AFRL-IF-RS-TR-2000-8
Final Technical Report
February 2000

AN INVESTIGATION OF SYSTEM IDENTIFICATION TECHNIQUES FOR SIMULATION MODEL ABSTRACTION

Systems View

Douglas A. Popken and Louis Anthony Cox

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

20000328 008

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

DTIC QUALITY INSPECTED 3

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-2000-8 has been reviewed and is approved for publication.

APPROVED: James M. Vaccaro

JAMES M. VACCARO
Project Engineer

FOR THE DIRECTOR: James W. Cusack

JAMES W. CUSACK, Chief
Information Systems Division

If your address has changed or if you wish to be removed from the Air Force Research Laboratory Rome Research Site mailing list, or if the addressee is no longer employed by your organization, please notify AFRL/IFSB, 525 Brooks Road, Rome, NY 13441-4505. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY *(Leave blank)*

2. REPORT DATE
FEBRUARY 2000

3. REPORT TYPE AND DATES COVERED
Final Mar 99 - Nov 99

4. TITLE AND SUBTITLE
AN INVESTIGATION OF SYSTEM IDENTIFICATION TECHNIQUES FOR SIMULATION MODEL ABSTRACTION

5. FUNDING NUMBERS
C - F30602-99-C-0052
PE - 62702F
PR - 459S
TA - BA
WU - 92

6. AUTHOR(S)
Douglas A. Popken and Louis Anthony Cox

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Systems View
9139 S Roadrunner St
Highlands Ranch CO 80126

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Research Laboratory/IFSB
525 Brooks Road
Rome NY 13441-4505

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFRL-IF-RS-TR-2000-8

11. SUPPLEMENTARY NOTES
Air Force Research Laboratory Project Engineer: James M. Vaccaro/IFSB/(315)330-3708

12a. DISTRIBUTION AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*
This report summarizes research into the application of system identification techniques to simulation model abstraction. System identification produces simplified mathematical models that approximate the dynamic behaviors of the underlying stochastic simulations. Four state-space system identification techniques were examined: Canonical State-space, Compartmental Models, Maximum Entropy, and Hidden Markov Models (HMM). Two stochastic simulation models were identified: the "Attrition Simulation", a simulation of two opposing forces, each operating with multiple weapon system types; and the "Mission Simulation," a simulation of a squadron of aircraft performing battlefield air interdiction. The system identification techniques were evaluated and compared under a variety of scenarios on how well they replicate the distributions of the simulation states and decision outputs. Encouraging results were achieved by the HMM technique applied to Attrition Simulation - and by the Maximum Entropy technique applied to the Mission Simulation. This report also discusses the run-time performance of the algorithms, the development of suitable model structures, and implications for future efforts.

14. SUBJECT TERMS
Model Abstraction, System Identification, State-Space Models, Multi-Resolution Modeling, Simulation

15. NUMBER OF PAGES
100

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Abstract

This report summarizes research into the application of system identification techniques to simulation model abstraction. System identification produces simplified mathematical models that approximate the dynamic behaviors of the underlying stochastic simulations. Four state-space system identification techniques were examined: Canonical State-Space, Compartmental Models, Maximum Entropy, and Hidden Markov Models (HMM). Two stochastic simulation models were identified: the "Attrition Simulation", a simulation of two opposing forces, each operating with multiple weapon system types; and the "Mission Simulation", a simulation of a squadron of aircraft performing battlefield air interdiction. The system identification techniques were evaluated and compared under a variety of scenarios on how well they replicate the distributions of the simulation states and decision outputs. Encouraging results were achieved by the HMM technique applied to the Attrition Simulation - and by the Maximum Entropy technique applied to the Mission Simulation. This report also discusses the run-time performance of the algorithms, the development of suitable model structures, and implications for future efforts.

Keywords: model abstraction, system identification, state-space models, multi-resolution modeling, simulation

Table of Contents

ABSTRACT 1/2

LIST OF FIGURES AND TABLES 5

INTRODUCTION 6

- MOTIVATION 6
- CONTEXT 7
- CHOICE OF SYSTEM IDENTIFICATION TECHNIQUES 8

SIMULATION MODEL DEVELOPMENT 10

- MODEL SCOPE 10
- MODEL 1 - ATTRITION SIMULATION 11
  - *Background* *11*
  - *Framework* *12*
  - *Variables* *13*
  - *Initial and Termination Conditions* *13*
  - *Model States* *14*
- MODEL 2 - MISSION SIMULATION 18
  - *Background and Framework* *18*
  - *Variables and Logic* *18*
  - *Model States* *19*

ALGORITHM SPECIFICATIONS 22

- GENERAL MODEL 22
- CANONICAL STATE SPACE TECHNIQUE 23
  - *Framework* *23*
  - *General Algorithm* *23*
- COMPARTMENTAL MODEL TECHNIQUE 29
  - *Framework* *29*
  - *General Solution Technique* *31*
  - *Simulation Estimation Algorithm* *32*
- MAXIMUM ENTROPY TECHNIQUE 36
  - *Framework* *36*
  - *Simulation Estimation Algorithm* *38*
- HMM TECHNIQUE 42
  - *Framework* *42*
  - *Recursive Estimators* *42*
  - *Simulation Estimation Algorithm* *44*

TEST PLAN 46

- CROSS VALIDATION TESTING OVERVIEW 46
- TESTING OF MODEL STATE MATCHING 47
- TESTING OF MODEL BASED DECISIONS 47
- SIMULATION SCENARIOS 48
  - *Attrition Simulation* *48*
  - *Mission Simulation* *49*

DETAILED RESULTS 51

- MISSION SIMULATION 51

State Matching *51*
Average Behavior *52*
Model Based Decisions *55*
ATTRITION SIMULATION 57
State Matching *57*
Average Behavior *58*
Model Based Decisions *61*
EFFECTIVENESS OF THE BEST TECHNIQUES 62
Method *62*
Effectiveness Results *64*
ALGORITHM PERFORMANCE CONSIDERATIONS 72
Canonical State Space *72*
Compartmental *72*
Maximum Entropy *73*
HMM *73*

RESULTS SUMMARY AND CONCLUSIONS 74

IDENTIFICATION TECHNIQUES 74
OVERALL EFFECTIVENESS 75
PERFORMANCE 76

DISCUSSION 77

REFERENCES 78

APPENDIX A - ATTRITION SIMULATION: METHODS FOR CALCULATING FORCE STRENGTHS, INITIAL FORCE LEVELS, AND FIRE ALLOCATION 81

FORCE STRENGTH CALCULATION 81
ALGORITHM FOR CALCULATING INITIAL WIS 83
FIRE ALLOCATION METHODOLOGY 83

List of Figures and Tables

FIGURE 1. A HIERARCHY OF MILITARY SIMULATION MODELS 6
FIGURE 2. ABSTRACTION TAXONOMY 7
TABLE 1. ILLUSTRATIVE SCOPE AND RESOLUTION OF DOD MODELS 10
FIGURE 3. TWO-SIDED STOCHASTIC ATTRITION SIMULATION 12
FIGURE 4. VECTOR BASED STATE COMPONENTS OF ATTRITION SIMULATION 14
FIGURE 5. STANDARD FORM OF VECTOR STATE TRANSITION MATRIX FOR ATTRITION SIMULATION FOR DETERMINISTIC TECHNIQUES 15
FIGURE 6. FORM OF PROBABILITY TRANSITION MATRIX FOR ATTRITION SIMULATION USING THE COMPARTMENTAL MODEL TECHNIQUE 15
FIGURE 7. FORM OF SYSTEM STATE TRANSITION MATRIX FOR ATTRITION SIMULATION USING THE HMM TECHNIQUE 16
FIGURE 8. SAMPLE OUTPUT - ATTRITION SIMULATION 17
FIGURE 9. CAS/BAI MISSION SIMULATION 18
FIGURE 10. ENUMERATION OF STATES FOR THE MISSION SIMULATION 19
FIGURE 11. SAMPLE OUTPUT - MISSION SIMULATION 20
FIGURE 12. FORM OF THE PROBABILITY TRANSITION MATRIX FOR THE MISSION SIMULATION 21
FIGURE 13. AVERAGE ABSOLUTE DIFFERENCES IN STATE VALUES - MISSION SIMULATION 51
FIGURE 14. CANONICAL STATE SPACE- BASELINE SCENARIO 53
FIGURE 15. COMPARTMENTAL MODEL – BASELINE SCENARIO 53
FIGURE 16. MAXIMUM ENTROPY – BASELINE SCENARIO 54
FIGURE 17. HMM – BASELINE SCENARIO 54
FIGURE 18. AVERAGE PROPORTIONAL DIFFERENCES IN MISSIONS COMPLETED - MISSION SIMULATION 55
FIGURE 19. AVERAGE PROPORTIONAL DIFFERENCES IN AIRCRAFT DAMAGED/DESTROYED - MISSION SIMULATION 56
FIGURE 20. AVERAGE ABSOLUTE DIFFERENCES IN STATE VALUES - ATTRITION SIMULATION 57
FIGURE 21. CANONICAL STATE SPACE – BASELINE SCENARIO 59
FIGURE 22. COMPARTMENTAL MODEL –BASELINE SCENARIO 59
FIGURE 23. MAXIMUM ENTROPY – BASELINE SCENARIO 60
FIGURE 24. HMM – BASELINE SCENARIO 60
FIGURE 25. AVERAGE PROPORTIONAL DIFFERENCE IN TIME TO TERMINATION - ATTRITION SIMULATION 61
FIGURE 26. AVERAGE DIFFERENCE IN BLUE WIN FRACTION 62
FIGURE 27. CONTROL CHARTS FOR AVERAGE COMPLETION TIME, ATTRITION SIMULATION, BASELINE SCENARIO 65
FIGURE 28. CONTROL CHARTS FOR RANGE OF COMPLETION TIMES, ATTRITION SIMULATION, BASELINE SCENARIO 66
FIGURE 29. CONTROL CHARTS FOR BLUE WIN FRACTION, ATTRITION SIMULATION, BASELINE SCENARIO 67
FIGURE 30. CONTROL CHARTS FOR AVERAGE MISSIONS COMPLETED, MISSION SIMULATION, BASELINE SCENARIO 68
FIGURE 31. CONTROL CHARTS FOR RANGE OF MISSIONS COMPLETED, MISSION SIMULATION, BASELINE SCENARIO 69
FIGURE 32. CONTROL CHARTS FOR AVERAGE AIRCRAFT DESTROYED, MISSION SIMULATION, BASELINE SCENARIO 70
FIGURE 33. CONTROL CHARTS FOR RANGE OF AIRCRAFT DESTROYED, MISSION SIMULATION, BASELINE SCENARIO 71
TABLE 2. ATTRITION SIMULATION – AVERAGE ERRORS 74
TABLE 3. MISSION SIMULATION – AVERAGE ERRORS 74

Introduction

Motivation

Model abstraction techniques are used to construct low-resolution approximations of higher-resolution models. In this research, the high-resolution models are stochastic, discrete-event simulations of military combat systems. The research is motivated by the hypothesis that model abstraction is a potential enabling technology for the larger goal of multi-resolution modeling.

In military simulations, multi-resolution modeling is often described with respect to a hierarchy of models similar to Figure 1. The idea is to execute a system model

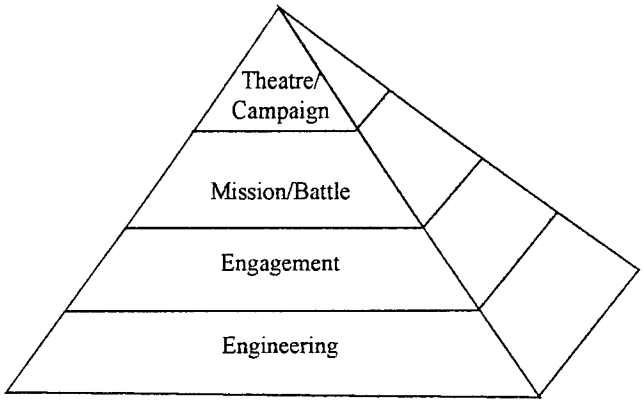

Figure 1. A Hierarchy of Military Simulation Models

operating at a given level of the model hierarchy, which is itself comprised of components defined at the next lower level. The strategy reduces simulation development and support costs through gains in software reuse. A key to practical application is the ability to substitute low-resolution, approximate versions of components for high-resolution versions. This provides gains in run-time efficiency and further reduction of support costs, but at the expense of some acceptable loss of model accuracy.

Even as computers become faster there will be a continuing need for low-resolutions models, high-resolution models, and multi-resolution modeling. Low-resolution models can be useful for:

- making initial cuts at problems,
- "comprehending the whole" without getting lost in detail,
- reasoning about issues quickly or under time pressure,
- analyzing choices in the presence of uncertainty,
- using low-resolution information, and
- calibrating higher-resolution models.

On the other hand, high-resolution models are useful for:

- understanding underlying phenomena,
- representing and reasoning about detailed knowledge,
- simulating "reality" and create virtual laboratories for studying phenomena that cannot be studied in any other way (e.g., a range of possible battles and wars),
- using high-resolution information, which is sometimes quite tangible (e.g., weapon performance), and
- calibrating lower-resolution models

(From Davis and Zeigler, 1997).

There are also strong motivations for having the ability to model at *multiple* levels of resolution, chief among them being speed and efficiency. By avoiding high levels of resolution across *all* model functions we can address large and complex scenarios while still being able to economize on resources needed for computation (hardware), data collection, model setup, validation, and analysis. Another historical motivation has been the desire to connect existing legacy models that operate at different levels of resolution.

Context

There are many forms of model abstraction. Figure 2 shows one useful taxonomy.

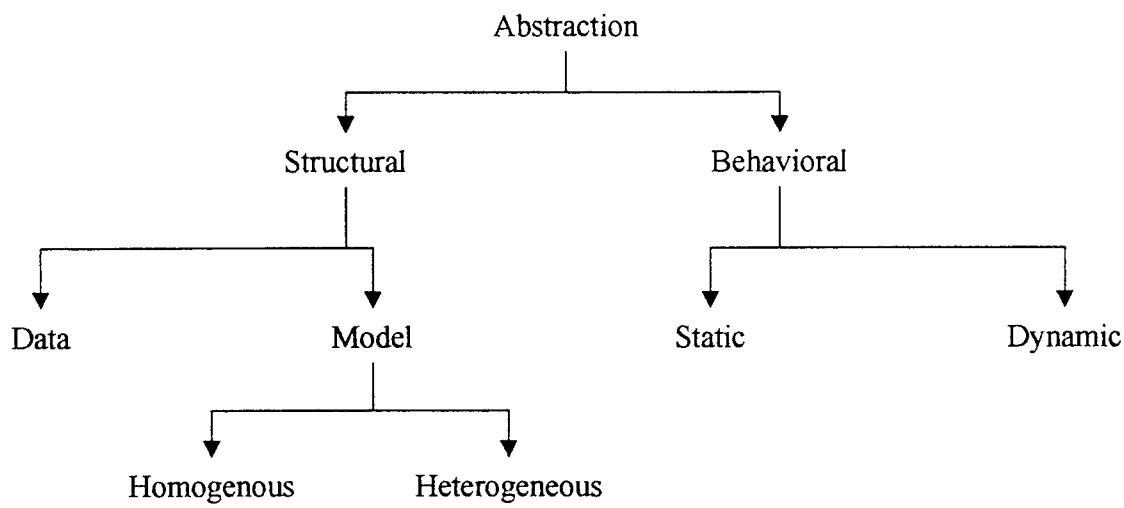

Figure 2. Abstraction Taxonomy
(from Lee and Fishwick, 1996)

Structural abstraction focuses on model abstraction levels and model types. *Data abstraction* uses statistical, mathematical, relational, or symbolic substitution methods to approximate time-dependent information (input, outputs, or parameters). *Model*

abstractions focus on the composition of model components and mappings between components residing at different levels. *Behavioral abstraction* replaces a system component with a simplified version that approximates the behavior of the original component. *Static* behavioral approaches are time independent, and capture only a steady state output value. *Dynamic* behavioral approaches associate input/output trajectories over time. System identification, the subject of this report, is a set of techniques that produce the latter - dynamic systems behavioral abstractions.

Behavioral abstractions are sometimes thought of as "black box" modeling because they are more concerned with the input-output characteristics of the original model than with its internal structure. However, in this research we use techniques that might more accurately be called "gray box" approaches. That is, we will also be concerned with the structural components of the abstraction. These components can be shown to be morphisms (Zeigler, 1998) of the components in the simulation model. Consequently, our approach combines aspects of both structural and behavioral abstraction.

System identification techniques are widely used in biomathematics, medicine, control system design, signal processing, speech recognition, and other fields to develop models of dynamical systems. There are many types of dynamical systems, e.g.: discrete/continuous time or state space, linear/nonlinear input-state-output mappings and dynamics, stochastic/deterministic/chaotic evolution, time varying/invariant state transition intensities, lumped/distributed parameters, centralized/decentralized control, etc. A correspondingly vast number of system identification techniques have been developed to handle these different characteristics (Ljung, 1987).

Choice of System Identification Techniques

Selecting the appropriate system identification technique(s) involves making trade-offs among accuracy, performance, the quantity of data available, and the quality of the data. First of all, the simulation models we wish to identify are stochastic; therefore, we want techniques that can handle random and noisy data. Second, we seek techniques that do not require excessive amounts of input/output data for the identification process. Such data may not be available within practical limits of time and cost. To maximize usability, the technique should be general enough to handle multiple inputs and multiple outputs ("MIMO" systems). Lastly, the resulting model should be compact and efficient from both a computational and a conceptual point of view.

Simultaneous pursuit of these objectives severely limits the set of possible techniques. We have selected a set of four, somewhat nontraditional, system identification techniques capable of producing linear time invariant (LTI) state-space models of the following general form:

$$x(t) = Ax(t-1) + Bu(t) + \varepsilon(t)$$
$$y(t) = Cx(t) + e(t)$$

where t is the discrete time index, y is an M element (noisy) observation vector, x is an N element state vector (the number of entities in each element), u is an R element input vector, ε is an N element system noise vector, e is an M element observation noise vector, A is an N x N state transition matrix, B is an N x R input matrix, and C is an M x N observation matrix. The techniques examined in this research are:

- Canonical State Space
- Compartmental Model
- Maximum Entropy
- Hidden Markov Model (HMM)

Each is described in detail within this report.

Simulation Model Development

In this research we use specially developed simulation models as sources of system identification data. We use these, rather than using existing military simulation models, to provide a better assessment of the overall utility of the proposed system identification algorithms. The specific advantages include:

- greater transparency
- greater control over structural elements
- greater control over dynamics (transition probabilities, event times, variance, etc.)
- ease of use
- generality

At the same time we want models that portray typical elements of military simulations with sufficient detail to draw initial conclusions on the general applicability of the proposed algorithms.

Model Scope

In general, military simulation models span a wide range of domain scope and resolution. Table 1 below summarizes major model categories and their characteristics.

Level of Model	Scope	Level of Detail	Time Span	Outputs	Illustrative Uses	Examples
Theater/ Campaign	Joint and combined	Highly aggregated	Days to weeks	Campaign dynamics, (e.g., force draw-downs, movement)	Evaluation of force structures, strategies, balances; wargaming	CEM, TACWAR, THUNDER, JICM
Mission/ Battle	Multi-platform	Moderate aggregation with some entities	Minutes to hours	Mission effectiveness (e.g., exchange ratios)	Evaluation of alternative force employment concepts, forces, systems; wargaming	Eagle, Vector II Suppressor, EADSIM, NSS
Engagement	One to a few friendly entities	Individual entities, some detailed subsystems	Seconds to minutes	System effectiveness (e.g. probability of kill)	Evaluation of alternative tactics and systems; training	JANUS, Brawler, ESAMS
Engineering	Single weapon systems and components	Detailed down to piece parts, plus physics	Sub-seconds to seconds	Measures of system performance	Design and evaluation of systems and subsystems; test support	Many throughout R&D Centers

From Davis and Bigelow, 1998.

Table 1. Illustrative Scope and Resolution of DoD Models

Two benchmark simulation models were developed for this research. The first is the "Two-Sided Stochastic Attrition Simulation" (a.k.a. Attrition Simulation). The second is the "CAS/BAI (Close Air Support/Battlefield Air Interdiction) Mission Simulation" (a.k.a. Mission Simulation). Within the framework provided above, our benchmark models could be considered as simplified versions of "Mission/Battle" models or "Theater/Campaign" models.[1] The simulation models are discussed in detail below.

Model 1 - Attrition Simulation

Background

Attrition simulations or attrition equations define the combat dynamics of combat simulation models. They are used to model the duration, lethality, and victor of a given combat scenario. The attrition simulation model developed for this research was inspired in part by Ancker (1995), who proposed two axioms for a "theory of combat": 1) "all combat is a hierarchical network of firefights", and especially 2) "a firefight is a *terminating stochastic target attrition process on a discrete state-space with a continuous time parameter.*" The "Attrition Simulation" was constructed to fit Ancker's second axiom.

A reading of the available literature on existing combat simulations (see, for example, Bracken, Kress, and Rosenthal [eds.], 1995) indicates that this second axiom is observed by the attrition logic of a number of existing combat simulation models. However, it should also be noted that many other combat simulations use deterministic methods or a process mean value algorithm only. A common approach within this category is to use the classic Lanchester differential equations -

$$dx_t/dt = -Ay_t$$
$$dy_t/dt = -Bx_t$$

to describe attrition dynamics. Here $x_t = (x_{1t}, x_{2t}, \ldots x_{mt})$ and $y_t = (y_{1t}, y_{2t}, \ldots y_{nt})$ represent the vector quantities of weapon systems by type on opposing sides, while $A = [A_{ij}]$ and $B = [B_{ij}]$ are the Lanchester coefficient matrices defining the rate at which *y* systems destroy *x* systems and vice versa. There has been some recent progress in aggregating models based on these dynamics (Fowler,1999; Hillestad and Juncosa, 1995). Fowler further demonstrates how his technique can be applied to the final output of a single simulation instance. However, it is well known that these Lanchester relations are deficient in several respects: 1) they ignore combat stochasticity, 2) they do not account for the stochastic terminal distributions (absorption probabilities), and 3) they do not

[1] Both simulation models are highly simplified when compared to military simulation models in actual use. The rationale is the need to focus on the essential dynamic elements.

account for the correlation between attrition on the two sides (Ancker and Gafarian, 1992). Techniques developed specifically for aggregating Lanchester equations can not be applied to identifying the general stochastic model that we consider in this research. However, the reverse may be true.

Framework

The framework of the Attrition Simulation is illustrated in Figure 3 below. There are two opposing sides, "Blue" and "Red", each having several weapon system "types". There are one or more weapon systems within each type. The simulation tracks each individual weapon system as it fires, and is fired upon, over time. Each weapon system has a probability distribution describing the time between its firing events; however, all weapon systems of a given type have the same distribution. The selection of a target (opposing weapon system type) is performed by a probabilistic fire allocation function (see Appendix A for details) specific to each type. The probability of a weapon system destroying a member of a targeted type is specific to that attacker/defender pair. Each side may be replenished by new weapon systems of specific types ("Arrivals") at specific points in time. The simulation continues until a predefined combat termination condition is reached.

