ABSTRACT

Title of Document: COMPUTATIONAL ALGORITHM FOR DYNAMIC HYBRID BAYESIAN NETWORK IN ON-LINE SYSTEM HEALTH MANAGEMENT APPLICATIONS

Chonlagarn Iamsumang, Ph.D., 2014

Directed By: Professor Ali Mosleh, and Professor Mohammad Modarres, Reliability Engineering Program, Department of Mechanical Engineering

With the increasing complexity of today's engineering systems that contain various component dependencies and degradation behaviors, there has been increasing interest in on-line System Health Management (SHM) capability to continuously monitor (via sensors and other methods of observation) system software, and hardware components for detection and diagnostic of safety-critical systems. Bayesian Network (BN) and their extension for time-series modeling known as Dynamic Bayesian Network (DBN) have been shown by recent studies to be capable of providing a unified framework for system health diagnosis and prognosis. BN has many modeling features, such as multi-state variables, noisy gates, dependent failures, and general posterior analysis. BN also allows a compact representation of the temporal and functional dependencies among system components. However, one of the barriers to applying BN in real-world problems is limitation in adequately handle “hybrid models”, which contain both discrete and continuous variables, with both static and time-dependent failure distributions.

This research presents a new modeling approach, computational algorithm, and an example application for health monitoring and learning in on-line SHM. A hybrid DBN is introduced to represent complex engineering systems with underlying physics of failure by modeling a theoretical or empirical degradation model with continuous variables. The methodology is designed to be flexible and intuitive, and scalable from small, localized functionality to large complex dynamic systems. Markov Chain Monte Carlo (MCMC) inference is optimized using a pre-computation strategy and dynamic programming for on-line monitoring of system health. Proposed Monitoring and Anomaly Detection algorithm uses pattern recognition to improve failure detection and estimation of Remaining Useful Life (RUL). Pre-computation inference database enables efficient on-line learning and maintenance decision-making. The scope of this research includes a new modeling approach, computation algorithm, and an example application for on-line SHM.

COMPUTATIONAL ALGORITHM FOR DYNAMIC HYBRID BAYESIAN NETWORK IN ON-LINE SYSTEM HEALTH MANAGEMENT APPLICATIONS

By

Chonlagarn Iamsumang

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park, in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2014

Advisory Committee:
Professor Ali Mosleh, Co-Chair
Professor Mohammad Modarres, Co-Chair
Professor Michel Cukier
Professor Monifa Vaughn-Cooke
Professor Gregory Baecher (Dean's Representative)

© Copyright by

Chonlagarn Iamsumang

2014

To my parents and my brother,

who always believe in me.

You sacrificed so much,

and made it all possible.

Acknowledgements

This dissertation would not have been possible without the support of my professors, colleagues, friends, and family. There are not enough words to describe how grateful and thankful I am for everything they have done for me during these years.

I would like to express my utmost gratitude to my advisors, Dr. Ali Mosleh and Dr. Mohammad Modarres, for their kind guidance, invaluable support, and trust above and beyond what I could have asked for. They have taught me everything about research and I am forever indebted for all the opportunity they have given me.

I would also like to thank my advisory committee, Dr. Gregory Baecher, Dr. Michel Cukier, and Dr. Monifa Vaughn-Cooke, for their helpful feedback and valuable advice throughout the completion of my dissertation.

I am tremendously grateful for the opportunity to work at the Office of Research Administration. I have met some of the most kind and thoughtful people here, and I enjoyed every minute working with them. More importantly, Sally Egloff and Paul Kurt Flicks are the best bosses I have worked with in my life.

I would like to thank my colleagues and classmates, Abdallah Al Tamimi, Elaheh Rabiei, Anahita Imanian, Yuandan Li, Gary Paradee, Victor Ontiveros, Azadeh Keshtgar, Suzi Schroer, Matthew Dennis, Nsimah Ekanem, Mehdi Amiri, Reuel Smith, and Nuhi Faradani. I feel privileged to have studied and worked with such an amazing group of people. Discussing and working together to solve various problems with them in the office was one of the most rewarding and satisfying time of my life.

I am grateful to all my friends who make the journey enjoyable and meaningful. I feel extremely fortunate to have met Claire Kao, Kyle Springer, Pedro Pires, Joo Park, JD Jones, Brooke Okada, Amy Ding, ThSA friends, and Frisbee friends. My thanks to all the great time we had together.

Finally and above all, I would like to thank my family for their love, understanding, and encouragement. Thanks to my parents who taught me the importance of education and sacrificed above and beyond for me to be able to achieve this accomplishment. Thanks to my brother who always be there and be such an example for me to lookup to. Thanks to Pa'Lek for her unconditional love and support since I was a born. This dissertation is dedicated to them.

Table of Contents

Dedication ii
Acknowledgements iii
Table of Contents v
List of Tables vii
List of Figures viii
Chapter 1: Introduction 1
  1.1 Bayesian Network in System Health Management 1
  1.2 Research Objectives 3
  1.3 Scope of this Research 4
  1.4 Organization of this Dissertation 6
Chapter 2: Review of the State of the Art 7
  2.1 Bayesian Network 7
    2.1.1 Definition 8
    2.1.2 Inference in a Bayesian Network 10
    2.1.3 Hybrid Bayesian Network 12
    2.1.4 Dynamic Bayesian Network 13
  2.2 Bayesian Network in Reliability Applications 14
    2.2.1 General reliability 14
    2.2.2 Reliability applications 16
    2.2.3 Bayesian Network in System Health Management 17
    2.2.4 Software 19
Chapter 3: Proposed Modeling Methodology 20
  3.1 Degradation Model for System Health Management 20
    3.1.1 Proposed Hybrid Bayesian Network 20
    3.1.2 Component Critical Parameters 22
    3.1.3 Functional Probability Node Interface 25
  3.2 Dynamic Bayesian Network 29
  3.3 Inference with Markov Chain Monte Carlo 32
    3.3.1 Bayesian inference 32
    3.3.2 Markov Chain Monte Carlo 33
  3.4 Example 38
Chapter 4: Computational Approach 44
  4.1 Inference pre-computation 44
  4.2 Dynamic Programming 49
  4.3 Computation Time Optimization 54
  4.4 Efficient Dependency Algorithm 57
  4.5 Example 62
Chapter 5: System Health Monitoring and Prognosis 72
  5.1 Sensors 72
  5.2 System Monitoring 75
  5.3 Probabilistic Prognosis of Remaining Useful Life 76

5.4 Anomaly Detection 78
5.5 Example 81
Chapter 6: Parameter and Structure Learning 83
6.1 Parameter learning 83
6.2 Structure learning and discovery 86
Chapter 7: Maintenance and Decision Making 88
7.1 Condition-based Maintenance 89
7.2 Risk Informed Decision Making 90
Chapter 8: Example Application 93
8.1 Unmanned Aerial Vehicle (UAV) 93
8.2 System Health Management for UAV 94
8.3 Quadcopter 95
8.4 Technical Specifications 96
8.5 Components and Functionalities 99
8.6 Component Failure Modes and Failure Mechanisms 103
8.7 Bayesian Network Modeling 105
8.8 Sensor Data 111
8.9 Pre-computation 112
8.10 System Monitoring 114
8.11 Remaining Useful Life 116
8.12 Anomaly Detection 117
8.13 Learning 119
8.14 Maintenance 121
Chapter 9: Conclusion 124
9.1 Summary 124
9.2 Contributions 125
9.3 Suggested Future Research 127
Appendices 128
Appendix A: Metropolis-Hasting Algorithm 128
Appendix B: Gibbs sampling 132
Appendix C: Adaptive Rejection Sampling 134
Appendix D: Kalman Filter and Sequential Importance Sampling 137

List of Tables

Table 4-1: Dynamic programming algorithm for pre-computation 52
Table 4-2: Common component factor algorithm 61
Table 4-3: Ranges and distribution parameters of J and T 62
Table 4-4: Ranges and distribution parameters of L and T 69
Table 5-1: RUL estimation algorithm.. 78
Table 8-1: Quadcopter prototype technical specifications.. 98
Table 8-2: Quadcopter components and functionalities 99
Table 8-3: Quadcopter component failure modes... 103
Table 8-4: Failure mode BN and degradation function 107
Table 8-5: Internal sensor data.. 111
Table 8-6: External sensor data... 111

List of Figures

Figure 3-1: Overview of different levels in SHM BN 20
Figure 3-2: SHM Bayesian network at specific time t. 22
Figure 3-3: Typical exponential degradation function. 26
Figure 3-4: Overlap of probability distribution of component status and its threshold. 27
Figure 3-5: Relationship between functionality probability node and its components. 28
Figure 3-6: Two-time-slice representation of a SHM DBN 30
Figure 3-7: Example of different degradation trajectories of C. 31
Figure 3-8: Example BN for an advanced integrated circuit 38
Figure 3-9: BN of EM in the IC example. 40
Figure 3-10: Plot of critical parameter C of the EM example over time. 41
Figure 3-11: Plot of example data of J and T over 100 hours. 42
Figure 3-12: Inference result for component status C with 10,000 iteration samples. 42
Figure 3-13: Reliability probability of Al-strip on EM failure mode over time. 43
Figure 4-1: Pre-computation process 45
Figure 4-2: Example component degradation with 5th percentile, and 95th percentile values. 47
Figure 4-3: Component Gaussian distribution. 48
Figure 4-4: Example cases where dynamic programming reduces number of computations 51
Figure 4-5: Inference pre-computation time with and without dynamic programming. 53
Figure 4-6: Illustrated plots for addition of inference results. 56
Figure 4-7: Proposed BN for SHM. 57
Figure 4-8: BN of components with common factor. 58
Figure 4-9: Proposed BN with an observable common factor. 59
Figure 4-10: Proposed BN with an unobservable common factor 59
Figure 4-11: Probability distribution of the common factor. 60
Figure 4-12: Plots of J and T parameters over all selected percentile values. 63
Figure 4-13: Current density and temperature parameters data set 64
Figure 4-14: Plot of component degradation under electromigration vs. time. 65
Figure 4-15: The difference between results with and without pre-computation method. 65
Figure 4-16: Error of inference result vs. number of parameter selections. 66
Figure 4-17: BN of a component with EM and SM degradations 67
Figure 4-18: Plot of number of computation vs. number of components with the same dependent factor 71
Figure 5-1: Example BN with observable and unobservable nodes. 73
Figure 5-2: Dynamic representation of a variable. 75
Figure 5-3: Timeline comparison for pattern recognition against traditional failure

detection. 80
Figure 5-4: Time difference for failure detection between pattern recognition and traditional method. 82
Figure 6-1: Example BN with a hidden structure. 86
Figure 7-1: Diagram of the proposed SHM process. 88
Figure 8-1: The Aeryon Scout. 97
Figure 8-2: Diagram of components in a quadcopter. 102
Figure 8-3: Quadcopter BN model. 106
Figure 8-4: Electromigration BN 107
Figure 8-5: Stress migration BN. 107
Figure 8-6: TDDB BN. 108
Figure 8-7: HCI BN. 108
Figure 8-8: Corrosion BN. 109
Figure 8-9: Fatigue BN. 109
Figure 8-10: Creep induced failure BN. 110
Figure 8-11: Thermal Expansion BN. 110
Figure 8-12: Example of quadcopter pre-computation results over the ranges of factors 113
Figure 8-13: Quadcopter component reliability with normal operation data (blue) and temperature/humidity cycle data (red). 114
Figure 8-14: Quadcopter functional probabilities with normal operation data (blue) and temperature/humidity cycle data (red). 115
Figure 8-15: Quadcopter system reliability with normal operation data (blue) and temperature/humidity cycle data (red). 116
Figure 8-16: Quadcopter component RUL estimates with 5th and 95th percentile uncertainties. 117
Figure 8-17: Quadcopter frame fatigue model estimations (blue line) and periodic measurements (black dots) with uncertainties. 118
Figure 8-18: Quadcopter frame fatigue model estimations (blue line), periodic measurements (black dots), updated model estimation with measurement error (red line). 119
Figure 8-19: Quadcopter frame fatigue model estimations (blue line), periodic measurements (black dots), updated model estimation with parameter learning (red line). 120
Figure 8-20: Quadcopter frame fatigue model estimations (blue line), periodic measurements (black dots), updated model estimation with structure learning (red line). 121
Figure 8-21: Quadcopter components RUL estimates at t = 0 over a range of operating temperature. 122
Figure 8-22: Quadcopter sensitivity analysis at different time during a normal operating condition. 123

Chapter 1: Introduction

1.1 Bayesian Network in System Health Management

With the increasing complexity of today's engineering systems that contain various component dependencies and degradation behaviors, there has been increasing interest in on-line System Health Management (SHM) capability to continuously monitor (via sensors and other methods of observation) system software, and hardware components for detection and diagnostic of safety-critical systems. The ability to have accurate on-line system monitoring improves maintenance decision-making to reduce cost and avoid possible critical failure. A key requirement for realization of such capability in practice is ability of the modeling and computational framework handle the complexity of component dependencies and failure behaviors, such as sequence-dependent failures and functional dependencies.

Bayesian Network (Pearl, 1986) (Jensen, 2001) and their extension for time-series modeling known as Dynamic Bayesian Network (Murphy K. , 2002) have been shown by recent studies to be capable of providing a unified framework for system health diagnosis and prognosis (Ferreiro, Arnaiz, Sierra, & Irigoien, 2011) (Schumann, Rozier, Reinbacher, Mengshoel, Mbaya, & Ippolito, 2013) (Pourali & Mosleh, 2013). BN has many modeling features, such as multi-state variables, noisy gates, dependent failures, and general posterior analysis (Wilson & Huzurbazar, 2007) (Langseth & Portinale, 2007). BN also allows a compact representation of the temporal and functional dependencies among system components (Boudali & Dugan, 2006).

The main advantages of using BN in system reliability are its simplicity in representing systems and efficiency in obtaining component associations. Another important benefit of BNs is that they enable us to integrate information from different sources, including experimental data, historical data, and prior expert opinion. This feature is particularly useful for the reliability assessment of fault tolerant systems, where failure data from tests and field operations are sparse and obtained from diverse sources of information. BN is particularly well suited for modeling systems that we need to monitor, diagnose, and make predictions about, all in the presence of uncertainty.

However, one of the barriers to applying BN in real-world problems is limitation in adequately handle “hybrid models”, which contain both discrete and continuous variables, with both static and time-dependent failure distributions. Despite the advances in needed methodologies,, applications of BNs as mainstream technology for SHM problems remain limited. To date, the BN framework has only partially addressed these limitations (Lauritzen & Jensen, 2001) (Lerner U. N., 2002) (Shenoy, 2006). For instance the vast majority of BNs used in real world applications are either purely discrete or purely continuous.

For hybrid BNs containing mixtures of discrete and continuous nodes with non-Gaussian distributions, exact inference becomes computationally intractable (Boyen

& Koller, 1998). The common approach to handling (non-Gaussian) continuous nodes is to discretize them using some pre-defined range and intervals (Neil, Tailor, Marquez, Fenton, & Hear, 2007). This is cumbersome, error prone and usually inaccurate.

Even though a universal framework for hybrid BN is currently impracticable, special case algorithms can be effective in SHM where a relatively small subset of possible values covers a large portion of all possible values typically encountered. This paper presents a hybrid BN-based methodology for component degradation modeling and a health monitoring of complex systems.

The approach enables on-line probabilistic diagnosis and prognosis of a system by optimizing Markov Chain Monte Carlo (MCMC) inference with pre-computation and dynamic programming to reduce the computation time and number of inferences required. The pre-computation inference database is then used for efficient health monitoring and system learning.

1.2 Research Objectives

Followings are the objectives of this research:

- Create modeling framework and algorithms for on-line health monitoring of complex systems. The structure must be flexible and intuitive, and can be scaled from localized functionality to a large complex dynamic system. The model is based on physics of failure and empirical results, minimizing probabilistic expert opinion.

- Address and overcome limitations of tracking and diagnosing complex systems with mixtures of discrete and continuous variables where the system dynamics are nondeterministic, not all aspects of the system are directly observed, and the sensors are subject to noise.
- Implement computational algorithm that allows on-line system health monitoring and remaining useful life prediction for efficient maintenance to avoid critical failure. The algorithm should include anomaly detection, parameter learning, and discovery of hidden network structure through continuous monitoring.

1.3 Scope of this Research

This research presents new modeling approach, computational algorithm, and example application for on-line SHM. Its contributions can be summarized into the following main categories:

- Introduce a new modeling approach using hybrid DBN. The model includes:
    a. Identifying and categorizing different layers within SHM BN.
    b. Using hybrid DBN with component-based model to represent complex engineering systems in a way that it allows accurate representation of underlying physics of failure by using empirical degradation model

with continuous variables.

   c. Creating a well-defined interface between continuous system component status and discrete system functionality part of the network.

- Develop computational algorithm for on-line monitoring and diagnosing complex systems. The algorithm includes:
   a. MCMC inference for hybrid DBN.
   b. Inference pre-computation algorithm to allow instantaneous inquiry of system health.
   c. Dynamic programming for MCMC inference to reduce the overall computation time and complexity.
- Implement on-line system monitoring and prognosis from the proposed modeling approach and computational algorithm. This includes:
   a. Monitoring system health and component status to detect any anomaly and predict remaining useful life.
   b. Continuous learning of network parameters and structure from data obtained during operation.
   c. Providing information to improve on-line decision making for system maintenance or in an event that a critical failure occurs
- Demonstrate the capabilities of this methodology by applying it to a real world application.

1.4 Organization of this Dissertation

This dissertation is arranged into the following chapters.

- Chapter 2 presents literature review and past works that are related to the use of BN in reliability and prognosis health management.
- Chapter 3 presents the proposed modeling approach for BN in SHM.
- Chapter 4 presents the computational algorithm, including precomputation and dynamic programming for the proposed model.
- Chapter 5 presents the use of proposed model and computational algorithm for health monitoring, anomaly detection, and prognosis in on-line SHM
- Chapter 6 presents the method for parameter and structure learning of the network during an on-line operation.
- Chapter 7 presents the methodology for decision-making regarding to maintenance and operation management.
- Chapter 8 presents an example in unmanned aerial vehicle application.
- Chapter 9 presents the summary of this research, contributions, and suggested future work.

Chapter 2: Review of the State of the Art

2.1 Bayesian Network

A Bayesian Belief Network, Bayesian Network, or hierarchical Bayesian model is a probability graphical model that represents a set of random variables and their conditional dependencies via a directed acyclic graph (DAG). It consists of a set of interconnected nodes, where each node represents a variable in the dependency model and the connecting arcs represent the causal relationships between these variables. Each node or variable may take one of a number of possible states or values. The belief in, or certainty of, each of these states is determined from the belief in each possible state of every node directly connected to it and its relationship with each of these nodes. The belief in each state of a node is updated whenever the belief in each state of any directly connected node changes.

BN and influence diagrams (IDs) were invented in the mid 1980s (Howard & Matheson, 1984) to represent and reason with large multivariate discrete probability models and decision problems. First major publication on the subject appeared in 1988 (Pearl, 1986). Using exact Inference, Pearl's message passing algorithm messages (probabilities/likelihood) propagate between variables. After finite number of iterations, each node has its correct beliefs. This only works for pure discrete or pure Gaussian and singly connected network where inference is done in linear time.

There are four advantages of using BN: (Heckerman D. , 2008)

One – BNs can readily handle incomplete data sets. When one of the inputs is not observed however most models will produce an inaccurate prediction because they do not encode the correlation between input variables. BNs offer a natural way to encode such dependencies.

Two – BNs allow one to learn about causal relationships. Help gain understanding about the problem domain and allow us to make predictions in the presence interventions

Three – BNs in conjunction with Bayesian statistical techniques facilitate the combination of domain knowledge and data

Four – Bayesian method in conjunction with BNs and other types of models offers an efficient and principled approach for avoiding the over fitting data. All the data can be use for training.

2.1.1 Definition

A Bayesian network for a set of variables $\boldsymbol{x} = \{x_1, \ldots, x_n\}$ consists of

1. a network structure S (direct acyclic graph) that encodes a set of conditional independence assertions about variables in **x**

2. a set P of local probability distributions associated with each variable

The joint probability distribution for **x** is given by

$$p(\boldsymbol{x}) = \prod_{i=1}^{n} p(x_i|\mathbf{pa}_i) \tag{2-1}$$

Where $\mathbf{pa}_i$ denotes the parents of node x_i

From the chain rule of probability

$$p(\boldsymbol{x}) = \prod_{i=1}^{n} p(x_i|\pi_i) \tag{2-2}$$

Where $\pi_i \subseteq \{x_1, \dots, x_{i-1}\}$ such that x_i and $\{x_1, \dots, x_{i-1}\}$ are conditionally independent.

The initial tasks to build a BN are (Heckerman D. , 1995):

1. Correctly identify the goals of modeling
2. Identify many possible observations that may be relevant to the problems
3. Determine what subset of those observations is worthwhile to model
4. Organize the observations into variables having mutually exclusive and collectively exhaustive states

2.1.2 Inference in a Bayesian Network

Because a BN for a given system determines a joint probability distribution for the whole system, in principle we can use the BN to compute any probability of interest. Using the conditional independencies encoded in a Bayesian network, the computation can be made more efficient.

Bayesian inference requires a prior probability distribution for the parameters $P\left(\theta_1, \dots, \theta_\gamma\right)$. The prior could be subjective based on expert opinions or objective based on observed frequencies.

When combine a prior distribution for the parameters with the conditional distribution for the observed data, we get a joint distribution for all quantities related to the problem:

$$P\left(\theta_1, \dots, \theta_\gamma, x_1, x_2, \dots,\right) = P\left(\theta_1, \dots, \theta_\gamma\right) P\left(x_1, x_2, \dots \middle| \theta_1, \dots, \theta_\gamma\right) \tag{2-3}$$

$$P\left(\theta_1, \dots, \theta_\gamma, x_1, x_2, \dots,\right) = P(\boldsymbol{\theta}) \prod_i P(x_i|\boldsymbol{\theta}) \tag{2-4}$$

From this, we can derive Bayes' Rule for the posterior distribution of the parameters, given observed values for $X_1, \dots, X_C$:

$$P(\boldsymbol{\theta}|x_1, \dots, x_C) = \frac{P(\boldsymbol{\theta}|x_1, \dots, x_C)}{P(x_1, \dots, x_C)} = \frac{P(\boldsymbol{\theta}) \prod_{i=1}^{C} P(x_i|\boldsymbol{\theta})}{\int P(\tilde{\theta}) \prod_{i=1}^{C} P(x_i|\tilde{\theta})\, d\tilde{\theta}} \tag{2-5}$$

The posterior can also be expressed as proportionality in terms of the likelihood:

$$P(\boldsymbol{\theta}|x_1, \dots, x_C) \propto P(\boldsymbol{\theta})L(\boldsymbol{\theta}|x_1, \dots, x_C) \qquad (2\text{-}6)$$

This shows how introduction of a prior converts the expression of relative plausibility contained in the likelihood into an actual probability distribution over parameter space.

The Bayesian framework can provide a predictive distribution for an unobserved case, X_{C+1}, given the values observed for $X_1, \dots, X_C$ (Neal, 1993):

$$P(x_{C+1}|x_1, \dots, x_C) = \int P\big(x_{C+1}|\tilde{\theta}\big)P(\tilde{\theta}|x_1, \dots, x_C)\, d\tilde{\theta} \qquad (2\text{-}7)$$

Note that the Bayesian predictive distribution is not based on a single estimate for the parameters, but is instead an average of the predictions using all possible values of the parameters, with each prediction weighted by the probability of the parameters having those values.

Once the BN structure and nodes probability distributions have been defined, reliability analysis can be carried out using standard BN inference algorithms (Lauritzen & Spiegelhalter, 1988) (Jensen, Lauritzen, & Olesen, 1990). Several efficient algorithms exist to compute exact marginals of posterior distributions for discrete BNs and to solve discrete influence diagrams exactly (Shachter, 1986) (Shenoy, 1992). Neapolitan (Neapolitan, 2004) shows that by assuming that every variable follows a Gaussian distribution, the inference process for continuous networks simplifies as it can be shown that a linear combination of Gaussian distributions results in another Gaussian distribution.

As inference in BNs was found to be NP-hard in general (Cooper G. F., 1990), attention was paid to heuristic and stochastic techniques to help solve the problem. It was then found that approximate inference is also NP-hard (Dagum & Luby, 1993). However approximate inference do have wider range of applicability. Some of the most prevalent inexact techniques are based on Monte Carlo methods; the paper of Cousins et al. (Cousins, Chena, & Frisse, 1993) have a short tutorial on the subject in relation to Bayesian network inference, whereas the paper of Dagum and Horvitz (Dagum & Horvitz, 1993) analyses the performance of simulation algorithms using a Bayesian perspective. However, exact inference is generally a computationally intractable problem (Boyen & Koller, 1998).

2.1.3 Hybrid Bayesian Network

The state of the art exact algorithm for mixtures of Gaussians hybrid BNs is the Lauritzen-Jensen (Lauritzen & Jensen, 2001) algorithm. This requires the conditional distributions of continuous variables to be conditional linear Gaussians (CLG), and that discrete variables do not have continuous parents.

If a BN has discrete variables with continuous parents, Murphy (Murphy K. , 1999) uses a variational approach to approximate the product of the potentials associated with a discrete variable and its parents with a CLG. Lerner (Lerner U. N., 2002) uses a numerical integration technique called Gaussian quadrature to approximate non-CLG distributions with CLG, and this same technique can be used to approximate the product of potentials associated with a discrete variable and its continuous parents. Shenoy (Shenoy, 2006) proposes approximating non-CLG distributions by mixtures of Gaussians using a nonlinear optimization technique, and using arc reversals to ensure discrete variables do not have continuous parents. The resulting mixture of Gaussians BN is then solved using Lauritzen-Jensen algorithm.

Moral *et al*. (Moral, Rumi, & Salmeron, 2001) proposes approximating probability density functions (PDFs) by mixtures of truncated exponentials (MTE), which are easy to integrate in closed form. Since the family of MTE is closed under combination and marginalization, the Shenoy-Shafer architecture can be used to solve the MTE BN. Another common method is dynamic discretization algorithm, which is given by Neil (Neil, Tailor, Marquez, Fenton, & Hear, 2007).

2.1.4 Dynamic Bayesian Network

Dynamic Bayesian Network (DBN) framework (Dean & Kanazawa, 1989) allows a compact representation of the temporal (and functional) dependencies among the system components and event-dependent failure behaviors, characteristic of fault-tolerant systems, avoiding the state space explosion problem of the Markov Chain based approaches to Dynamic Fault Tree (DFT) analysis (Bechta Dugan, Bavuso, & Boyd, 1992). The key to DBNs is that they are specified in two parts, a prior

Bayesian network that specifies the initial conditions and a transition Bayesian network that specifies how variables change from time to time.

The most original paper is presented by Dean and Kanazawa (Dean & Kanazawa, 1989) or Freidman et al. (Friedman N. , 1998). In addition Murphy and Mian (Murphy & Mian, 1999) show modeling of data using DBN. Ghahramani (Ghahramani, 1998) examines the topic from the perspective of learning and Flesch and Lucas (Flesch & Lucas, 2007) consider DBNs where the transition network can change over time.

2.2 Bayesian Network in Reliability Applications

Estimation of systems reliability using BN dates back as early as 1988 (Barlow, 1988). The first real attempt to merge the efforts of the two communities is probably the work of Almond (Almond, 1992), where he proposes the use of the graphical belief tool for calculating reliability measures concerning a low-pressure coolant injection system for a nuclear reactor A number of early studies have attempted to use BNs (Pearl, 1993) (Jensen, 2001) to provide a unified framework for reliability modeling and analysis of complex systems. Works on system safety and Bayesian Networks (BNs) were originally developed in (Kang & Golay, 1999).

2.2.1 General reliability

Bobbio et al. (Bobbio, Portinale, Minichino, & Ciancamerla, 2001) show how to improve the analysis of dependable systems by mapping fault trees into Bayesian networks. Singh et al. (Singh, Cortellessa, Cukic, Gunel, & Bharadwaj, 2001) presents their work on reliability estimation in component-based systems. They classify the component-based system reliability estimation methods into three as state-based models, path-based models and additive models.

Murphy (Murphy K. , 2002) introduces DBN to provide a unified framework for reliability modeling and analysis of complex systems. Lerner et al. (Lerner, Parr, Koller, & Biswas, 2000) propose a new approach to this task, based on the framework of hybrid DBN. These models contain both continuous variables representing the state of the system and discrete variables representing discrete changes such as failures; they can model a variety of faults, including burst faults, measurement errors, and gradual drifts.

Mahadevan and Rebba (Mahadevan & Rebba, 2005) propose a methodology based on Bayesian statistics to assess the validity of reliability computational models when full-scale testing is not possible. Sub-module validation results are used to derive a validation measure for the overall reliability estimate. BNs are used for the propagation and updating of validation information from the sub-modules to the overall model prediction.

Boudali and Dugan present two great papers to cover both discrete-time BN reliability modeling and analysis framework (Boudali & Dugan, 2005) and introduce a method for reliability assessment in dynamic systems by using temporal BN

(Boudali & Dugan, 2006). Langseth and Portinale (Langseth & Portinale, 2007) discuss the properties of the modeling framework that make BNs particularly well suited for reliability applications, and point to ongoing research that is relevant for practitioners in reliability.

Wilson and Huzurbazar (Wilson & Huzurbazar, 2007) extend their use to multilevel discrete data and discuss how to make joint inference about all of the nodes in the network. This article extends complex systems modeling using BNs for multilevel discrete data. Weber and Jouffe (Weber & Jouffe, 2006) present a methodology that will help developing Dynamic Object Oriented Bayesian Networks (DOOBNs) to formalize such complex dynamic models that are dynamically modeled and controlled to optimize the diagnosis and the maintenance policies. Recently, Doguc et al. (Doguc & Ramirez-Marquez, 2009) present a holistic method for constructing a Bayesian network (BN) model for estimating system reliability. BN is a probabilistic approach that is used to model and predict the behavior of a system based on observed stochastic events.

2.2.2 Reliability applications

Bayesian network has been applied to many applications during the course of the years as seen in the example papers below:

- Bouissou and Ourghanlian (Bouissou, Martin, & Ourghanlian, 1999) developed assessment of a safety- critical system including software using a

Bayesian belief network for evidence sources.

- Gran and Helminen (Gran & Helminen, 2001) provide a BN structure for nuclear power plants and introduce a hybrid method for estimating the reliability of the plant.
- Wooff et al. (Wooff, Goldstein, & Coolen, 2002) use Bayesian graphical models for software testing.
- Neil et al. (Neil, Fenton, Forey, & Harris, 2003) suggest using Bayesian Networks to access vehicle reliability.
- Helminen and Pulkkinen (Helminen & Pulkkinen, 2003) present a BN-based method for reliability estimation of computer-based motor protection relay.
- Kipersztok and Provan (Kipersztok & Provan, 2003) show framework for diagnostic inference of commercial aircraft systems.
- Fenton et al. (Fenton, Neil, & Marquez, 2008) use Bayesian Networks to predict software defects and reliability.

