Model-Based Prognostics

Matthew Daigle
Prognostics Center of Excellence
Intelligent Systems Division
NASA Ames Research Center

Objectives of This Tutorial

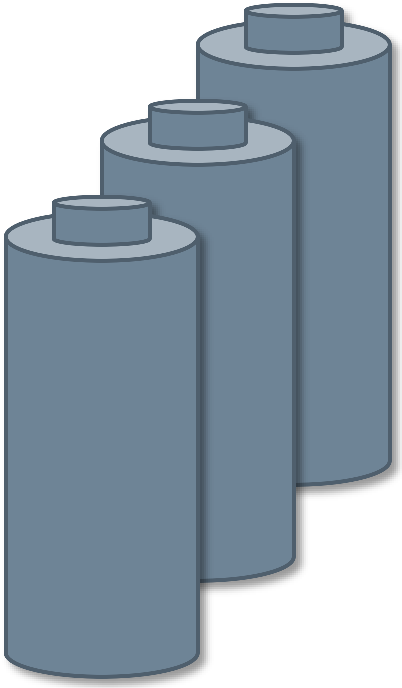
- What is meant by **model-based prognostics** and why it is a preferred approach
- The kinds of models needed and the tradeoffs involved
- Formal mathematical framework for model-based prognostics
- What are the constituent problems and how do we solve them

Scope of This Tutorial

- The focus here is on defining the model-based prognostics problem in a general way, with the most recent perspective
  - Formal/mathematical problem definition
  - Building models
  - Algorithms for solving the constituent problems
- For other material, see prognosis tutorials from previous PHM conferences
  - Requirements
  - Verification and validation
  - Performance metrics
  - Maintenance and logistics view
  - Other perspectives on prognostics

Running Example: Batteries

- Batteries are ubiquitous – laptops, mobile phones, electric cars, electric aircraft, etc.
- They will be used as a running example throughout this tutorial in various contexts
  - Cell prognostics
  - Battery prognostics
  - Power system prognostics
  - Vehicle system prognostics



Outline

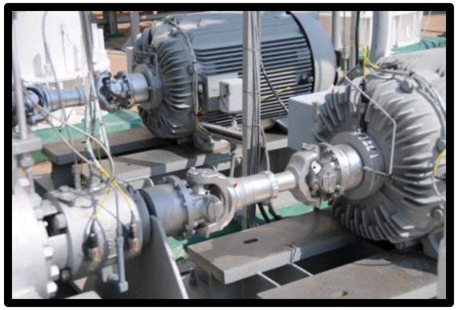
- Preliminaries
- Fundamentals
- Modeling
- Estimation
- Prediction
- Distributed Prognostics
- Putting It All Together
- Conclusions

Preliminaries

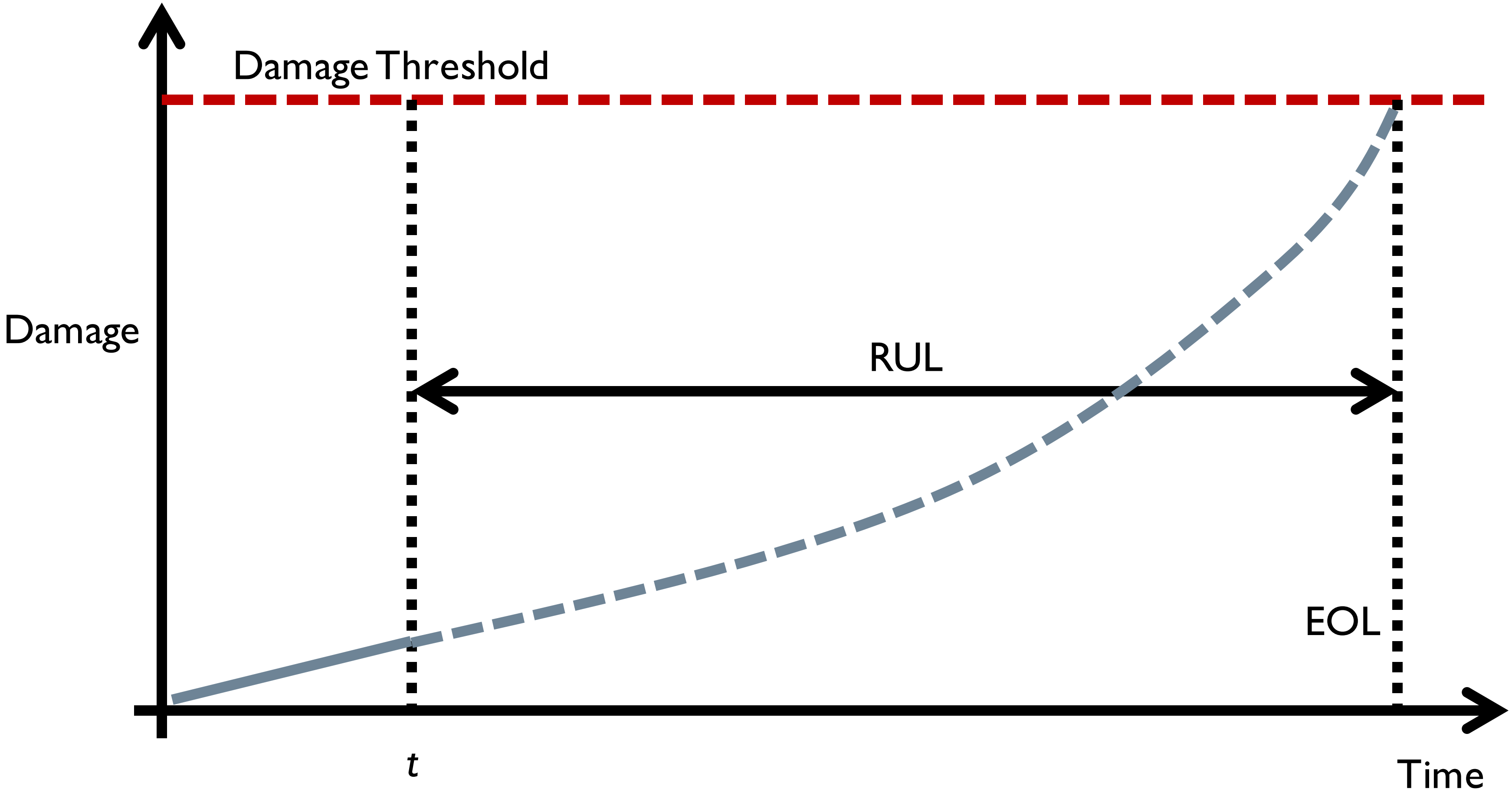
What is prognostics?
Why prognostics?
What is model-based prognostics?
Why model-based prognostics?

What is Prognostics?

- Prognosis = A forecast of the future course, or outcome, of a situation; a prediction
- We are more familiar with prognosis in a health management context:
  - Prediction of end of life (EOL) and/or remaining useful life (RUL)
  - EOL refers to a failure of the component as defined by its functional specifications



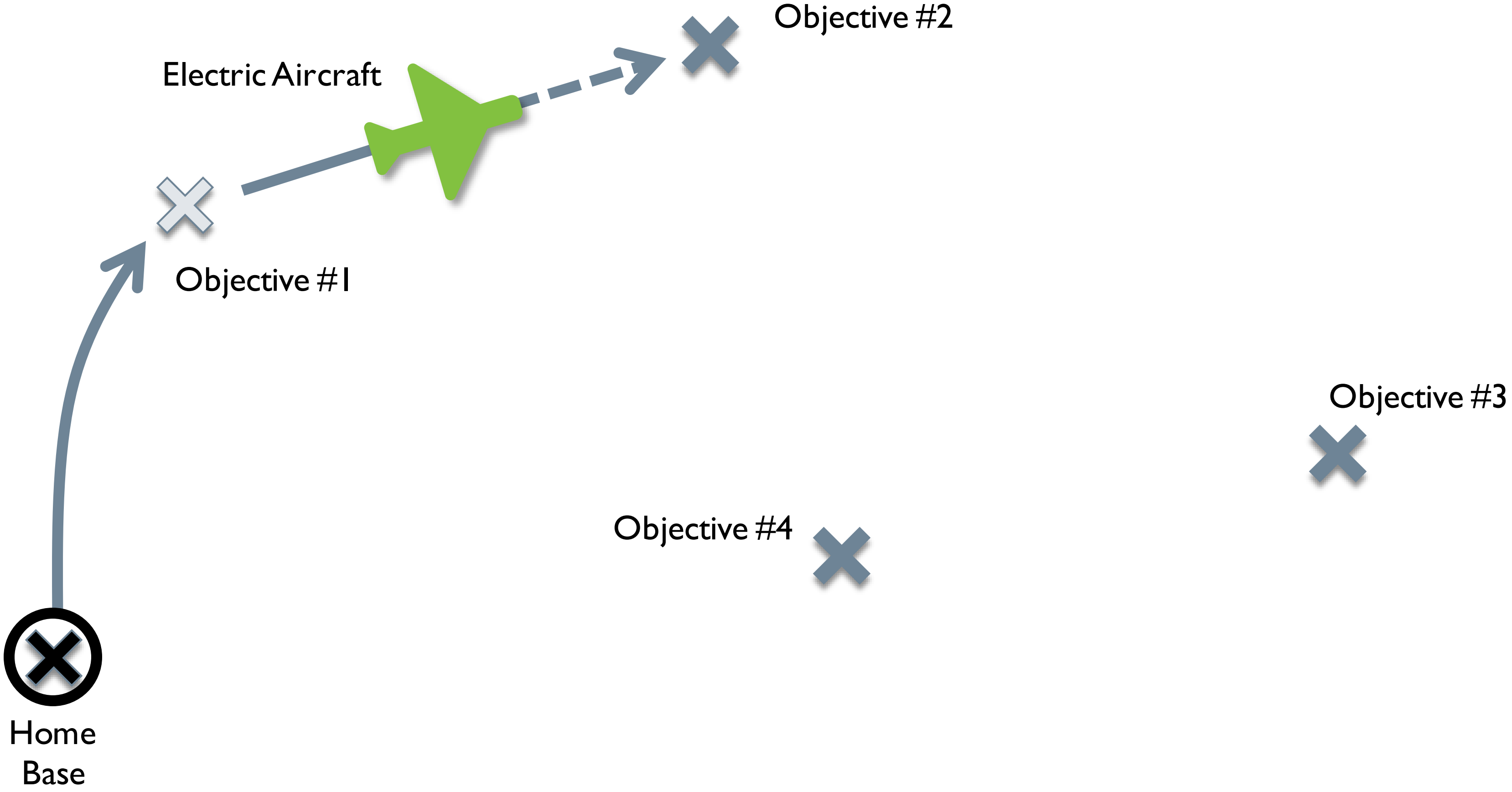
The Basic Idea



Why Prognostics?

Example: UAV Mission

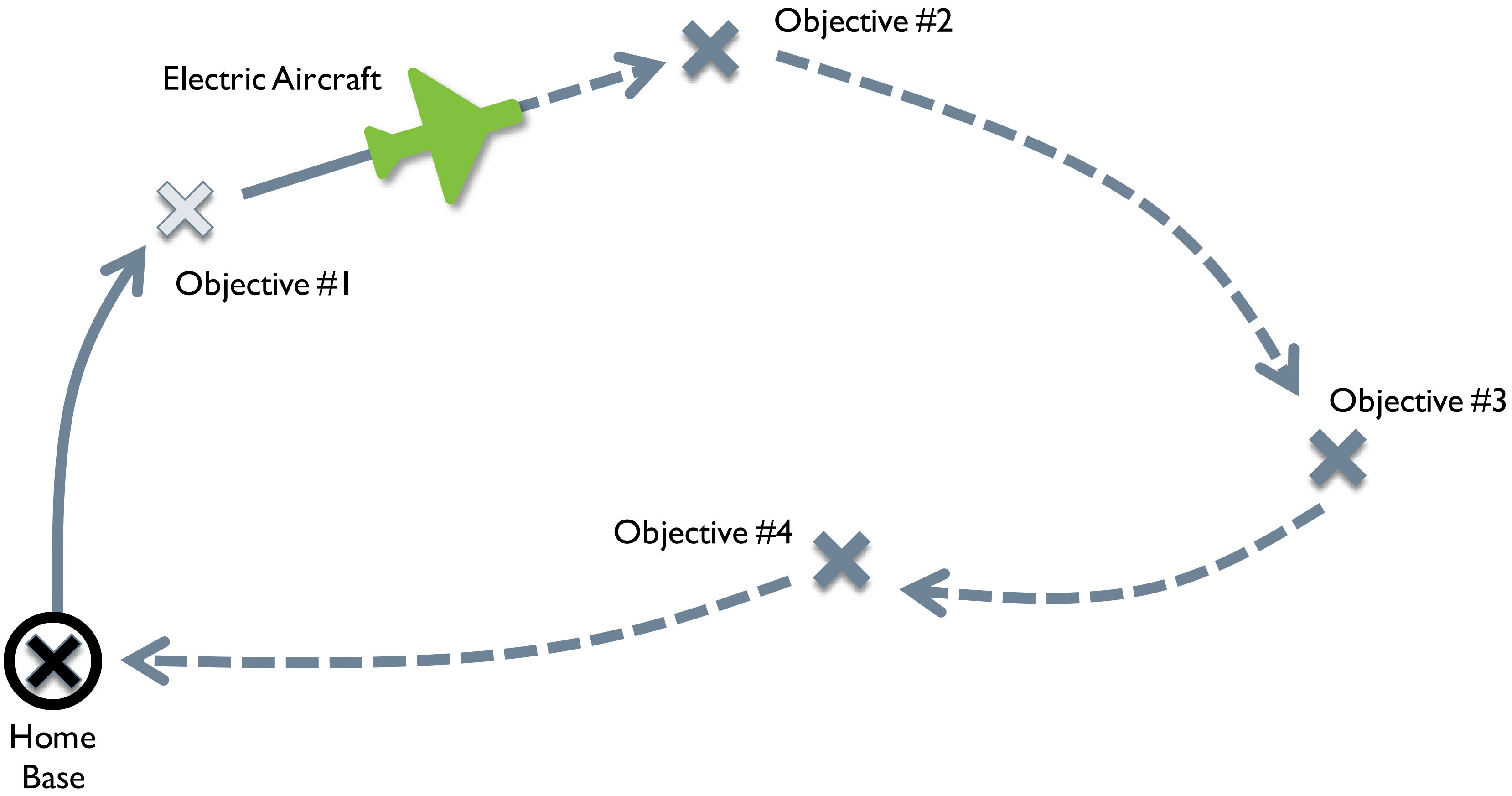
Visit waypoints to accomplish science objectives. Predict aircraft battery end of discharge to determine which objectives can be met. Based on prediction, plan optimal route. Replan if prediction changes.



Why Prognostics?

Example: UAV Mission

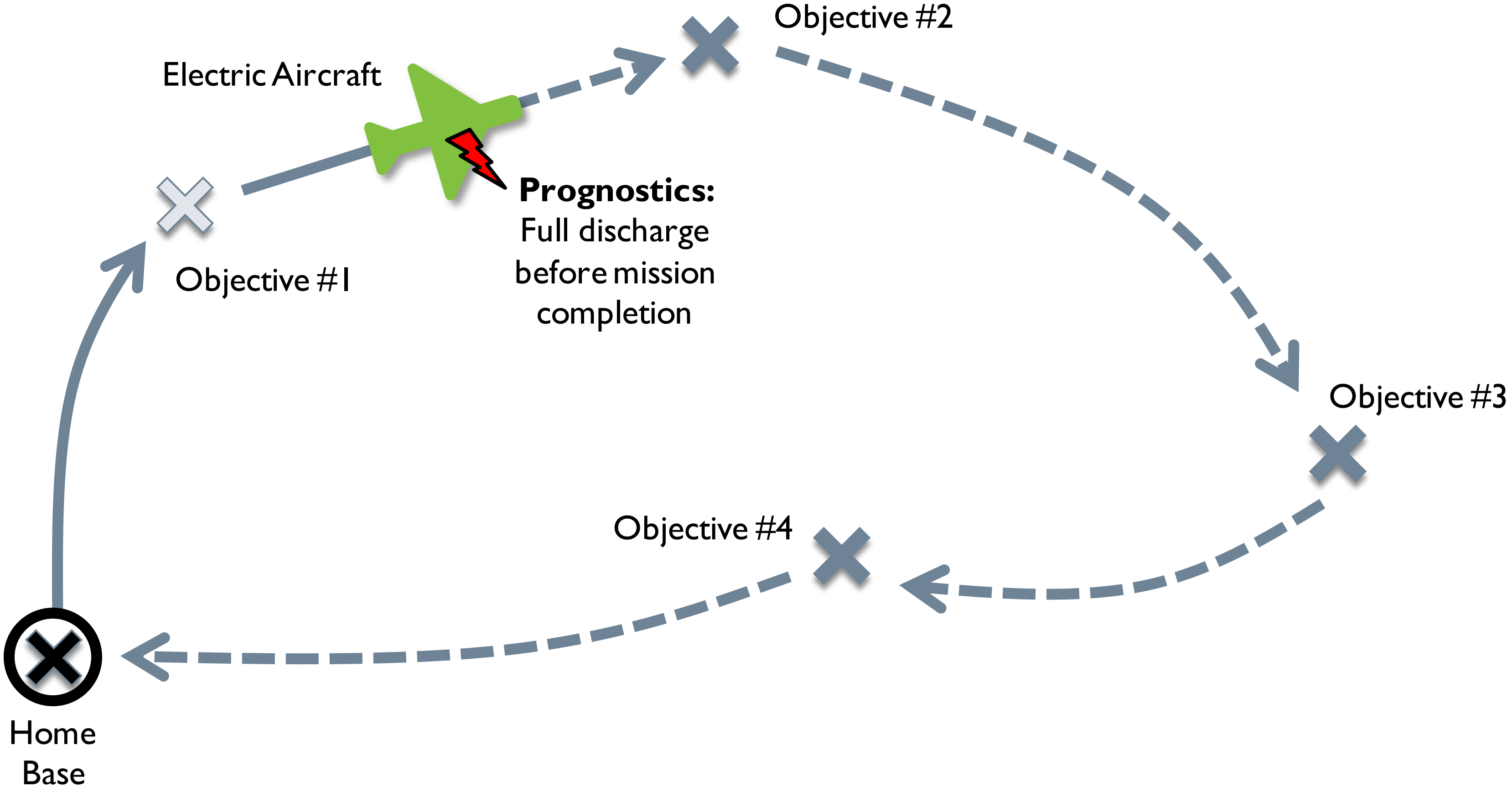
Visit waypoints to accomplish science objectives. Predict aircraft battery end of discharge to determine which objectives can be met. Based on prediction, plan optimal route. Replan if prediction changes.



Why Prognostics?

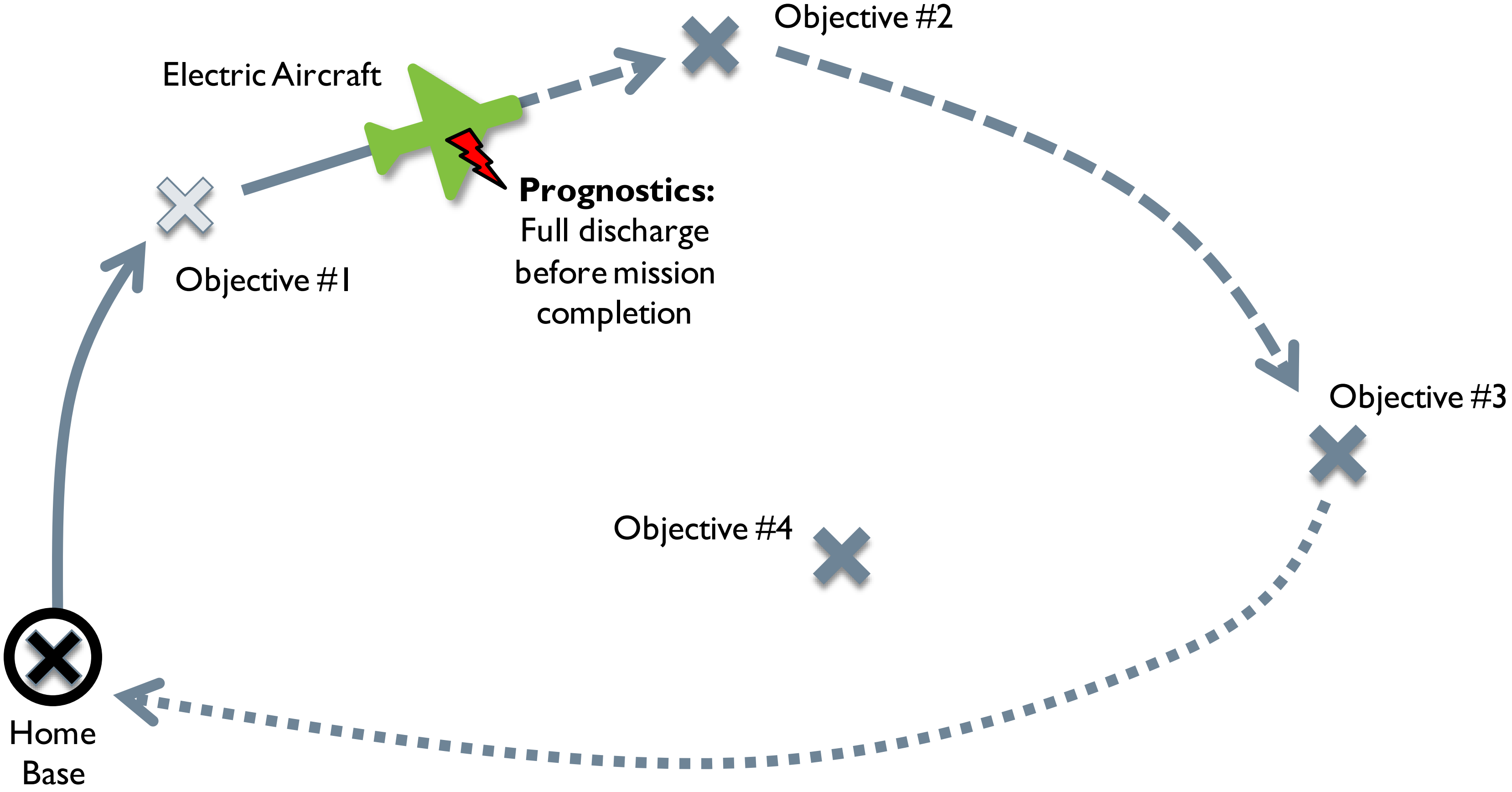
Example: UAV Mission

Visit waypoints to accomplish science objectives. Predict aircraft battery end of discharge to determine which objectives can be met. Based on prediction, plan optimal route. Replan if prediction changes.



Why Prognostics?

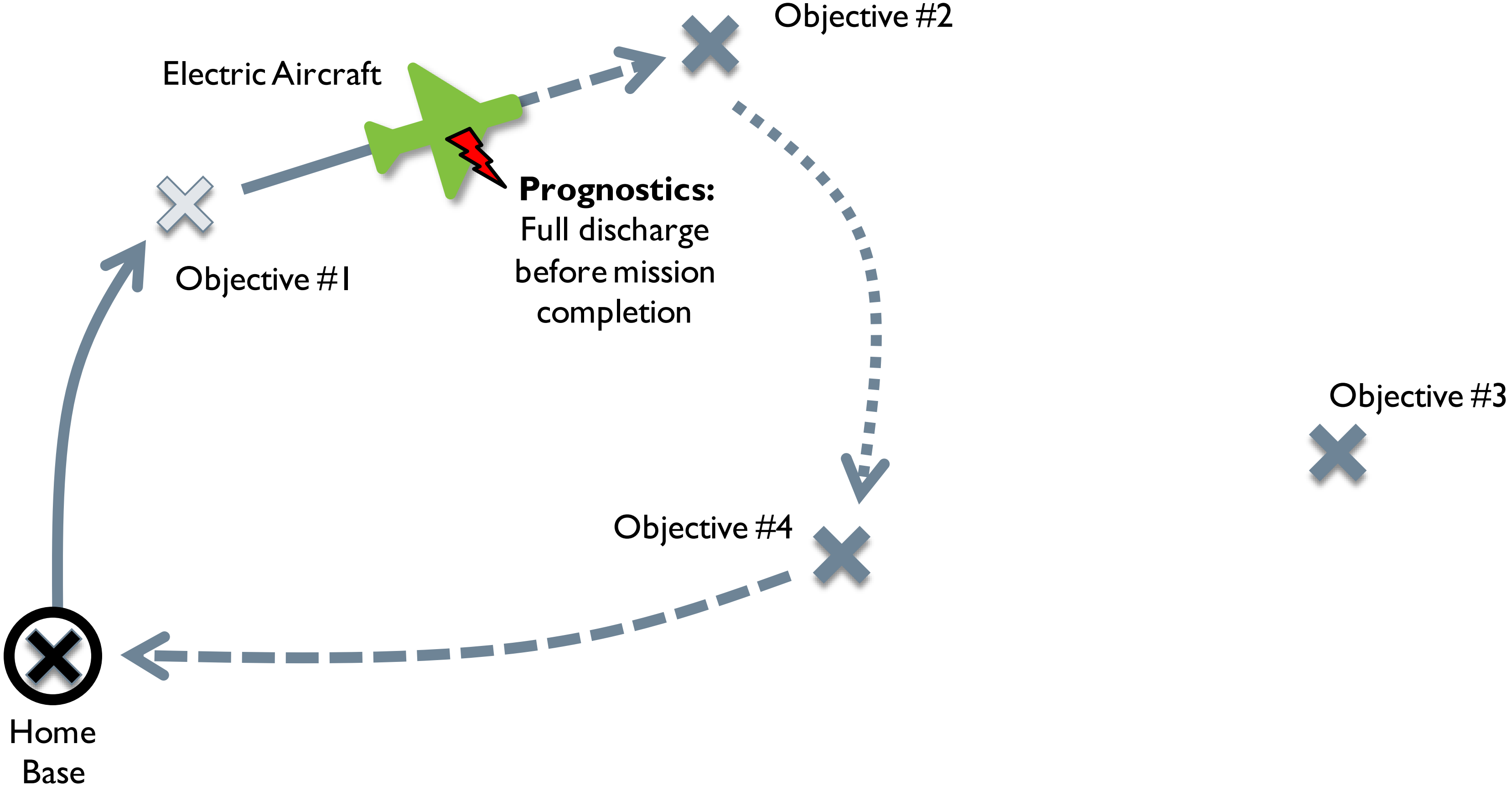
Example: UAV Mission
Visit waypoints to accomplish science objectives. Predict aircraft battery end of discharge to determine which objectives can be met. Based on prediction, plan optimal route. Replan if prediction changes.



