QUANTITATIVE DECISION SUPPORT FOR UPGRADING COMPLEX SYSTEMS OF SYSTEMS

By

RONALD R. LUMAN

B.A. May 1976, Middlebury College
M.S. May 1978, Michigan State University
M.S. May 1986, Johns Hopkins University

A Dissertation submitted to

The Faculty of

The School of Engineering and Applied Science
of the George Washington University in partial satisfaction
of the requirements for the degree of Doctor of Science

November 19, 1997

Dissertation directed by

Dr. Howard Eisner
Professor of Engineering Management
Systems Engineering

ABSTRACT

The engineering of complex systems of systems has been receiving greatly increased amounts of attention in recent years. Within the Department of Defense, “system of systems” terminology is now widely used to describe how the successful, combined operation of many platforms, weapon systems, and communication systems is necessary to achieve an overall warfare objective, especially in joint operations. Although the characteristics and system engineering challenges associated with systems of systems are becoming well understood, effective architecting approaches that enable cost/performance trades are still immature.

A systematic approach to considering how best to upgrade specific, complex systems of systems is postulated and demonstrated. The process treats cost as the independent variable (CAIV) and seeks to find the “best” point design that may involve upgrading all component systems simultaneously, not just one at a time. The process has been demonstrated on a naval mine countermeasures (MCM) system of systems representation of sufficient complexity and detail to demonstrate the feasibility of the approach. The process formulates a constrained, nonlinear optimization problem whose objective function is a representation of the top-level measure of effectiveness (MOE), with constraints represented by functionalized Performance Based Cost Models, secondary MOEs, and technology-driven bounds on system measures of performance (MOPs). Both closed-form and simulation-based optimization approaches have been demonstrated and differences quantified, including the suboptimality of considering just one system at a time.

Due to the nature of complex system of systems interactions, implementation of this optimization technique on problems of national interest will require a simulation to represent the mapping of system MOPs to single system MOEs and on to the overarching system of systems MOE. A stochastic simulation of the MCM system of systems was therefore also implemented and optimized utilizing a constrained variant of the Simultaneous Perturbation Stochastic Approximation method.

This process therefore demonstrates a disciplined, quantitative approach to developing system of systems upgrade options for very complex situations, which can result in more effective and comprehensive systems acquisition and technology investment strategies. A secondary benefit is that the process can be used as a framework for the utilization of campaign-level simulations to support acquisition decisions.

TABLE OF CONTENTS

Chapter

LIST OF ILLUSTRATIONS

Figure

CHAPTER 1

INTRODUCTION

Background

The engineering of complex systems of systems has been receiving greatly increased amounts of attention recently. Within the Department of Defense, system of systems terminology is now widely used to describe how the successful, combined operation of many platforms, weapon systems, and communication systems is necessary to achieve an overall warfare objective, especially in joint operations. This increased level of complexity has become a concern at the highest levels of command, as General John Sheehan, Commander in Chief of U.S. Atlantic Forces, recently wrote:

> *"Victory will depend on the ability to master the 'system of systems' composed of multiservice hard- and soft-kill capabilities linked by advanced information technologies."* [1]

And Admiral Owens, Vice Chairman of the Joint Chiefs of Staff notes that systems of systems have arisen not by design, but in response to the vision of users who recognize the tremendous potential of systems working together towards broad, common objectives:

> *"We have cultivated a planning programming and budgeting system that tends to handle programs as discrete entities…Yet, the interactions and synergisms of these systems constitute something new and very important. What is happening is driven in part by broad conceptual architectures---and in part by serendipity: It is the creation of a new system of systems."* [2]

[1] Sheehan, Gen. J.J., "Next Steps in Joint Force Integration", *Joint Force Quarterly (Supplement, 1/6/97),* Autumn 1996.

[2] Owens, Adm. W.A., "The Emerging System of Systems", *U.S. Naval Institute Press*, May 1995.

Although the characteristics and system engineering challenges associated with systems of systems are becoming well understood, effective architecting approaches are still immature for systems of systems [3,4].

This dissertation specifically addresses the issue of how best to upgrade a complex system of systems, once the need to do so has been realized. The system of systems is considered as a whole entity and a quantitative methodology for determination of an optimal upgrade suite under cost and technology constraints is demonstrated. The methodology utilizes a multi-disciplinary approach including operations analysis, cost modeling, nonlinear optimization, and stochastic modeling and simulation.

[3] Manthorpe, W.H.J., Jr., "The Emerging Joint System of Systems: A Systems Engineering Challenge and Opportunity for APL", *Johns Hopkins APL Technical Digest*, Vol. 17, No. 3, 1996.

[4] Luman, R.R., and Scotti, R.S., (1996). "The System Architect Role in Acquisition Program Integrated Product Teams", *Acquisition Review Quarterly,* DSMC Press, Fort Belvoir, VA; Vol.3 No.2.

Systems of Systems Definitions and Concepts

A complex system of systems is generally viewed as having the following characteristics:[5]

- comprised of several independently acquired systems, each under a nominal systems engineering process
- time phasing between each system's development is arbitrary and not contractually related
- system couplings are neither totally dependent or independent, but rather interdependent
- individual systems are generally uni-functional when viewed from the system of systems perspective
- optimization of each system does not guarantee overall system of systems optimization
- combined operation of the systems constitutes and represents satisfaction of an overall mission or objective

Some examples of existing, complex systems of systems that exhibit all of these characteristics are:

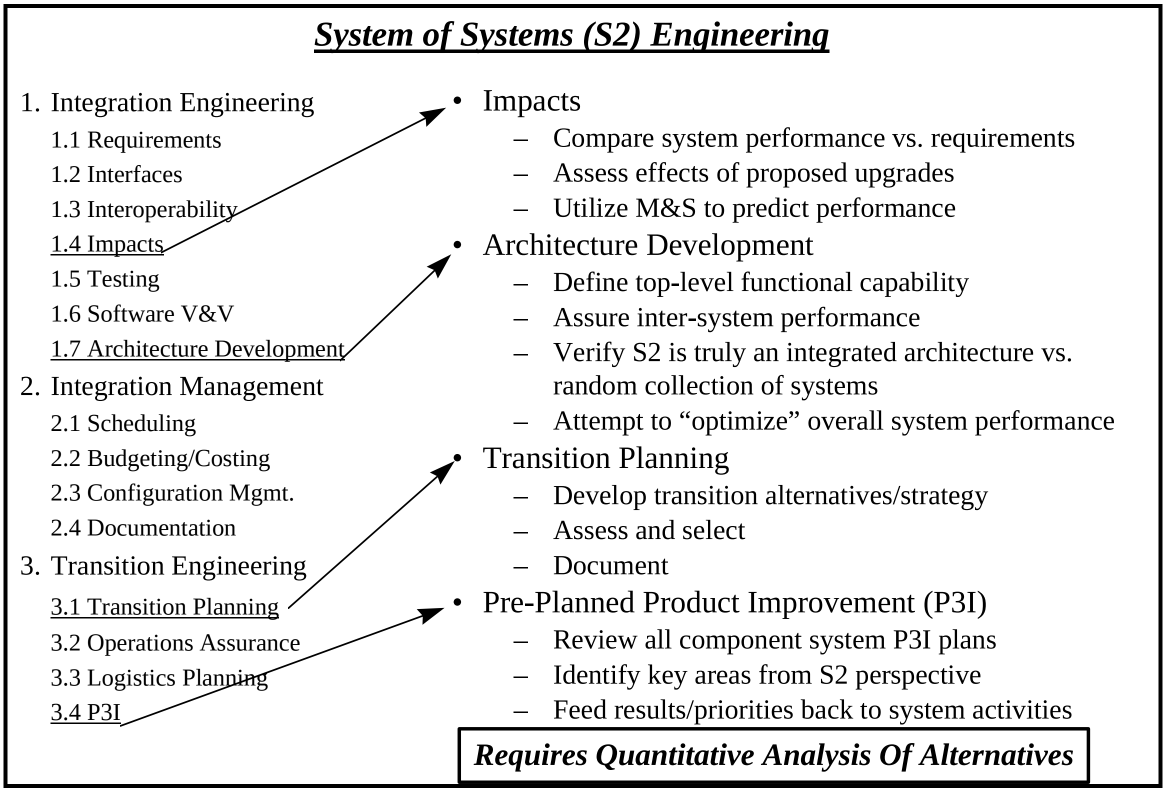
- National Aviation System: planes, airports, airways, air traffic control
- Naval Mine Countermeasures Force: search, sweep, neutralization systems
- Naval Surface Fire Support: reconnaissance, targeting, weapon systems
- Theater Ballistic Missile Defense: surveillance, tracking, interceptor systems

[5] Eisner, H., Marciniak, J., and McMillan, R., "Computer-Aided System of Systems (S2) Engineering", *Proceedings of the 1991 IEEE International Conference on Systems, Man, and Cybernetics*, 13-16 October 1991, University of Virginia, Charlottesville, VA.

Although it is difficult to know where to draw the line to form a boundary to describe a particular system of systems, it is generally viewed as a coherent entity when there is a recognition that overall management control over the autonomously managed systems has become mandatory. Unfortunately, it is rare that a large, complex system of systems is developed under a single, architecture resulting from a strategic development decision. Component systems are developed one by one, and the full system of systems evolves over a period of time that may be measured in decades as various leadership entities develop enhanced visions of how systems can be used together to achieve larger objectives. And although each system may have been justified and designed based upon sound system engineering principles to fulfill a perceived functional or performance need, its requirements and design most likely did not develop in response to concerns over the complete system of systems objectives.

A framework for conducting system engineering at the system of systems (S2) level has been developed[6], but has not achieved widespread acceptance. Figure 1 lists the elements of S2 Engineering and highlights those aspects that require a quantitative analysis of alternatives to upgrading an extant system of systems—the subject of this dissertation.

[6] Eisner, H., Marciniak, J., and McMillan, R., "Computer-Aided System of Systems (S2) Engineering", *Proceedings of the 1991 IEEE International Conference on Systems, Man, and Cybernetics*, 13-16 October 1991, University of Virginia, Charlottesville, VA.



