INTRODUCTION

• NASA
  – Aeronautics & Space
• Ames Research Center
  – Aeronautics research
  – Mission operations (Kepler, LADEE)
  – Synthetic biology
  – Quantum computing
• Intelligent Systems Division
  – Robotics
  – Automatic control
  – Planning & scheduling
  – Collaborative systems
  – Systems health
• Discovery & Systems Health Technical Area
  – Diagnostics & Prognostics Group
  – Prognostics Center of Excellence
OUTLINE

• Part I: Fundamentals of Model-Based Prognostics
  – What is prognostics?
  – Why prognostics?
  – Problem formulation
  – Prognostics architecture

• Part II: Algorithms & Advanced Concepts
  – Modeling approaches
  – Estimation algorithms
  – Prediction algorithms
  – Uncertainty quantification
  – Distributed prognostics
  – Prognostics and decision making
Part I

Fundamentals of Model-Based Prognostics
What is Prognostics?

- Prognostics is the process in which the occurrence of some system event is predicted, e.g.,
  - Predict when/if a mission objective is achieved
  - Predict when a component no longer meets desired functional requirements
- With knowledge of when the event will occur, we can take some sort of action to optimize operations
Systems Health Management

• Prognostics is often viewed as a fundamental technology within system health management (SHM)
  – Prognostics can be informed by diagnostics, and feeds into health-based decision making
• In the SHM context, we predict end of life (EOL) and remaining useful life (RUL) of systems and components
• EOL occurs when functional requirements are no longer met, e.g.,
  – Battery end of discharge
  – Valve fails to open/close
  – Degraded pump efficiency
**Example: Batteries**

- Batteries are ubiquitous – laptops, mobile phones, electric cars, electric aircraft
- 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
- Such predictions can be very useful, but this is, in general, a difficult problem
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 enables, e.g.,
  – 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
What is Model-Based Prognostics?

• Prognostics approaches often differentiated between “model-based” and “data-driven”
  – These terms are not very useful… all approaches use models of some kind, and all are driven by data
  – “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)

• Our definition of 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

• Models are typically developed from a mix of system knowledge and system data

• Models are typically adapted online in some fashion
Problem Formulation

• System described by

\[ x(k + 1) = f(k, x(k), \theta(k), u(k), v(k)) \]
\[ y(k) = h(k, x(k), \theta(k), u(k), n(k)) \]

– \( x \): states, \( \theta \): parameters, \( u \): inputs, \( y \): outputs, \( v \): process noise, \( n \): sensor noise

• Define system event of interest \( E \)

• Define threshold function, that evaluates to true when \( E \) has occurred

\[ T_E(x(k), \theta(k), u(k)) \]
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 $A$, eventually evolves to some new state at which $E$ occurs and moves to region $B$
- $T_E$ defines the boundary between $A$ and $B$
- Must predict the time of event $E$, $k_E$, and the time until event $E$, $\Delta k_E$
**Problem Formulation**

- Define $k_E$
  
  $$k_E(k_P) \triangleq \inf\{k \in \mathbb{N} : k \geq k_P \land T_E(x(k), \theta(k), u(k)) = 1\}$$

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

- May also be interested in the values of some system variables at $k_E$
  
  $$z(k) = \psi(k, x(k), \theta(k), u(k))$$
  
  $$z_E(k_P) \triangleq z(k_E(k_P))$$
  
  $$\Delta z_E(k_P) = z_E(k_P) - z(k_P)$$

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

- System actually takes one path out of many possible paths to region $B$
  - 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
  - Adds complexity to the problem
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

True $p(k_E)$ (what algorithm should produce)

Predicted $p(k_E)$ (what algorithm actually produces)
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
Example: Batteries

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

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

Algorithms & Advanced Concepts
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:
\[ \text{Li}_{1-n}\text{CoO}_2 + n\text{Li}^+ + ne^- \rightarrow \text{LiCoO}_2 \]
Oxidation at neg. electrode:
\[ \text{Li}_n\text{C} \rightarrow n\text{Li}^+ + ne^- + \text{C} \]
Current flows + to –
Electrons flow – to +
Lithium ions flow – to +

**Charge**
Positive electrode is anode
Negative electrode is cathode
Oxidation at pos. electrode:
\[ \text{LiCoO}_2 \rightarrow \text{Li}_{1-n}\text{CoO}_2 + n\text{Li}^+ + ne^- \]
Reduction at neg. electrode:
\[ n\text{Li}^+ + ne^- + \text{C} \rightarrow \text{Li}_n\text{C} \]
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 $V < V_{EOD}$
BATTERY MODEL VALIDATION

“Open-Circuit” Discharge Curve

Nominal 2A Discharge Curve

Rover Battery Discharge Curve

Model matches well for open-circuit, nominal discharge, and variable-load discharges on the rover.