2.2.3 Bayesian Network in System Health Management

BNs have established themselves as an indispensable tool in artificial intelligence and in the domain of system health management, including diagnosis and prognosis. They are being used effectively by researchers and practitioners more broadly in science and engineering. BNs are particularly well suited to modeling systems that need to be monitored, diagnose, and make predictions about, all under the presence of uncertainty.

BNs provide a simple and natural language for modeling problems in systems health management. Moreover, they support a variety of probabilistic queries, which provide a means to qualitatively and quantitatively reason about system health and reliability. Further, there are a variety of effective and principled approaches to inducing Bayesian networks from data, with or without prior expert knowledge.

DBNs in System Heath Management provide diagnostic and prognostic capabilities. They have shown promise in several recent applications. Dong and Yang (Ming & Yang, 2008) use DBNs combined with particle filtering to estimate the RUL distribution of drill bits in a vertical drilling machine. Tobon-Mejia et al. (Tobon-Mejia, Medjaher, Zerhouni, & Tripot, 2012) use mixtures of Gaussian HMMs (a form of DBN) to estimate the RUL distributions for bearings. The junction tree algorithm is used for exact inference.

The following are a few examples of research works that were done recently in the area of System Health Management.

- Ferreiro et al. (Ferreiro, Arnaiz, Sierra, & Irigoien, 2011) use BN model in prognostics and SHM for aircraft line maintenances. The article presents the global framework of a health management system as a new concept in aircraft line maintenance. This framework allows the transformation of the traditional maintenance (preventive and corrective, time based) into a predictive maintenance based on prognostic techniques.

- Schumann's (Schumann, Rozier, Reinbacher, Mengshoel, Mbaya, & Ippolito, 2013) implementation provides a novel approach of combining modular building blocks, integrating responsive runtime monitoring of temporal logic system safety requirements with model-based diagnosis and Bayesian network-based probabilistic analysis.
- Choi et al. (Choi, Zheng, Darwiche, & Mengshoel, 2011) has recently written a tutorial on BNs for SHM.

2.2.4 Software

Many BN tools are available to the practitioners. Examples of commercial tools available online include Hugin (http://www.hugin.com/), BayesiaLab (http://www.bayesia.com/) and Netica (http:// www.norsys.com/). BUGS (http://www.mrc-bsu. cam.ac.uk/bugs/) is a general-purpose modeling framework where inference is based on simulation.

Other popular alternatives include, UCLA's SamIam (Sensitivity Analysis, Modeling, Inference and More), Kevin Murphy's Bayesian Network Toolbox for use in Matlab computing environments, and the University of Pittsburgh's GeNIe & SMILE system, which includes specialized features for BNs used in diagnostic applications. There are also a variety of commercial systems for modeling and reasoning in BNs.

Chapter 3: Proposed Modeling Methodology

3.1 Degradation Model for System Health Management

3.1.1 Proposed Hybrid Bayesian Network

For SHM modeling, it is advantageous and intuitive to consider a hybrid system, typically with the variables such as time and temperature being modeled as continuous and the system's functionality probability being discrete.

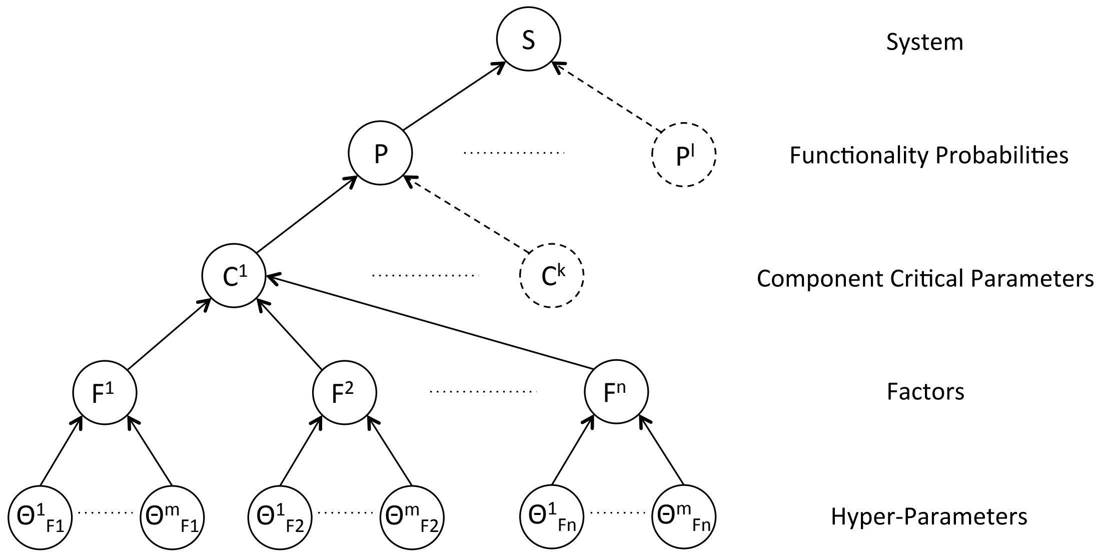


Figure 3-1: Overview of different levels in SHM BN

The proposed complex system hybrid BN can be separated into 5 levels as shown in Figure 3-1, according to the typical characteristics of the nodes. The BN combines high-level functionality nodes with low-level physical variables representing failure.

Here are the descriptions of each level:

1. System node: this is the highest level of nodes with no children. It represents the state of the whole system and usually indicates whether or not the system is working as intended.
2. Functionality probability nodes: these nodes are designed to be abstract discrete nodes that represent various functionalities, which are required for the system to operate.
3. Component critical parameter nodes: these are continuous nodes representing status of physical components and structures susceptible to specific failure mechanisms in the system. These values should be measurable directly or indirectly.
4. Factor nodes: these nodes show contributors to the degradation of the components. They can be component internal factors related to material properties or physical characters, or they can be external factors such as environmental stress or temperature.
5. Hyper-parameter nodes: these nodes are hyper-parameters that describe probability distributions of the factors.

It is to be noted that each level does not have to be only one layer as shown in Figure 3-1, it can be a combination of different layers of nodes that have the same type.

3.1.2 Component Critical Parameters

Reliability concerns arise when some critically important materials or devices degrade with time. Let *C* represent a critically important material/device parameter. This parameter degrades over the life of the component. The value itself can either increase (threshold voltage of a semiconductor device, increase in leakage of a capacitor, increase in resistance of a conductor) or decrease (decrease of pressure in a vessel, decrease of spacing between mechanical components, decrease in lubricating properties of a fluid). Figure 3-2 presents the SHM Bayesian network at specific time t. The shade areas show continuous nodes that are related to each component.

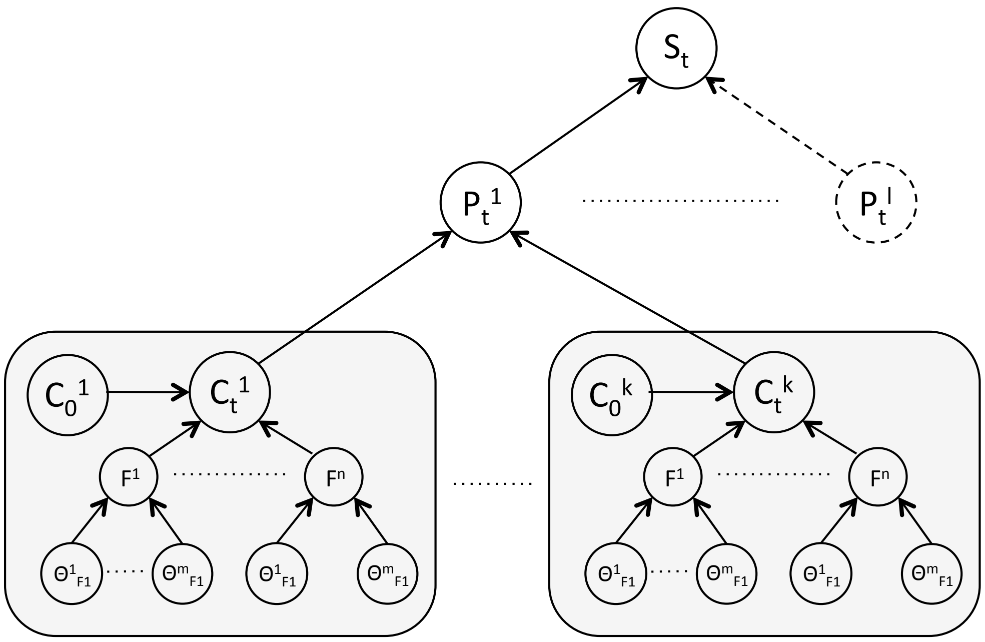


Figure 3-2: SHM Bayesian network at specific time *t*.

An example of a degradation process can be derived from a Taylor expansion about

$t = 0$, which produces the Maclaurin Series, assuming that *C* changes monotonically and relatively slowly over the lifetime of the material/device:

$$C(t) = C_{t=0} + \left(\frac{\partial C}{\partial t}\right)_{t=0} t + \frac{1}{2}\left(\frac{\partial^2 C}{\partial t^2}\right)_{t=0} t^2 + \cdots \tag{3-1}$$

By assuming that the higher order terms in the expansion can be approximated by simply modeling degradation of component/device parameter *C* with a power-law equation:

$$C = C_0[1 \pm A_0 t^m] \tag{3-2}$$

where C_0 is the value of C at $t = 0$, A_0 is material/device-dependent coefficient, and m is the power-law exponent. Both A_0 and m are parameters that can be learned from component/device degradation data. Summation (+) is used when the parameter C increases with time, while subtraction (-) is used when the parameter C decreases with time

The parameter A_0 is generally material/microstructure dependent. It is not only a function of material variations, but also a function of other factors, such as electrical, thermal, mechanical and chemical environments to which the device is exposed.

$$A_0 = A_0(F_1, \dots, F_n) \tag{3-3}$$

Therefore, we have:

$$C = C_0[1 \pm A_0(F_1, \dots, F_n)t^m] \tag{3-4}$$

$$\frac{dC}{dt} = \pm m C_0 A_0(F_1, \dots, F_n)t^{m-1} \tag{3-5}$$

m and other parameters are considered to be constant for the component/device. Considering a Bayesian network at a time slice of a given system, t is then constant and indicates the current life of the component/device.

For a component/device to fail, the amount of degradation must reach a critical value, C_{crit}. Therefore, the time to failure, $T_{failure}$, is then:

$$T_{failure} = \left[\frac{1}{\pm A_0(F_1, \dots, F_n)}\left(\frac{C_{crit} - C_0}{C_0}\right)\right]^{1/m} \tag{3-6}$$

Values of the factors are continuously distributed and parameters of the distribution can be shown explicitly as θ node.

$$F = F(\theta_1, \dots \theta_m) \tag{3-7}$$

There are three general time-dependent models for degradation. The choice of model and the model parameters can be extracted from the observed degradation data.

Power-Law:

$$C = C_0[1 \pm A_0(F_1, \dots, F_n)t^m] \tag{3-8}$$

Exponential:

$$C = C_0[\exp(\pm A_0(F_1, \dots, F_n)t)] \tag{3-9}$$

Logarithmic

$$C = C_0[1 \pm \ln(A_0(F_1, \dots, F_n)t + 1)] \tag{3-10}$$

The choice of model and the model parameters can be extracted from observed degradation data.

3.1.3 Functional Probability Node Interface

Since the component parameter and their parents are continuous nodes, and the functionality probability nodes are discrete, the interface between these different types of nodes becomes critical. In general hybrid BNs, when continuous nodes have discrete parents, there are simple conditional inference techniques such as in conditional linear Gaussian (CLG) model. Difficulty arises when discrete nodes have continuous parents, which is the case for our SHM network. However in this case, even though discrete functionality probability nodes have continuous component status nodes, they are related by degradation thresholds.

Discrete functionality nodes can contain more than two states with thresholds between the transitions of one state to the other. Let the threshold value between functionality states i and j be $C_{th,i/j}$. The most common case would be state i denoting "component function", and state j denoting "component does not function". Let P_i be the probability of functional state being i. The probability P_i is then the probability that the component status C is lower than the threshold value $C_{th,i/j}$. Figure 3-3 shows a hypothetical component exponential degradation function and the overlap of probability distributions of C and $C_{th,i/j}$.

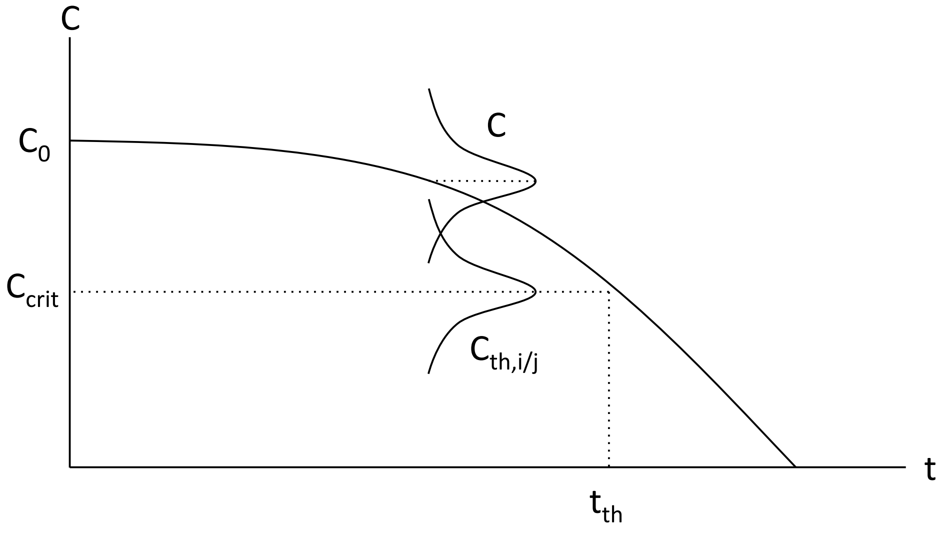


Figure 3-3: Typical exponential degradation function.

Figure 3-4 shows the overlap of probability distributions of C, lower threshold $C_{th,i-1/i}$, and higher threshold $C_{th,i/i+1}$. If the distribution C is moving to the left as the component is degrading, the probability of being in state $i-1$ is increasing, and the probability of being in state $i+1$ is decreasing.

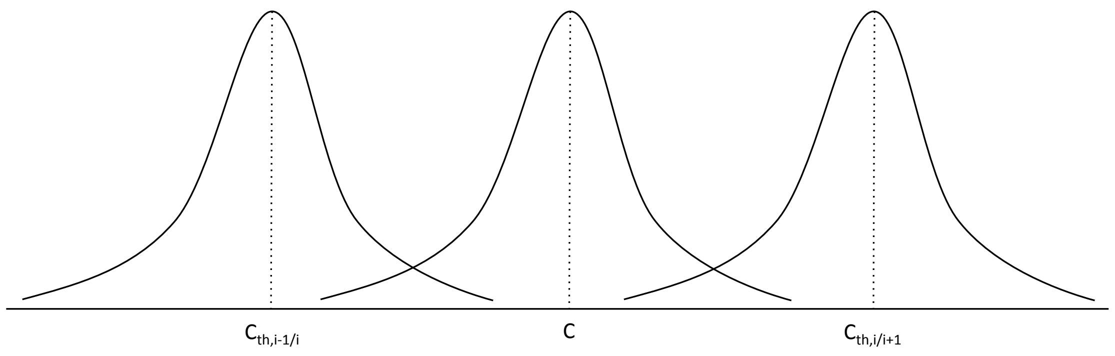


Figure 3-4: Overlap of probability distribution of component status and its threshold.

Let a functionality node have n states, the probabilities of being in these states are $P_1, \dots, P_n$. Assume the state of the functionality node changes monotonically according to the component degradation status:

$$C_{th,i-1/i} < C_{th,i/i+1} \text{ for } i = 2, \dots, n-1 \tag{3-11}$$

Therefore,

$$P_i = prob\left(C > C_{th,i-1/i} \; and \; C < C_{th,i/i+1}\right) \tag{3-12}$$

Analytically, P_i can be calculated from the following convolution equation:

$$P_i = \int_{-\infty}^{C_{th,i/i+1}} \int_{C_{th,i-1/i}}^{\infty} \int_{C_{th,i-1/i}}^{C_{th,i/i+1}} p\left(C_{th,i-1/i}\right) \cdot p(C) \cdot p\left(C_{th,i/i+1}\right) dC dC_{th,i/i+1} dC_{th,i-1/i} \tag{3-13}$$

If there are many component critical parameters contributing to this functionality then the state of the functionality node conditionally depends on comparison between the status of each component and its threshold values.

$$P_{i^1 \dots i^n} = prob\big({C^1}_{th,i^1-1/i^1} < C^1 < {C^1}_{th,i^1/i^1+1}, \dots, {C^n}_{th,i^n-1/i^n} < C^n < {C^n}_{th,i^n/i^n+1}\big) \tag{3-14}$$

For example, the functionality is in functioned state when $C^1 > {C^1}_{th,1/2}$ and $C^2 > {C^2}_{th,1/2}$, but it is in not functioned state when $C^1 > {C^1}_{th,1/2}$ or $C^2 > {C^2}_{th,1/2}$. This conditional probability can be dealt with conditional probability table (CPT) the same way as in discrete Bayesian network.

Probability functionality part of the SHM Bayesian network is shown in Figure 3-5.

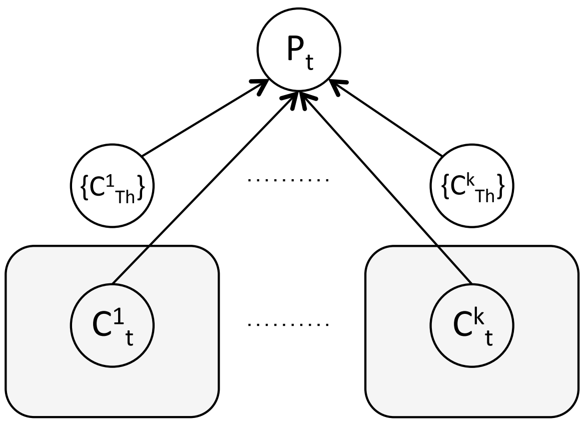


Figure 3-5: Relationship between functionality probability node and its components.

Please note that C_{th} is shown in brackets because it is possible to have more than one threshold values, depending on the number of states of the functionality.

3.2 Dynamic Bayesian Network

DBN is a BN that includes a temporal dimension. This new dimension is managed by time-indexed value t to indicate time stage of the nodes. A set of nodes at certain stage contains random variables relative to time slice t. An arc that links two variables belonging to different time slices represents a temporal probabilistic dependence between these variables. Variables can be modeled to have impact on the future distribution of the other variables. These impacts are defined as transition probabilities between the stats of variables at time step t and $t + \Delta t$.

A DBN describes the joint distribution of a set of variables $\boldsymbol{\theta}$**.** This is a complex distribution, but may be simplified by using the Markov assumption due to slow changing degradation mechanisms in SHM. The Markov assumption requires only the present state of the variables $\boldsymbol{\theta}_t$ to estimate $\boldsymbol{\theta}_{t+1}$, i.e. $p(\boldsymbol{\theta}_{t+1}|\boldsymbol{\theta}_0, \dots, \boldsymbol{\theta}_t) = p(\boldsymbol{\theta}_{t+1}|\boldsymbol{\theta}_t)$ where p indicates a probability density function and bold letters indicate a vector quantity. Additionally, the process is assumed to be stationary, meaning that $p(\boldsymbol{\theta}_{t+1}|\boldsymbol{\theta}_t)$ is independent of t.

For SHM Bayesian network, the main variables that change between time slices are component parameters. Components degrades over time, therefore, the status of components at a certain time slice depend on their status at the previous time slice and the factors affecting the degradation processes during that transition.

$$p(C_t) = p(C|C_{t-\Delta t}, \{F_t^1, \dots, F_t^n\}) \tag{3-15}$$

Given that F_t^i is the average value of factor i between time slice $t - \Delta t$ and t.

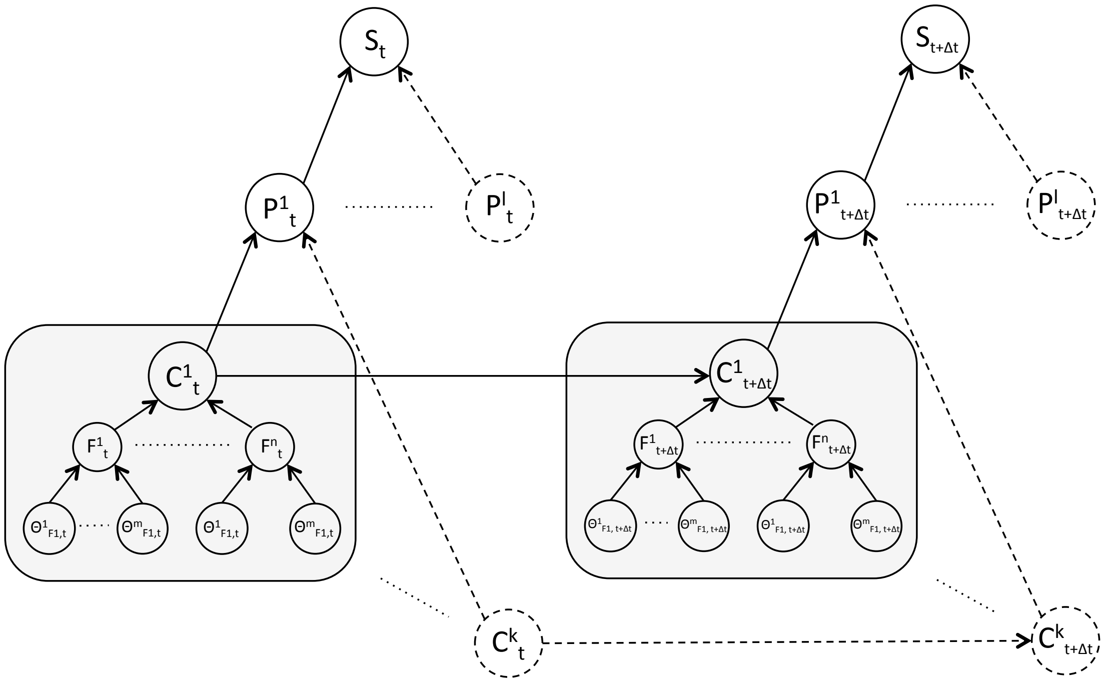


Figure 3-6: Two-time-slice representation of a SHM DBN

Figure 3-6 shows a two-time-slice representation of a dynamic SHM Bayesian network. Δt should be set according to the system under interest and how often the parameters can be observed, such as frequency of sensor signals.

At any point in time during system operation, any value of variables in the system can be derived by probabilistic inference to compare with its expected value to see if the probability is still in the acceptable range and the system as a whole is working as intended. With continuous monitoring, the trajectories of the degradation processes can be estimated form our knowledge of the health of the system. We can then use this information to estimate remaining useful life (RUL) of components and plan maintenance accordingly.

Figure 3-7 shows example of different degradation trajectories of component critical parameter C depending on the conditions and factors of the system during operation.

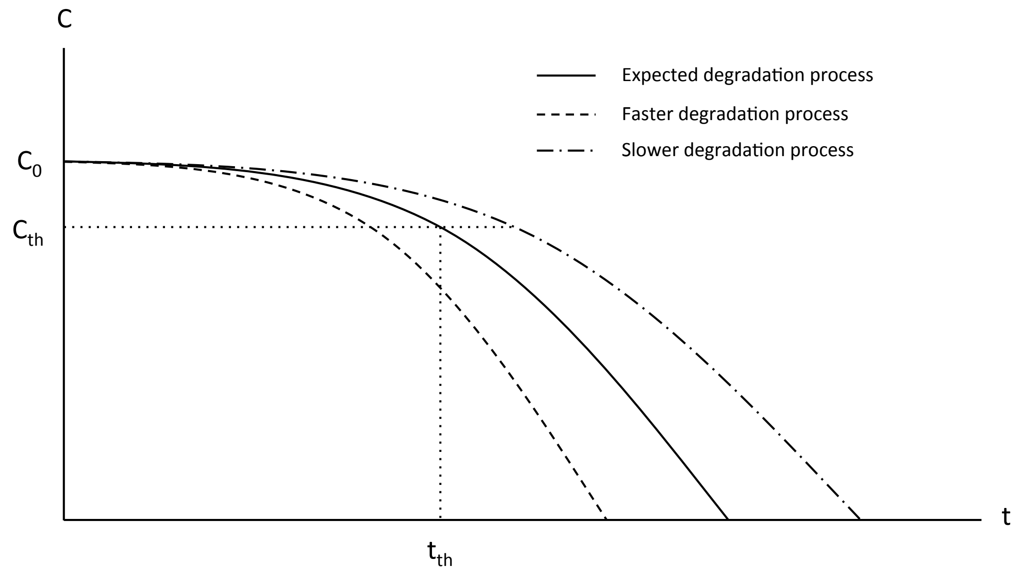


Figure 3-7: Example of different degradation trajectories of C.

Change to C can be measured either as degradation of a component's critical health parameter or as damage to the component. Health degradation and damage are related, but one is usually easier to be measured than the other. However, both will reach critical thresholds where the component fails.

3.3 Inference with Markov Chain Monte Carlo

3.3.1 Bayesian inference

The BN is a model for the variables and their relationships. Therefore, it can be used to answer probabilistic queries about them. The main application is to use BN to realize updated knowledge of the states of a subset of variables, when the other variables (the evidence variables) are observed.

To begin, we recall that Bayes' rule with continuous variables is written as:

$$p(\theta|D) = \frac{p(D|\theta)p(\theta)}{\int d\theta \, p(D|\theta)p(\theta)} \tag{3-16}$$

where θ is the parameter of interest and D is data or evidence. $p(\theta|D)$ is then the posterior probability of getting parameter value θ when data value D is presented.

Therefore, the Bayes' rule in this case is:

$$\underbrace{p(\theta|D)}_{\text{posterior}} \quad = \quad \underbrace{p(D|\theta)}_{\text{likelihood}} \quad \underbrace{p(\theta)}_{\text{prior}} \quad / \quad \underbrace{\int d\theta \, p(D|\theta)p(\theta)}_{\text{evidence}} \tag{3-17}$$

In real world SHM applications, it may be difficult to calculate full marginal distributions analytically. Therefore, sampling techniques need to be used to approximate the distributions instead. Expected values of a distribution can be estimated as follow:

$$E[p(\theta|D)] \approx \frac{1}{N}\sum_{n=1}^{N} p\left(\theta^{(n)}|D\right) \tag{3-18}$$

where $\theta^{(1)}, \dots, \theta^{(n)}$ are the sample values of parameter θ.

There are many ways to sample these values, the key idea is to let θ values be points in state space and find a way to walk around so that the likelihood of visiting any point θ is proportional to $p(\theta)$. Therefore, the sampler will spend more time sampling from the distribution where the probability is high, and spending less time sampling from where the probability is low.

3.3.2 Markov Chain Monte Carlo

In Bayesian inference, we need a good description of the posterior distribution. If we cannot achieve that description through formal analysis, nor through dense-grid approximation, then we can generate a lot of representative values from the posterior distribution and use those values to approximate the posterior.

Markov Chain Monte Carlo (MCMC) algorithm (Cousins, Chena, & Frisse, 1993) (Dagum & Horvitz, 1993) generates a random walk such that each step in the walk is completely independent of the steps before the current position. The proposed next step has no dependence on where the walk has been before, and the decision to reject or accept the proposed step has no dependence on where the walk has been before. Any such process in which each step has no memory for states before the current one is called a (first order) Markov process, and a succession of such steps is a Markov chain.

A Markov chain can be defined as a sequence of generated random variables, say $\{X_0, X_1, X_2, \dots\}$, such that at each time $t \geq 0$, the next state X_{t+1} is sampled from a distribution $p(X_{t+1}|X_t)$. The main property of a Markov chain is the fact that, given that one knows X_t, the next state does not depend of any state before X_t.

An important definition is the stationary distribution, which does not depend on t or the initial state, given that the current state depends directly from the initial one. We can represent this distribution as ϕ. Furthermore, we can note that, as *t* increases, the sampled points will look like dependent ones from ϕ.

Thus, after several generations of samples, these ones will be approximately dependent from ϕ. Under this assumption, the expectation of $g(x)$ can be done according with these several generations.

$$\hat{\theta}_{mk} = \frac{1}{n-m} \sum_{i=m+1}^{n} f(X_i) \qquad (3\text{-}19)$$

Any simulation that samples a lot of random values from a distribution is called a Monte Carlo simulation. For integration simulation, the algorithm is called Monte Carlo integration.

Let $g(x)$ be a function and suppose we want to compute φ.

$$\varphi = \int_0^1 g(x)dx \tag{3-20}$$

Thus, the value of φ can be approximated by sampling values $X_1, X_2, \dots, X_n$ from a distribution.

$$\varphi = E(g(x)) \approx \frac{1}{m}\sum_{i=1}^{m} g(X_i) \tag{3-21}$$

According to the laws of large number, increasing the value of m corresponds to more accuracy in this approximation but only if the samples $\{X_i\}$ are independent.

Therefore, in Bayesian analysis, expected values of a posterior distribution can be estimated as follow:

$$E[p(\theta|D)] \approx \frac{1}{T}\sum_{t=1}^{T} p\left(\theta^{(t)}|D\right) \tag{3-22}$$

Where $\theta^{(1)}, \dots, \theta^{(t)}$ are the sample values of parameter θ.

There are many ways to sample these values, the key idea is to let θ values be points in state space and find a way to walk around so that the likelihood of visiting any point θ is proportional to $p(\theta)$. Therefore, the sampler will spend more time sampling from the distribution where the probability is high, and spending less time sampling from where the probability is low.

MCMC algorithms are a general class of computational methods used to produce samples from distribution (in Bayesian, MCMC is used to produce samples from posterior distribution). They are often easy to implement and can be used to simulate very high dimensional posterior distribution.

The basic goal of an MCMC algorithm is to simulate values (also called samples or draws) from the posterior distribution of a parameter vector. Inference about likely parameter values, or functions of parameter values, is then based on these simulated values. Letting the j^{th} value in such a sequence of draws of the parameter vector θ be denoted by $\theta^{(j)}$. MCMC algorithms have the property that the distribution of the j^{th} iterate in the sequence of sampled values converges to a random sample drawn from the posterior distribution as j becomes large. In general, successive draws from the posterior are correlated, but this correlation tends to die out as the interval between draws increases. Thus, if a large number of sample updates are performed, the last group of sampled values in the sequence, say $\theta^{(m)}, \theta^{(m+1)}, \dots, \theta^{(m+K)}$, represents a (dependent) sample from the posterior distribution of interest. The iterations, $\theta^{(1)}, \dots, \theta^{(m-1)}$, are known as burn-in and do not represent samples from the posterior distribution.

Viewed from slightly more general perspective, MCMC algorithms produce random walks over a probability distribution. By taking a sufficient number of steps in this random walk, the MCMC simulation algorithm visits various regions of the parameter space in proportion to their posterior probabilities. We can, for inferential purposes, summarize the iterates obtained in these random walks much as we would summarize an independent sample from the posterior distribution.