System of Systems (S2) Engineering

1. Integration Engineering
   - 1.1 Requirements
   - 1.2 Interfaces
   - 1.3 Interoperability
   - 1.4 Impacts
   - 1.5 Testing
   - 1.6 Software V&V
   - 1.7 Architecture Development
2. Integration Management
   - 2.1 Scheduling
   - 2.2 Budgeting/Costing
   - 2.3 Configuration Mgmt.
   - 2.4 Documentation
3. Transition Engineering
   - 3.1 Transition Planning
   - 3.2 Operations Assurance
   - 3.3 Logistics Planning
   - 3.4 P3I

- Impacts
  - Compare system performance vs. requirements
  - Assess effects of proposed upgrades
  - Utilize M&S to predict performance
- Architecture Development
  - Define top-level functional capability
  - Assure inter-system performance
  - Verify S2 is truly an integrated architecture vs. random collection of systems
  - Attempt to "optimize" overall system performance
- Transition Planning
  - Develop transition alternatives/strategy
  - Assess and select
  - Document
- Pre-Planned Product Improvement (P3I)
  - Review all component system P3I plans
  - Identify key areas from S2 perspective
  - Feed results/priorities back to system activities

Requires Quantitative Analysis Of Alternatives

Figure 1. System of Systems (S2) Engineering Elements

CHAPTER 2

MANAGEMENT ISSUES

Usual Approach

Often, as in the case of the DoD, a program executive officer will be responsible for a collection of system acquisition programs, each of which can be viewed as part of a larger system of systems—though this collection may not necessarily fully comprise that system of systems. Were (s)he to have the luxury to architect a complete system of systems from scratch, it could be done by applying an extension of the usual system engineering approach, treating each acquisition system as a sub-system of the larger entity[7].

Rather than architecting a system of systems in its entirety, the programs executive is often faced with deciding how best to *upgrade* an existing system of systems. This generally means either beginning a new acquisition program to add a new system to the overall system of systems (additional functionality) or inserting advanced technology into an existing system via the upgrade or modification process[8]. Significant constraints and boundary conditions are placed upon these executives, including budgets, politics, ill-defined and competing mission objectives, and of course, technology itself. Many new initiatives are underway under the umbrella of "Acquisition Reform" to encourage

[7] Eisner, H., McMillan, R., Marciniak, J., Pragluski, W., "RCASSE: Rapid Computer-Aided System of Systems (S2) Engineering," *Proceedings of the National Council on Systems Engineering*, 26-28 July 1993, Washington, D.C.

[8] Evans, LtCol. T.R., Lyman, Cdr. K.M, and Ennis, LtCol. M.S., "Modernization in Lean Times: Modifications and Upgrades". *Report of the 1994-1995 DSMC Military Research Fellows,* Defense Systems Management College Press, Fort Belvoir, VA, July 1995.

acceleration of systems development time, delivery of affordable systems, and risk mitigation through adoption of commercial off the shelf (COTS) components or technologies. These attempts at accelerating the usual acquisition cycle include such innovative and complementary measures as Advanced Technology Demonstrations (ATDs) and Advanced Capability Technology Demonstrations (ACTDs); often described, respectively, as "technology pushes" and "military need pulls"[9].

Although these initiatives promote the quick fielding of new, militarily useful technologies, they do not represent a disciplined approach to considering how best to upgrade specific, complex systems of systems under the constraints mentioned above. Development of such an approach is the objective of this dissertation effort.

The usual approach in DoD to assessing whether to go forward with a new system development has been to conduct a Cost and Operational Effectiveness Analysis (COEA). The objective of the COEA (or the replacement "analysis of alternatives" procedure) is to determine whether the proposed system is the most cost effective alternative to meeting a certified military need[10]. A typical analysis approach is to utilize modeling and simulation (M&S) to estimate the marginal utility of proposed system point designs (across a range of system measures of performance (MOPs)) to a larger warfare or campaign objective. The simulation is run on a carefully selected set of applicable scenarios with and without the system alternatives to determine the best alternative among those hypothesized. A multi-objective metric that reflects costs and other relevant

[9] Lynn, L., "The Role of Demonstration Approaches in Acquisition Reform", *Acquisition Review Quarterly*, (1994, Vol. 1, No. 2).

[10] Department of Defense. (1996). DoD 5000.2-R, "Mandatory Procedures for Major Defense Acquisition Programs and Major Automated Information Systems,. Washington, DC.

factors may be used to compare alternatives. This metric may attempt to reflect expert opinion as to military value of the alternatives that are not captured by the quantitative analysis due to limitations of fidelity or scope. However, the primary shortcoming of the general approach to making upgrade decisions to a system of systems is that just one component system is considered at a time, in a “stovepipe” fashion. It cannot generate the “best” alternative from the system of systems perspective, since it considers replacement or addition of just one component system rather than enhancements across the full system of systems.

A significant observation is that the DoD acquisition community strongly prefers quantitative “engineering analysis” over qualitative “decision support” methods such as the Analytic Hierarchy Process. This is perhaps because the community is dominated by engineers and scientists who eschew attempts to convert opinion and judgments into metrics—hence the heavy emphasis on modeling and simulation as the basis for decisionmaking. A recent article in IEEE Engineering Management shows this to be widespread throughout the technical and scientific community.[11]

In summary, we are focused on “upgrading” vs. “design” of systems of systems because (1) all proposed systems/upgrades must fit into an extant system of systems, (2) there is rarely an opportunity to architect a major system of systems from scratch, (3) requirements usually evolve in consideration of legacy systems’ capabilities and management, and (4) we can often take advantage of available M&S that can be adapted for decision support use if we take the view of upgrading an extant system of systems.

[11] Cabral-Cardoso, C. and Payne, R. L., “Instrumental and Supportive Use of Formal Selection Methods in R&D Project Selection”, IEEE Transactions on Engineering Management, Vol. 43, No. 4, November 1996, pp. 402-410.

System of Systems Upgrade Decision Objectives

The decision maker is generally trying to solve one of two problems, though not always in an explicit manner: (1) maximize the system of systems' performance subject to a cost constraint or (2) minimize additional cost under performance constraints. Cost constraints usually appear very rigid at the outset. Recent DoD acquisition reform initiatives have softened hard budget allocations in favor of an approach known as Cost as Independent Variable (CAIV). Application of the CAIV approach requires a representation of a system's performance as a function of cost, referred to as a Performance-Based Cost Model (PBCM). This is almost never applied at the system of systems level, however. Explicit performance constraints are expressed as minimum performance requirements and may be self-imposed for political or strategic reasons, or perhaps externally mandated due to advanced competition/military threats. Implicit performance constraints are generally due to technological limitations. Again, the effects of component systems' performance constraints on overall system of systems' effectiveness is almost never well-understood.

These upgrade decisions are generally made and reviewed annually for all warfare or program areas as part of strategic planning and budgeting processes in DoD. Upgrade options generally take four forms, depending upon which forcing conditions are most pressing:

1. a new type of system (i.e., additional functionality) must be added to the system of systems

2. additional numbers of existing component systems must be procured (enlarging the scope and capability of the system of systems and offering an opportunity to insert advanced technology)
3. existing component systems must be replaced due to aging or obsolescence (also offering an opportunity to enhance the system of systems' performance and/or functionality through advanced technology insertion)
4. existing component systems must be upgraded due to requirements pressure or availability of advanced technology

Of course, final acquisition upgrade decisions are based upon a variety of factors, many of which are not amenable to quantification and objective analyses. However, decision makers generally agree that their decisions are easier when a thoughtful analysis that provides a measure of the marginal utility of each upgrade option to their system of systems' effectiveness is available. This is easy to say, but hard to do. The most common approach in DoD to getting a grip on system of systems' effectiveness is through the use of wargames and campaign-level simulations. Unfortunately, these approaches are often insensitive to all but the most dramatic changes in capability. That is, it is often impossible to isolate the contribution to an overall campaign due to small changes in component systems' functionality and/or performance. Therefore, acquisition trade analyses similar to the COEA M&S analysis described above are often done on a scale more amenable to quantitative analysis. Some of the challenges inherent in objectively trading off system upgrade options:

- the system of systems itself is not well defined (i.e., what is the boundary of the system of systems with regard to environment and other systems?)

- the measures of effectiveness (MOEs) for the system of systems are not well-defined
- field data on (MOPs) for existing component systems is limited or altogether non-existent (a challenging M&S VV&A issue)
- MOPs and CONOPS for upgrade options are not well-defined
- budget constraints are not fixed and usually shrinking (both current and out-year)
- marginal utility of proposed additional functionality and/or enhanced performance is not well understood

Regarding models and simulation, a well-accepted and validated representation of the system of systems is not usually available that would be suitable for MOE/MOP analysis purposes. This is a reflection of the usual single-system focus as well as the tendency to create sophisticated M&S first and seek ways to use it afterwards.

CHAPTER 3

APPROACH

Dissertation Objective

The dissertation objective is to develop a quantitative process/methodology to support system of systems upgrade decisions so we can answer the question: "From the system of systems perspective, where are the limited upgrade resources best applied?"

The overall approach is to develop and demonstrate a process that will enable a domain expert systems architect or engineering team to generate an optimal suite of upgrade design requirements subject to stated constraints in accordance with a specified MOE for a particular complex system of systems. This process will be demonstrated on a real world, contemporary system of systems in sufficient detail to demonstrate the feasibility of the approach—a practical, proof-of-principle demonstration. The mature process will feature a constrained, nonlinear optimization algorithm whose objective function is a simulation that represents the defined system of systems' effectiveness. This is necessary to take advantage of substantial investments in system of systems M&S and to avoid unnecessary simplification of the system abstraction required to obtain closed form expressions of typical, complex systems of systems behavior. As is usually the case, a balance must be struck between model fidelity and execution time due to intense computational burden of advanced M&S. These considerations will drive selection of the system of systems' MOE/MOP and PBCM structure.

Proposed System of Systems Quantitative Decision Support Process

Key steps of the process are as follows:

1. Define the overall system of systems, its components, and its missions or scenarios of interest.
2. Define critical MOPs and MOEs:
    a) overarching MOE for the full system of systems that expresses the decision makers' objective
    b) one characteristic MOE for each system and how it contributes to the overarching MOE
    c) component systems' MOPs
3. Specify initial boundary conditions for the upgrade process
    a) cost constraints on component systems and the overall system of systems
    b) technological and requirements constraints on MOPs
    c) force structure constraints, such as minimum and maximum numbers of each type of system
    d) potential secondary MOE threshold function constraint
4. Formulate Performance Based Cost Models (PBCM) for each component system by parameterizing system cost as a function of its MOPs.
5. Formulate an appropriate first order model that will capture the mapping from component system MOPs to system MOEs and eventually the overarching MOE. Alternatively, select an appropriate M&S implementation that evaluates the desired objective function and MOE constraints as a function of component

systems' MOPs. (Constructing expressions that model the system of systems' top-level performance is important for initial problem understanding, but will probably not be sufficient to adequately capture system interactions and performance drivers. It is envisioned that this step will expose the requirement to utilize advanced M&S to represent sufficient complexity necessary to provide credible analyses to support decisions regarding complex, high value systems.)