Spike in predicted voltage is due to gap in data (incorrect sample time being fed to model).
**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 “$q^{\text{max}}$” parameter, used to compute mole fraction
  - Modeled with increase in “$R_o$” parameter capturing lumped resistances

---

**Simulated**

**Measured**

---

© 2014 NASA Ames Research Center :: Prognostics Center of Excellence
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.
- Number of sigma points is linear in the size of the state dimension.

\[
\begin{align*}
\bar{x} & \quad P_{xx} \\
\mathbf{x} & \quad \text{Unscented transform} \quad \mathbf{x}'
\end{align*}
\]

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

\[
\mathbf{x}' = \begin{cases} 
\bar{x}, & i = 0 \\
\bar{x} + (\sqrt{(n_x + \kappa)P_{xx}})^i, & i = 1, \ldots, n_x \\
\bar{x} - (\sqrt{(n_x + \kappa)P_{xx}})^i, & i = n_x + 1, \ldots, 2n_x
\end{cases}
\]

Symmetric Unscented Transform
Unscented Kalman Filter

- Kalman filter equations extended to use sigma points

Prediction Step

\[
\hat{\dot{x}}_{k|k-1} = f(\hat{x}_{k-1|k-1}, u_{k-1}), i = 1, \ldots, n_s \\
\hat{y}_{k|k-1} = h(\hat{x}_{k|k-1}), i = 1, \ldots, n_s \\
\hat{x}_{k|k-1} = \sum_{i} w_i \hat{x}_{i|k-1} \\
\hat{y}_{k|k-1} = \sum_{i} w_i \hat{y}_{i|k-1} \\
P_{k|k-1} = Q + \sum_{i} w_i (\hat{x}_{i|k-1} - \hat{x}_{k|k-1})(\hat{x}_{i|k-1} - \hat{x}_{k|k-1})^T.
\]

Update Step

\[
P_{yy} = R + \sum_{i} w_i (\hat{y}_{i|k-1} - \hat{y}_{k|k-1})(\hat{y}_{i|k-1} - \hat{y}_{k|k-1})^T \\
P_{xy} = \sum_{i} w_i (\hat{x}_{i|k-1} - \hat{x}_{k|k-1})(\hat{y}_{i|k-1} - \hat{y}_{k|k-1})^T \\
K_k = P_{xy} P_{yy}^{-1} \\
\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k (y_k - \hat{y}_{k|k-1}) \\
P_{k|k} = P_{k|k-1} - K_k P_{yy} K_k^T
\]