Previous descriptions show how the expected value of θ by Monte Carlo integration and Markov chains can be obtained. The latter seems to solve the problem of finding a suitable approximation of the expected value to $g(x)$. However, we must guarantee that the stationary distribution corresponds exactly to the distribution of interest. The method that constructs a Markov chain and guarantees this condition is the well-known Metropolis-Hastings algorithm (Appendix A). For the case that the complete joint posterior cannot be analytically determined and cannot be directed sampled, but all the conditional distributions can be determined and directly sampled, Gibbs sampling technique can be useful (Appendix B). Gibbs sampling simplifies a complex high-dimensional problem by breaking down into simple, low dimensional problems. In the cases where evaluation of full conditional density is computationally expensive, Adaptive Rejection Sampling (ARS) can be used to draw samples with fewer evaluations (Appendix C). The algorithm only works with probability density functions that are log-concave, which is usually the case for Bayesian applications.

3.4 Example

To demonstrate the proposed methodology, SHM of an advanced integrated circuit (IC) was considered. Reliability of semiconductor devices may depend on assembly, use, and environmental conditions. Stress factors affecting device reliability include gas, dust, contamination, voltage, current density, temperature, humidity, mechanical stress, vibration, shock, radiation, pressure, and intensity of magnetic and electrical fields.

The dominant failure modes of ICs are different depending on materials, usage, configurations, and applications. Let *C* be component critical parameter related to each failure mechanism and *P* be the functionality probability of failure of each material/device. Figure 3-8 shows an example BN for ICs with the following failure mechanisms: electromigration (EM), stress migration (SM), and corrosion (Cor) in Al-alloy metal strips, thermal-cycling fatigue (TCF) in Si-chips, and time-dependent dielectric breakdown (TDDB) and hot-carrier injection (HCI) in MOSFET devices.

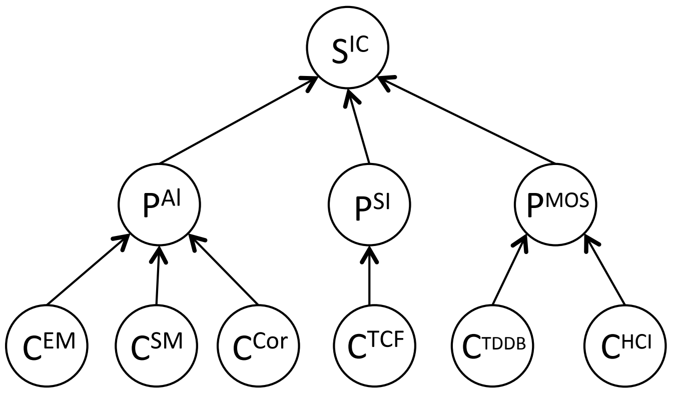


Figure 3-8: Example BN for an advanced integrated circuit

The conditional probability density of the system node, S^{IC}, is then:

$$\pi(S^{IC}) = p(S^{IC}|P^{Al}, P^{SI}, P^{MOS}) \tag{3-23}$$

$$\pi(S^{IC}) = p(S^{IC}|\{C^{EM}, C^{SM}, C^{Cor}\}, \{C^{TCF}\}, \{C^{TDDB}, C^{HCI}\}) \tag{3-24}$$

Each component critical parameter contains sub-network with continuous factor nodes and structure for that specific failure mechanism. For example the model generally used to describe EM time-to-failure takes the form:

$$TF = B_0\left(J^{(e)} - J_{crit}^{(e)}\right)^{-n} \exp\left(\frac{Q}{K_B T}\right), \tag{3-25}$$

where TF is the component time to failure. B_0 is a process/material-dependent coefficient. $J^{(e)}$ is the electron current density. $J_{crit}^{(e)}$ is a critical (threshold) current density which must be exceeded before significant EM is expected. n is the current density exponent. Q is the activation energy.

Using degradation model of component/device parameter C with the power-law equation:

$$C = C_0\left[1 - A_0\left(J^{(e)}, T\right)t^m\right] \tag{3-26}$$

We can derive at the following relationship:

$$C = C_0\left[1 - A_0 \cdot \left(J^{(e)} - J^{(e)}_{crit}\right)^r \cdot \exp\left(\frac{-Q}{K_B T}\right) t^m\right] \tag{3-27}$$

Since both current density $J^{(e)}$ and temperature T are expected to be normally distributed between time *t-1* to *t*,

$$J^{(e)} = \mathcal{N}(\mu_J, \sigma_J), T = \mathcal{N}(\mu_T, \sigma_T) \tag{3-28}$$

In the context of simple health monitoring in this example, A_0, Q, r, and m are considered to be constant parameters representing material/device internal factors. These parameters can also be modeled with probabilistic distributions.

The BN model of EM in the IC example is shown in Figure 3-9.

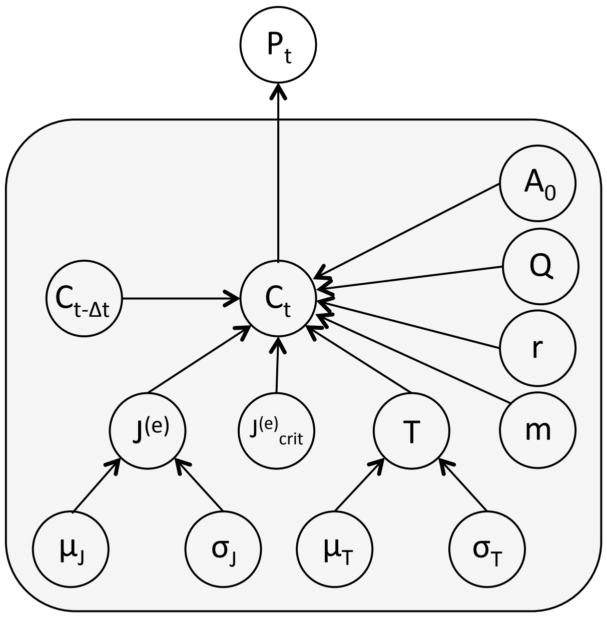


Figure 3-9: BN of EM in the IC example.

Consider an Al-alloy under high temperature operation, with expected current density of J = 2×10^6 A/cm^2 and at expected temperature of T = 200 °C. Assuming an activation energy of Q = 0.8 eV and the current density exponent of n = 2. Using conservative design approach, assume $J_{crit} = 0$.

Let $C_0 = 1$. The following parameters were derived empirically and K_B is the Boltzmann's constant. Time is in hours.

$$A_0 = 5\times10^{-12}, r = 2, m = 3, Q = 0.8, K_B = 0.0000862 \tag{3-29}$$

Figure 3-10 shows a plot of the critical parameter C degrades over time.

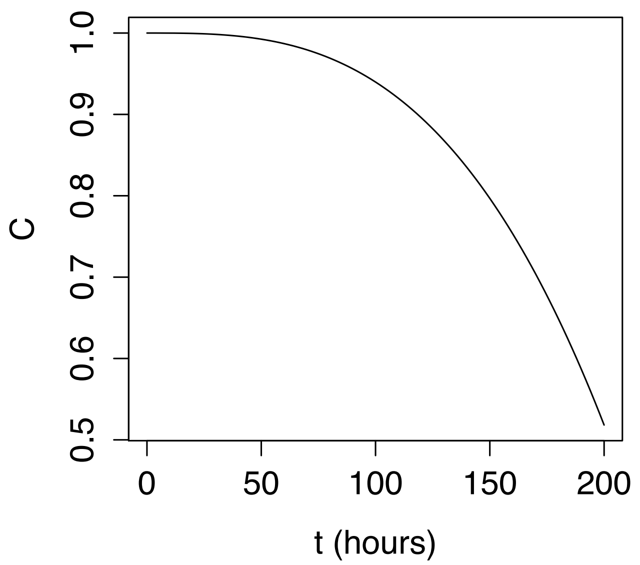


Figure 3-10: Plot of critical parameter C of the EM example over time.

For example at $t = 100$, we can infer the value and its distribution of component critical parameter C from the evidence data of J and T, shown in Figure 3-11.

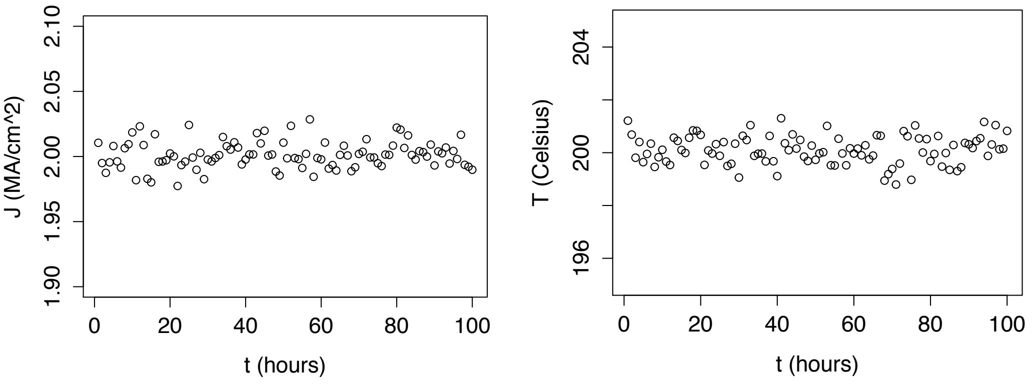


Figure 3-11: Plot of example data of J and T over 100 hours.

From the data, we have normal distributions for both J and T, with the following means and standard deviations.

$$\mu_J = 2\times10^6 A/cm^2, \sigma_J = 1\times10^5 A/cm^2, \mu_T = 200℃, \sigma_T = 0.5℃ \quad (3\text{-}30)$$

Figure 3-12 shows the MCMC inference result for C with 10,000 iteration samples (trace plot on the left and density plot on the right).

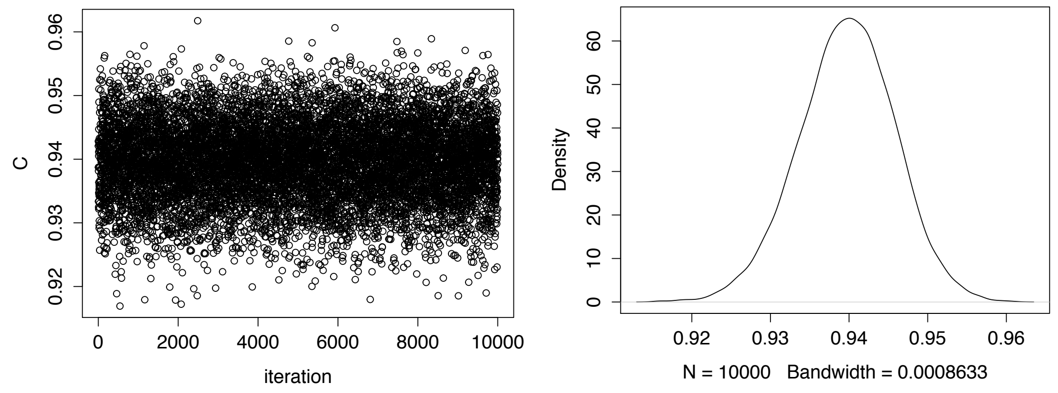


Figure 3-12: Inference result for component status C with 10,000 iteration samples.

The mean value of C is 0.940, with standard deviation of 0.00612.

Given that from the testing, the threshold distribution of the component critical parameter for EM failure mode has the following mean and standard deviation:

$$\mu_{C_{th}} = 0.7, \sigma_{C_{th}} = 0.1 \tag{3-31}$$

Using Equation (3-10), we can calculate the probability that the component critical parameter has not reach the failure threshold. Figure 3-13 shows reliability probability of Al-strip on EM failure mode over time.

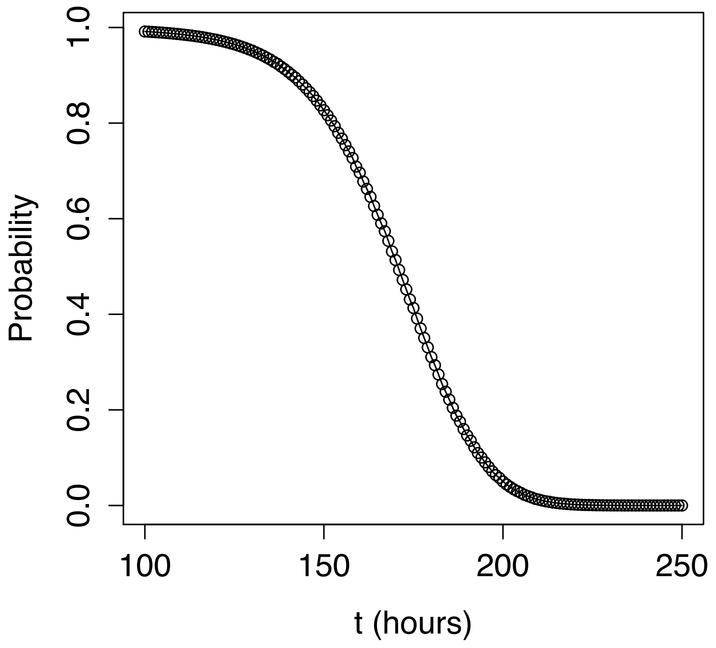


Figure 3-13: Reliability probability of Al-strip on EM failure mode over time.

Chapter 4: Computational Approach

In highly complex systems, a MCMC algorithm requires large amount of computational time for inference in hybrid DBN. The computation time grows exponentially with each additional layer of network and becomes infeasible with a large number of nodes. The computation time makes the use of Hybrid DBNs impractical for on-line health monitoring of complex systems. To solve this problem, special case algorithms for SHM are introduced to reduce the number of computations and the amount of time required for each computation.

4.1 Inference pre-computation

Since the physical parameter values are expected to be in certain ranges, it is possible to perform pre-computation for all combinations of possible values in the known ranges. The results are then stored in a database, such that they can be pulled quickly to approximate the BN inferences. More computation may be conducted and more results added to the database while monitoring the health of the system such that the database could provide better coverage of the possible computations that may be needed in the future. Figure 4-1 illustrates the pre-computation process to replace the computations required during the operating phase.

With a continuous range of parameter values, it is impossible to pre-compute every possible outcome. The goal of pre-computation is to cover enough values of the observable parameters so that the values of unobservable parameters can be interpolated from the results. There are two factors in considering the selection of possible values.

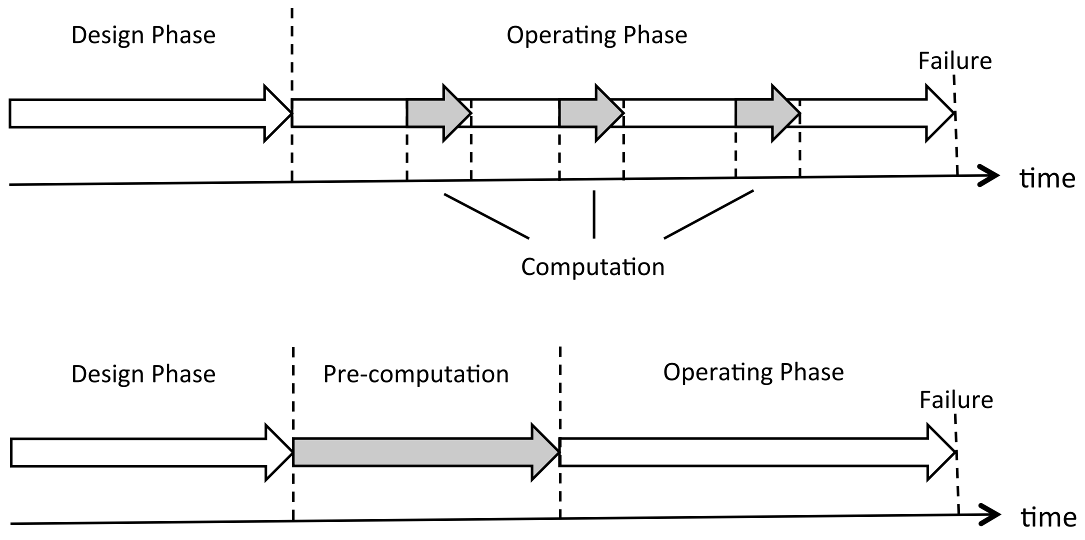


Figure 4-1: Pre-computation process

There are two types of parameters:

1. Observable parameters: these parameters that can be directly observed or measured. The values then can be used to infer other parameter values
2. Unobservable parameters: these parameters cannot be observed directly given the system and available sensors/measurements. The values can only be inferred from other parameters.

Our pre-computation then consists of inferences from values of observables parameters to unobservable parameters. Essentially, we have to select all different combinations of possible observable parameters values to perform MCMC inference

There are two factors in considering the selection of possible values.

First is the range of observable parameters after a time period Δt. The selections should cover full range of possible values. There should be at least one selected value at lower bound and one selected value at upper bound. The common range is from 5th percentile to 95th percentile, or more accurately 0.5th percentile to 99.5th percentile.

Second is the number of selections within the bound: the higher the number of selections, the more accurate results from interpolation will be. The density of selections should be proportional to the probabilistic density of the observable parameters. For example, if there is N number of selections per variable, the selections are:

$$C_s = \left\{C^{p_{low}th}, C^{p_{low}+\delta th}, C^{p_{low}+2\delta th}, \dots C^{p_{high}th}\right\} \tag{4-1}$$

$$\delta = \frac{p_{high} - p_{low}}{N_{selections} - 1} \tag{4-2}$$

Therefore, for a given measurement interval Δt, we can estimate the set of possible values and use those values to pre-computed possible outcomes.

There are two different types of observable parameters. The first one is the parameters that change over time. This is usually the case for component status parameters. For pre-computation to be feasible, the changes must be predictable. For a component status parameter, the change in value can be computed from its degradation equation for a given Δt. Figure 4-2 shows example expected value, 5th percentile, and 95th percentile values.

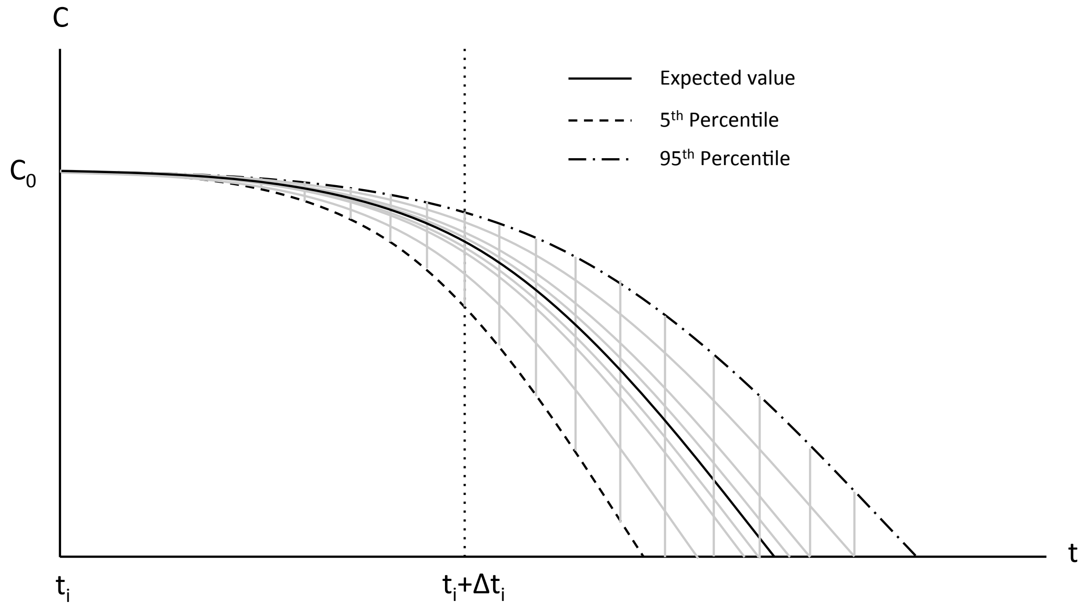


Figure 4-2: Example component degradation with 5th percentile, and 95th percentile values.

For this case, the range of possible values grows over time. Therefore, the number of selection should increase proportionally with the range to keep the interval between selected values the same, thus, keep the accuracy of interpolation constant.

The other type of observable parameters is constant parameters. These parameters are usually Gaussian distributed, shown in Figure 4-3. For this case, the range always stay constant, therefore, the selections remain the same throughout the life of the component.

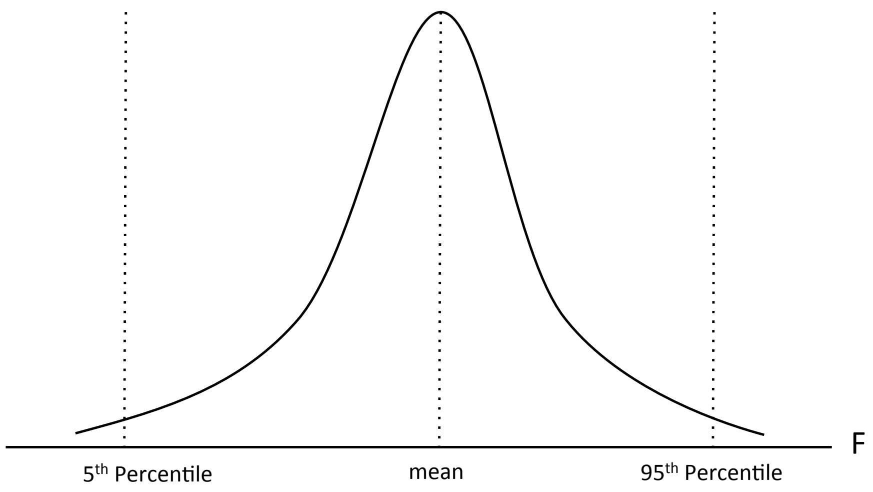


Figure 4-3: Component Gaussian distribution.

If the observed values are always in the predicted range, the accuracy of the results depends upon the number of selections for pre-computation. The number of selections is the number of selections at each time-slice multiplied by the number of measurement intervals. The number of pre-computations is then the number of selections for each observable, times the number of observables parameters.

$$N_{pre-computation} = \sum_{j=0}^{T_c/\Delta t} \left(\prod_{i=1}^{n} N_{selections,i,t+(j\Delta t)} \right) \tag{4-3}$$

Where $N_{selections,i,t}$ is the number of selections of observable parameter i at time t. n is the number of observable parameters, and T_c is the component life.

The total computation time then can be estimated.

$$T_{pre-computation} = N_{pre-computation} \cdot T_{average-per-computation} \tag{4-4}$$

For MCMC computation, the average computation time is proportional to the number iterations. The higher the number of iterations, the higher accuracy of the result will be. Therefore, there is a tradeoff between computation time and accuracy. For pre-computation, the decision between higher number of value selections or higher number of iteration per computation must be made.

4.2 Dynamic Programming

Dynamic programming is a method for solving complex problems by breaking them down into simpler subproblems. It is applicable to problems exhibiting the properties of overlapping subproblems and optimal substructure. When applicable, the method takes far less time than naive methods that do not take advantage of the subproblem overlap.

In general, to solve a given problem, we need to solve different parts of the problem (subproblems), and then combine the solutions of the subproblems to reach an overall solution. Often when using a more naive method, many of the subproblems are generated and solved many times. The dynamic programming approach seeks to solve each subproblem only once, thus reducing the number of computations: once the solution to a given subproblem has been computed, it is stored the next time the same solution is needed, it is simply looked up. This approach is especially useful when the number of repeating subproblems grows exponentially as a function of the size of the input.

Using dynamic programming can reduce the pre-computation time for BN inference drastically. Instead of computing full inferences for each set of evidence values, dynamic programming algorithm retain marginal results that can be reused with similar set of evidence values.

The core of dynamic programming is the Memoization Methodology, which consists of the following:

- Start with a backtracking algorithm
- Look up the problem in a table; if there's a valid entry for it, return that value
- Otherwise, compute the problem recursively, and then store the result in the table before returning the value

The algorithm has three steps:

1. Use a logic-sampling algorithm and degradation model to generate all possible evidence values according to its probability of occurring. Not all evidence nodes have to be instantiated for each case, only the evidence nodes that are required for observing nodes are instantiated.
2. Check and construct a cache by comparing each generated case to those already in the cache. If the case is found to be new, then determine the joint probability of the case's evidence using the algorithm in the third step.

3. The marginal posterior-probability distributions over the diagnosis nods are determined, then the values of the evidence nodes, the joint probability of the evidence set, and the marginal posterior-probability distributions for the diagnosis node are stored in the cache.

Figure 4-4 shows two example cases where dynamic programming can reduce the number of computation. The first case is when nodes have the same set of parent nodes, thus the same sets of possible marginal probability distributions for discrete nodes. The second case is when continuous parameters have several trajectories that can reach the same values after some period of time.

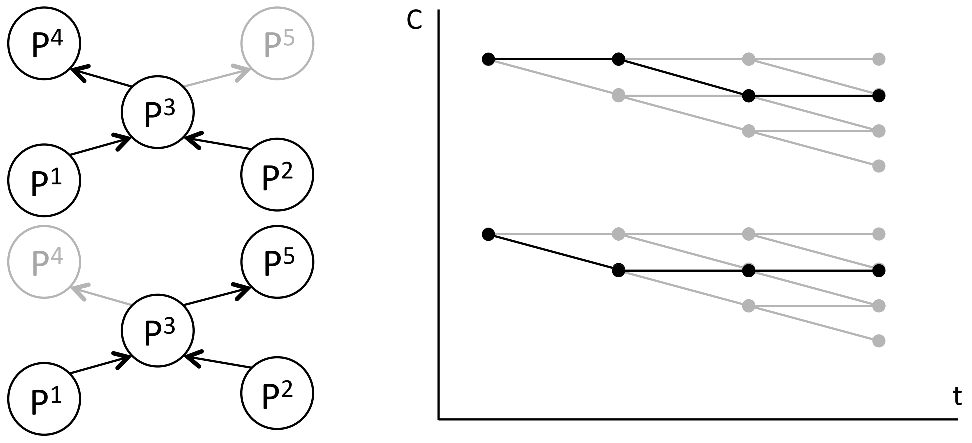


Figure 4-4: Example cases where dynamic programming reduces number of computations

In addition, if more computations are needed during an operation in the event where evidence values reaches the bound of expected values, dynamic programming provide a set of marginal results that can be used for possible faster inference of values outside the pre-computed cache.

According to the algorithm and the proposed modeling approach, the computation should start from each individual component failure mechanism, and then compute the functionality probability from lower-level nodes (related to failure mode) to higher-level (system functionality) nodes. Table 4-1 shows the algorithm to generate pre-computed database, using dynamic programming.

Table 4-1: Dynamic programming algorithm for pre-computation

Algorithm: Dynamic Programming for Pre-Computation

Input: unobserved variables ($\boldsymbol{X^U}$), observed variables ($\boldsymbol{X^E}$), variable starting values ($\boldsymbol{X_0}$), variable change function over time ($\boldsymbol{F_x}$), variable number of selections ($\boldsymbol{N_S}$), time range ($\boldsymbol{T}$), time step ($\Delta \boldsymbol{t}$)

Output: database containing unobserved values given observed values and time

```
01. for $\boldsymbol{t}$ increment by $\Delta\boldsymbol{t}$ until $\boldsymbol{T}$
02.     generate selected possible values of $\boldsymbol{X_t^E}$ according to $\boldsymbol{N_S}$ and $\boldsymbol{t}$
03.     for each combination of selected $\boldsymbol{X_t^E}$
04.         if the inference result of $\boldsymbol{X_t^U}$ given $\boldsymbol{X_t^E}$ is not in the database
05.             compute inferences for $\boldsymbol{X_t^U}$ given $\boldsymbol{F_x}$ and $\boldsymbol{t}$, and $\boldsymbol{X_{t-1}^U}$ in the database
06.             if $\boldsymbol{X_t^U}$ is within the range $\boldsymbol{\varepsilon}$ of any $\boldsymbol{X'^U_t}$ already in the database
07.                 store $\boldsymbol{X'^U_t}$ in the database
08.             else
09.                 store $\boldsymbol{X_t^U}$ in the database
10. return the database
```

Consider a complex system with N number of components in the system,

$$\pi(S) = p(S|P^1, P^2, \ldots, P^N) \tag{4-5}$$

Since both exact and approximate inferences have been found to be NP-hard (Cooper G. F., 1990) (Dagum & Luby, 1993), the computation complexity for both discrete functionality and continuous component degradation model are exponential in the network's treewidth. Figure 4-5 shows a plot presenting differences between pre-computation time with and without dynamic programming. Without storing marginal probability distribution results for further computations, all approximate inference computations are required for pre-computation, increasing the computation time exponentially with network's treewidth.

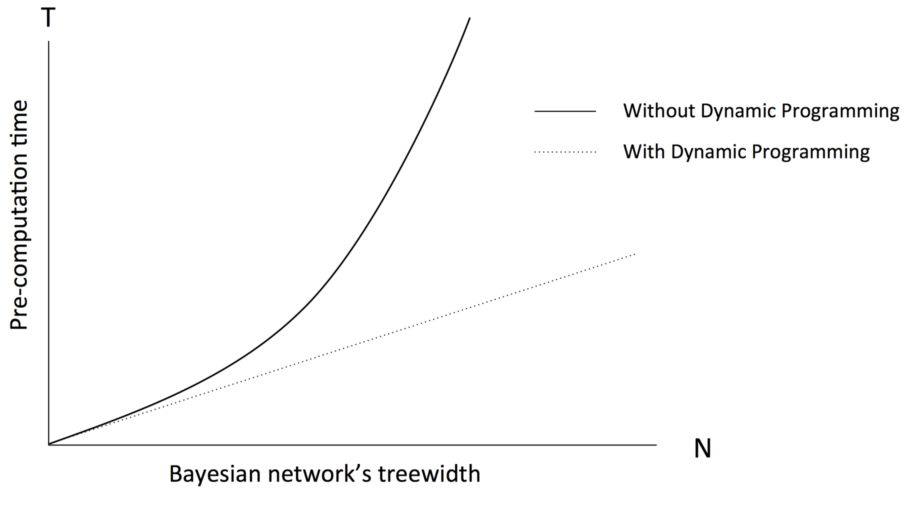


Figure 4-5: Inference pre-computation time with and without dynamic programming.

Using dynamic programming, the number of computation can be reduced drastically depending on the amount of approximation. Without dynamic programming, the number of computation required for C is:

$$N_{computation} = N_{J-selections} \cdot N_{T-selections} \cdot N_{C-values} \cdot N_{time-period} \tag{4-6}$$

With dynamic programming, results in the cases where C values are close to previous calculations can be pulled from the cache, thus reduce $N_{C-values}$ and $N_{time-period}$, and reduce $N_{computation}$. The reduction in number of computation depends on the range of parameter distributions and the level of acceptable approximation.

4.3 Computation Time Optimization

To optimize for the best pre-computation results, priority of computation becomes significant. The order of computation should start from small localized parts of the network, and then go up to the functionality probabilities and system to maximize the benefits of dynamic programming.

One advantage of the isolation among component sub-tree is that time intervals do not have to be uniform for all components. Measurement/inspection intervals can be based on the rate of component degradation and possible change to component parameters. They can also be dynamically changed during the life a component depending on its status.