6. Solve the resulting constrained, nonlinear (stochastic) system of systems performance optimization problem under sets of constraints and scenarios as defined by the expert system architect that are sufficiently broad to provide a complete range of applicable upgrade options to the decision maker. [A solution to a specific constrained problem formulation yields parameters that represent one system of systems requirements suite, or upgrade option. The solution will still require further evaluation to determine design implications for each system. In this way, the process provides *support* to the decision process rather than make the decision per se.]

7. Effectively communicate results of the process to the decision maker or decision making body.

Development Approach

The process will be developed utilizing an evolutionary prototyping approach in a cycle of problem formulation, solution, evaluation, refinement, expansion and generalization. Critical stages in the development are as follows:

- Formulate the general nonlinear optimization problem for this process. The particular form has two nonlinear constraints, upper and lower MOP bounds, and a nonlinear objective function, as shown in Chapter 4.
- Apply the optimization process to an existing, complex system of systems for which the author has current domain expertise: naval mine countermeasures (MCM). Model formulation has two parts:
    1. System Performance Model. A simplified, but realistic performance model of a dual system of systems is described in Chapter 5, in which closed form expressions for MOPs, MOEs, and constraints are developed. Through this example, it became clear that the objective function is generally a non-linear function of all component systems' MOPs. This non-linear objective function therefore represents complex interactions between component systems. As simplifying assumptions (engagement rules, environmental dependencies, etc.) are relaxed, it becomes impractical to obtain closed-form expressions that map MOPs up to system of system level MOEs. However, it is assumed that the rules governing the interaction between the component systems and the environment are known and can be simulated.

2. System Cost Model. A variant of a "Cost-Based Performance Model" is also formulated in Chapter 5, in which cost is parameterized by the MOPs. Essentially, each MOP is considered to be a cost driver to an identifiable subsystem, who's cost can be estimated based on the cost-driver MOP values.

- Investigate various applicable optimization techniques and select the most promising for this particular application. Although constrained, nonlinear optimization does not appear to have yet been applied to the system of systems upgrade problem, optimization and simulation have been combined before, and the literature has been further examined for practical insights.[12,13,14] Algorithm efficiency has been a primary consideration, with the long term perspective that the objective function will eventually be evaluated via simulation (see Chapter 6).
- Solve the MCM closed form problem to gain insight into process viability and experience with the candidate search techniques. Then, revise the process and general problem formulation as necessary.
- Generalize and demonstrate the process for the case where it is impractical to obtain closed form expressions for the objective function to sufficiently represent the complex interactions of systems, environment, and scenario. For a complex system of systems, the use of advanced modeling and simulation will be necessary to overcome the difficulties in obtaining closed form expressions for the objective function.

[12] Lee, K-H, Eom, I-S, Park, G-J, Lee, W-I, (1996). "Robust Design for Unconstrained Optimization Problems Using the Taguchi Method", *AIAA Journal*, Vol.34, No.5, 1059-1063.

[13] Fu, M.C., (1994). "Optimization via Simulation: A Review", *Annals of Operations Research*, **53**, 199-248.

Embedding a simulation inside a non-linear, constrained search algorithm has been done before to determine control settings on a variety of single system simulation types[15,16]. Extension of that application to efficiently determine system design parameters in the system of systems context described here can revolutionize the way campaign-level M&S is utilized to support acquisition decisions. A variety of simulation products and system domains were considered for this proof-of-concept analysis. Candidate system of system domains that were investigated to varying degrees include:

- Naval Air Defense. Systems: aircraft, missiles, radars, acquisition and tracking software, etc. Available simulation: TACAIR (Tactical Air).
- Anti-Submarine Warfare. Systems: submarines, torpedoes, sonars. Available simulation: ORBIS (Object Oriented Rule Based Interactive Simulation).
- Ballistic Missile Defense. Systems: sensors, platforms, battle management system (identification, tracking, targeting, battle damage assessment, etc.) and interceptor missiles/devices. Available simulation: EADSIM (Extended Air Defense Simulation).
- Naval MCM. Systems: ships, aircraft, detection, classification, identification, neutralization sensors and devices.

[14] Glynn, P.W., (1989).   "Optimization of Stochastic Systems via Simulation", Proceedings of the 1989 Winter Simulation Conference, ed. E.A. MacNair, K.J. Musselman, P. Heidelberger, IEEE, Piscataway, NJ, 90-105.

[15] Hill, S.D. and Fu, M.C., (1997). "Optimization of Discrete Event Systems via Simultaneous Perturbation Stochastic Approximation", *Transactions of the Institute of Industrial Engineers, Special Issue on Operations, Engineering, and Simulation, Vol. 29, Issue #3, pp.233-243*.

[16] Kleinman, N.L., Hill, S.D., and Ilenda, V.A., (1997). "SPSA/SIMMOD Optimization of Air Traffic Delay Cost", *Transportation Science*, to appear.

- Air Traffic Control. Systems: aircraft, radars, positioning systems, tracking systems, air traffic delays. Available simulation: SIMMOD (Simulation Model).

Balancing simulation (1) availability, (2) validity, (3) applicability (including PBCM aspects), (4) efficiency, and (5) author's domain expertise, it was decided to generate an analytic naval MCM system of systems model (Chapter 5) and implement it both analytically (expected value sense) and as a Monte Carlo Simulation using MATLAB® (Chapters 7 and 8). This facilitates a self-consistent comparison of the closed form analytic model results and those obtained via simulation.

Finally, the fully developed process should be compared against existing processes and evaluated for its utility to the decision maker when attempting to upgrade a complex system of systems. We are investigating a new process for supporting acquisition decisions, which is not unlike an expert system, except that we have no accepted knowledge base to capture. For expert systems, validation efforts generally concentrate on comparisons against documented test cases. O'Keefe et al[17], recommend testing against a small number of complex cases and asking a panel of experts to assess how well the process handles them.

The appropriate process to compare against is that of the COEA. Unfortunately, utilization of quantitative methods in COEAs are somewhat unique to each study, and concentrate on a single system. Furthermore, optimization methods have not been

[17] O'Keefe, R.M, Balci, O., and Smith, E.P., "Validating Expert System Performance", *IEEE Expert*, Winter 1987, pp. 81-87.

applied in COEA analyses in the sense of a rigorous “search” for the best set of system requirements. Rather, a “best” option is selected from a finite set of point designs/requirements.

The comparison approach selected here (Chapter 7) is to compare system of systems optimization results to results obtained by optimizing one system at a time. This verifies implementation, self-consistency, and quantifies the improvement to be realized in taking the system of systems viewpoint for the MCM situation. It also highlights assumptions (implicit and explicit) that a single system analyst must make concerning the other systems in the system of systems, and assesses the impact of self-consistent but erroneous assumptions.

CHAPTER 4

GENERAL SYTEM OF SYSTEMS PERFORMANCE MODEL

Consider *n* types of systems, S_i, that comprise a system of systems, *S*, with the following characteristics and constraints:

- $S = \{S_1, \ldots, S_n\}$ [note: could index as S_{ij} to indicate the j^{th} system of type *i*)
- There are m_i systems of type *i*, and the total number of systems is $m = \sum_{i=1}^{n} m_i$, $\mathbf{m} = \{m_1, \ldots, m_n\}$. The minimum number of each system type required for the system of systems is designated as $\mathbf{m}^L$.
- Each system type has a set of r_i measures of performance (MOP): $\mathbf{p}_i = \{p_{i,1}, \ldots, p_{i,ri}\}$. Thus each $\mathbf{p}_i$ has dimension r_i and $r = \sum_{i=1}^{n} r_i$.
- Each system type has one overall measure of effectiveness (MOE), $E_i = \mathrm{f}_i(\mathbf{m}, \mathbf{p}_1, \ldots, \mathbf{p}_n)$, <u>developed specifically to reflect how it contributes to the overarching MOE for *S*</u>. Each component system's effectiveness may depend upon the MOPs of other systems in the system of systems as well as how many of each type. Implications of this point are further discussed below and are illustrated by the example in Chapter 5.

- Each system's MOPs are constrained by low performance threshold specification values, $\mathbf{p}_i^*$, and realistic technology limitations at the high performance end, resulting in the following upper and lower bound constraints: $\mathbf{p}_i^L \le \mathbf{p}_i \le \mathbf{p}_i^U$, or $p_{ij}^L \le p_{ij} \le p_{ij}^U, \forall j$. Note that for some parameters, such as navigation accuracy, small values are better than large values, hence $\mathbf{p}_i^*$ is not simply the lower bound, $\mathbf{p}_i^L$. In the most general case, these MOP constraints could be functions of time as well, in anticipation of requirements creep and advancing technology.
- Each system's unit cost is a (possibly nonlinear) function of performance, expressed in terms of its critical MOPs: $c_i = \mathrm{h}_i(\mathbf{p}_i)$, $\mathbf{c} = \{c_1, \ldots, c_n\}$. We denote $c_i^* = \mathrm{h}_i(\mathbf{p}_i^*)$ as the cost associated with the threshold system. This performance based cost model (PBCM) is generated by considering each critical MOP as a cost driver of a particular subsystem, whose cost can be parameterized on that MOP. (Clearly, more complex cost models representing various aspects of life cycle costs could be formulated, but this form is sufficiently complex to demonstrate feasibility.) Total system of systems' cost is then: $C = \mathbf{mc}^T$.

The system of systems has one overarching performance metric, *E*, a function of each system's overall MOE and the number of systems: $E = \mathrm{g}(\mathbf{m}, E_1, \ldots, E_n)$. If any E_i depends upon not just $\mathbf{p}_i$ but some elements of $\mathbf{p}_j$ where $i \ne j$ then the system of systems is interdependent. So in general, *E* will turn out to be complicated function of the full set of component systems' MOPs: $E = \mathrm{G}\left(\mathbf{m}, \mathbf{p}_1, \ldots, \mathbf{p}_n\right)$. That is, some systems' performance impacts other systems' performance.