Why Prognostics?

Example: UAV Mission

Visit waypoints to accomplish science objectives. Predict aircraft battery end of discharge to determine which objectives can be met. Based on prediction, plan optimal route. Replan if prediction changes.

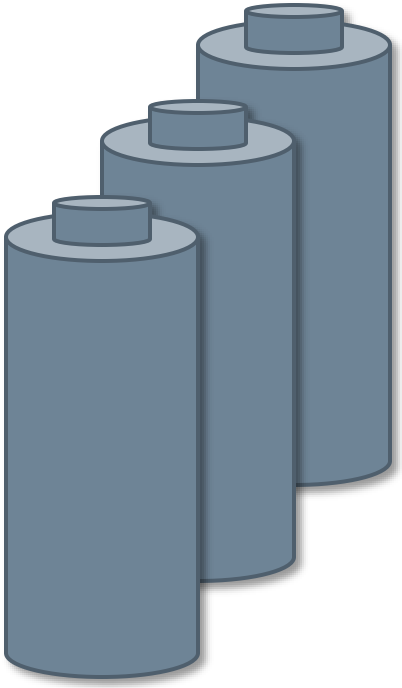


Why Prognostics?

- Prognostics can enable:
  - Adopting condition-based maintenance strategies, instead of time-based maintenance
  - Optimally scheduling maintenance
  - Optimally planning for spare components
  - Reconfiguring the system to avoid using the component before it fails
  - Prolonging component life by modifying how the component is used (e.g., load shedding)
  - Optimally plan or replan a mission
- System operations can be optimized in a variety of ways

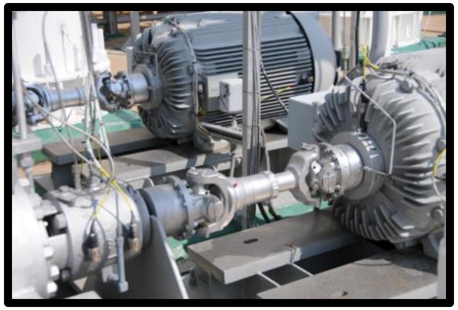
Why Battery Prognostics?

- Countless systems use batteries
- Prognostics can be used to
  - – Predict end of discharge
    - how long device/system can be used
    - when to charge
  - – Predict end of usable capacity
    - when to replace the battery
- In the context of a system like an electric vehicle, battery prognostics informs you how to use the vehicle in an optimal fashion



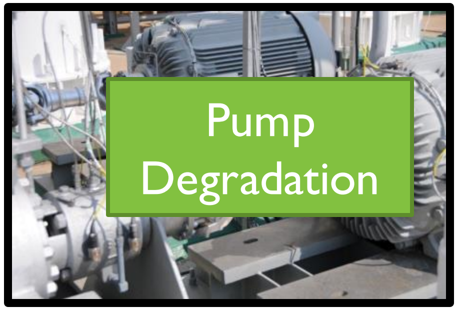
A More General Definition

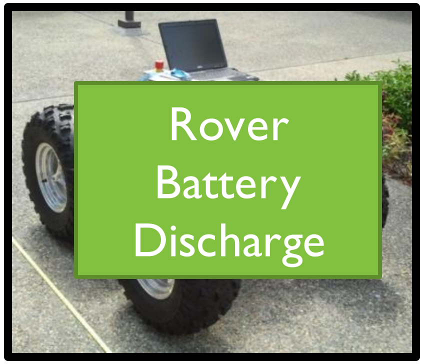
- Prognosis = A prediction of the occurrence of some event of interest to the system
- This event could be
  - Component failure
  - Violation of functional or performance specifications
  - Accomplishment of some system function
  - End of a mission
  - … anything of importance you want to predict, because that knowledge is useful to a decision
- What this event represents does not matter to the framework



A More General Definition

- Prognosis = A prediction of the occurrence of some event of interest to the system
- This event could be
  - Component failure
  - Violation of functional or performance specifications
  - Accomplishment of some system function
  - End of a mission
  - … anything of importance you want to predict, because that knowledge is useful to a decision
- What this event represents does not matter to the framework

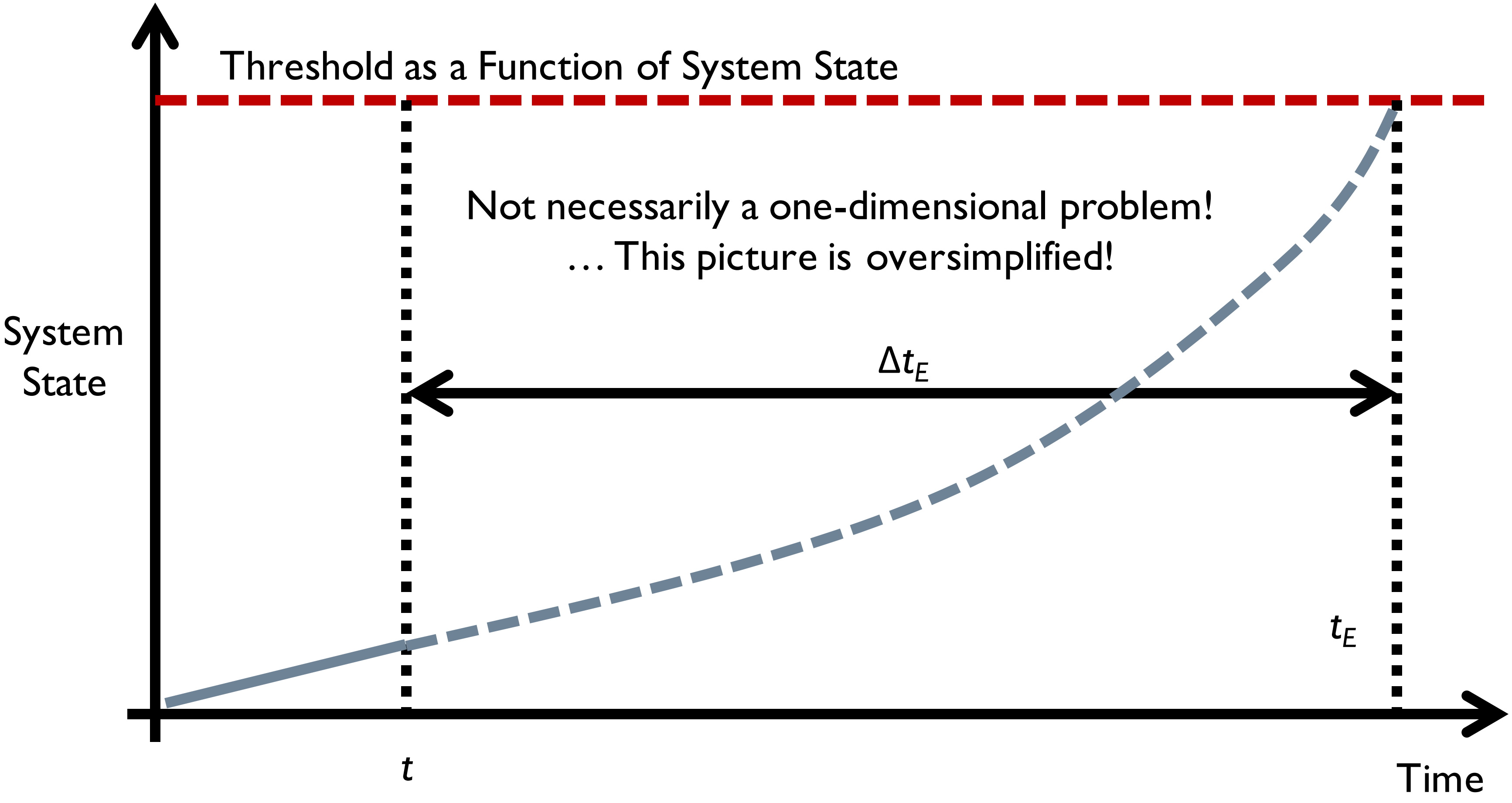

Pump Degradation


Rover Battery Discharge

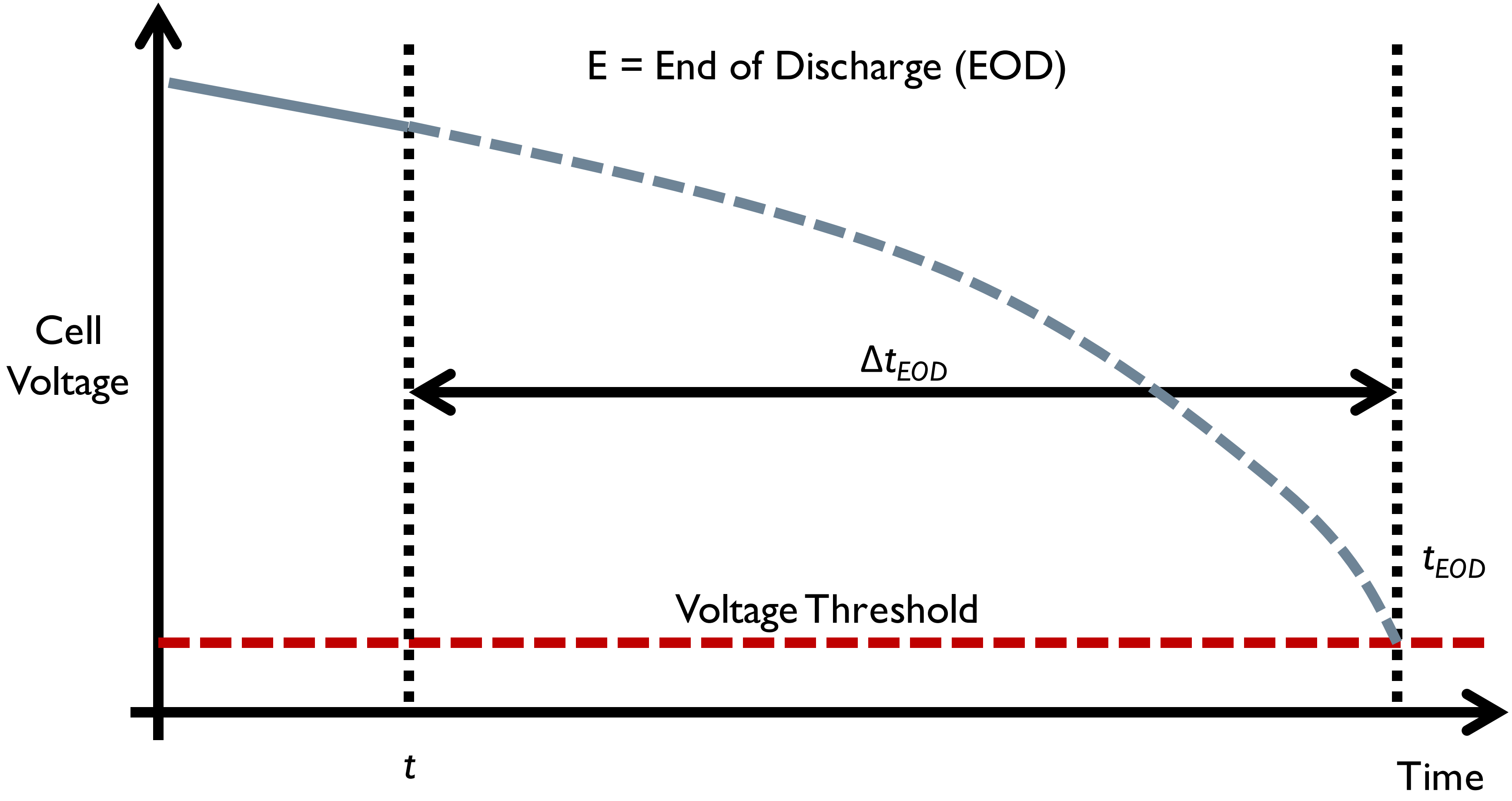
Completion of Fueling

End of Flight

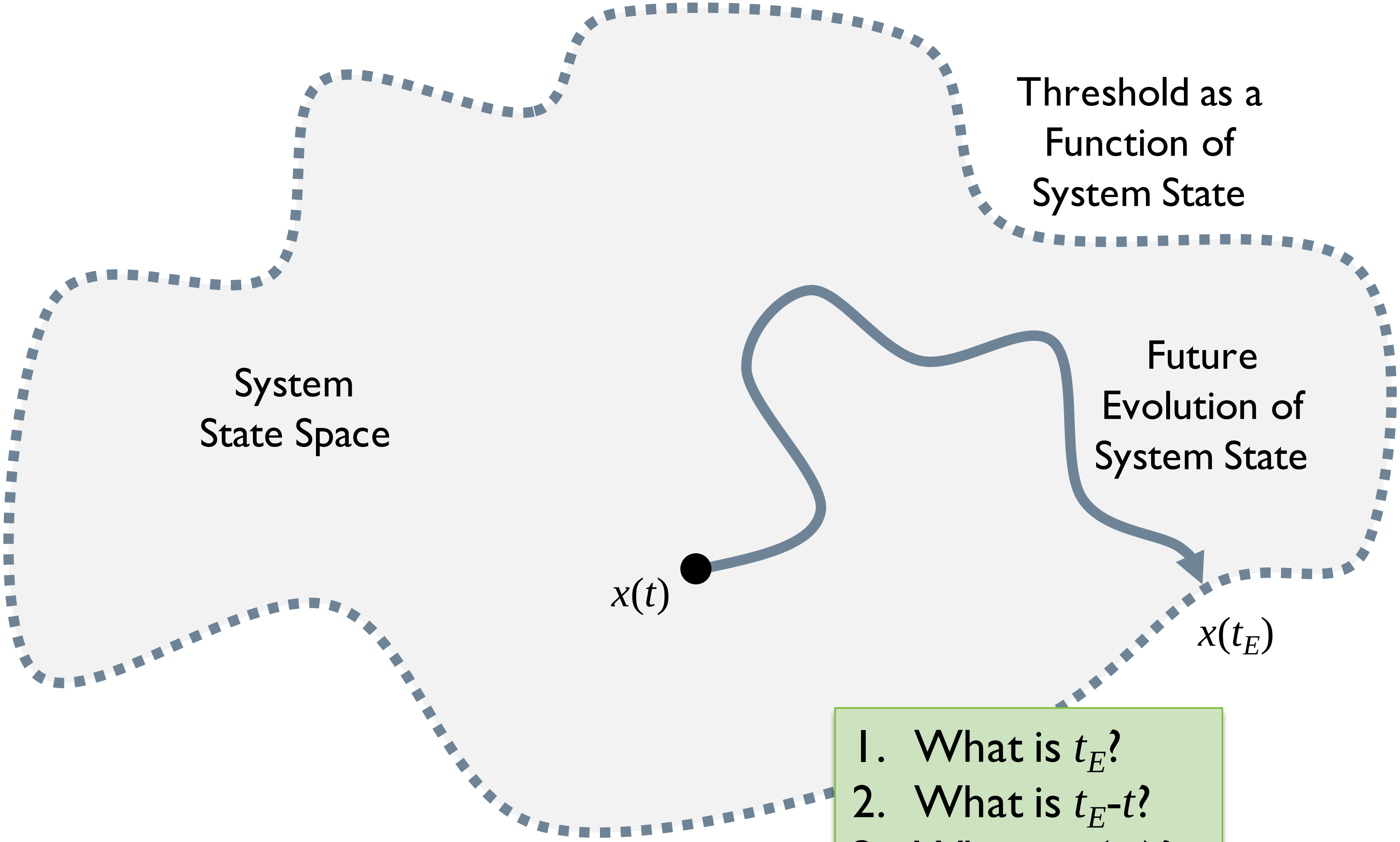
The Basic Idea Revisited


Threshold as a Function of System State
Not necessarily a one-dimensional problem!
… This picture is oversimplified!
System State
Δt_E
t_E
t
Time

The Basic Idea Revisited: Batteries



The Basic Idea Revisited


Threshold as a Function of System State
System State Space
Future Evolution of System State
$x(t)$
$x(t_E)$

1. What is t_E?
2. What is t_E-t?
3. What is $x(t_E)$?

What is Model-Based Prognostics?

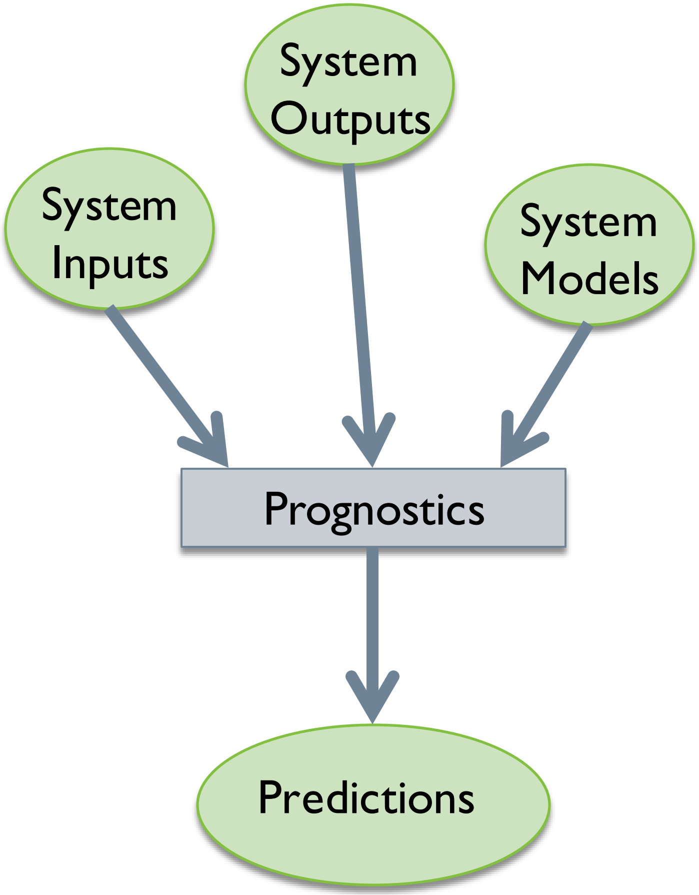
- “Model-based” vs “data-driven”
  - “Model-based” typically refers to approaches using models derived from first principles (e.g., physics-based)
  - “Data-driven” typically refers to approaches using models learned from data (e.g., NNs, GPR)
- These terms are not very useful!
  - All approaches use models of some kind, and all are driven by data
  - In practice, models are typically developed from a mix of system knowledge and system data and are typically adapted online in some fashion

Our Definition

- Model-based prognostics refers simply to approaches that use mathematical models of system behavior
  - When available, knowledge from first principles, known physical laws, etc, should be used to develop models
  - When a large amount of data is available (for both nominal and degraded behavior), models can be learned from the data
- The general framework will be defined in this context
  - It does not matter how the model was developed
  - It does not matter what the model looks like

Why Model-Based Prognostics?

- With model-based algorithms, models are **inputs**
  - This means that, given a new problem, we use the same general algorithms
  - Only the models should change
- Model-based prognostics approaches are applicable to a large class of systems, given a model
- Approach can be formulated mathematically, clearly and precisely



Fundamentals

> How do we formulate the problem?
> Where does uncertainty come from?
> What are the constituent problems?
> What is the computational architecture?

Problem Formulation

- System described by

$$\begin{aligned}\mathbf{x}(k+1) &= \mathbf{f}(k, \mathbf{x}(k), \boldsymbol{\theta}(k), \mathbf{u}(k), \mathbf{v}(k))\\ \mathbf{y}(k) &= \mathbf{h}(k, \mathbf{x}(k), \boldsymbol{\theta}(k), \mathbf{u}(k), \mathbf{n}(k))\end{aligned}$$

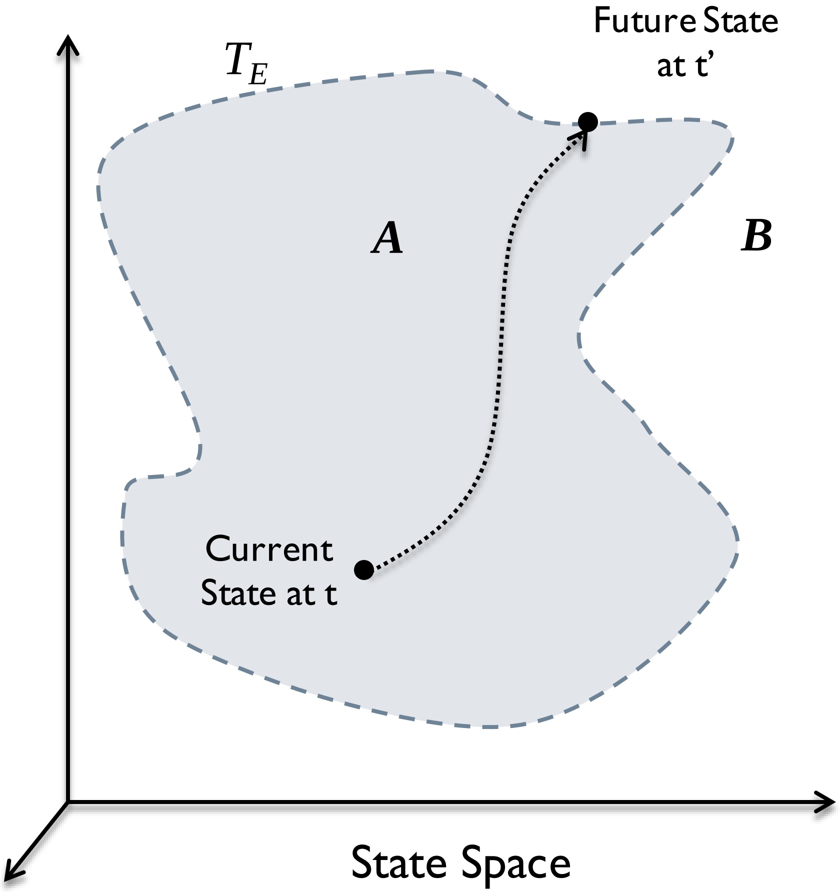
  – $\mathbf{x}$: states, $\boldsymbol{\theta}$: parameters, $\mathbf{u}$: inputs, $\mathbf{y}$: outputs, $\mathbf{v}$: process noise, $\mathbf{n}$: sensor noise

- Define system event of interest E
- Define threshold function, that evaluates to true when E has occurred

$$T_E(\mathbf{x}(k), \boldsymbol{\theta}(k), \mathbf{u}(k))$$

Problem Formulation

- Interested in predicting E
  - E.g., battery voltage falls below cutoff voltage to define end-of-discharge
- System starts at some state in region $\boldsymbol{A}$, eventually evolves to some new state at which E occurs and moves to region $\boldsymbol{B}$
- T_E defines the boundary between $\boldsymbol{A}$ and $\boldsymbol{B}$
- Must predict the time of event E, k_E, and the time until event E, Δk_E



Problem Formulation

- Define k_E

$$k_E(k_P) \triangleq \inf\{k \in \mathbb{N} \colon k \geq k_P \wedge T_E(\mathbf{x}(k), \boldsymbol{\theta}(k), \mathbf{u}(k)) = 1\}$$

- Define Δk_E

$$\Delta k_E(k_P) \triangleq k_E(k_P) - k_P$$

- May also be interested in the values of some system variables at k_E

$$\mathbf{z}(k) = \boldsymbol{\psi}(k, \mathbf{x}(k), \boldsymbol{\theta}(k), \mathbf{u}(k))$$