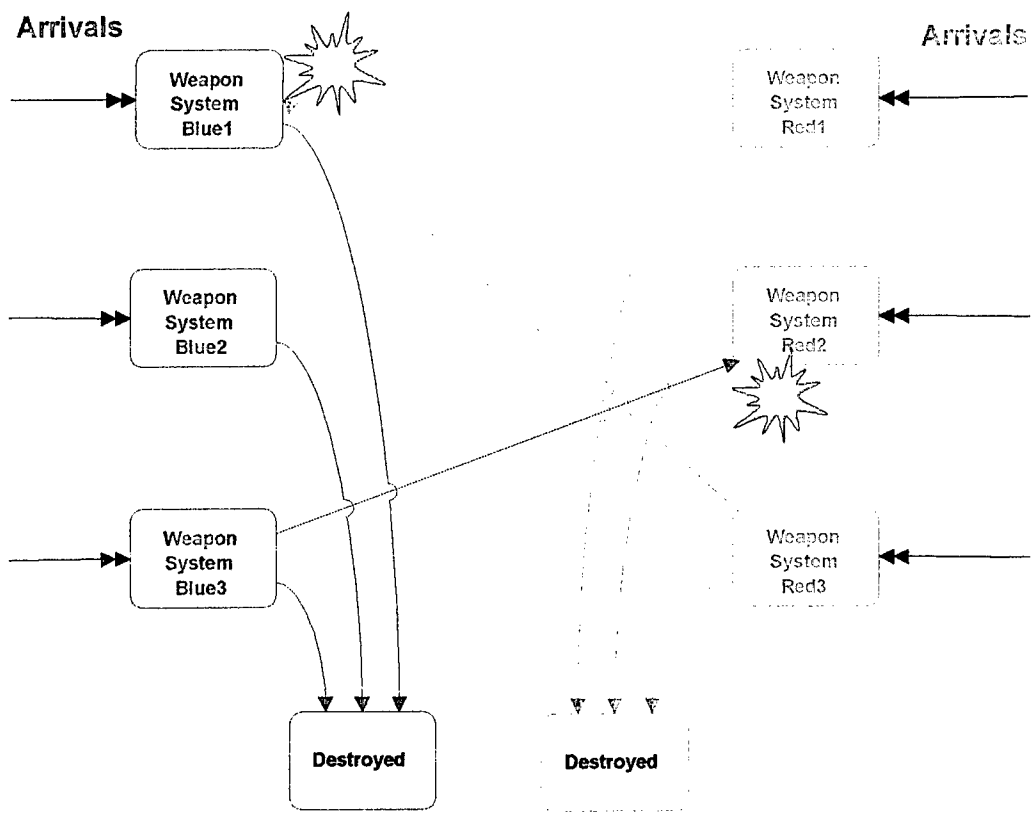

Figure 3. Two-Sided Stochastic Attrition Simulation

The attrition simulation can be applied to a wide variety of opposing weapon systems, including aircraft. Combat aircraft can be modeled by grouping them into types, where the types can be distinguished by differences in missions and, optionally, by differences in airframes. For example, missions might include defensive, escort, and ground attack. Airframes might include F-15's, F-16's, and A-10's. Depending on the objectives of the simulation, we could map individual aircraft within different airframe categories to different missions, resulting in a mixture of airframe subcategories within each mission grouping. Each of these subcategories becomes a "type" – equivalent to a "weapon system" in Figure 3 above.

Variables

Each weapon system type within the Attrition Simulation has attributes of lethality and vulnerability that are characterized by: 1) a vector of kill probabilities (one probability for each defender weapon system), 2) a probability distribution of inter-firing times, and 3) a fire allocation function to distribute engagements among defender weapon system types (one probability for each defender weapon system).

The other major variables describe the form of the arrival process (if any). Note that for the purposes of system identification, it is not necessary to construct a set of arrivals typical of the problem domain. We are free to use whatever arrival process will allow us to best identify the simulation model. However, when *testing* the resulting identified model, via comparisons to the original simulation, realistic arrival patterns would become more important.

Initial and Termination Conditions

The initial state is the number of weapon systems of each type on each side. The termination condition is a description of the state vector that signals the end of the simulation. One of four rules, each based on force strengths (discussed below and in Appendix A) may be used:

1. Absolute Decision Rule – Combat termination occurs when the force strength (of either side) reaches a given threshold value.
2. Proportional Decision Rule – Combat terminates when the force ratio reaches a specified threshold value.
3. AOP Rule – Combat terminates when the force strength crosses either the absolute or proportional threshold.
4. AAP Rule – Combat terminates when the force strength curve crosses both the absolute and proportional threshold .

For further details on the characteristics and use of these rules see Jaiswal and Nagabhushana (1995).

Model States

The critical modeling feature of the Attrition Simulation is that the behavior of each simulation entity depends on the number and types of the other entities within the system. In other words, the rate at which a weapon system of a given type is destroyed is highly dependent on the current levels of both opposing and friendly weapon systems. This implies that we must capture all of the weapon system levels in our state space model structure. The simplest way to handle this is to work strictly at an aggregated level, where our state vector is a simple count of the number of weapon systems of each type, illustrated in Figure 4 below[2]. This approach is consistent with many existing large-scale military simulation models (Davis, et. al.; 1997). Within this approach we can easily model weapon system replenishments as external inputs to the weapon system state vector.

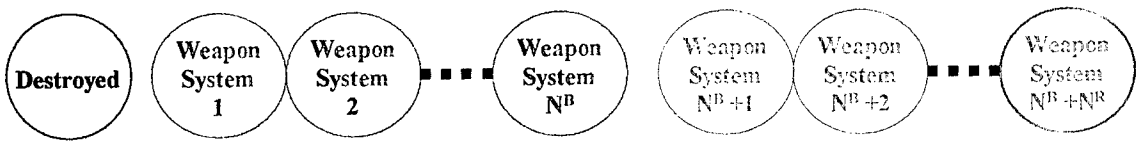

Figure 4. Vector Based State Components of Attrition Simulation

For purposes of system identification we convert the weapon system quantity vectors into *force strength* vectors. By using force strengths we have a common unit of measure for all weapon systems, thereby facilitating aggregation methods, termination rules (see below) and assumptions regarding "flow" between state vector components. The methodology for calculating force strengths is from Anderson and Miercort (1989, 1995) and is described in Appendix A. Under this approach, the "entities" being modeled are the units of force strength, which are initially distributed among the various weapon system types. The current number of "entities" of a given type is $\tilde{V}_i W_i$, where $\tilde{V}_i$ is the normalized strength factor and W_i is the quantity for weapon system type i.

Using a vector of force strengths is a reasonable approach for the Canonical State Space, Maximum Entropy , and Compartmental Model techniques. For the first two techniques, this approach to state vector realization results in a deterministic state transition matrix relating current force levels to force levels in the next period. The standard form of the state transition matrix is shown in Figure 5 below[3].

---

[2] N^B = number of Blue weapon system types, N^R = number of Red weapon system types

[3] The *algebraically equivalent* canonical form (as produced by the Canonical State Space technique) would look quite different.

	Destroyed	wsB1	wsB2	wsB3	wsR1	wsR2	wsR3
Destroyed	1	+	+	+	+	+	+
wsB1	0	+	+	+	-	-	-
wsB2	0	+	+	+	-	-	-
wsB3	0	+	+	+	-	-	-
wsR1	0	-	-	-	+	+	+
wsR2	0	-	-	-	+	+	+
wsR3	0	-	-	-	+	+	+

Figure 5. Standard Form of Vector State Transition Matrix for Attrition Simulation for Deterministic Techniques

A Compartmental Model assumes a system in which the various subsystems interact by exchanging "flows of materials" (force strengths in this case). The contents of the compartments are then inspected at discrete points in time. The result of the technique is a probability transition matrix, P(t), of the form shown in Figure 6.

	Destroyed	wsB1	wsB2	wsB3	wsR1	wsR2	wsR3
Destroyed	1	p12(t)	p13(t)	p14(t)	p15(t)	p16(t)	p17(t)
wsB1	0	p22(t)	p23(t)	p24(t)	p25(t)	p26(t)	p27(t)
wsB2	0	p32(t)	p33(t)	p34(t)	p35(t)	p36(t)	p37(t)
wsB3	0	p42(t)	p43(t)	p44(t)	p45(t)	p46(t)	p47(t)
wsR1	0	p52(t)	p53(t)	p54(t)	p55(t)	p56(t)	p57(t)
wsR2	0	p62(t)	p63(t)	p64(t)	p65(t)	p66(t)	p67(t)
wsR3	0	p72(t)	p73(t)	p74(t)	p75(t)	p76(t)	p77(t)

Figure 6. Form of Probability Transition Matrix for Attrition Simulation Using the Compartmental Model Technique

The HMM technique works differently than the others; it assumes single entities traversing the possible system states, with transitions between states governed by probabilities. However, our simulation model does not have actual individual force strength entities that can be tracked through the system. An alternative is to consider the entire system as a single entity that traverses the possible system states. The potential problem with this approach is the huge number of states. For example, if the quantity of each weapon system type varies between 0 and 10 (11 value levels), and we have 6 weapon system types, we would have $11^6 = 1{,}771{,}561$ possible system states! A further difficulty is that any one state has a low probability of occurrence, creating additional computational difficulties. This full enumeration approach would clearly be impractical for even the smallest problems.

The solution is to aggregate "equivalent" (or nearly equivalent) states so that we can describe the essential system state without tracking every possible combination of weapon system quantities. The assumption is that certain combinations of weapon systems are equivalent to others in terms of simulation dynamics. For example, a current weapon system quantity vector of $(W_1^B, W_2^B, W_3^B, W_1^R, W_2^R, W_3^R)$ might be equivalent to a vector of $(W_1^B+1, W_2^B-1, W_3^B+1, W_1^R-1, W_2^R+1, W_3^R+1)$ in determining how the remainder of the simulation progresses. Fortunately, we already have a mechanism for determining state equivalency via the *force strengths*. The force strength for side *s* is given by:

$$S^s = \left[\sum_{i=1}^{N^s} V_i^s W_i^s\right]^{\rho} \quad ; \quad i = 1,2,3,..N^s \; ; \; s \in \{R, B\}$$

We also rescale the V_i^s so that the score of the average weapon is always equal to 1.0. This allows for more rational period-by-period comparisons, and facilitates state-based analysis by narrowing the range of possible values for the force strengths (Anderson and Miercort, 1989). Let $W = [W^B, W^R]$ and $N = N^B + N^R$. We can compute a scale factor:

$$\alpha = \frac{\sum_{i=1}^{N} V_i W_i}{\sum_{i=1}^{N} W_i} \quad \text{so that:} \quad \tilde{V}_i = V_i / \alpha \quad \text{are the scaled strength factors.}$$

For a given simulation scenario, let the maximum force strength of either side be given by S^s_{max} and the minimum by S^s_{min}. Divide the interval $(S^s_{max} - S^s_{min})$ into k subintervals. Each subinterval becomes our aggregate state for side *s*, resulting in a total of k^2 system states. The parameter *k* can be varied to evaluate tradeoffs between the number of states and the accuracy of the resulting system identification. Figure 7 below is an example of the form of the resulting state transition matrix[4] for k = 3.

	sB1sR1	sB1sR2	sB1sR3	sB2sR1	sB2sR2	sB2sR3	sB3sR1	sB3sR2	sB3sR3
sB1sR1	p11	p12		p14	p15				
sB1sR2		p22	p23	p24	p25	p26			
sB1sR3			p33		p35	p36			
sB2sR1		p42		p44	p45		p47	p48	
sB2sR2			p53		p55	p56	p57	p58	p59
sB2sR3						p66		p68	p69
sB3sR1					p75		p77	p78	
sB3sR2						p86		p88	p89
sB3sR3									p99

Figure 7. Form of System State Transition Matrix for Attrition Simulation Using the HMM Technique

[4] The blank cells of the matrix are either 0 or ε, where ε is small relative to the p's

Figure 8 provides a sample output from the Attrition Simulation showing the aggregated force strengths for Red and Blue. Note the high variability of the outputs.

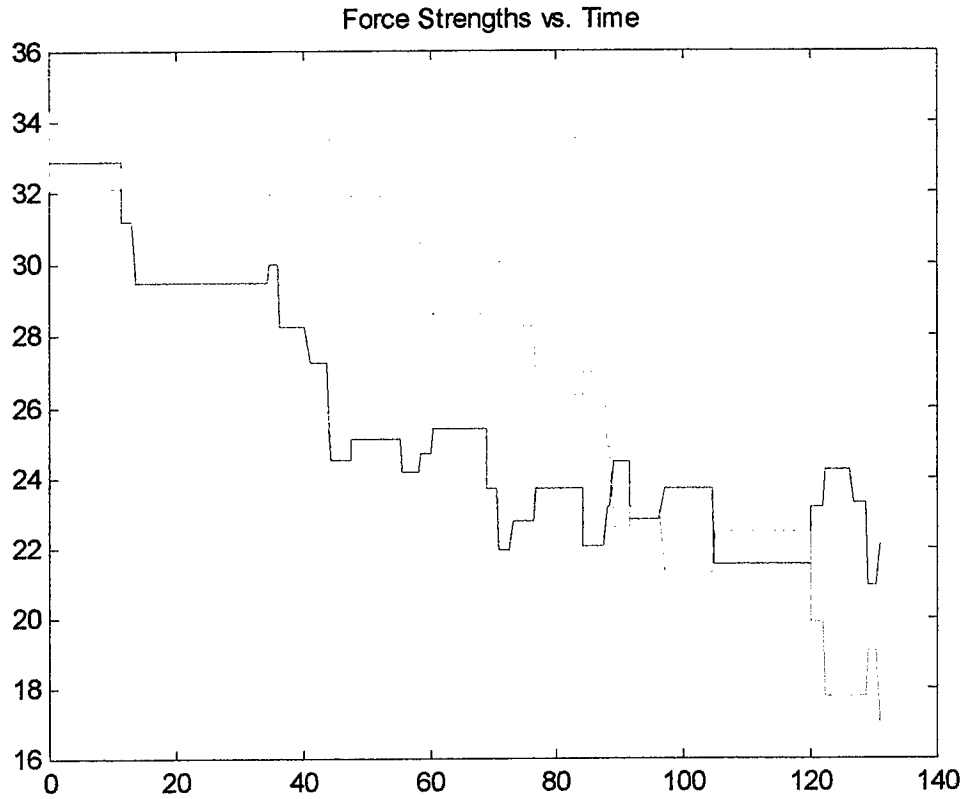

Figure 8. Sample Output - Attrition Simulation

Model 2 - Mission Simulation

Background and Framework

The CAS/BAI Mission Simulation is based on a scenario described in Samuelson and Sims (1995). We have added to the scenario the possibility of an aircraft being destroyed during a mission (see Figure 9). The objective of the original model is to analyze mission performance under various levels of jamming. The more jamming, the longer it takes the Forward Air Controller (FAC) to match a plane to a specific target. The more time spent in target matching, the less time there is to complete missions before fuel runs out. Queueing at the FAC will increase as target matching time and aircraft arrivals increase. We assume that the planes are operating in a target and threat rich environment that is constant over the simulation time horizon. Aircraft combat missions are limited by the amount of fuel remaining. If aircraft have sufficient fuel after the completion of a mission they will return to the FAC for another assignment, otherwise they will return to base.

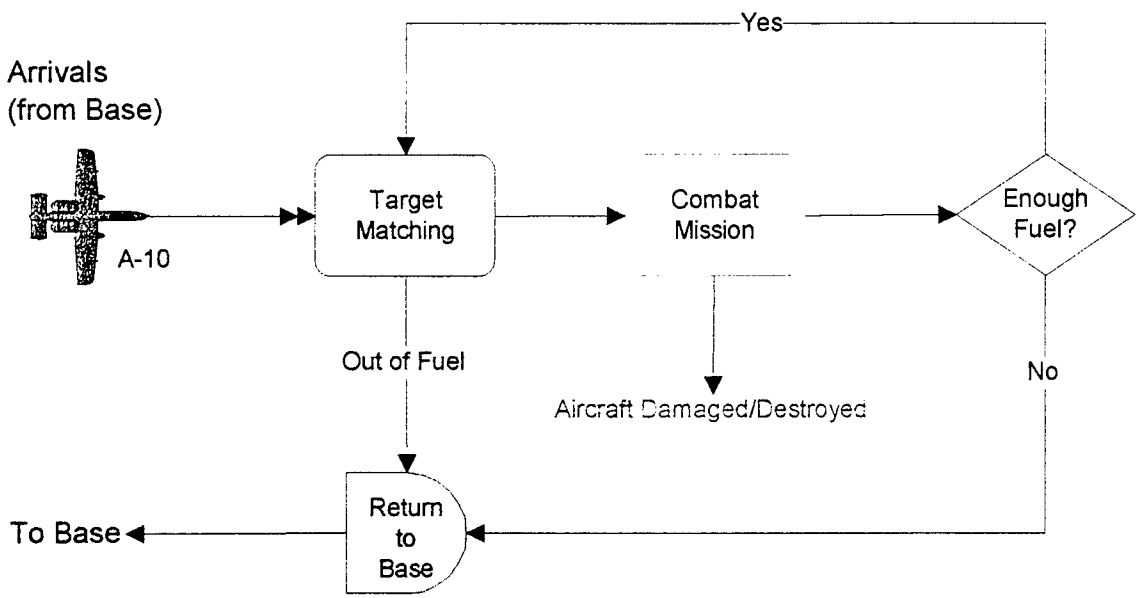

Figure 9. CAS/BAI Mission Simulation

Variables and Logic

We assume a constant value for the probability of an aircraft being damaged or destroyed during a mission (typically 0.0 to 0.25). The duration of the combat mission is described by a uniform probability distribution ranging from 3 to 10 minutes. The mission includes attack maneuvering, weapons release, and target area escape. Fuel consumption occurs at a fixed rate over time and aircraft have about 1 hour of fuel to carry out their mission. Aircraft that have completed a combat mission check their fuel. If they have used 55

minutes or more of fuel they return to base, otherwise they enter the waiting area for the FAC. The fuel level of each aircraft in the FAC queue is also checked at each simulation event time - if they have used 55 minutes or more of fuel they return to base. The time required to match targets to a given aircraft is described by the probability distributions described below.

Target matching time and arrival scenarios are paired so that arrivals do not overwhelm the FAC. A total of 24 aircraft (a squadron) will arrive during the simulation. Target matching time is described by uniform or normal probability distributions. A "no jamming" scenario is achieved by the use of an Automatic Target Handoff System (ATHS). Our simulations begin with no aircraft except that the first pair of arrivals will be at time 0. The termination condition is when all 24 aircraft have either returned to base or been damaged/destroyed.

Model States

Unlike the Attrition Simulation, the behavior of a given simulation entity (aircraft) in this model structure is largely independent of the other entities. The exception to this is the potential queueing at the FAC. To begin our analysis of the model structure, we see that aircraft are either "at the FAC", "in combat", "damaged/destroyed", or "returning to base". Since the current fuel load is a major part of the event logic, we must also capture it as part of the state of an individual aircraft. The best way to handle all of this is to discretize the fuel load into, say, quarters. Similarly we can discretize the size of the FAC queue (upon time of entry by the aircraft) into something like "light", "medium", and "heavy". The result is a state-space model with 18 states (see Figure 10 below).

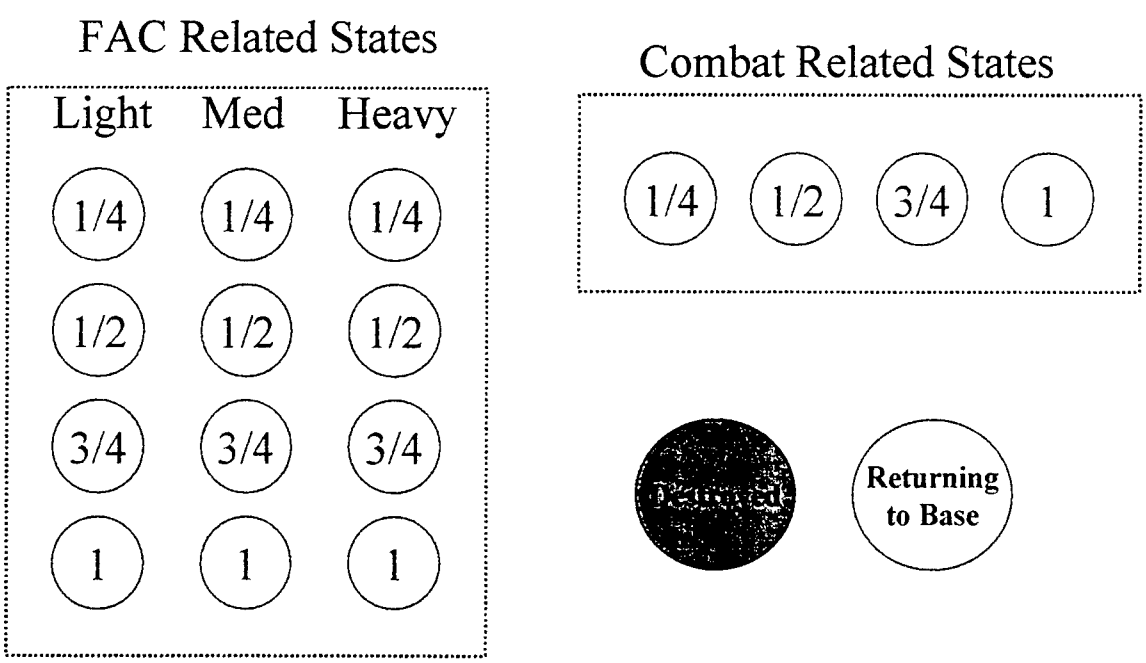

Figure 10. Enumeration of States for the Mission Simulation

In Figure 11 we see a sample output of the Mission Simulation, aggregating state values within the four groups shown in Figure 10.

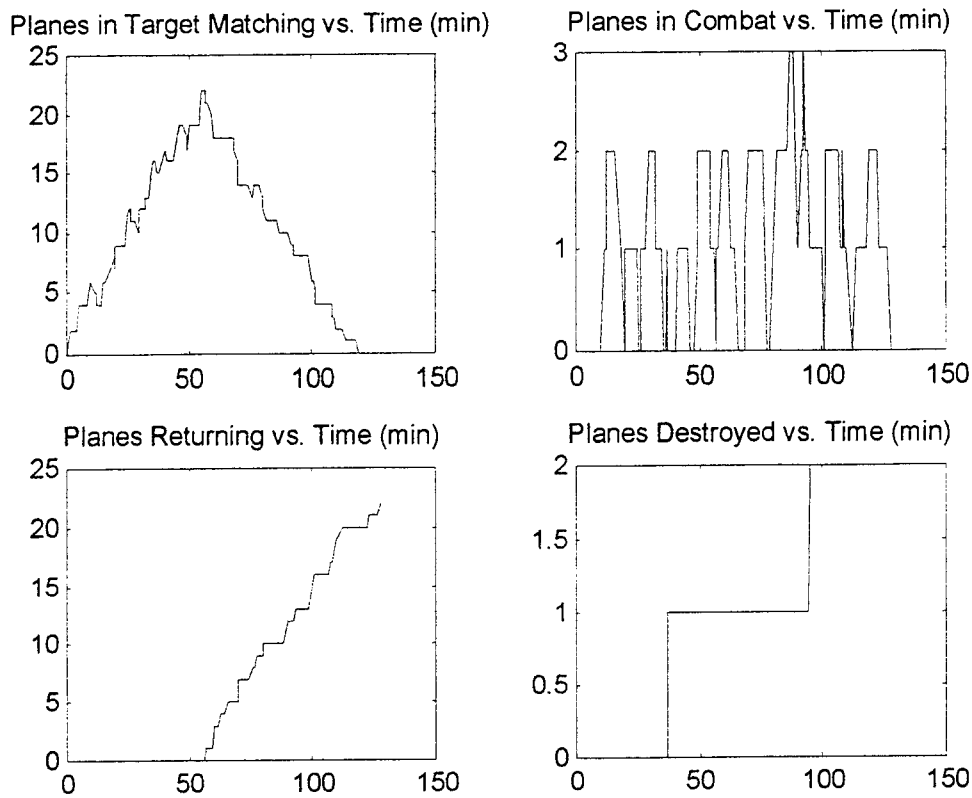

Figure 11. Sample Output - Mission Simulation

This model structure can accommodate all of our identification approaches. The Maximum Entropy and Canonical State Space approaches can operate directly on a state (observation) vector defined as in Figure 10. These approaches can also both handle the arriving aircraft as input vectors mapped through input transition matrices.

The Compartmental Model approach operates on a similar state (observation) vector. However, due to computational considerations for arrivals, the state vectors need to be segregated by aircraft arriving at different times so that each arriving cohort is identified separately. The resulting transition probability matrix is similar to Figure 5 but with two, rather than one, absorbing states.

For the HMM technique we can easily construct time-dependent state (observation) vectors that are specific to individual aircraft. These vectors take the form of identity vectors - a vector of zeros with a 1 in the position indicating the current state - which are then used as input to the Hidden Markov Model algorithm.

With the exception of the Canonical State Space technique, each identification algorithm produces a transition probability matrix of the form illustrated by Figure 12 below. Empty spaces indicate a 0 probability.