When describing a system of systems comprised of relatively simple component systems, or utilizing simplified models of complex systems, each $\mathrm{f}_i\left(\mathbf{m}, \mathbf{p}_1, \ldots, \mathbf{p}_n\right)$ can be expressed in closed form. The simplified (but realistic) mine countermeasures example in Chapter 5 develops a closed form, nonlinear expression for *E*, which is intuitive and quite useful. However, MOPs are themselves typically sensitive to scenarios, concepts of operation (CONOPs), and environments. So in order to obtain representative, robust, full fidelity, results it will generally be necessary to utilize a simulation to evaluate each $\mathrm{f}_{\mathrm{i},}$ and of course, G.

The usual approach when considering upgrades and/or new systems is to treat them individually, perhaps never even defining the full system of systems to which it belongs. That is, define one overarching MOE (or attribute) for the component system and evaluate alternatives in accordance with a cost/MOE ratio. In some cases, multi-attribute evaluations are conducted, (with a weighted factor representing impacts on the larger system) but this is not widely done in DoD systems Cost Effectiveness and Analysis (COEA) studies. This individualized approach is equivalent to assuming that all component systems' MOEs are independent of each other. Under that assumption, maximizing overall effectiveness, *E,* is a matter of maximizing each individual system's effectiveness, perhaps subject to an overall constraint on cost. However, even this approach of constraining overall system of systems' cost is not widely done, as the tendency is to handle each system and its constraints individually, which can lead to poor overall decisions from the system of systems perspective—especially with limited resources.

It should also be noted that the set of MOEs appropriate for this system of systems analysis, $\left\{E_i\right\}$, are not necessarily the same MOEs that are appropriate for evaluating each system apart from the full system of systems. A single system evaluation attempts to reflect requirements on the system's performance levels by attempting to combine all critical design MOPs and measures of value to the larger system of systems into one decision metric. When considering the full system of systems as proposed here, the individual systems MOE set, $\left\{E_i\right\}$, should be formulated specifically to represent each system's contribution to the overarching quantitative effectiveness measure, *E*. Looking ahead to the optimization algorithm, it will not be necessary to express each E_i separately, but it is good to do so for later insight as to what the particular (or marginal) contribution of each system is to the overarching MOE.

In addition to the constraints on measures of performance shown above, several other constraint types can occur and should be considered:

- <u>"Force structure" constraints.</u> There is generally a practical operational or programmatic limitation as to how many systems of each type can comprise the system of systems, known as "force structure" constraints: $\mathbf{m}^L \le \mathbf{m} \le \mathbf{m}^{\mathbf{U}}$, or $m_i^L \le m_i \le m_i^L, \forall i$.
- <u>System effectiveness constraints.</u> In a similar fashion to the MOP constraints, there is generally a minimum threshold for each system's measure of effectiveness (MOE). This can be generated through a technical performance

analysis or (more likely) because the existing component system performs at the threshold level and it is desired to meet or exceed that level. Therefore, the threshold MOE for each system, S_i , is: $E_i^* = \mathrm{f}_i\left(\mathbf{m}^L, \mathbf{p}_1^*, \ldots, \mathbf{p}_n^*\right) \leq E_i, \ \forall i.$ When trying to minimize cost subject to performance constraints, there should be a minimum overall system of systems MOE constraint as well: $E^L \leq E$.

- <u>Cost constraints.</u> When applicable, there can be cost constraints on individual systems as well as the full system of systems: $C \leq C^U$ and $\mathbf{c} \leq \mathbf{c}^U$. Implicitly, $\mathbf{c}$ is also bounded below due to the presence of minimum performance thresholds as discussed above. Hence, we have: $\mathbf{c}^* \leq \mathbf{c} \leq \mathbf{c}^U$. For the purposes of this study, we will take the system of systems viewpoint, and consider only the cost constraint at the macro level, $C \leq C^U$.
- <u>Secondary MOE as quality constraint.</u> As will be illustrated by the MCM example in Chapter 5, it may be necessary to specify a secondary overall MOE for the system of systems which will act as a quality constraint. This can be necessary in the case where the overall MOE is time to complete the system of systems functionality. To ensure that the function is completed to a minimum performance threshold it is necessary to add the secondary MOE as a quality constraint: $\mathrm{q}(\mathbf{m}, \mathbf{p_1}, \ldots, \mathbf{p_n}) \geq \mathrm{q}_\mathrm{T}$.

Definition of system of systems upgrade options is accomplished through control of the constraint set. For example, if only certain component systems are candidates for upgrades (sometimes referred to as “advanced technology insertion”), then their parameters can be constrained to current values—in this manner, the analysis retains full consideration of their influence on the overall system of systems while still investigating upgrade options and their effects on the stable system components as well. The option of replacing some number of existing component systems with advanced systems of the same type while retaining some of the existing systems can be expressed by fixing m_i and defining a new system, S_{i+1}, of essentially the same type but with its MOPs and cost allowed to be variable.

There are two general cases of interest in when considering upgrading (or designing) an existing system of systems, summarized below:

Case 1: Maximize $S = \{S_1, \ldots, S_n\}$ system of systems performance subject to force level, technology, cost, and performance threshold constraints:

Max $E = \mathrm{g}(\mathbf{m}, E_1, \ldots, E_n) = \mathrm{G}(\mathbf{m}, \mathbf{p}_1, \ldots, \mathbf{p}_n)$ where $E_i = \mathrm{f}_i(\mathbf{m}, \mathbf{p}_1, \ldots, \mathbf{p}_n)$, subject to:

$\mathbf{m}^L \leq \mathbf{m} \leq \mathbf{m}^{\mathbf{U}}$

$\mathbf{p}_i^L \leq \mathbf{p}_i \leq \mathbf{p}_i^U$ and $E_i^* \leq E_i, \forall i$ where $E_i^* = \mathrm{f}\,_i(\mathbf{m}^L, \mathbf{p}_1^*, \ldots, \mathbf{p}_n^*)$.

$C \leq C^U$ and $\mathbf{c} \leq \mathbf{c}^U$

[$\mathrm{q}(\mathbf{m}, \mathbf{p}_1, \ldots, \mathbf{p}_n) \geq \mathrm{q}_{\mathrm{T}}$, potential secondary MOE performance threshold function constraint]

Case 2: Minimize $S = \{S_1, \ldots, S_n\}$ system of systems cost subject to performance, force level, technology, and individual systems cost constraints:

Min $C = \mathbf{mc}^T$, subject to:

$E^L \leq E$ where $E = \mathrm{g}(\mathbf{m}, E_1, \ldots, E_n) = \mathrm{G}(\mathbf{m}, \mathbf{p}_1, \ldots, \mathbf{p}_n)$ and $E_i = \mathrm{f}_i(\mathbf{m}, \mathbf{p}_1, \ldots, \mathbf{p}_n)$,

$\mathbf{m}^L \leq \mathbf{m} \leq \mathbf{m}^{\mathbf{U}}$

$\mathbf{p}_i^L \leq \mathbf{p}_i \leq \mathbf{p}_i^U$ and $E_i^* \leq E_i, \forall i$ where $E_i^* = \mathrm{f}_i(\mathbf{m}^L, \mathbf{p}_1^*, \ldots, \mathbf{p}_n^*)$.

[$\mathrm{q}(\mathbf{m}, \mathbf{p}_1, \ldots, \mathbf{p}_n) \geq \mathrm{q}_\mathrm{T}$, potential secondary MOE performance threshold function constraint]

When addressing the system of systems upgrade from the CAIV perspective, we would optimize a sequence of Case 1 problems formed by discretely parameterizing the system of systems cost constraint. This is accomplised by defining a sequence of upper cost bounds, $C_k^U = \mathrm{costfactor}_\mathrm{k} \cdot C^*$, where C^* is the cost to produce the threshold system of systems defined by the parameter set, $\{\mathbf{m}^L, \mathbf{p}_1^*, \ldots, \mathbf{p}_n^*\}$.

CHAPTER 5

MINE COUNTERMEASURES SYTEM OF SYSTEMS PERFORMANCE MODEL

This appendix describes a simplified, but realistic model of naval mine countermeasures (MCM) operations and systems. This limited system of systems consists of two systems: a minefield reconnaissance system and a separate, mine neutralization system. The reconnaissance system first conducts a reconnaissance survey of the entire suspected minefield area, attempting to detect, classify, and localize mine-like objects. These contacts are then passed to the neutralization system, which must re-acquire the contacts and neutralize each mine-like object, if necessary (that is, if the mine-like object is indeed identified as an actual mine). System descriptions, functionality, measures of effectiveness, measures of performance, and PBCM are provided in sufficient detail to support system of system upgrade decisions and trade-off analyses. Before formulating the system of system's model as defined in Chapter 4, we first define the following mine countermeasures analysis terminology:

α	Desired MCM area clearance rate: at least 100α percent of the mines have been cleared.
A	Reconnaissance system area coverage rate during detection pass (nm^2/day)
d_{mines}	Average distance between mines (yards)
F_0	Number of false targets contained in the MCM area, $S_{minefield}$
λ	Minefield density (mines/nm^2)
λ_{ft}	False target (non-mine minelike object) density (objects/nm^2);
M_0	Number of mines originally laid in the MCM area, $S_{minefield}$
P_{α}	Probability that the MCM area will be cleared to the desired minefield clearance rate, α.
p	Mine clearance probability: probability that a mine randomly placed in the MCM area will be cleared.
P_c	Probability of correctly classifying a detection as mine-like or nonmine-like, at range R_c

P_d	Detection probability at range R_d
P_{fa}	Detection false alarm rate, (false alarms/nm^2)
P_{ID}	Probability of correct ID; P_{ID} =1 assumed for military minefields
P_L	Localization (or re-acquisition) probability
P_n	Neutralization probability; P_n =1 assumed for these operations
R_c	Minelike object classification range (yards)
R_d	Target detection range (yards)
R_r	Range at which S_2 has an 80% chance of re-acquiring S_1's detections
σ	Standard deviation of minelike object localization error (yards)
$S_{minefield}$	Area to be searched (nm^2), referred to as the MCM area
T_c	Time required to classify a mine (min)
T_{class}	Time required to classify all detections within $S_{minefield}$ (hours)
T_{cf}	Time required to classify a non-mine (min)
T_{detect}	Time required to complete detection pass through $S_{minefield}$ (hours)
T_n	Time spent neutralizing (prosecuting) a classified mine (min)
T_{pf}	Time spent prosecuting a non-mine classified as a mine (min) or time spent unsuccessfully attempting to re-acquire a correctly classified mine.
$V_{transit}$	Reconnaissance system speed during detection and transit (knots)
V_{class}	Reconnaissance system speed during classification operations (knots)

The overarching MOE, *E*, for this MCM system of systems, *S*, is the time required to achieve a specified MCM area clearance rate, α, with specified confidence level, β. Knowing the form of *E* guides our performance model formulation for the component systems, S_1 and S_2. For the purposes of this analysis, we assume there is only one system of each type, therefore n=2 and $\mathbf{m} = \{1,1\}$.