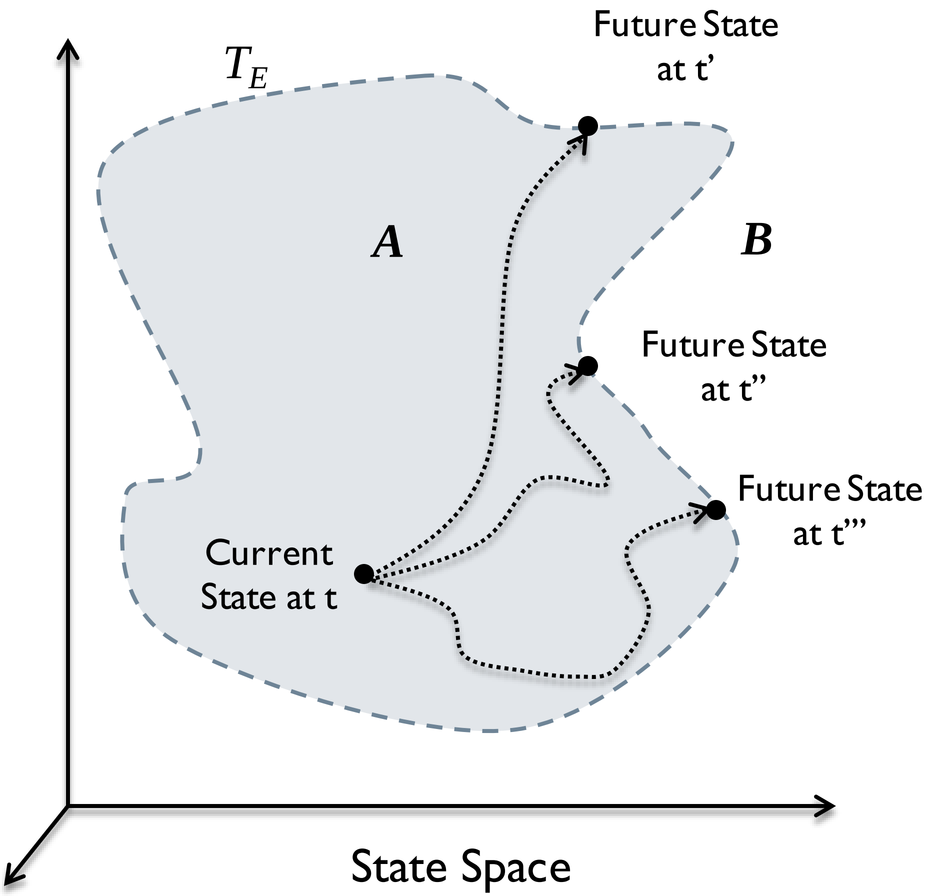
$$\mathbf{z}_E(k_P) \triangleq \mathbf{z}(k_E(k_P))$$

$$\Delta \mathbf{z}_E(k_P) = \mathbf{z}_E(k_P) - \mathbf{z}(k_P)$$

- **Goal** is to compute k_E and its derived variables

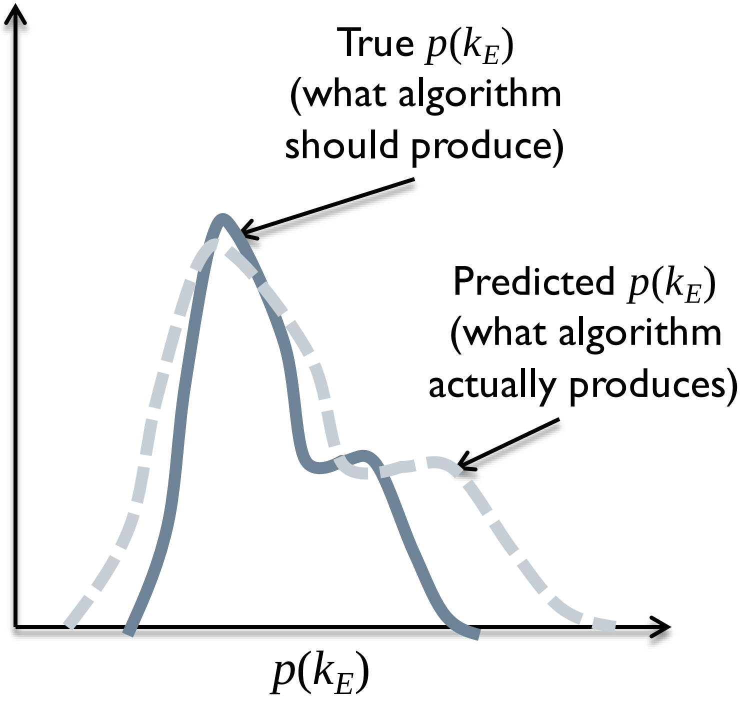
Uncertainty

- There is uncertainty inherent to the system
- System actually takes one path *out of many possible paths* to region $\boldsymbol{B}$
  - System dynamics are stochastic (modeled as process noise)
  - Future system inputs are stochastic (many possible future usage profiles, system disturbances)
- So, k_E is a random variable, and we must predict its *probability distribution*



Uncertainty

- Goal of prognostics algorithm is to predict true distribution of k_E
  - A misrepresentation of true uncertainty could be disastrous when used for decision-making
- Prognostics algorithm itself adds additional uncertainty
  - Initial state not known exactly
  - Sensor and process noise (stochastic processes with unknown distributions)
  - Model not known exactly
  - System state at k_P not known exactly
  - Future input trajectory distribution not known exactly

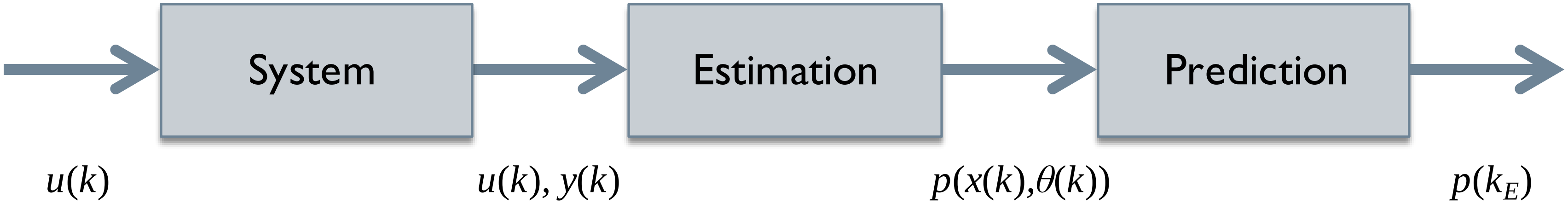


Uncertainty **added** by algorithm should be minimized

Constituent Problems

- In order to compute k_E, we need to know
  - What is the system state at k_P?
  - What potential inputs will the system have from k_P to k_E?
  - What model describes the system evolution?
  - What is the process noise distribution?
  - What is the future input trajectory distribution?
- Prognostics is often split into two sequential problems
  - Estimation: determining the system state at k_P
  - Prediction: determining k_E

Prognostics Architecture



- System gets input and produces output
- Estimation module estimates the states and parameters, given system inputs and outputs
  - Must handle sensor noise
  - Must handle process noise
- Prediction module predicts k_E
  - Must handle state-parameter uncertainty at k_P
  - Must handle future process noise trajectories
  - Must handle future input trajectories
  - A diagnosis module can inform the prognostics what model to use

Two Kinds of Problems

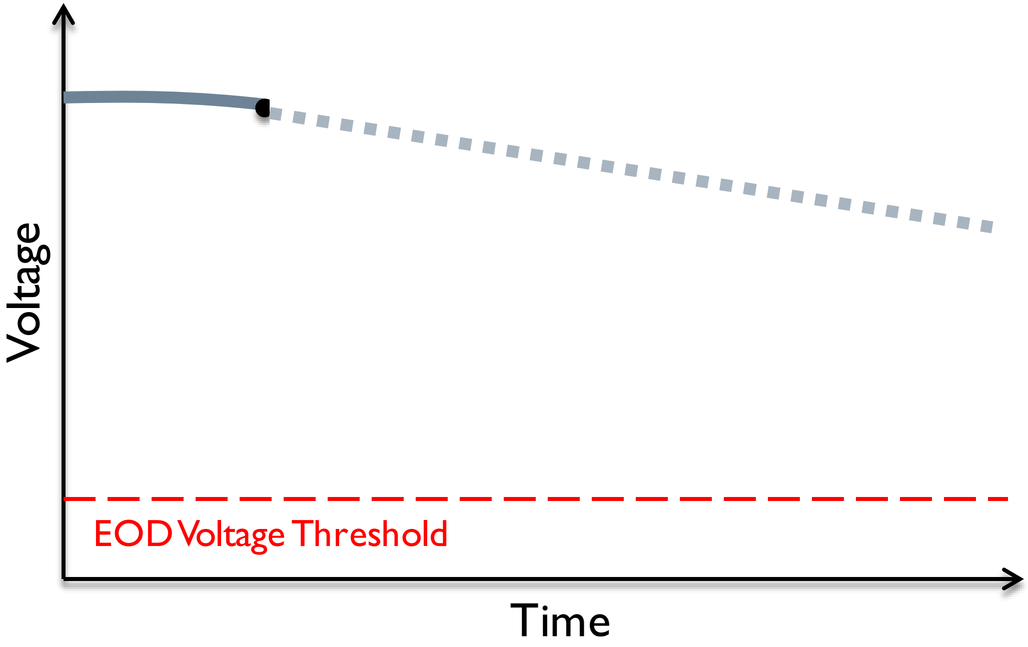
1. Modeling problems
   - Dynamic system model
   - Process noise model
   - Sensor noise model
   - Future input model
2. Algorithm problems
   - Estimating system state at t
   - Estimating uncertainty in system state
   - Predicting E
   - Predicting uncertainty in E

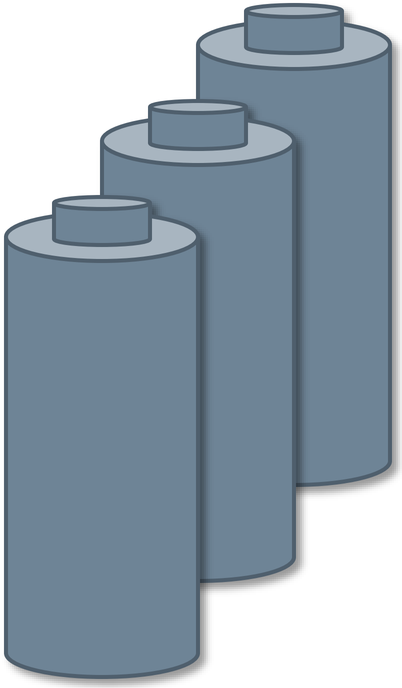
Modeling

What needs to be modeled?
What features do models need?
What are the modeling trade-offs?

Example: Batteries

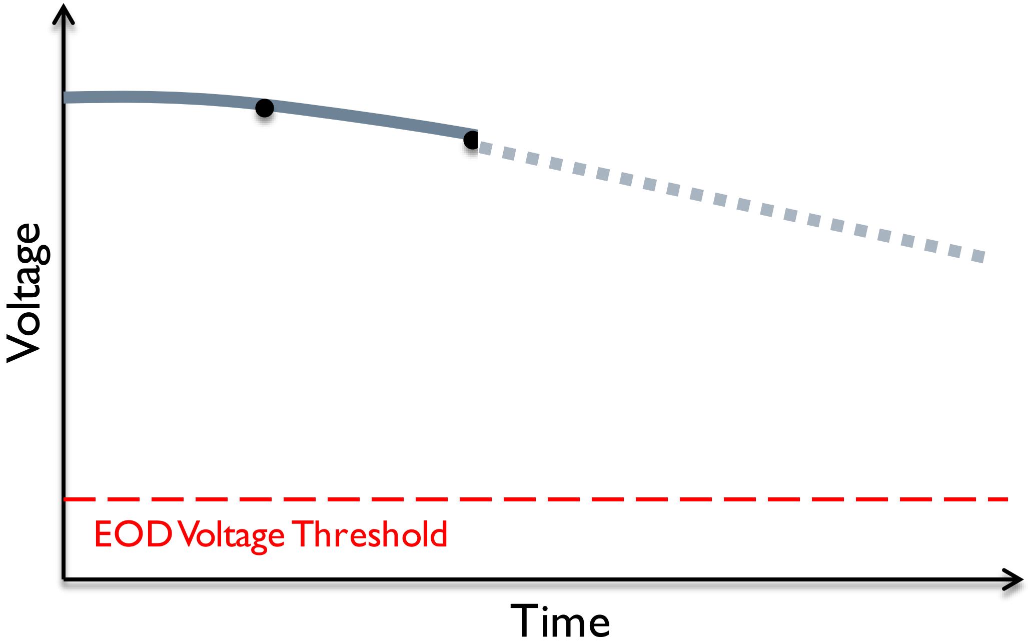
Predict end of discharge, defined by a voltage threshold.
Assume a prediction model: $V(k) = V_0 - m\, k$.
Estimate m at each time of prediction.

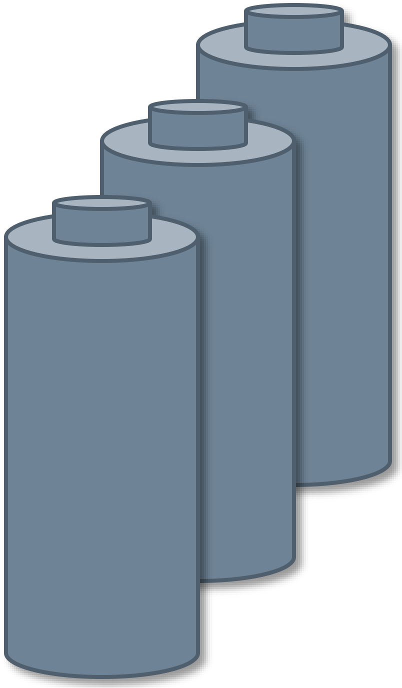




Example: Batteries

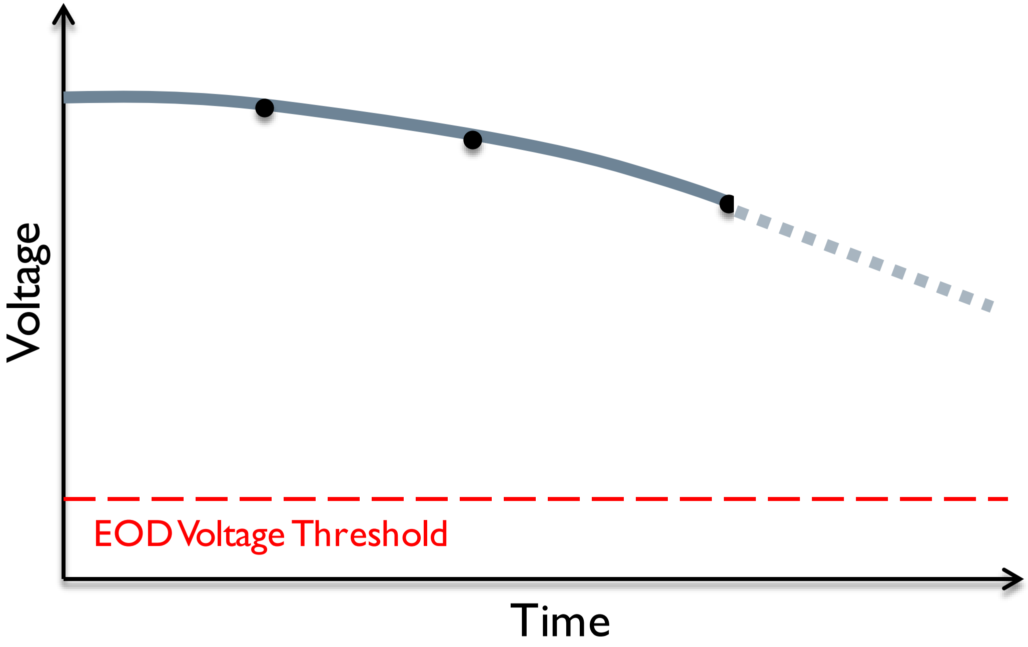
Predict end of discharge, defined by a voltage threshold.
Assume a prediction model: $V(k) = V_0 - m\ k$.
Estimate m at each time of prediction.

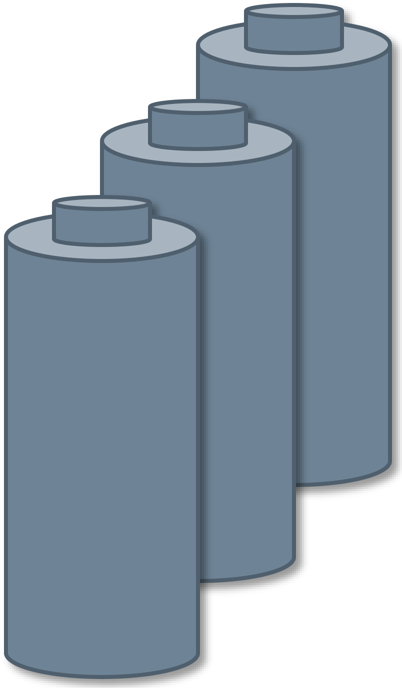




Example: Batteries

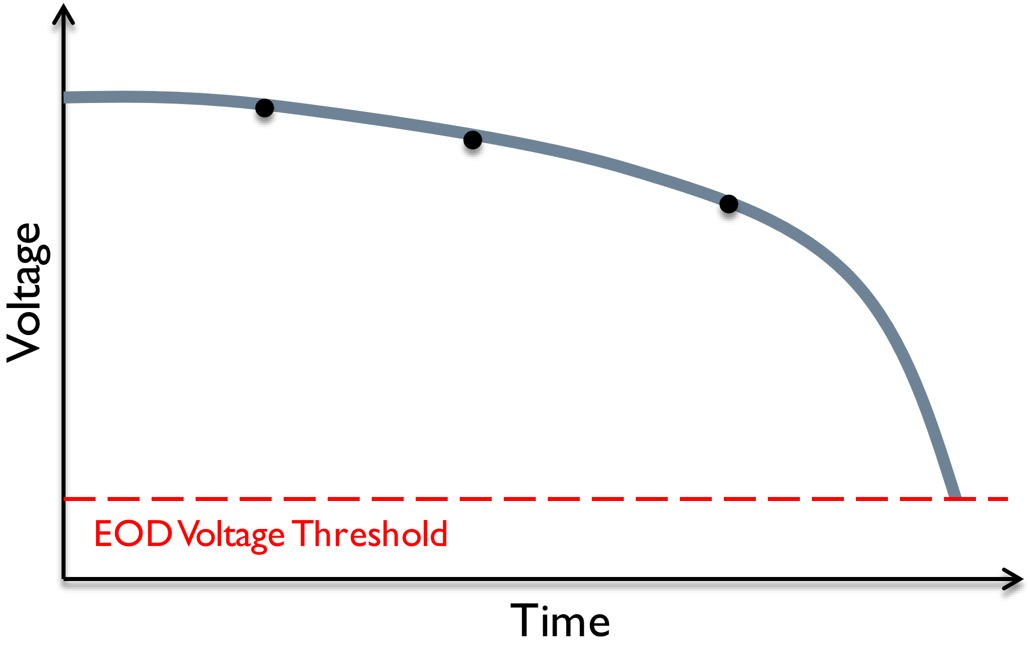
Predict end of discharge, defined by a voltage threshold.
Assume a prediction model: $V(k) = V_0 - m\,k$.
Estimate m at each time of prediction.

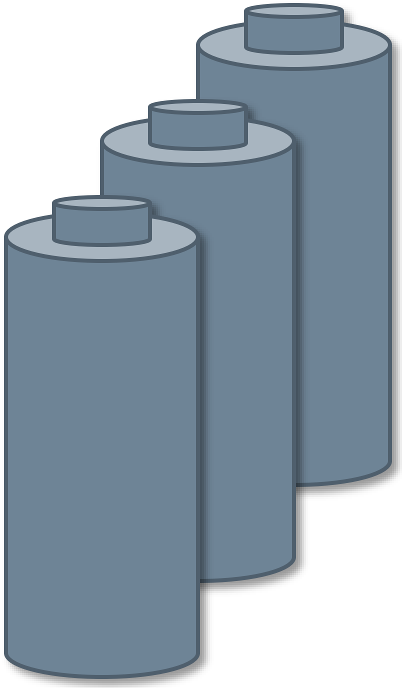




Example: Batteries

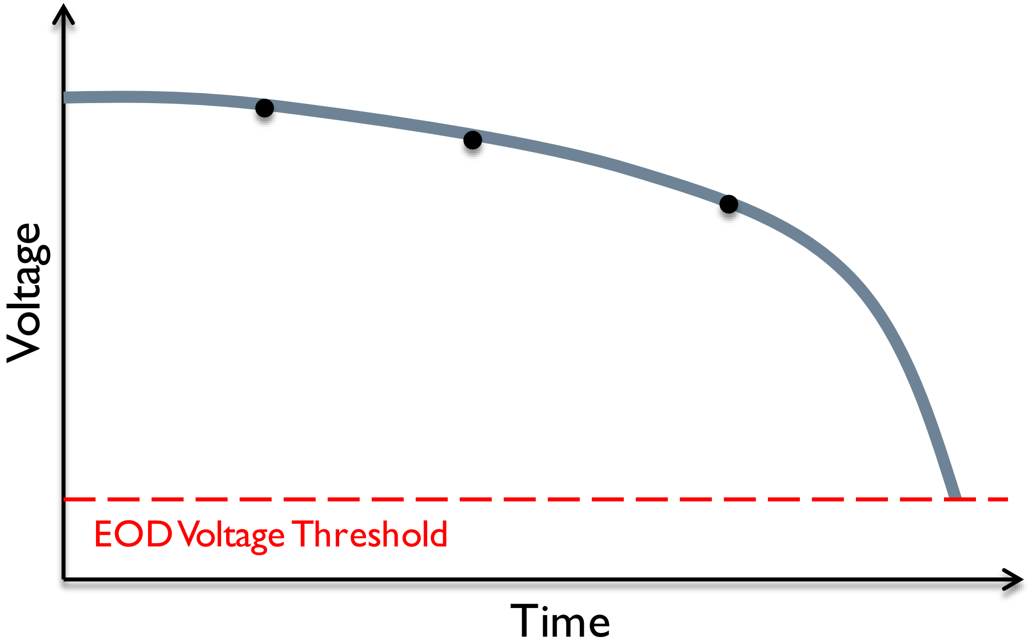
Predict end of discharge, defined by a voltage threshold.
Assume a prediction model: $V(k) = V_0 - m\ k$.
Estimate m at each time of prediction.

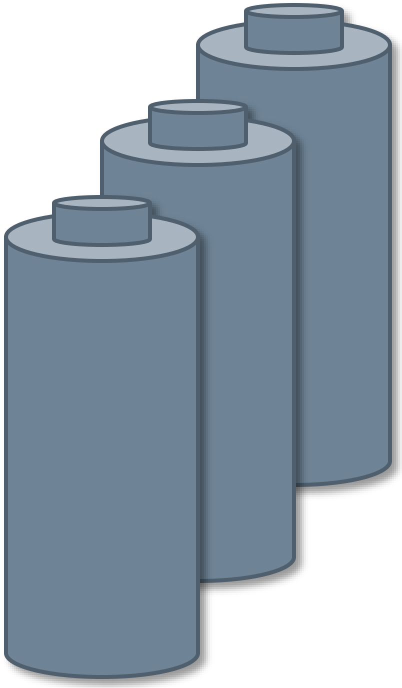




Example: Batteries

Predict end of discharge, defined by a voltage threshold.
Assume a prediction model: $V(k) = V_0 - m\,k$.
Estimate m at each time of prediction.

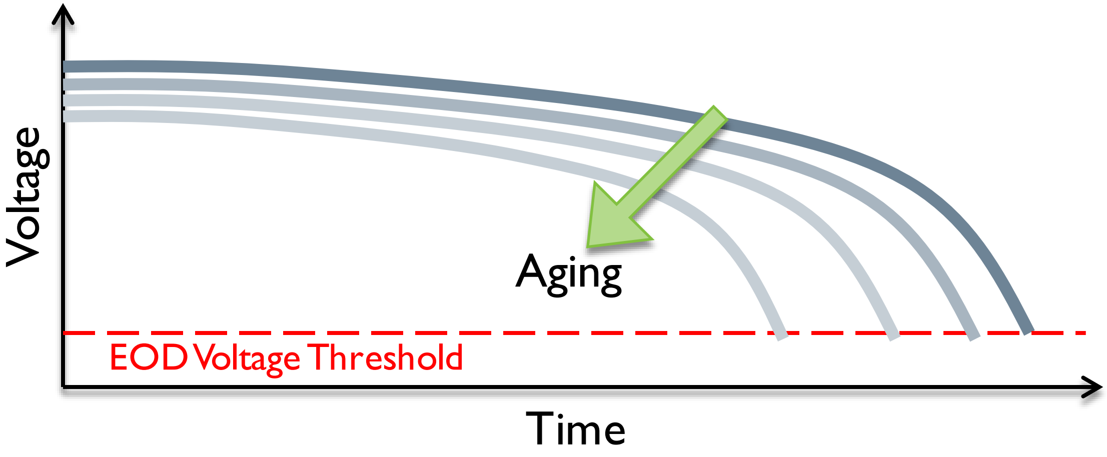




In order to obtain accurate predictions, we need to understand the system!

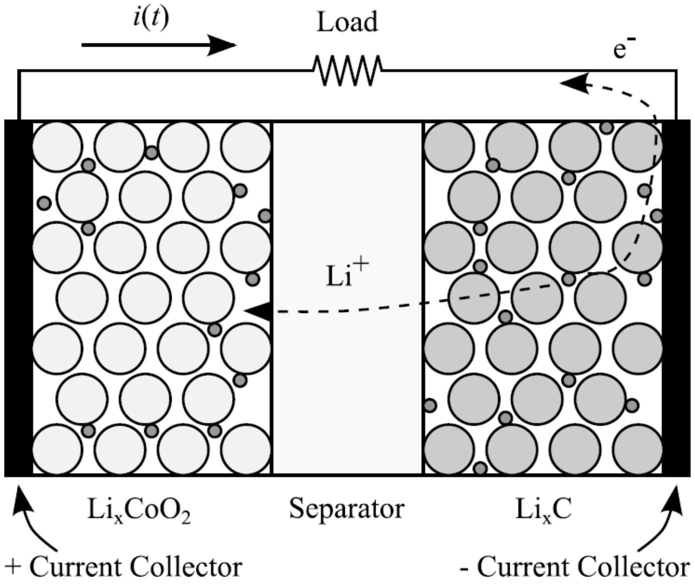
What Kind of Models?

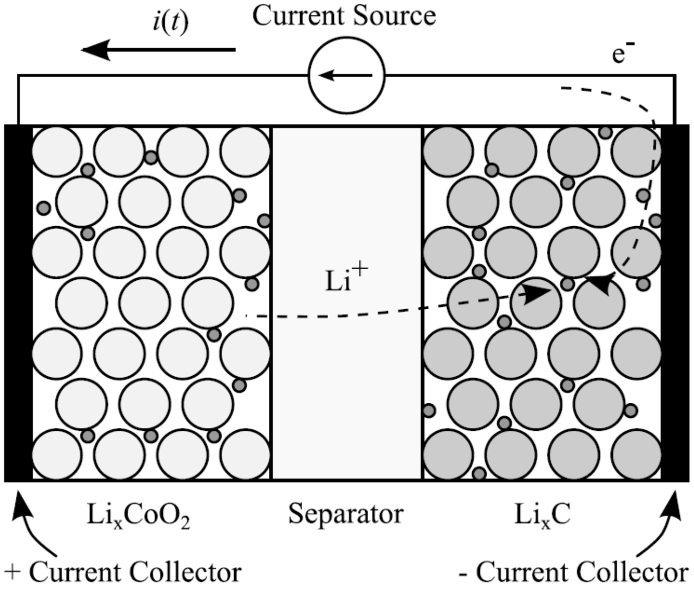
- Models for prognostics require the following features
  - Describe dynamics in nominal case (no aging/degradation)
  - Describe dynamics in the faulty/degraded/damaged case
  - Describe dynamics of aging/degradation



- What are the dynamics describing discharge?
- What model parameters change as a result of aging?
- How do the aging parameters change in time?