- 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):
  \[ \{ x_k^i, w_k^i \}_{i=1}^N \]
  - Suboptimal, but approach optimality as \( N \to \infty \)
- Approximates posterior as
  \[
  p(x_k|y_{0:k}) \approx \sum_{i=1}^N w_k^i \delta_{x_k^i}(dx_k)
  \]
**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: \{x_{k-1}^i, w_{k-1}^i\}_{i=1}^N, u_{k-1:k}, y_k
Outputs: \{x_k^i, w_k^i\}_{i=1}^N
for i = 1 to N do
    x_k^i \sim p(x_k | x_{k-1}^i, u_{k-1})
    w_k^i \leftarrow p(y_k | x_k^i, 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
\{x_k^i, w_k^i\}_{i=1}^N \leftarrow \text{Resample}(\{x_k^i, w_k^i\}_{i=1}^N)
```
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, but for the Daum and UKF filters, 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.
Variance Control

- \( \xi \) 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 (e.g., 50%) until threshold reached (e.g., 60%)
  - Tracking: Control to desired spread (e.g., 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 \( v_\xi \) Adaptation**

Inputs: \( p(x_k; \theta_k | y_{0:k}) \)
State: \( v_{\xi,k-1}, 1 \leftarrow 1 \)
Outputs: \( v_{\xi,k} \)

for all \( j \in \{1, 2, \ldots, n_0\} \) do
  \( v_j \leftarrow \text{RelativeSpread}(p(\theta_k(j) | y_{0:k})) \)
  if \( v_j < t_j(s(j)) \) then
    \( s(j) \leftarrow s(j) + 1 \)
  end if
  \( v_{\xi,k}(j) \leftarrow \xi_{k-1}(j) \left( 1 + P_j(s(j)) \frac{v_j - v_j^*(s(j))}{v_j^*(s(j))} \right) \)
end for

\( v_{\xi,k-1} \leftarrow v_{\xi,k} \)

Proportional control based on error between actual and desired relative spread

Move to next stage when threshold crossed
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 Representation

• To predict $k_E$, need to account for following sources of uncertainty:
  – Initial state at $k_P$: $x(k_P)$
  – Parameter values for $k_P$ to $k_E$: $\Theta_{k_P}$
  – Inputs for $k_P$ to $k_E$: $U_{k_P}$
  – Process noise for $k_P$ to $k_E$: $V_{k_P}$

• 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, $u(k) = c_1k + c_2k^2$, …
**Prognostics Architecture (Revisited)**

1. System receives inputs, produces outputs

2. Estimate current state and parameter values

3. Use surrogate variable distributions

4. Predict probability distributions for $k_E$, $\Delta k_E$
**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** \( k_E(k_P) \) = \( P(x(k_P), \Theta_{k_P}, U_{k_P}, V_{k_P}) \)

1: \( k \leftarrow k_P \)
2: \( x(k) \leftarrow x(k_P) \)
3: while \( T_E(x(k), \Theta_{k_P}(k), U_{k_P}(k)) = 0 \) do
4: \( x(k+1) \leftarrow f(k, x(k), \Theta_{k_P}(k), U_{k_P}(k), V_{k_P}(k)) \)
5: \( k \leftarrow k + 1 \)
6: \( x(k) \leftarrow x(k + 1) \)
7: end while
8: \( k_E(k_P) \leftarrow k \)

**Algorithm 2** \( \{k^{(i)}_E\}_{i=1}^N = MC(p(x(k_P), \theta(k_P)|y(k_0:k_P)), p(\lambda_\theta), p(\lambda_u), p(\lambda_v), N) \)

1: for \( i = 1 \) to \( N \) do
2: \( (x^{(i)}(k_P), \theta^{(i)}(k_P)) \sim p(x(k_P), \theta(k_P)|y(k_0:k_P)) \)
3: \( \lambda^{(i)}_\theta \sim p(\lambda_\theta) \)
4: \( \Theta^{(i)}_{k_P} \leftarrow \text{construct}\Theta(\lambda^{(i)}_\theta, \theta^{(i)}(k_P)) \)
5: \( \lambda^{(i)}_u \sim p(\lambda_u) \)
6: \( U^{(i)}_{k_P} \leftarrow \text{construct}U(\lambda^{(i)}_u) \)
7: \( \lambda^{(i)}_v \sim p(\lambda_v) \)
8: \( V^{(i)}_{k_P} \leftarrow \text{construct}V(\lambda^{(i)}_v) \)
9: \( k^{(i)}_E \leftarrow P(x^{(i)}(k_P), \Theta^{(i)}_{k_P}, U^{(i)}_{k_P}, V^{(i)}_{k_P}) \)
10: 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

\[
\begin{align*}
\bar{x} &
\end{align*}
\]

\[
\tilde{\chi}^i = \begin{cases} 
\bar{x}, & i = 0 \\
\bar{x} + \sqrt{(n_x + \kappa) P_{xx}}, & i = 1, \ldots, n_x \\
\bar{x} - \sqrt{(n_x + \kappa) P_{xx}}, & i = n_x + 1, \ldots, 2n_x 
\end{cases}
\]

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

**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.

![Graphs showing RUL predictions and ranges for 10 samples and 100 samples](attachment:image.png)