For the purposes of analysis, we will need to specify the mission scenario and minefield that is to be cleared. The examples used in the analyses to follow will assume an MCM area of 20 nm^2, seeded with 100 mines, corresponding to $S_{minefield}$=20 and M_0=100. The mines are laid out in four rows of 25 each, with a 400 yard separation between mines within each row, and 800 yards between the rows. Hence d_{mines}=600 yards. Figure 2 illustrates a minefield layout with these characteristics.

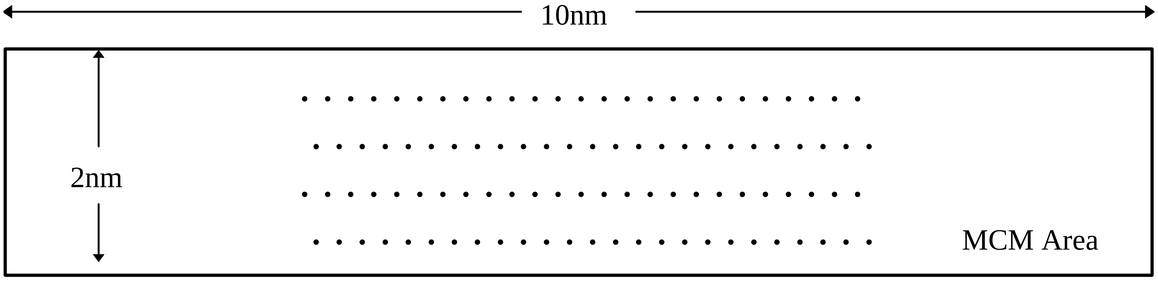


Figure 2: Minefield Layout and Area to be Searched/Cleared

S₁: MCM Reconnaissance System

This system is used to survey a suspected minefield area, performing the typical MCM minehunting functions of *detection, classification, and localization.* It is assumed here that the area is completely covered first with a detection pass. Then, classification (done at much reduced standoff range necessitated by the much higher frequency sensor) of each detected object is attempted. Localization is done concurrently with detection and classification, and therefore takes no additional time. In consideration of the overarching system of systems measure of effectiveness, the MOE for S_1 is then:

$$E_1 = \text{Time (hours) to complete reconnaissance of area } \mathrm{S}_{\text{minefield}}$$
$$\text{given } \lambda,\ \lambda_{ft}, \text{and } \mathrm{M}_0,\ \text{where } \mathrm{M}_0 = \lambda \cdot \mathrm{S}_{\text{minefield}} \text{ and } \mathrm{F}_0 = \lambda_{ft}\mathrm{S}_{\text{minefield}}.$$

The time to complete the detection pass over the area is simply:

$$\mathrm{T}_{\text{detect}} = \frac{24 \cdot \mathrm{S}_{\text{minefield}}}{\mathrm{A}} = \frac{24 \cdot \mathrm{M}_0}{\lambda \cdot \mathrm{A}}.$$

Following the detection pass over the MCM area, the reconnaissance system will its localized contacts and attempt to classify each contact as either mine-like or nonmine-like. (Later, the neutralization system will attempt to re-acquire and neutralize all declared mine-like objects.) In order to calculate time to complete classification, we must know how many detections are expected to be made, and of what type:

$D_m = \mathrm{P_d} \cdot \mathrm{M_0}$ = Number of detected mines

$D_{fa} = \mathrm{P_{fa}} \cdot \mathrm{S_{minefield}}$ = Number of mine false alarms

$D_{ft} = \mathrm{P_d} \cdot \mathrm{F_0}$ = Number of false targets detected

To generate expressions for $\mathrm{T_c}$ and $\mathrm{T_{cf}}$, we must assume a specific classification concept of operations (CONOP). If we assume that S_1 takes the shortest route between contact locations and then executes a semi-circle of radius $\mathrm{R_c}$ about the contact location, then an approximate expression for the time to classify is:

$$\mathrm{T_c} = \frac{60 \cdot \mathrm{d_{mine}}}{2000 \cdot \mathrm{V_{transit}}} + \frac{60 \cdot \pi \mathrm{R_c}}{2000 \cdot \mathrm{V_{class}}}$$

What about time spent attempting to classify a target which is in reality a false alarm? Lets assume that the CONOP would be to execute a full circle about the contact location in the event that the first classification pass was unsuccessful during the first half-circle maneuver. The time required to travel to the contact and execute the full circle is then:

$$\begin{aligned} \mathrm{T_{cf}} &= \frac{60 \cdot \mathrm{d_{mine}}}{2000 \cdot \mathrm{V_{transit}}} + \frac{60 \cdot 2\pi \mathrm{R_c}}{2000 \cdot \mathrm{V_{class}}} \\ &= 2\mathrm{T_c} - \frac{60 \cdot \mathrm{d_{mine}}}{2000 \cdot \mathrm{V_{transit}}} \end{aligned}$$

As will be seen in the Performance Based Cost Model (PBCM) formulation later in this chapter, this formulation for $\mathrm{T_{cf}}$ keeps it independent of cost drivers for the classification sonar performance, which reduces the number of MOPs necessary in the optimization problem, as the terms $\mathrm{d}_{\mathrm{mine}}$ and $\mathrm{V}_{\mathrm{transit}}$ will be considered as fixed for the scenario. The time (hours) required to classify all detections is then:

$$\begin{aligned}
\mathrm{T}_{\mathrm{class}} &= \frac{1}{60}\left[\mathrm{P_c T_c} D_m + (1-\mathrm{P_c})\mathrm{T_{cf}} D_m + \mathrm{T_{cf}}\left(D_{fa} + D_{ft}\right)\right] \\
&= \frac{1}{60}\left[\mathrm{P_c T_c P_d M_0} + (1-\mathrm{P_c})\left(2\mathrm{T_c} - \frac{\mathrm{d_{mine}}}{2000\cdot \mathrm{V_{transit}}}\right)\mathrm{P_d M_0} + \left(2\mathrm{T_c} - \frac{\mathrm{d_{mine}}}{2000\cdot \mathrm{V_{transit}}}\right)\left(\mathrm{P_{fa} S_{minefield}} + \mathrm{P_d F_0}\right)\right]
\end{aligned}$$

and we can now formulate the system measure of effectiveness:

$$E_1 = \frac{24\cdot \mathrm{M_0}}{\lambda\cdot \mathrm{A}} + \frac{1}{60}\left[\mathrm{P_c T_c P_d M_0} + (1-\mathrm{P_c})\mathrm{T_{cf} P_d M_0} + \mathrm{T_{cf}}\left(\mathrm{P_{fa} S_{minefield}} + \mathrm{P_d F_0}\right)\right]$$

$$\text{where}\ \mathrm{T_{cf}} = \left(2\mathrm{T_c} - \frac{60\cdot \mathrm{d_{mine}}}{2000\cdot \mathrm{V_{transit}}}\right).$$

Under the assumptions stated above, we can now list the minimum set of measures of performance that are necessary to formulate an expression for *E_1* as well as describe performance parameters necessary to formulate *E_2*.

$\mathbf{p}_1 = \left\{p_{1,1},\ldots,p_{1,5}\right\}$, hence r_1 =5.

$p_{1,1} = \mathrm{A} = \dfrac{2\cdot \mathrm{R_d}\ \cdot \mathrm{V}}{2000}$ [This expression represents a two-sided detection sonar. A typical approximation is that for a particular sonar/target/environment set, $\mathrm{R_d}$ is determined by fixing $\mathrm{P_d}$ and $\mathrm{V_{transit}}$. For the PBCM and analysis, we assume $\mathrm{P_d}$=0.90 and $\mathrm{V_{transit}}$=7 knots.]

$p_{1,2} = \mathrm{P_c}$ [For this analysis, the sidescan sonar's $\mathrm{P_c}$ is determined at fixed classification range, in this case, we set $\mathrm{R_c}$=70 yards.]

$p_{1,3} = \mathrm{P_{fa}}$

$p_{1,4} = \mathrm{T_c}$

$p_{1,5} = \sigma$ [The localization accuracy is a critical parameter for re-acquisition, a major function of *S₂*. As a simplification, we have chosen to neglect its effect on *S₁*'s re-acquisition during the classification pass, because the re-acquisition would be done with the identical sensor suite that performed the initial detections.]

Therefore, the final form of the MOE for S_1 as a function of the MOP vector is:

$$E_1(\mathbf{p}_1) = \frac{24 \cdot \mathrm{M}_0}{\lambda \cdot \mathrm{A}} + \frac{1}{60}\left[\mathrm{P_c T_c P_d M_0} + (1-\mathrm{P_c})\mathrm{T_{cf} P_d M_0} + \mathrm{T_{cf}}\left(\mathrm{P_{fa} S_{minefield}} + \mathrm{P_d F_0}\right)\right]$$

$$= \frac{24 \cdot \mathrm{S_{minefield}}}{\mathrm{A}} + \frac{1}{60}\left[\mathrm{P_c T_c P_d} \lambda \mathrm{S_{minefield}} + (1-\mathrm{P_c})\mathrm{T_{cf} P_d} \lambda \mathrm{S_{minefield}} + \mathrm{T_{cf}}\left(\mathrm{P_{fa} S_{minefield}} + \mathrm{P_d} \lambda_{ft} \mathrm{S_{minefield}}\right)\right]$$

$$= \mathrm{S_{minefield}}\left[\frac{24}{p_{1,1}} + \frac{1}{60}\left[\lambda \cdot p_{1,2}\, p_{1,4} \mathrm{P_d} + \left(2 p_{1,4} - \mathrm{T_{transit}}\right)\left((1 - p_{1,2})\mathrm{P_d}\lambda + p_{1,3} + \mathrm{P_d}\lambda_{ft}\right)\right]\right]$$

where $\mathrm{T_{transit}} = \left(\dfrac{\mathrm{d_{mine}}}{2000 \cdot \mathrm{V_{transit}}}\right)$.