Example: Batteries





Discharge

Positive electrode is cathode

Negative electrode is anode

Reduction at pos. electrode:

$Li_{1-n}CoO_2 + nLi^+ + ne^- \rightarrow LiCoO_2$

Oxidation at neg. electrode:

$Li_nC \rightarrow nLi^+ + ne^- + C$

Current flows + to –

Electrons flow – to +

Lithium ions flow – to +

Charge

Positive electrode is anode

Negative electrode is cathode

Oxidation at pos. electrode:

$LiCoO_2 \rightarrow Li_{1-n}CoO_2 + nLi^+ + ne^-$

Reduction at neg. electrode:

$nLi^+ + ne^- + C \rightarrow Li_nC$

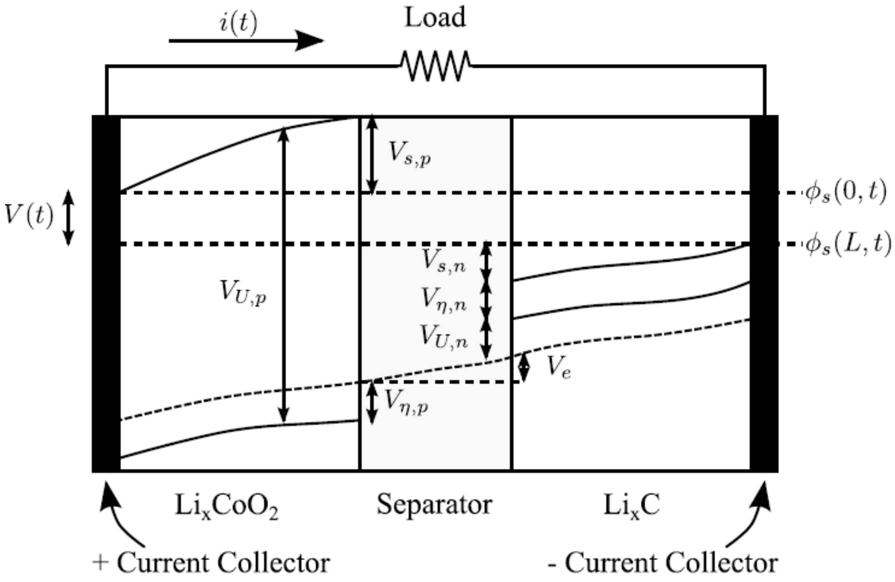
Current flows – to +

Electrons flow + to –

Lithium ions flow + to –

Example: Battery Modeling

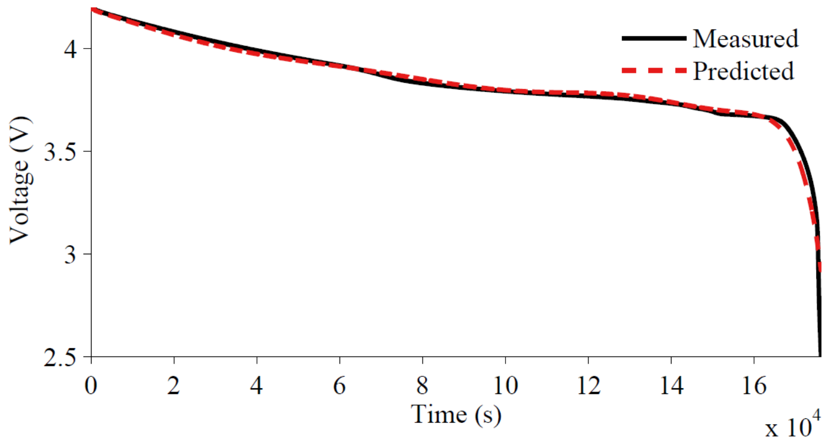
- Lumped-parameter, ordinary differential equations
- Capture voltage contributions from different sources
  - Equilibrium potential →Nernst equation with Redlich-Kister expansion
  - Concentration overpotential → split electrodes into surface and bulk control volumes
  - Surface overpotential → Butler-Volmer equation applied at surface layers
  - Ohmic overpotential → Constant lumped resistance accounting for current collector resistances, electrolyte resistance, solid-phase ohmic resistances
- T_E defined using a voltage cutoff
  - T_E is crossed once $V<V_{EOD}$



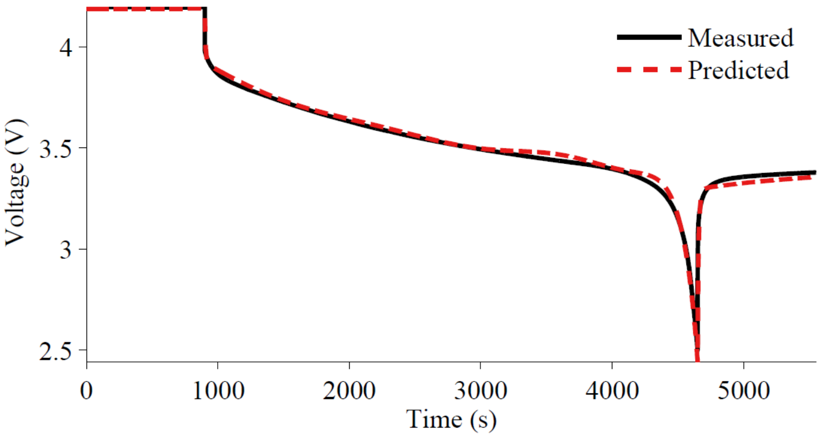
Example: Battery Modeling

- State vector
  - Lithium ions in positive electrode, surface
  - Lithium ions in positive electrode, bulk
  - Lithium ions in negative electrode, surface
  - Lithium ions in negative electrode, bulk
  - Ohmic drop voltage
  - Surface overpotential in negative electrode
  - Surface overpotential in positive electrode
  - Cell temperature
- Parameter vector (for end of capacity prediction)
  - Ohmic resistance
  - Maximum mobile lithium ions
- Input vector
  - Cell current
- Output vector
  - Cell voltage
  - Cell temperature

Battery Model Validation



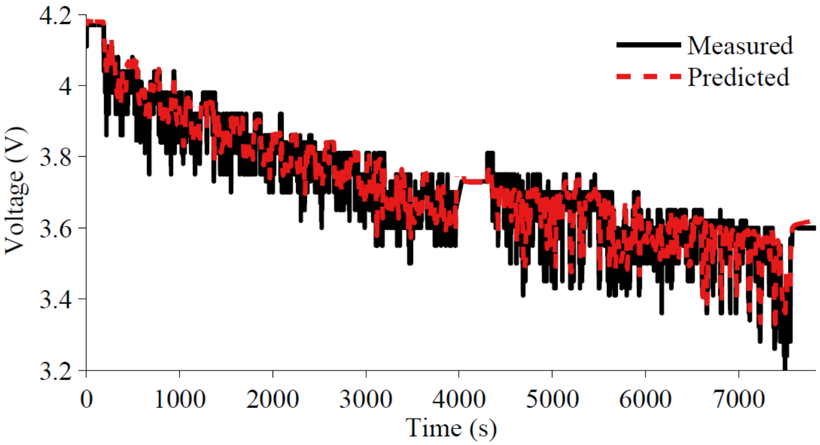
“Open-Circuit” Discharge Curve



Nominal 2A Discharge Curve

Model matches well for open-circuit test (0.04 A discharge) and nominal discharge (2 A) on battery test stand.

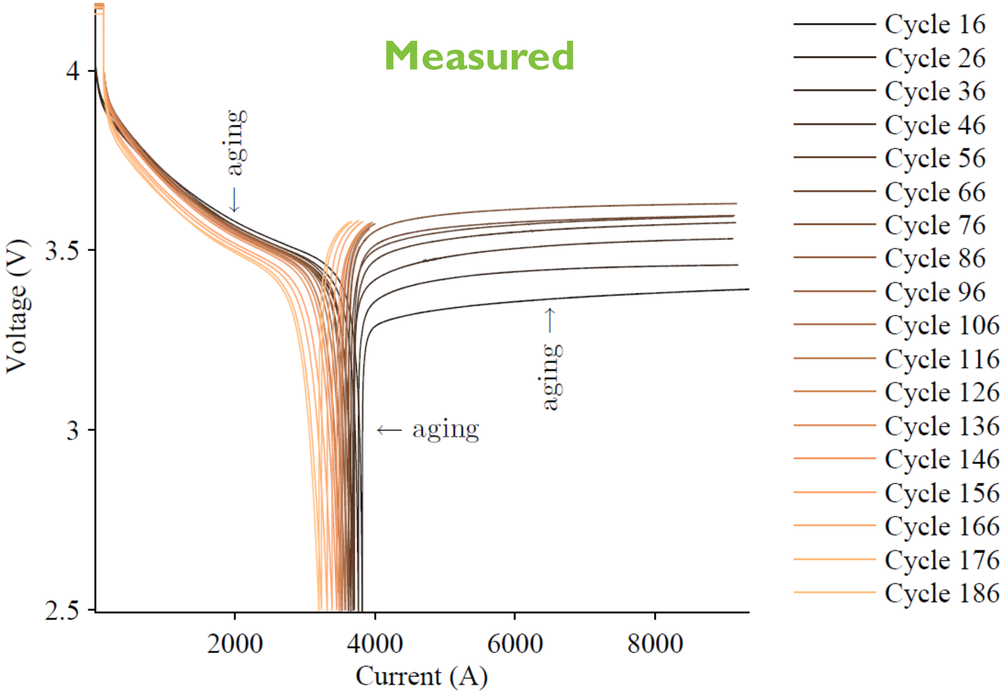
Model matches well for variable-load discharges on the rover.

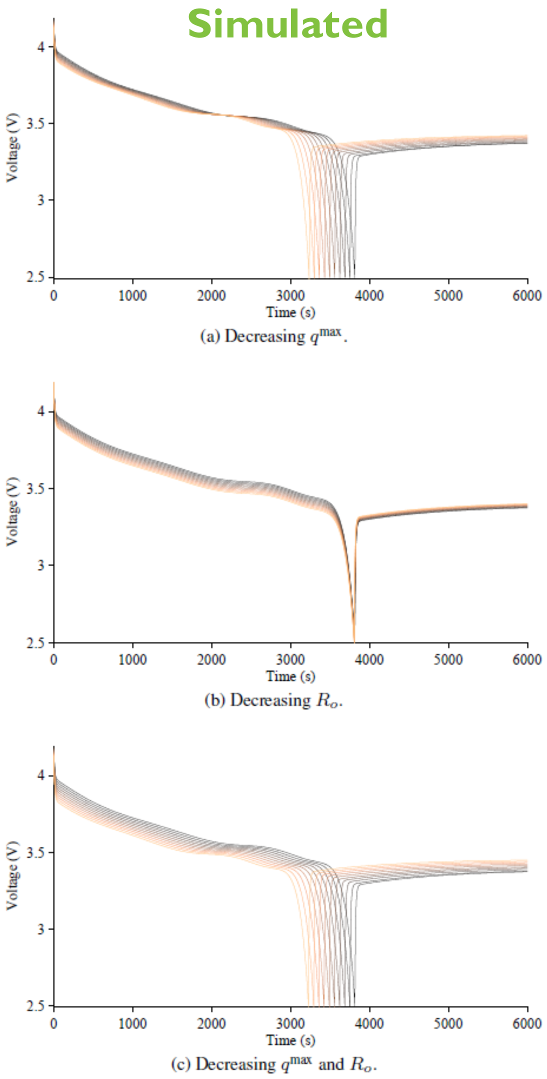


Rover Battery Discharge Curve

Battery Aging

- Contributions from both decrease in mobile Li ions (lost due to side reactions related to aging) and increase in internal resistance
  - Modeled with decrease in “q^{max}” parameter, used to compute mole fraction
  - Modeled with increase in “R_o” parameter capturing lumped resistances





(a) Decreasing $q^{\max}$.

(b) Decreasing R_o.

(c) Decreasing $q^{\max}$ and R_o.

Estimation

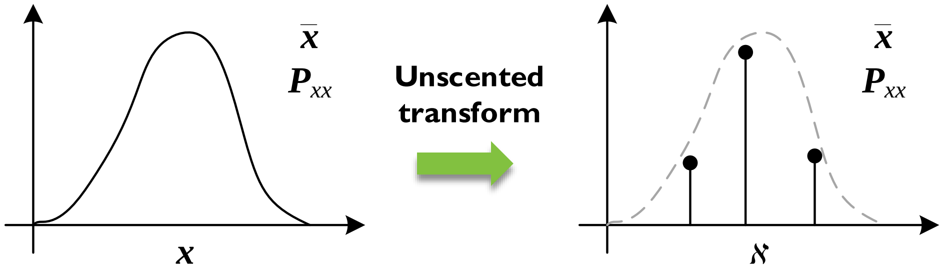
> How can the system state be estimated?
> How does fault diagnosis fit in?
> How is uncertainty in estimation handled?

Estimation Problem

- First problem of prognostics is state-parameter estimation
  - What is the current system state and its associated uncertainty?
  - Input: system outputs y from k_0 to k, $y(k_0{:}k)$
  - Output: $p(x(k),\theta(k)|y(k_0{:}k))$
- There are several algorithms that accomplish this, e.g.,
  - Kalman filter (linear systems, additive Gaussian noise)
  - Extended Kalman filter (nonlinear systems, additive Gaussian noise)
  - Unscented Kalman filter (nonlinear systems, additive Gaussian noise)
  - Particle filter (nonlinear systems)

Unscented Kalman Filter

- The UKF is an approximate nonlinear filter, and assumes additive, Gaussian process and sensor noise
- Handles nonlinearity by using the concept of sigma points
  - Transform mean and covariance of state into set of samples, called sigma points, selected deterministically to preserve mean and covariance
  - Sigma points are transformed through the nonlinear function and recover mean and covariance of transformed sigma points



$$w^i = \begin{cases} \dfrac{\kappa}{(n_x+\kappa)}, & i = 0 \\ \dfrac{1}{2(n_x+\kappa)}, & i = 1,\ldots,2n_x \end{cases}$$

$$\mathcal{X}^i = \begin{cases} \bar{\mathbf{x}}, & i = 0 \\ \bar{\mathbf{x}}+\left(\sqrt{(n_x+\kappa)\mathbf{P}_{xx}}\right)^i, & i = 1,\ldots,n_x \\ \bar{\mathbf{x}}-\left(\sqrt{(n_x+\kappa)\mathbf{P}_{xx}}\right)^i, & i = n_x+1,\ldots,2n_x \end{cases}$$

Symmetric Unscented Transform

- Number of sigma points is linear in the size of the state dimension

Unscented Kalman Filter

- Kalman filter equations extended to use sigma points

Prediction Step

$$\hat{\mathcal{X}}^i_{k|k-1} = \mathbf{f}(\hat{\mathcal{X}}^i_{k-1|k-1}, \mathbf{u}_{k-1}), i = 1, \ldots, n_s$$

$$\hat{\mathcal{Y}}^i_{k|k-1} = \mathbf{h}(\hat{\mathcal{X}}^i_{k|k-1}), i = 1, \ldots, n_s$$

$$\hat{\mathbf{x}}_{k|k-1} = \sum_i^{n_s} w_i \mathcal{X}^i_{k|k-1}$$

$$\hat{\mathbf{y}}_{k|k-1} = \sum_i^{n_s} w_i \mathcal{Y}^i_{k|k-1}$$

$$\mathbf{P}_{k|k-1} = \mathbf{Q} + \sum_i^{n_s} w_i (\mathcal{X}^i_{k|k-1} - \hat{\mathbf{x}}_{k|k-1})(\mathcal{X}^i_{k|k-1} - \hat{\mathbf{x}}_{k|k-1})^T.$$

Update Step

$$\mathbf{P}_{yy} = \mathbf{R} + \sum_i^{n_s} w_i (\mathcal{Y}^i_{k|k-1} - \hat{\mathbf{y}}_{k|k-1})(\mathcal{Y}^i_{k|k-1} - \hat{\mathbf{y}}_{k|k-1})^T$$

$$\mathbf{P}_{xy} = \sum_i^{n_s} w_i (\mathcal{X}^i_{k|k-1} - \hat{\mathbf{x}}_{k|k-1})(\mathcal{Y}^i_{k|k-1} - \hat{\mathbf{y}}_{k|k-1})^T$$

$$\mathbf{K}_k = \mathbf{P}_{xy}\mathbf{P}_{yy}^{-1}$$

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k(\mathbf{y}_k - \hat{\mathbf{y}}_{k|k-1})$$

$$\mathbf{P}_{k|k} = \mathbf{P}_{k|k-1} - \mathbf{K}_k\mathbf{P}_{yy}\mathbf{K}_k^T$$

- Has medium computational complexity and covers a very large class of dynamics, but is an approximate filter

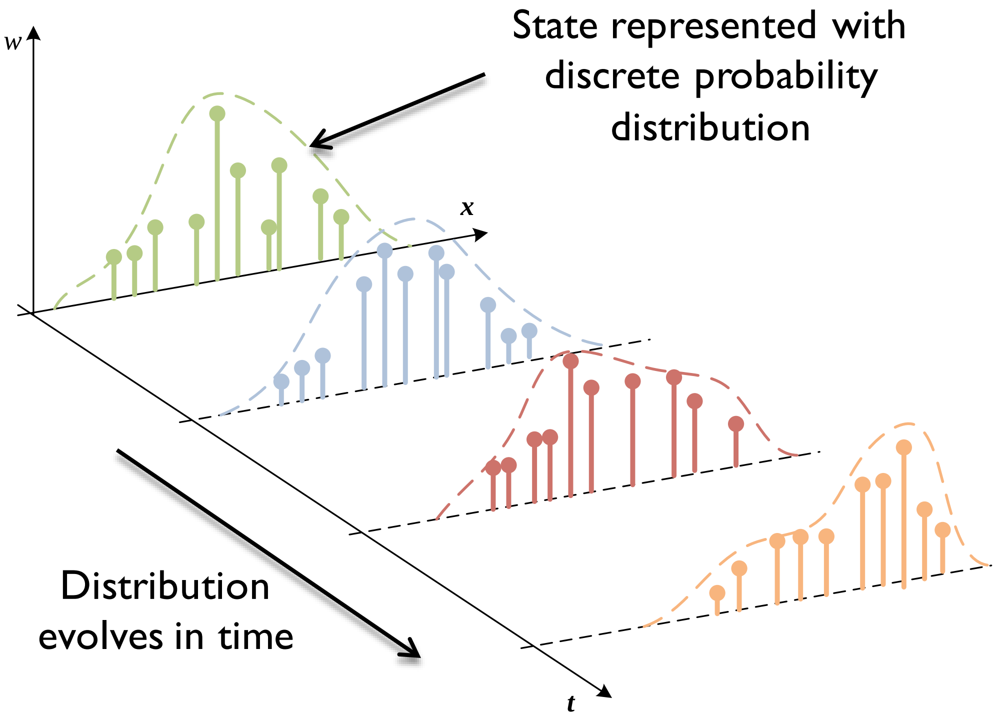
Particle Filter

- Particle filters can be applied to general nonlinear processes with non-Gaussian noise – does not restrict the dynamics in any way
  - But is an approximate filter, and is stochastic in nature
- Approximate state distribution by set of discrete weighted samples (i.e., particles):

$$\{\mathbf{x}_k^i, w_k^i\}_{i=1}^N$$

  - Suboptimal, but approach optimality as N→ ∞
- Approximates posterior as

$$p(\mathbf{x}_k|\mathbf{y}_{0:k}) \approx \sum_{i=1}^{N} w_k^i \delta_{\mathbf{x}_k^i}(d\mathbf{x}_k)$$

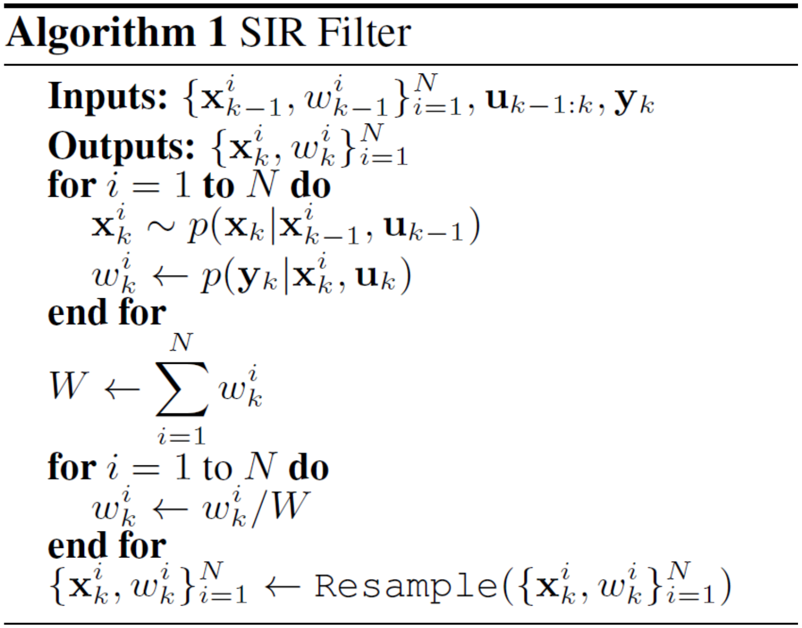


State represented with discrete probability distribution

Distribution evolves in time

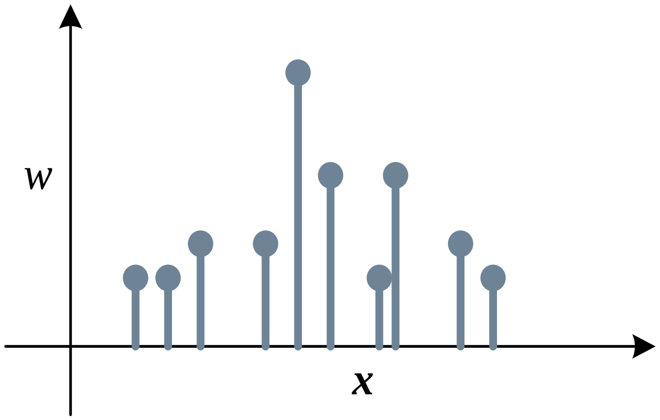
Particle Filter

- Begin with initial particle population
- Predict evolution of particles one step ahead
- Compute particle weights based on likelihood of given observations
- Resample to avoid degeneracy issues
  - Degeneracy is when small number of particles have high weight and the rest have very low weight
  - Avoid wasting computation on particles that do not contribute to the approximation



Algorithm 1 SIR Filter

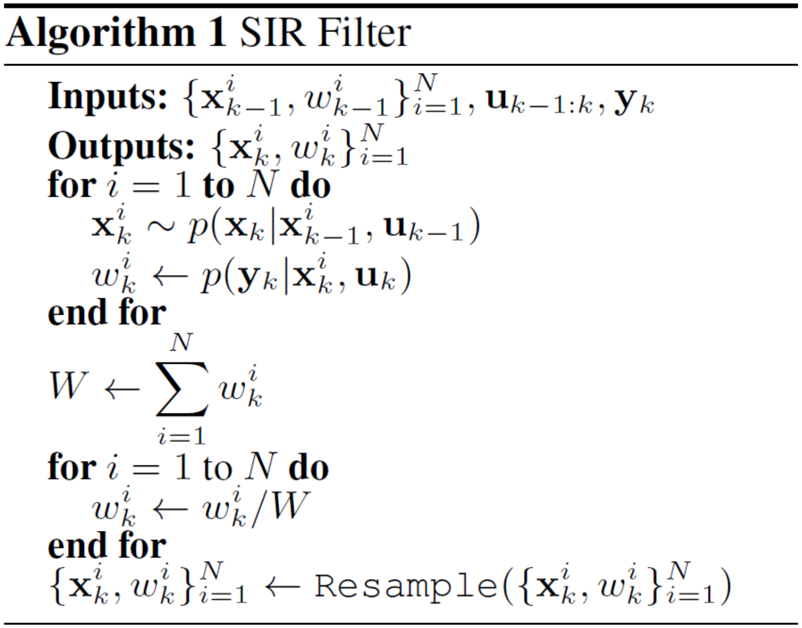
```
Inputs: $\{\mathbf{x}_{k-1}^i, w_{k-1}^i\}_{i=1}^N, \mathbf{u}_{k-1:k}, \mathbf{y}_k$
Outputs: $\{\mathbf{x}_k^i, w_k^i\}_{i=1}^N$
for $i = 1$ to $N$ do
    $\mathbf{x}_k^i \sim p(\mathbf{x}_k | \mathbf{x}_{k-1}^i, \mathbf{u}_{k-1})$
    $w_k^i \leftarrow p(\mathbf{y}_k | \mathbf{x}_k^i, \mathbf{u}_k)$
end for
$W \leftarrow \sum_{i=1}^{N} w_k^i$
for $i = 1$ to $N$ do
    $w_k^i \leftarrow w_k^i / W$
end for
$\{\mathbf{x}_k^i, w_k^i\}_{i=1}^N \leftarrow \texttt{Resample}(\{\mathbf{x}_k^i, w_k^i\}_{i=1}^N)$
```



Initial Particle Population

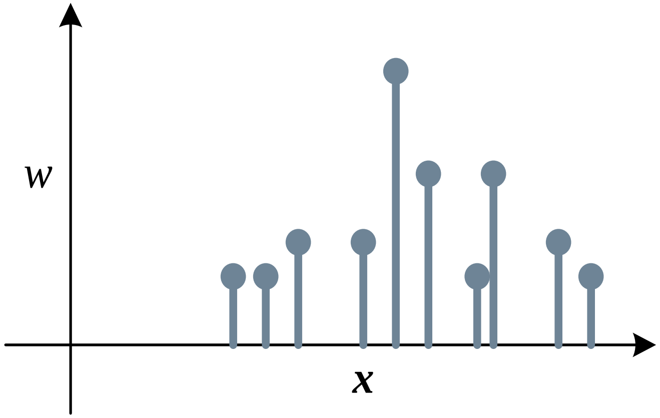
Particle Filter

- Begin with initial particle population
- Predict evolution of particles one step ahead
- Compute particle weights based on likelihood of given observations
- Resample to avoid degeneracy issues
  - Degeneracy is when small number of particles have high weight and the rest have very low weight
  - Avoid wasting computation on particles that do not contribute to the approximation



