although this does not occur in any of these estimations. The stopping criteria is based on the spread of the top 50% (Stopping Criteria Value 1) of the population, and
TABLE III
DE ESTIMATOR VALUES

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Population Members</td>
<td>40</td>
</tr>
<tr>
<td>C_r</td>
<td>0.9</td>
</tr>
<tr>
<td>Stopping Criteria Value 1</td>
<td>0.5</td>
</tr>
<tr>
<td>Stopping Criteria Value 2</td>
<td>$1 \times 10^{-4}$</td>
</tr>
<tr>
<td>$\max(d_2)$</td>
<td>$1 \times 10^{-8} m^2 s^{-1}$</td>
</tr>
<tr>
<td>$\min(d_2)$</td>
<td>$0 m^2 s^{-1}$</td>
</tr>
<tr>
<td>$\max(R_{SE1})$</td>
<td>$1 \times 0.01 \Omega$</td>
</tr>
<tr>
<td>$\min(R_{SE1})$</td>
<td>$0 \Omega$</td>
</tr>
</tbody>
</table>

stops when the maximum distance between these selected population members is less than $10^{-4}$ (Stopping Criteria Value 2). This is the MaxDistQuick criteria from [30]. We base this not on absolute distance in the parameter space (as the parameters are scaled differently) but base it on the maximum and minimum values that the parameters are bound (artificially) by. This distance between the population members is computed as follows:

$$
popDistance(x,y) = \left( \frac{x_{d_2} - y_{d_2}}{\max(d_2) - \min(d_2)} \right)^2 + \ldots
$$

$$
\left( \frac{x_{R_{SE1}} - y_{R_{SE1}}}{\max(R_{SE1}) - \min(R_{SE1})} \right)^2
$$

(7)

We note that using DE for this estimation works very well, as does an excellent job quickly searching the space and the population members always come within close agreement.

We use the estimation sets to compute a Fisher information based on estimator performance. In a linear system this would be in near agreement with the trajectory optimization Fisher information. However in a nonlinear system these can vary from each other by quite a bit (as is shown later). To compute the Fisher information based on estimator performance we compute the covariance matrix of the estimations, invert this and compute the determinant:

$$
\hat{F}_D = \det \left( Covariance(\text{Estimations})^{-1} \right).
$$

(8)

The agreement and disagreement between the predicted and actual Fisher information is discussed further in the results and discussion sections.

IV. TRAJECTORY OPTIMIZATION RESULTS

This section discusses the trajectory optimizations and their related estimation results. These optimizations yield Pareto fronts of trajectories that increase the Fisher information content. However, since the DFN model is nonlinear, this improvement cannot be directly quantified through the Cramér-Rao bound. Instead, we quantify this informational improvement by applying an estimator to sets of simulated data containing measurement noise. This allows us to generate sets of estimations whose statistics quantify the trajectories’ practical estimation improvements. Some of the more interesting estimation sets are presented. These are followed by plots similar to Pareto fronts showing the improvements offered by the optimized trajectories as quantified by the estimator results. Together, these results demonstrate that using Fisher information as a multi-objective creates dramatic improvements in estimator performance.

Optimization Results

Figure 1 presents damage results, with Fisher information presented on a logarithmic scale. However, Fisher information is only a guide for this nonlinear model and next we present a more accurate quantification metric based on simulating estimator performance. Unfortunately, simulating estimator performance is not computationally tractable for use as a multi-objective in optimization (see the discussion section).

![Trajectory optimization Pareto front: Fisher information and battery capacity loss](image-url)

Estimation Results

By using an estimator on simulated data we are better able to quantify the improvements of using these optimized trajectories. Unfortunately it was not computationally tractable to run an estimator on every single trajectory. Instead we selected subsets of interesting trajectories from each Pareto front. Then, to compare with the initial population, we took the trajectories from the best initial front that were closest in value to the damage levels of interest. Table IV summarizes our selection of damage values.

To create a statistically meaningful set of estimations we ran the estimator 50 times on for each trajectory on simulated data. This data took the voltage measurement error to be zero mean Gaussian with $\sigma = 1 mV$ which is well within what is possible with laboratory equipment. The results of these estimator runs are a cloud each of 50 estimations of the $R_{SE1}$ and $d_2$ pair. The statistics of these clouds allow us to compute an estimator based Fisher information, which is used for comparisons both between estimators and from the initial predicted Fisher information from the optimization.