For example there can be less frequency of measurements during the early life of a component due to less probability of failure. Then increase the frequency when the component approaches the end of life.

$$\Delta t \propto \frac{1}{\Delta C} \tag{4-7}$$

The time interval between measurements, Δt, should then be inverse proportional to the amount of change of the parameter C. Therefore, the sampling rate around a certain evidence value will be proportional to the probability that the evidence value could happen and how much different in values to the possible values around it at certain period of time.

If the observed values are always in the predicted range, the accuracy of the results depends upon the number of selections for pre-computation. The number of selections is the number of selections at each time-slice multiplies be the number of measurement intervals. The number of pre-computations is then the number selections for each observable times the number of observables parameters.

$$N_{pre-computation} = \sum_{j=0}^{T_c/\Delta t} \left(\prod_{i=1}^{n} N_{selections,i,t+(j\Delta t)} \right) \tag{4-8}$$

Where $N_{selections,i,t}$ is the number of selections of observable parameter i at time t. n is the number of observable parameters. T_c is the component life.

The total computation time then can be estimated.

$$T_{pre-comp} = N_{pre-comp} \cdot T_{average-per-comp} \tag{4-9}$$

For MCMC computation, the average computation time is proportional to the number iterations. The higher the number of iterations, the higher accuracy of the result will be. Therefore, there is a tradeoff between computation time and accuracy. For pre-computation, the decision between higher number of value selections or higher number of iteration per computation must be made.

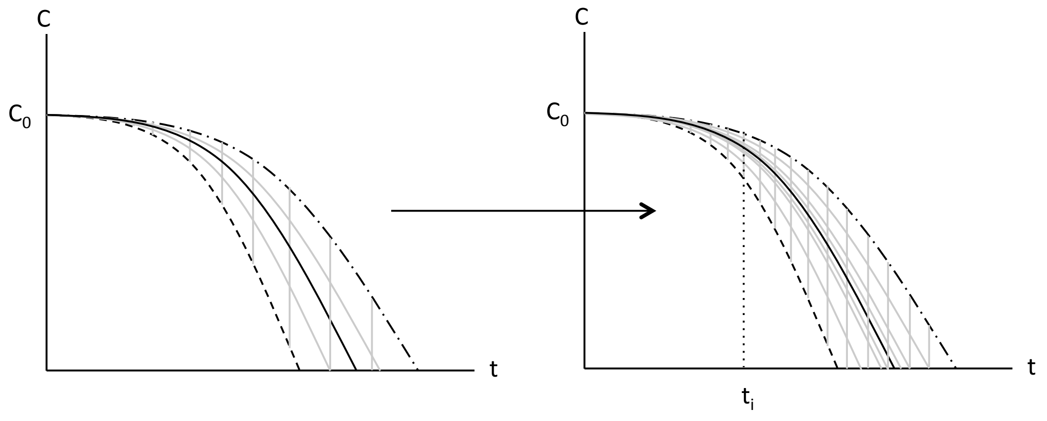


Figure 4-6: Illustrated plots for addition of inference results.

In addition, the pre-computation process does not have to stop at the start of an operation. Database of pre-computation results can be updated for more possible inference computation values, resulting in higher accuracy and larger range of covered values. Figure 4-6 shows how more results can be added between existing results.

4.4 Efficient Dependency Algorithm

In the case that components in the system are dependent on each other because they have common factors, an efficient algorithm is required to maintain efficiency of the proposed modular component model. Figure 4-7 shows a modular component Bayesian Network model for System Health Management.

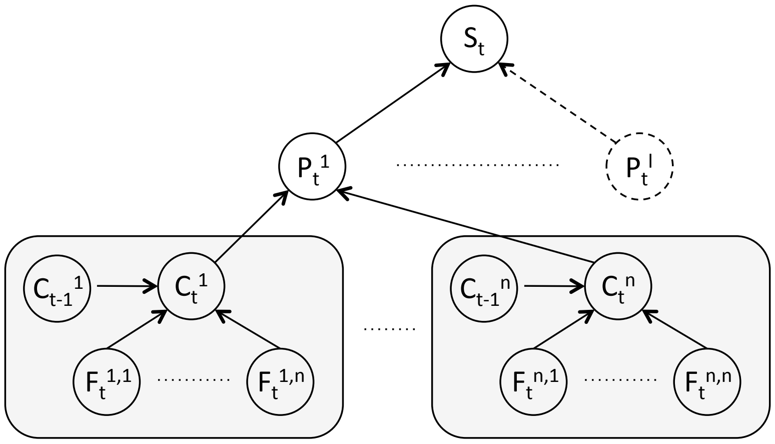


Figure 4-7: Proposed BN for SHM.

While the components are separated by their physical and functionality, there are possibilities that different components are sharing the same environmental factors, such as temperature, humidity, etc.

Figure 4-8 shows an example of a 2-component system where both components share the same factor ($F_t^{1,n}$ and $F_t^{2,1}$). The common approach is to combine the nodes, however, this method is not ideal due to the following reasons:

1. Even though the components share the same factor, it is very likely that there

is a spatial difference between them. By combining the nodes, the possibility of decoupling them is eliminated from future analysis. For example if two components are directly in contact with each other and are assumed to always have the same temperature, there is a chance that in some scenarios, the two components are separated due to an external event or unexpected degradation. The model should be flexible enough to handle this situation.

2. When identical nodes are combined into common nodes of multiple components, the component models can no longer be treated in separate modules computationally. This leads to a large increase in complexity and computation time.

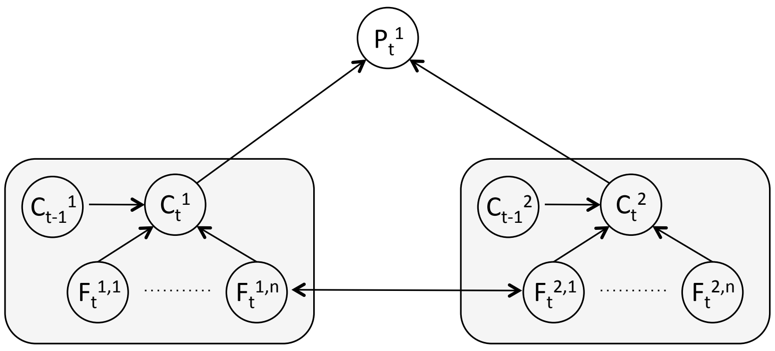


Figure 4-8: BN of components with common factor.

Let D_t be a node representing the value observed from a detector/sensor that measures the common factor. If the common factor is observable, factor $F_t^{1,n}$ and $F_t^{2,1}$ can be directly derived from the measurement value of D_t. Therefore, inference calculations for each component stay modular. Figure 4-9 show the proposed BN with an observable common factor.

$$p(C_t) = p(C|C_{t-\Delta t}, \{F_t^1, \dots, F_t^n\}) \tag{4-10}$$

$$p\left(F_t^{1,n}\right) = p\left(F_t^{1,n}|D_t\right) \tag{4-11}$$

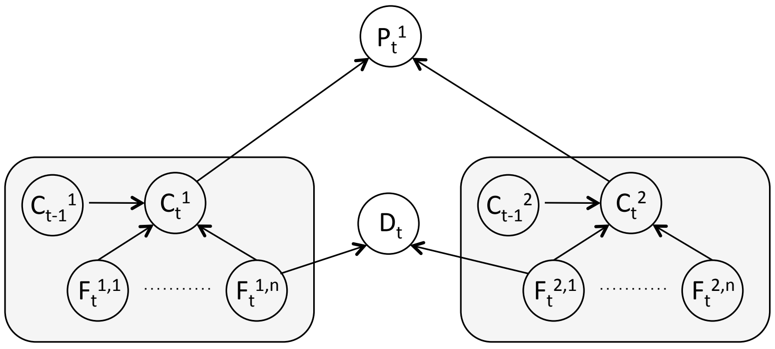


Figure 4-9: Proposed BN with an observable common factor.

If the common factor is unobservable, the inference calculation can be done by placing a “hidden node” D_t as an imaginary measurement node between $F_t^{1,n}$ and $F_t^{2,1}$, shown in Figure 4-10.

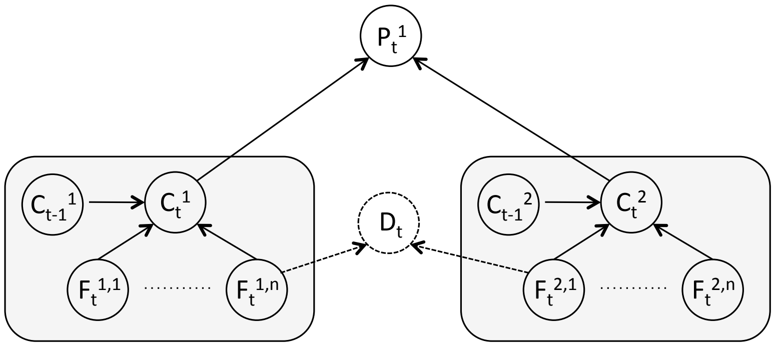


Figure 4-10: Proposed BN with an unobservable common factor

Since $F_t^{1,n}$ and $F_t^{2,1}$ are more likely to have the same value, $p(F_t^{1,n},F_t^{2,1})$ is expected to have a distribution similar to the distribution shown in Figure 4-11.

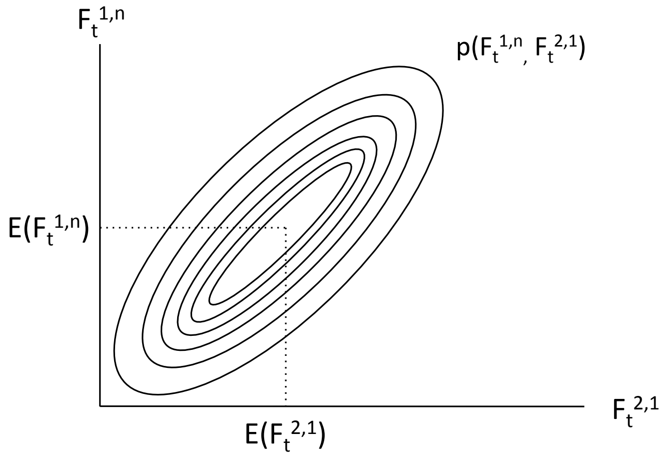


Figure 4-11: Probability distribution of the common factor.

One method to reduce computation complexity and keep the inference calculation modular is to incorporate pre-computation approximation. Pre-computation generates possible subsets of values of variables according to their probability distribution.

For this case:

$$for\ p(P_t^1|C_t^1, C_t^2) \sim \forall i, j\ p\{P_t^1|(C_t^1)_i, (C_t^2)_j\} \quad (4\text{-}12)$$

Therefore, the combination of $\{(C_t^1)_i, (C_t^2)_j\}$ that have higher probability are those for which values of $F_t^{1,n}$ and $F_t^{2,1}$ are close to each other.

$$p(\{(C_t^1)_i, (C_t^2)_j\}\,|F_t^{1,n} \approx F_t^{2,1}) > p(\{(C_t^1)_i, (C_t^2)_j\}\,|F_t^{1,n} \neq F_t^{2,1}) \quad (4\text{-}13)$$

Using this method, the most probable explanation (MPE) can be derived from the pre-computation database. Table 4-2 shows the proposed algorithm to compute inferences for common component factors.

Table 4-2: Common component factor algorithm

Algorithm: Common component factor inference

Input: unobserved common factor variables ($\boldsymbol{X^U}$), observed variables ($\boldsymbol{X^E}$), variable starting values ($\boldsymbol{X_0}$), variable change function over time ($\boldsymbol{F_x}$)**,** variable number of selections ($\boldsymbol{N_S}$), time $\boldsymbol{t}$

Output: inference values of unobserved common factor variables

```
01. for each $\boldsymbol{X^U}$ as $\boldsymbol{x^C}$
02.     for each component that has $\boldsymbol{x^C}$ factor
03.         if full conditional probability $\boldsymbol{x^C}$ has been stored
04.             retrieve and return $\boldsymbol{x'^C}$
05.         else check if $\boldsymbol{x^C}$ can be computed for full conditional probability
06.             compute  and store $\boldsymbol{x'^C}$
07.         if none of $\boldsymbol{x^C}$ can be computed for full conditional probability
08.             compute marginal probability for each $\boldsymbol{x^C}$
09.             compute most probable expected value of $\boldsymbol{x'^C}$ from marginal probabilities
10.             store $\boldsymbol{x'^C}$
11. return inference values of $\boldsymbol{X^U}$
```

4.5 Example

Consider an Al-alloy in the previous example under the same high temperature operation, with current density J = 2×10^6 A/cm^2 and at a metal temperature T = 200 °C. Assuming an activation energy of Q = 0.8 eV and the current density exponent of n = 2. Using conservative design approach, assume J_{crit} = 0.

To generate pre-computation results for EM, first we need to find the ranges of the observable factors, J and T, shown in Table 4-3.

Table 4-3: Ranges and distribution parameters of J and T

Parameter	5th Percentile	95th Percentile	Mean	Standard Deviation
μ_J	1.7×10^6	2.3×10^6	2.0×10^6	1.8×10^5
σ_J	0.5×10^5	1.5×10^5	1.0×10^5	3.0×10^4
μ_T	190	210	200	6.0
σ_T	0.5	1.5	1	0.30

Let $N_{selections} = 19$:

$$\delta = \frac{p_{high} - p_{low}}{N_{selections} - 1} = \frac{95 - 5}{19 - 1} = 5 \tag{4-14}$$

Therefore,

$$\{\mu_J\}_{selected} = \{\mu_J{}^{5th}, \mu_J{}^{10th}, \mu_J{}^{15th}, \ldots \mu_J{}^{95th}\}$$

$$\{\sigma_J\}_{selected} = \{\sigma_J{}^{5th}, \sigma_J{}^{10th}, \sigma_J{}^{15th}, \ldots \sigma_J{}^{95th}\}$$

(4-15)

$$\{\mu_T\}_{selected} = \{\mu_T{}^{5th}, \mu_T{}^{10th}, \mu_T{}^{15th}, \ldots \mu_T{}^{95th}\}$$

$$\{\sigma_T\}_{selected} = \{\sigma_T{}^{5th}, \sigma_T{}^{10th}, \sigma_T{}^{15th}, \ldots \sigma_T{}^{95th}\}$$

Figure 4-12 shows plots of J and T parameters over all selected percentile values.

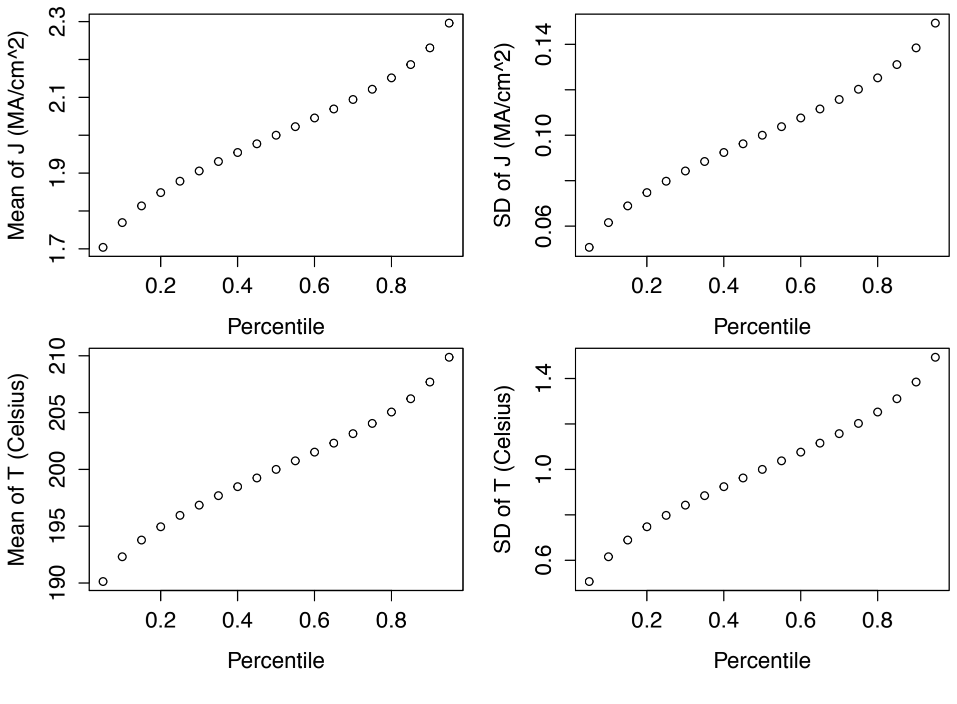


Figure 4-12: Plots of J and T parameters over all selected percentile values.

Then we compute *C* from all the possible combinations values of J and T parameters, and possible set of value *C* at time *t*. Then store the results in the database. With the proposed SHM BN modeling method, these data can be used for all similar components at any location and time.

Given the data set shown in Figure 4-13 of $J^{(e)}$ and T during an operation.

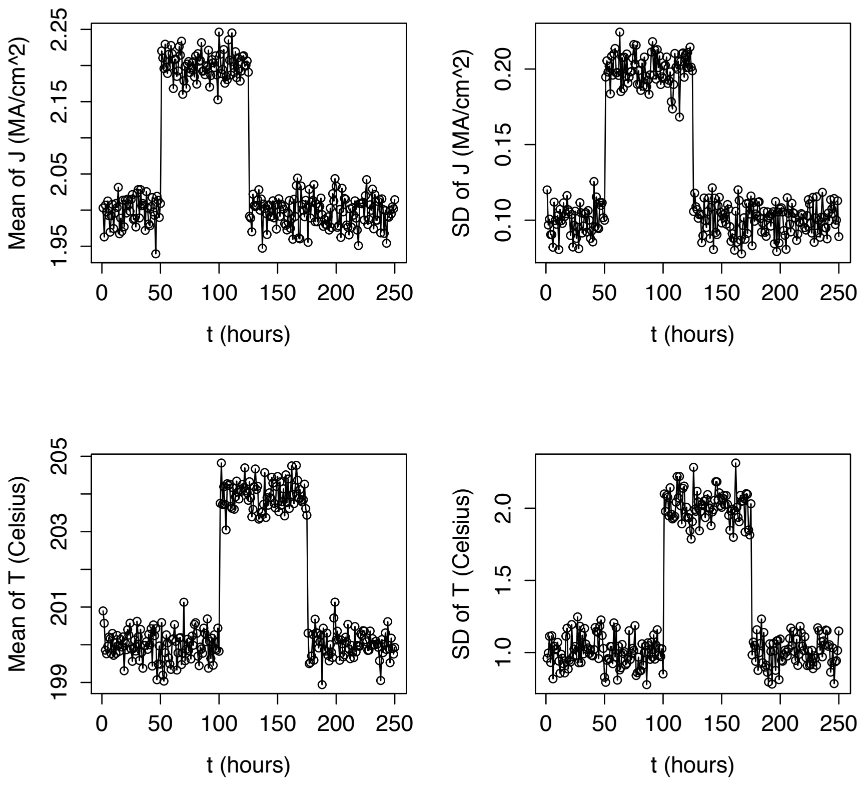


Figure 4-13: Current density and temperature parameters data set

Figure 4-14 shows a plot of component degradation under electromigration vs. time at different current density and temperature from the data set, with expected degradation trend for J=2×10^6 A/cm^2, T=200°C and J=2.3×10^6 A/cm^2, T=210°C.

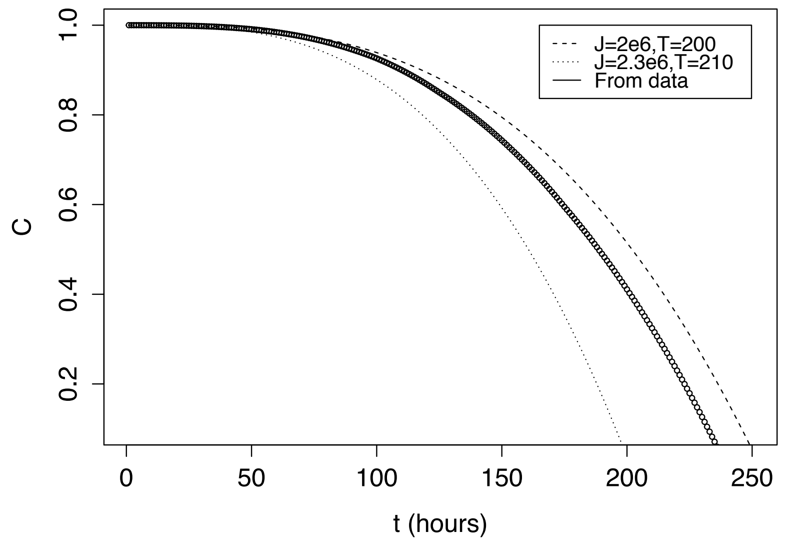


Figure 4-14: Plot of component degradation under electromigration vs. time.

Figure 4-15 shows the difference between the traditional inference result and the result from pre-computation method. The results are almost the same, with less than 0.25% error.

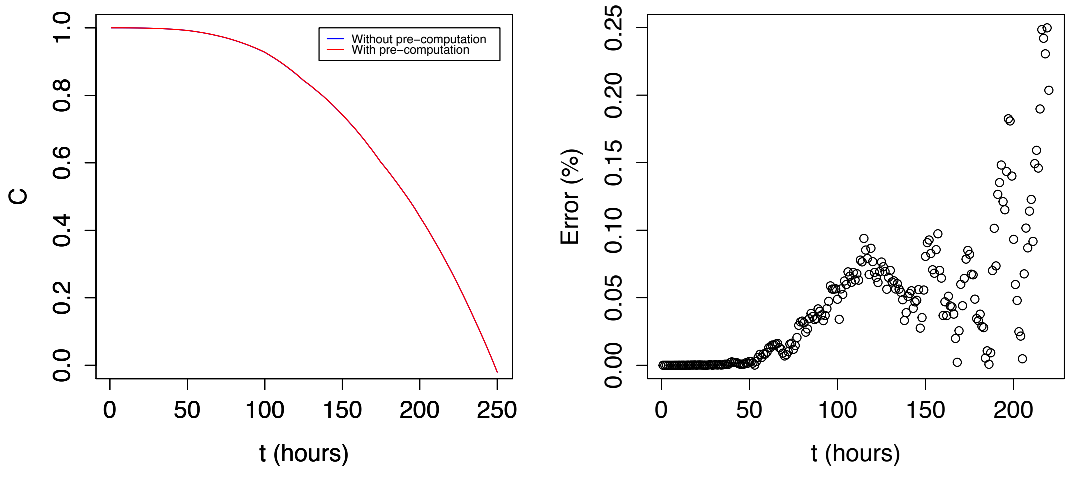


Figure 4-15: The difference between results with and without pre-computation method.

As mentioned in the previous section, with pre-computation method, the accuracy of inference computation depends mainly on the number of selections of possible values of the variables. Figure 4-16 shows the percentage error of inference result of reliability probability due to the EM failure mechanism at t = 200 hours as a function of number of selections of J and T parameters. The error decreases as the number of selection increases until it reaches a point where the error remains roughly the same. From this result, the optimal number of selection is around 10 to 15 selections.

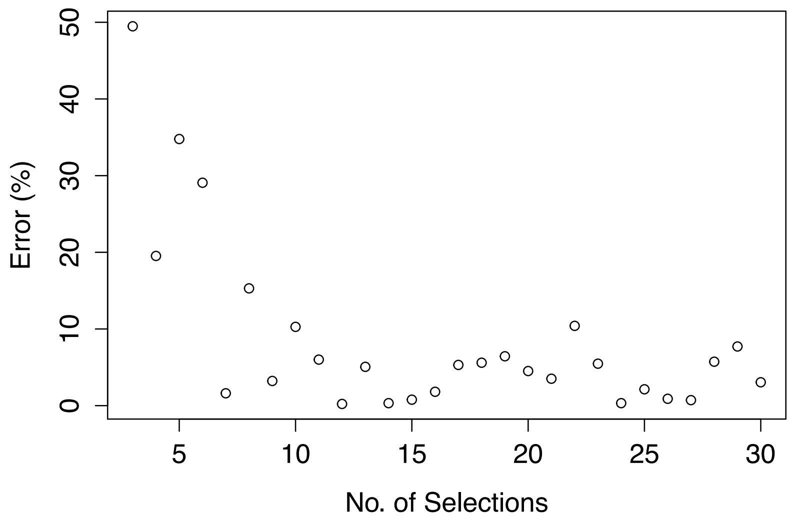


Figure 4-16: Error of inference result vs. number of parameter selections.

The optimal number of selections depends on the accuracy required by the particular application and how much pre-computation time is available. In more complex system, the number of selections should also be varied depending on the sensitivity of each variable.

To demonstrate common factor computational algorithm, consider the Al-strip in the

IC example with both electromigration (EM) and stress migration (SM) degradations. Let C^{EM} and C^{SM} be component critical parameter degrading under EM and SM respectively.

$$C^{EM} = {C_0}^{EM}\left[1 - {A_0}^{EM}\left(J^{(e)} - J^{(e)}_{crit}\right)^{r^{EM}} \exp\left(\frac{-Q^{EM}}{K_B T}\right) t^{m^{EM}}\right] \tag{4-16}$$

$$C^{SM} = {C_0}^{SM}\left[1 - {A_0}^{SM}(L)^{r^{SM}} \exp\left(\frac{-Q^{SM}}{K_B T}\right) t^{m^{SM}}\right] \tag{4-17}$$

$J^{(e)}$ is the electron current density. $J^{(e)}_{crit}$ is a critical (threshold) current density which must be exceeded before significant EM is expected. L is the tensile stress in the metal for a constant strain.

The BN model of a component affected by EM and SM is shown in Figure 4-17.

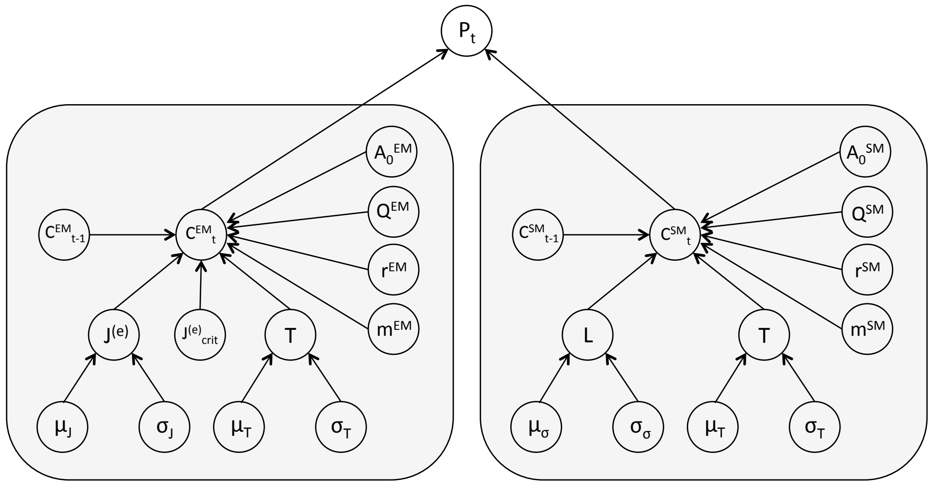


Figure 4-17: BN of a component with EM and SM degradations

Assume $J^{(e)}$, L, and T are expected to be normally distributed between time *t-1* to *t*,

$$J^{(e)} = \mathcal{N}\left(\mu_J, \sigma_J\right), L = (\mu_L, \sigma_L), T = \mathcal{N}(\mu_T, \sigma_T) \tag{4-18}$$

With traditional BN modeling, both failure modes have temperature as a common factor. Therefore, the component parameters, C^{EM} and C^{SM}, have the same parent node, T. In this case, any approximate inference will require full marginal distribution of both failure mode variables. The amount of time for sampling and computation increases exponentially with the number of variables in the inference calculation. With the proposed technique, the failure modes stay modular and approximate inferences can be achieved at much lower cost because of lower number of variables in the calculation. For this example, approximate inference calculation will only involve parameters of failure mode EM and failure mode SM, but not both of them combined.

Given an example where C^{EM}, $J^{(e)}$, and L are observables and T is an unobservable. Traditionally, to get and inference for T, the following marginal distribution computation is required.

$$\pi(T_t) = p(T_t | C_t^{EM}, C_t^{SM}, C_{t-1}^{EM}, C_{t-1}^{SM}, J_t^{(e)}, L_T) \tag{4-19}$$

With the proposed methodology, pre-computation of the following inferences for T_t are computed for all possible values of other parameters with respect to each failure mode.

$$\pi(T_t^{EM}) = p(T_t | C_t^{EM}, C_{t-1}^{EM}, J_t^{(e)}) \tag{4-20}$$

$$\pi(T_t^{SM}) = p(T_t | C_t^{SM}, C_{t-1}^{SM}, L_T) \tag{4-21}$$

The following table shows the ranges of the parameter of L in N/m^2 and T in Celsius, shown in Table 4-4.

Table 4-4: Ranges and distribution parameters of L and T

Parameter	5th Percentile	95th Percentile	Mean	Standard Deviation
μ_L	1.5×10^8	2.5×10^8	2.0×10^8	3.0×10^7
σ_L	0.5×10^6	1.5×10^6	1.0×10^6	3.0×10^5
μ_T	190	210	200	6.0
σ_T	0.5	1.5	1	0.30

If at time t = 100 hours,

$$\mu_{C_{t-1}^{EM}} = 0.975, \sigma_{C_{t-1}^{EM}} = 0.005$$

$$\mu_{C_t^{EM}} = 0.973, \sigma_{C_t^{EM}} = 0.005 \tag{4-22}$$

$$\mu_{J_t} = 2\times10^6, \mu_{J_t} = 1\times10^5$$

From the pre-computation result of EM failure mode,

$$\mu_{T_t} = 202.51, \sigma_{T_t} = 1.214 \tag{4-23}$$

With 19 selections in the previous example, the closest parameter values are:

$$\mu_T{}^{65th} = 202.31, \sigma_T{}^{75th} = 1.202 \tag{4-24}$$

Given that,

$$\mu_{C_{t-1}^{SM}} = 0.975, \sigma_{C_{t-1}^{SM}} = 0.005 \tag{4-25}$$

Therefore, from the pre-computation of SM failure mode,

$$\mu_{C_t^{SM}} = 0.895, \sigma_{C_t^{SM}} = 0.013 \tag{4-26}$$

This also allows instantaneous inquiry of the states of degradations of all components in the system without computing full inference of all nodes every time there is new information. In real applications, a sensor on tensile stress may collect data every second, while another sensor on current density collect data every a tenth of a second. The health information of the system can be updated every tenth of a second, without having to performing approximate inference for EM failure mode as often.

Consider a more complex example where the system consists of 50 electrical components that have 2 failure modes. Figure 4-18 shows a plot of amount of time required as a function of number of failure modes that have the same dependent factor. Assume an approximate inference requires 10,000 iterations to reach reasonably accurate result. Using the proposed technique, the computation stays roughly the same, while traditional computation time increases exponentially with the number of dependent failure modes.