Algorithm 1 SIR Filter

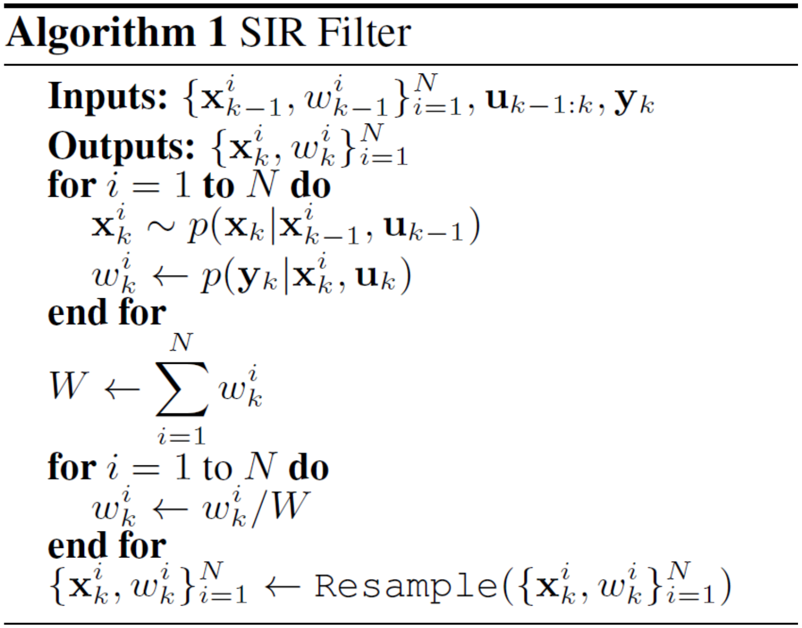
```
**Inputs:** $\{\mathbf{x}_{k-1}^i, w_{k-1}^i\}_{i=1}^N, \mathbf{u}_{k-1:k}, \mathbf{y}_k$
**Outputs:** $\{\mathbf{x}_k^i, w_k^i\}_{i=1}^N$
for $i = 1$ to $N$ do
    $\mathbf{x}_k^i \sim p(\mathbf{x}_k | \mathbf{x}_{k-1}^i, \mathbf{u}_{k-1})$
    $w_k^i \leftarrow p(\mathbf{y}_k | \mathbf{x}_k^i, \mathbf{u}_k)$
end for
$W \leftarrow \sum_{i=1}^N w_k^i$
for $i = 1$ to $N$ do
    $w_k^i \leftarrow w_k^i / W$
end for
$\{\mathbf{x}_k^i, w_k^i\}_{i=1}^N \leftarrow \texttt{Resample}(\{\mathbf{x}_k^i, w_k^i\}_{i=1}^N)$
```



Predict Evolution

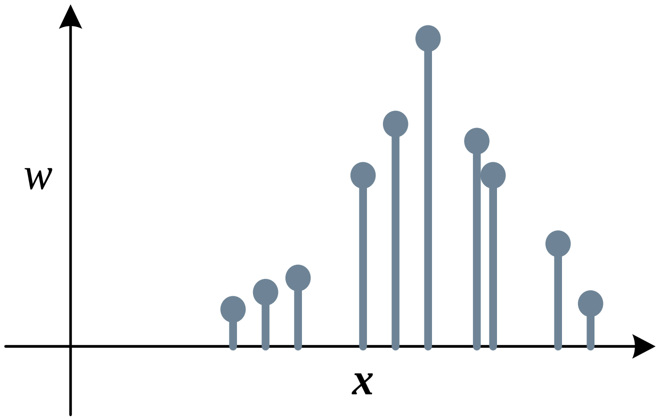
Particle Filter

- Begin with initial particle population
- Predict evolution of particles one step ahead
- Compute particle weights based on likelihood of given observations
- Resample to avoid degeneracy issues
  - Degeneracy is when small number of particles have high weight and the rest have very low weight
  - Avoid wasting computation on particles that do not contribute to the approximation



Algorithm 1 SIR Filter

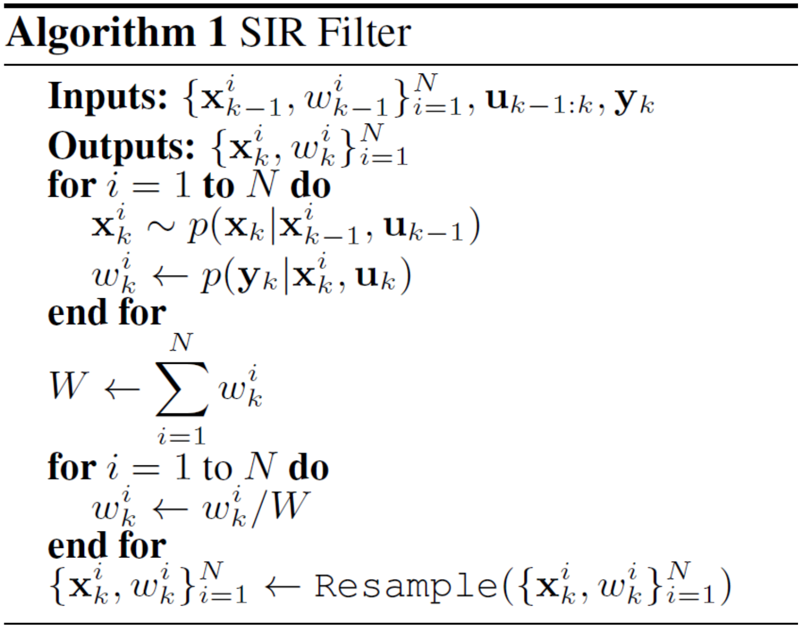
```
**Inputs:** $\{\mathbf{x}_{k-1}^i, w_{k-1}^i\}_{i=1}^N, \mathbf{u}_{k-1:k}, \mathbf{y}_k$
**Outputs:** $\{\mathbf{x}_k^i, w_k^i\}_{i=1}^N$
**for** $i = 1$ **to** $N$ **do**
    $\mathbf{x}_k^i \sim p(\mathbf{x}_k | \mathbf{x}_{k-1}^i, \mathbf{u}_{k-1})$
    $w_k^i \leftarrow p(\mathbf{y}_k | \mathbf{x}_k^i, \mathbf{u}_k)$
**end for**
$W \leftarrow \sum_{i=1}^{N} w_k^i$
**for** $i = 1$ to $N$ **do**
    $w_k^i \leftarrow w_k^i / W$
**end for**
$\{\mathbf{x}_k^i, w_k^i\}_{i=1}^N \leftarrow \texttt{Resample}(\{\mathbf{x}_k^i, w_k^i\}_{i=1}^N)$
```



Compute Weights

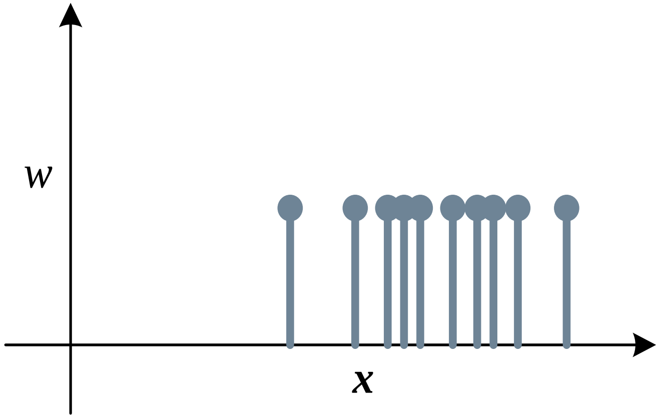
Particle Filter

- Begin with initial particle population
- Predict evolution of particles one step ahead
- Compute particle weights based on likelihood of given observations
- Resample to avoid degeneracy issues
  - Degeneracy is when small number of particles have high weight and the rest have very low weight
  - Avoid wasting computation on particles that do not contribute to the approximation



Algorithm 1 SIR Filter

```
Inputs: $\{\mathbf{x}_{k-1}^i, w_{k-1}^i\}_{i=1}^N, \mathbf{u}_{k-1:k}, \mathbf{y}_k$
Outputs: $\{\mathbf{x}_k^i, w_k^i\}_{i=1}^N$
for $i = 1$ to $N$ do
    $\mathbf{x}_k^i \sim p(\mathbf{x}_k | \mathbf{x}_{k-1}^i, \mathbf{u}_{k-1})$
    $w_k^i \leftarrow p(\mathbf{y}_k | \mathbf{x}_k^i, \mathbf{u}_k)$
end for
$W \leftarrow \sum_{i=1}^N w_k^i$
for $i = 1$ to $N$ do
    $w_k^i \leftarrow w_k^i / W$
end for
$\{\mathbf{x}_k^i, w_k^i\}_{i=1}^N \leftarrow \texttt{Resample}(\{\mathbf{x}_k^i, w_k^i\}_{i=1}^N)$
```



Resample

Joint State-Parameter Estimation

- Joint state-parameter estimation is performed within a filtering framework by augmenting the state vector with the unknown parameter vector
- Must assign an evolution to the parameters, typically a random walk

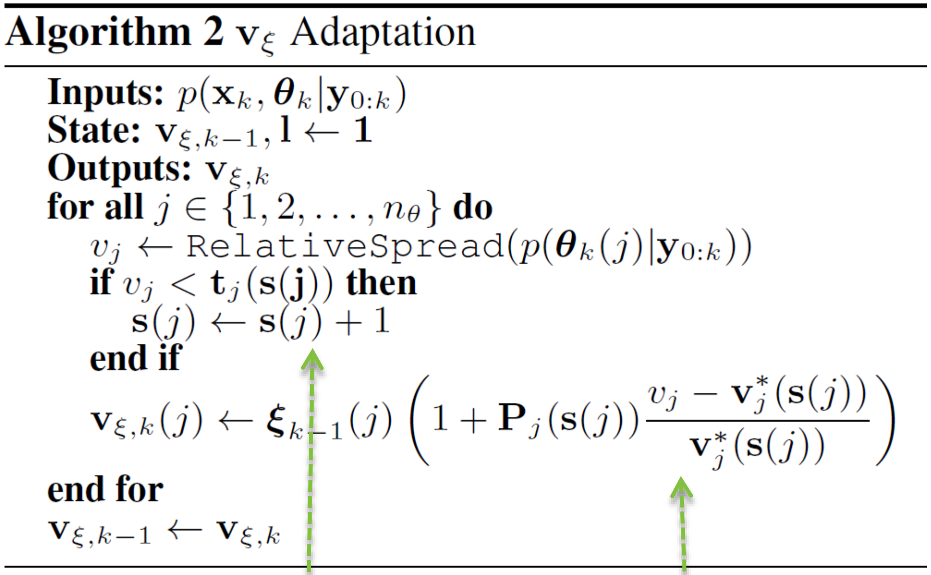
$$\theta_k = \theta_{k-1} + \xi_{k-1}$$

- The particle filter adopts this equation directly; for the UKF filter, it is represented in the corresponding diagonal of the process noise matrix
- Selection of variance of random walk noise is important
  - Variance must be large enough to ensure convergence, but small enough to ensure precise tracking
  - Optimal value depends on unknown parameter value
  - Should tune online to maximize performance

M. Orchard, F. Tobar, and G. Vachtsevanos, "Outer feedback correction loops in particle filtering-based prognostic algorithms: Statistical performance comparison," Studies in Informatics and Control, no. 4, pp. 295–304, Dec. 2009.
B. Saha and K. Goebel, "Model adaptation for prognostics in a particle filtering framework," International Journal of Prognostics and Health Management, vol. 2, no. 1, 2011.

Variance Control

- ξ values tuned initially for maximum possible wear rates
- Try to control the amount of relative spread of parameter estimate to a desired level (e.g., 10%)
  - Since it is relative, applies equally to any wear parameter value
  - Can use relative median absolute deviation (RMAD), relative standard deviation (RSD), among others
- Several stages to control adaptation
  - Convergence: Control to large spread (eg 50%) until threshold reached (eg 60%)
  - Tracking: Control to desired spread (eg 10%)
- Control based on percent error between actual spread and desired spread with parameter P
  - Increase random walk variance if parameter variance is too low, else decrease



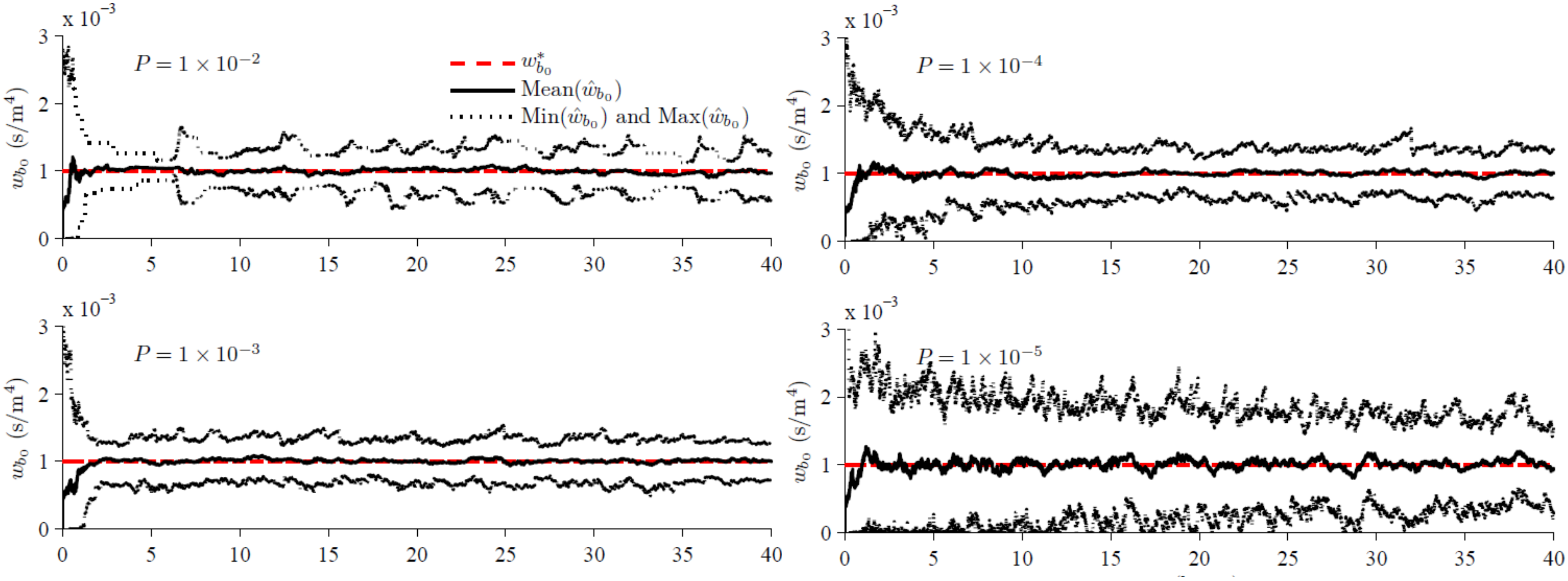
```
Algorithm 2 $\mathbf{v}_\xi$ Adaptation
Inputs: $p(\mathbf{x}_k, \boldsymbol{\theta}_k|\mathbf{y}_{0:k})$
State: $\mathbf{v}_{\xi,k-1}, \mathbf{l} \leftarrow \mathbf{1}$
Outputs: $\mathbf{v}_{\xi,k}$
for all $j \in \{1, 2, \ldots, n_\theta\}$ do
    $v_j \leftarrow \texttt{RelativeSpread}(p(\boldsymbol{\theta}_k(j)|\mathbf{y}_{0:k}))$
    if $v_j < \mathbf{t}_j(\mathbf{s}(\mathbf{j}))$ then
        $\mathbf{s}(j) \leftarrow \mathbf{s}(j) + 1$
    end if
    $\mathbf{v}_{\xi,k}(j) \leftarrow \boldsymbol{\xi}_{k-1}(j)\left(1 + \mathbf{P}_j(\mathbf{s}(j))\frac{v_j - \mathbf{v}^*_j(\mathbf{s}(j))}{\mathbf{v}^*_j(\mathbf{s}(j))}\right)$
end for
$\mathbf{v}_{\xi,k-1} \leftarrow \mathbf{v}_{\xi,k}$
```

Proportional control based on error between actual and desired relative spread

Move to next stage when threshold crossed

Variance Control Tuning

- Initial spread needs to be large enough to find the right value
- Final spread needs to be small enough for accurate tracking
- Proportional gain needs to be adjusted so that it converges
- Want fast convergence with small spread afterwards



What About Diagnosis?

- Before estimating the system state, need to know if the model is correct!
  - Have a nominal model $\mathcal{M}_n$
  - When a fault occurs, the model has changed in some way (different parameter value(s) and/or different structure)
  - Now we have a new model $\mathcal{M}_f$ for fault f
- Diagnosis gives an informed state estimate, and can add additional uncertainty to the problem

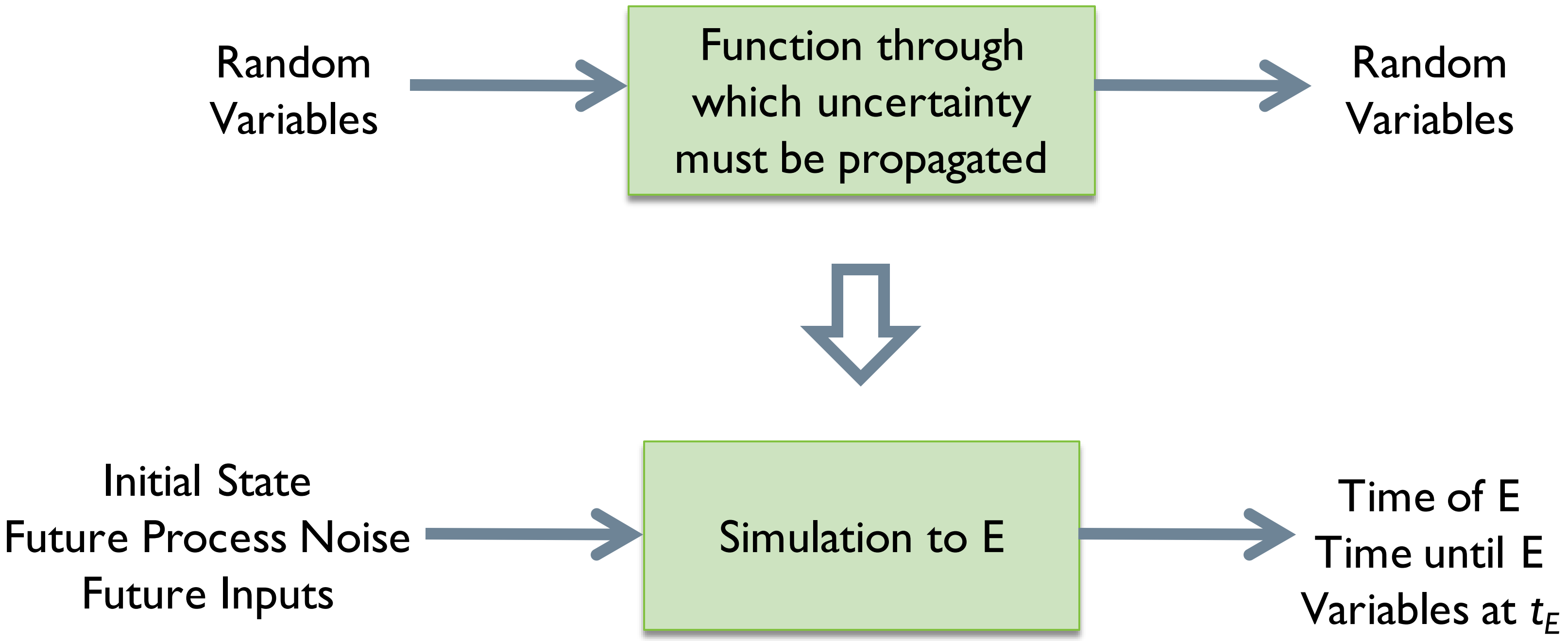
Prediction

> How is uncertainty represented concisely?
> How is uncertainty folded into prediction?
> What algorithms are used for prediction?

Prediction Problem

- Second problem of prognostics is prediction
  - What is k_E and what is its uncertainty?
  - Input: $p(x(k),\theta(k)|y(k_0{:}k))$
  - Output: $p(kE)$
- Most algorithms operate by simulating samples forward in time until E
- Algorithms must account for several sources of uncertainty besides that in the initial state
  - A representation of that uncertainty is required for the selected prediction algorithm
  - A specific description of that uncertainty is required (e.g., mean, variance)

Uncertainty Quantification



Random Variables

Function through which uncertainty must be propagated

Random Variables

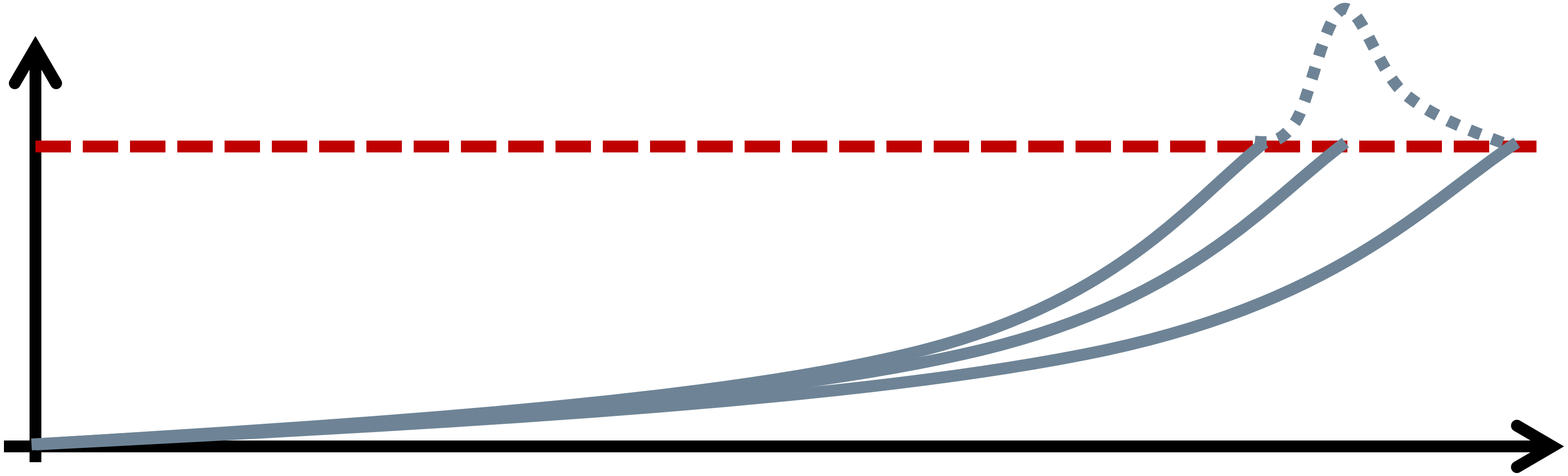
Initial State
Future Process Noise
Future Inputs

Simulation to E

Time of E
Time until E
Variables at t_E

Are There Closed Form Solutions?

- Almost always, no
- Why?
  - Even with a linear system degradation function and normal noise, the addition of the threshold function makes the problem nonlinear
  - In any case, degradation functions are almost always nonlinear



S. Sankararaman, K. Goebel. Uncertainty in Prognostics: Computational Methods and Practical Challenges. In the Proceedings of the 2014 IEEE Aerospace Conference, Big Sky, MT, Mar 1 - 8, 2014.