Note that this MOE does not reflect the quality to which the reconnaissance is accomplished, only how long it takes. If we were considering the effectiveness of the stand-alone reconnaissance system, then we would want to have E_1 reflect the other MOPs as well, in order to effect a measure of "minefield characterization". Reconnaissance survey quality will be reflected in E_2, via expressions that utilize all the elements of $\mathbf{p}_1$ that affect initialization of the neutralization function provided by S_2. Additionally, a minimum threshold quality constraint at the system of systems level will also be imposed.

<u>S_2: MCM Neutralization System</u>

The MCM neutralization system attempts to *re-locate*, *identify, and neutralize* all minelike objects detected and classified as such by the reconnaissance system. For the purposes of this analysis, the effects of identification and subsequent neutralization are ignored, and we will focus on re-acquisition of all minelike objects passed to S_2 from S_1 as contacts. In consideration of the overarching system of systems measure of effectiveness, the MOE for S_2 is:

E_2 = Time (hours) to complete neutralization and neutralization attempts on all contacts/objects classified as mine-like by the reconnaissance system, S_1.

Clearly, E_2 will depend upon how many objects of what type are detected and subsequently classified as minelike objects by S_1. Since the neutralization system will attempt to neutralize all declared mine-like objects, it is important to know how many such objects are expected. The following describes the expected results of the combined detection and classification activities of S_1:

$$C_m = D_m \cdot \mathrm{P_c} = \mathrm{P_d} \cdot \mathrm{P_c} \cdot \mathrm{M_0}$$

Number of mines correctly classified as mine-like

$$\begin{aligned} C_f &= \left(D_{fa} + D_{ft}\right) \cdot \left(1 - \mathrm{P_c}\right) \\ &= \left(\mathrm{P_{fa}} \cdot \mathrm{S_{minefield}} + \mathrm{P_d} \cdot \mathrm{F_0}\right) \cdot \left(1 - \mathrm{P_c}\right) \end{aligned}$$

Number of non-mines incorrectly classified as mine-like

E_2 can now be formulated using the following measures of performance:

$\mathbf{p}_2 = \left\{ p_{2,1}, \ldots, p_{2,3} \right\}$, hence r_2 =3.

$p_{2,1}$ = R_r, the contact localization error standoff which yields an 80% probability of re-acquisition.

$p_{2,2} = \mathrm{T}_\mathrm{pf}$

$p_{2,3} = \mathrm{T}_\mathrm{n}$.

We model the probability of re-acquisition (a.k.a. localization) as: $\mathrm{P}_\mathrm{L} = \mathrm{e}^{\frac{-\sigma}{4.481\mathrm{R}_\mathrm{r}}} = \mathrm{e}^{\frac{-p_{1,5}}{4.481 p_{2,1}}}$, which yields P_L=0.80 when $\mathrm{R}_\mathrm{r} = \sigma$. This model assumes an exponential decay depending upon localization accuracy, an S_1 MOP, and re-acquisition capability of the neutralization system, an MOP for S_2. Dependence of P_L on R_r is illustrated in Figure 3. Also indicated is the feasibility region generated by the upper and lower bounds for R_r, which correspond to technology and threshold system limitations presented in this chapter's PBCM section.

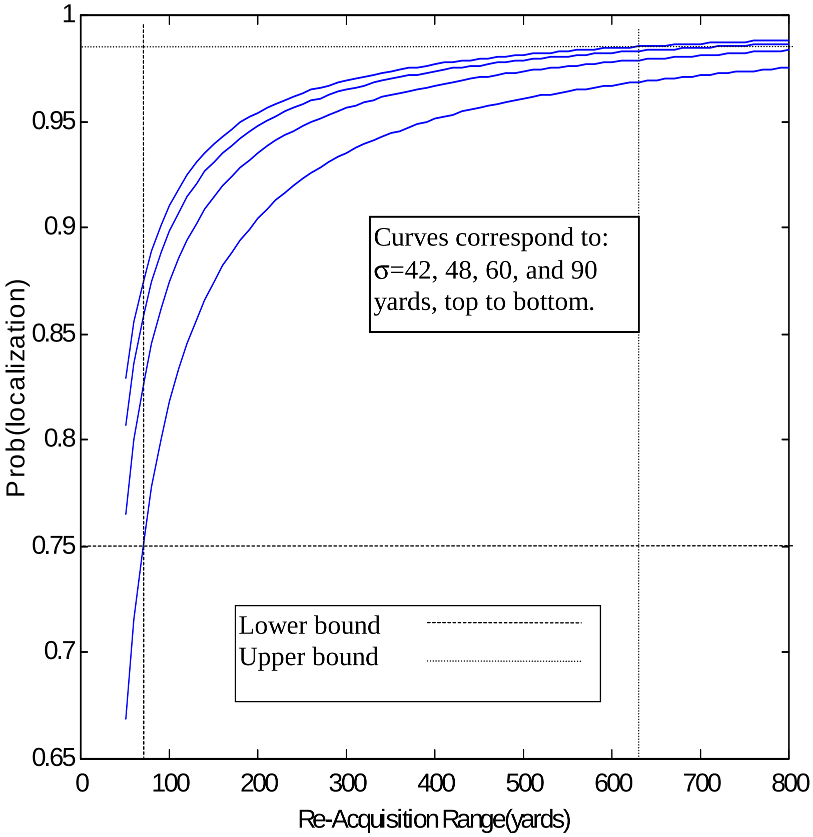


Figure 3 Probability of Localization as a Function of Re-Acquisition Range

Therefore,

E_2 = {time to successfully re-acquire and neutralize minelike objects}

+ {time spent in unsuccessful attempts to re-acquire MLOs}

+ {time spent prosecuting non-minelike objects classified incorrectly)

$$
\begin{aligned}
E_2 &= \mathrm{f}_2\left(\mathbf{p}_1, \mathbf{p}_2\right) \\
&= \frac{1}{60}\left[\mathrm{P_L} C_m \mathrm{T_n} + \left(1 - \mathrm{P_L}\right) C_m \mathrm{T_{pf}} + C_f \mathrm{T_{pf}}\right] \\
&= \frac{1}{60}\left[\mathrm{P_L}\mathrm{P_d}\, p_{1,2} \mathrm{M}_0\, p_{2,3} + \left(1 - \mathrm{P_L}\right)\mathrm{P_d}\, p_{1,2} \mathrm{M}_0\, p_{2,2} + \left(1 - p_{1,2}\right)\left(p_{1,3} \mathrm{S_{minefield}} + \mathrm{P_d} \mathrm{F}_0\right) p_{2,2}\right] \\
&= \frac{\mathrm{S_{minefield}}}{60}\left[\mathrm{P_d}\, p_{1,2}\, p_{2,3} \lambda \mathrm{e}^{\frac{-p_{1,5}}{4.481 p_{2,1}}} + \left(1 - \mathrm{e}^{\frac{-p_{1,5}}{4.481 p_{2,1}}}\right)\mathrm{P_d}\, p_{1,2}\, p_{2,2} \lambda + \left(1 - p_{1,2}\right)\left(p_{1,3} + \mathrm{P_d} \lambda_{ft}\right) p_{2,2}\right]
\end{aligned}
$$

S: MCM Clearance System of Systems

For the full system of systems, the overarching MOE is then simply the total time to complete clearance operations:

$$E = \mathrm{G}(\mathbf{m}, \mathbf{p}_1, \mathbf{p}_2) = E_1 + E_2$$

However, an overall performance or quality constraint must be imposed on the clearance operations, otherwise the optimization will find a very fast yet ineffective system of systems. Specifically, this constraint specifies an MCM area clearance rate, α, with an associated confidence level, β. This should actually be considered as a secondary quality MOE that has a threshold requirement. Recall that p is to be the probability that a particular mine will be cleared. It is clear that:

$$p = \mathrm{P_d}\mathrm{P_c}\mathrm{P_L} = \mathrm{P_d}\, p_{1,2}\, \mathrm{e}^{\frac{-p_{1,5}}{4.481 p_{2,1}}}$$

and the expected number of mines successfully cleared is $p\mathrm{M}_0$.

Then the probability that the number of cleared mines, $\mathrm{M_c}$, is at least $\alpha\mathrm{M}_0$, is called the clearance confidence level, P_α, and binomially distributed as follows:

$$\mathrm{P}_{\alpha} = \mathrm{P}\left(\mathrm{M}_{\mathrm{c}} \geq \alpha \mathrm{M}_0\right) = \sum_{i=\alpha M_0}^{M_0} \left(C_i^{M_0} p^i \left(1-p\right)^{M_0-i}\right)$$

For large M_0 this can be approximated with the normal distribution in accordance with DeMoivre's theorem as follows:

$$\mathrm{P}_{\alpha} \cong \mathrm{P}\left(X \geq \frac{\alpha \mathrm{M}_0 - p\mathrm{M}_0}{\sqrt{\mathrm{M}_0 p(1-p)}}\right) = 1 - \Phi\left(\frac{\alpha \mathrm{M}_0 - p\mathrm{M}_0}{\sqrt{\mathrm{M}_0 p(1-p)}}\right),$$

where Φ is the cumulative probability function for $X \sim N(0,1)$. This is intuitively clear, for in the case where $p=\alpha=0.95$, $\mathrm{P}_{\alpha}=0.5$, indicating that we would then have a 50-50 chance of clearing 95% of the mines. P_{α} rapidly rises as a decreases. For example, with $\alpha=0.90$, $p=0.95$, and $\mathrm{M}_0=100$, then $\mathrm{P}_{.90}=0.971$. Figure 4 illustrates the behavior of P_{α} for a selection of values for p.