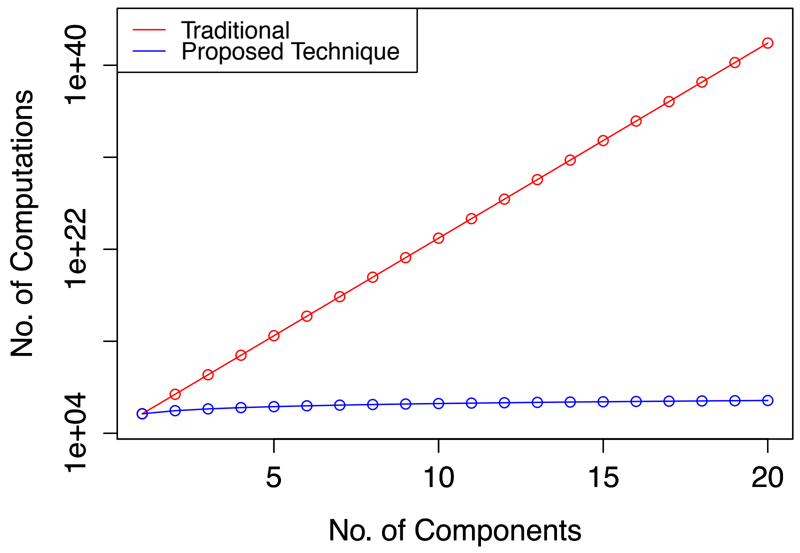


Figure 4-18: Plot of number of computation vs. number of components with the same dependent factor

Chapter 5: System Health Monitoring and Prognosis

This section presents methods to apply the proposed modeling approach and computation algorithms for on-line health monitoring, prognosis, and anomaly detection of a complex system.

5.1 Sensors

One of the most important steps when implementing designing the system for health monitoring is identifying the observables and how to obtain the measurements. This is accomplished by adding sensors. It's crucial to the effectiveness of health monitoring to deploy the correct sensors at the best possible placements.

There are two classes of sensors that are important to system monitoring for fault diagnosis and prognosis. The first one is traditional transducers aimed at monitoring mechanical, structural, performance and operational, and electrical/electronic properties that relate to failure mechanisms of mechanical, structural, and electrical systems. The second class is for sensor systems that are placed specifically to track system properties that are related directly to their failure mechanisms.

The main categories of sensors include mechanical/structure sensor systems, such as, accelerometers for vibration measurement, strain gauges, and ultrasonic sensor systems, performance/operational sensors, temperature sensors/thermography, electric measurements, such as Eddy-current proximity probes and microelectromechanical system (MEMS) sensors. The physical effects these sensors are detecting are thermal, electrical, mechanical, humidity, biological, chemical, optical, and magnetic. A typical sensor system includes sensors, onboard analog-to-digital (A/D) converters, onboard memory, embedded computational capabilities, data transmission, and power source or supply.

With the propose methodology, BN shows clearly what critical parameters and factors required measurement in order to infer the health of the system due to dominant degradation processes. Figure 5-1 shows and example BN with observable and unobservable nodes. Identifying observables and unobservables shows which sensors are important and which are not. The pre-computation results can emphasize what part of the measurement needs improvement in terms of rate and accuracy.

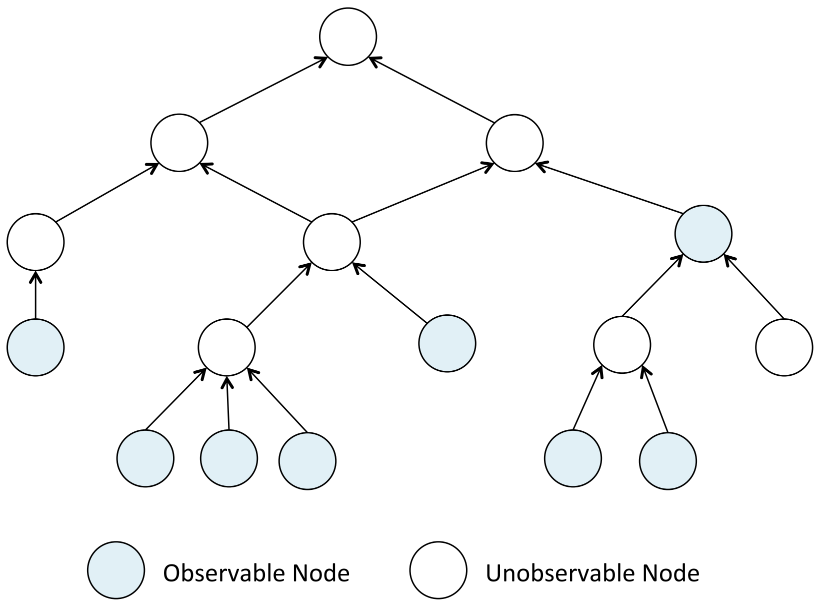


Figure 5-1: Example BN with observable and unobservable nodes.

Consideration for sensors selection includes the parameters to be monitored, requirements for physical characteristics of PHM sensor system, requirements for functional attributes of PHM sensor system, cost, reliability, and availability. Sensor system performances to be considered are accuracy, sensitivity, precision, resolution, measurement range, repeatability, linearity, uncertainty, response time, stabilization time, and physical attributes such as size, weight, and shape.

Other functional attributes to be considered are power management, memory management, sampling rate, cost, reliability, and availability. Signals coming from sensors can be noisy, low amplitude, or biased. Therefore, there is need for signal pre-processing, such as signal conditioning, denoising, vibration signal compression. After that signals are then processing in both time domain and frequency spectrum for feature selection and extraction.

BN modeling enables easy integration of information from different sources, including experimental data, historical data, and prior expert opinion. Therefore, different type of sensors can be combined within the network. BN also includes and retains all the uncertainties within the system. These uncertainties generated from sensors may be induced by the varying and uncertain environments, the unpredictability of hardware and software, noisy and ambiguous sensor systems, and incomplete and uncertain knowledge of the physics of dynamic systems.

5.2 System Monitoring

Pre-computation allows on-line tracking of the state of the system, including the status of unobserved variable. While some factors are expected to remain constant or go through predictable cycle over system operating period, some factors change over time depending on degradation of components in the system. Inference calculations of these factors require a combination of their empirical or physics of failure model and the measurement values. Recursive (sequential) Bayesian estimation is required for probabilistic inference process in which the hidden (unobserved) variables of a dynamic system are estimated based on uncertain observations, e.g. due to error in measurement. (Rabiei & Modarres, 2013).

Let D_t be a node representing the value observed from a detector/sensor to measure variable θ_t, $p(\theta_t|\theta_{t-1})$ represent system dynamic, and $p(D_t|\theta_t)$ represent observation model. Figure 5-2 shows dynamic representation of variable θ_t and its observed value over time.

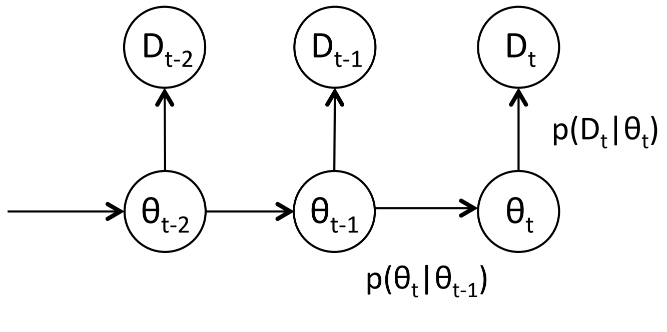


Figure 5-2: Dynamic representation of a variable.

The procedure for updating the belief about the system state as new information becomes available is called Bayesian recursive filtering.

$$p(\theta_t|D_{1:t}) = \frac{p(D_t|\theta_t)p(\theta_t|D_{1:t-1})}{\int d\theta\, p(D_t|\theta_t)p(\theta_t|D_{1:t-1})} \tag{5-1}$$

Under certain assumptions, such as when the system is linear Gaussian, the belief state will be of a known parametric form and computationally efficient solutions to the filtering problem (e.g. Kalman filter, extended Kalman filter, unscented Kalman filter) are available (Kalman, 1960) (Julier & Uhlmann, 1997). Outside such assumptions, a computationally feasible method for inference in the DBN is particle filtering, a form of sequential Monte Carlo based on Bayesian recursive filtering. Common particle filtering methods are based on sequential importance sampling (SIS) (Chen, 2003) (Appendix D).

Using the pre-computation method, values of these variables can be derived instantaneously, making it possible to continuously monitor all variables in the system. This provides more up-to-date and complete information of the system state than the traditional RUL estimation method.

5.3 Probabilistic Prognosis of Remaining Useful Life

Diagnostic capabilities traditionally have been applied at or between the initial detection of a system, component, or subcomponent failure and complete system catastrophic failure. More recent diagnostic technologies are enabling detections to be made at much earlier incipient fault stages.

Prognostics and health management (PHM) refers specifically to the phase involved with predicting future behavior, including remaining useful life (RUL), in terms of current operating state and the scheduling of required maintenance actions to maintain system health. RUL quantifies the amount of time until a system reaches some failure criterion, e.g. fault magnitude or performance metric crosses a threshold or system is no longer operable. Ideally, the uncertainty in RUL is quantified by estimating the distribution of RUL, resulting in a probabilistic prognosis.

RUL is estimated by calculating the amount of time that the component critical parameter will reach a critical value, C_{crit} given the model and the current state of degradation. For an exponential model:

$$RUL = T_{failure} = \left[\frac{1}{\pm A_0(F_1, \dots, F_n)}\left(\frac{C_{crit} - C}{C}\right)\right]^{1/m} \tag{5-2}$$

With inference pre-computation results, RULs related to each failure mechanism can be estimated at any state of the system and at any point in time, given the trajectory of the degradation process. Table 5-1 shows the RUL estimation algorithm.

Table 5-1: RUL estimation algorithm

Algorithm: RUL estimation

Input: component critical parameter history ($\{\boldsymbol{C_0}, \dots, \boldsymbol{C_T}\}$), component related variables ($\boldsymbol{X^C}$), degradation function ($\boldsymbol{F^C}$), critical parameter threshold ($\boldsymbol{C_{th}}$), current time ($\boldsymbol{T}$), time step ($\Delta\boldsymbol{t}$), database of inference results

Output: RULs related to possible trajectories for all $\boldsymbol{C}$

```
01. for each $\boldsymbol{C}$ as $\boldsymbol{C^i}$
02.     match $\{\boldsymbol{C_0^i}, \dots, \boldsymbol{C_T^i}\}$ to a trajectory of $\boldsymbol{C^i}$ with certain values of $\boldsymbol{X^{C,i}}$ given $\boldsymbol{F^C}$,
03.     for $\boldsymbol{t}$ increment by $\Delta\boldsymbol{t}$ until $\boldsymbol{C_t^i}$ reaches $\boldsymbol{C_{th}^i}$
04.         look up the database for $\boldsymbol{C_t^i}$ given $\boldsymbol{C_{t-\Delta t}^i}$ and $\boldsymbol{X^{C,i}}$
05.         if $\boldsymbol{C_t^i}$ reaches $\boldsymbol{C_{th}^i}$
06.             estimate and store RUL for $\boldsymbol{C^i}$
07. return RUL estimates
```

Even though there are many possible trajectories, no MCMC inference is required because RULs can be retrieved by recursively going through the inference database. And since it is only for component critical parameters, which are modular according to the proposed modeling approach, the number of computations is much smaller compared to large number of iterations in MCMC inference.

5.4 Anomaly Detection

Aside from monitoring the health of the system, it is crucial to be able to detect any unexpected anomaly that could potentially cause system failure. There are three types of anomaly detection:

1. Value Checks: check whether the values are plausible. The values must follow certain constraints of the particular model, such as “the value cannot be zero” or “the value must always increase or decrease over time”. They also have to be within the expected ranges.

$$|\theta_t - \theta_{t-1}| < \epsilon_{\theta,t} \tag{5-3}$$

Where $\epsilon_{\theta,t}$ depends on the sensitivity of variable status due to the change over time $\Delta t = t_i - t_{i-1}$.

2. Relationships Checks: check the relationship between nodes in the system, whether they follow the dependencies encoded in the model. Given the model S, the values of observable parameters $\{\theta_t^o\}$, and the state of system in the previous time slice $\{\theta_{t-1}\}$, the observable values follow the relationship constraints within the model.

$$p(\{\theta_t^o\}|S, \{\theta_{t-1}\}) > \epsilon_p \tag{5-4}$$

Where ϵ_p is the minimum acceptable probability of the observable values.

3. Failure trajectories check: check whether the component degradation processes are within the expected ranges. This detects if certain trajectories of a processes will

make the component parameters reach critical thresholds earlier than anticipated.

$$T_{failure}(\theta|S,\{\theta_0,\theta_1,\dots,\theta_t\}) \ll T_{failure}(\theta|S) \tag{5-5}$$

These trajectories can be detected using pattern recognition of the current state of the system comparing to the predicted trajectories in the database from pre-computation. Since pattern recognition captures trajectories of failure processes before component critical value reaches a certain threshold, it can detect possible failure earlier than checking failure probability of the component. Figure 5-3 shows a timeline comparison between pattern recognition and traditional failure detection estimate.

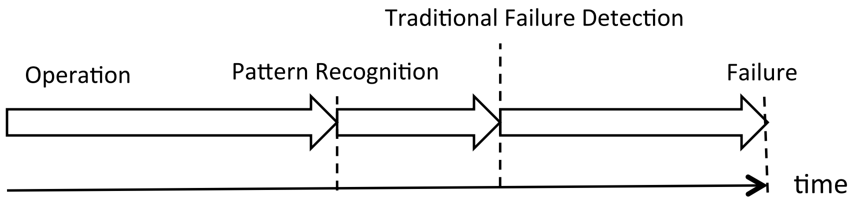


Figure 5-3: Timeline comparison for pattern recognition against traditional failure detection.

The pattern recognition algorithm uses the RUL estimates from pre-computation inference database to predict time to failure.

$$RUL_C = \mathrm{RUL}\left(\mathrm{trajectory}^i\right) \approx RUL(matched\left(\{\theta_0,\theta_1,\dots,\theta_t\}^i\right))$$

Therefore, using the database, the RUL of the system can be tracked, and continuously monitored for possible failure much earlier before failure probability reaches the threshold.

5.5 Example

Consider the EM failure example in previous chapter. The trajectory RUL database can be created from the generated inference pre-computation database.

Using the same setup. Assuming the following values:

$$A_0 = 5\times10^{-12}, r = 2, m = 3, Q = 0.8, K_B = 0.0000862$$
$$\mu_{C_{th}} = 0.7, \sigma_{C_{th}} = 0.1 \tag{5-6}$$

By simulating 50 data sets of J and T, Figure 5-4 shows the time differences between trajectory pattern recognition method and traditional failure detection estimation. The difference shows how much earlier the pattern recognition method detects incoming failure before the traditional method and it increases proportionally with the total time to failure of the component for the given dataset.

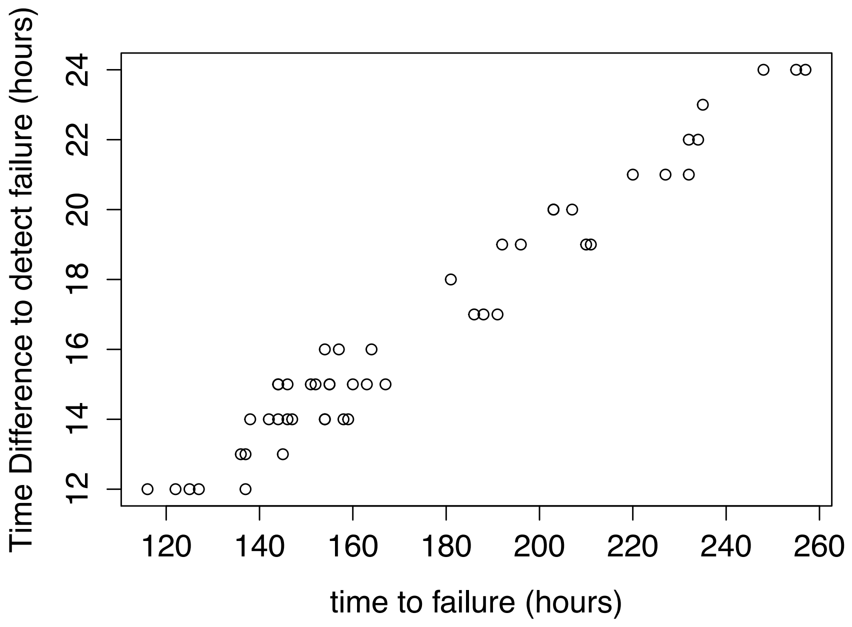


Figure 5-4: Time difference for failure detection between pattern recognition and traditional method.

Chapter 6: Parameter and Structure Learning

In addition to monitoring health of the system, if data obtained during an operation turn out to be much different from the expected model, it's crucial that the model gets updated to correct the discrepancies. Learning the parameters and structure of a BN can be considered a specific example of the general problem of selecting a probabilistic model that explains a given set of data (Friedman, Nachman, & Peer, 1999).

6.1 Parameter learning

Assume that the joint probability distribution for **X** can be encoded in some network structure *S*.

$$p(\boldsymbol{x}|\boldsymbol{\theta}_s, S^h) = \prod_{i=1}^{n} p(x_i|\mathbf{pa}_i, \boldsymbol{\theta}_i, S^h) \tag{6-1}$$

$\boldsymbol{\theta}_i$ is the vector of parameters for the distribution $p(x_i|\mathbf{pa}_i, \boldsymbol{\theta}_i, S^h)$

$\boldsymbol{\theta}_s$ is the vector of parameter $(\theta_i, \ldots, \theta_n)$

S^h is the event (hypothesis) that the joint probability distribution can be factored according to S

The problem of learning probabilities in a Bayesian network can be stated as follows: given a random sample $D = \{x_1, \ldots, x_N\}$ and the prior distribution $p(\boldsymbol{\theta}_s|\, S^h)$, compute the posterior distribution $p(\boldsymbol{\theta}_s|\ D, S^h)$.

To compute the posterior efficiently and in closed form, two assumptions must be made.

1. There are no missing data (the data set *D* is complete)
2. The elements of parameter vector $\boldsymbol{\theta}_s$ are mutually independent. That is,

$$p(\boldsymbol{\theta}_s|\ S^h) = \prod_{i=1}^{n}\prod_{j=1}^{q_i} p(\boldsymbol{\theta}_{ij}|S^h) \qquad (6\text{-}2)$$

In the case of Bayesian learning, instead of seeking point estimates, we seek a posterior distribution (density) over network parameterizations p(θ|D), conditioned on the data. This posterior can in turn be used to identify point estimates (such as the mean or the mode), or one can otherwise take the average over all possible parameterizations. In the complete data case, we have a simple closed form, when a suitable prior over parameters is assumed.

If data set D is incomplete (has missing values), the parameter learning task is more difficult since, in general, there is no tractably computable closed form. However, there are still effective learning algorithms for this case, such as gradient ascent, Gibbs sampling and expectation-maximization (EM). Such algorithms are often iterative, and intuitive, make inferences about the data to complete the missing values, which in turn are used to update the model parameters.

A further approximation is based on the observation that, as the sample size increases, the effect of the prior $p(\tilde{\boldsymbol{\theta}}_s|\mathrm{S}^{\mathrm{h}})$ diminishes. Thus, we can approximate $\tilde{\boldsymbol{\theta}}_{\mathrm{s}}$ by the maximum likelihood (ML) configuration of $\boldsymbol{\theta}_s$:

$$\hat{\boldsymbol{\theta}}_s = \arg\max_{\boldsymbol{\theta}_s}\{p(D|\boldsymbol{\theta}_s, S^h)\} \tag{6-3}$$

There are many techniques for finding a maximum likelihood (ML) or maximum a posteriori (MAP). One class of them is gradient-based optimization. For example, using gradient ascent, a local maximum can be found by following the derivatives of $g(\boldsymbol{\theta}_s)$ or the likelihood $p(D|\boldsymbol{\theta}_s, S^h)$.

Another technique for finding a local ML or MAP is the expectation-maximization (EM) algorithm (Dempster, 1977). EM is an efficient iterative procedure that searches for maximum likelihood estimates, in the presence of missing data. At a high level, each iteration of the EM algorithm consists of two steps: The E-step, and the M-step. We start with some initial guess of the network parameters. In the expectation, or E-step, we compute expected counts, based on our current estimates. In the M-step, we treat the expected counts like complete data, and compute the maximum likelihood estimates from them. We typically repeat this iterative process until it no longer improves the likelihood.

With the proposed pre-computation database, the observed variables can be adjusted according to the physics of failure model to compensate for the error in model parameters.

6.2 Structure learning and discovery

Besides differences in model parameters, there is a possibility of a hidden structure that is not expected when BN is implemented. Figure 6-1 shows an example of an unanticipated factor of a degradation process affecting another degradation process. With modular design and pre-computation database, hidden relationship can be discovered simply by Bayesian inference.

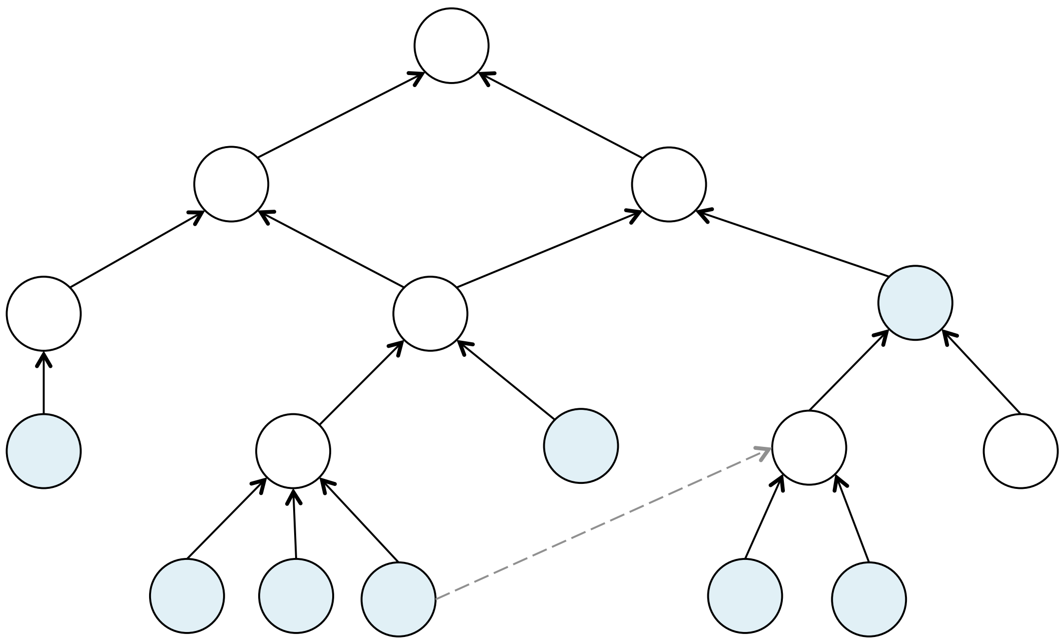


Figure 6-1: Example BN with a hidden structure.

If the network structure that encodes the physical joint probability distribution for ***X*** is uncertain and can be improved. The uncertainty can be encoded by defining a variable whose states correspond to the possible network-structure hypotheses S^h, and assessing the probabilities $p(S^h)$. Then, given a random sample *D*, the posterior distribution $p(S^h|D)$ and $p(\boldsymbol{\theta}_s|D,S^h)$ can be computed and used to compute expectations of interest. For example:

$$p(x_{N+1}|D) = \sum_{S^h} p(S^h|D) \int p(x_{N+1}|\boldsymbol{\theta}_s, S^h) p(\boldsymbol{\theta}_s|D,S^h)\, d\boldsymbol{\theta}_s \qquad (6\text{-}4)$$

In performing the sum, we assume that the network-structure hypotheses are mutually exclusive.

The computation of $p(S^h|D)$ is straightforward using Bayes' theorem:

$$p(S^h|D) = \frac{p(S^h)p(D|S^h)}{p(D)} \qquad (6\text{-}5)$$

Where p(D) is a normalization constant that does not depend upon structure. Thus, to determine the posterior distribution of network structure, we need to compute the marginal likelihood of the data $p(D|S^h)$ for each possible structure.

Chapter 7: Maintenance and Decision Making

Decision-making methodologies in SHM take RUL estimates derived from prognosis to develop and select maintenance policies or perform condition-based maintenance. While advanced diagnostics and prognostics provide vital information, good decision making with that information requires a methodology that effectively utilizes available information.

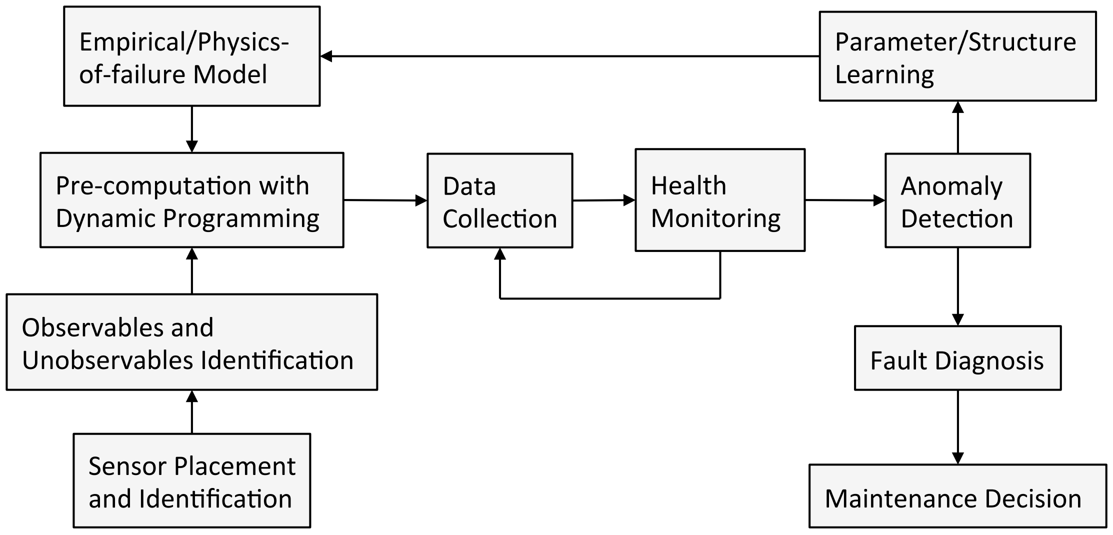

Empirical/Physics-of-failure Model
Parameter/Structure Learning
Pre-computation with Dynamic Programming
Data Collection
Health Monitoring
Anomaly Detection
Observables and Unobservables Identification
Fault Diagnosis
Sensor Placement and Identification
Maintenance Decision

Figure 7-1: Diagram of the proposed SHM process.

Figure 7-1 shows a diagram of the proposed SHM process where empirical model and sensors information are used to generate pre-computation results, which are use for health monitoring to the system. If anomaly is detected during an operation, fault diagnosis is processed to help with maintenance decision-making. Past operation data that leads to anomaly is then used for parameter and structure learning to update the model for future health monitoring and prognosis.

7.1 Condition-based Maintenance

Condition-based maintenance (CBM) is the use of machinery run-time data to determine the machinery condition and hence its current fault/failure condition, which can be used to schedule required repair and maintenance prior to breakdown. Since corrective/reactive maintenance can have severe performance costs, preventive/ scheduled maintenance is important to replaces parts before the end of their useful life.

CBM optimizes the tradeoff between maintenance costs and performance costs by increasing availability and reliability while eliminating unnecessary maintenance activities. The benefits of implementing CBM include: increased system availability, increased system reliability, reduced maintenance costs, and reduced inventories.

Potential of PHM is to reduce operational and support cost (O&S), reduce life-cycle total ownership cost (TOC), and improved safety of machinery and complex systems. There are several ways to management the cost, such as maintenance planning cost avoidance, discrete event simulation maintenance planning model, fixed-schedule maintenance interval, and precursor to failure monitoring.

To perform a cost-benefit analysis, first there needs to be an established baseline condition. Then use a condition-based maintenance to reduce breakdown maintenance cost. The number of maintenance events that may be avoided is estimated by thresholding failure probabilities of critical components and computing approximate times to failure. The aggregate life-cycle cost can then be calculated, including intangible benefits and the projected cost of CBM. CBM implementation costs include nonrecurring costs, recurring costs, infrastructure costs and nonmonetary considerations and maintenance culture.

7.2 Risk Informed Decision Making

Risk-informed decision-making is the process of using information about risk to assist in decision-making. This would include decisions regarding actions such as: frequency of inspection; need for increased instrumentation; need for additional technical studies; assessment of how uncertainties affect the level of risk; sufficiency of evidence to support the need for remedial action; selection of a remedial action to address an identified deficiency; prioritization of projects or actions; and the sequence in which remedial actions are taken.

Risk-informed decision making is distinguished from risk-based decision making in that it is a fundamentally deliberative process that uses a diverse set of performance measures, along with other considerations, to inform decision making. The process acknowledges the role that human judgment plays in decisions, and that technical information cannot be the sole basis for decision making. This is not only because of inevitable gaps in the technical information, but also because decision making is an inherently subjective, values-based enterprise. In the face of complex decision making involving multiple competing objectives, the cumulative wisdom provided by experienced personnel is essential for integrating technical and nontechnical factors to produce sound decisions (NASA, 2010).

It is crucial for decision maker to have as much as information as possible at the time of decision making to ensure the best possible outcome. In mission critical applications where decisions need to be made in a short period of time, on-line system health monitoring becomes crucial. A complete picture of the state of the system helps prioritize the response action to avoid failures.

The proposed methodology not only provides on-line system health information, the inference pre-computation database gives the decision makers a tool to inquire expected values of system parameters given knowledge about some of the parameters. This is very useful for comparing different available choices and scenarios. And the results can be computed quickly even in a large complex engineering system.

Pre-computation database enables system operator to perform online sensitivity analysis of all components in the system. The analysis computes the overall system reliability given that a particular component is considered to have zero failure probability.

$$p(S)_i = P(S|\{C_1, C_2, \dots, C_i, \dots, C_n\}, p(C_i) = 1)\ for\ i = 1 \dots n \qquad (7\text{-}1)$$

Result of the analysis will show which component degradation process predominantly influents the overall system health, and how much system reliability will increase after a repair or replacement. Combine this information with the costs of components or the repair costs, the operator can make the most cost effective decision to improve the system reliability.

Chapter 8: Example Application

8.1 Unmanned Aerial Vehicle (UAV)

Unmanned aerial vehicle (UAV) is an aircraft without a human pilot aboard. Its flight is controlled either autonomously by onboard computers or by the remote control of a pilot on the ground or in another vehicle.

Since their creation, UAVs have found many uses in police, military, and in some cases, civil applications. Currently, UAVs are most often used for the following tasks:

- Aerial Reconnaissance – UAVs are often used to get aerial video of a remote location, especially where there would be unacceptable risk to the pilot of a manned aircraft.
- Scientific Research – In many cases, scientific research necessitates obtaining data from hazardous, or remote locations.
- Logistics and Transportation – UAVs can be used to carry and deliver a variety of payloads.
- Oil, gas, and mineral exploration and production. UAVs can be used to perform geophysical surveys, in particular geomagnetic surveys. For above-ground pipelines, this monitoring activity could be performed using digital cameras mounted on one or more UAVs.