	D/D	Base	F1/4L	F1/2L	F3/4L	F1L	F1/4M	F1/2M	F3/4M	F1M	F1/4H	F1/2H	F3/4H	F1H	C1/4	C1/2	C3/4	C1
D/D	1														p	p	p	p
Base		1	p				p				p				p			
F1/4L			p	p											p	p		
F1/2L				p	p											p	p	
F3/4L					p	p											p	p
F1L						p												p
F1/4M							p	p							p	p		
F1/2M								p	p							p	p	
F3/4M									p	p							p	p
F1M										p								p
F1/4H											p	p			p	p		
F1/2H												p	p			p	p	
F3/4H													p	p			p	p
F1H														p				p
C1/4			p	p			p	p			p	p			p	p		
C1/2				p	p			p	p			p	p			p	p	
C3/4					p	p			p	p			p	p			p	p
C1						p				p				p				p

Figure 12. Form of the Probability Transition Matrix for the Mission Simulation

In contrast, the Canonical State Space technique will produce a generalized deterministic state transition matrix.

Algorithm Specifications

General Model

This section describes the four system identification techniques. Each was used to identify both the Attrition Simulation and the Mission Simulation models from sample (simulated) data. Each technique estimates a variant of the following general form of discrete, Linear Time Invariant (LTI) state-space model structure:

$$x(t) = Ax(t-1) + Bu(t) + \varepsilon(t)$$
$$y(t) = Cx(t) + e(t)$$

where t is the discrete time index, y is an M element (noisy) observation vector, x is an N element state vector (the number of entities in each element), u is an R element input vector, ε is an N element system noise vector, e is an M element observation noise vector, A is an N x N state transition matrix, B is an N x R input matrix, and C is an M x N observation matrix.

This section provides a detailed mathematical description of each of the system identification techniques proposed for this effort, i.e.,

- Canonical State Space
- Compartmental Model
- Maximum Entropy
- Hidden Markov Model (HMM)

The description includes the algorithm itself, how it is adapted to the selected simulation models, and methods for calculation of prior estimates of the model parameters.

Canonical State Space Technique

The algorithms described in this section are based on work by Guidorzi (1982, 1981, 1975).

Framework

Given a set of (noisy) system inputs and outputs:

u(1), u(2), u(3),.....u(T)
y(1),y(2), y(3),.....y(T)

where u(t) is an R element input vector and y(t) is an M element output vector such that

u(t) = u'(t) + d(u(t))
y(t) = y'(t) + d(y(t))

where d(u(t)),d(y(t)) are vectors of additive, zero mean, uncorrelated noise and u'(t) and y'(t) are the underlying noise free inputs and outputs.

The LTI system is defined as:

x(t+1) = Ax(t) + Bu(t) t = 1,2,.....T
y(t) = Cx(t)

with an algebraically equivalent form:

$$y_i(t+v_i) = \sum_{j=1}^{M}\sum_{k=1}^{v_{ij}} \alpha_{ijk} y_j(t+k+1) + \sum_{j=1}^{R}\sum_{k=1}^{v_i} \beta_{jik} u_j(t+k-1) \quad t = 1,2,.....T$$

General Algorithm

The approach uses 4 major phases:

1. Structural Identification
2. Parametric Identification
3. Conversion to Canonical State Space Form
4. Recovery of state vector

Structural Identification[5]

Determines a set of scalars: v_i, $i = 1,2,\ldots M$, ($\Sigma_i v_i = N$) and v_{ij}, which completely determine the canonical structure of A and C, where

$v_{ij} = v_i$ for $i=j$
$v_{ij} = \min(v_i+1,v_j)$ for $i>j$
$v_{ij} = \min(v_i,v_j)$ for $i<j$

By structure we mean that certain elements must necessarily contain the values 0,1, or a parameter (determined via the Parametric Identification phase). Structural identification is equivalent to determining the degrees of delay in equation error model structures. (This phase alone provides sufficient information to construct C).

Consider the set of input-output data

$$\left|\begin{matrix} y_1(t) & y_1(t+1) & \ldots \\ y_1(t+1) & y_1(t+2) & \ldots \\ \vdots & \vdots & \vdots \\ y_1(t+L) & y_1(t+L+1) & \ldots \end{matrix}\right| \vdots \left|\begin{matrix} y_M(t) & y_M(t+1) & \ldots \\ y_M(t+1) & y_M(t+2) & \ldots \\ \vdots & \vdots & \vdots \\ y_M(t+L) & \ldots & \ldots \end{matrix}\right| \begin{matrix} u_1(t) & \ldots \\ u_1(t+1) & \ldots \\ \vdots & \vdots \\ u_1(t+L) & \ldots \end{matrix} \left|\begin{matrix} \ldots \\ \ldots \\ \vdots \\ \ldots \end{matrix}\right| \begin{matrix} u_R(t) & \ldots \\ u_R(t+1) & \ldots \\ \vdots & \vdots \\ u_R(t+L) & \ldots \end{matrix}\right|$$

$= (y_1(t),y_1(t+1)\ldots|\ldots.|y_M(t),y_M(t+1)\ldots.|u_1(t),\ldots.|\ldots|u_R(t),\ldots..)$

Let:

$L_i(y_j) = (y_j(t),y_j(t+1), \ldots..y_j(t+i-1))$
$L_i(u_j) = (u_j(t),u_j(t+1), \ldots..u_j(t+i-1))$

$$R(\delta_1, \delta_2, \delta_3,\ldots \quad \delta_{M+R})=\{L_{\delta_1}(y_1),\ldots L_{\delta_M}(y_M) \mid L_{\delta_{M+1}}(u_1),\ldots L_{\delta_{M+R}}(u_R)\}$$

$$S(\delta_1, \delta_2, \delta_3,\ldots \quad \delta_{M+R})= R(\delta_1, \delta_2, \delta_3,\ldots \quad \delta_{M+R})' R(\delta_1, \delta_2, \delta_3,\ldots \quad \delta_{M+R})$$

The approach operates on a sequence of matrices, $S(\delta_1, \delta_2, \delta_3,\ldots \quad \delta_{M+R})$ where the indices, $\delta_1, \delta_2, \delta_3,\ldots \quad \delta_{M+R}$, correspond to the orders used in the underlying input/output data. The sequence of these matrices is constructed such that the values of the δ_k increase monotonically:

$S(2,1,\ldots,1)$ $S(2,2,\ldots.,1)$ $\ldots.S(2,2,\ldots..2)$ $S(3,2,\ldots.2)$ $S(3,3,\ldots.2)\ldots..$

---

[5] Note that in our identification of simulation models the system states are all directly observable (M=N). In that case we already know that all of the structural parameters v_i will equal one and this step may be skipped. This also implies that C=I.

To each S_i (i = 1,2,...M) we can associate a PPCRE (predicted percent reconstruction error) calculated as:

$$PPCRE_i = 100 * (\lambda_i/(L-1))^{1/2} / \sigma_i$$

S_i = the S matrix where index i has just increased by 1
$\lambda_i = det(S_i)/det(S_{i-1})$ (S_{i-1} is the matrix where index i has not yet been increased)
L = the length of i/o streams analyzed
σ_i = the standard deviation of the ith output stream y_i*

The algorithm examines the PPCRE in sequence. If the PPCRE associated with the matrix $S(\delta_1, \delta_2, \ldots\ldots\delta_i,\ldots \quad \delta_{M+R})$, where the last increased argument with respect to the the preceding matrix in the sequence is δ_i , does not decrease "significantly" with respect to the PPCRE for $S(\delta_1\text{-}1, \delta_2\text{-}1,\ldots. \delta_i\text{-}1,\ldots \quad \delta_{M+R}\text{-}1)$ then we fix v_i to δ_i-2. The sequence is then restarted at $S(\delta_1\text{-}1, \delta_2\text{-}1,.. \delta_i\text{-}2, \delta_{i+1}\text{-}1\ldots \quad \delta_{M+R}\text{-}1)$ with index *i* fixed. The algorithm resumes with

$$S(v_{i1}, v_{i2},\ldots., v_i, v_{i,i+1},\ldots)$$

and continues until all M v_i have been calculated. (Note that in the above description, an index will not be decreased if it has already been fixed).

We extended the original Guidorzi algorithm to handle two considerations: 1) the sum of the v_i's must be equal to a known previously determined value of N, and 2) it is difficult to *a priori* determine the decrease in PPCRE that should be considered significant.

Without modification, the algorithm above can produce an N that is either less than or greater than the desired value. If N is less than that desired, we can say that the algorithm was too willing to accept a decrease in PPCRE as "significant", while the opposite is true for an N greater than that desired. To handle this, an outer loop was constructed where the initial significance is set relatively high (a difference of 10 in PPCRE). If the basic algorithm ends with N too low, the significance is reduced by half and the algorithm is restarted. On the other hand, if N is about to become too large, the entire algorithm is terminated with the current set of v_i's.

Parametric Identification

This procedure determines the parameters α_{ijk}, which are placed into specific elements of A, and the parameters β_{ijk}, which are placed into specific elements of an intermediate matrix, $\overline{B}$. The parameters are determined via least-squares regression.

Let:

$$\gamma_s = (\alpha_{s11}....\alpha_{s1v_1} \mid ... \mid \alpha_{sM1}...\alpha_{sMv_M} \mid \beta_{1s1}...\beta_{1sv_s} \mid ... \mid \beta_{Rs1}....\beta_{Rsv_s})'$$
$$R_s = R(v_{s1},....v_{sM}, v_s....,v_s)$$

Then we solve for γ_s via

$$\gamma_s = (R_s' R_s)^{-1} R_s' y(t+v_s) \qquad s = 1,2,....M$$

Conversion to Canonical State Space Form

The matrices A and C are constructed directly from the parameters already determined. A=[Aij] where:

$$A_{ii} = \begin{bmatrix} 0 & & & \\ \vdots & & I_{v_i-1} & \\ 0 & & & \\ \alpha_{ii1} & \dots & \dots & \alpha_{iiv_i} \end{bmatrix} \qquad A_{ij} = \begin{bmatrix} 0 & \dots & \dots & \dots & \dots & 0 \\ \vdots & & & & & \vdots \\ 0 & & & & & 0 \\ \alpha_{ij1} & \dots & \alpha_{ijv_{ij}} & 0 & \dots & 0 \end{bmatrix}$$

$(v_i \text{ x } v_i)$ $\qquad$ $(v_i \text{ x } v_j)$

We also have:

$$C = \begin{bmatrix} 1 & 0 & \dots & \dots & \dots & \dots & \dots & \dots & \dots & .. & \dots & 0 \\ 0 & \dots & .... & 0 & 1 & 0 & \dots & \dots & \dots & \dots & \dots & 0 \\ . & . & . & . & . & . & . & . & . & . & . & . \\ 0 & \dots & \dots & \dots & \dots & \dots & \dots & 0 & 1 & 0 & \dots & 0 \end{bmatrix}$$

$\uparrow$ $\qquad$ $\uparrow$ $\qquad$ $\uparrow$

1 $\qquad$ (v_1+1) $\qquad$ $(v_1+...+v_{M-1}+1)$

Determination of the B matrix requires the construction of an intermediate matrix, $\overline{M}$. $\overline{M}$ is constructed from the parameters, α_{srk} while B is found from $B = \overline{M}^{-1}\overline{B}$ as follows: $\overline{M} = [M_{ij}]$ (i, j = 1,2,....M) where

$$M_{ii} = \begin{bmatrix} -\alpha_{ii2} & -\alpha_{ii3} & \dots & -\alpha_{iiv_i} & 1 \\ -\alpha_{ii3} & -\alpha_{ii4} & \dots & 1 & 0 \\ \vdots & \vdots & . & 0 & \vdots \\ -\alpha_{iiv_i} & 1 & 0 & .. & \vdots \\ 1 & 0 & \dots & \dots & 0 \end{bmatrix} \qquad M_{ij} = \begin{bmatrix} -\alpha_{ij2} & \dots & -\alpha_{ijv_{ij}} & 0 \\ \vdots & \dots & 0 & \vdots \\ -\alpha_{ijv_{ij}} & 0 & \dots & \vdots \\ 0 & & & 0 \\ 0 & \dots & \dots & 0 \end{bmatrix}$$

$(v_i \text{ x } v_i)$ $\qquad$ $(v_i \text{ x } v_j)$

$$\overline{B} = \begin{bmatrix} \overline{B}_1 \\ \vdots \\ \overline{B}_M \end{bmatrix} \qquad \overline{B}_i = \begin{bmatrix} \beta_{i11} & \cdots & \beta_{iR1} \\ \vdots & & \vdots \\ \beta_{i1v_i} & & \beta_{iRv_i} \end{bmatrix}$$

Recovery of the State Vector

The state vector is found as a function of the input-output sequences using the relation:

x(t) = V(z)y(t) - WZ(z)u(t)

V(z) and Z(z) contain various degrees of the delay operator, z^{-1}, and have structures that are determined from N,M,R and the v_i's.

$$V(z) = \begin{bmatrix} 1 & \cdots & 0 \\ z & & \vdots \\ \vdots & & \vdots \\ z^{v_1-1} & & \vdots \\ 0 & & 0 \\ \vdots & & \vdots \\ 0 & & 1 \\ \vdots & & z \\ \vdots & & \vdots \\ 0 & \cdots & z^{v_M-1} \end{bmatrix} \qquad Z(z) = \begin{bmatrix} I \\ zI \\ \vdots \\ z^{v_{Max}-2} \end{bmatrix}$$

(N x M) $\qquad (R(v_{Max}-1)\, x\, R) \qquad v_{Max} = \max_i(v_i)$.

W is a matrix that contains parameters from the B matrix.

$$W = \begin{bmatrix} 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ b_1 & 0 & \cdots & \cdots & \cdots & 0 \\ \vdots & b_1 & 0 & \cdots & \cdots & \vdots \\ b_{v_1-1} & \cdots & b_1 & 0 & \cdots & 0 \\ \vdots & & & & & \vdots \\ 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ b_{N-v_M+1} & 0 & \cdots & \cdots & \cdots & 0 \\ \vdots & b_{N-v_M+1} & 0 & \cdots & \cdots & \vdots \\ b_{N-1} & \cdots & b_{N-v_M+1} & 0 & \cdots & 0 \end{bmatrix} \qquad B = \begin{bmatrix} b_1 \\ \vdots \\ b_N \end{bmatrix}$$

$(N \times R(v_M\text{-}1))$

No modifications to the general algorithm are required for the simulation identification, nor are prior estimates required.

Compartmental Model Technique

The algorithms in this section have been adapted from Seber and Wild(1989).

Framework

Here we assume that our system can be described as a *continuous time* linear flow system with N states. State 1 is assumed to be an external or "sink" state. The state vector provides the number of customers in each state. We wish to estimate the fractional transfer coefficients γ_{sr} which are defined as the rate at which customers are passing from state r to state s, divided by the number of customers in state r. (Note that the subscripts are reversed with respect to the standard literature on Markov chains.) It can be shown that the intensity matrix (or instantaneous transfer-rate matrix), A, will have the following form:

$$A = \begin{matrix} 0 & \gamma_{12} & \gamma_{13}\cdots\cdots & \gamma_{1N} \\ 0 & -\Sigma_{s\neq 2}\gamma_{s2} & \gamma_{23}\cdots\cdots & \gamma_{2N} \\ 0 & \gamma_{32} & -\Sigma_{s\neq 3}\gamma_{s3}\cdots & \gamma_{3N} \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \gamma_{N2} & \gamma_{N3}\cdots\cdots & -\Sigma_{s\neq N}\gamma_{sN} \end{matrix}$$

We use θ_k to denote the estimate of the kth unknown γ_{sr}; that is, $\theta_1 = \gamma_{12}$, $\theta_2 = \gamma_{13}$, $\theta_{N-1} = \gamma_{1N}$, $\theta_N = \gamma_{23}$, etc., and that the transition-probability matrix P(t) will have the following form:

$$P(t) = \begin{matrix} 1 & p_{12}(t) & p_{13}(t)\cdots & p_{1N}(t) \\ 0 & p_{22}(t) & p_{23}(t)\cdots & p_{2N}(t) \\ \vdots & \vdots & \vdots \quad \vdots & \vdots \\ 0 & p_{N2}(t) & p_{N3}(t)\cdots & p_{NN}(t) \end{matrix}$$

We assume that we can observe the state vector over T time periods and will estimate the transfer coefficients from these observations. Let:

y_{ij} = the ith measurement on state j
$x_j(t)$ = customers in state j at time t, observed at times $t_1, t_2, \ldots . t_T$ to produce $y_{ij} = x_j(t_i)$
$y^{(i)} = (y_{i2}, y_{i3}, \ldots . y_{iN})' = (x_2(t_i), x_3(t_i) \ldots . x_N(t_i))'$

$$y = (y^{(1)\prime}, y^{(2)\prime}, \ldots . y^{(T)\prime})' = \begin{bmatrix} x_2(t_1) \\ x_3(t_1) \\ \vdots \\ x_N(t_1) \\ \vdots \\ x_2(t_T) \\ x_3(t_T) \\ \vdots \\ x_N(t_T) \end{bmatrix}$$

Note that state 1 is not included. Use of it would introduce an exact linear dependence in the data. It can be derived from knowledge of the other states at time t.
Let X(t) be the state occupied by any particular customer in the system at time t. Then:

$$E[x_j(t)] = \sum_{r=1}^{N} x_r(0) p_{jr}(0,t) \qquad \text{where } p_{jr}(0,t) = pr[X(t) = j \mid X(0) = r]$$

Since we are assuming that the Markov transition probabilities are time independent, then $p_{jr}(0,t) = p_{jr}(t)$, where $p_{jr}(t)$ is the j,rth element of the transition probability matrix $P^{(t)}$. It follows that:

$$E[x_j(t)] = \Sigma_r\, x_r(0) p_{jr}(t) \sum_{r=1}^{N} x_r(0) p_{jr}(t)$$

We can characterize the expected system response as:

$$f(\theta) = \begin{bmatrix} E[x_2(t_1)] \\ E[x_3(t_1)] \\ \vdots \\ E[x_N(t_1)] \\ E[x_2(t_2)] \\ E[x_3(t_2)] \\ \vdots \\ E[x_N(t_2)] \\ \vdots \\ E[x_2(t_T)] \\ E[x_3(t_T)] \\ \vdots \\ E[x_N(t_T)] \end{bmatrix}$$

Let $\varepsilon_{ij} = x_j(t_i) - E[x_j(t_i)]$ and $V(\theta)$ be the covariance matrix of the ε_{ij}

The terms of $V(\theta)$ are defined as follows:

$$\text{var}[x_j(t)] = \sum_{r=1}^{N} x_r(0) p_{jr}(t)\,(1 - p_{jr}(t)) \qquad \text{(diagonal elements)}$$

$$\text{cov}[x_j(t), x_k(t)] = -\sum_{r=1}^{N} x_r(0) p_{jr}(t)\, p_{kr}(t)$$

$$\text{cov}[x_j(t), x_k(t+\tau)] = p_{kj}(\tau)\text{var}[x_j(t)] + \Sigma_{r\neq j}\, \text{cov}[x_j(t), x_r(t)]\, p_{kr}(\tau)$$

General Solution Technique

To solve for the γ_{sr} (values for θ_k) we use an iterated two stage estimation procedure. That is, for $i = 1,2,\ldots$, obtain θ^{i+1} by minimizing:

$$[y - f(\theta)]'V^{-1}(\theta^0)[y - f(\theta)]$$

via the following series of Gauss-Newton update steps:

$$\theta^{i+1} = \theta^i + [\nabla f(\theta^i)'V^{-1}(\theta^0)\, \nabla f(\theta^i)]^{-1}\, \nabla f(\theta^i)'V^{-1}(\theta^0)[y - f(\theta^i)]$$

Upon calculating optimal parameter values for a given $V(\theta^0)$, a new V is computed using the new parameters. The optimization is then repeated using the revised $V(\theta^0)$. The process continues until convergence is achieved (detailed steps are shown below).

To compute $f(\theta^i)$ and $\nabla f(\theta^i)$ we can use a spectral decomposition where

$$A = S\Lambda S^{-1}$$

S is a matrix such that the rth column is a right eigenvector of A corresponding to the eigenvalue, λ_r, and $\Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \ldots \lambda_N)$. It can be shown that

$$P(t) = e^{At} = Se^{\Lambda t}S^{-1}$$

where $e^{\Lambda t} = \mathrm{diag}(e^{\lambda 1 t}, e^{\lambda 2 t}, \ldots e^{\lambda N t})$. This allows use to compute $f(\theta^i)$ via

$$f(t) = P(t)x(0)'$$

It should be noted that the above decomposition strategy is not robust. In some cases the decomposition does not exist; S may not be invertible (is singular) or the matrix P can contain complex numbers. (Indeed both of these situations occurred frequently with both of our simulation models). Sophisticated numerical approaches have been developed to handle these types of situations (see for example: Bates and Watts, 1988). Practically speaking, the developers of MATLAB (The Mathworks, 1999 - the implementation language of the algorithms in this study) have worked out these complexities in their built-in function "expm(At)" which computes e^{At} without a spectral decomposition.

It remains to define the elements of $\nabla f(\theta^i)$. It can be shown that in the case of known initial conditions and no inputs:

$$\partial x(t) / \partial \theta_k = e^{At} * [A^{(k)} e^{At}]\, x(0) = SB^{(k)}S^{-1}\, x(0)$$

where $A^{(k)} = \partial A / \partial \theta_k$ and "*" denotes the mathematical convolution operator. Note that for a given k, $A^{(k)}$ will be all zeros except for a 1 in the position indicated by $k \equiv (s,r)$ and a -1 in the position A(r,r). The matrix $B^{(k)}$ is defined with elements:

$$b_{sr}^{(k)} = g_{sr}^{(k)} \lambda_{sr}(t)$$

$$\lambda_{sr}(t) = \begin{cases} (e^{\lambda_s t} - e^{\lambda_r t}) / (\lambda_s - \lambda_r) & \lambda_s \neq \lambda_r \\ te^{\lambda_s t} & \lambda_s \approx \lambda_r \end{cases}$$

The terms $g_{sr}^{(k)}$ are elements of the matrix, $G^{(k)}$, defined as

$$G^{(k)} = [S^{-1} A^{(k)} S]$$

As with the computation of expected values, the above approach relies on the S matrix of eigenvectors, and suffers the same potential weaknesses. Again we can rely on the MATLAB developers to provide a numerical solution, which they have done with the built-in function "conv2(A,B)" which provides the convolution of two matrices A and B.

Simulation Estimation Algorithm

Prior Estimates

Let:

Y_{tj}^{r} = the rth element of the observation vector at the beginning of time period t in the jth simulation run (amount of "material" in compartment r).

d_{tj}^{sr} = the amount of flow from state vector element (compartment) r to state vector element s during time period t in the jth simulation run.

Δt = the time between simulation observations (a constant value)

In the Mission Simulation we know the values of the f_{tj}^{sr} since we are tracking individual aircraft as they move from state to state. In that case we can easily estimate the fractional transfer coefficients, $\hat{\gamma}_{sr}$, by using the average historical rate:

$$\hat{\gamma}_{sr} = \frac{\sum_{j=1}^{nruns} \sum_{t=1}^{T_j - 1} d_{tj}^{sr} \Big/ \Delta t Y_{tj}^{r}}{\sum_{j=1}^{nruns} (T_j - 1)}$$

In the Attrition Simulation we don't know the specific state to state flows by aircraft so we instead estimate the fractional transfer coefficients via a regression based technique.

Let:

$\delta_{rj}(t) = (Y^r_{t+1,j} - Y^r_{tj})/\Delta t$ t = 1,2,....T_j-1; j = 1,2,...nruns; r = 1,2,...N

$\overline{Y}^r_{tj} = (Y^r_{tj} + Y^r_{t+1,j})/2$ t = 1,2,....T_j-1; j = 1,2,...nruns; r = 2,...N

The $\delta_{rj}(t)$ can be interpreted as the average change in state r per unit time during period t of run j, and the $\overline{Y}^r_{tj}$ are the average state values during period t of run j. We can form

$$\delta_{rj} = (\delta_{rj}(1), \delta_{rj}(2), \ldots \delta_{rj}(T_j - 1))' \ , \delta_r = (\delta'_{r1}, \delta'_{r2}, \ldots \delta'_{r,nruns})' ; \ \delta = (\delta'_1, \delta'_2, \ldots \delta'_N)'$$

$$\overline{Y}_{tj} = (\overline{Y}^2_{tj}, \overline{Y}^3_{tj}, \ldots \overline{Y}^N_{tj}) \ ; \ \overline{Y}_j = (\overline{Y}_{1j}, \overline{Y}_{2j}, \ldots \overline{Y}_{T_j,j})' \ ; \ \overline{Y} = (\overline{Y}_1, \overline{Y}_2, \ldots \overline{Y}_{nruns})' ;$$

and $\hat{\vartheta} = (\hat{\theta}_1, \hat{\theta}_2, \ldots \hat{\theta}_{(N-1)(N-1)})'$ is the vector of unknown rate coefficients whose values we seek (the nondiagonal elements in columns 2 through N of the rate matrix, A). Some manipulation is still required to set up the regression matrix to account for the balancing conditions in each element (net increase = flow in - flow out).