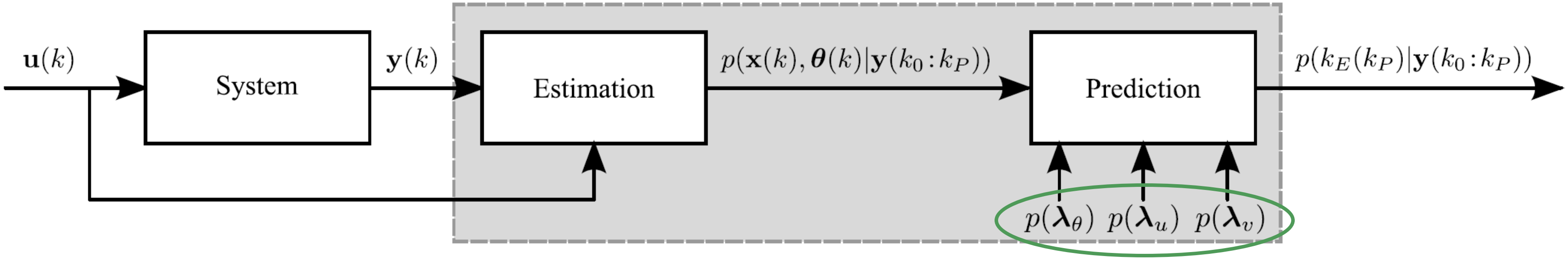
Uncertainty Representation

- To predict k_E, need to account for following sources of uncertainty:
  - Initial state at k_P: $\mathbf{x}(k_p)$
  - Parameter values for k_P to k_E: $\mathbf{\Theta}_{k_P}$
  - Inputs for k_P to k_E: $\mathbf{U}_{k_P}$
  - Process noise for k_P to k_E: $\mathbf{V}_{k_P}$
- These are all trajectories…
  - Difficult to represent directly uncertainty in trajectories, instead represent indirectly through concept of *surrogate variables*
    - Surrogate variables are random variables that parameterize a trajectory
    - Describe probability distributions for these variables
    - Sample these random variables to sample a trajectory
  - For example, if trajectory is constant selected from some distribution, we sample that variable, i.e, $u(k) = c$, for all $k>k_P$
    - Or, $u(k) = c_1k+ c_2k^2, \ldots$

Prognostics Architecture (Revisited)

1 System receives inputs, produces outputs

2 Estimate current state and parameter values



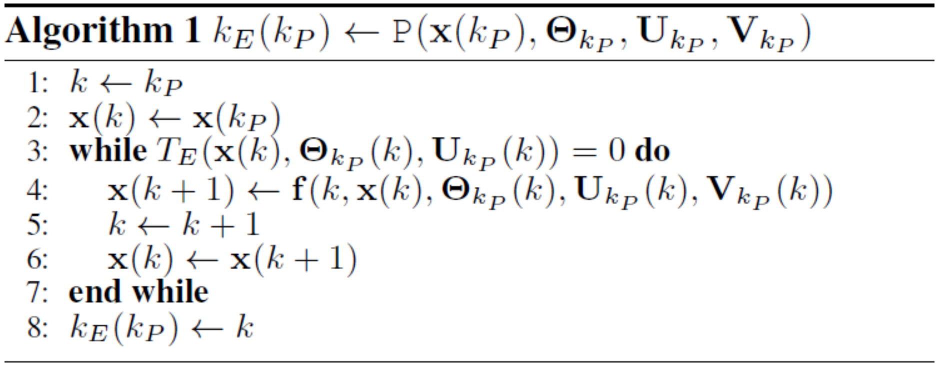
$\mathbf{u}(k)$
System
$\mathbf{y}(k)$
Estimation
$p(\mathbf{x}(k),\boldsymbol{\theta}(k)|\mathbf{y}(k_0:k_P))$
Prediction
$p(k_E(k_P)|\mathbf{y}(k_0:k_P))$
$p(\boldsymbol{\lambda}_\theta)$
$p(\boldsymbol{\lambda}_u)$
$p(\boldsymbol{\lambda}_v)$

3 Use surrogate variable distributions

4 Predict probability distributions for k_E, Δk_E

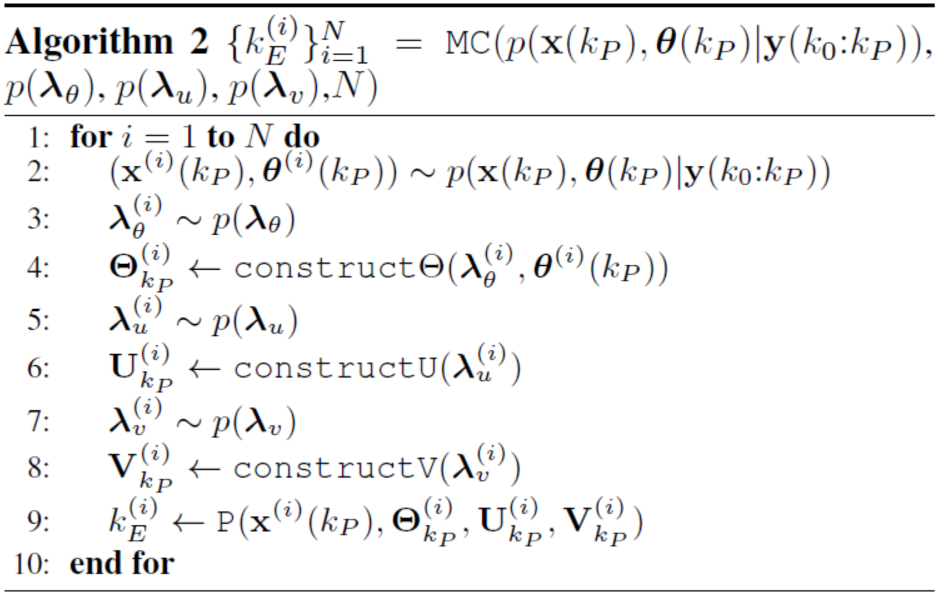
Prediction

- The `P` function takes an initial state, and a parameter, an input, and a process noise trajectory
  - Simulates state forward using **f** until E is reached to computes k_E for a single sample
- Top-level prediction algorithm calls `P`
  - These algorithms differ by how they compute samples upon which to call `P`
- Monte Carlo algorithm (`MC`) takes as input
  - Initial state-parameter estimate
  - Probability distributions for the surrogate variables for the parameter, input, and process noise trajectories
  - Number of samples, N
- `MC` samples from its input distributions, and computes k_E
- The "`construct`" functions describe how to construct a trajectory given surrogate variable samples



Algorithm 1 $k_E(k_P) \leftarrow \texttt{P}(\mathbf{x}(k_P), \boldsymbol{\Theta}_{k_P}, \mathbf{U}_{k_P}, \mathbf{V}_{k_P})$

```
$k \leftarrow k_P$
$\mathbf{x}(k) \leftarrow \mathbf{x}(k_P)$
while $T_E(\mathbf{x}(k), \boldsymbol{\Theta}_{k_P}(k), \mathbf{U}_{k_P}(k)) = 0$ do
    $\mathbf{x}(k+1) \leftarrow \mathbf{f}(k, \mathbf{x}(k), \boldsymbol{\Theta}_{k_P}(k), \mathbf{U}_{k_P}(k), \mathbf{V}_{k_P}(k))$
    $k \leftarrow k+1$
    $\mathbf{x}(k) \leftarrow \mathbf{x}(k+1)$
end while
$k_E(k_P) \leftarrow k$
```

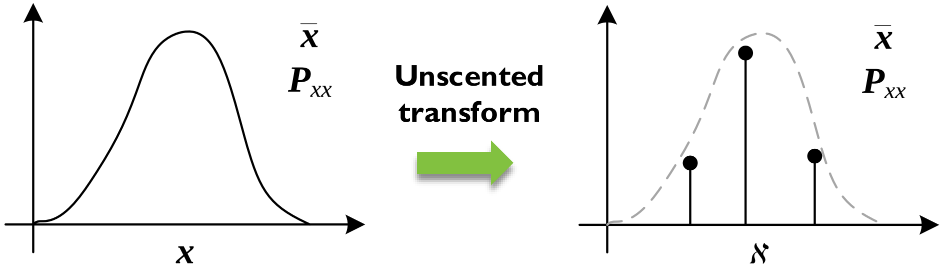


Algorithm 2 $\{k_E^{(i)}\}_{i=1}^N = \texttt{MC}(p(\mathbf{x}(k_P), \boldsymbol{\theta}(k_P)|\mathbf{y}(k_0{:}k_P)), p(\boldsymbol{\lambda}_\theta), p(\boldsymbol{\lambda}_u), p(\boldsymbol{\lambda}_v), N)$

```
for $i = 1$ to $N$ do
    $(\mathbf{x}^{(i)}(k_P), \boldsymbol{\theta}^{(i)}(k_P)) \sim p(\mathbf{x}(k_P), \boldsymbol{\theta}(k_P)|\mathbf{y}(k_0{:}k_P))$
    $\boldsymbol{\lambda}_\theta^{(i)} \sim p(\boldsymbol{\lambda}_\theta)$
    $\boldsymbol{\Theta}_{k_P}^{(i)} \leftarrow \texttt{construct}\Theta(\boldsymbol{\lambda}_\theta^{(i)}, \boldsymbol{\theta}^{(i)}(k_P))$
    $\boldsymbol{\lambda}_u^{(i)} \sim p(\boldsymbol{\lambda}_u)$
    $\mathbf{U}_{k_P}^{(i)} \leftarrow \texttt{constructU}(\boldsymbol{\lambda}_u^{(i)})$
    $\boldsymbol{\lambda}_v^{(i)} \sim p(\boldsymbol{\lambda}_v)$
    $\mathbf{V}_{k_P}^{(i)} \leftarrow \texttt{constructV}(\boldsymbol{\lambda}_v^{(i)})$
    $k_E^{(i)} \leftarrow \texttt{P}(\mathbf{x}^{(i)}(k_P), \boldsymbol{\Theta}_{k_P}^{(i)}, \mathbf{U}_{k_P}^{(i)}, \mathbf{V}_{k_P}^{(i)})$
end for
```

Input Sampling Methods

- Exhaustive
  - Sample entire input space (if finite and not too large)
- Random
  - Sample randomly from input space (a sufficient number of times)
- Unscented Transform
  - Transform mean and covariance of state into set of samples, called sigma points, selected **deterministically** to preserve mean and covariance
  - Sigma points are transformed through the nonlinear function and recover mean and covariance of transformed sigma points
  - Number of sigma points is linear in the dimension of the space being sampled



$$w^i = \begin{cases} \dfrac{\kappa}{(n_x+\kappa)}, & i=0 \\ \dfrac{1}{2(n_x+\kappa)}, & i=1,\ldots,2n_x \end{cases}$$

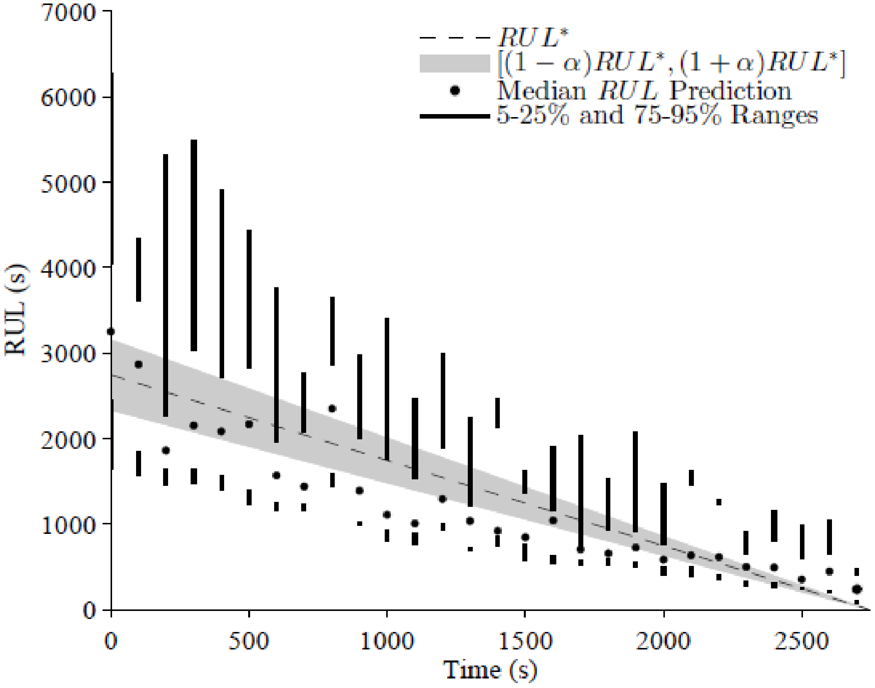
$$\mathcal{X}^i = \begin{cases} \bar{\mathbf{x}}, & i=0 \\ \bar{\mathbf{x}}+\left(\sqrt{(n_x+\kappa)\mathbf{P}_{xx}}\right)^i, & i=1,\ldots,n_x \\ \bar{\mathbf{x}}-\left(\sqrt{(n_x+\kappa)\mathbf{P}_{xx}}\right)^i, & i=n_x+1,\ldots,2n_x \end{cases}$$

Symmetric Unscented Transform

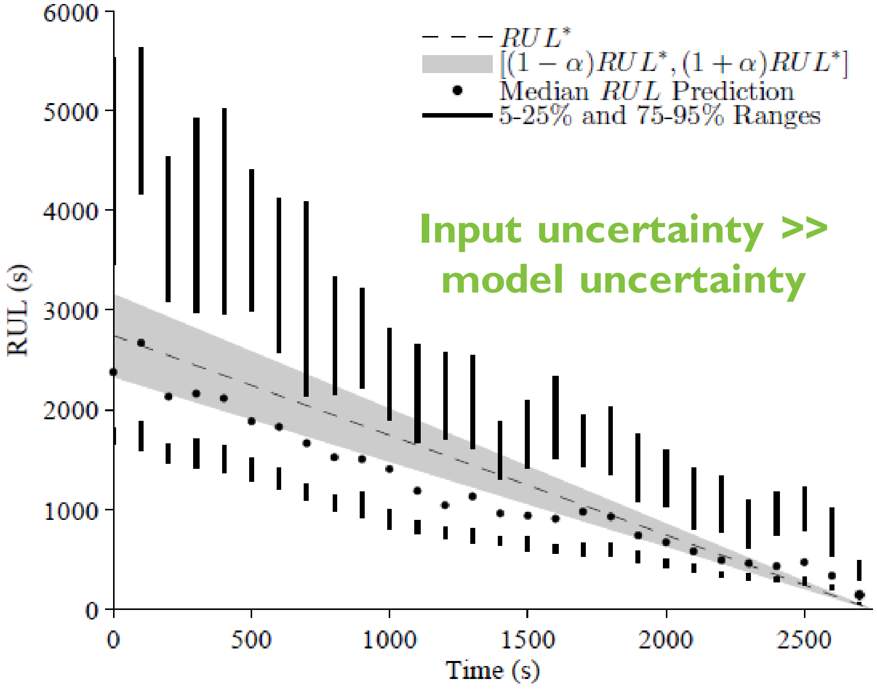
Example: Batteries

- Predicting end of discharge (EOD), where RUL is time until EOD
- Assume future inputs are unknown, with constant discharge drawn from uniform distribution from 1 to 4 A: one surrogate variable for input trajectories
- Sample randomly from this distribution at each prediction point

10 samples



100 samples

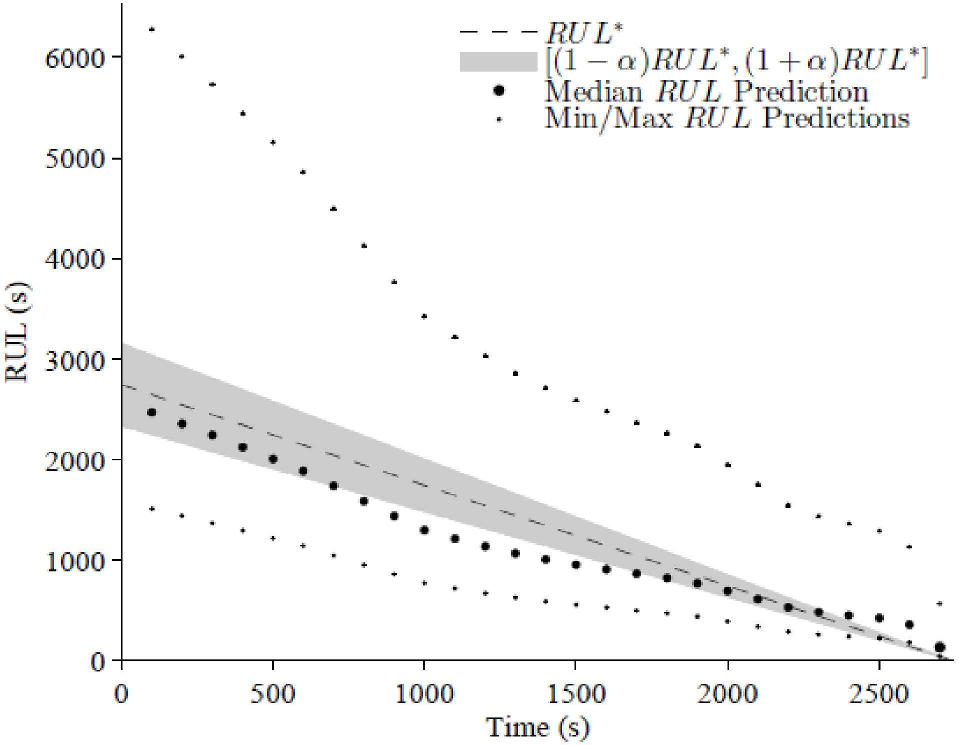


Input uncertainty >> model uncertainty

Example: Batteries

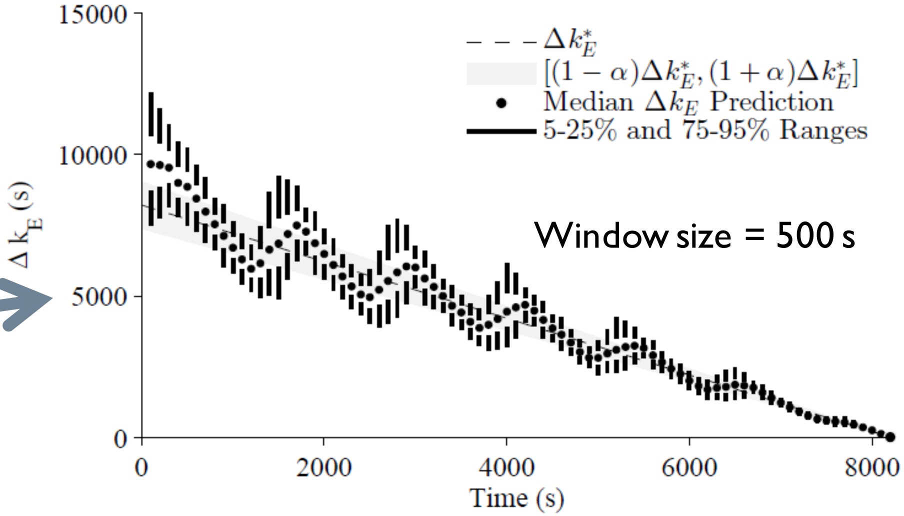
- Can sample from future input trajectories using unscented transform
- For selected tuning parameter, sigma points correspond to mean and bounds of uniform distribution
- Simulate forward three trajectories for each prediction point
- Mean and variance of RUL distribution match closely those obtained through random sampling

Get same mean/variance as with sampling approach at **3% of the computational cost** (3 samples vs 100 samples)

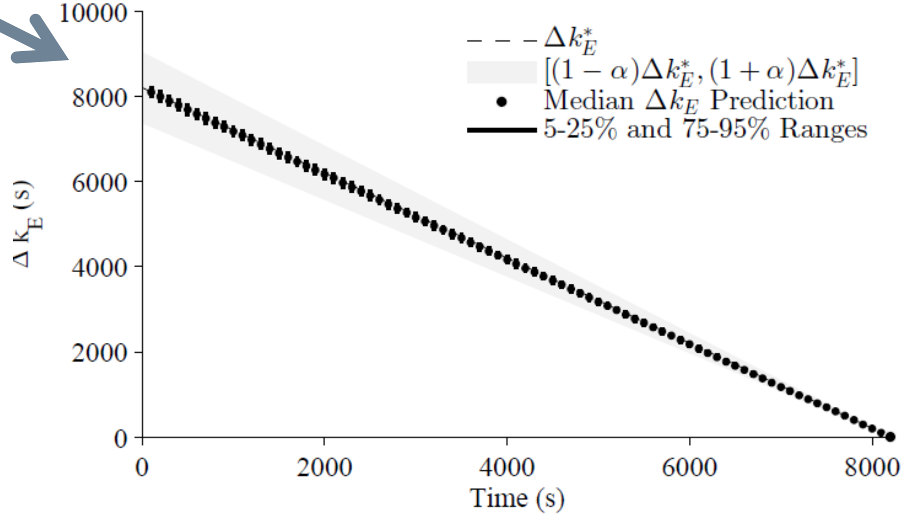


Example: Rover

- Rover must visit different waypoints at known speed, battery input is motor power
- How to describe future input trajectories?
  - Method 1: Assume future motor power is the same as past motor power over some finite time window
  - Method 2: Construct a trajectory based on a set of surrogate variables for distance traveled between consecutive waypoints and average power between them



Window size = 500 s



Uncertainty is reduced because use knowledge of future waypoints and speeds

Distributed Prognostics

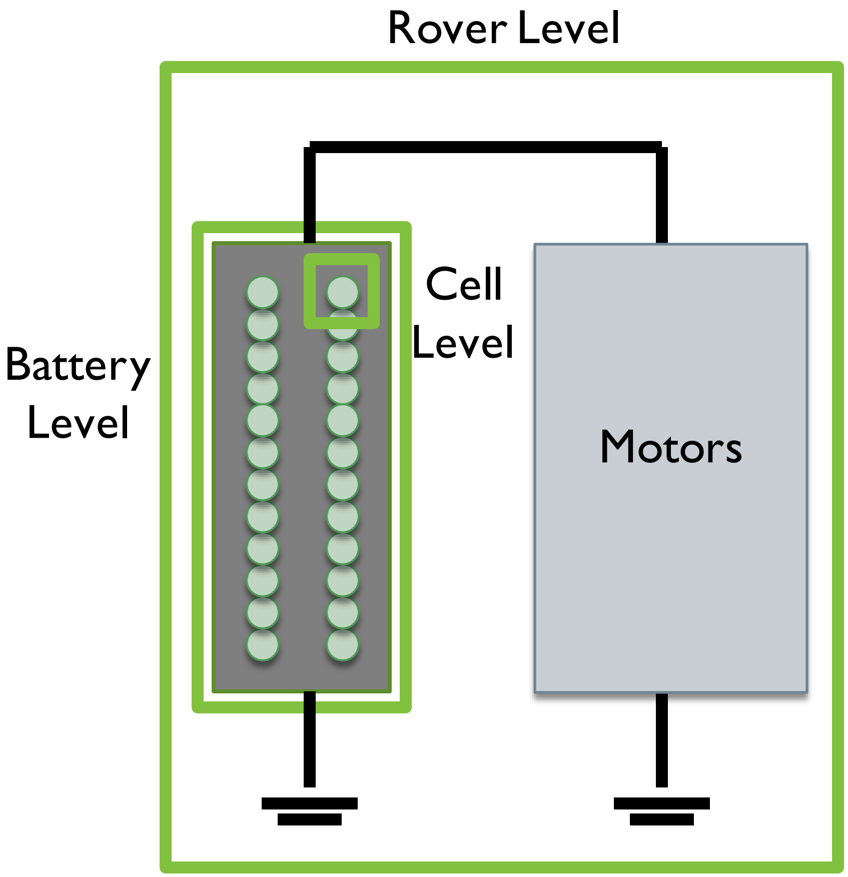
What about prognostics at the system level?

How do we distribute prognostics?

How do we use structural model decomposition?

System-Level Prognostics

- Most prognostics approaches focus on components, and not the systems they reside in
- For the rover, we want to predict a system-level event, i.e., when the rover can no longer provide enough power to the motors
  - Cell-level event: end of discharge (EOD)
  - Battery-level event: EOD (when any one cell within the battery reaches EOD)
  - Rover-level event: EOD or end of mission (EOM) (when any single battery at EOD)

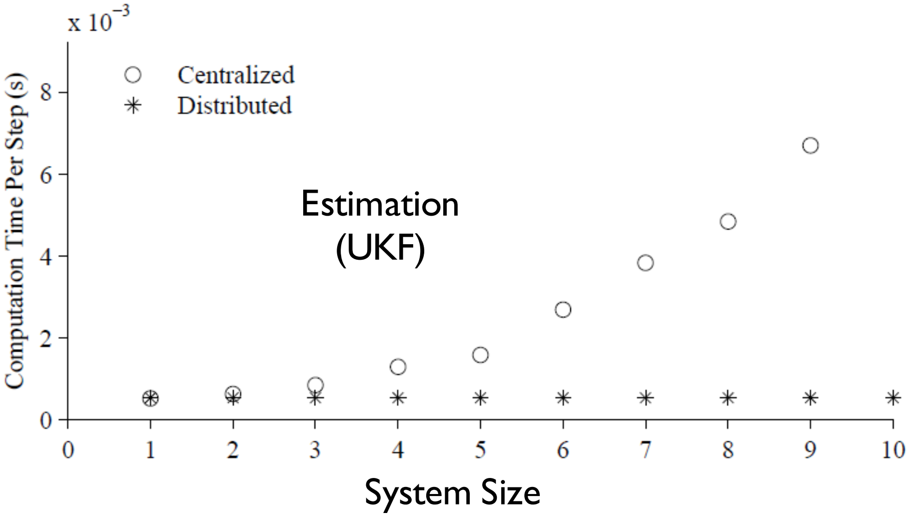


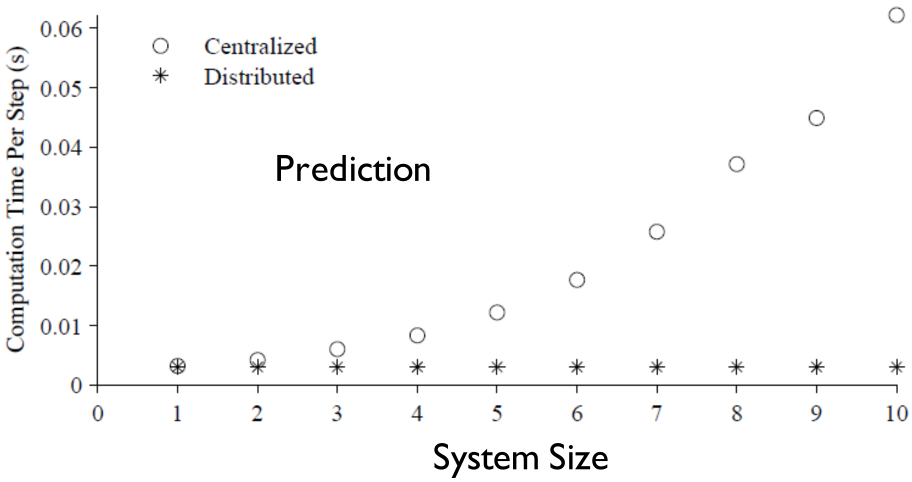
System-Level Prognostics

- In order to make accurate system-level predictions, we cannot ignore the interactions of the different components
  - The rover commands determine the local future inputs to the battery cells, so ignoring this interaction adds prediction uncertainty, a system-level perspective is required
- The problem formulation remains the same, only the model changes
  - Have local events E_i, where global event E occurs when any of the local events occurs
  - For each E_i, can define a local T_{Ei}
  - T_E can be composed from the T_{Ei}s
- Can simply use the previous algorithms

Distributed Prognostics

- … but the previous algorithms do not scale!