Some of the first UAVs were called "drones" and were not autonomous, because they required constant control input from a remote human pilot. Computer technology now allows UAVs to make their own decisions, or fly autonomously. Autonomous flight involves the UAV making decisions as it flies.

Generally, autonomous flight consists of the following operations:

- Interpreting sensor input, and merging the input of multiple sensors
- Communicating with ground stations, satellites, and other UAVs and aircraft
- Determining the ideal course to fly for a given mission, based on sensor input.
- Determining the best maneuvers to perform for a given task
- In some cases, cooperating with other UAVs to accomplish a common task.

8.2 System Health Management for UAV

There is a need for advanced health management systems for UAV to be able to do the following, in case of anomalies:

- Can quickly and reliably pinpoint failures,
- Carry out accurate diagnosis of unexpected scenarios, and,
- Based upon the determined root causes, make informed decisions that maximize capabilities to meet mission objectives while maintaining safety requirements and avoiding safety hazards.

This system is perfect for demonstration of the proposed methodology because of the following reasons:

- It’s a mission critical application. Failure is catastrophic.
- It requires high precision information with uncertainties, such as flying path and altitude.
- It’s a fairly complex system. There are many components/functionalities and many types of failure modes.
- There are possibilities of operating under harsh conditions/environment.
- It benefits from remote computation/decision. A powerful onboard computer is not available. With this methodology it only need onboard storage unit.

8.3 Quadcopter

A quadcopter is picked for the example application due to its ubiquity recently in many applications. A quadcopter, also called a quadrotor helicopter, quadrotor is a multirotor helicopter that is lifted and propelled by four rotors. Quadcopters are classified as rotorcraft, as opposed to fixed-wing aircraft, because their lift is generated by a set of rotors.

Unlike most helicopters, quadcopters use 2 sets of identical fixed pitched propellers; 2 clockwise (CW) and 2 counter-clockwise (CCW). These use variation of RPM to control lift and torque. Control of vehicle motion is achieved by altering the rotation rate of one or more rotor discs, thereby changing its torque load and thrust/lift characteristics.

Quadcopter is useful in public safety applications, such as crime or accident investigation, intelligence and evidence gathering, traffic and crowd control, tactical operations (barricade, hostage, raid) search and rescue, fire control and damage assessment, and emergency and disaster response. It also has been use in industrial commercial, such as infrastructure inspection, 3D models and volumetric analysis, precision agriculture, gas leak detection, environmental and wildlife monitoring, photography, site and infrastructure security, and construction site planning and monitoring

8.4 Technical Specifications

The prototype that will be used for technical specifications in this analysis is the Aeryon Scout show in Figure 8-1.

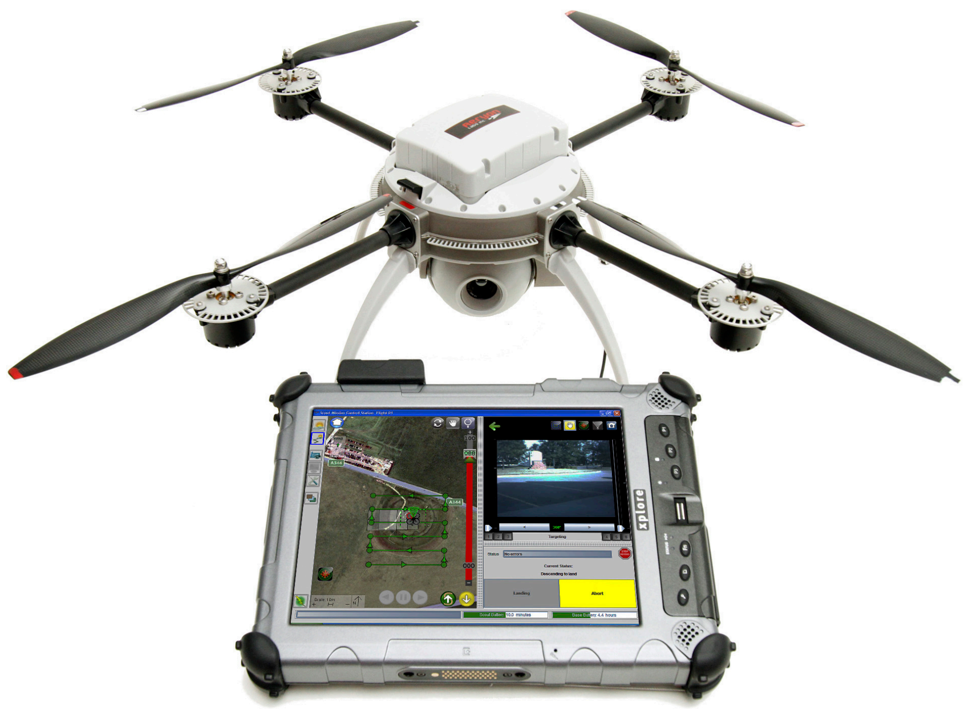


Figure 8-1: The Aeryon Scout.

The Aeryon Scout (Aeryon Labs Inc., 2014) is a Vertical Take-Off and Landing (VTOL) sUAS – ideal for providing an immediate eye in the sky for public safety, commercial and industrial users. The Scout can be operated beyond the line of sight up to 3 kilometres (1.9 miles) from the user. It is controlled with a Tablet PC-based interface and piloted by pointing to an area on the map. The Scout constantly monitors external conditions such as wind speed, as well as internal functions, such as battery level, allowing it to make an automated decisions en route to return home, land immediately or hover and wait.

Table 8-1 shows specifications of the Aeryon Scout that will be used in this analysis.

Table 8-1: Quadcopter prototype technical specifications

Technical Specification	Value
Endurance	Up to 25 minute flight time (with payload)
Wind Tolerance	30 mph (50 kph) sustained 50 mph (80 kph) gusts
Environmental Temperature Range	-22 –122°F (-30 – 50°C)
Line-of-Sight Range	2,500 ft. (750 m) integrated capability
Altitude	1,500 ft. (450 m) AGL 15,000 ft. (4,500 m) MSL
Launch & Recovery Method	Vertical Take-Off and Landing (VTOL)
Dimension	28 in. (72 cm) diameter, 8.5 in. (20 cm) height
Weight	3 lbs (1.4 kg)
Radio Frequencies	900 Mhz, 2.4 GHz
Control and Data Link	Low-latency all-digital network
Security	Secure network pairing, AES 256 bit encryption

8.5 Components and Functionalities

Table 8-2 lists the components of quadcopter and their related functionalities. The table also includes details of each component for use in the analysis.

Table 8-2: Quadcopter components and functionalities

Component	**Functionality**	**Detail**
Frame	The structure that holds all the components together	Carbon fiber, Aluminum alloy connector, Width: 498mm, Height: 80mm, Weight 240g
Propeller	Converting rotational motion into thrust	10x4.5 SF Props 2pc Standard Rotation/2 pc RH Rotation
Rotors	Provide the necessary thrust to propel the craft	Brushless DC motors, 1650rpm/V, 7.2v-11.1v, Max Power: 180w, Max Current: 17.5A, No Load Current: 1.3A, Thrust: 520g
Electronic Speed Controller (ESC)	Motor controller board that has a battery input and a three phase output for the motor	Plush 25 Brushless Speed Controller, Continuous Current 25A, Burst Current: 35A, BEC Mode Linear, BEC: 5v/2A, • Motor speed: (Maximum): 210000 RPM (2 poles), 70000 RPM (6 poles), 35000 RPM (12 poles)

Component	Functionality	Detail
Inertial Measurement Unit (IMU)	An electronic sensor device that measures the velocity, orientation and gravitational forces	Triple Axis Accelerometer & Gyro Breakout, 3-axis gyroscope and a 3-axis accelerometer, onboard Digital Motion ProcessorTM (DMPTM), I2C Digital-output of 6 or 9-axis MotionFusion data, Input Voltage: 2.3 - 3.4V, Tri-Axis angular rate sensor (gyro) with a sensitivity up to 131 LSBs/dps and a full-scale range of ±250, ±500, ±1000, and ±2000dps, Tri-Axis accelerometer with a programmable full scale range of ±2g, ±4g, ±8g and ±16g, Digital-output temperature sensor
Flight Control Board (FCB)	Microcontroller	KK2.0 Multi-rotor LCD Flight Control Board, Atmel Mega324PA 8-bit AVR RISC-based microcontroller with 32k of memory, Piezo buzzer, 6 Pin USBasp AVR Programming interface, Signal from Receiver: 1520us, Signal to ESC: 1520us

Component	Functionality	Detail
Power Distribution Board	Distribute power to ESC and rotors	Gold plated, Current: 4 x 20A Outputs (MAX), Power Input: XT60 with 12AWG wire, Motor output: 4 x 3.5mm Female bullet plug, Aux output: 2 pin JST compatible
Wiring	Transfer power and electronic signal	Pure copper, pure silicone, Temp rating: 200Deg C Guage: AWG 16#
Battery	Power Source	Lithium Polymer (LiPo), 3 Cell (3S) 11.1 volts, 2200MAH, 20C constant discharge rate (30C burst discharge rate), Connector Plug: XT60 connector
RC Transmitter/ Receiver	Transmit/Receive control signal	2.4ghz ISM Frequency Range, Resolution: 1024, Modulation: GFSK, Spread Spectrum Mode: FHSS, Channel: 7 (inc RX Battery Input, Power: 4.5v ~ 9.6v/<30ma

The components can be separated in to 4 categories: flight control system, structure, power, and communication. Figure 8-3 presents a diagram showing the categories and the relationships among components.

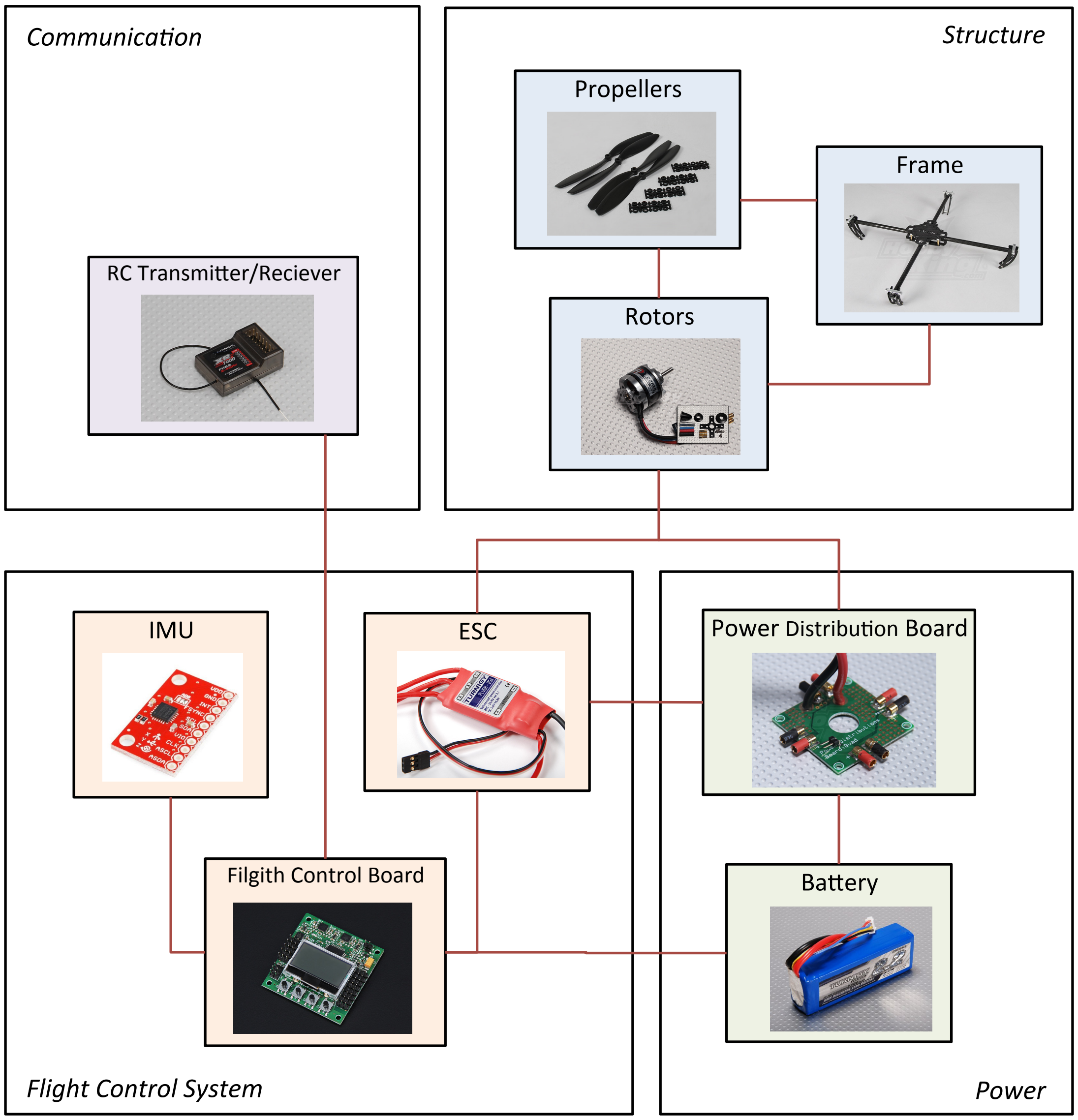


Figure 8-2: Diagram of components in a quadcopter.

8.6 Component Failure Modes and Failure Mechanisms

The first step is to identify the failure modes and their factors within the system. For each component, the dominant failure modes and mechanisms are shown in Table 8-3.

Table 8-3: Quadcopter component failure modes

Component	Failure Modes	Failure Mechanisms	Factors
Frame	Structure Failure Mechanical Fastener Failure	Fatigue-induced Fracture (fr-fat) Creep-induced Fracture (fr-cr)	Stress Vibration Temperature
Propeller	Loss of function Overspeeding Overtorque Blade/propeller separation	Fatigue-induced Fracture (pr-fat) Creep-induced Fracture (pr-cr) Corrosion-induced Failures (pr-co)	Cycle stress Moisture Vibration Temperature
Rotor	Actuator Fault Overheating Motor shaft failure Bearing failure Rotor failure Misalignment Overspeeding	Fatigue (rot-fat) Corrosion (rot-co) Thermal Expansion (rot-te)	Moisture Vibration Temperature Cycle stress Current Transient

Component	Failure Modes	Failure Mechanisms	Factors
Electronic Speed Controller (ESC)	Burnout Conduction failure Software failure Overvoltage	Electromigration (esc-em) Thermal Expansion (esc-te)	Voltage Vibration Temperature
Inertial Measurement Unit (IMU)	Sensor Fault Loss of signal Incorrect signal Calibration Error	Stress Migration (imu-sm) Corrosion (imu-co)	Vibration Temperature Humidity
Flight Control Board	Loss of Control Board Overheating Software/Firmware Errors Overvoltage	Electromigration (fcb-em) Stress Migration (fcb-sm) Time-Dependent Dielectric Breakdown (fcb-tddb) Hot-Carrier Injection (fcb-hci)	Temperature Voltage Vibration Humidity Electric field Current Tensile stress
Battery	Overheating Depletion Electrode Cracking	Corrosion (bat-co) Thermal Expansion (bat-te)	Voltage Temperature Vibration Humidity

Component	**Failure Modes**	**Failure Mechanisms**	**Factors**
RC Transmitter/ Receiver	Loss of transmitter Overvoltage	Electromigration (rc-em) Time-Dependent Dielectric Breakdown (rc-tddb) Hot-Carrier Injection (rc-hci)	Current Electric field Voltage Temperature
Wiring	Cable Failures Connector Failure Short Circuits Excessive Ohmic heating Overvoltage	Electromigration (wi-em) Corrosion (wi-co) Thermal Expansion (wi-te)	Vibration Temperature Voltage Current Humidity

8.7 Bayesian Network Modeling

Using the component functionalities and their failure modes, we can construct the BN for this quadcopter, shown in Figure 8-3. The nodes can be identified using the following abbreviations.

Functionalities: system (sys), flight related (flight), structure support (struct), flight control (contr), electronics (elec), power (pow), communication (com).

Components: frame (fr), propeller (prop, pr), rotor (rot), electronic speed controller (esc), inertia measurement unit (imu), flight control board (fcb), wire (wi), battery (bat), remote control transmitter/receiver (rc).

Degradation mechanisms: fatigue (fat), creep (cr), corrosion (co), thermal expansion (te), electromigration (em), stress migration (sm), time-dependent dielectric breakdown (tddb), hot-carrier injection (hci).

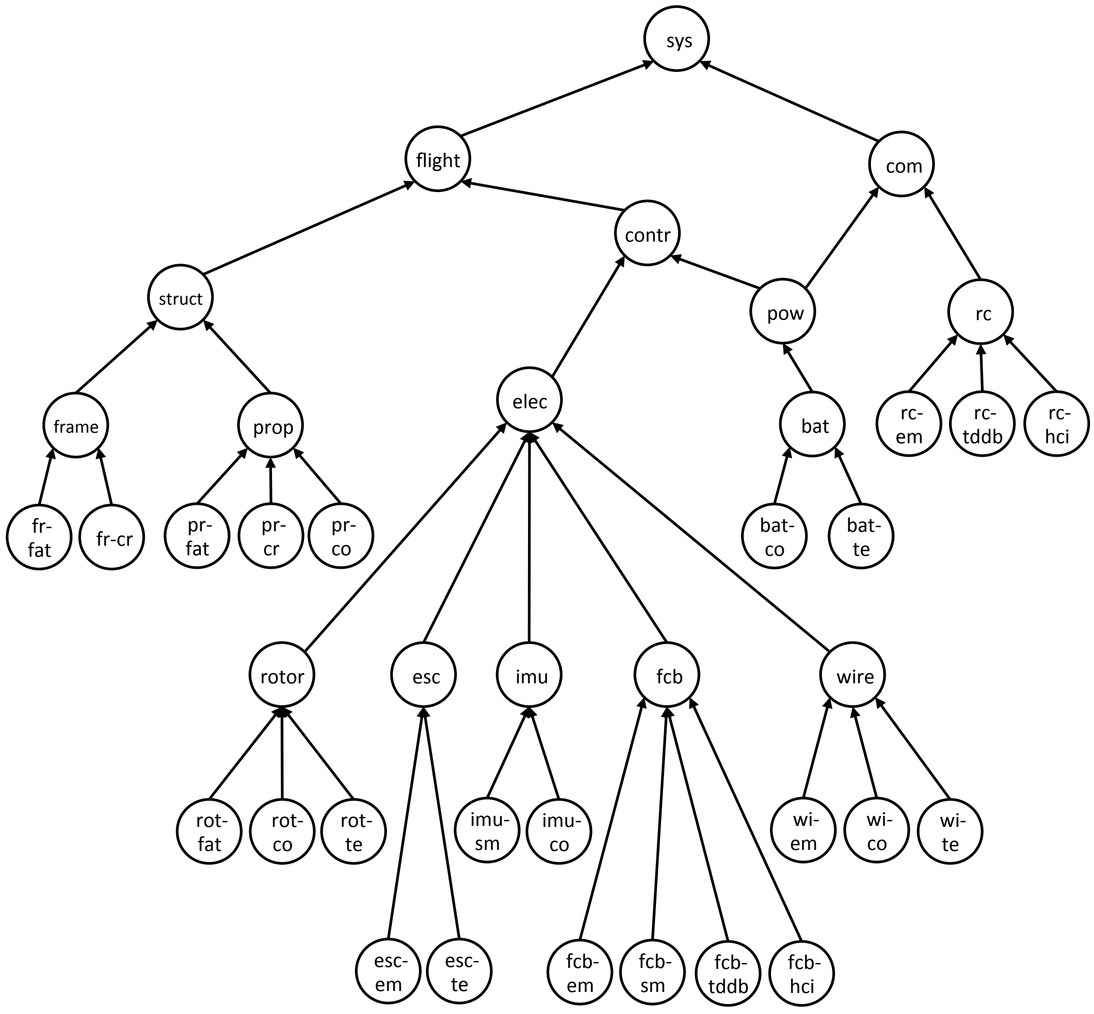
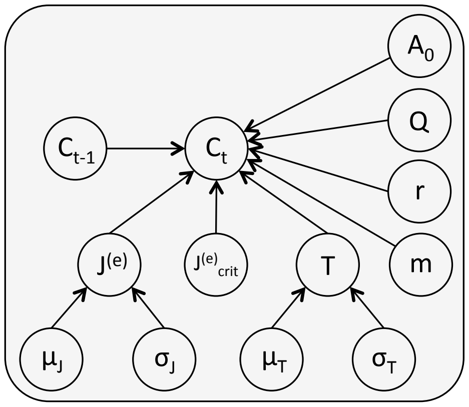
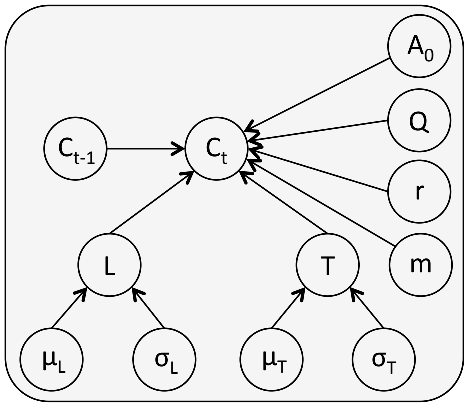
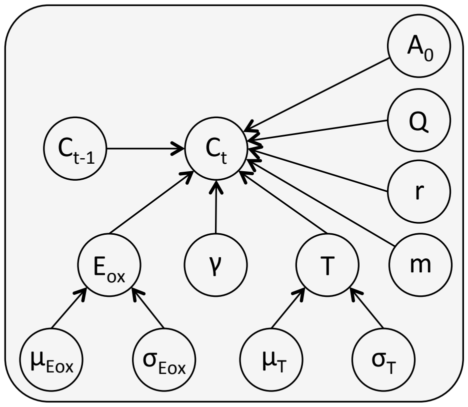
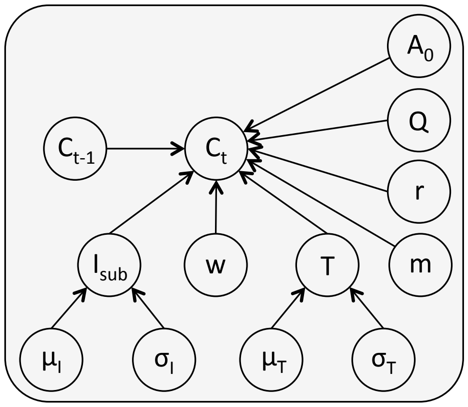


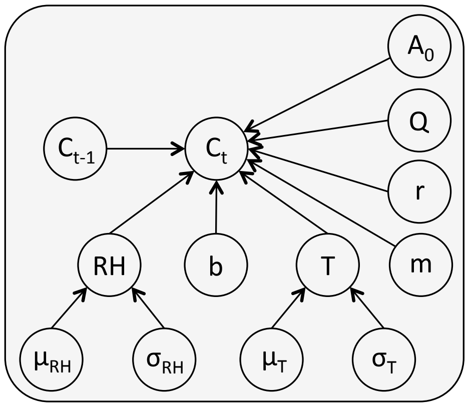
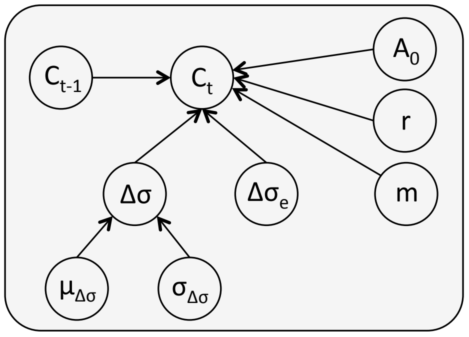
Figure 8-3: Quadcopter BN model.

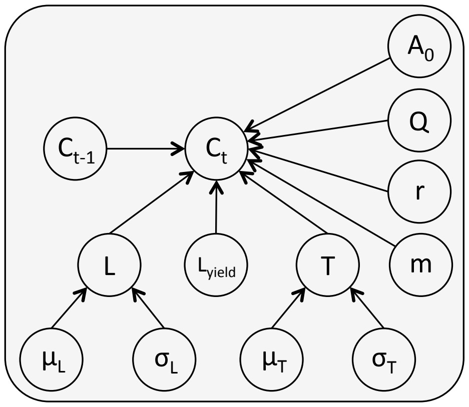
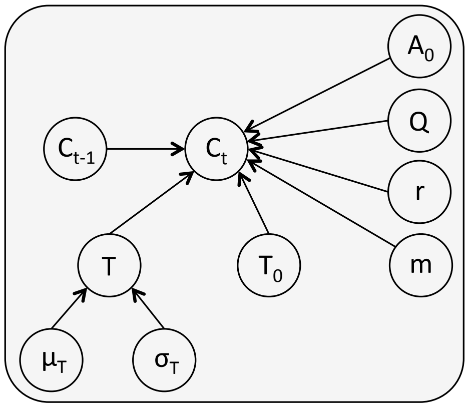
Table 8-4 shows each failure mode BN with their degradation function and explanation of the related factors.

Table 8-4: Failure mode BN and degradation function

Failure Mechanism	Degradation Function
Electromigration  Figure 8-4: Electromigration BN	$C = C_0\left[1 - A_0\left(J^{(e)} - J^{(e)}_{crit}\right)^r \mathrm{e}^{\left(\frac{-Q}{K_B T}\right)} t^m\right]$ • A_0 is a process/material-dependent coefficient • $J^{(e)}$ is the electron current density • $J^{(e)}_{crit}$ is a critical (threshold) current density • r is an empirically determined exponent • Q is the activation energy
Stress Migration  Figure 8-5: Stress migration BN.	$C = C_0\left[1 - A_0 L^r \mathrm{e}^{\left(\frac{-Q}{K_B T}\right)} t^m\right]$ • A_0 is a process/material-dependent coefficient • L is the tensile stress in the metal for a constant strain • r is an empirically determined exponent • Q is the activation energy

Failure Mechanism	**Degradation Function**
Time-Dependent Dielectric Breakdown (TDDB)  A0 Q r m Ct-1 Ct Eox γ T μEox σEox μT σT Figure 8-6: TDDB BN.	$C = C_0\left[1 - A_0 e^{(\gamma E_{ox})} e^{\left(\frac{-Q}{K_B T}\right)} t^m\right]$ • A_0 is a process/material-dependent parameter • γ is the field-acceleration parameter • E_{ox} is the electric field in the oxide and is given by the voltage dropped V_{ox} across the dielectric divided by the oxide thickness t_{ox} • Q is the activation energy
Hot-Carrier Injection (HCI)  A0 Q r m Ct-1 Ct Isub w T μI σI μT σT Figure 8-7: HCI BN.	$C = C_0\left[1 - A_0 \left(\frac{I_{sub}}{w}\right)^r e^{\left(\frac{-Q}{K_B T}\right)} t^m\right]$ • A_0 is a process/material-dependent coefficient • I_{sub} the peak substrate current during stressing • w is the width of the transistor • r is an empirically determined exponent • Q is the activation energy

Failure Mechanism	**Degradation Function**
Corrosion  A₀ Q C_{t-1} C_t r RH b T m μ_{RH} σ_{RH} μ_T σ_T Figure 8-8: Corrosion BN.	$C = C_0\left[1 - A_0 \mathrm{e}^{\left(-\frac{b}{RH}\right)} \mathrm{e}^{\left(\frac{-Q}{K_B T}\right)} t^m\right]$ • A_0 is a process/material-dependent parameter • b is the reciprocal humidity dependence parameter • RH is the relative humidity expressed as a % • Q is the activation energy
Fatigue  C_{t-1} C_t A₀ r Δσ Δσ_e m μ_{Δσ} σ_{Δσ} Figure 8-9: Fatigue BN.	$C = C_0[1 - A_0(\Delta\sigma - \Delta\sigma_{elastic})^r t^m]$ • A_0 is a process/material-dependent parameter • $\Delta\sigma$ is the total stress range • $\Delta\sigma_{elastic}$ is the portion of total stress range that is in elastic region • r is an empirically determined exponent

Failure Mechanism	**Degradation Function**
Creep Induced Failure  A₀ Q C_{t-1} C_t r L L_{yield} T m μ_L σ_L μ_T σ_T Figure 8-10: Creep induced failure BN.	$C = C_0\left[1 - A_0\left(L - L_{yield} \right)^r \mathrm{e}^{\left(\frac{-Q}{K_B T}\right)} t^m\right]$ • A_0 is a process/material-dependent coefficient • L is the constant applying stress • L_{yield} is the yielding stress • r is an empirically determined exponent • Q is the activation energy
Thermal Expansion  A₀ Q C_{t-1} C_t r T T₀ m μ_T σ_T Figure 8-11: Thermal Expansion BN.	$C = C_0\left[1 - A_0(T - T_0)^r \mathrm{e}^{\left(\frac{-Q}{K_B T}\right)} t^m\right]$ • A_0 is a process/material-dependent coefficient • T_0 is the temperature at which zero stress exists in the material • r is an empirically determined exponent • Q is the activation energy

8.8 Sensor Data

There are sensors in both the IMU and flight control board that measure temperature, vibration, current, voltage, and humidity. Strain gauge can be installed on the frame and propeller to measure stress. Table 8-5 and Table 8-6 shows the sensor data and there ranges for this applications. Flight control board collects data from the sensors at the sampling rate of 10 Hz.

Table 8-5: Internal sensor data

Sensor/Data	**Unit**	**Range**
Voltage	Volt	0-15
Current	A	0-25
Propeller Rotation Speed	RPM	0-1650
Battery Storage	mAh	0-2200

Table 8-6: External sensor data

Sensor/Data	**Unit**	**Range**
Temperature	Celsius	-25 - 100
Humidity	% RH	0 - 100
Mechanical Stress	N/m^2	1×10^8 - 8×10^8

With the above sensor information, we can identify observables and unobservables in the system. Assuming all components experience the same temperature and humidity in the air, and the same vibration throughout the structure, these factors are classified as common and observables. Voltage, current, and stress are different for each component. Some of them are observables sand some of them are unobservables.

8.9 Pre-computation

The next step is to do inference pre-computation for unobservable values from all the possible observable values. Figure 8-12 show examples of pre-computation inference results for the critical failure parameters of all the failure modes in the quadcopter model at the over the range of external factors. Model parameters used in the calculations are derived from various empirical sources (McPherson, 2010) (Ashby & Jones, 2005) (Anderson, 1995) (Silbey & Alberty, 2001) (Sze, 2002).