Let the regression matrix be

$$\tilde{Y} = (\overline{Y}^1, \overline{Y}^2, \ldots \overline{Y}^N)'$$

where each $\overline{Y}^r$ has (N-1)*(N-1) columns. The nonzero columns of $\overline{Y}^r$ are constructed as follows:

- if r =1, the first (N-1) columns are set to $\overline{Y}$
- if r >1:
  1. Divide the columns of $\overline{Y}^r$ into N groups, where the first group has (N-1) columns, and the (N-1) remaining groups have (N-2) columns.
  2. Modify $\overline{Y}$ by removing the (r-1)st column; call the resulting matrix $\overline{Y}_{/r-1}$ and call the removed column, $Y_{\bullet(r-1)}$.
  3. Set the rth group of columns in $\overline{Y}^r$ to $\overline{Y}_{/r-1}$.
  4. Set the (r-1)st column of every other group in $\overline{Y}^r$ to $-Y_{\bullet(r-1)}$.

We can now define the total set of equations as:

$$\tilde{Y}\hat{\theta} = \delta$$

We can solve for $\hat{\theta}$ using non-negative least squares. Non-negative least squares is similar to ordinary least squares but constrains the coefficients to be non-negative.

Algorithm

In the Mission Simulation, the algorithm is performed on each cohort of arriving aircraft while in the Attrition Simulation we do not need arrivals to perform the identification. In that case we have just 1 "arriving" cohort, the initial weapon systems.

Let H be the number of cohorts in the simulation[6].

For h = 1:H

{

a) Set the time indices of the data so that t=1 corresponds to the arrival period of cohort *h*

b) Select an initial set of prior estimates, $\hat{\vartheta}$, for the unknown elements of A.

c) Set the composite rate matrix, A_h, to zeros

For j = 1:nruns

a) $\vartheta^0 = \{\hat{\vartheta}_i : \hat{\vartheta}_i > 0; i = 1,2,..(N-1)^2\}$; $\theta^v = \theta^0$; a = 0

b) Compute the spectral decomposition of $A(\theta^0)$, i.e., find Λ and S or an equivalent technique to find $P(t) = e^{At}$ for t = 1,2,... T

c) Compute E[x(t)] (via P(t)'s) to obtain $f(\theta^0)$ terms

d) Compute $V(\theta^0)$

e) objt = objv = $(y- f(\theta^0))'V^{-1}(\theta^0)\,(y- f(\theta^0))$; objtold = objvold = inf.; search_tolerance = objv/10000

While (objvold – objv > search_tolerance)

objvold = objv

While (objtold – objt > search_tolerance)

objtold = objt

Step 1: Compute $\partial x(t) / \partial \theta_k^a$ via convolution method for k = 1,2,... length(θ^a) to obtain $\nabla f(\theta^a)$ terms for t = 1,2,... T_j

---

[6] In the Mission Simulation the algorithm is performed separately on each cohort of arriving aircraft while in the Attrition Simulation we do not need arrivals to perform the identification. In that case we have just 1 "arriving" cohort, the initial weapon systems.

Step 2: Compute θ^{a+1} using a modified Gauss-Newton update formula and $V = V(\theta^v)$:

$$\Delta\theta = [\nabla f(\theta^a)' V^{-1}(\theta^v)\, \nabla f(\theta^a)]^{-1}\, \nabla f(\theta^a)' V^{-1}(\theta^v)[y - f(\theta^a)]$$

Ensure nonnegative θ:

$$\alpha_{max} = \max_{0 \le \alpha \le 1} : \theta^a + \alpha\, \Delta\vartheta \ge 0$$

$$\theta^{a+1} = \theta^a + \alpha_{max}\Delta\theta$$

Step 3:

a) Compute the spectral decomposition of $A(\theta^{a+1})$ (Λ and S) or an equivalent technique to find $P(t) = e^{At}$ for $t = 1,2,\ldots T$

b) Compute $E[n(t)]$ (via $P(t)$'s) to obtain $f(\theta^{a+1})$ terms

Step 4. $objt = (y - f(\theta^{a+1}))' V^{-1}(\theta^v)\, (y - f(\theta^{a+1}))$; $a = a+1$

end while

$\theta^v = \theta^{a+1}$; $\theta^0 = \theta^{a+1}$; $a = 0$

Compute $V(\theta^v)$

$objv = (y - f(\theta^v))' V^{-1}(\theta^v)\, (y - f(\theta^v))$

end while

Add in matrix values:

$A_h = A_h + A(\theta^v)$

end nruns

Compute final weighted average composite rate matrix:

$\mathbf{A_h = A_h/nruns}$

}

The identified system is the sum of the identified cohort systems, appropriately indexed for the current time.

Maximum Entropy Technique

The Maximum Entropy technique is based on methods described in Golan, et. al.. (1996).

Framework

In this case we will model our simulations in the form of the following LTI system:

$x(t)= Ax(t-1) + Bu(t) + \varepsilon(t) \quad t = 2,3,\ldots.T$

where x(t) is a known N x 1 state vector providing the numbers of entities in every state, u(t) is a known R x 1 input vector providing the number of entities entering each state, A is an unknown N x N state-transition matrix, and B is a known, or partially known, N x R input transformation matrix. The terms $\varepsilon(t)$ are unknown N x 1 vectors of system errors. We assume that the $\varepsilon(t)$ are distributed uniformly around 0 and their covariance matrix is unknown.

In the Attrition Simulation, R=N and B=I, so that there are no parameters in B to estimate. In the Mission Simulation R=1, and B is an N x 1 vector of probabilities mapping the entering plane to one of the "queue states". Therefore, B will have "queueStates" number of non-zero entries, since we assume that the plane arrives with a full tank of fuel.

Define the following discrete support points:

$z_{sr} = [z_{sr1}, z_{sr2} \ldots\ldots z_{srD}]$ $\quad s = 1,2,..,N;\ r = 1,2,\ldots,N$

$zb_r = [zb_{r1}, zb_{r2}, \ldots.zb_{r,Db}]$ $\quad r = 1,2,\ldots N$

$v^{\varepsilon}_r(t) = [v^{\varepsilon}_{r1}(t)\ v^{\varepsilon}_{r2}(t) \ldots\ldots v^{\varepsilon}_{rE}(t)]$ $\quad r = 1,2,\ldots N;\ t = 2,3,\ldots\ldots T$

The discrete support points have corresponding probabilities:

$p_{sr} = [p_{sr1}, p_{sr2} \ldots\ldots p_{srD}]$ $\quad s = 1,2,..,N;\ r = 1,2,\ldots,N$

$pb_r = [pb_{r1}, pb_{r2}, \ldots.pb_{r,Db}]$ $\quad r = 1,2,\ldots N$

$w^{\varepsilon}_r(t) = [w^{\varepsilon}_{r1}(t)\ w^{\varepsilon}_{r2}(t) \ldots\ldots w^{\varepsilon}_{rE}(t)]$ $\quad r = 1,2,3,\ldots N;\ t = 2,3,\ldots\ldots T$

such that:

$a_{sr} = z_{sr}\, p_{sr}'$ $\quad s = 1,2,..,N;\ r = 1,2,\ldots,N$

$\varepsilon_r(t) = v^{\varepsilon}_r(t)\, w^{\varepsilon}_r(t)'$ $\quad r = 1,2,3,\ldots N;\ t = 2,3,\ldots\ldots T$

$b_{r1} = z_{br} p_{br}'$ $\quad r = 1,2,\ldots.N$

Then let:

$\mathbf{1}_D = [1,1,\ldots 1]$ (vector of length D)
$\mathbf{1}_{Db} = [1,1,\ldots 1]$ (vector of length Db)
$\mathbf{1}_E = [1,1,\ldots 1]$ (vector of length E)

The probabilities must also satisfy:

$\mathbf{1}_D\, p_{sr}' = 1$	$s = 1,2,..,N;\ r = 1,2,\ldots,N$
$\mathbf{1}_{Db}\, pb_r' = 1$	$r = 1,2,\ldots,N$
$w^{\varepsilon}_r(t)\, \mathbf{1}_E' = 1$	$r = 1,2,3,\ldots N;\ t = 2,3,\ldots\ldots T$
$0 \le p_{srk} \le 1$	$s = 1,2,..,N;\ r = 1,2,\ldots,N;\ k = 1,2,\ldots D$
$0 \le pb_{rk} \le 1$	$r = 1,2,\ldots,N;\ k = 1,2,\ldots Db$
$0 \le w^{\varepsilon}_{rk}(t) \le 1$	$r = 1,2,..,N;\ t = 1,2,\ldots,T;\ k = 1,2,\ldots E$

Now also define prior estimates (relative weights) corresponding to each of the probabilities defined above:

$q_{sr} = (q_{sr1}, q_{sr2} \ldots\ldots q_{srD})$	$s = 1,2,..,N;\ r = 1,2,\ldots,N$
$qb_r = (q_{r1}, q_{r2} \ldots\ldots q_{rDb})$	$r = 1,2,\ldots,N$
$q^{\varepsilon}_r(t) = (q^{\varepsilon}_{r1}(t), q^{\varepsilon}_{r2}(t), \ldots q_{rE}(t))$	$r = 1,2,..,N;\ t = 2,3,\ldots,T$

We can then form a constrained nonlinear mathematical program[7] to determine the p_{srk}, pb_{rk}, and the $w^{\varepsilon}_{rk}(t)$:

Minimize: $$\sum_{s=1}^{N}\sum_{r=1}^{N}\sum_{k=1}^{D} p_{srk} \ln\left(\frac{p_{srk}}{q_{srk}}\right) + \sum_{r=1}^{N}\sum_{k=1}^{Db} pb_{rk} \ln\left(\frac{pb_{rk}}{qb_{rk}}\right) + \sum_{r=1}^{N}\sum_{k=1}^{E}\sum_{t=2}^{T} w^{\varepsilon}_{rk}(t) \ln\left(\frac{w^{\varepsilon}_{rk}(t)}{q^{\varepsilon}_{rk}(t)}\right)$$

Subject to:

$x(t) = [ZP]x(t-1) + Bu(t) + V^{\varepsilon}(t)w^{\varepsilon}(t)$	$t = 2,3,\ldots T$
$\mathbf{1}_D\, p_{sr}' = 1$	$s = 1,2,..,N;\ r = 1,2,\ldots,N$
$\mathbf{1}_{Db}\, p_r' = 1$	$r = 1,2,\ldots,N$
$w^{\varepsilon}_r(t)\, \mathbf{1}_E' = 1$	$r = 1,2,3,\ldots N;\ t = 2,3,\ldots\ldots T$
$0 \le p_{srk}$	$s = 1,2,..,N;\ r = 1,2,\ldots,N;\ k = 1,2,\ldots D$
$0 \le pb_{rk}$	$r = 1,2,\ldots,N;\ k = 1,2,\ldots Db$
$0 \le w^{\varepsilon}_{rk}(t)$	$r = 1,2,..,N;\ t = 2,3,\ldots,T;\ k = 1,2,\ldots E$

[7] Assume that terms of the form x ln(x) are equal to 0 for x = 0.

where:

$ZP_{\bullet r}$ = r^{th} "column" of [ZP] = diag(z_{1r}, z_{2r},....z_{Nr})*(p_{1r}', p_{2r}',.... p_{Nr}')' r=1,2,....N

B = diag(zb_1, zb_2, ...zb_N)*(pb_1', pb_2', ... pb_N')'

$V^{\varepsilon}(t)$ = diag($v^{\varepsilon}_1(t)$, $v^{\varepsilon}_2(t)$, ... $v^{\varepsilon}_N(t)$) t=2,3,....T

$w^{\varepsilon}(t)$=($w^{\varepsilon}_1(t)'$, $w^{\varepsilon}_2(t)'$,.... $w^{\varepsilon}_N(t)'$)' t=2,3,...T

In the Mission Simulation we also constrain the columns of A and B to sum to 1.0:

$$\sum_{s=1}^{N}\sum_{k=1}^{D} z_{srk}\, p_{srk} = 1 \qquad r=1,2,\ldots N$$

$$\sum_{s=1}^{N}\sum_{k=1}^{Db} zb_{sk}\, pb_{sk} = 1$$

Simulation Estimation Algorithm

Prior Estimates

Attrition Simulation

To find the support points, z_{srk}, for the probabilities, p_{srk}, we will perform regressions on the state element values in each simulation run.

We use the convention that state 1 represents the "sink" (destroyed weapon systems). With six weapon system types, three on each side, N=7 and the system transition matrix must be of the form shown in Figure 5.

Let:

Y_{tj}^{r} = the rth element of the observation vector in time period t in the jth simulation run;
r =1,2,...N; j = 1,2,...nruns; t=1,2,... T_j
$Y_{tj} = (Y_{tj}^{1} - Y_{t-1,j}^{1}, Y_{tj}^{2}, Y_{tj}^{3}, \ldots Y_{tj}^{N})'$; t=2,3,... T_j , $Y^{j} = (Y'_{2j}, Y'_{3j}, \ldots Y'_{T_j j})'$
$x_{tj} = (Y_{tj}^{2}, Y_{tj}^{3}, \ldots Y_{tj}^{N}), X_{tj} = diag(x_{tj}, x_{tj}m'_1, \ldots x_{tj}m'_{N-1})$, where $m_i = (m_{i1}, m_{i2}, \ldots m_{iN-1})$
and

$$m_{ij} = \begin{cases} 1 & \text{if } i,j \le N_B \text{ OR } i,j > N_B \\ -1 & \text{otherwise} \end{cases}$$

is the "mask" function to enforce the matrix form in the table above. (N_B is the number of weapon system types on the "Blue" side.) Then if
$X^j = (X_{1j}, X_{2j}, ..X_{T_j j})'$, $a_{i\bullet} = (a^j_{i2}, a^j_{i3}, ...a^j_{iN})'$, $A^j = (a^j_{1\bullet}', a^j_{2\bullet}', ...a^j_{N\bullet}')'$,
for each run j we can specify:

$$\mathbf{X^j A^j = Y_j}$$

which we can solve for A^j using nonnegative least squares. To attain current estimates of the elements of A^j we again utilize our mask vector so that:

$\hat{A}^j_{1\bullet} = (1, A^j_{1\bullet})$, and $\hat{A}^j_{i\bullet} = (0, A^j_{i2} m_{i1}, A^j_{i3} m_{i2}, ...A^j_{iN} m_{i,N-1})$ for i = 2,3,...N

Mission Simulation

To find the support points, z_{srk}, for the p_{srk} we can proceed as follows. Let:

Y^r_{tj} = the rth element of the observation vector at the beginning of time period t in the jth simulation run (amount of "material" in compartment r).
d^{sr}_{tj} = the amount of flow from state vector element (compartment) r to state vector element s during time period t in the jth simulation run.
In the Mission Simulation we know the values of the f^{sr}_{tj} since we are tracking individual aircraft as they move from state to state. In that case we can easily estimate the state transition coefficients of a given run j by extracting the average historical rate from the simulation:

$$a^j_{sr} = \frac{\sum_{t=1}^{T_j-1} d^{sr}_{tj}}{\sum_{t=1}^{T_j-1} Y^r_{tj}} \qquad s = 1,2,\ldots N;\ r = 3,\ldots N$$

Note that states 1 and 2 are "sinks" corresponding to the "destroyed" and "returning" states. Therefore we already know that columns 1 and 2 are all zeros with the exception of a "1" in the 1st and 2nd rows respectively.

$\hat{A}^j_{1\bullet} = (1,0,a^j_{1\bullet})$,
$\hat{A}^j_{2\bullet} = (0,1,a^j_{2\bullet})$,
and $\hat{A}^j_{i\bullet} = (0,0,a^j_{i\bullet})$ for i = 2,3,...N

Note that the resulting values will be average transition proportions between the values of zero and one.

To get estimates for the support points zb_{rk} for the pb_{rk} we can use the historical average proportion of aircraft that enter the simulation in each state. Let:

$\hat{B}^j = (b_1^j, b_2^j, ... b_N^j)$ where b_i^j is the proportion of aircraft entering state i during run j.
$\overline{B} = (\hat{B}^1, \hat{B}^2, ... \hat{B}^{nruns})$

Our mean estimate for parameter j, j = 1,2,... N, is the mean of the values in column j. Similarly, a standard deviation can be computed for each parameter j from the data in the corresponding column. From these, we can calculate a 95% (say) confidence interval for each parameter. One support point will correspond to each end of this confidence interval while any additional support points will be evenly distributed over the interval.

Attrition and Mission Simulation

Let $\overline{A} = (\hat{A}^1, \hat{A}^2, ... \hat{A}^{nruns})$ be a matrix where each row i represents the vector of parameters estimates from run i. Our mean estimate for parameter j, j = 1,2,... N*N, is the mean of the values in column j. Similarly, a standard deviation can be computed for each parameter j from the data in the corresponding column. From these, we can calculate a 95% (say) confidence interval for each parameter. One support point will correspond to each end of this confidence interval while any additional support points will be evenly distributed over the interval.

We can also use our statistical model to find a range of support points, $v^{\varepsilon}_r(t)$, for the noise terms, $\varepsilon(t)$. Let e_{tj}^r be the residual term for the rth element from the jth simulation run at time t. The residual is the difference between the observed simulation value, Y_{tj}^r, and the estimated value x_t^r, resulting from a model whose parameters are estimated as described above. We can set an upper support point, $v^{\varepsilon}_{rE}(t)$, to $\max_j(abs(e_{tj}^r))$ and a lower support point, $v^{\varepsilon}_{rI}(t)$ to $-\max_j(abs(e_{tj}^r))$, with any other support points distributed evenly across the so defined interval[8].

We set the q_{srk}, qb_{rk}, and $q^{\varepsilon}_{rk}(t)$ to 1/D, 1/Db, and 1/E, respectively, to provide uniform weights for the support points. We still need initial estimates of the p_{srk}, pb_{rk}, and $w^{\varepsilon}_{rk}(t)$. These are found by solving a linear programming version of the problem ("linprog" in the MATLAB Optimization Toolbox). That is, we formulate the constraints to the problem identically to the setup for the nonlinear programming version, but instead minimize a linear sum of the probabilities.

[8] In practice, we found that the algorithm behaves better when we also put a floor on the values of the $v^{\varepsilon}_{rE}(t)$ of say, 1.0, rather than 0.

Algorithm

Step 0:

a) Set D, Db, and E , the number of support points per element of each type.
b) Set $T = \max_j(T_j)$
c) Set the support weights: $q_{sr} = (q_{sr1}, q_{sr2} \ldots\ldots q_{srD})$ (s = 1,2,..,N; r = 2,3,…,N); $qb_r = (q_{r1}, q_{r2} \ldots\ldots q_{r,Db})$ (r = 2,3,…,N); $q_r^{\varepsilon}(t) = (q_{r1}^{\varepsilon}(t), q_{r2}^{\varepsilon}(t), \ldots q_{rE}^{\varepsilon}(t))$ (r = 2,3,..,N, t = 2,3,…,T) to 1/D, 1/Db, and 1/E respectively.
d) Using the data from the nruns simulations, determine the support points: $z_{sr} = [z_{sr1}, z_{sr2} \ldots\ldots z_{srD}]$ (s = 1,2,..,N, r = 2,3,…,N); $zb_{sr} = [z_{r1}, z_{r2} \ldots\ldots z_{r,Db}]$ (r = 2,3,…,N); $v^{\varepsilon}_r(t) = [v^{\varepsilon}_{r1}(t)\ v^{\varepsilon}_{r2}(t) \ldots\ldots v^{\varepsilon}_{rE}(t)]$ (r = 1,2,3,…N, t = 2,3,……T)
e) Set the composite matrices, A and B, to all zeros

for j = 1:nruns

{

Step j1. From the simulation data in iteration, j, $Y_{tj} = (Y_{tj}^1, Y_{tj}^2, \ldots Y_{tj}^N)$ and U_{tj}, t = 1,2,…T_j, construct the constrained nonlinear mathematical program that, when solved, will provide the values for: $p_{sr} = [p_{sr1}, p_{sr2} \ldots\ldots p_{srD}]$ s = 1,2,..,N, r = 2,3,…,N; $pb_r = [p_{r1}, p_{r2} \ldots\ldots p_{r,Db}]$ r = 2,3,…,N; $w^{\varepsilon}_r(t) = [w^{\varepsilon}_{r1}(t)\ w^{\varepsilon}_{r2}(t) \ldots\ldots w^{\varepsilon}_{rE}(t)]$ r = 1,2,3,…N, t = 2,3,……T_j. Let **Aeq** be the resulting constraint matrix for all equality constraints and **beq** the corresponding right-hand side vector. To determine an initial solution, solve the linear programming model: min **c**x0 subject to **Aeq**x0 = **beq**, where **c** is a vector of all ones. The initial estimates, x0, are taken from the resulting solution vector

Step j2. Use the MATLAB "fmincon" nonlinear programming algorithm (large scale version) to solve for the unknown parameters. Use the results to construct the posterior matrices $\tilde{A}_j, \tilde{B}_j$.

Step j.4. Set A = A + $\tilde{A}_j$, B = B + $\tilde{B}_j$,

}

Final Step. Calculate the final estimated posterior matrices: A=A/nruns, B=B/nruns.

HMM Technique

The algorithms in this section are based on work by Elliot, et. al. (1997).

Framework

Here we consider a discrete *Hidden Markov Model* (HMM) of the following form:

$X_{t+1} = AX_t + \varepsilon_{t+1}$ $\quad$ $t = 0,1,\ldots T\text{-}1$
$Y_{t+1} = CX_t + e_{t+1}$ $\quad$ $t = 0,1,\ldots T\text{-}1$

where:

$X_t \in S_X = \{g_1, g_2, ,\ldots.g_N\}$; $g_r = (0, \ldots 1,0\ldots.,0)'$ (vector of 0's with 1 in i^{th} position)
$Y_t \in S_Y = \{f_1, f_2, ,\ldots.f_M\}$; $f_r = (0, \ldots 1,0\ldots.,0)'$ (vector of 0's with 1 in i^{th} position).

Accordingly:
$\mathbf{1}_N X_t = 1$
$\mathbf{1}_M Y_t = 1$

where $\mathbf{1}_n$ is a row vector of ones with dimension n. A and C are matrices of transition probabilities, such that:

$$\sum_{s=1}^{N} a_{sr} = \sum_{s=1}^{M} c_{sr} = 1$$

ε_t and e_t are driving noise and measurement noise in the form of *Martingale increments* that satisfy:

$E[\varepsilon^i_{t+1}] = 0 \quad E[e^i_{t+1}] = 0 \quad \mathbf{1}_N\varepsilon_t = 0 \quad \mathbf{1}_M e_t = 0$
$\varepsilon_{t+1} = \mathrm{diag}(AX_t) - A\,\mathrm{diag}X_tA'$
$e_{t+1} = \mathrm{diag}(CX_t) - C\,\mathrm{diag}X_tC'$

Recursive Estimators

The revised estimates $\widetilde{A}_{sr}, \widetilde{C}_{sr}$ of the parameters A_{sr} , C_{sr} at time t can be determined via:

$$\widetilde{A}_{sr}(t) = \gamma_t(J_t^{rs}) \big/ \gamma_t(O_t^r) \qquad 1 \le r \le N;\ 1 \le s \le N$$

$$\widetilde{C}_{sr}(t) = \gamma_t(T_t^{rs}) \big/ \gamma_t(O_t^r) \qquad 1 \le r \le N;\ 1 \le s \le M\text{-}1$$

$$\widetilde{C}_{Mr}(t) = 1 - \sum_{s=1}^{M-1} \widetilde{C}_{sr}(t) \qquad 1 \le r \le N$$

where:

J_t^{rs} = the number of jumps from state g_r to state g_s up to time t

O_{t+1}^r = the number of occasions up to time t for which the Markov chain X has been in state g_r (occupation time)

T_t^{rs} = the number of times up to time t that the observation process is in state f_s given the Markov chain at the preceeding time is in state g_r (state to observation transitions).