The threshold performance constraint at the system of systems level is then:

$$\mathrm{P}_{\alpha} \geq \beta, \text{ or } 1 - \Phi\left(\frac{\alpha \mathrm{M}_0 - p\mathrm{M}_0}{\sqrt{\mathrm{M}_0 p(1-p)}}\right) \geq \beta.$$

Unfortunately, this added constraint at the system of systems level is nonlinear, affecting our choice of optimization algorithms (see Chapter 6). As an illustration with realistic values, choose $\alpha=0.90$, $\mathrm{M}_0=100$, and $\beta=0.95$. In other words, with 100 mines present, we want to be <u>at least</u> 95% confident that <u>at least</u> 90 mines will be cleared:

$$0.95 \leq 1 - \Phi\left(\frac{90-100p}{\sqrt{100p(1-p)}}\right).$$

Then, $\Phi\left(\frac{90-100p}{\sqrt{100p(1-p)}}\right) \geq 0.05$, or $\left(\frac{90-100p}{\sqrt{100p(1-p)}}\right) \leq -1.65$. This implies that

$p \geq 0.9394$, or $p = \mathrm{P_d P_c P_L} = \mathrm{P_d}\, p_{1,3}\, \mathrm{e}^{\frac{-p_{1,5}}{4.481 p_{2,1}}} \geq 0.9394$. Unfortunately, looking ahead to the PBCM, the system MOPs under consideration will not support this level of performance (recall that $\mathrm{P_d}$ will be fixed at 0.90). Relaxation of this constraint consistent with α=0.80, $\mathrm{M_0}$=100, and β=0.90 will yield a constraint that $p \geq 0.846$. Therefore, with with 100 mines present, we will be satisfied to be <u>at least</u> 90% confident that <u>at least</u> 80 mines will be cleared. The practical constraint we will use is therefore:

$$\mathrm{q}(\mathbf{p_1}, \mathbf{p_2}) = p = \mathrm{P_d P_c P_L} = \mathrm{P_d}\, p_{1,2}\, \mathrm{e}^{\frac{-p_{1,5}}{4.481 p_{2,1}}} \geq 0.846$$

Figure 5 is a parameter dependency diagram (PDD) of the full system of systems. The PDD illustrates the interdependence of the two systems' parameters and how they map to the overall system of systems. Also illustrated is the dependence of the secondary quality MOE that constitutes a constraint on overall system of systems performance.

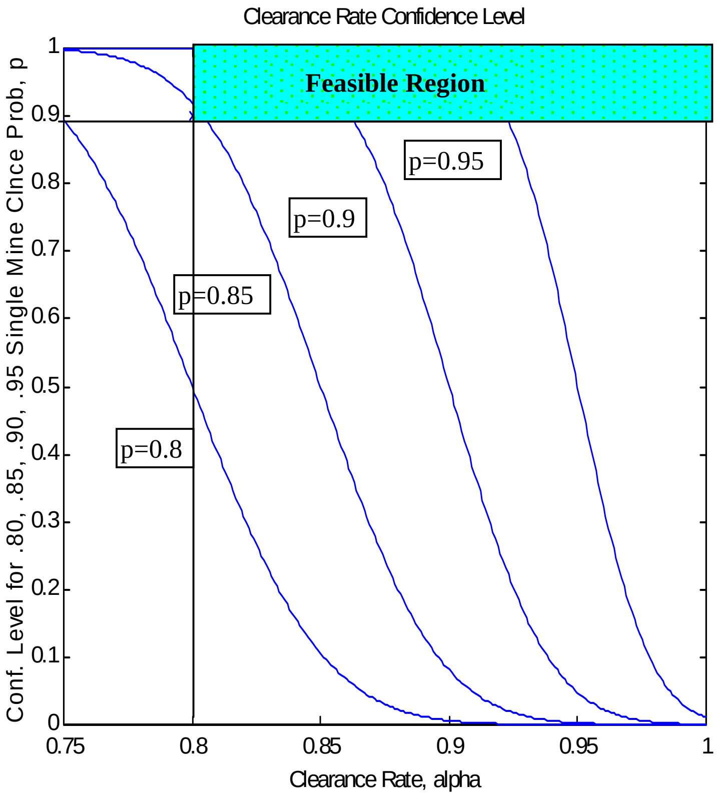


Figure 4: Clearance Confidence Level as a Function of Required Clearance Rate

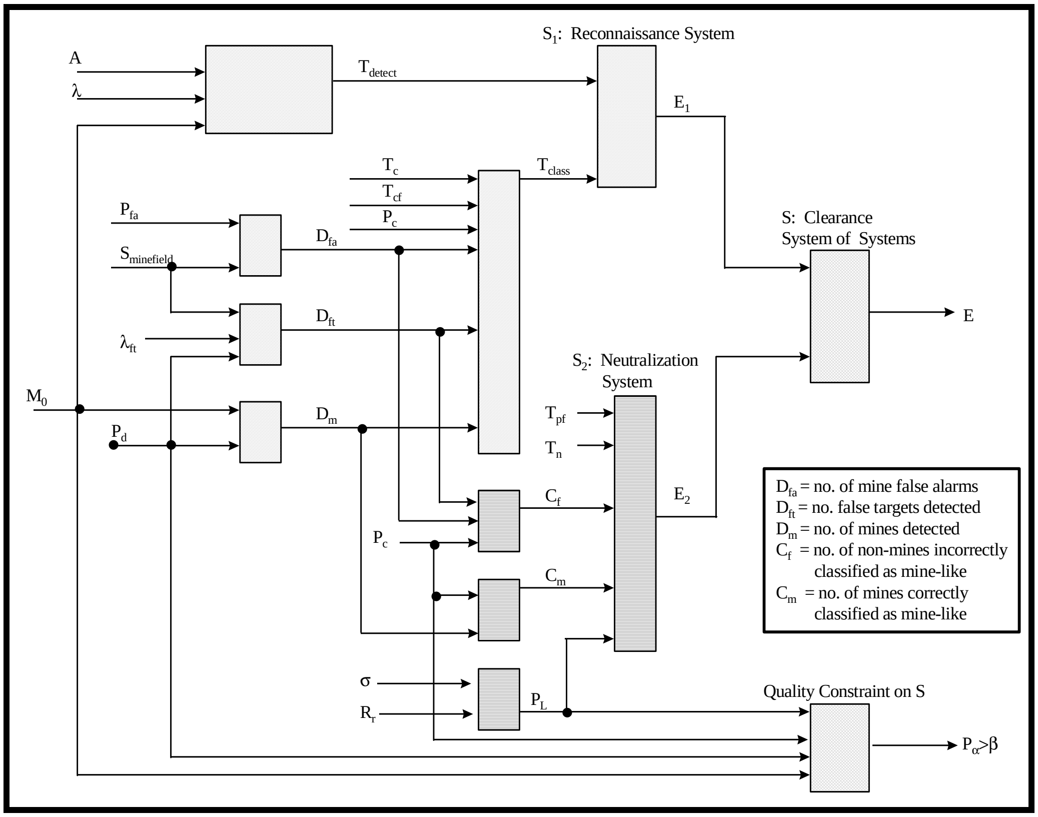

S1: Reconnaissance System
A
λ
T_detect
E1
T_c
T_cf
P_c
T_class
P_fa
S_minefield
D_fa
S: Clearance System of Systems
D_ft
E
λ_ft
S2: Neutralization System
M0
P_d
D_m
T_pf
T_n
C_f
E2
P_c
C_m
D_fa = no. of mine false alarms
D_ft = no. false targets detected
D_m = no. of mines detected
C_f = no. of non-mines incorrectly classified as mine-like
C_m = no. of mines correctly classified as mine-like
σ
R_r
P_L
Quality Constraint on S
P_α>β

Figure 5: Mine Clearance System of Systems Parameter Dependency Diagram

α	Desired MCM area clearance rate: at least 100α percent of the mines have been cleared.
A	Reconnaissance system area coverage rate during detection pass (nm^2/day)
d_{mines}	Average distance between mines (yards)
F_0	Number of false targets contained in the MCM area, $S_{minefield}$
λ	Minefield density (mines/nm^2)
λ_{ft}	False target (non-mine minelike object) density (objects/nm^2);
M_0	Number of mines originally laid in the MCM area, $S_{minefield}$
P_{α}	Probability that the MCM area will be cleared to the desired minefield clearance rate, α.
p	Mine clearance probability; i.e., probability that a mine in the MCM area will be cleared.
P_c	Probability of correctly classifying a detection as mine-like or nonmine-like, at range R_c
P_d	Detection probability at range R_d
P_{fa}	Detection false alarm rate, (false alarms/nm^2)
P_{ID}	Probability of correct ID; P_{ID} =1 assumed for military minefields
P_L	Localization (or re-acquisition) probability
P_n	Neutralization probability; P_n =1 assumed for these operations
R_c	Minelike object classification range (yards)
R_d	Target detection range (yards)
R_r	Range at which S_2 has an 80% chance of re-acquiring S_1's detections
σ	Standard deviation of minelike object localization error (yards)
$S_{minefield}$	Area to be searched (nm^2), referred to as the MCM area
T_c	Time required to classify a mine (min)
T_{class}	Time required to classify all detections within the search area, $S_{minefield}$ (hours)
T_{cf}	Time required to classify a non-mine (min)
T_{detect}	Time required to complete detection pass through search area, $S_{minefield}$ (hours)
T_n	Time spent neutralizing (prosecuting) a classified mine (min)
T_{pf}	Time spent unsuccessfully attempting to re-acquire a detection (min)
$V_{transit}$	Reconnaissance system speed during detection and transit (knots)
V_{class}	Reconnaissance system speed during classification operations (knots)

Performance Based Cost Model (PBCM) and Parameter Bounds

The reconnaissance system performance ranges and cost modeling are derived from design considerations for the U.S. Navy's Long-Term Mine Reconnaissance System (LMRS), a submarine-based autonomous undersea vehicle (UUV)[18]. The neutralization system performance ranges and cost models are based upon a combination of LMRS factors, certain U.S. Navy operational MCM systems, and commercial off-the-shelf (COTS) information regarding marine navigation systems. Due to classification and data availability issues, considerable license has been taken in developing the PBCM, with the primary intent to provide sufficient complexity and realism to show feasibility of the methodology.

MOPs developed earlier in this chapter are now grouped by the major sub-system to which they act as major cost drivers. The PBCM provides an approximation of subsystem cost as a function of those primary subsystem MOPs. Moreover, since this type of MCM system would be produced in very small numbers, only developmental costs are considered, neglecting the full system life cycle costs. Commercial off-the-shelf (COTS) or non-developmental item (NDI) technologies are also assumed so that developmental costs approximate R&D and production costs combined. The subsystem and associated MOPs are as illustrated in Figure 6. Costs are then added together to get total cost. The cost constraint indicated in Figure 15 is parameterized by "costfactor" which is a factor on the threshold system costs indicating the maximum amount the

[18] Benedict. J. R., (1996). Final Report: Long-Term Mine Reconnaissance System (LMRS) Cost and Operational Effectiveness Analysis (COEA), Johns Hopkins University Applied Physics Laboratory Report NWA-96-009, September 1996.

decisionmaker is willing to consider spending. In this way, we will consider a series of optimization problems that will provide insight from the CAIV perspective.