Estimator results for 1h and 2h are shown in Figures 2 and 3. Both of these yield excellent improvements over the best
trajectories in the initial set with similar damage values (see Table IV). 3h and 4h yield some improvement (plots not shown) and then 5h and 6h yield no meaningful improvements over their associated initial trajectories. Substantial improvements are made at the low to mid ranges and then no meaningful improvements are made at the high range.

Table IV

<table>
<thead>
<tr>
<th>Trajectory Number</th>
<th>Damage [µ Amp-Hr] Value</th>
<th>Optimal</th>
<th>Best Initial</th>
</tr>
</thead>
<tbody>
<tr>
<td>1h</td>
<td>1</td>
<td>1.022</td>
<td>1.011</td>
</tr>
<tr>
<td>2h</td>
<td>2</td>
<td>1.910</td>
<td>1.925</td>
</tr>
<tr>
<td>3h</td>
<td>4</td>
<td>4.035</td>
<td>4.014</td>
</tr>
<tr>
<td>4h</td>
<td>6</td>
<td>5.880</td>
<td>5.861</td>
</tr>
<tr>
<td>5h</td>
<td>8</td>
<td>7.976</td>
<td>7.980</td>
</tr>
<tr>
<td>6h</td>
<td>10</td>
<td>9.978</td>
<td>10.00</td>
</tr>
</tbody>
</table>

Now we use the Fisher informations calculated from the estimator results to compute the improvement of the optimizations. Figure 4 shows these results for the optimization. This has been normalized so that $\sigma = 1V$ to be consistent with the predicted Fisher information. The estimator based Fisher information is much higher than originally predicted by the optimization. However, the qualitative trends are similar when compared to figure 1. Thus we have found Fisher information to be an excellent guide for these optimizations.

V. DISCUSSION

This paper investigates two specific cases of designing current trajectories to gather information using NSGAII-DE. This can be thought of as demonstrating a general method, in which many variations are possible. The most obvious variation is that one could easily change the parameters used in estimation. Additionally, while the DFN model is a very interesting and important model, it is by no means the only model that could be used. If one is more focused on less complicated models (perhaps for control or optimization applications) then one could replace the DFN model with a single particle models [7], [31], [32].

While the method presented in this paper works well in simulation the discrepancies between the predicted Fisher information and the Fisher information observed by the estimator require some addressing. To reiterate, we observed similar trends for both, but the estimator based information is orders of magnitude higher than that predicted (this is after normalizing for the signal noise). Thus we suggest that the predicted information is a guide and computational tool. For example, to use the estimator based Fisher information in an optimization one would need a dramatically larger number of simulations. This is because while the predicted information can be computed with 3 simulations (2 perturbations and a center) the estimated information requires running the
estimator enough times to create a statistically meaningful cloud of estimations. For the health estimation we needed 104.5 simulations on average per estimation point. Assuming 50 points are needed for estimation (this is a rough number, but in the right order of magnitude), then we would need $50 \times 104.5 = 5225$ simulations for the estimator based Fisher information optimization. Clearly when compared with the 3 simulations needed to get the predicted Fisher information this becomes computationally intractable. With unlimited computing power using the estimator based method would be preferable, but given that these optimizations already take approximately 3 days when using 56 cores the use of predicted Fisher information has great practical value.

An interested researcher with (even greater) resources could scale this up and see what the true results would be.

VI. CONCLUSIONS

This paper creates and demonstrates a method for designing maximally informative and minimally invasive input trajectories for battery experiments. This method bridges an important gap between the dynamic optimal experimental design literature and the battery modeling literature. By using NSGAIIE the method allows one to algorithmically design these experiments making it more efficient to accurately characterize battery parameters. We note that substantial improvements at equal levels of invasiveness were possible in both the predicted information and the information observed by simulating the estimator on data with measurement noise. We believe this method is useful for both directly designing battery experiments and as a foundation for future dynamic optimal experimental design work for battery modeling experiments.

REFERENCES