$\gamma_t(H_t)$ = the expectation under the change of measure of the random variable (vector process) H_t

Let:

$c_j \equiv$ the jth column of C

$a_j \equiv$ the jth column of A

$$c_s(Y_t) = M\prod_{r=1}^{M} c_{rs}^{Y_t^r}$$

Then define $\gamma_{t,t}(J_t^{rs}), \gamma_{t,t}(O_t^r), \gamma_{t,t}(T_t^{rs})$, via the recursive functions:

$$\gamma_{t+1,t+1}(J_{t+1}^{rs}) = \sum_{j=1}^{N} c_j(Y_{t+1})(\gamma_{t,t}(J_t^{rs})'g_j)a_j + c_r(Y_{t+1})(q_t'g_r)a_{sr}g_s \quad t = 0,1,2,\ldots T\text{-}1$$

$$\gamma_{t+1}(J_{t+1}^{rs}) = \mathbf{1_N}\,\gamma_{t+1,t+1}(J_{t+1}^{rs}) \qquad t = 0,1,2,\ldots T\text{-}1$$

$$\gamma_{t+1,t+1}(O_{t+1}^r) = \sum_{j=1}^{N} c_j(Y_{t+1})(\gamma_{t,t}(O_t^r)'g_j)a_j + c_r(Y_{t+1})(q_t'g_r)a_r \qquad t = 0,1,2,\ldots T\text{-}1$$

$$\gamma_{t+1}(O_{t+1}^r) = \mathbf{1_N}\,\gamma_{t+1,t+1}(O_{t+1}^r) \qquad t = 0,1,2,\ldots T\text{-}1$$

$$\gamma_{t+1,t+1}(T_{t+1}^{rs}) = \sum_{j=1}^{N} c_j(Y_{t+1})(\gamma_{t,t}(T_t^{rs})'g_j)a_j + M(q_t'g_r)(Y_{t+1}'f_s)c_{sr}a_r \qquad t = 0,1,2,\ldots T\text{-}1$$

$$\gamma_{t+1}(T_{t+1}^{rs}) = \mathbf{1_N}\,\gamma_{t+1,t+1}(T_{t+1}^{rs}) \qquad t = 0,1,2,\ldots T\text{-}1$$

$$\text{where } q_{t+1} = \sum_{j=1}^{N} c_j(Y_{t+1})(q_t'g_j)a_j; \quad q_t = (q_t(g_1),\ldots\ldots q_t(g_N))' \qquad t = 0,1,2,\ldots T\text{-}1$$

The $q_t(g_r)$ are the unnormalized conditional probability distribution for state *r* at time *t*. Thus the normalized estimates are:

$$p_t(g_r) = \frac{q_t(g_r)}{\sum_{r=1}^{N} q_t(g_r)}$$

Simulation Estimation Algorithm

Prior Estimates

Let Y_{tj}^{r} be the rth element of the observation vector at time period t from the jth simulation run (indicator for state r). Then we can define the elements of the prior transition matrix $\hat{A}$ as

$$\hat{A}_{sr} = \frac{\sum_{j=1}^{nruns} \sum_{t=2}^{T_j} \Delta(Y_{tj}^{s}, Y_{t-1,j}^{r})}{\sum_{s=1}^{N} \sum_{j=1}^{nruns} \sum_{t=2}^{T_j} \Delta(Y_{tj}^{s}, Y_{t-1,j}^{r})}$$

where $\Delta(Y_1, Y_2) = \begin{cases} 1 \text{ if } Y_1 = 1, Y_2 = 1 \\ 0 \ \textit{otherwise} \end{cases}$ (AND function)

Let q_{0j} be the prior estimate for the state vector in period 0 of the jth simulation run. Note that Y_{1j}, the first observation, is one period later; that is, after a pass through the transition matrix. To properly handle this, we define a "dummy" initial state, say, N+1. We set up the model so that at time 0 we are in state N+1 with probability 1 and that we will transition to the state observed in period 1 with probability 1. Thus, the (N+1)[st] element of q_{0j} is set to 1 and all other elements of the vector are set to 0. We add row N+1 and column N+1 to our prior matrix, Ahat. The new row and column contain zeros with the exception of the kth row of the (N+1)st column, which is set to 1. Similarly, we add an (N+1)[st] column to our Y data containing all zeros. Now the algorithm will perform properly for our situation.

In our state space *model* of the Attrition Simulation we do not have "simulation entities" in the sense of multiple interacting entities traversing states. Instead the system itself is a single entity that traverses possible Red/Blue force strength combinations. We know the initial state since we always know the initial Red/Blue force strength. The Mission Simulation is somewhat different. We know the *historical* initial state of each entity (aircraft) in the simulations from the "statesByPlane" output file. However, in our state space model, we do not know the precise starting state of an aircraft arriving at an arbitrary time in the trajectory of that model. In that case, to determine a starting state we must make use of recorded frequency data regarding the states which aircraft enter[9]. For additional fidelity, these probabilities can also easily be made conditional on the number of aircraft currently in the model.

[9] In effect, a pseudo "B" matrix.

Algorithm

Step 0:

a) Set $a_{sr} = \hat{A}_{sr}$ ($1 \le r \le N$; $1 \le s \le N$). For the Attrition Simulation: $a_{sr} = 0$ ($1 \le r \le N+1$; $s = N+1$), $a_{sr} = 0$ ($r = N+1$; $1 \le s \le N+1$; $s \ne k$), $a_{k,N+1} = 1$, where k is the observed state in time period 1.

b) Set the weighted average composite transition matrix, A, to zeros.

For j = 1:nruns

{ **Step j1:** Initialize the recursive elements:

$\gamma_{0,0}(O_0^r) = \mathbf{0}$ ($1 \le r \le N$), and

$\gamma_{0,0}(J_0^{rs}) = \mathbf{0}$ ($1 \le r \le N$; $1 \le s \le N$)

$q_0 = q_{0j}$

Step j2: For $t = 1,2\ldots T_j$, recursively update the estimators:

$\gamma_{t,t}(J_t^{rs})$, $\gamma_{t,t}(O_t^r)$, and q_t

Step j3: Update the period T_j estimates of A_{sr} via:

$$\tilde{a}_{sr} = \gamma_{T_j}(J_{T_j}^{rs}) \Big/ \gamma_{T_j}(O_{T_j}^r) \qquad 1 \le r \le N;\ 1 \le s \le N$$

Step j4: Add weighted estimates to the composite transition matrix. For each column, *r*, perform the following update:

$$a_r = a_r + \left(\sum_{t=1}^{T_j - 1} Y_{tj}^r\right)\tilde{a}_r \qquad r = 1,2,\ldots N$$

}

Final Step: Compute the final weighted average composite matrix by columns:

$$a_r = a_r \Big/ \left(\sum_{j=1}^{nruns} \sum_{t=1}^{T_j - 1} Y_{tj}^r\right) \qquad r = 1,2,\ldots N$$

Test Plan

Cross Validation Testing Overview

To quantitatively estimate and compare the *relative* performances of the different algorithms, we use a test plan and analysis approach based on the statistical technique of model cross-validation. The general technical approach is as follows.

1. We evaluate the algorithms under a variety of simulation scenarios. A scenario defines the inputs and parameters of the simulation models. The different scenarios are outlined in the following section – "Simulation Scenarios". Each scenario is run 3 times, each time with a different "seed" (starting state) for the random number generators. The seeds were obtained sequentially from a table of random digits (REF, 1984).

2. We use the Attrition Simulation and Mission Simulation models to generate multiple data sets, or sample realizations for each scenario. Each scenario/seed within a given simulation model produces 11 x 10 simulation output data sets (11 different data files, each containing the outputs of 10 simulation runs).

3. These 11 data files are presented as training data to each of the four algorithms in Step 3. That is, each algorithm is used 11 times to produce 11 identified systems. Each system identification utilizes the outputs of 10 simulation runs.

4. We next use each fitted model (identified system) to predict outputs in each data file *not* used to fit it. (This is a variation of the technique of model *cross-validation.)* For probabilistic fitted models (HMMAttrition, HMMMission, EntropyMission, and CompartmentalMission), the *predictions* are the frequency distributions of the outputs over 100 runs of the identified model. Otherwise the predictions are the outputs of a single run of the model.

5. The *actual* values used for comparisons are the frequency distribution of outputs from the simulation runs in the other 10 data files. Generally speaking, since 11 data files were created in Step 2, the number of models fit to them is 4 x 11, and the number of predictions made is 4 x 11 x 10 (for each seed of each scenario of each of the two simulation models).

The general approach above is modified somewhat in the case of the Canonical State Space algorithm. Recall that this approach requires a relatively rich set of inputs and outputs to operate upon. However, none of the other algorithms require inputs. (The Mission Simulation is defined as always having arriving aircraft/inputs, but the arrival pattern is not used directly by algorithms other than the CanonicalMission algorithm. In that case the algorithm uses an "enhanced" arrival set with extra aircraft.) This presents a

problem when comparing the absolute differences in means generated by each algorithm, the scale of the outputs is different because of the differences in inputs (weapon system reinforcements or extra aircraft) that arrive during the course of the simulation. We resolve this by using two simulation data sets for the CanonicalAttrition and CanonicalMission models. One set is used for training only, while the second data set is used for testing the identified model. The latter is the same data set used to test the other algorithms.

Testing of Model State Matching

In the Attrition Simulation, the outputs evaluated are the Red and Blue force strengths over time. The system identification algorithm typically operate on a state vector of force strengths by weapon system type (except for HMMAttrition), however, the outputs of concern are still the aggregate force strength on each side. The rationale is that the HMMAttrition algorithm can only operate on a state framework defined by the aggregate Red/Blue forces strengths. To maintain comparability between all 4 algorithms we must use this same metric in each case.

In the Mission Simulation, the outputs evaluated are the aircraft populations in each state of the model over time.

The key metric is the absolute differences in mean output values, over time, and also averaged over all time periods. In deterministic models, the mean value will simply be the single output value. A secondary metric is the average differences in standard deviation, over time and also averaged over all time periods. (This metric applies mainly to probabilistic models). The rationale for choosing these metrics is to provide a consistent and relatively simple means of comparing the distributions of the model output to the distributions of the simulation outputs.

Testing of Model Based Decisions

We would also like to see if the identified models would lead a decision-maker (person or higher order software module) to the same conclusions as would the underlying simulation model.

In the Attrition Simulation the key outcomes tested are:

1. Which side is ahead at combat termination (who wins)?
2. How long before combat termination is achieved?

In the Mission Simulation the key outcomes tested are:

3. How many missions are completed?
4. How many aircraft are destroyed?

We also use cross validation methods for these tests. For (1) the metric will be the fraction of times that Blue wins in the 11 models compared to the fraction of times that Blue wins in each of the 10 comparison data files. For (2) the metric is the absolute differences in termination time, both averages and standard deviations. For (3) and (4) we also measure the absolute differences between models and simulations, both averages and standard deviations.

Simulation Scenarios

Both simulations provide a large number of parameters that can be adjusted to describe a desired scenario. Rather than trying every combination of every parameter against each other, we vary parameters individually against a baseline. This seems to be a reasonable compromise between thoroughness and practicality. For example, using an all combinations approach with 6 parameters, each having 3 possible values, we would have to run $3^6 = 729$ different simulation scenarios. Under our approach the example would result in 6 x 3 – 6 = 12 simulation scenarios. (The minus 6 is due to the baseline simulation using 1 value from the domain of each parameter.) We actually wind up with 8 scenarios for the Attrition Simulation and 9 scenarios for the Mission Simulation as described below. Each scenario is run 3 times with 3 different random number seeds. Baseline values are shown in bold face font.

Attrition Simulation

Kill Probabilities

Scenarios examined include the following probability distributions:

1. uniformly distributed between .01 and .05 (Low)
2. **uniformly distributed between .01 and .10 (Mixed)**
3. uniformly distributed between .05 and .10 (High)

There are 9 kill probability values in each model.

Inter-Event (firing) Time Distributions

The mean inter-firing time for each weapon system type is uniformly distributed between 10 and 20 time units. The times chosen are somewhat arbitrary. The objective is to obtain a mixture of lethality factors (see below) through randomization, but to keep simulation results within the same order of magnitude. Distributions include:

1. **LogNormal (std dev. = mean/2)**
2. Negative Exponential

There are 9 firing time distributions in the model.

Starting Force Strengths

This refers to the aggregate Red/Blue force strengths at the beginning of the simulation. We vary starting force levels because the variability of the force strengths for small units is much greater than for large units. With a small unit, a few probabilities simultaneously going in the "wrong" direction can spell the difference between victory and defeat. Accordingly, relative changes in overall force strengths may vary widely from one period to the next. The levels are:

1. **Low (10 weapons per type, 3 types per side)**
2. High (50 weapons per type, 3 types per side)

Note that because of the differences in the firing times and kill probabilities it is usually the case that to achieve equal starting forces the weapon system vectors on either side must be different. An initialization routine in the model takes a starting guess for weapon system quantities (as given above) and then perturbs them to achieve equal initial force strengths. This also affects the precise weapon quantities that achieve "low" and "high" force strengths.

Termination Conditions

1. **Absolute Decision Rule** – Combat termination occurs when the force strength (of either side) reaches a given threshold value (1/2 the starting strength).
2. Proportional Decision Rule – Combat terminates when the force ratio reaches a specified threshold value (two to one).
3. AOP Rule – Combat terminates when the force strength crosses either the absolute or proportional threshold.
4. AAP Rule – Combat terminates when the force strength curve crosses both the absolute and proportional threshold .

The net result is 8 different scenarios for the Attrition Simulation.

Mission Simulation

Probability of Aircraft Damaged/Destroyed

Each aircraft mission exposes the aircraft to potential damage/destruction. The per mission probabilities are:

1. .00 (None)
2. **.10 (Med)**
3. .25 (High)

Target Matching Time/Number of FAC

Target matching time, the number of forward air controllers (FAC), and the arrival patterns are arranged so that arrivals do not overwhelm the FAC. A total of 24 aircraft (a squadron) will arrive during the simulation. Target matching time is described by uniform or normal probability distributions as follows (parameters in minutes):

Environment	FAC time distn.	# FAC	Arrival Pattern
heavy jamming	Normal(10,2)	1	2/10 minutes
moderate jamming	Normal(5,1)	1	2/5 minutes
heavy jamming	Normal(10,2)	2	2/5 minutes
moderate jamming	Normal(5,1)	2	2/2.5 minutes
no jamming (ATHS[10])	Uniform(.5,1.5)	1	2/minute

Discretization of States

To fit the Mission Simulation within our model framework we divided up the queue sizes and current fuel loads into discrete states. We can vary the level of discretization to measure the effects on model identification. The following combinations are examined:

Fuel States	Queue States
5	4
4	**3**
3	2

The net result is 9 different simulation scenarios in the Mission Simulation.

[10] Automatic Target Handoff System

Detailed Results

The analysis of results first examines the relative performance of the four algorithms in identifying the simulation models. The analysis focuses on the cross-validation statistics for both state matching and model-based decisions. Using the principle of cross-validation discussed in the Test Plan, we compare the behavior of 11 identified models to the average behavior of each of the 10 sets of 10 simulations that were *not* used to identify the model.

Mission Simulation

For all but the Canonical State Space algorithm, whose behavior is deterministic, the behavior of the models is determined by averaging across 100 stochastic runs.

State Matching

The first test reports the average absolute difference between model and simulation state values (quantities of aircraft) over all time periods of the simulation. Recall that in the Mission Simulation the state vector contains the quanties of aircraft in the following states: Returning (out of fuel), Damaged/Destroyed, Target Matching, and Combat. Target Matching is broken down further into "Queue States" x "Fuel States" number of substates, while Combat is broken down into "Fuel States" number of substates. For most scenarios, the result was a total of 18 states. The results were then averaged across all nine scenarios to produce the results displayed in Figure 13 below.

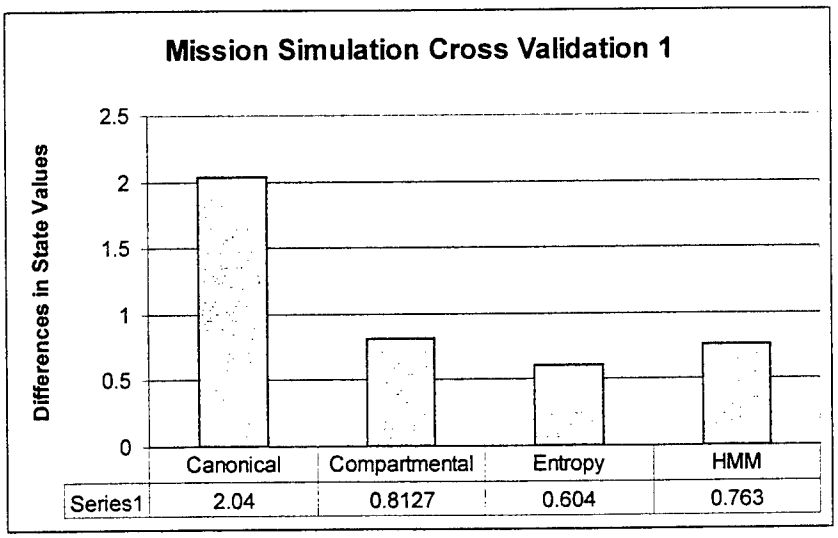

Figure 13. Average Absolute Differences in State Values - Mission Simulation

We see that the Maximum Entropy ("Entropy") technique is superior, with the Compartmental Model ("Compartmental") and HMM techniques contending for 2nd and 3rd place. The Canonical State Space ("Canonical") technique is not a contender in this test. If we break the analysis down by scenario (not shown): the Compartmental Model technique was best for all three random seeds of one scenario, the HMM technique was best for all three random seeds of another scenario, while the Maximum Entropy technique was best for all other scenarios (21).

The Mission Simulation models tended to have very little variability in state values across different simulation runs for a given scenario. Thus, the Canonical State Space technique, being deterministic (with zero variability), was closest in terms of differences in standard deviations, with an average difference of .250 . The Compartmental Model technique was second with .359, the Maximum Entropy technique had .432, while the HMM technique had .452.

Average Behavior

To develop additional insight into model behavior we can compare graphs of aggregated state values. In Figures 14-17 below we present illustrations of average model behavior versus average simulation behavior over time for one baseline scenario of the simulation. The figures display averages of model/simulation outputs over all time periods, therefore they dampen variability. Nor are they based on cross validation. However, they provide a quick, visual means of assessing relative algorithm performance. Keep in mind that the figures show *typical* behavior. Behaviors vary slightly with different random number seeds

The baseline scenario for the Mission Simulation is where:

- mission damage probability = .10
- 2 FAC's, FAC time ~N(10,2), arrivals every 5 minutes
- Fuel States = 4, Queue States = 3

Figure 14 shows that the Canonical State Space technique is clearly not appropriate for the Mission Simulation. It performed very poorly in this scenario (and in many others - although sometimes better than shown in the figure). In Figures 15 and 16 we see that both the Compartmental Model and HMM techniques seemed to do a fair job of state matching this scenario of the Mission Simulation. In Figure 17 we can see that the Maximum Entropy technique appears to have performed very well on this scenario of the Mission Simulation.