**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

Uncertainty is reduced because use knowledge of future waypoints and speeds
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)
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 $T_{E_i}$. $T_E$ can be composed from the $T_{E_i}$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 = \((X, \theta, U, Y, C)\), set of states \(X\), parameters \(\theta\), inputs \(U\), outputs \(Y\), constraints \(C\)
- Submodel = \((X_i, \theta_i, U_i, Y_i, C_i)\), set of states \(X_i\), parameters \(\theta_i\), 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), …
Model Decomposition Algorithm

- Unobserved State variables: \( x_1, x_2, x_3 \)
- Observed variables
  - Inputs: \( u_1 \)
  - Outputs: \( f_1, f_{12}, f_2, f_{23}, f_3 \)

Let’s assume \( f_{23} \) is not observed.
System receives inputs, produces outputs

1

Estimate state of local submodel

2

Merge local estimates

3

Predict local EOL and RUL as probability distributions

4

Merge local EOL/RUL into global EOL/RUL

5

Distributed Prog. Architecture

System

\[ u_k \rightarrow y_k \rightarrow \begin{align*}
\text{Estimation} & \quad \text{Prediction} \\
M_1 \text{ Estimation} & \quad M_1 \text{ Prediction} \\
p(x_k^1, \theta_k^1 | y_{0:k}^1) & \quad p(EOL_{k_1}^1 | y_{0:k_1}^1) \quad p(RUL_{k_1}^1 | y_{0:k_1}^1) \\
M_2 \text{ Estimation} & \\
p(x_k^2, \theta_k^2 | y_{0:k}^2) & \\
M_3 \text{ Estimation} & \\
p(x_k^3, \theta_k^3 | y_{0:k}^3) & \\
M_4 \text{ Estimation} & \\
p(x_k^4, \theta_k^4 | y_{0:k}^4) & \\
\end{align*} \]

\[ u_k^2 \rightarrow y_k^2 \rightarrow \begin{align*}
M_2 \text{ Estimation} & \quad M_2 \text{ Prediction} \\
p(x_k^2, \theta_k^2 | y_{0:k}^2) & \quad p(EOL_{k_2}^2 | y_{0:k_2}^2) \quad p(RUL_{k_2}^2 | y_{0:k_2}^2) \\
M_3 \text{ Estimation} & \\
p(x_k^3, \theta_k^3 | y_{0:k}^3) & \\
M_4 \text{ Estimation} & \\
p(x_k^4, \theta_k^4 | y_{0:k}^4) & \\
\end{align*} \]

\[ u_k^3 \rightarrow y_k^3 \rightarrow \begin{align*}
M_3 \text{ Estimation} & \quad M_3 \text{ Prediction} \\
p(x_k^3, \theta_k^3 | y_{0:k}^3) & \quad p(EOL_{k_3}^3 | y_{0:k_3}^3) \quad p(RUL_{k_3}^3 | y_{0:k_3}^3) \\
M_4 \text{ Estimation} & \\
p(x_k^4, \theta_k^4 | y_{0:k}^4) & \\
\end{align*} \]

\[ u_k^4 \rightarrow y_k^4 \rightarrow \begin{align*}
M_4 \text{ Estimation} & \quad M_4 \text{ Prediction} \\
p(x_k^4, \theta_k^4 | y_{0:k}^4) & \quad p(EOL_{k_4}^4 | y_{0:k_4}^4) \quad p(RUL_{k_4}^4 | y_{0:k_4}^4) \\
\end{align*} \]
**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
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.
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
- Four lithium-ion battery packs 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
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 (e.g., 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.
**Decision-Making**

- Responsible for determining the ordered set of waypoints that the rover should visit to optimize mission objectives
- **Inputs:**
  - Estimated pose
  - Unordered set of potential waypoints
  - Remaining useful life of batteries, motors, rover
  - Diagnosis result
- **Outputs**
  - Ordered set of waypoints
- **The challenge is in exploring the (potentially) very large search space quickly**
  - Probabilistic Policy Generator (PPG) is based on the principles of Probability Collectives
  - Most conventional optimization methods reason directly over the values $x$ in a search space $X$ in order to minimize an objective function $f(x)$
  - In PPG, reasoning is performed over the probability distribution $P(X)$ that indicates the likelihood of $f(x)$ being minimized in a particular region of $X$.
  - PPG is multi-objective and can work with constraints set for state variables
• Demonstration…
CONCLUSIONS

• 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
REFERENCES


• M. Daigle, A. Bregon, and I. Roychoudhury, "A Distributed Approach to System-Level Prognostics," Annual Conference of the Prognostics and Health