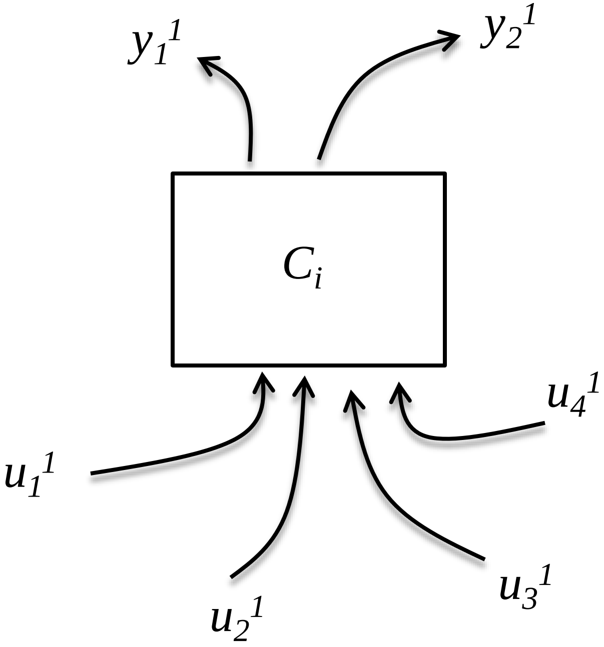




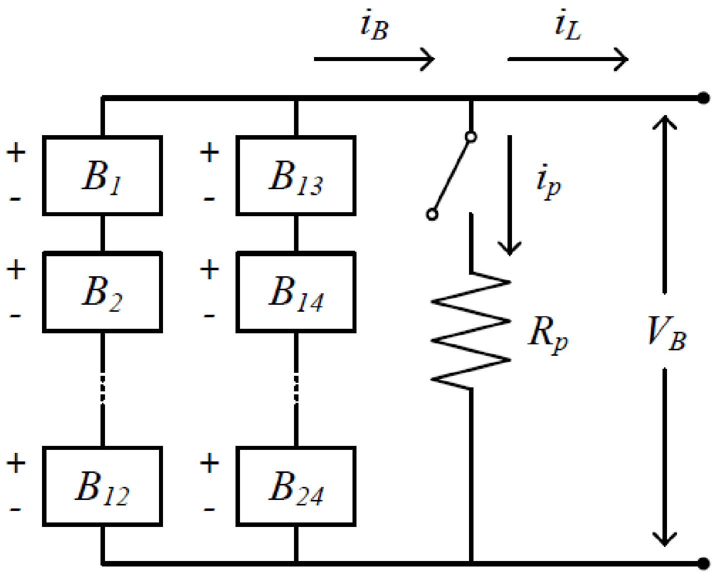
- A distributed solution is needed for large-scale systems, and for system-level prognostics problems
- Propose to *decompose* the *global* prognostics problem, by decomposing the *global model*, into *local* independent subproblems for *local submodels*
  - Use structural model decomposition
- Independent subproblems are trivially distributed and parallelized

Structural Model Decomposition

- Model = (X,θ,U,Y,C), set of states X, parameters θ, inputs U, outputs Y, constraints C
- Submodel = $(X_i,\theta_i,U_i,Y_i,C_i)$, set of states X_i, parameters θ_i, inputs U_i, outputs Y_i, constraints C_i
  - Variables can be assigned as local inputs if their values are known (e.g., they are measured)
- Find minimal submodels that satisfy a certain set of requirements
  - For distributed estimation, Y_i is a singleton, U_i chosen from U and Y- Y_i, generate one submodel for each sensor (for each y in Y_i)
  - For distributed prognostics, U_i chosen from U_P, the set of variables whose future values may be hypothesized *a priori*, generate one submodel for each T_{Ei} constraint
- Approach related to Analytical Redundancy Relations (ARRs), Possible Conflicts (PCs), …

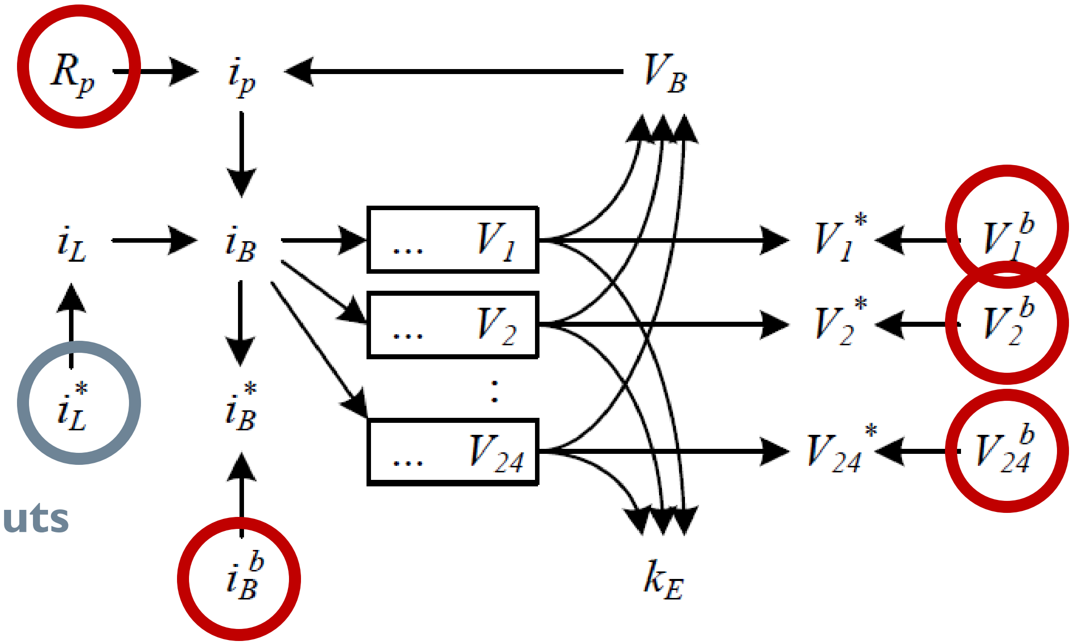


Example: Rover EPS Modeling



EPS Schematic

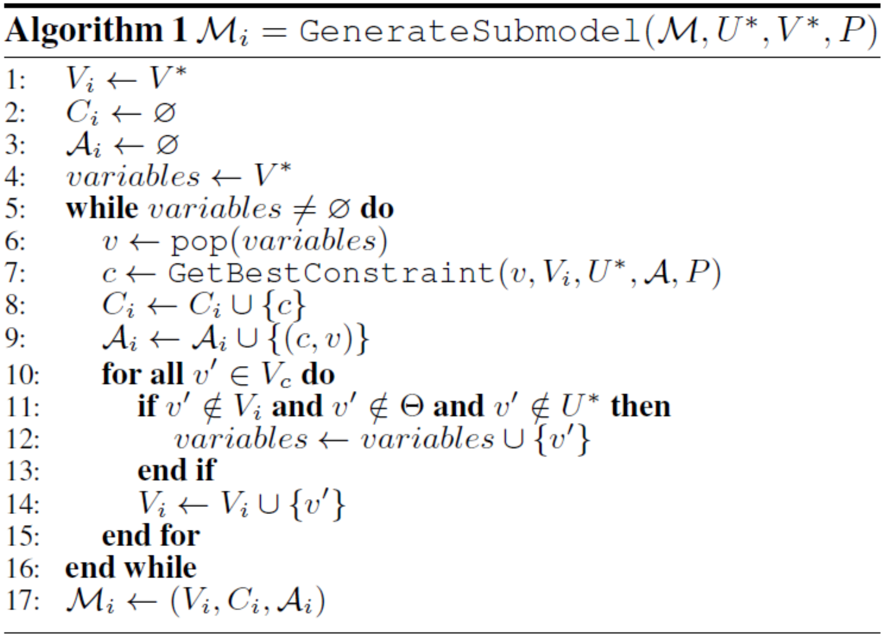
Global Model Causal Graph



Inputs

Faults

Model Decomposition Algorithm

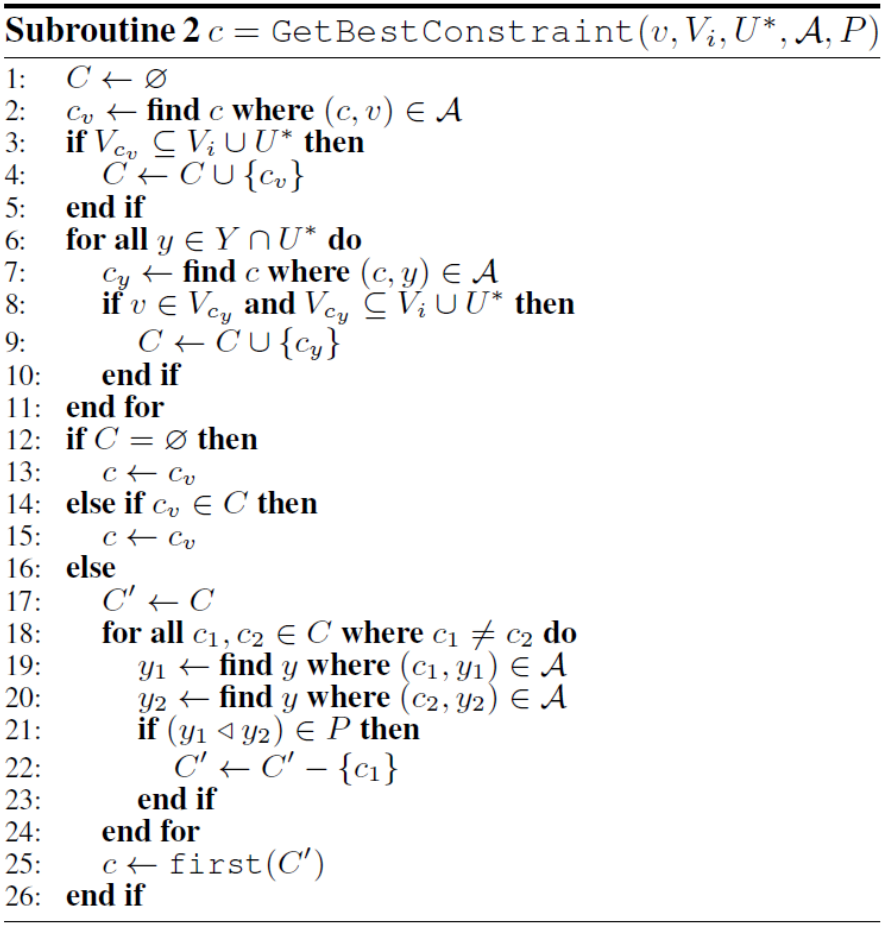


Algorithm 1 $\mathcal{M}_i = \texttt{GenerateSubmodel}(\mathcal{M}, U^*, V^*, P)$

```
$V_i \leftarrow V^*$
$C_i \leftarrow \varnothing$
$\mathcal{A}_i \leftarrow \varnothing$
$variables \leftarrow V^*$
while $variables \neq \varnothing$ do
    $v \leftarrow \mathrm{pop}(variables)$
    $c \leftarrow \texttt{GetBestConstraint}(v, V_i, U^*, \mathcal{A}, P)$
    $C_i \leftarrow C_i \cup \{c\}$
    $\mathcal{A}_i \leftarrow \mathcal{A}_i \cup \{(c, v)\}$
    for all $v' \in V_c$ do
        if $v' \notin V_i$ and $v' \notin \Theta$ and $v' \notin U^*$ then
            $variables \leftarrow variables \cup \{v'\}$
        end if
        $V_i \leftarrow V_i \cup \{v'\}$
    end for
end while
$\mathcal{M}_i \leftarrow (V_i, C_i, \mathcal{A}_i)$
```

Algorithm propagates backwards from desired outputs, finding the best constraints to resolve the variable.

Inputs in U* associated with sensors can have their causality flipped to resolve a variable.



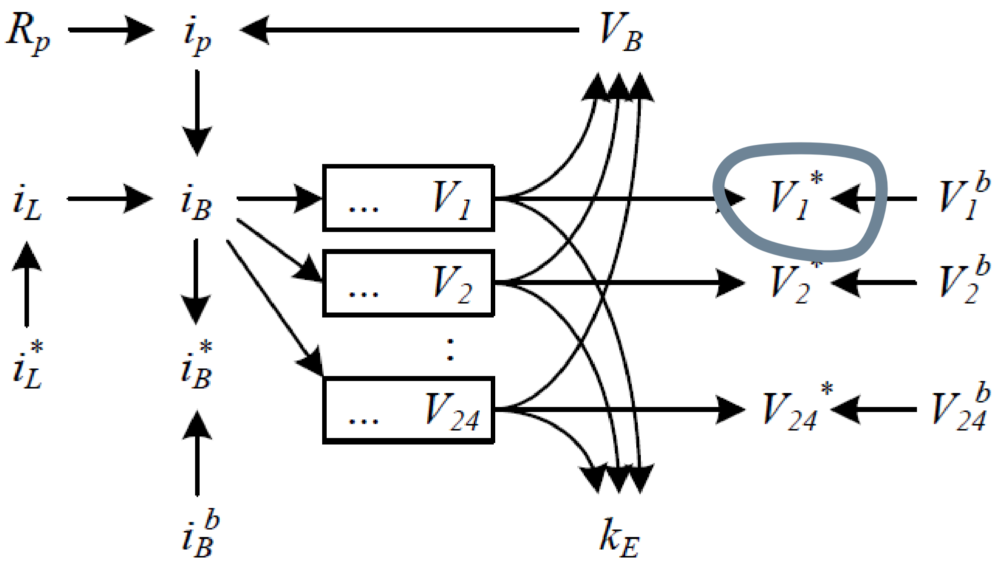
Subroutine 2 $c = \texttt{GetBestConstraint}(v, V_i, U^*, \mathcal{A}, P)$

```
$C \leftarrow \varnothing$
$c_v \leftarrow$ find $c$ where $(c, v) \in \mathcal{A}$
if $V_{c_v} \subseteq V_i \cup U^*$ then
    $C \leftarrow C \cup \{c_v\}$
end if
for all $y \in Y \cap U^*$ do
    $c_y \leftarrow$ find $c$ where $(c, y) \in \mathcal{A}$
    if $v \in V_{c_y}$ and $V_{c_y} \subseteq V_i \cup U^*$ then
        $C \leftarrow C \cup \{c_y\}$
    end if
end for
if $C = \varnothing$ then
    $c \leftarrow c_v$
else if $c_v \in C$ then
    $c \leftarrow c_v$
else
    $C' \leftarrow C$
    for all $c_1, c_2 \in C$ where $c_1 \neq c_2$ do
        $y_1 \leftarrow$ find $y$ where $(c_1, y_1) \in \mathcal{A}$
        $y_2 \leftarrow$ find $y$ where $(c_2, y_2) \in \mathcal{A}$
        if $(y_1 \lhd y_2) \in P$ then
            $C' \leftarrow C' - \{c_1\}$
        end if
    end for
    $c \leftarrow \texttt{first}(C')$
end if
```

SMD Example: Rover EPS

- States: internal to cell models
- Parameters: parasitic resistance, sensor biases
- Inputs: measured load current
- Outputs: battery current, cell voltages

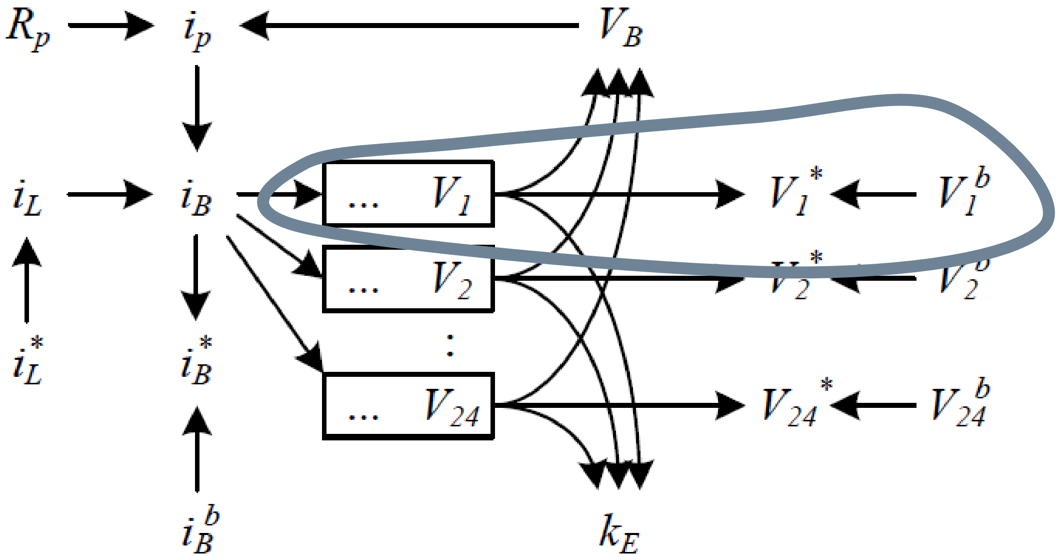
Example: Find a submodel to compute V_1^* using measured values as inputs



SMD Example: Rover EPS

- States: internal to cell models
- Parameters: parasitic resistance, sensor biases
- Inputs: measured load current
- Outputs: battery current, cell voltages

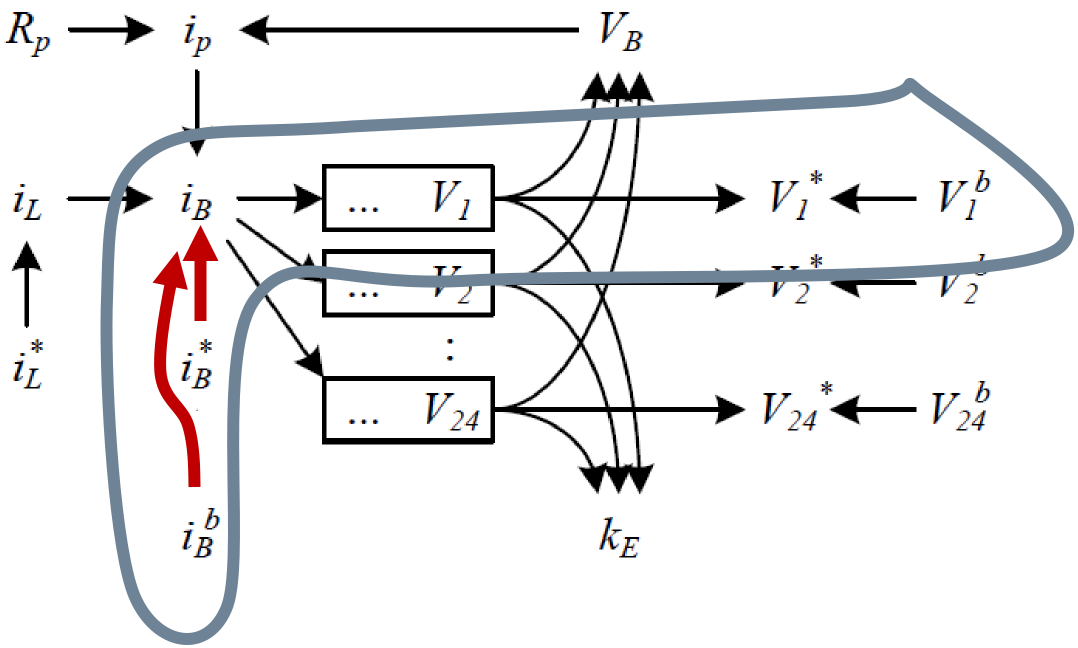
Example: Find a submodel to compute V_1^* using measured values as inputs



SMD Example: Rover EPS

- States: internal to cell models
- Parameters: parasitic resistance, sensor biases
- Inputs: measured load current
- Outputs: battery current, cell voltages

Example: Find a submodel to compute V_1^* using measured values as inputs



Distributed Prog. Architecture

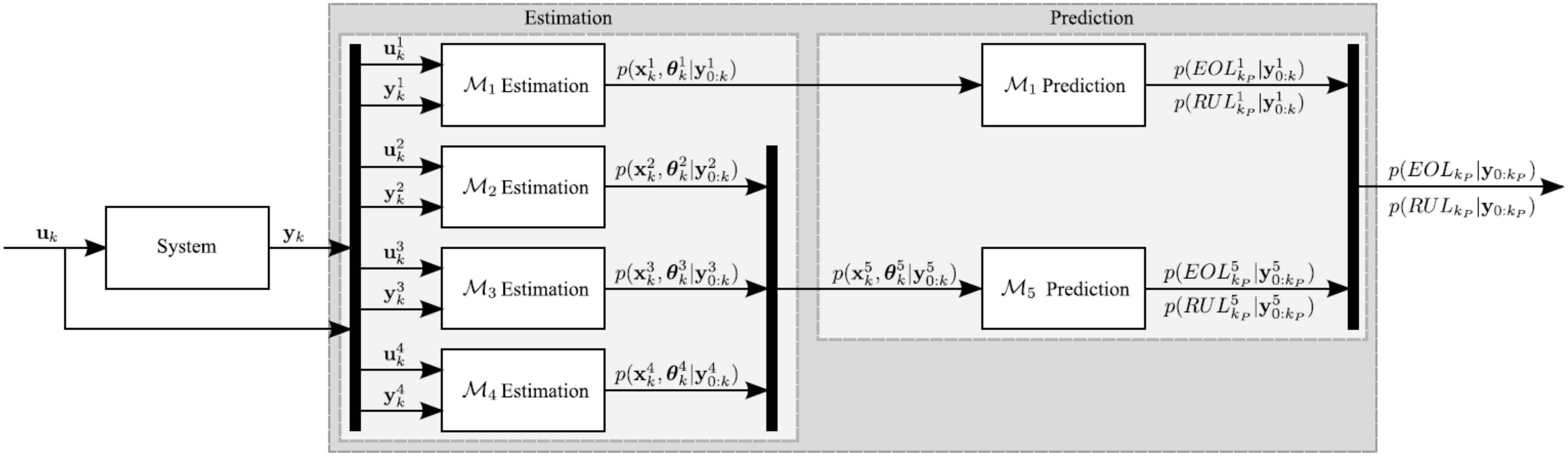
1 System receives inputs, produces outputs

2 Estimate state of local submodel

3 Merge local estimates

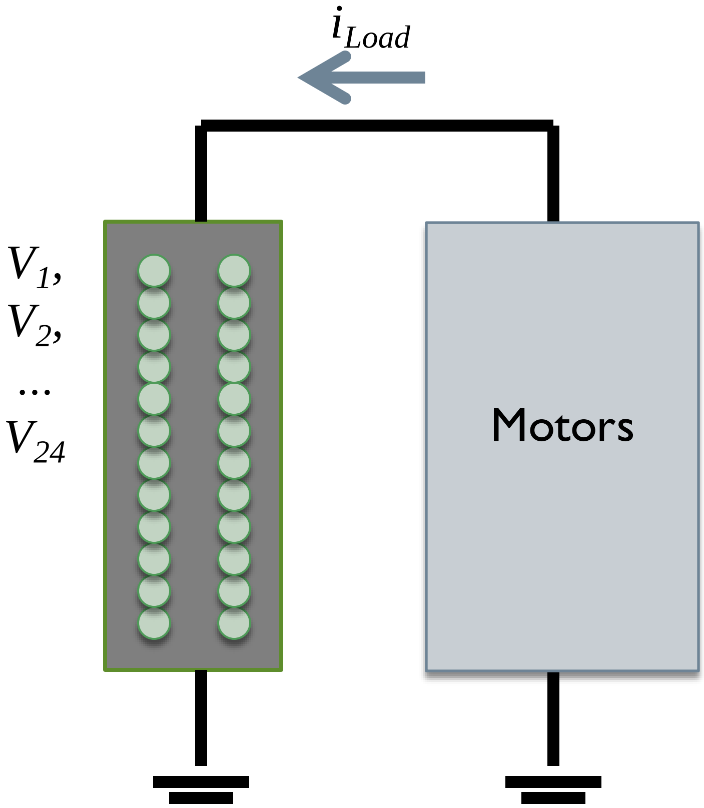
4 Predict local EOL and RUL as probability distributions

5 Merge local EOL/RUL into global EOL/RUL

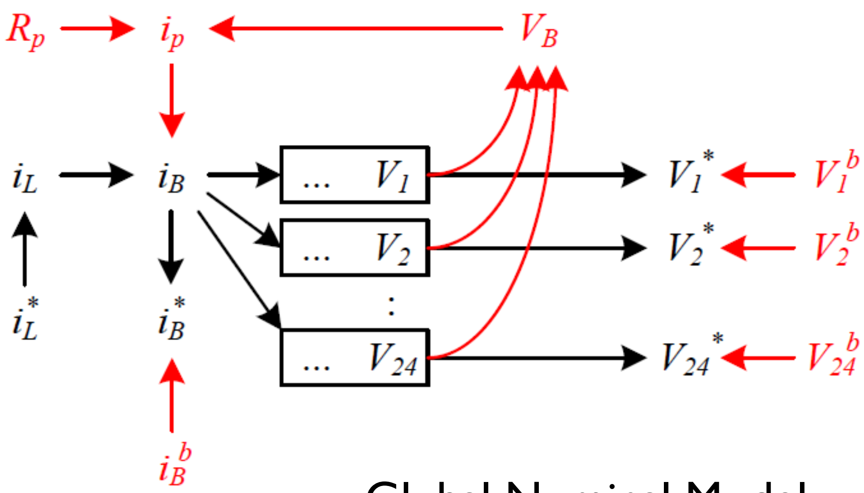

Estimation
Prediction
$\mathbf{u}_k$
System
$\mathbf{y}_k$
$\mathbf{u}_k^1$
$\mathbf{y}_k^1$
$\mathcal{M}_1$ Estimation
$p(\mathbf{x}_k^1, \boldsymbol{\theta}_k^1 | \mathbf{y}_{0:k}^1)$
$\mathbf{u}_k^2$
$\mathbf{y}_k^2$
$\mathcal{M}_2$ Estimation
$p(\mathbf{x}_k^2, \boldsymbol{\theta}_k^2 | \mathbf{y}_{0:k}^2)$
$\mathbf{u}_k^3$
$\mathbf{y}_k^3$
$\mathcal{M}_3$ Estimation
$p(\mathbf{x}_k^3, \boldsymbol{\theta}_k^3 | \mathbf{y}_{0:k}^3)$
$\mathbf{u}_k^4$
$\mathbf{y}_k^4$
$\mathcal{M}_4$ Estimation
$p(\mathbf{x}_k^4, \boldsymbol{\theta}_k^4 | \mathbf{y}_{0:k}^4)$
$\mathcal{M}_1$ Prediction
$p(EOL_{k_P}^1 | \mathbf{y}_{0:k}^1)$
$p(RUL_{k_P}^1 | \mathbf{y}_{0:k}^1)$
$p(\mathbf{x}_k^5, \boldsymbol{\theta}_k^5 | \mathbf{y}_{0:k}^5)$
$\mathcal{M}_5$ Prediction
$p(EOL_{k_P}^5 | \mathbf{y}_{0:k_P}^5)$
$p(RUL_{k_P}^5 | \mathbf{y}_{0:k_P}^5)$
$p(EOL_{k_P} | \mathbf{y}_{0:k_P})$
$p(RUL_{k_P} | \mathbf{y}_{0:k_P})$