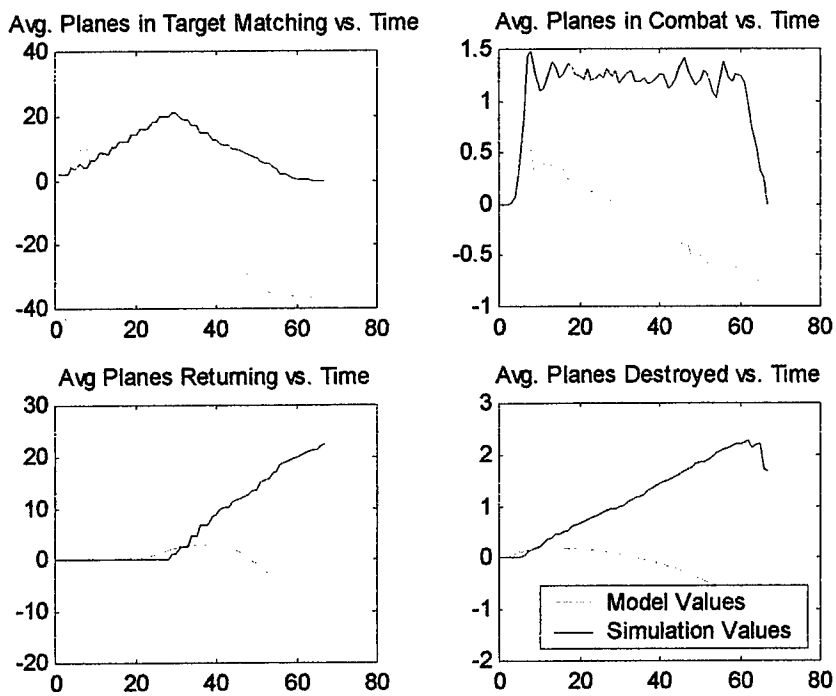

Figure 14. Canonical State Space- Baseline Scenario

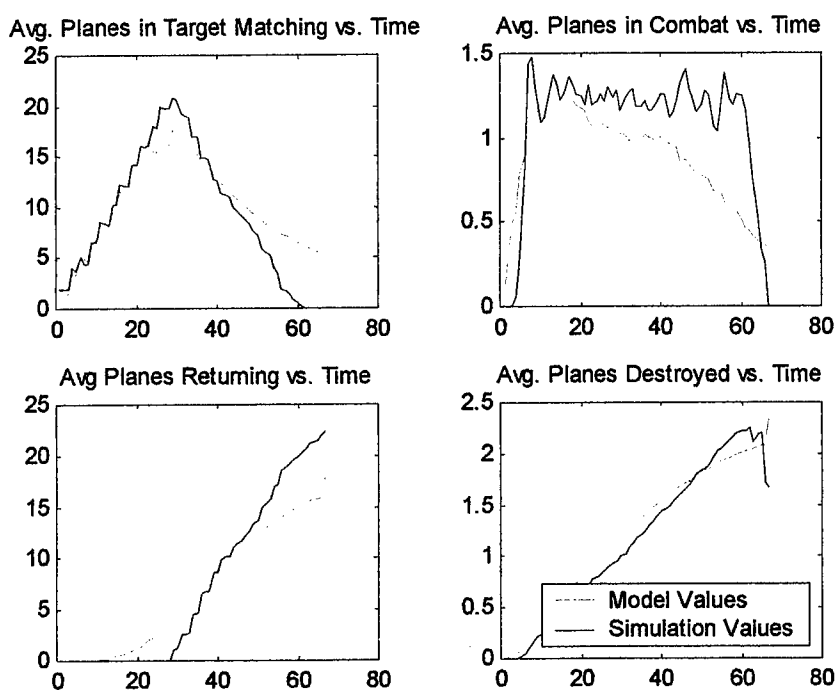

Figure 15. Compartmental Model – Baseline Scenario

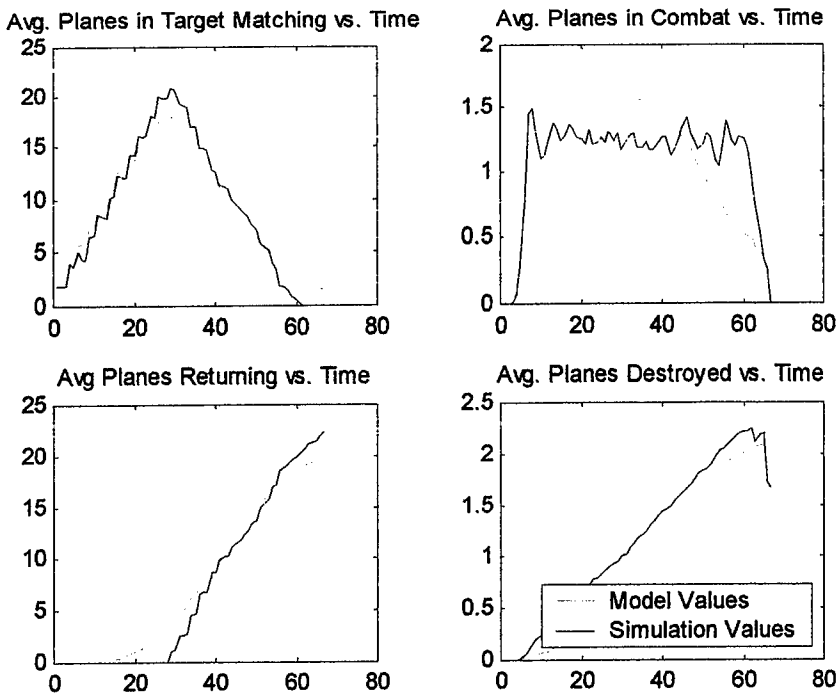

Figure 16. Maximum Entropy – Baseline Scenario

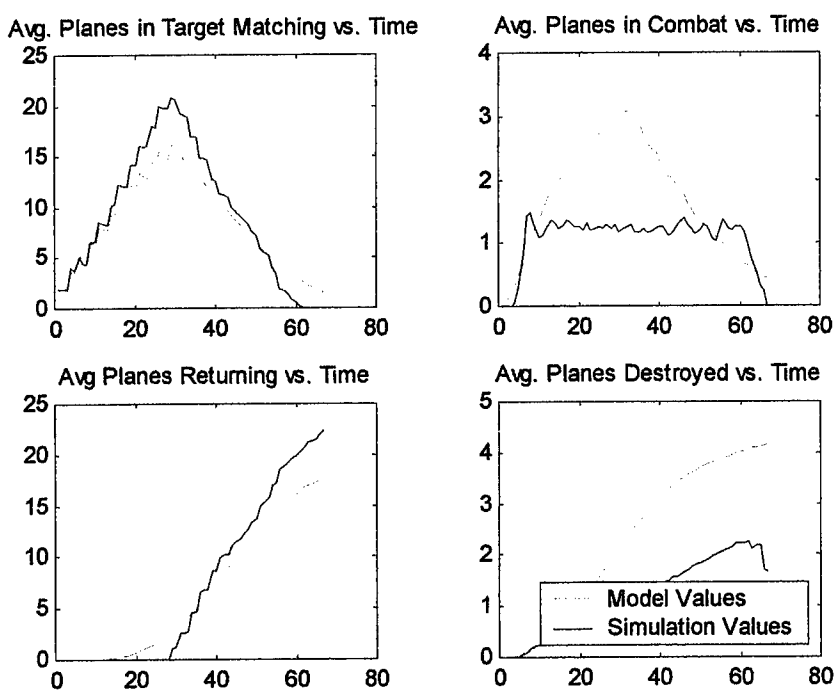

Figure 17. HMM – Baseline Scenario

Model Based Decisions

We next examine the ability of the identified models to provide key decision making outputs similar to those produced by the underlying simulations. For the Mission Simulation, the outputs of concern are the number of missions completed and the number of aircraft damaged/destroyed. For this test, we look at the proportional differences in the values reported, and average these across all scenarios. Dividing the average absolute difference in values by the mean simulation values creates the proportion. The results are shown in Figures 18 and 19 below. We see that the Maximum Entropy technique is best at predicting the number of missions completed. Its estimates were off the true value by an average of 9.1%. In individual scenarios (not shown), the difference was almost always about 9% and the model always under-predicts the average simulation value. The Maximum Entropy technique provided the best predictor in each and every scenario. Returning to Figure 18, we see that the Compartmental Model and especially the HMM techniques were significantly worse. The Canonical State Space technique was 100% off, the reason being that the model has no way of predicting missions completed. The number of missions completed is a count of the aircraft transitions out of the "Combat" states, but the Canonical State Space technique can not track discrete movements between states, it is only capable of predicting total state values. Interestingly, the Maximum Entropy and Compartmental Model techniques tended to underestimate the number of missions completed, while the HMM technique tended to overestimate the number of missions completed.

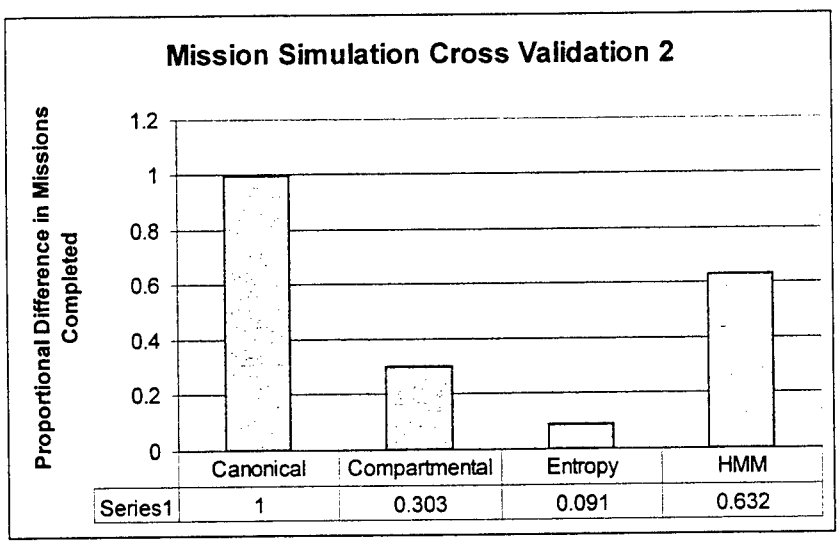

Figure 18. Average Proportional Differences in Missions Completed - Mission Simulation

In Figure 19 we see that the Maximum Entropy technique was also best in predicting the number of aircraft damaged/destroyed. It was off in its predictions by an average of 18.6%. It was the best predictor in each and every scenario. Again, the Compartmental

Model and HMM techniques were significantly worse, while the Canonical State Space technique is "off the charts". The Maximum Entropy technique provided the best predictor in each and every scenario. The Maximum Entropy and Compartmental Model techniques had no clear pattern of under or over estimation of this value, while the HMM technique tended to overestimate.

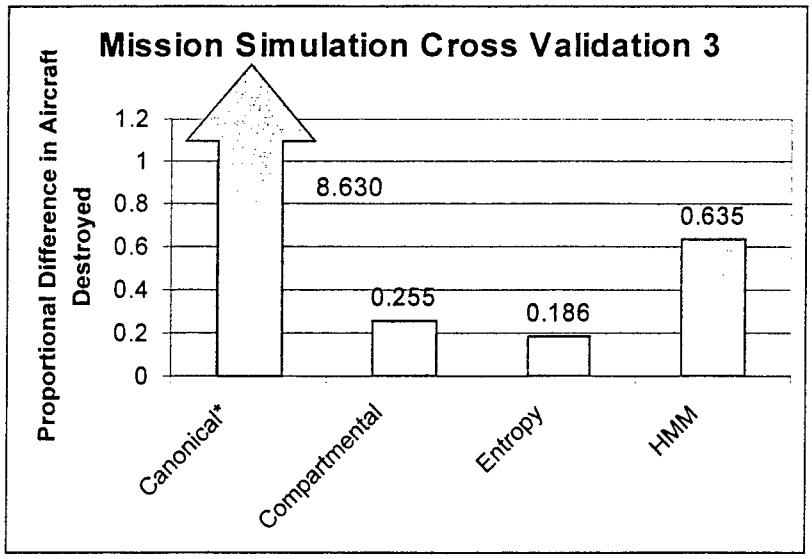

Figure 19. Average Proportional Differences in Aircraft Damaged/Destroyed - Mission Simulation

Attrition Simulation

For the HMM algorithm, the behavior of the models is determined by averaging across 100 stochastic runs. All other algorithms produce a single deterministic model.

State Matching

The first test reports the average absolute difference between model and simulation state values over all time periods of the simulation. Recall that in the Attrition Simulation the state values represent the Red/Blue force strengths. The values displayed in Figure 20 below are averages over all time periods of all eight scenarios. We see that the Compartmental Model and HMM techniques approximately tie for first place, while the Canonical State Space and Maximum Entropy techniques are "off the charts". However, for the latter two techniques it should be noted that the majority of the contribution to the average was from two or three scenarios, other scenarios had average values much closer to the former two techniques. Additional explanation is provided in the next section.

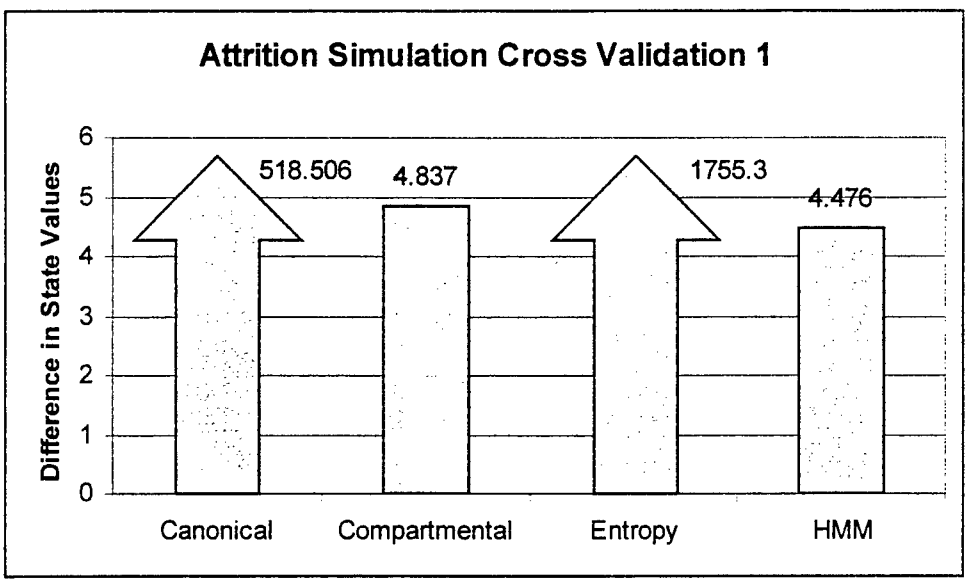

Figure 20. Average Absolute Differences in State Values - Attrition Simulation

In individual scenarios (not shown), the Compartmental Model technique was best 8 out of 24 times, the Canonical State Space technique was best 2 times, and the HMM technique was best 14 times.

The differences in the standard deviation of state values was smallest for the HMM technique in every scenario. Overall, the average value of the difference in standard deviation of differences was 2.202 for the HMM technique, and 3.868 for all other techniques (being deterministic, they each had standard deviations of 0 within a given model).

Average Behavior

As was the case with the Mission Simulation, we can develop additional insight into model behavior by comparing graphs of aggregated state values, shown in Figures 21-24 below.

The baseline scenario for the Attrition Simulation is the situation where:

- kill probabilities are uniformly distributed between .01 and .10
- inter-firing time distributions are LogNormal
- starting force strengths are "Low"
- Combat termination occurs when the force strength (of either side) reaches a given threshold value (1/2 of the starting strength).

In Figure 21 below, we see that the Canonical State Space technique appears to have done a fair job of state matching this scenario of the Attrition Simulation. Note that there is no graph of standard deviations for the Red and Blue models. The standard deviations of these are actually zero, the reason being that the identified model is deterministic.

In Figure 22, we see that the Compartmental Model technique appears to have some predictive value for the Attrition Simulation, while Figure 23 shows that the Maximum Entropy technique performed poor to fair overall. The main problem with the Maximum Entropy technique was its tendency to diverge over time from the average simulation behavior. In some cases the divergence was much more pronounced than shown below. This is the reason for the "off the charts" difference in state values in Figure 20. A similar phenomenon would sometimes occur with the Canonical State Space technique. In Figure 24 we see that the HMM technique appears to have performed very well on this scenario of the Attrition Simulation.

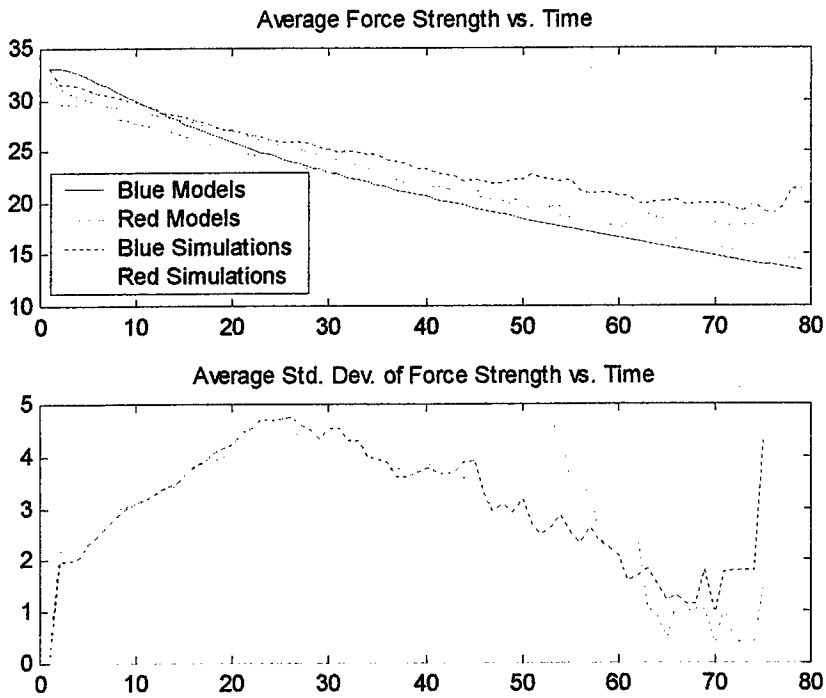

Figure 21. Canonical State Space – Baseline Scenario

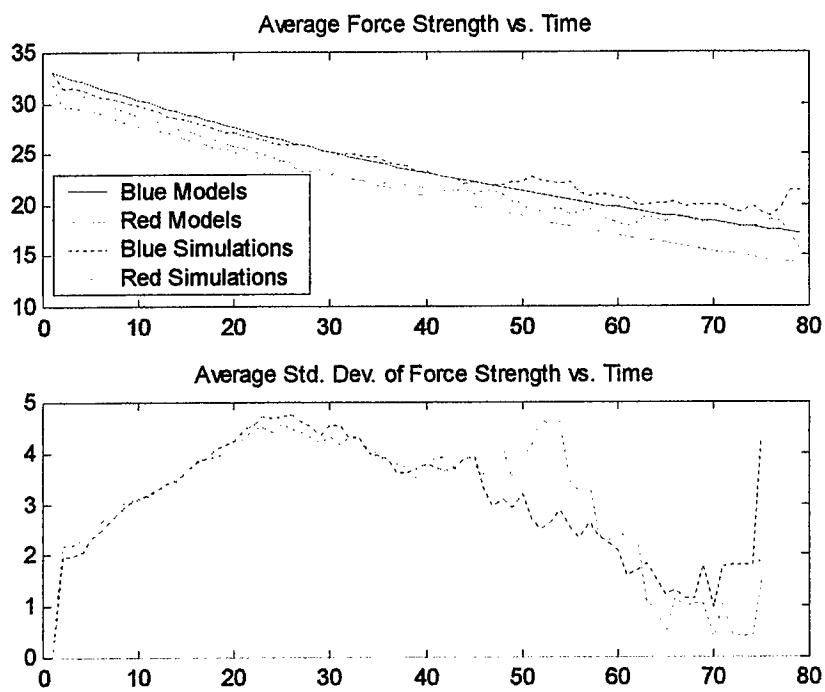

Figure 22. Compartmental Model –Baseline Scenario

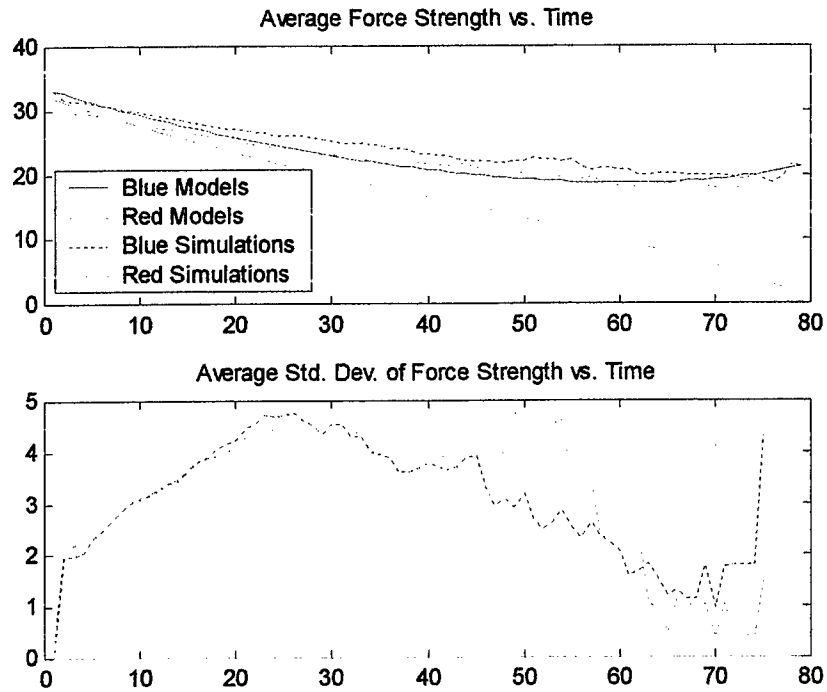

Figure 23. Maximum Entropy – Baseline Scenario

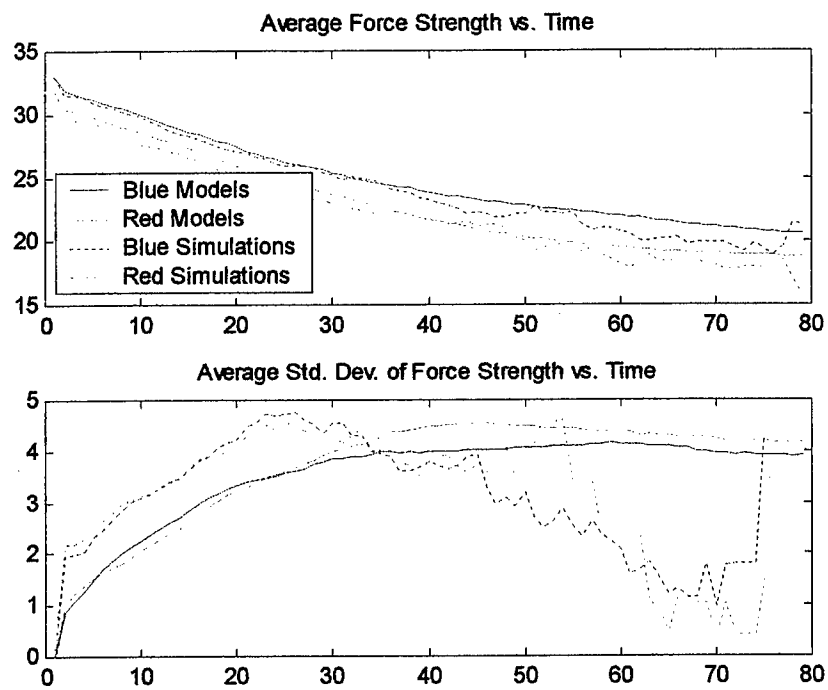

Figure 24. HMM – Baseline Scenario

Model Based Decisions

In these tests we compare the models in their ability to predict two key output values, the average time to termination (battle end time), and the fraction of times that "Blue" wins the battle. In this test, we convert the absolute difference in termination times to a proportion of the average simulation values. For this output, we see in Figure 25 below that the Maximum Entropy and HMM techniques are approximately tied for first place, while the Canonical State Space and Compartmental Model techniques are significantly worse. The Maximum Entropy technique is off by an average of 22.9%. In individual scenarios (not shown), the Maximum Entropy technique was best 14 out of 24 times, and the HMM technique was best the other 10. Interestingly, the HMM technique tended to overestimate the termination time, while the Maximum Entropy technique tended to underestimate the termination time.

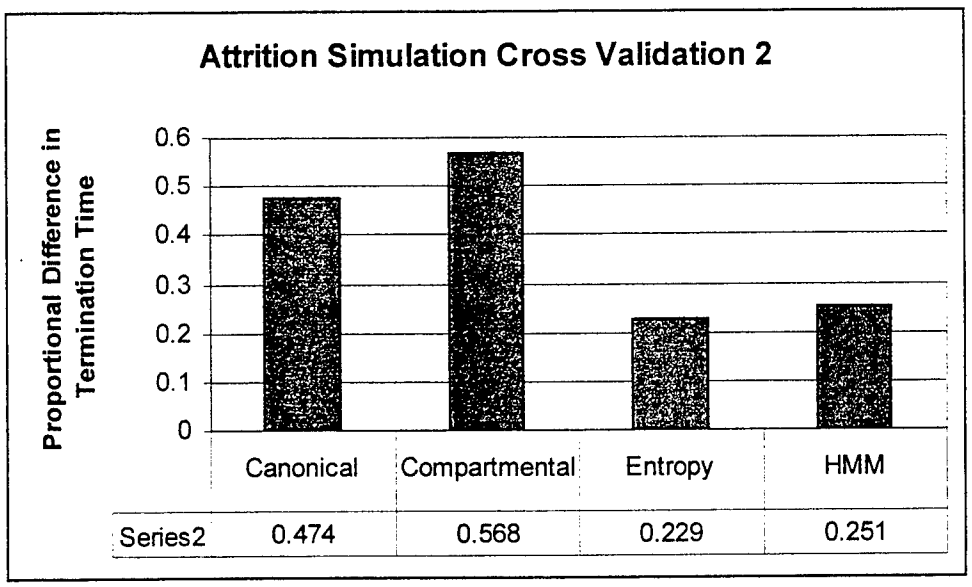

Figure 25. Average Proportional Difference in Time to Termination - Attrition Simulation

The final test examines how well the models perform in matching the simulations in terms of the fraction of wins by "Blue". In Figure 26 below we again see the HMM technique doing well, the Maximum Entropy and Compartmental Models techniques doing significantly worse, and the Canonical State Space technique doing quite poorly (it is almost a counter-predictor!). In individual scenarios (not shown), the Maximum Entropy technique was best 6 out of 24 times, and the HMM technique was best the other 18. Here, there was no clear pattern of over or under estimation.

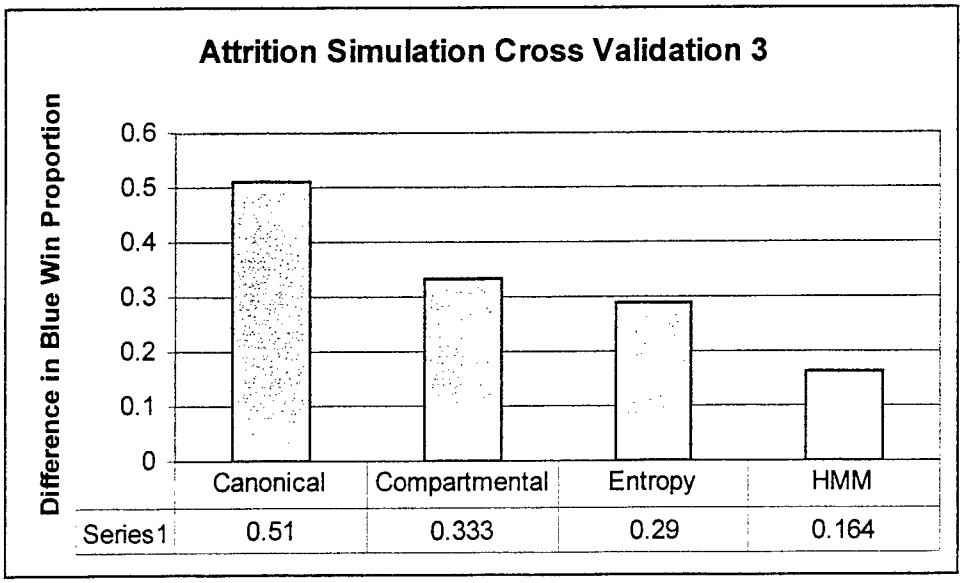

Figure 26. Average Difference in Blue Win Fraction

Effectiveness of the Best Techniques

The preceding discussion described the results of a battery of cross-validation tests that focused on determining the best system identification technique for each simulation model. In this section, we use a method for determining the absolute, rather than relative, effectiveness of these best techniques. A key factor in this analysis is the measurement of random variations in the output of the simulation models. Because of this variability, no technique can predict the precise simulation outputs, however, any forecasts should fall within a range determined by the mean output value(s), the expected variation about this mean, and a given level of statistical confidence. We will present the results of this analysis on our "best" techniques.