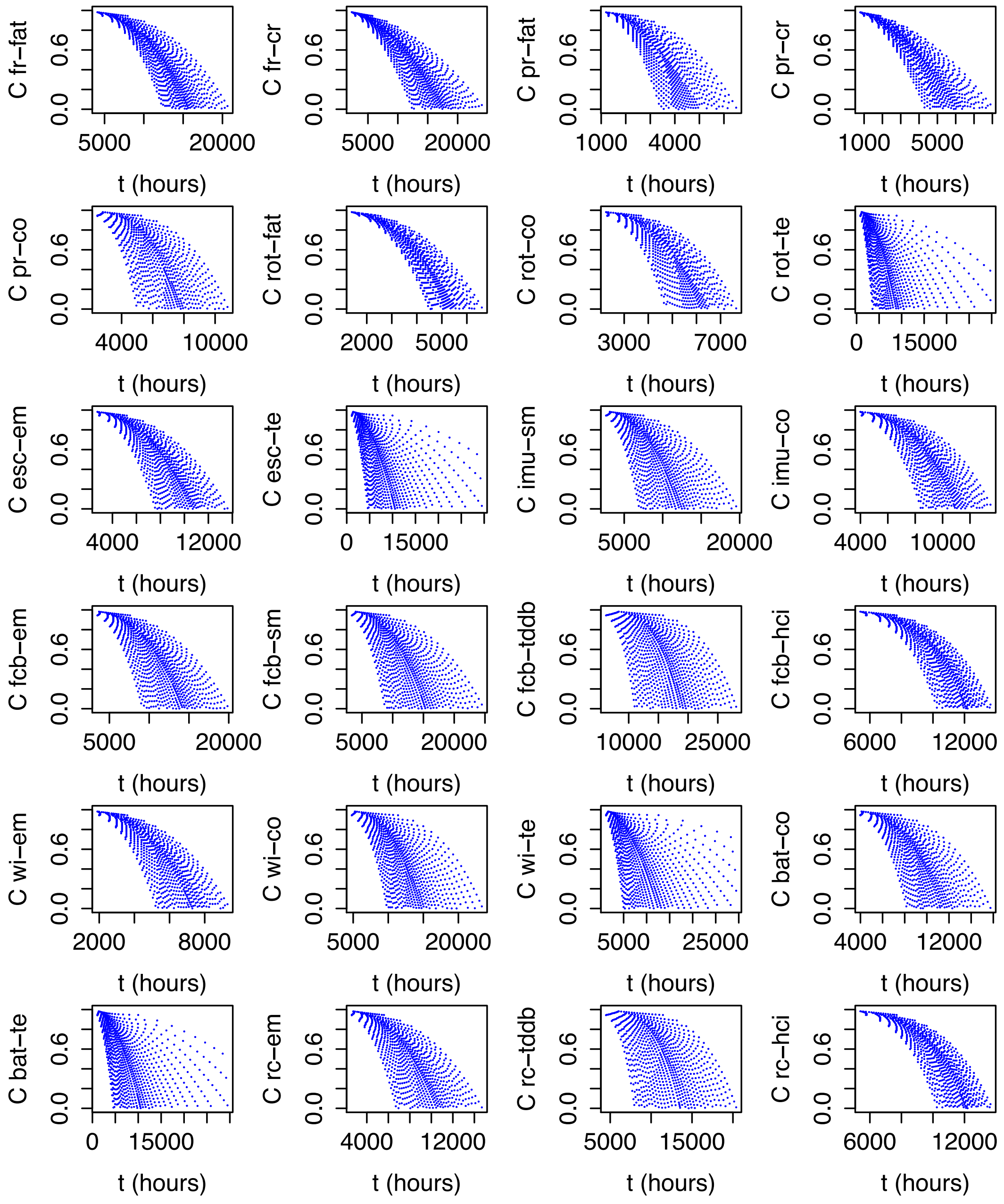


Figure 8-12: Example of quadcopter pre-computation results over the ranges of factors

8.10 System Monitoring

With the inference pre-computation database, we have complete information about the state of the system over time given data of the observables. Figure 8-13, Figure 8-14, and Figure 8-15 show the reliability of components, functionality probabilities, and system reliability, respectively. All of the plots show results from two different sets of data; one is under normal conditions, and the other is under 100 hours temperature and humidity cycle (T = 25-75°C, RH = 50-100%).

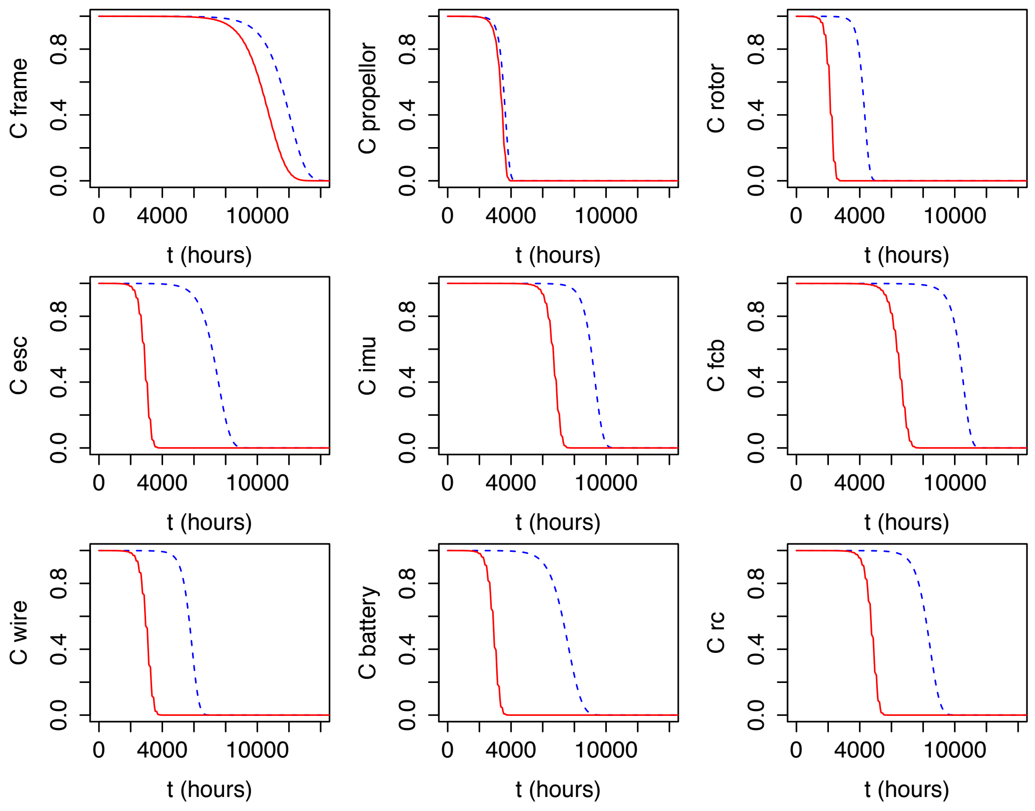


Figure 8-13: Quadcopter component reliability with normal operation data (blue) and temperature/humidity cycle data (red).

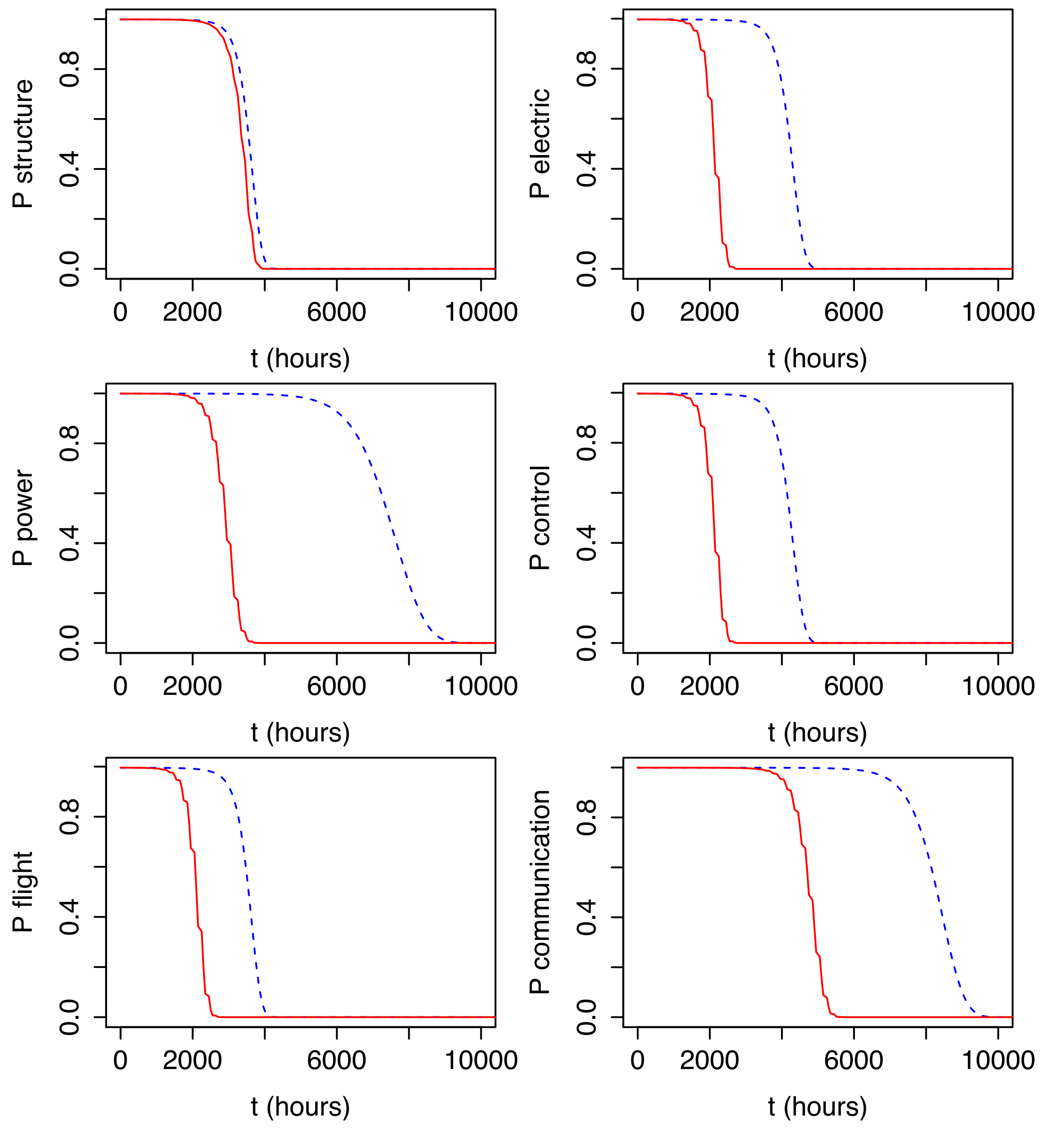


Figure 8-14: Quadcopter functional probabilities with normal operation data (blue) and temperature/humidity cycle data (red).

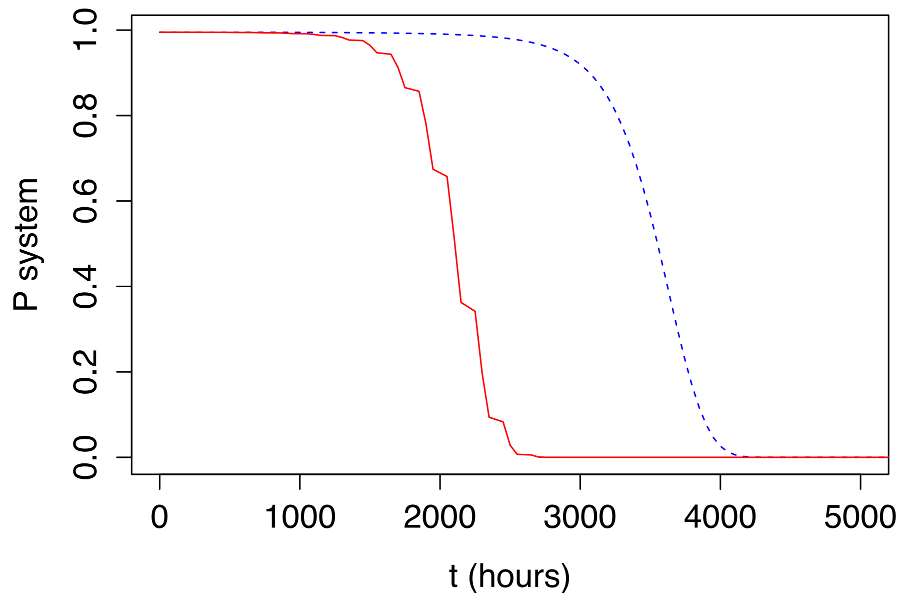


Figure 8-15: Quadcopter system reliability with normal operation data (blue) and temperature/humidity cycle data (red).

8.11 Remaining Useful Life

With the inference pre-computation database, searching through the data, given the trajectory of parameter values provide RUL estimation of all components at any point in time. Using the previous temperature/humidity cycle data example, Figure 8-16 shows RUL estimates of all 9 components in the system with uncertainties.

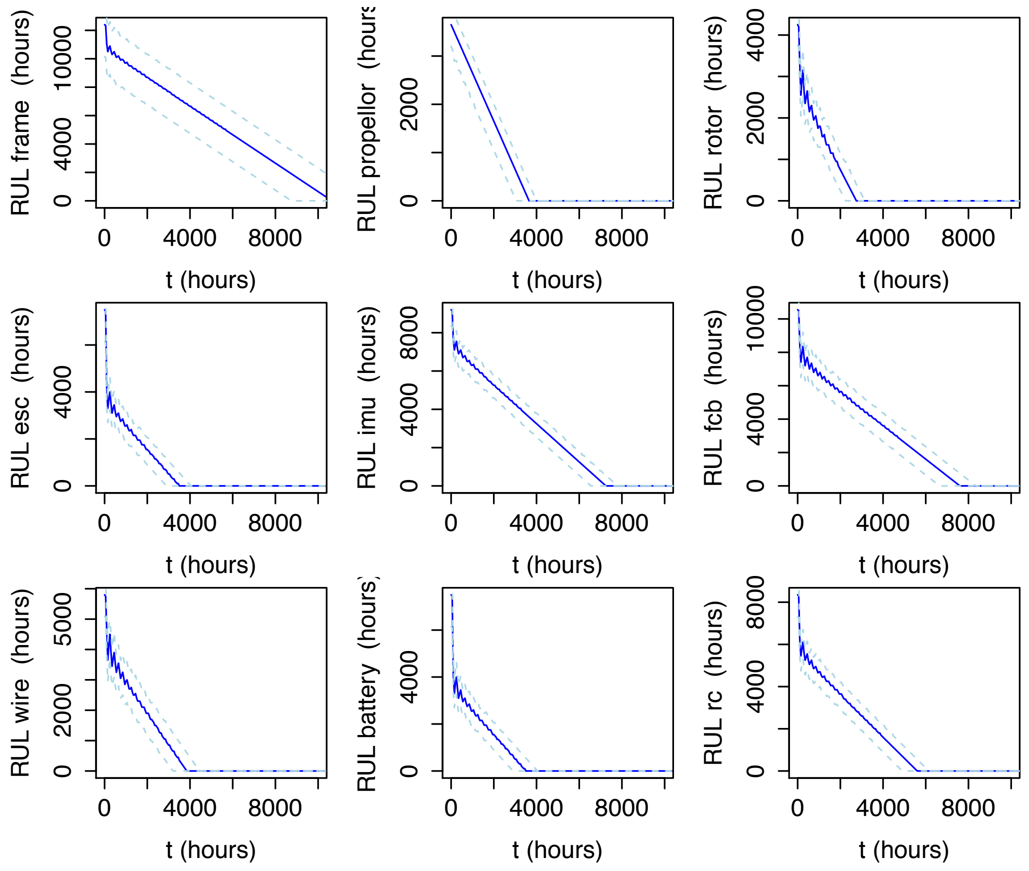


Figure 8-16: Quadcopter component RUL estimates with 5th and 95th percentile uncertainties.

Notice that the RUL estimates becomes steadier over time as it gains more trajectories information, even though the data cycle remains the same.

8.12 Anomaly Detection

With the RUL information, the system is able to monitor and detect when a component reaches certain RUL threshold and flag it for maintenance. The system is also capable of detecting an anomaly where an observable is out of its expected operating range and provide an upper bound of the RUL of the component. For example, the expected possible temperature range of the quadcopter is -25 to 100°C. If a user operate the quadcopter at temperature higher than 100°C, the system will detect that the temperature is out of range and provide the updated maximum RUL estimate. This is crucial information to have when using the system in extreme conditions.

Besides detecting and predicting failures, a periodic measurement of normally unobservable parameters in the system can be used to detect model errors. Figure 8-17 shows a plot of fatigue degradation model of the quadcopter frame and a series of visual crack measurements translated to component critical parameter scale.

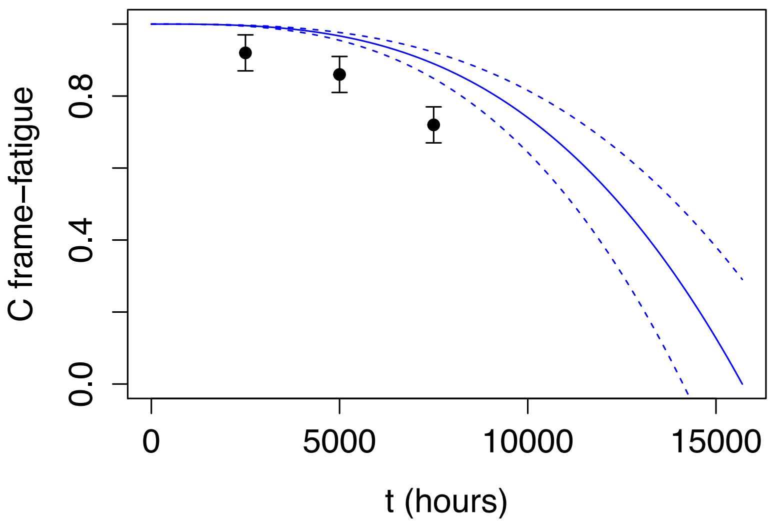


Figure 8-17: Quadcopter frame fatigue model estimations (blue line) and periodic measurements (black dots) with uncertainties.

Anomaly is detected when the differences of model estimated and the measured values are higher than the expected measurement uncertainties. The frame is subjected to 400 MN/m^2, with C expected to be 0.996, 0.969, 894 at 2500, 5000, and 7500 hours respectively. However, C is measured to be 0.92, 0.86, and 0.72 with possible 0.05 measurement uncertainty instead. When anomaly is detected, the component is then flagged for system learning.

8.13 Learning

There are three possible model errors to be considered for system learning in order to correct the model according to the evidences. First is error of the sensor or measurement device. In this case, the differences between measurement values and model values are consistent. Figure 8-18 shows fatigue degradation of the quadcopter frame with $\Delta C = 0.1$ measurement error.

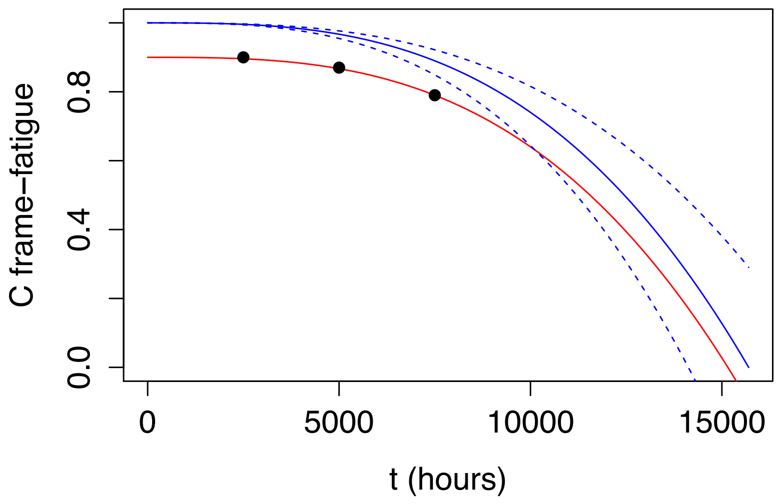


Figure 8-18: Quadcopter frame fatigue model estimations (blue line), periodic measurements (black dots), updated model estimation with measurement error (red line).

Second is error in model parameters. In this case, actual degradation process appears to be faster or slower than the expected model degradation. The model parameters can be updated using the parameter learning algorithm with the measurement values evidence. Figure 8-19 shows fatigue degradation of the quadcopter frame where the updated model A_0 is twice the value of expected model A_0.

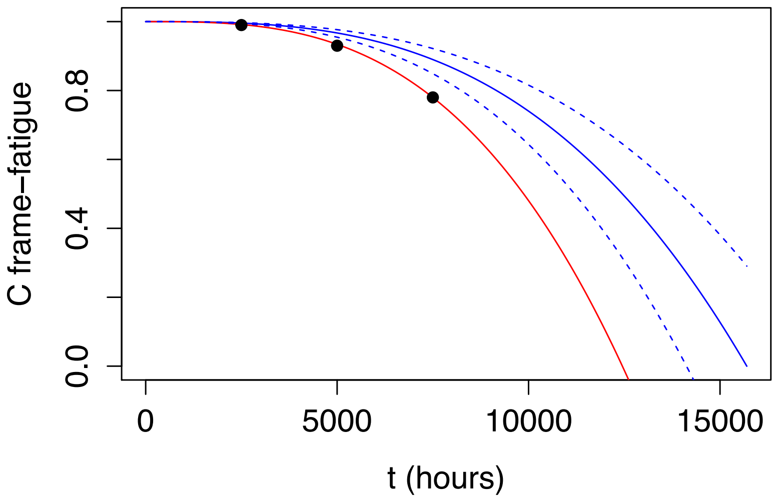


Figure 8-19: Quadcopter frame fatigue model estimations (blue line), periodic measurements (black dots), updated model estimation with parameter learning (red line).

Third is error in model structure. In this case, degradation process changes according to an outside factor that was not included in the model. The model structure can be updated using the structure learning algorithm to find hidden relationship between other observables in the system with this degradation process. Figure 8-20 shows fatigue degradation of the quadcopter frame where fatigue degradation is affected by relative humidity of 100% during the time between 2500 and 500 hours.

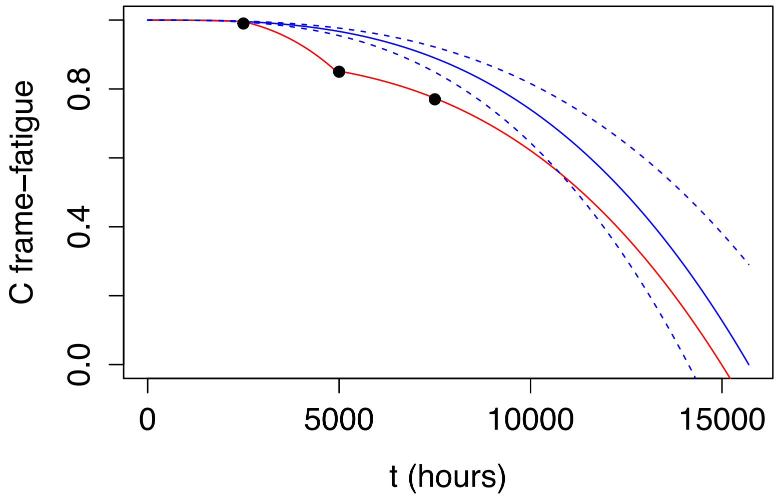


Figure 8-20: Quadcopter frame fatigue model estimations (blue line), periodic measurements (black dots), updated model estimation with structure learning (red line).

8.14 Maintenance

By comparing RUL estimates over the range of operating conditions, the operator is able plan maintenance for each individual component according to the expected use. Figure 8-21 shows each component RUL estimates at $t = 0$ over a range of temperature from 0 to 100°C. This plot provides information for maintenance planning and comparison between component degradation processes.

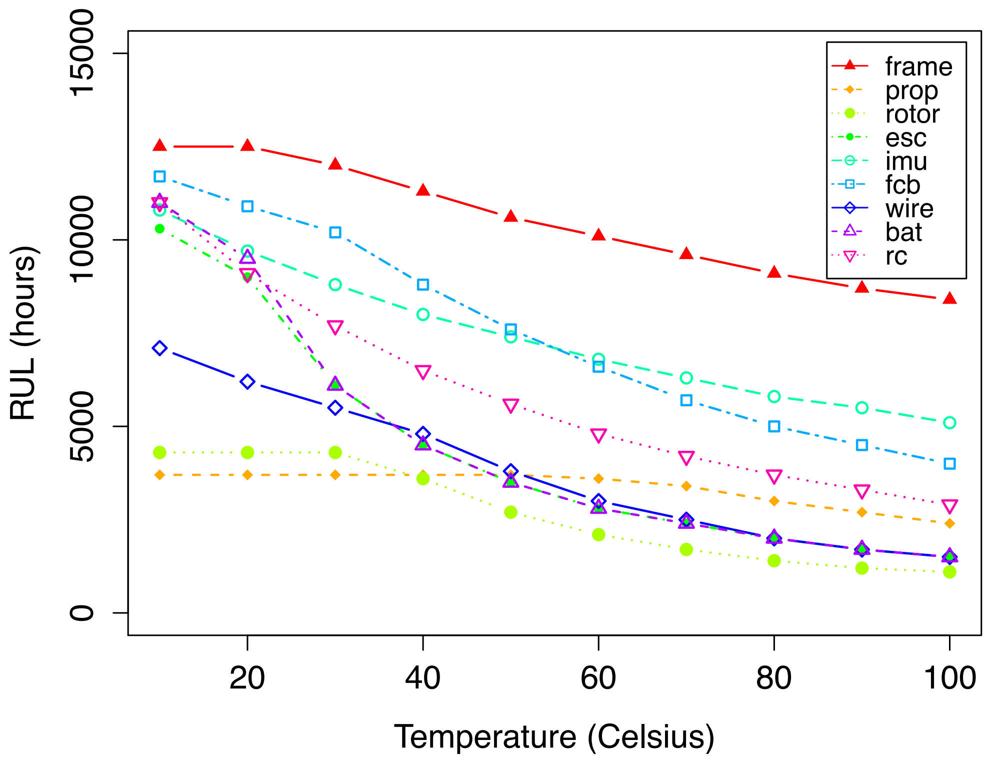


Figure 8-21: Quadcopter components RUL estimates at t = 0 over a range of operating temperature.

During an online operation, sensitivity analysis can be performed at any point in time to identify which component mostly affects reliability of the system. Figure 8-22 shows sensitivity analysis of the system operating under a normal condition at different time. Each point represents system reliability probability given that the specific component has zero probability of failure. The operator can use this information to decide which component replacement is the most cost-effective to increase the system reliability.

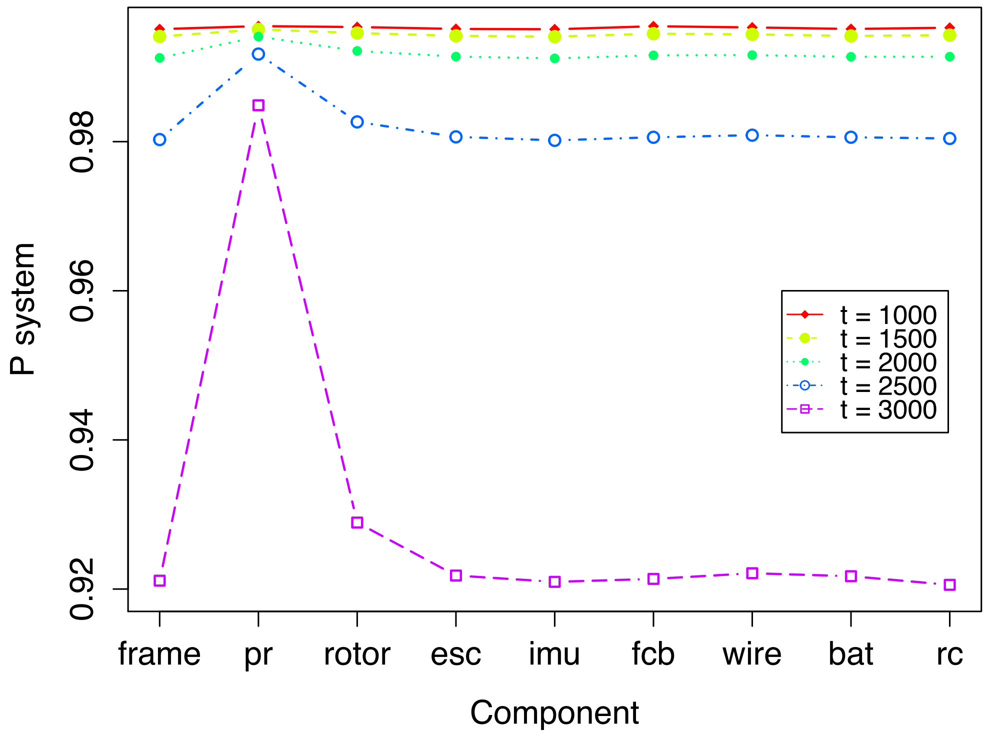


Figure 8-22: Quadcopter sensitivity analysis at different time during a normal operating condition.

Chapter 9: Conclusion

9.1 Summary

This research presents new modeling approach, computational algorithms, and an example application for on-line System Health Management. Hybrid dynamic Bayesian Network modeling is introduced to represent complex engineering systems in a way that it allows accurate representation of underlying physics of failure by using empirical degradation model with continuous variables. The proposed computational algorithms enable on-line monitoring and diagnosing complex systems by utilizing pre-computation and dynamic programming methods with Markov Chain Monte Carlo inference. Pre-computation inference database can then be used for efficient continuous health monitoring, probabilistic prognosis of remaining useful life, and anomaly detection with pattern recognition. Algorithm for system parameter and structure learning is also included along with methods for maintenance decision-making.

Main advantages of the proposed methodology:

- Systematic approach based on physics of failure makes it easy to be modeled and implemented in any engineering system.
- The model allows different types of information to be combined together and includes complete information about the uncertainties.

- The method can be used in many applications from small to large scale and at any location, since it does not required powerful computation unit during an operation.
- Possible applications include nuclear power plant, oil pipeline, automobile, aircraft, and any mission-critical system that requires on-line health monitoring.

9.2 Contributions

The main contributions of this work can be summarized as follows:

- Introduced a new modeling approach using hybrid DBN.
    a. Proposed a systematic modeling of SHM with 5 different layers to include both higher-level discrete functional probability part and lower-level continuous component critical parameters related to degradation failure modes.
    b. Created a well-defined interface between discrete and continuous variables that allows tractable approximate MCMC inference of any variable in the network.
- Developed computational algorithm for on-line monitoring and diagnosing of complex systems.
    a. Developed inference pre-computation algorithm to store pre-computed values of unobservables to allow instantaneous inquiry of system

health.

b. Applied dynamic programming algorithm to significantly reduce the overall computation time and complexity of pre-computation process.

c. Implemented an efficient algorithm for dependency between variables in the network.

- Used the proposed modeling approach and computational algorithm for on-line system monitoring and prognosis.

    a. Created efficient method for monitoring of system health and component status to detect any anomaly and predict remaining useful life.

    b. Implemented pattern matching algorithm for failure trajectories to identify possible failure earlier than the traditional method.

    c. Applied BN learning algorithm to give the system continuous update of network parameters and structure from data obtained during operation.

    d. Showed that the proposed method provides information to improve on-line decision-making for system maintenance or in an event that a critical failure occurs.

- Demonstrated the capabilities of this methodology by applying it to an unmanned aerial vehicle application.