Example: Rover

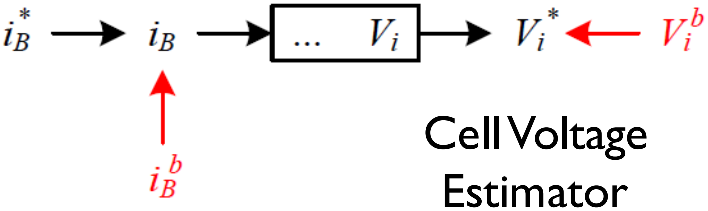
- Estimation
  - One local estimator per cell, taking measured battery current as input and estimating cell voltage
- Prediction
  - Use load power as an input for prediction, since for a given motor speed power is constant but current changes with battery voltage
  - If cells are balanced in voltage, then current split evenly between parallel sets of cells, and can have local predictors for each cell
  - Otherwise (in general), the prediction problem cannot be decomposed, because the current input to each cell depends on the voltages of the other cells



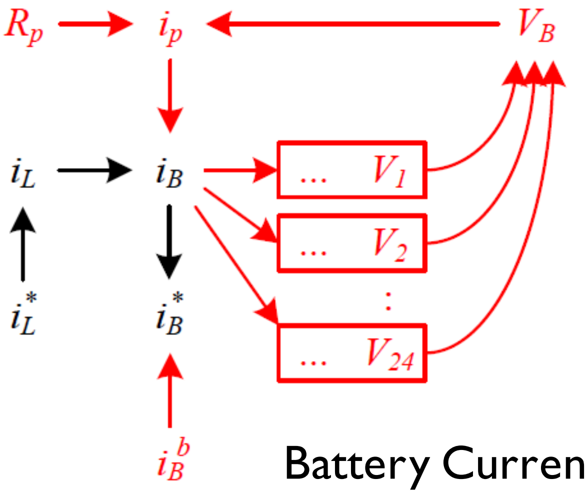
Example: Rover EPS Estimation



Global Nominal Model

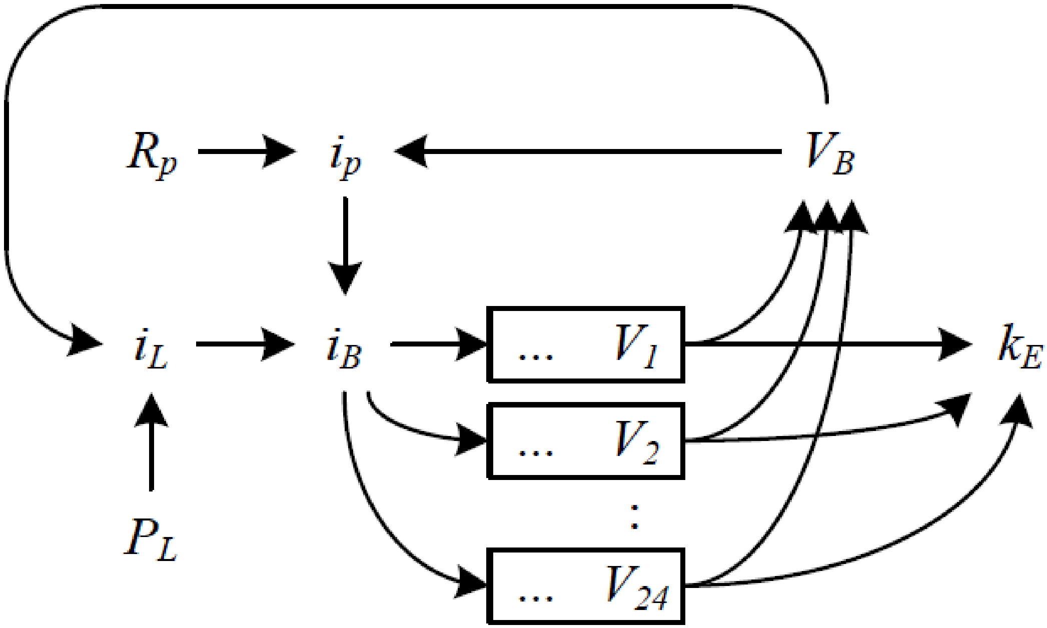


Cell Voltage Estimator

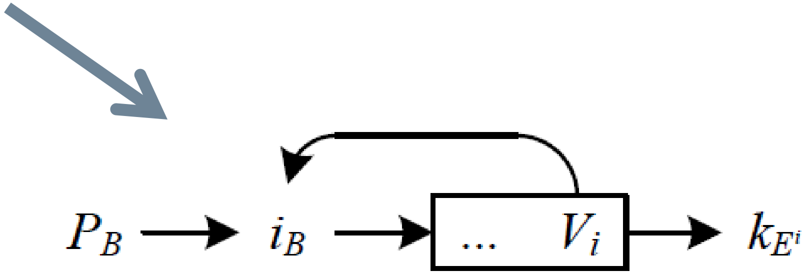


Battery Current Estimator

Example: Rover EPS Prognosis



Global Prediction Model



Local Prediction Submodel

Putting It All Together

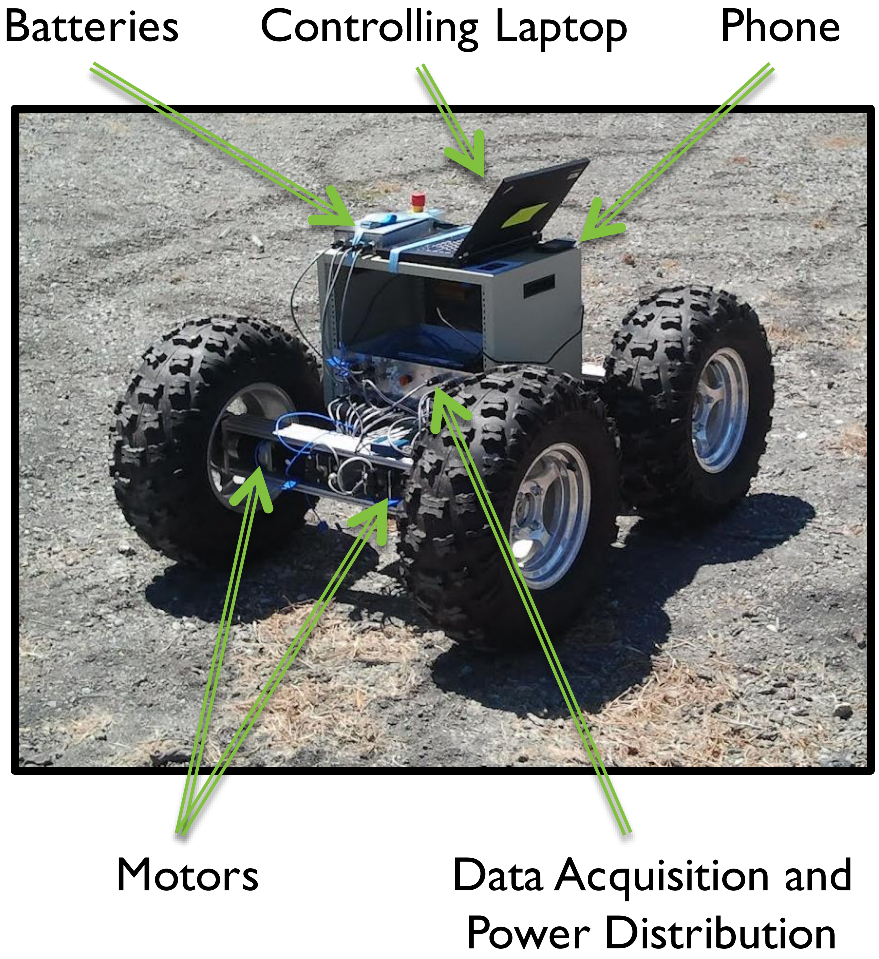
How does prognostics fit into an integrated systems health management architecture?

Prognostics & Decision-Making

- We employ prognostics in order to inform some type of action
- Autonomous vehicles like UAVs and rovers receive command sequences from humans
  - E.g., as a set of waypoints with scientific objectives to achieve at each
- Unexpected situations can cause the vehicle to go into a safe mode while engineers diagnose the problem, which might take a long time
- An autonomous decision-making system that includes automated diagnosis and prognosis in making optimal decisions can save time, money, and increase mission value

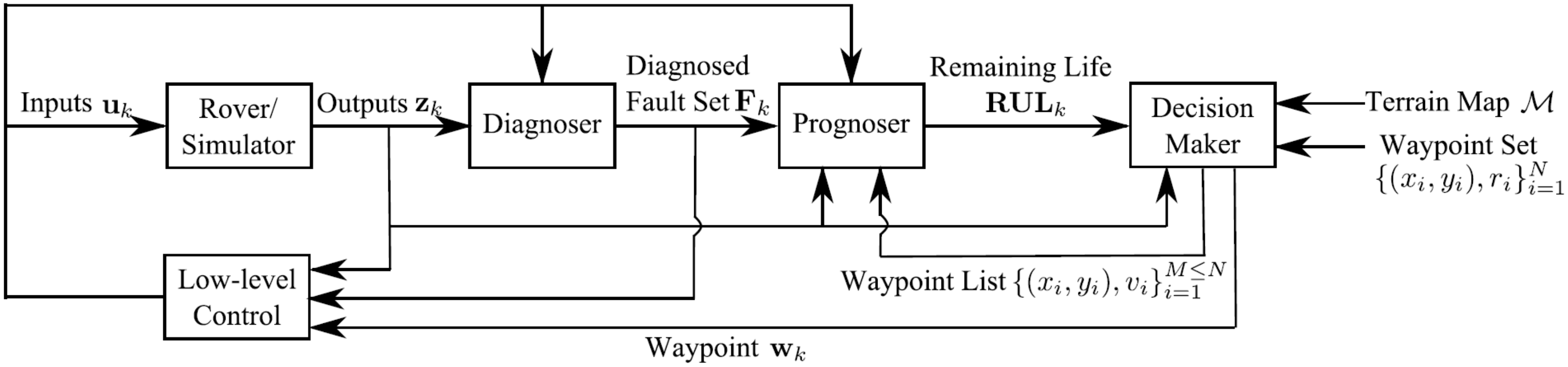
Example: Rover Testbed

- Developed rover testbed for hardware-in-the-loop testing and validation of control, diagnosis, prognosis, and decision-making algorithms
- Skid-steered rover (1.4x1.1x0.63 m) with each wheel independently driven by a DC motor
- Two parallel lithium-ion battery packs (12 cells in series) provide power to the wheels
- Separate battery pack powers the data acquisition system
- Onboard laptop implements control software
- Flexible publish/subscribe network architecture allows diagnosis, prognosis, decision-making to be implemented in a distributed fashion



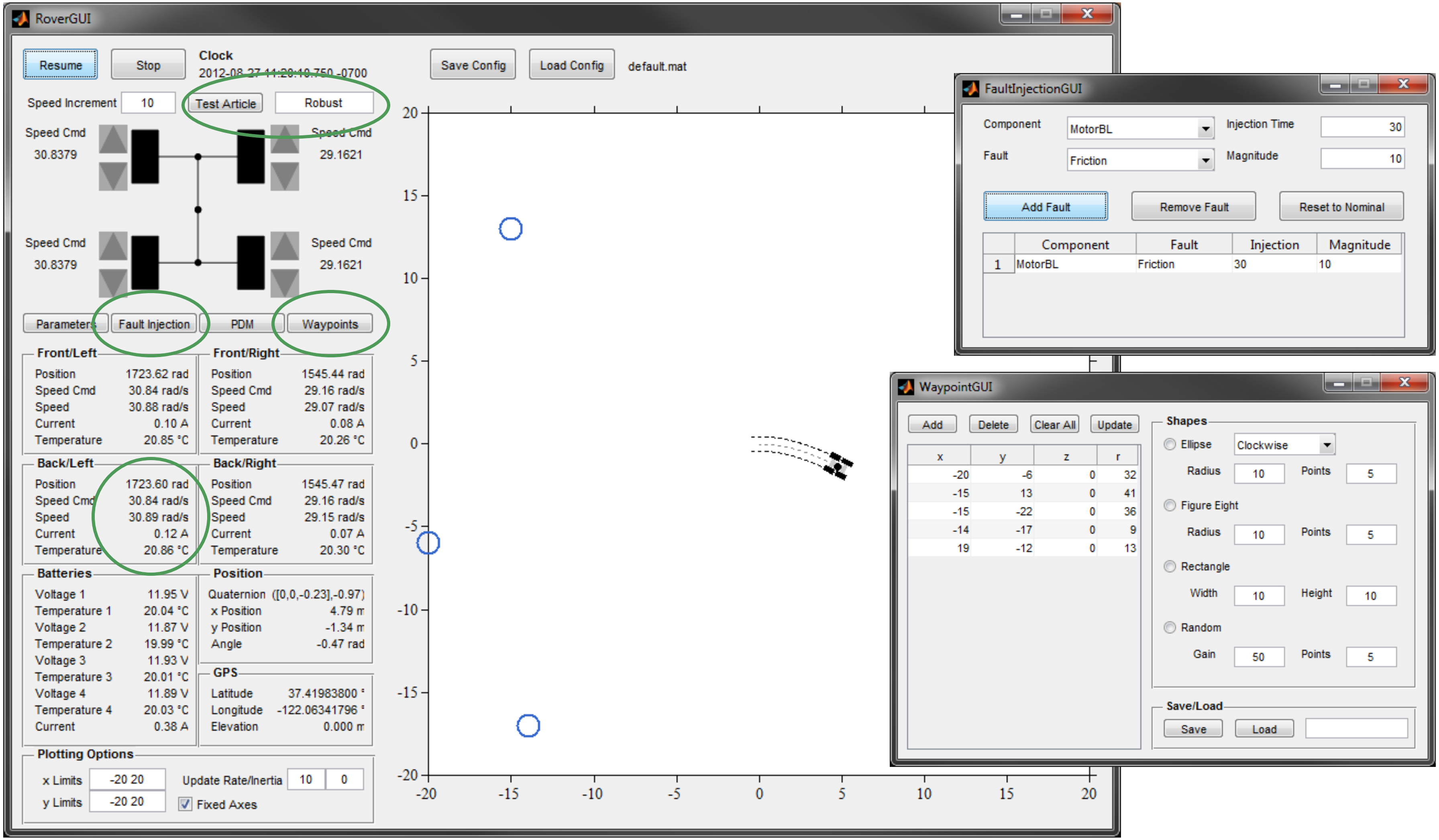
Example: Integrated Architecture

1. Rover receives control inputs (individual wheel speeds) and sensors produce outputs
2. Low-level control modifies wheel speed commands to move towards a given waypoint in the presence of diagnosed faults
3. Diagnoser receives rover inputs and outputs and produces fault candidates
4. Prognoser receives rover inputs and outputs and predicts remaining useful life (RUL) or rover and/or its components (eg, batteries, motors)
5. Decision maker plans the order to visit the waypoints (science objectives) given diagnostic and prognostic information. It can also selectively eliminate some of the waypoints if all of them are not achievable due to vehicle health or energy constraints.



Inputs $\mathbf{u}_k$
Rover/ Simulator
Outputs $\mathbf{z}_k$
Diagnoser
Diagnosed Fault Set $\mathbf{F}_k$
Prognoser
Remaining Life $\mathbf{RUL}_k$
Decision Maker
Terrain Map $\mathcal{M}$
Waypoint Set $\{(x_i, y_i), r_i\}_{i=1}^{N}$
Low-level Control
Waypoint List $\{(x_i, y_i), v_i\}_{i=1}^{M \leq N}$
Waypoint $\mathbf{w}_k$

Example: Simulation Testbed



Clock
2012-08-27 11:20:10.750 -0700

default.mat

Speed Increment 10

Robust

Speed Cmd 30.8379

Speed Cmd 29.1621

Speed Cmd 30.8379

Speed Cmd 29.1621

Front/Left

Position	1723.62 rad
Speed Cmd	30.84 rad/s
Speed	30.88 rad/s
Current	0.10 A
Temperature	20.85 °C

Front/Right

Position	1545.44 rad
Speed Cmd	29.16 rad/s
Speed	29.07 rad/s
Current	0.08 A
Temperature	20.26 °C

Back/Left

Position	1723.60 rad
Speed Cmd	30.84 rad/s
Speed	30.89 rad/s
Current	0.12 A
Temperature	20.86 °C

Back/Right

Position	1545.47 rad
Speed Cmd	29.16 rad/s
Speed	29.15 rad/s
Current	0.07 A
Temperature	20.30 °C

Batteries

Voltage 1	11.95 V
Temperature 1	20.04 °C
Voltage 2	11.87 V
Temperature 2	19.99 °C
Voltage 3	11.93 V
Temperature 3	20.01 °C
Voltage 4	11.89 V
Temperature 4	20.03 °C
Current	0.38 A

Position

Quaternion	([0,0,-0.23],-0.97)
x Position	4.79 m
y Position	-1.34 m
Angle	-0.47 rad

GPS

Latitude	37.41983800 °
Longitude	-122.06341796 °
Elevation	0.000 m

Plotting Options

x Limits -20 20

y Limits -20 20

Update Rate/Inertia 10 0

☑ Fixed Axes

Component MotorBL

Fault Friction

Injection Time 30

Magnitude 10

	Component	Fault	Injection	Magnitude
1	MotorBL	Friction	30	10

x	y	z	r
-20	-6	0	32
-15	13	0	41
-15	-22	0	36
-14	-17	0	9
19	-12	0	13

Shapes

○ Ellipse Clockwise

Radius 10 Points 5

○ Figure Eight

Radius 10 Points 5

○ Rectangle

Width 10 Height 10

○ Random

Gain 50 Points 5

Save/Load

Example Demo

- Demonstration…
  - Fault diagnosis: determining which faults are present
  - Prognosis: predicting remaining driving time
  - Decision-making: mission replanning

Conclusions

Summary
Bibliography
Additional Sources of Information
Acknowledgements

Summary

- Model-based prognostics is a growing research area consisting of several problems
  - Model building
  - Estimation
  - Prediction
  - Uncertainty quantification
  - System-level and distributed prognostics
  - Integration with diagnosis & decision-making
- Goal has been to develop formal mathematical framework, and a modular architecture where algorithms can easily be substituted for newer, better algorithms

Selected Bibliography

- M. Daigle, A. Bregon, and I. Roychoudhury, "Distributed Prognostics Based on Structural Model Decomposition," *IEEE Transactions on Reliability*, vol. 63, no. 2, pp. 495-510, June 2014.
- S. Sankararaman, M. Daigle, and K. Goebel, "Uncertainty Quantification in Remaining Useful Life Prediction using First-Order Reliability Methods," *IEEE Transactions on Reliability*, vol. 63, no. 2, pp. 603-619, June 2014.
- M. Daigle and C. Kulkarni, "A Battery Health Monitoring Framework for Planetary Rovers," *2014 IEEE Aeros*
- M. Daigle and C. Kulkarni, "Electrochemistry-based Battery Modeling for Prognostics," *Annual Conference of the Prognostics and Health Management Society 2013*, pp. 249-261, October 2013.*pace Conference*, March 2014.
- M. Daigle and S. Sankararaman, "Advanced Methods for Determining Prediction Uncertainty in Model-Based Prognostics with Application to Planetary Rovers," *Annual Conference of the Prognostics and Health Management Society 2013*, pp. 262-274, October 2013.
- E. Balaban, S. Narasimhan, M. Daigle, I. Roychoudhury, A. Sweet, C. Bond, and G. Gorospe, "Development of a Mobile Robot Test Platform and Methods for Validation of Prognostics-Enabled Decision Making Algorithms," *International Journal of Prognostics and Health Management*, vol. 4, no. 1, May 2013.
- M. Daigle and K. Goebel, "Model-based Prognostics with Concurrent Damage Progression Processes," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 43, no. 4, pp. 535-546, May 2013.
- I. Roychoudhury, M. Daigle, A. Bregon, and B. Pulido, "A Structural Model Decomposition Framework for Systems Health Management," *Proceedings of the 2013 IEEE Aerospace Conference*, March 2013.
- M. Daigle, A. Bregon, and I. Roychoudhury, "A Distributed Approach to System-Level Prognostics," *Annual Conference of the Prognostics and Health Management Society 2012*, pp. 71-82, September 2012.
- A. Bregon, M. Daigle, and I. Roychoudhury, "An Integrated Framework for Model-based Distributed Diagnosis and Prognosis," *Annual Conference of the Prognostics and Health Management Society 2012*, pp. 416-426, September 2012.
- I. Roychoudhury and M. Daigle, "An Integrated Model-Based Diagnostic and Prognostic Framework," *Proceedings of the 22nd International Workshop on Principles of Diagnosis*, pp. 44-51, October 2011.
- M. Daigle, I. Roychoudhury, S. Narasimhan, S. Saha, B. Saha, and K. Goebel, "Investigating the Effect of Damage Progression Model Choice on Prognostics Performance," *Proceedings of the Annual Conference of the Prognostics and Health Management Society 2011*, pp. 323-333, September 2011.
- M. Daigle and K. Goebel, "A Model-based Prognostics Approach Applied to Pneumatic Valves," *International Journal of Prognostics and Health Management*, vol. 2, no. 2, August 2011.
- Orchard, M., Cerda, M., Olivares, B., and Silva, J., "Sequential Monte Carlo Methods for Discharge Time Prognosis in Lithium-Ion Batteries," International Journal of Prognostics and Health Management, Vol. 3, Issue 2 (010), pp. 1-12, 2012.
- Orchard, M., Tobar, F., and Vachtsevanos, G., "Outer Feedback Correction Loops in Particle Filtering-based Prognostic Algorithms: Statistical Performance Comparison," Studies in Informatics and Control, vol. 18, Issue 4, pp. 295-304, December 2009.
- Orchard, M. and Vachtsevanos, G., "A Particle Filtering Approach for On-Line Fault Diagnosis and Failure Prognosis," Transactions of the Institute of Measurement and Control, vol. 31, no. 3-4, pp. 221-246, June 2009.
- B. Saha and K. Goebel, "Modeling Li-ion battery capacity depletion in a particle filtering framework," in Proceedings of the Annual Conference of the Prognostics and Health Management Society 2009, Sept. 2009.
- J. Luo, K. R. Pattipati, L. Qiao, and S. Chigusa, "Model-based prognostic techniques applied to a suspension system," IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans, vol. 38, no. 5, pp. 1156 – 1168, Sept. 2008.
- B. Saha and K. Goebel, "Model adaptation for prognostics in a particle filtering framework," International Journal of Prognostics andHealth Management, vol. 2, no. 1, 2011.
- A. Saxena, J. Celaya, B. Saha, S. Saha, and K. Goebel, "Metrics for offline evaluation of prognostic performance," International Journal of Prognostics and Health Management, vol. 1, no. 1, 2010.
- E. Zio and G. Peloni, "Particle filtering prognostic estimation of the remaining useful life of nonlinear components," Reliability Engineering & System Safety, vol. 96, no. 3, pp. 403–409, 2011.
- M. Roemer, C. Byington, G. Kacprzynski, and G. Vachtsevanos, "An overview of selected prognostic technologies with reference to an integrated PHM architecture," in Proceedings of the First International Forum on Integrated System Health Engineering and Management in Aerospace, 2005.
- C. S. Kulkarni, J. R. Celaya, G. Biswas, and K. Goebel, "Physics Based Degradation Models for Capacitor Prognostics under Thermal Overtress Conditions," International Journal of Prognostics and Health Management, Vol. 4 No. 1, 2013.

Additional Information Sources

- Some Conferences
  - Annual Conference of the Prognostics and Health Management Society
    - http://www.phmsociety.org/
  - IEEE Aerospace Conference
    - http://www.aeroconf.org/
  - IFAC SAFEPROCESS
    - http://safeprocess15.sciencesconf.org/
  - MFPT
    - http://www.mfpt.org/MFPT2015/MFPT2015.htm
  - IEEE AUTOTESTCON
    - http://ieee-autotest.com/
- Some Journals
  - International Journal on Prognostics and Health Management
    - http://www.phmsociety.org/journal
  - IEEE Transactions on Reliability
    - http://rs.ieee.org/transactions-on-reliability.html
  - IEEE Transactions on Systems, Man, and Cybernetics
    - http://www.ieeesmc.org/publications

Acknowledgements

- Collaborators
  - Kai Goebel, NASA Ames Research Center
  - Indranil Roychoudhury, SGT, Inc., NASA Ames Research Center
  - Anibal Bregon, University of Valladolid
  - Chetan Kulkarni, SGT, Inc., NASA Ames Research Center
  - Shankar Sankararaman, SGT, Inc., NASA Ames Research Center
  - Abhinav Saxena, SGT, Inc., NASA Ames Research Center
  - José Celaya, SGT, Inc., NASA Ames Research Center
  - Brian Bole, SGT, Inc., NASA Ames Research Center
  - Christopher Teubert, SGT, Inc., NASA Ames Research Center
  - Sriram Narasimhan, UC Santa Cruz, NASA Ames Research Center
  - Edward Balaban, NASA Ames Research Center
  - Adam Sweet, NASA Ames Research Center
  - George Gorospe, SGT, Inc., NASA Ames Research Center
  - Belarmino Pulido, University of Valladolid
- Funding Projects
  - NASA System-wide Safety and Assurance Technologies Project
  - NASA Autonomous Cryogenic Loading Operations Project
  - NASA Advanced Ground Systems Maintenance Project
  - NASA Integrated Ground Operations Demonstration Unit Project
  - NASA Fault Detection Isolation and Recovery Project