Method

The method is based on statistical sampling as commonly applied to process control (see, for example, Chase and Aquilano, 1995). A simulation scenario provides the "process", and batches of simulation runs provide a set of samples for determining our limits of variation ("control limits"). We then extract "samples" from the output of our identified models and compare them to the control limits derived from the simulation data. Sample model data falling outside the control limits provides evidence that the simulations were not correctly identified. The tests will be applied to our "decision-making outputs". For the Attrition Simulation these are the completion/termination time and the "Blue" win fraction. For the Mission Simulation these are the number of missions completed and the number of aircraft destroyed.

For all but the Blue win fraction we proceed as follows. Let:

n = sample size (number of simulation runs in a sample)
X_i = output value from simulation run *i*
m = total number of samples
R_j = range of output values in sample *j*

For each sample, the mean value is given by:

$$\overline{X} = \sum_{i=1}^{n} X_i \Big/ n$$

The average of the mean of samples is:

$$\hat{X} = \sum_{j=1}^{m} \overline{X}_j \Big/ m$$

The average range of the samples is:

$$\overline{R} = \sum_{j=1}^{m} R_j \Big/ m$$

We can then compute upper and lower control limits on both X and R as:

$$UCL_{\overline{X}} = \hat{X} + A_2\overline{R}$$
$$LCL_{\overline{X}} = \hat{X} - A_2\overline{R}$$
$$UCL_R = D_4\overline{R}$$
$$LCL_R = D3\overline{R}$$

where A_2, D_3, and D_4 can be found in statistical tables as a function of confidence level and sample size.

Once these control limits have been calculated, we can compare the $\overline{X}$ and R sample values (model outputs) to them to determine model effectiveness. Values falling outside of the limits suggest that the model has not correctly identified the underlying simulation.

To calculate control limits for the Blue win fraction let:

$\bar{p}$ = the overall blue win fraction from all runs in all samples

$$s_p = \sqrt{\frac{\bar{p}(1-\bar{p})}{n}} = \text{standard deviation of the fraction}$$

Then:

$$UCL_p = \bar{p} + zs_p$$
$$LCL_p = \bar{p} - zs_p$$

where z is the number of standard deviations for a specific confidence, typically 3.

Effectiveness Results

The effectiveness analysis was applied to the three baseline scenarios (1 for each of three seeds) of each simulation. Based on fairly obvious results from the previous section, the HMM technique was tested against the Attrition Simulation, while the Maximum Entropy technique was tested against the Mission Simulation. To establish control limits, the simulations were first run in 25 batches with a sample size of 10 runs per batch. We used "3 sigma" control limits, which is equivalent to saying that the process should provide values within these limits 99.7% of the time.

Two system identification types were performed using the three baseline simulation scenarios, one used 10 simulation runs, while a second used 20 simulation runs. As with the cross validation, 11 separate identifications were performed (for each type). Once the models were identified, a sample batch of 10 runs was produced by each of the 11 identified models. The averages and ranges from these sample batches were compared to the control limits. Values outside of the control limits provide evidence that the underlying process has not been correctly identified.

Attrition Simulation

We first examine the completion (termination) times of the Attrition Simulation. Figure 27 below shows the batch averages versus control limits for the model effectiveness test runs. Some values fall outside of the control limits, less in the 20-run identification than in the 10-run identification. The former has 4 out of 33 values outside of the limits while the latter has 10 of 33 values outside of the limits. The simulation average values for the 3 seeds were (88.3, 119.6, 103.9) while the corresponding model averages were (97.9, 140.2, 136.7) for the 10-run identifications, and (97.4, 128.7, 127.7) for the 20-run identifications.

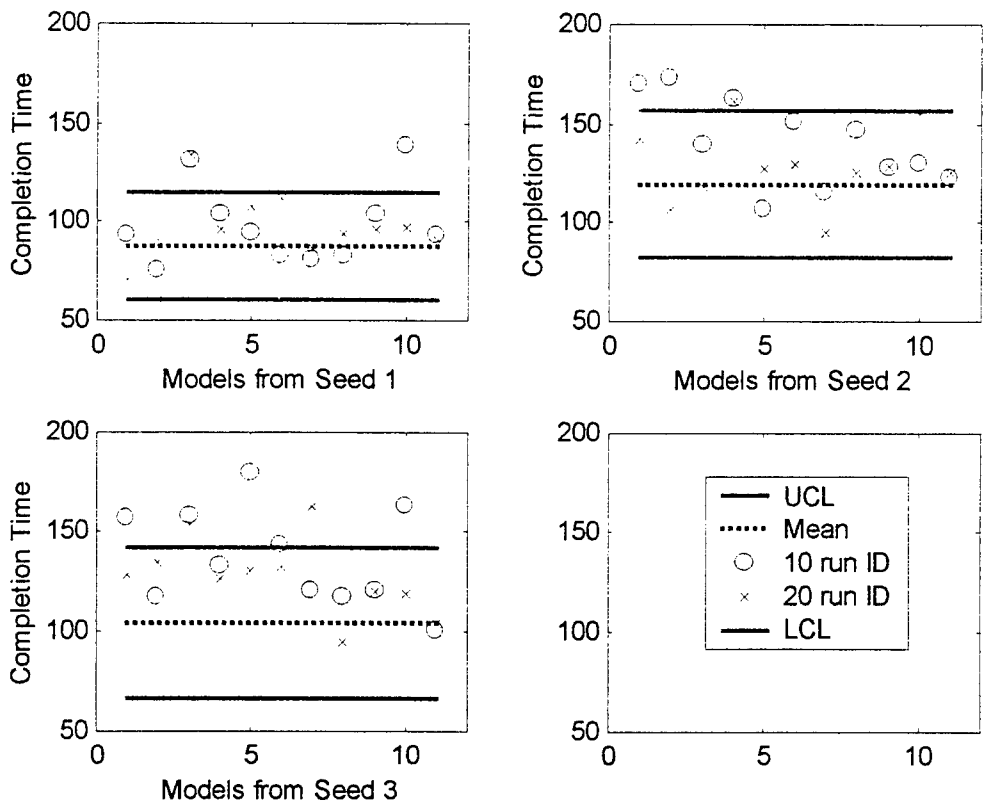

Figure 27. Control Charts for Average Completion Time, Attrition Simulation, Baseline Scenario

We also examine control charts for the range of completion times within a sample. These are illustrated in Figure 28 below. These control charts measure the variability of the process. We see that the 20-run identification had no values outside of the control limits, while the 10-run identification had 4 of 33 values outside of the control limits.

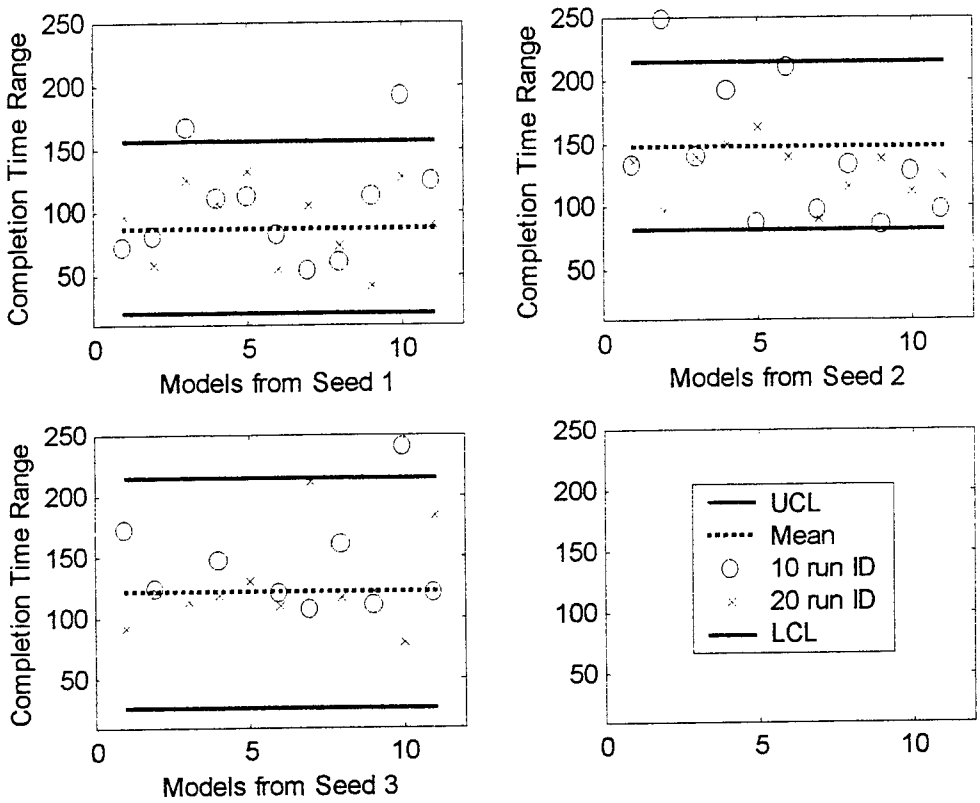

Figure 28. Control Charts for Range of Completion Times, Attrition Simulation, Baseline Scenario

Figure 29 below presents control charts for the fraction of times that "Blue" wins in a given sample. The 10-run identifications produced 2 values outside of the control limits, while the 20-run identification produced 1. The simulation average values for the 3 seeds were (.68, .5520, .776) while the corresponding model averages were (.672, .409, .70) for the 10-run identifications, and (.609, .473, .618) for the 20-run identifications.

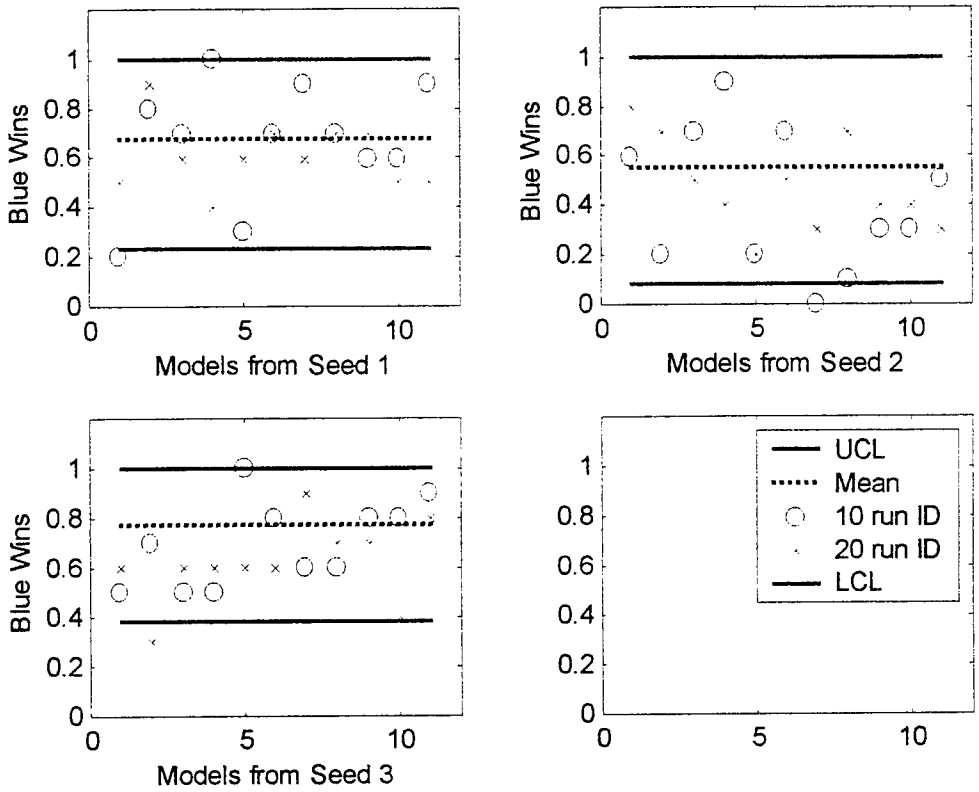

Figure 29. Control Charts for Blue Win Fraction, Attrition Simulation, Baseline Scenario

Mission Simulation

We next examine the number of missions completed in the Mission Simulation. Figure 30 below shows the batch averages versus control limits for the model effectiveness test runs. In this case, more values fall outside (below) the control limits than inside, in both the 20-run identifications and 10-run identifications. The simulation average values for the 3 seeds were (20.8, 20.9, 20.6) while the corresponding model averages were (18.2, 18.8, 18.4) for the 10-run identifications, and (18.2, 18.6, 18.5) for the 20-run identifications.

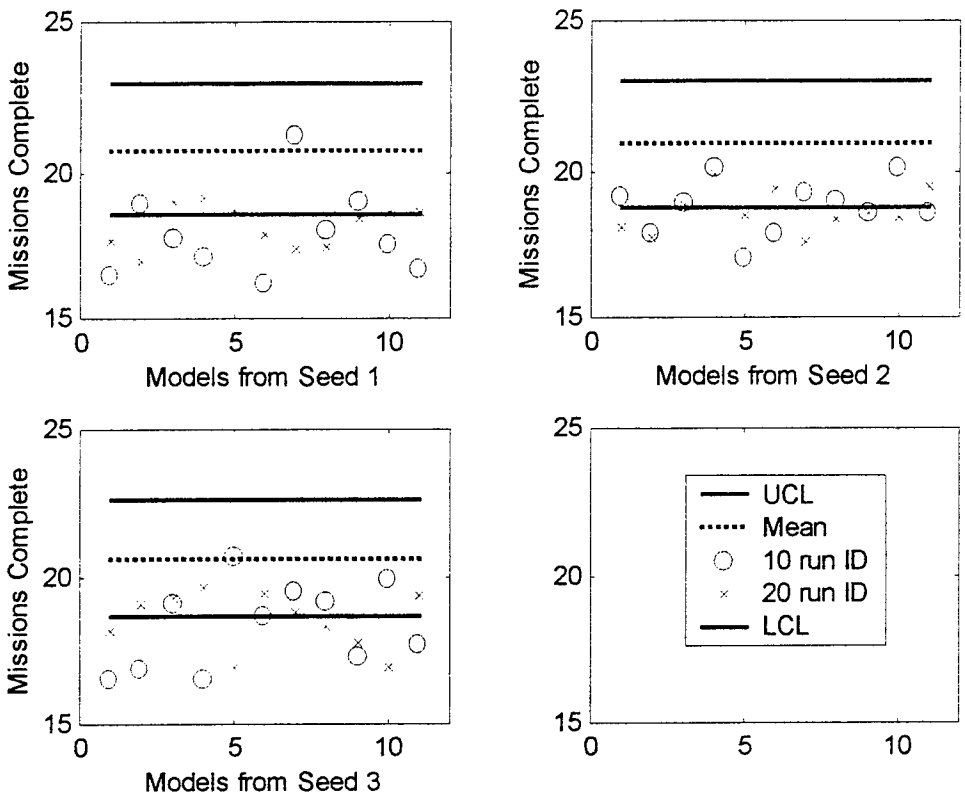

Figure 30. Control Charts for Average Missions Completed, Mission Simulation, Baseline Scenario

We also examine control charts for the range of missions complete within a sample. These are illustrated in Figure 31 below. These control charts measure the variability of the process. We see that both the 20-run identification and the 10-run identifications had most values outside (above) the control limits. In fact, the 20-run identification had more values outside than the 10-run identification.

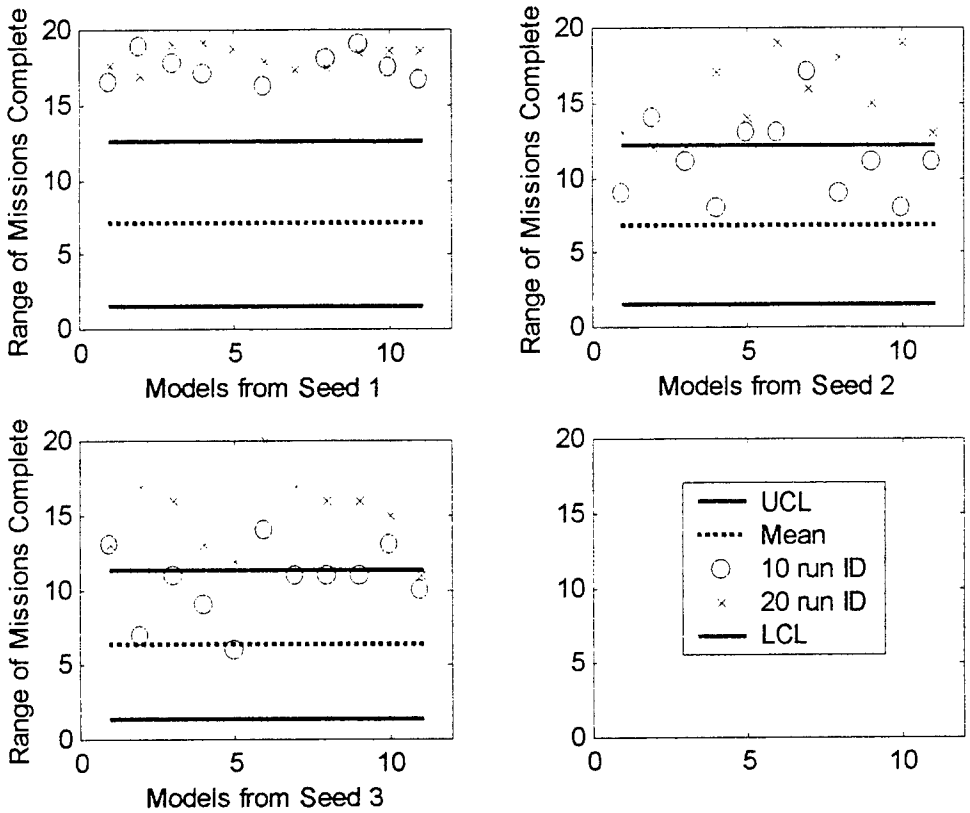

Figure 31. Control Charts for Range of Missions Completed, Mission Simulation, Baseline Scenario

We next examine the number of aircraft destroyed in the Mission Simulation. Figure 32 below shows the batch averages versus control limits for the model effectiveness test runs. In this case, all values but 2 fall inside of the control limits, both of these in 10-run identifications. The simulation average values for the 3 seeds were (2.4, 2.4, 2.4) while the corresponding model averages were (2.2, 1.9, 2.4) for the 10-run identifications, and (2.2, 2.1, 2.3) for the 20-run identifications.

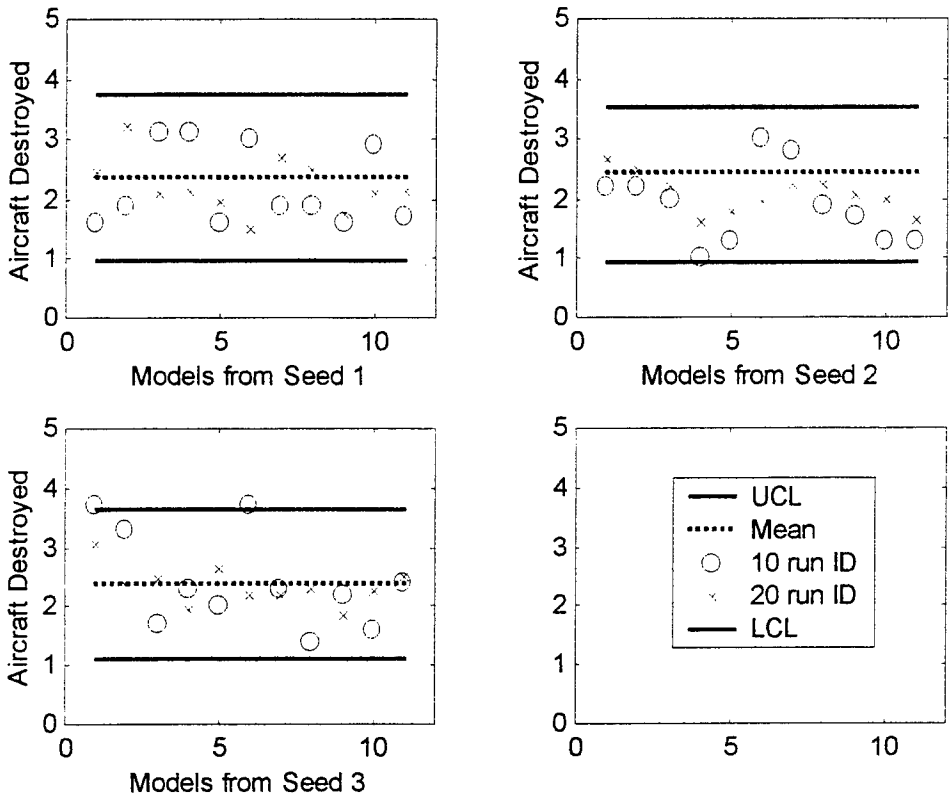

Figure 32. Control Charts for Average Aircraft Destroyed, Mission Simulation, Baseline Scenario

Last, we examine control charts for the range of aircraft destroyed within a sample. These are illustrated in Figure 33 below. We see that both the 20-run identification and the 10-run identifications had all values inside the control limits.

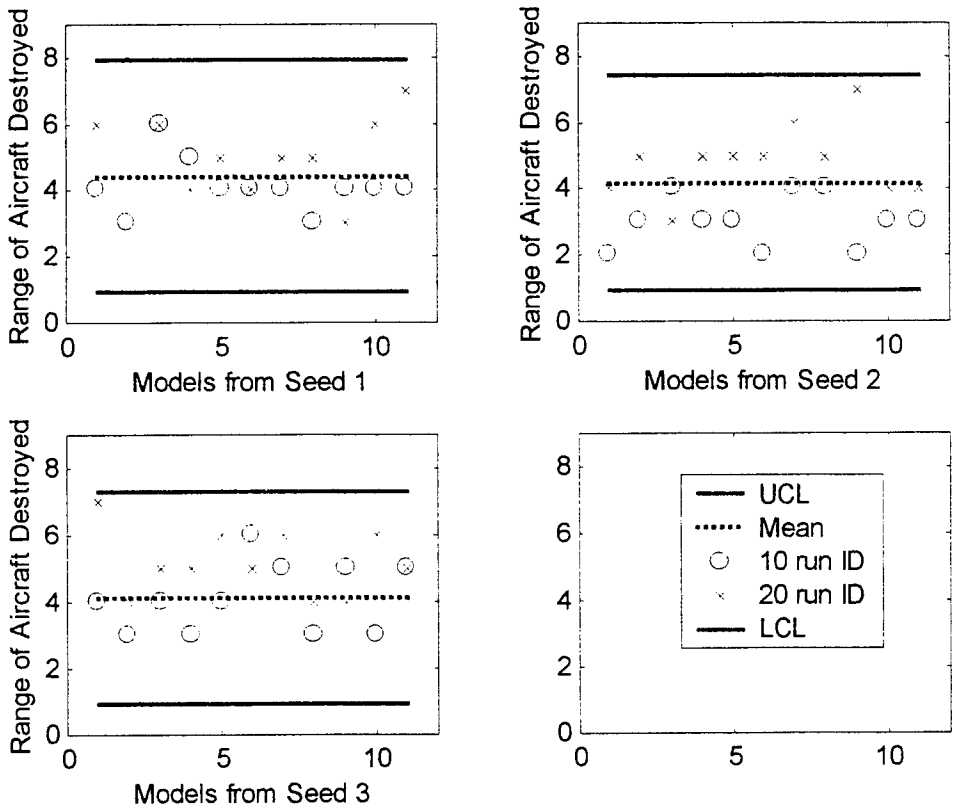

Figure 33. Control Charts for Range of Aircraft Destroyed, Mission Simulation, Baseline Scenario

Algorithm Performance Considerations

This section compares the algorithms in terms of speed, memory usage, and robustness. The implementation platform was a Windows NT Workstation with a 433 MHz processor and 256 MB of RAM. The algorithms were implemented in MATLAB, Version 5.3, with the exception of two key subroutines (described below), that were implemented as C dynamic link libraries. The Maximum Entropy algorithm requires an add-in package to MATLAB, known as the Optimization Toolbox (Coleman, et. al.; 1999). Note that MATLAB is an interpreted language and was used in that mode during this research. MATLAB software can also be compiled into C and C++ code via MATLAB compiler software and libraries, resulting in programs that run much faster than the original MATLAB code. Run times provided below result from a single identification, whereas each cross-validation test run required 110 identifications (there were 204 test runs – one for each scenario/seed combination). The RAM requirements discussed below are in terms of what is needed *in addition to* the RAM taken up by the operating system and MATLAB software (about 60 MB).