9.3 Suggested Future Research

- Recode the prototype R program in Java to improve efficiency and reduce computation time. Java is also cross-platform, so the program can be run in any operating system with Java runtime environment.
- Implement Graphical User Interface (GUI) for the software.
- Add built-in models for dominant failure modes/mechanisms of common materials/components.

Appendices

Appendix A: Metropolis-Hasting Algorithm

Metropolis-Hastings algorithm (Hastings, 1970) structure is extremely simple and can be described by the following steps:

```
1. Initialize $X_0$; set $t = 0$
2. Repeat {
    a. Sample a point Y from $q(.|X_t)$
    b. Sample a Uniform (0,1) random variable U
    c. If ($U <= \alpha(X_t,Y)$) set $X_{t+1} = Y$
    d. Otherwise set $X_{t+1} = X_t$
    e. Increment t }.
```

Here, Y is defined as a *candidate* point from a *proposal* distribution $q(.|X_t)$, where this distribution can have any form and the stationary distribution of the chain will be π.

Furthermore, the candidate Y can be chose by a probability α.

$$\alpha(X,Y) = min\left(1, \frac{\pi(Y)q(X|Y)}{\pi(X)q(Y|X)}\right)$$

Thus, it follows that:

$$\pi(X_t)q(X_{t+1}|X_t)\alpha(X_t,X_{t+1}) = \pi(X_{t+1})q(X_t|X_{t+1})\alpha(X_{t+1},X_t)$$

And now we can obtain the detailed balance equation.

$$\pi(X_t)P(X_{t+1}|X_t) = \pi(X_{t+1})P(X_t|X_{t+1})$$

Finally, integrating both sides of above equation with respect to X_t we have:

$$\int \pi(X_t)P(X_{t+1}|X_t)\, dX_t = \pi(X_{t+1})$$

The marginal distribution of X_{t+1} is given by the equation above under the assumption that X_t is from π. Thus, we can conclude that if X_t is from π, then X_{t+1} will be also. However, this only proves that the stationary distribution is π. It is necessary to prove that $P^{(t)}(X_t|X_0)$ will converge to the stationary distribution for a more complete justification.

The first step in Metropolis-Hastings algorithms is to generate a candidate point, denoted here by θ^*. Often, the candidate point differs from the current value of the parameter in only one or two components; for example, in the Weibull example, we may alternate between updating the value of α and the value of β.

Some common method for generating the candidate value is via a uniform density or normal density. In the case Weibull example, we can choose a uniform density and a normal distribution with a modification to draw α and β.

In the case of using normal density in generating candidate θ^* we add mean-zero normal deviate to a single component of $\theta^{(j-1)}$, say $\theta_i^{(j-1)}$. This means that we can express the candidate value θ^* as the vector with components

$$\theta^* = \theta_i^{(j\text{-}1)} + sZ,$$

$$\theta^* = \theta_k^{(j\text{-}1)}, \quad \text{for } k \neq i,$$

where Z is a standard normal deviate and s is an arbitrary constant. For continuous-valued components of the parameter vector, let $f(\theta^*|\theta_i^{(j\text{-}1)})$ denote the *proposal density* used to generate θ^* from $\theta^{(j\text{-}1)}$. For example, the proposal density $f(\cdot)$ represents a normal distribution with mean $\theta_i^{(j\text{-}1)}$ and standard deviation s. For discrete-valued components of the parameter vector, $f(\ldots)$ represents the probability mass function used to generate candidate points. The probability of moving from the candidate point back to the original value is denoted, in a similar way, by $f(\theta^{(j\text{-}1)}|\theta^*)$.

If we use normal density for generating a candidate in the Weibull example, we have to modify the generating density because we want the proposed values of α and β to have positive values. One way of simulating positive candidate values is to generate candidates on the logarithmic scale and then transform them to the original scale.

That is, we might define candidate draws for α and β according to

$$\log(\alpha^*) = \log(\alpha)^{(j\text{-}1)} + v_\alpha \text{ where } v_\alpha \text{ is sampled from Normal}(0,\sigma_\alpha)$$

$$\log(\beta^*) = \log(\beta)^{(j\text{-}1)} + v_\beta \text{ where } v_\beta \text{ is sampled from Normal}(0,\sigma_\beta)$$

In theory, any density of mass function can serve as the proposal density as long as it satisfies three conditions. First, the proposal density must allow us to move from any subset of the parameter space to any other subset of the parameter space in a finite number of moves. Second, the proposal density cannot be periodic. Informally this means that, in the long run, moves to any subset of the parameter space can occur at any time. Finally, we require that the rule used to specify the proposal density satisfies:

$$0 < \frac{f\left(\theta^{*}|\theta^{(j-1)}\right)}{f(\theta^{(j-1)}|\theta^{*})} < \infty$$

for all values $\theta^{(j-1)}$ and θ^{*}.

Having generated a candidate point θ^{*}, we perform the second step in a Metropolis-Hastings algorithm; we compute the probability that the candidate value will be accepted as the next simulated value in the sequence. We call this quantity the *acceptance probability* and denote its value by *r*. With this notation, the acceptance probability *r* is defined as

$$r = min\left(1, \frac{p(\theta^{*}|data)}{p(\theta^{(j-1)}|data)} \frac{f\left(\theta^{(j-1)}|\theta^{*}\right)}{f(\theta^{*}|\theta^{(j-1)})}\right)$$

In this formula, the acceptance probability represents the product of the ratio of the posterior density evaluated at the candidate and current parameter values, $p(\theta^{*}|data)/\, p\left(\theta^{(j-1)}|data\right)$, and the ratio of the proposal densities of the current and candidate point, $f\left(\theta^{(j-1)}|\theta^{*}\right)/\, f\left(\theta^{*}|\theta^{(j-1)}\right)$. The first ratio encourages the algorithm to move to parameter values that have high posterior probability, and the second ratio accounts for the fact that the proposal density might favor some values of the parameter over others. Note that if the proposal density is symmetric – that is, if $f(\theta^{(j-1)}|\theta^*) = f(\theta^*|\theta^{(j-1)})$ – this second ratio is 1 and can be omitted from the formula for the acceptance probability.

Having computed an acceptance probability, we perform the third step in a Metropolis-Hastings algorithm. We accept or reject the candidate point with probability equal to *r*. To do so, we draw a *Uniform* (0,1) random variable, say u, and compare *u* to *r*. If $u \leq r$, then we accept the candidate value and set $\theta^{(j)} = \theta^*$. On the other hand, if $u > r$, then we reject the candidate value and set $\theta^{(j)} = \theta^{(j-1)}$ (that is, we keep the same value). This process is repeated for each component of θ.

The Metropolis algorithm is very general and broadly applicable. One problem with it, however, is that the proposal distribution must be properly tuned to the posterior distribution if the algorithm is to work well. If the proposal distribution is too narrow or too broad, a large proportion of proposed jumps will be rejected and/or the trajectory will get bogged down in a localized region of the parameter space.

Appendix B: Gibbs sampling

Gibbs sampling is a type of Markov chain Monte Carlo process where a random walk starts at some arbitrary point, and at each point in the walk, the next step depends only on the current position, and on no previous positions. The difference to other random walk algorithms, such as Metropolis, is that at each point in the walk, one of the component parameters is selected. The component parameter can be selected at random, but typically the parameters are cycled through in order. Gibbs sampling then choose a new value for that parameter, by generating a random value directly from the conditional probability against the other parameters.

$$p\left(\theta_i|\left\{\theta_{j\neq i}\right\}, D\right)$$

Where θ_i is the selected parameter.

Gibbs sampling is especially useful when the complete joint posterior $p(\theta_i, D)$ cannot be analytically determined and cannot be directed sampled, but all the conditional distributions, $p\left(\theta_i|\left\{\theta_{j\neq i}\right\}, D\right)$, can be determined and directly sampled.

Let $\theta^{(j)}$ be a set of parameter values at each step of the walk, the Gibbs sampler algorithm is as follow:

1. Specify initial values $\theta^{(0)} = \left(\theta_1^{(0)}, \dots, \theta_p^{(0)}\right)$
2. Repeat for $j = 1, 2, \dots, M$

   Generate $\theta_1^{(j+1)}$ from $p\left(\theta_1|\theta_2^{(j)}, \theta_3^{(j)}, \dots, \theta_p^{(j)}\right)$

   Generate $\theta_2^{(j+1)}$ from $p\left(\theta_2|\theta_1^{(j+1)}, \theta_3^{(j)}, \dots, \theta_p^{(j)}\right)$

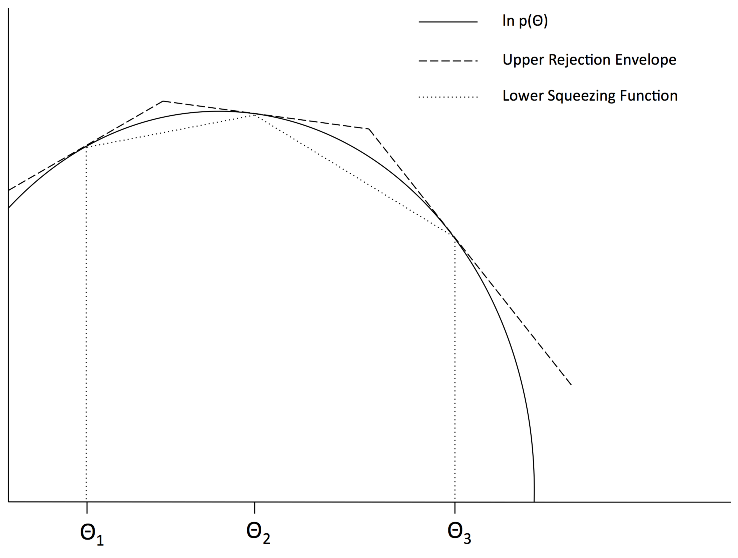
   $\vdots$

   Generate $\theta_p^{(j+1)}$ from $p\left(\theta_p|\theta_1^{(j+1)}, \theta_2^{(j+1)}, \dots, \theta_{p-1}^{(j+1)}\right)$
3. Return the value $\left\{\theta^{(1)}, \theta^{(2)}, \dots, \theta^{(M)}\right\}$

Appendix C: Adaptive Rejection Sampling

In the cases where evaluation of full conditional density is computationally expensive, Adaptive Rejection Sampling (ARS) can be used to draw samples with fewer evaluations. The algorithm only works with probability density functions that are log-concave, which is usually the case for Bayesian applications.

The basic idea of this algorithm is to adaptively form an upper envelope and lower squeezing functions, which creates upper and lower bounds of $p(\theta)$. The envelope and squeezing functions are formed using sets of piecewise exponential distributions. These sets contain segments of one or more exponential distributions attaching from end to end. It can be easier to visualized in log space where logarithm of the probability density is covered be a series of straight-line segments.



Log-concavity density function with upper rejection envelope and lower squeezing function

By assuming log-concavity of $p(\theta)$, we avoid the need to locate the supremum of $p(\theta)$. Adaptive rejection sampling reduces the probability of needing to evaluate $p(\theta)$ further by updating the envelope and squeezing functions to incorporate the most recently acquired information about $p(\theta)$ after each rejection.

The ARS algorithm is as follow (Gilks & Wild, 1992):

Preliminaries:

- Assume that $g(\theta) = c \cdot p(\theta)$ and $g(\theta)$ is continuous and differentiable everywhere in the domain of $p(\theta)$
- Let $h(\theta) = \ln g(\theta)$ such that $h(\theta)$ is concave everywhere in the domain

Initialization Step:

- Let $T_k = \{\theta_i; i = 1, \dots, k\}$ be the k starting points.
- If the domain is unbound to the left, choose θ_1 such that $h'(\theta_1) > 0$. If the domain is unbound to the right choose θ_1 such that $h'(\theta_k) < 0$.
- Calculate the following functions with the k starting points:
    1. $u_k(\theta)$, which is the piece-wise linear upper envelope formed from the tangents to $h(\theta)$ at each point in T_k.
    2. $s_k(\theta) = \frac{\exp u_k(\theta)}{\int \exp u_k(\theta')d\theta'}$
    3. $l_k(\theta)$, which is the piece-wise linear lower squeezing function formed

from the chords between adjacent points in in T_k.

The tangent lines at θ_j and θ_{j+1} intersect at the point

$$z_j = \frac{h(\theta_{j+1}) - h(\theta_j) - \theta_{j+1}h'(\theta_{j+1}) + \theta_j h'(\theta_j)}{h'(\theta_j) - h'(\theta_{j+1})}$$

For $j = 1, \ldots, k-1$ and z_0 is the lower bound of the domain and z_k is the upper bound of the domain.

The piecewise upper envelope is defined as

$$u_k(\theta) = h(\theta_j) + (\theta - \theta_j)h'(\theta_j)$$

for $\theta \in [z_{j-1}, z_j]$, similarly, the piece-wise lower squeezing function is defined as

$$l_k(\theta) = \frac{(\theta_{j+1} - \theta)h(\theta_j) + (\theta - \theta_j)h(\theta_{j+1})}{\theta_{j+1} - \theta_j}$$

for $\theta \in [\theta_j, \theta_{j+1}]$, if $\theta < \theta_1$ or $\theta < \theta_k$, $l_k(\theta)$ is set equal to $-\infty$.

Sampling Step:

- Sample a value θ^* from $s_k(\theta)$ and sample a value w independently from a uniform distribution $U(0,1)$.
- Squeezing Test: if $w \leq \exp\{l_k(\theta^*) - u_k(\theta^*)\}$ then accept θ^*, otherwise

evaluate $h(\theta^*)$ and $h'(\theta^*)$, and perform the following rejection test.

- Rejection Test: if $w \leq \exp\{h_k(\theta^*) - u_k(\theta^*)\}$ then accept θ^* otherwise reject θ^*.

Updating Step:

- If $h(\theta^*)$ and $h'(\theta^*)$ were evaluated in the sampling step, include θ^* in T_k to form T_{k+1}.
- Re-label the elements of T_{k+1} in ascending order and construct functions $u_{k+1}(\theta)$, $s_{k+1}(\theta)$, and $l_{k+1}(\theta)$ on the basis of T_{k+1}.
- Increment k. Return to the sampling step if n points have not yet been accepted.

With ARS, sampling of continuous, differentiable, and log-concave distribution using Gibbs sampling becomes more efficient. The number of evaluations decreases and has been shown empirically to be approximately in proportion of $\sqrt[3]{n}$.

Appendix D: Kalman Filter and Sequential Importance Sampling

The process used by Kalman filter equations can be separated into a two- step updating process:

- Step 1 is the *time update* in which the process model is used to project forward (in time) the state estimate and the error covariance estimate at the previous time step t-1 to obtain the *a priori* estimates at the current time step t.

- Step 2 is the *measurement update* in which the most recent observation D_t is used to update the *a priori* estimates to obtain the *posterior* estimates. The goal in this step is to use the additional information in the observation to improve the *a priori* estimates that were solely based on the process model.

Only specific implementations of the Bayesian recursive filter such as the Kalman filter or grid-based methods for domains consisting of discrete variables provide analytical methods. Thus it is necessary to pursue approximate inference algorithms. Particle filtering is one technique that makes the filtering problem tractable. Particle filtering, i.e. sequential Monte Carlo (MC), is a method for approximating the distribution of the belief state. Common particle filtering method are based on sequential importance sampling (SIS), which improves upon the basic sequential MC by weighting point masses (particles) according to their importance sampling density, thus focusing on the samples that affect the posterior belief state the most. A SIS filter builds upon the basic ideas of MC integration and importance sampling.

SIR is a sequential (i.e., recursive) version of importance sampling. As in importance sampling, the expectation of a function can be approximated as a weighted average:

$$\int g(x_t)dx_t = \int f(x_t)\pi(x_t)dx_t \approx \sum_{i=1}^{N} w_t^i f(x_t^i) \tag{9-1}$$

The transition prior probability distribution is often used as importance function, since it is easier to draw particles (or samples) and perform subsequent importance weight calculations

$$\pi(x_t|x_{0:t-1}, D_{0:t}) = p(x_t|x_{t-1})$$

A single step of sequential importance sampling is as follows:

1. For i=1,…,N draw samples from the proposal distribution
$$x_t^i \sim \pi\left(x_t|x_{0:t-1}^i, D_{0:t}\right)$$

2. For i=1,…,N update the importance weights up to a normalizing constant
$$w_t^i = w_{t-1}^i \frac{p\left(D_t|x_t^i\right)p(x_t^i|x_{t-1}^i)}{\pi\left(x_t|x_{0:t-1}^i, D_{0:t}\right)}$$

3. For i=1,…,N compute the normalized importance weights
$$w_t^i = \frac{\widehat{w}_t^i}{\sum_{i=1}^N \widehat{w}_t^i}$$

Many variations of the SIS particle filter exist using different importance sampling densities, resampling techniques, and combinations of discrete and continuous variables. Resampling is a technique used to prevent a phenomenon called degeneracy, where a single particle accounts for a large proportion of the total weight of all particles.

Bibliography

Aeryon Labs Inc. (2014). Retrieved from http://www.aeryon.com

Almond, R. G. (1992). *An extended example for testing graphical belief* . Technical Report 6, tatistical Sciences Inc.

Anderson, T. (1995). *Fracture Mechanics, 2nd Edition.* CRC Press.

Ashby, M., & Jones, D. (2005). *Engineering Materials 1.* Elsevier Publishing .

Barlow, R. (1988). Using influence diagrams. In *Accelerated Life Testing and Experts' Opinions in Reliability* (pp. 145-50).

Bechta Dugan, J., Bavuso, S. J., & Boyd, M. A. (1992). Dynamic fault-tree models for fault-tolerant computer systems. *Reliability, IEEE Transactions on , 41* (3), 363-377.

Bobbio, A., Portinale, L., Minichino, M., & Ciancamerla, E. (2001). Improving the analysis of dependable systems by mapping fault trees into Bayesian networks. *Reliability Enginnering System Safety , 71* (3), 249-60.

Boudali, H., & Dugan, J. B. (2006). A continuous-time Bayesian network reliability modeling, and analysis framework. *Reliability, IEEE Transactions on , 55* (1), 86-97.

Boudali, H., & Dugan, J. B. (2005). A discrete-time Bayesian network reliability modeling and analysis framework. *Reliability Engineering & System Safety , 87* (3), 337-349.

Bouissou, M., Martin, F., & Ourghanlian, A. (1999). Assessment of a safety- critical system including software: a Bayesian belief network for evidence sources. *Proceedings of the annual reliability and maintainability symposium;*, (pp. 142-50).

Boyen, X., & Koller, D. (1998). Tractable inference for complex stochastic processes. *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence* .

Buntine, W. L. (1996). A guide to the literature on learning probabilistic networks from data. *IEEE Transactions on Knowledge and Data Engineering , 8* (2), 195-210.

Buntine, W. L. (1994). Operations for learning with graphical models . *Journal of*

Artificial Intelligence Research , 2, 159-225.

Chen, Z. (2003). Bayesian filtering: From Kalman filters to particle filters, and beyond. *Statistics* , 1-69.

Choi, A., Zheng, L., Darwiche, A., & Mengshoel, O. J. (2011). A Tutorial on Bayesian Networks for System Health Management.

Cooper, G. a. (1992). A Bayesian method for the induction of probabilistic networks from data. *Machine Learning* , 309-347.

Cooper, G. F. (1990). The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence , 42* (2-3), 393-405.

Cousins, S. B., Chena, W., & Frisse, M. E. (1993). A tutorial introduction to stochastic simulation algorithms for belief networks. *Artificial Intelligence in Medicine , 5* (4), 315-340.

Dagum, P., & Horvitz, E. (1993). A Bayesian analysis of simulation algorithms for inference in belief networks. Networks. *23* (5), 499-516.

Dagum, P., & Luby, M. (1993). Approximating probabilistic inference in Bayesian belief networks is NP-hard . *Artificial Intelligence , 60* (1), 141-154.

Daly, R., Shen, Q., & Aitken, S. (2011). Learning Bayesian networks: approaches and issues. *The Knowledge Engineering Review , 26* (2), 99-157.

Dean, T., & Kanazawa, K. (1989). A model for reasoning about persistence and causation. *Computational Intelligence , 5* (2), 142-150.

Dempster, A. L. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B* , 1-38.

Devroye, L. (1986). Non-uniform Random Variate Generation. *New York: Springer-Verlag* .

Doguc, O., & Ramirez-Marquez, J. E. (2009). A generic method for estimating system reliability using Bayesian networks. *Reliability Engineering & System Safety , 94* (2), 542-550.

Fenton, N. E., Neil, M., & Marquez, D. (2008). Using Bayesian Networks to Predict Software Defects and Reliability. *Proceedings of the Institution of Mechanical Engineers, Part O, Journal of Risk and Reliability , 222* (04), 701-712.

Ferreiro, S., Arnaiz, A., Sierra, B., & Irigoien, I. (2011). A Bayesian network model integrated in a prognostics and health management system for aircraft line maintenance. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering , 225* (8), 886-901.

Flesch, I., & Lucas, P. (2007). Independence decomposition in dynamic Bayesian networks. In K. Mellouli (Ed.), *Proceedings of the Ninth European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECS- QARU 2007). Lecture Notes in Artificial Intelligence. 4724*, pp. 560-571. Springer.

Friedman, N. (1998). The Bayesian structural EM algorithm. In G. F. Cooper (Ed.), *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI-98)* (pp. 129-138). Morgan Kaufmann.

Friedman, N., Nachman, I., & Peer, D. (1999). Learning bayesian network structure from massive datasets. *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence.* Morgan Kaufmann Publishers Inc.

Geman, S., & Geman, D. (1984). Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intel ligence* , 721{742.

Ghahramani, Z. (1998). Learning dynamic Bayesian networks. In *Lecture Notes in Artificial Intelligence* (Vol. 1387, pp. 168-197). Springer.

Gilks, W. R. (1992). Adaptive rejection sampling for Gibbs sampling. *Applied Statistics* , vol. 41, pp. 337-348.

Gilks, W. R., & Wild, P. (1992). Adaptive Rejection Sampling for Gibbs Sampling. *Appl. Statist. , 41* (2), 337-348.

Gran, B., & Helminen, A. (2001). A Bayesian belief network for reliability assessment. *Computer Safety, Reliability and Security* (pp. 35-45). Springer Berlin Heidelberg.

Hastings, W. K. (1970). Monte Carlo Sampling Methods Using Markov Chains and Their Applications. *Biometrika , 57* (1), 97-109.

Heckerman, D. (1995). *A Tutorial on Learning with Bayesian Networks.* Microsoft Research.

Heckerman, D. (1995). *A Tutorial on Learning with Bayesian Networks.* Redmon, WA: Microsoft Research Advanced Technology Division, Microsoft Corporation.

Heckerman, D. (2008). *A tutorial on learning with Bayesian networks.* Springer

Berlin Heidelberg.

Heckerman, D. M. (1995b). Real-world applications of Bayesian networks. *Communications of the ACM* , 38.

Helminen, A., & Pulkkinen, U. (2003). Quantitative reliability estimation of a computer-based motor protection relay using Bayesian networks. In *Computer Safety, Reliability, and Security* (pp. 92-102). Springer Berlin Heidelberg.

Howard, R. A., & Matheson, J. E. (1984). Influence Diagrams. In *The Principles and Application of Decision Analysis* (Vol. 2, pp. 719-762). Strategic Decisions Group, Menlo Park.

Hugin Expert. (2007). *http://www.hugin.com* . Aalborg,, Denmark.

Jensen, F. (2001). *Bayesian Networks and Decision Graph* . Springer.

Jensen, F., Lauritzen, S. L., & Olesen, K. (1990). Bayesian updating in recursive graphical models by local computations. *Computational Statistics Quarterly , 4*, 260-282.

Julier, S. J., & Uhlmann, J. K. (1997). A new extension of the Kalman filter to nonlinear systems. *Int. symp. aerospace/defense sensing, simul. and controls , 3* (26).

Kalman, R. E. (1960). A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering , 82* (1), 35-45.

Kang, C. W., & Golay, M. W. (1999). A Bayesian belief network-based advisory system for operational availability focused diagnosis of complex nuclear power systems. *Expert Systems with Applications , 17* (1), 21-32.

Kass, R. E. (1995). Bayes Factors. *Journal of the American Statistical Association* , Vol. 90, No. 430, pp. 773-795.

Kipersztok, O., & Provan, G. (2003). A framework for diagnostic inference of commercial aircraft systems. In K. O. Agosta JM (Ed.), *The first Bayesian applications modeling workshop* .

Langseth, H., & Jensen, F. V. (2003). Decision theoretic troubleshooting in coherent systems. *Reliab Eng Syst Safety , 80* (1), 49-61.

Langseth, H., & Portinale, L. (2007). Bayesian networks in reliability. *Reliability Engineering & System Safety , 92* (1), 92-108.

Lauritzen, S. L., & Jensen, F. (2001). Stable Local Computation with Conditional

Gaussian Distributions. *Stat. & Comp. , 11*, 191-203.

Lauritzen, S. L., & Spiegelhalter, D. J. (1988). Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society. Series B (Methodological)* , 157-244.

Lauritzen, S. T. (1994). Diagnostic systems created by mo del selection metho ds: A case study. *AI and Statistics IV, volume Lecture Notes in Statistics, 89,* , 143{152.

Lerner, U. N. (2002). *Hybrid Bayesian Networks for Reasoning About Complex Systems.* Standford University, Dep. of Comp. Sci. Stanford.

Lerner, U., Parr, R., Koller, D., & Biswas, G. (2000). Bayesian fault detection and diagnosis in dynamic systems. In *AAAI/IAAI* (pp. 531-537).

Madigan, D. R. (1996). Bayesian model averaging. *In Proceedings of the AAAI Workshop on Integrating Multiple* .

Mahadevan, S., & Rebba, R. (2005). Validation of reliability computational models using Bayes networks. *Reliability Engineering & System Safety , 87* (2), 223-232.

McPherson, J. W. (2010). *Reliability Physics and Engineering.* Springer.

Ming, D., & Yang, Z.-b. (2008). Dynamic Bayesian network based prognosis in machining processes. *Journal of Shanghai Jiaotong University (Science) , 13*, 318-322.

Moral, S., Rumi, R., & Salmeron, A. (2001). Mixtures of Truncated Exponentials in Hybrid Bayesian Networks. In *ECSQARU 2001. LNCS (LNAI)* (Vol. 2143, pp. 156-167). Springer, Heidelberg.

Murphy, K. (1999). A Variational Approximation for Bayesian Networks with Discrete and Continuous Latent Variables. In *Uncertainty in Artificial Intelligence* (pp. 457-466). Morgan Kaufmann, San Francisco.

Murphy, K. (2002). *Dynamic Bayesian Networks: Representation, Inference and Learning.* PhD thesis, UC Berkeley , Dept. Computer Science .

Murphy, K., & Mian, S. (1999). *Modelling Gene Expression Data Using Dynamic Bayesian Networks.* Technical report, University of California, Berkeley, Computer Science Division.

NASA. (2010). *Handbook, NASA Risk-Informed Decision Making.* , Office of Safety and Mission Assurance NASA Headquarters.

Neal, R. M. (1993). *Proabilitic Inference Using Markov Chain Monte Carlo Methods.* Technical Report, University of Toronto, Department of Computer Science, 1993.

Neapolitan, R. E. (2004). *Learning Bayesian Networks.* Pearson Education.

Neil, M., Fenton, N., Forey, S., & Harris, R. (2003). Assessing Vehicle Reliability using Bayesian Networks. In *Global Vehicle Reliability, Edited by J. E. Strutt and P.L. Hall* (pp. 25-42). Professional Engineering Publishing.

Neil, M., Tailor, M., Marquez, D., Fenton, N., & Hear. (2007). Inference in Bayesian Networks using dynamic discretisation. *Statistics and Computing , 17* (3), 219-233.

Pearl, J. (1986). Fusion, Propagation and Structuring in Belief Networks. *Artificial Intelligent , 29*, 241-288.

Pearl, J. (1993). Graphical models, causality, and intervention. *Statistical Science , 8* (3), 266-273.

Pourali, M., & Mosleh, A. (2013). A Bayesian Approach to Sensor Placement Optimization and System Reliability Monitoring. *Journal of Risk and Reliability , 227* (3).

Rabiei, M., & Modarres, M. (2013). A recursive Bayesian framework for structural health management using online monitoring and periodic inspections. *Reliability Engineering & System Safety* (112), 154-164.

Raftery, A. E. (1995). Bayesian Model Selection in Social Research. *Sociological Methodology, Vol. 25* , 111-163.

Robinson, R. (1977). Counting unlabeled acyclic digraphs. *C.H.C. Little (Ed.), Lecture notes in mathematics,* , 622.

Schumann, J., Rozier, K. Y., Reinbacher, T., Mengshoel, O. J., Mbaya, T., & Ippolito, C. (2013). Towards Real-time, On-board, Hardware-supported Sensor and Software Health Management for Unmanned Aerial Systems. *Annual Conference of the Prognostics and Health Management Society.*

Shachter, R. D. (1986). Evaluating Influence Diagrams. *Operation Research , 34* (6), 871-882.

Shenoy, P. P. (2006). Inference in Hybrid Bayesian Networks Using Mixtures of Gaussians. In *Uncertainty in Artificial Intelligence* (pp. 428-436). AUAI

Press, Corvallis.

Shenoy, P. P. (1992). Valuation-Based Systems for Bayesian Decision Analysis. *Operation Research , 40* (3), 463-484.

Silbey, R., & Alberty, R. (2001). *Physical Chemistry, 3rd Edition.* John Wiley and Sons.

Singh, H., Cortellessa, V., Cukic, B., Gunel, E., & Bharadwaj, V. (2001). A bayesian approach to reliability prediction and assessment of component based systems. *Software Reliability Engineering, 2001. ISSRE 2001. Proceedings. 12th International Symposium on* (pp. 12-21). IEEE.

Spiegelhalter, D. a. (1990). Sequential updating of conditional probabilities on directed graphical structures. *Networks* , 20:579-605.

Spiegelhalter, D., Thomas, A., & Best, N. (1996). Bugs 0.5 Bayesian inference using Gibbs sampling manual (Version Ii). *MRC Biostatistic Unit* (1), 1-59.

Sze, S. (2002). *Semiconductor Devices: Physics and Technology, 2nd Edition.* John Wiley and Sons.

Tobon-Mejia, D. A., Medjaher, K., Zerhouni, N., & Tripot, G. (2012). A Data-Driven Failure Prognostics Method Based on Mixture of Gaussians Hidden Markov Models. *Reliability, IEEE Transactions on , 61* (2), 491-503.

Wakefield, J. C. (1991). Efficient generation of random variates via the ratio-of-uniforms method. *Statistics and Computing* , vol. 1, pp. 129-133.

Weber, P., & Jouffe, L. (2006). Complex system reliability modelling with dynamic object oriented bayesian networks (DOOBN). *Reliability Engineering & System Safety , 91* (2), 149-162.

Wilson, A. G., & Huzurbazar, A. V. (2007). Bayesian networks for multilevel system reliability. *Reliability Engineering & System Safety , 92* (10), 1413-1420.

Wooff, D., Goldstein, M., & Coolen, F. (2002). Bayesian graphical models for software testing. *IEEE Trans Software Eng , 28* (5), 510-25.