Canonical State Space

This algorithm operates relatively fast (less than a second), and does not require a lot of memory (perhaps 1 – 3 MB depending on the simulation and the scenario). Its operation was simplified by the fact that the states we attempt to identify are fully observable, thus the structural indices need not be calculated. An occasional numerical problem occurs where the estimation matrix, "S", is singular. As a practical work-around, the matrix was perturbed to non-singular form without significant degradation of results.

Compartmental

This algorithm also operates relatively fast (less than a second), and does not require a lot of memory (perhaps 1-5 MB depending on the simulation and the scenario). However, a robust implementation requires attention to potential numerical difficulties. First, the technique involves the computation of terms of the form: e^{At}, where A is a transition rate matrix. These can be computed via eigenvalue decomposition methods, however, the potential exists for the decomposition to not exist or to involve complex numbers (we experienced both of these situations in early trials). The built in MATLAB function for this type of exponential calculation handles these situations automatically. Similarly, the MATLAB matrix convolution function performs calculations needed to find derivatives, thus sidestepping the need to deal with potentially singular decomposition matrices. These MATLAB functions were substituted for our earlier, more fundamental code. A final difficulty can result when the variance-covariance matrix, "V", is singular. As a practical work-around, the matrix was perturbed to non-singular form without significant degradation of results. Coding the variance-covariance matrix building subroutine in C averted a computational bottleneck.

Maximum Entropy

This algorithm operates relatively slow, and sometimes requires large amounts of memory; even with "sparse" versions of the constraint matrices and the "large-scale" mode of the optimization function. In the Attrition Simulation, about 40 MB of additional RAM was needed and the time required for a single identification ranged from about 10 seconds to about 3 minutes. In the Mission Simulation, up to 200 MB of additional RAM was needed and the identification time ranged from about 30 seconds to about 8 minutes. This slowness results partly from the relatively large constraint matrix needed to find the parameter values. In the baseline scenario of the Attrition Simulation, the matrices typically had around 600 rows (constraints) and 1300 columns (variables). In the Mission Simulation we could typically have about 1800 rows and 5500 columns for the baseline scenario. Another reason for slow speed was that in both simulations, the nonlinear programming subroutine would sometimes converge very slowly. As a practical work-around, an iteration limit of 100 was employed without significant degradation of results. In the case of the Attrition Simulation, the algorithm would also sometimes "lock up" while performing the linear programming subroutine that calculates prior estimates. In these instances, what normally required a few seconds might take up to 30 minutes, and produce an infeasible solution (these were eliminated from the parameter averages). This problem has been isolated to scenarios where a few simulations were significantly longer than the average within an identification group, leading to an unusually large constraint matrix (e.g. 3000 x 3000). The behavior of this algorithm is strongly influenced by the choice of optimization software. Particularly with nonlinear programming, there can be large differences in efficiency and solution quality between packages. In this study, we used MATLAB's Optimization Toolbox add-on package. A compiled version would likely run much faster. Systems View has also had some previous experience with a C-based mathematical programming package known as LOQO (Vanderbei, 1999), which seems to converge very quickly on these sorts of problems. However, for this high-level comparison, the acquisition and integration costs of this package were not warranted, but might be for a more in depth study that focused on the Maximum Entropy technique.

HMM

The worst-case performance of this algorithm increases with N^3T and memory requirements increase with N^3, where N is the number of states, and T is the number of time periods. We developed an optimized C code version of the HMM computational subroutine that exploits the sparsity (zero elements) typically found in the state transition matrix. This allows us to handle state transition matrices of up to 400 x 400 elements (N=400) and a T of about 50, without major difficulty. The Mission Simulation had no more than 26 states, so HMM was very fast (milliseconds) and required negligible memory. In the Attrition Simulation we set N=256, representing 16 subdivisions of Red and Blue output values, and T ranged from about 50 to 150, depending upon the scenario. The algorithm operated on the order of 10 to 90 seconds per identification. Memory requirements were small, a marginal increase of 6 MB or so for the HMM subroutine.

Results Summary and Conclusions

Identification Techniques

The results clearly demonstrated that the Hidden Markov Model (HMM) technique was superior at identifying the Attrition Simulation (See Table 2 below). It was slightly better than the next best technique (Compartmental Models) at matching states, second best at predicting completion/termination times (Maximum Entropy was best), and superior in predicting the Blue win fraction. This conclusion is strengthened by the fact that the Compartmental Models technique was worst at predicting completion time and second worst in predicting the Blue win fraction. Similarly, the Maximum Entropy technique was worst in matching states (it was "off the charts"). Only HMM excelled in all tests.

Technique	State	Comp. Time (proportion)	Blue Win Fraction
Canonical State Space	518.506	.474	.51
Compartmental Model	4.837	.568	.333
Maximum Entropy	1755.3	**.229**	.29
Hidden Markov Model (HMM)*	**4.476**	.251	**.164**

Table 2. Attrition Simulation – Average Errors

It is equally clear that the Maximum Entropy technique was superior at identifying the Mission Simulation (See Table 3 below). It was slightly better than the next best technique (HMM) at matching states, superior in predicting the number of missions completed, and slightly better than the next best technique (Compartmental Models) in predicting the number of aircraft destroyed. While the HMM technique did very well in the Attrition Simulation, it was very poor at predicting missions completed and aircraft destroyed in the Mission Simulation.

Technique	State	Missions Completed (proportion)	Aircraft Destroyed (proportion)
Canonical State Space	2.04	1	8.630
Compartmental Model	.8127	.303	.255
Maximum Entropy*	**.604**	**.091**	**.186**
Hidden Markov Model (HMM)	.763	.632	.635

Table 3. Mission Simulation – Average Errors

It was initially surprising that, while the HMM technique did very well with the Attrition Simulation , it did poorly on the Mission Simulation. The same is true in reverse for the Maximum Entropy technique. After all, both produce a stochastic identified model. We believe there are two main reasons for this.

The first is that the Mission Simulation had inputs (aircraft), while the Attrition Simulation did not.[11] The only way that the HMM technique can incorporate entities entering different states of the model is through the prior distribution of the state, q_0. The prior distribution is a model input that we calculated off-line via historical frequencies, conditioned on the number of aircraft already in the model. The Maximum Entropy technique handles inputs directly, via the input transition matrix, B. Calculation of the input transition matrix is performed simultaneously with the calculation of the state transition matrix, allowing for joint optimization. This leads us to a conclusion that simulations with inputs are best identified by techniques that include the input transitions in the identification process.

The second reason is that in the Attrition Simulation, the HMM technique operated upon aggregated force strengths while the Maximum Entropy technique operated upon force strength by weapon system type. The force strengths of individual weapon system types had a lot of variability. In addition, they could start out large, and then go to zero, remaining that way for extended periods until termination conditions were reached. This creates the potential for scaling problems, a frequent difficulty with mathematical programming algorithms. This leads us to a conclusion that the Maximum Entropy technique is not suitable for models where state values have extreme amounts of variability and/or scale differences.

Overall Effectiveness

The effectiveness testing on these two best techniques provided mostly positive results. In the case of the HMM/Attrition algorithm, we saw that both completion times and the winning side were predicted fairly well. Specifically, in the more refined models (20-run identifications) only 4 of 33 average completion time values fell outside the control limits, while none of the sample ranges fell outside of the control limits. Only 1 of 33 values fell outside of the control limits for the blue win fraction.

In the case of the Entropy/Mission algorithm the results were promising, but not as good as with the HMM/Attrition algorithm. The main problem was that in predicting the number of missions completed, the model consistently underestimated the simulations by about 10%. This result was true in both the 10-run and 20-run identifications. In addition, the ranges of values within a sample were consistently too high. On the other hand, the Entropy/Mission algorithm did a good job in predicting the number of aircraft

[11] Except of course when required for the Canonical State Space technique.

destroyed. In the 20-run identifications, no values were outside of the control limits for either the average value or the range of values within a sample.

Performance

The HMM algorithm, as implemented in this research, is relatively fast and space efficient. One caveat is that the run-time performance could degrade in situations where there are more than about 400 states (with current hardware) and/or the state transition matrix is dense.

The Maximum Entropy algorithm, as implemented in this research, is relatively slow and requires relatively large amounts of memory. However, the key driver is the mathematical programming subroutines. Obviously, efficient, compiled C code implementations would be much faster than our MATLAB version. It is not clear how much more space efficient other implementations might be.

Discussion

This research has demonstrated the viability of abstracting stochastic simulation models via state-based system identification techniques. It was significant that system identification techniques resulting in stochastic models were best at identifying stochastic simulations. The results did vary by simulation model, scenario, and identification technique in that no one technique is universally superior for all simulations. Even the "worst" methods would sometimes provide the best estimate in a given scenario. However, the Hidden Markov Model (HMM) technique and Maximum Entropy technique showed the most promise. The results suggest that a hybrid method that combines the results of several identification techniques would likely provide improved estimates. The statistical technique of classification trees is one approach for combining the results of identified models.

Choice of an appropriate state-space analogue to the underlying simulation is important. The number of possible states or observations can not be too large, both for computational performance, and for numerical tractability. The state space must be kept reasonably small since the run time of most identification techniques increases in a polynomial fashion with the number of states. State abstraction/aggregation prior to identification may be critical for success. For example, in the Attrition Simulation we performed two levels of abstraction. First, we converted weapon system counts, lethality, and vulnerability to force strengths. Second, (in the HMM identification) we aggregated the force strengths by weapon system type to a single value per side. In the Mission Simulation, we needed a state space that would capture the different activities of the individual aircraft (targeting, combat, damaged/destroyed, and returning), their current fuel level, as well as the queuing effect at the forward air controller. We used an approach of discretizing the fuel levels (a continuous to discrete abstraction) and aggregating the possible queue sizes. Without these abstractions the number of possible states in either simulation would have been astronomical.

The modeled states must also be chosen so that counts of either state populations or state transitions suffice to provide outputs analogous to those of the underlying simulation. For example, in the stochastic identifications of the Mission Simulation, the number of missions completed was determined by observing the number of entity transitions between "combat" states and states other than "destroyed". The number of aircraft destroyed was a simple count of a state population. In the HMM identifications of the Attrition Simulation, each state corresponded to a pair of Red/Blue force strengths. We could determine termination time by capturing the time period that the identified model entered one of a set of designated terminal states, while the winner was found by comparing the Red/Blue force strength pair at termination to see which was greater. All of this points to the fact that the purpose of the model must drive its structure, scope, and resolution.

References

C.J. Ancker, A Proposed Foundation for a Theory of Combat, in Bracken, Kress, and Rosenthal (eds.) *Warfare Modeling*, published by John Wiley & Sons for the Military Operations Research Society, pp. 165-197, 1995.

C.J. Ancker and A.V. Gafarian, *Modern Combat Models: A Critique of their Foundations, Topics in Operations Research Series*, Operations Research Society of America (now INFORMS), Baltimore, MD, 1992.

L.B. Anderson and F.A. Miercort, Combat: A Computer Program to Investigate Aimed Fire Attrition Equations, Allocation of Fire, and the Calculations of Weapons Scores, IDA Paper P-2248 (DTIC AD-A213660), 1989.

L.B. Anderson and F.A. Miercort, On Weapons Scores and Force Strengths, in Bracken, Kress, and Rosenthal (eds.) *Warfare Modeling*, published by John Wiley & Sons for the Military Operations Research Society, pp. 229-249, 1995.

D.M. Bates and D.G. Watts, *Nonlinear Regression Analysis and its Applications*, Wiley, 1988.

Bracken, Kress, and Rosenthal (eds.) *Warfare Modeling*, published by John Wiley & Sons for the Military Operations Research Society, 1995.

R. Chase and N. Aquilano, *Production and Operations Management*, Irwin, Chicago, 1995.

T. Coleman, M.A. Branch, and A. Grace, *Optimization Toolbox Users Guide (Version 2)*, The MathWorks, Inc., 1999.

P. Davis, D. Blumenthal, and D. Gaver, "Appendix I: Combat Modeling Issues" in Technology for the United States Navy and Marine Corps, 2000-2035: Becoming a 21st Century Force, Volume 9: Modeling and Simulation, National Academy of Sciences, Panel on Modeling and Simulation, 1997. (http://www.nationalacademies.org/cpsma/nsb/msi.htm)

P. Davis and J.Bigelow, Experiments in Multiresolution Modeling (MRM), RAND Report MR-1004-DARPA, 1998. (http://www.rand.org/publications/MR/MR1004)

P. Davis and B. Zeigler, "Appendix E: Multi-resolution Modeling and Integrated Families of Models" in Technology for the United States Navy and Marine Corps, 2000-2035: Becoming a 21st Century Force, Volume 9: Modeling and Simulation, 1997. (http://www.nationalacademies.org/cpsma/nsb/mse.htm)

R.J. Elliot, L. Aggoun, and J.B. Moore, *Hidden Markov Models: Estimation and Control*, Springer-Verlag, 1997.

B.W. Fowler, Towards a Formal Theory of Aggregation, *Military Operations Research*, 4(1), pp. 5-19, 1999.

A. Golan, G. Judge, and D. Miller, *Maximum Entropy Economics: Robust Estimation with Limited Data*, Wiley, 1996.

R.P. Guidorzi, N.P. Losito, and T. Muratori, "The Range Error Test in the Structural Identification of Linear Multivariable Systems", *IEEE Transactions on Automatic Control*, 27(5), pp. 1044-1053, 1982.

R.P. Guidorzi, "Invariants and Canonical Forms for Systems Structural and Parametric Identification", *Automatica*, 17(1), pp. 117-133, 1981.

R.P. Guidorzi, "Canonical Structures in the Identification of Multivariable Systems", *Automatica*, 11(4), pp. 361-374, 1975.

Handbook of Mathematical, Scientific, and Engineering Formulas, Tables, Functions, Graphs, Transforms, Research and Education Foundation (REF), NY, 1984, pg 588.

R.J. Hillestad and M.L. Juncosa, Cutting Some Trees to See the Forest: On Aggregation and Disaggregation in Combat Models, in Bracken, Kress, and Rosenthal (eds.) *Warfare Modeling*, published by John Wiley & Sons for the Military Operations Research Society, pp. 37-62, 1995.

N.K. Jaiswal and B.S. Nagabhushana, Termination Decision Rules in Combat Attrition Models, in Bracken, Kress, and Rosenthal (eds.) *Warfare Modeling*, published by John Wiley & Sons for the Military Operations Research Society, pp. 273-287, 1995.

K. Lee and P.A. Fishwick, "Dynamic Model Abstraction", in *Proceedings of the 1996 Winter Simulation Conference*, San Diego, CA, (1996), pp. 764-771. (http://www.cise.ufl.edu/~fishwick/tr/tr96-031.html)

L. Ljung, *System Identification: Theory for the User*, Prentice-Hall, 1987.

D.A. Samuelson and R.L. Sims, Quick-Response Assessment of Forward Air Control and Communications Using a Queueing Model and Digital Simulation, in Bracken, Kress, and Rosenthal (eds.) *Warfare Modeling*, published by John Wiley & Sons for the Military Operations Research Society, pp. 407-415, 1995.

G.A. Seber and C.J. Wild, "Chapter 8: Compartmental Models" in *Nonlinear Regression*, pp. 367-432, Wiley, 1989.

Using MATLAB (Version 5.3), The Mathworks, Inc., 1999.

R.J. Vanderbei, LOQO Users Manual – Version 4.05, Princeton University, Operations Research and Financial Engineering, Technical Report ORFE-99-xx, July, 1999.

B.P. Zeigler, Review of Theory in Model Abstraction, in *Proceedings of the SPIE Conference on Enabling Technology for Simulation Science II*, Orlando, FL, April, 1998.

Appendix A - Attrition Simulation: Methods for Calculating Force Strengths, Initial Force Levels, and Fire Allocation

Force Strength Calculation

A force strength , S, measures the relative strength of a collection of weapon systems. It can be computed as follows:

N^s = the number of different weapon types on side s; $s \in \{R, B\}$
V_i^s = the weapon score for weapon system i on side s; $i = 1,2,3, .. N^s$, $s \in \{R, B\}$
W_i^s = the number of weapons of type i on side s; $i = 1,2,3, .. N^s$, $s \in \{R, B\}$

$$\text{Then: } S^s = \left[\sum_{i=1}^{N^s} V_i^s W_i^s\right]^\rho ; \qquad s \in \{R, B\} \qquad [A.1]$$

where ρ may vary according to the force scoring assumptions. Typical values are 1 or ½. The force ratios are then determined by:

$$r_\rho = S^B / S^R = \frac{\left[\sum_{i=1}^{N^B} V_i^B W_i^B\right]^\rho}{\left[\sum_{i=1}^{N^R} V_i^R W_i^R\right]^\rho} \qquad [A.2]$$

The terms, V_i^s, are functions of the known kill probabilities, inter-firing times, and the fire allocation assumptions. To calculate, let:

E_i^s = the average number of engagements per time period made by a weapon of type *i* on side *s* (against all enemy weapons). Note that E_i^s is the inverse of the mean of the interfiring time distribution for weapon system *i*; $i = 1,2,3, .. N^s$, $s \in \{R, B\}$.

P_{ij}^s = the probability of kill per engagement by a weapon of type *i* on side *s* when that weapon is engaging an enemy of type *j*; $i = 1,2,3, .. N^s$, $j = 1,2,3, .. N^{s'}$, $s \in \{R, B\}$.

A_{ij}^s = the allocation of fire from a weapon of type *i* on side *s* when that weapon is engaging an enemy of type *j*; $i = 1,2,3, .. N^s$, $j = 1,2,3, .. N^{s'}$, $s \in \{R, B\}$. Note that $\sum_{j=1}^{N^{s'}} A_{ij} = 1$.

K_{ij}^{s} = the expected rate at which weapon systems of type *i* on side *s* kill weapon systems of enemy type *j*, i = 1,2,3, .. N^s , j = 1,2,3, .. $N^{s'}$, s ∈ {R, B} This is equivalent to $E_i^s A_{ij}^s P_{ij}^s$.

A variety of scoring equations based on these quantities has been proposed. The best for our purposes appears to be the DYNPOT (dynamic potential) method, which considers both lethality and vulnerability in both the short and long run time frames. An earlier and more well known method, Anti-Potential Potential (APP), has been used in various ways in a number of military simulations such as IDAGAM, INBATIM, JCS FPM, and IDAPLAN, which are all dynamic theatre-level models of ground and air combat. However, APP has the flaws that 1) it addresses lethality but ignores the relative *vulnerability* of weapon systems, and 2) it computes an instantaneous score for a weapon, but not a long run score. The DYNPOT technique addresses these shortfalls, resulting in a set of equations for the weapon scores (V_i^s 's) as follows:

$$\beta V_i^s = \frac{\sum_{j=1}^{N^{s'}} K_{ij}^s V_j^{s'}}{\sum_{j'=1}^{N^{s'}} \hat{K}_{j'i}^{s'}}; \quad i = 1,2,3, .. N^s , s \in \{R, B\} \qquad [A.3]$$

where β is a coefficient that is constant across all weapon systems and is calculated along with the V_i^s 's. The denominator terms, $\hat{K}$, are defined by:

$$\hat{K}_{ij}^s = \begin{cases} W_i^s K_{ij}^s \big/ W_j^{s'}, & W_j^{s'} > 0 \\ 0, & \text{otherwise} \end{cases} \quad ; i = 1,2,3, .. N^s , j = 1,2,3, .. N^{s'} , s \in \{R, B\} \quad [A.4]$$

which can be interpreted as the rate at which weapon systems of type *j* on side s' is being killed by all weapons of type *i* on side *s*.

To solve for the V_i^s values, we set one of them to 1.0, and assuming that we know the K_{ij}^s, we can solve a set of equations[12] for β and the remaining V_i^s.

When initializing our attrition simulation model we will have a known desired force strength ratio, but will not know the W_i^s's that along with the K_{ij}^s's will achieve that ratio. These can be determined via an iterative algorithm as shown below.

[12] Following the guidance of Anderson and Miercort (1995) we set β to 1.0 and solve the $N^R + N^B$ linear equations for $N^R + N^B - 1$ unknowns.

Algorithm for Calculating Initial W_i^s

0. Select initial values for $W_i^s > 0$, i = 1,2,3, .. N^s ,s $\in$ {R, B}; set $V_1^s = 1$; select ρ, and the desired force strength ratio r_ρ^0; select $\varepsilon > 0$ = the maximum deviation from r_ρ^0.
1. Solve equations [A.4] and then [A.3] for the V_i^s's and β.
2. Solve equation [A.2] for the force strength ratio, r_ρ.
3. If $|r_\rho^0 - r_\rho| \leq \varepsilon$ then STOP, otherwise go to step 4.
4. If $(r_\rho^0 - r_\rho) < 0$ (force strength ratio is too high) set s = R; otherwise (force strength ratio is too low) set s = B and find:

$$\nabla r_\rho^i(s) = \frac{\partial r_\rho}{\partial W_i^s}; \text{ i} = 1,2,3, .. \ N^s$$

$$\text{i}': \ \nabla r_\rho^{i'}(s) \leq \nabla r_\rho^i(s) \ \text{ i,i}' = 1,2,3,\ldots \text{N}^s \qquad \text{(see note)}$$

5. Set $W_{i'}^s = W_{i'}^s + 1$ and GO TO Step 1.

Note: The inequality, $\leq$, is used to distribute the increases among the various weapon system types. The inequality, $\geq$, would be faster, but would cause all the increases to occur in a single weapon system type.

Note that, at each iteration, the side that is "too weak" with respect to the desired force strength ratio has the population of one weapon system type increased by 1. We do this, rather than decrease weapons on the "too strong" side, to ensure convergence and to avoid numerical problems resulting from weapon system populations possibly going to zero.

Fire Allocation Methodology

In general, fire allocation is a function of the current force levels, the average rates of fire, the kill probabilities, and any additional externally provided allocation parameters, $\tilde{A}$ (Anderson and Miercort, 1989). That is:

$$A_{ij}^s = F_{ij}^s(W, E, P, \tilde{A}) \text{ where } \sum_{j=1}^{N^i} A_{ij}^s = \begin{cases} 0 & \text{if weapon type i engages no targets} \\ 1 & \text{otherwise} \end{cases}$$

There are two major types of fire allocation rules within this framework – strict priorities, and fractional allocations. Strict priority methods select a single target weapon system at each firing event, while fractional allocation methods distribute the fire proportionally (*probabilistically*, in our stochastic model) among targets at each firing event. The simplest fractional allocation method is one in which the fractions are fixed throughout the course of the model (except when the quantity of an opposing weapon system type becomes 0, in which case a valid reallocation is performed). The problem with this technique is that the target choice is then independent of the number of targets of each type present, which is clearly unrealistic. Strict priority rules also have problems related

to their inability to incorporate the random effects of unmodeled variables such as enemy detection, local terrain features, or proximity. We use a fractional allocation method that is dependent on current model conditions. The method provides values proportional to the number of enemy weapons:

$$A_{ij}^{s} = \frac{C_{ij}^{s} W_{j}^{s'}}{\sum_{j=1'}^{N^{s'}} C_{ij}^{s} W_{j}^{s'}}$$

The C_{ij}^{s} terms can be provided externally[13] or calculated from other model parameters. We use a form of the latter strategy that exploits our knowledge of E and P. Namely:

$$C_{ij}^{s} = E_{i}^{s} P_{ij}^{s} E_{j}^{s'} P_{ij}^{s'}$$

which tends to focus fire on targets that are most effective by weapons which are the most effective against them.

☆U.S. GOVERNMENT PRINTING OFFICE: 2000-510-079-81242

[13] This general approach is used in combat simulations such as IDAGAM, INBATIM, TACWAR, JCS FPM, and IDAPLAN.

MISSION OF AFRL/INFORMATION DIRECTORATE (IF)

The advancement and application of information systems science and technology for aerospace command and control and its transition to air, space, and ground systems to meet customer needs in the areas of Global Awareness, Dynamic Planning and Execution, and Global Information Exchange is the focus of this AFRL organization. The directorate's areas of investigation include a broad spectrum of information and fusion, communication, collaborative environment and modeling and simulation, defensive information warfare, and intelligent information systems technologies.