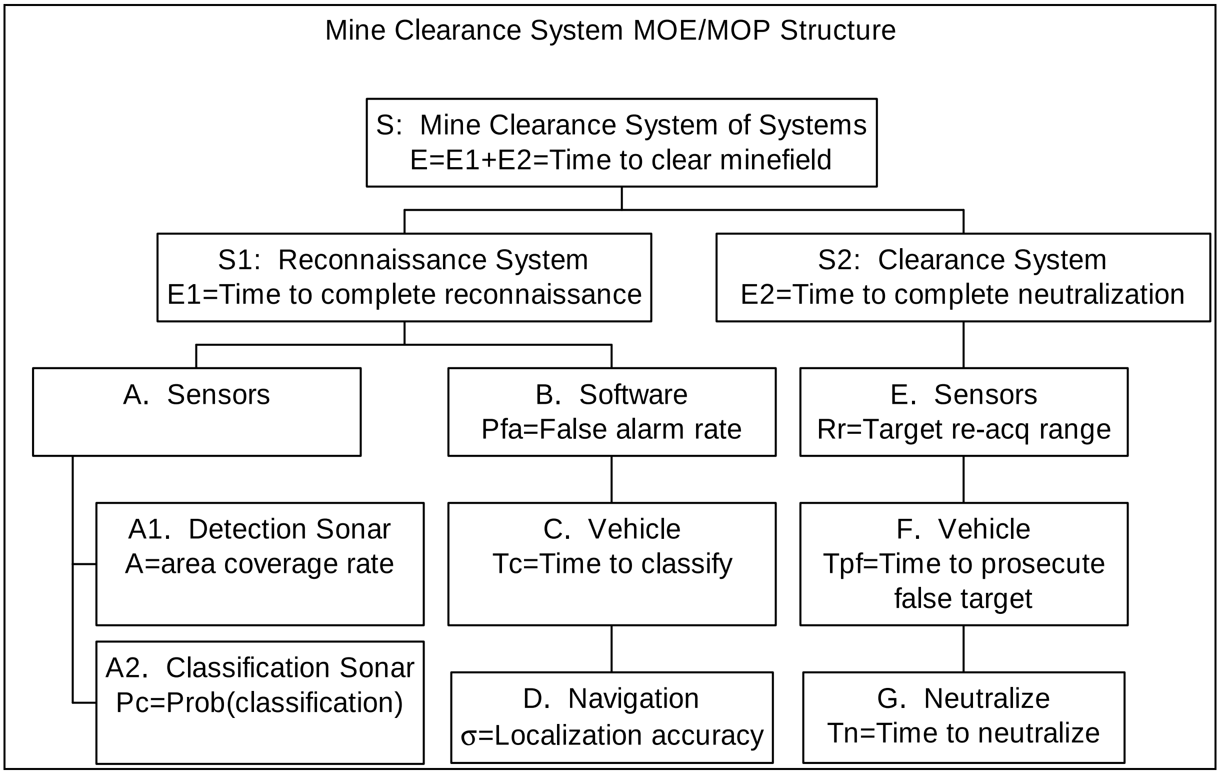


Figure 6 Mine Clearance System of Systems MOE/MOP Structure

A. System S_1: Sensors

There are two sonars in the sensor subsystem: detection and classification sonars.

A1. Detection Sonar

Critical parameters for search sonars are probability of detection, range, maximum vehicle speed at which the sonars can remain effective due to flow noise. They are of course sensitive to several environmental parameters as well as assumed target characteristics. The approach here is to assume one environment, fix P_d at 0.90, speed at 7 knots, and utilize the modeled results in Benedict[18] to derive the following PBCM for the overall MOP, area coverage rate, A (nm^2/day). The following table represents the PBCM:

A (nm^2/day)	10	57	82	94
Cost ($M)	3	4.483	7.655	11.445

The following routine fits a third order polynomial to the data, and will form the cost model:

```
%  PBCM for A1, Search Sonars
x=[10,57,82,94]
y=[3,4.483,7.655,11.445]
p=polyfit(x,y,3)
x2=10:.1:100;
y2=polyval(p,x2);
plot(x,y,'o',x2,y2)
xlabel('Area Coverage Rate (nm^2/day)')
ylabel('Cost ($M)')

» a1cost
x =    10   57      82     94
y =     3   4.483   7.655  11.445
p =  4.5034e-005   -0.0053861      0.21593       1.3342
```

Therefore, $p_{1,1}^{L} = 10$, $p_{1,1}^{U} = 100$, $p_{1,1}^{*} = 10$, and

$$\mathrm{h}_{1,1}\left(p_{1,1}\right) = (4.5034\mathrm{e}-005)\, p_{1,1}{}^{3} \ - \ 0.0053861 p_{1,1}{}^{2} \ + \ 0.21593\, p_{1,1} \ + 1.3342\,.$$

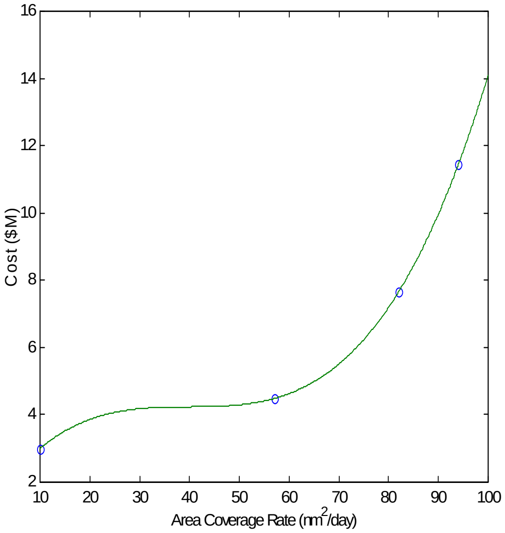


Figure 7: PBCM for Area Coverage Rate

A2. Classification Sonar

The cost driving critical parameter for sidescan classification search sonars is its probability of classification at a given range, in this case chosen to be 70 yards. Again, vehicle speed (flow noise), environmental parameters, and assumed target characteristics are essential to making effective P_c predictions. The approach here is to assume one environment, the required classification range (important because of vehicle vulnerability to the mines), adequate vehicle control or motion compensation, and utilize the modeled results from Benedict[18]. While some detection sonars may be considered as non-developmental items (NDI), many sidescan classification sonars are now truly COTS, which have substantially lower cost. Vehicle integration costs are considered in that "subsystem". The following table represents the PBCM:

P_c at 70 yards	0.9	0.93	0.96	0.98
Cost ($M)	0.2	0.5	1.4	2.2

The following routine fits a second order polynomial to the data, and will form the cost model:

```
%  PBCM for A2, Classify Sonars
figure
x=[0.9,0.93,0.96,0.98]
y=[.2,.5,1.4,2.2]
p=polyfit(x,y,2)
x2=0.9:.01:1.0;
y2=polyval(p,x2);
plot(x,y,'o',x2,y2)
xlabel('Classification Probability')
ylabel('Cost ($M)')
```

```
» a2cost
x =    0.9000    0.9300    0.9600    0.9800
y =    0.2000    0.5000    1.4000    2.2000
p =  283.4646 -507.6378  227.4598
```

Therefore, $p^L_{1,2} = 0.9$, $p^U_{1,2} = 0.98$, $p^*_{1,2} = 0.9$, and

$$\mathrm{h}_{1,2}\left(p_{1,2}\right) = 283.4646 p_{1,2}{}^2 - 507.63784 p_{1,2} + 227.4598 .$$

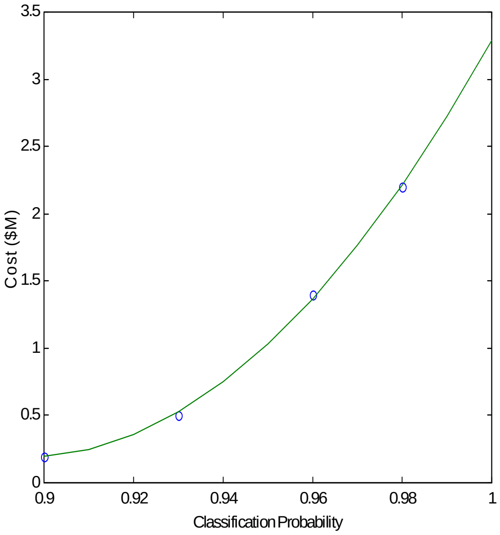


Figure 8: PBCM for Classification Probability

B. System S_1: Software

Modern MCM sonar systems are incorporating Computer-Aided Detection and Computer-Aided Classification (CAD/CAC) processors in order to keep false alarm rates low while maintaining high probability of detection/classification. Consequently, requirements on false alarm rates are the cost driver on the software subsystem. The following table represents the PBCM:

P_{fa} (#false alarms/nm^2)	2.0	1.0	0.5	0.25
Cost ($M)	8.0	10.319	13.592	16.429

The following routine fits a third order polynomial to the data, and will form the cost model:

```
%  PBCM for B, Software in support of FAR, or CAD/CAC.
x=[2,1,0.5,0.25]
y=[8,10.319,13.592,16.429]
p=polyfit(x,y,3)
x2=0.0:0.01:2;
y2=polyval(p,x2);
```

```
plot(x,y,'o',x2,y2)
xlabel('False Alarm Rate (#/nm^2)')
ylabel('Cost ($M)')

» bcost
x =           2           1         0.5        0.25
y =           8      10.319      13.592      16.429
p =     -2.0484      9.9873     -17.942      20.322
```

Therefore, $p_{1,3}^{L} = 0.25$, $p_{1,3}^{U} = 2.0$, $p_{1,3}^{*} = 2.0$, and

$$\mathrm{h}_{1,3}\left(p_{1,3}\right) = -2.0484\, p_{1,3}{}^{3} + 9.9873 p_{1,3}{}^{2} - 17.942\, p_{1,3} + 20.322\,.$$

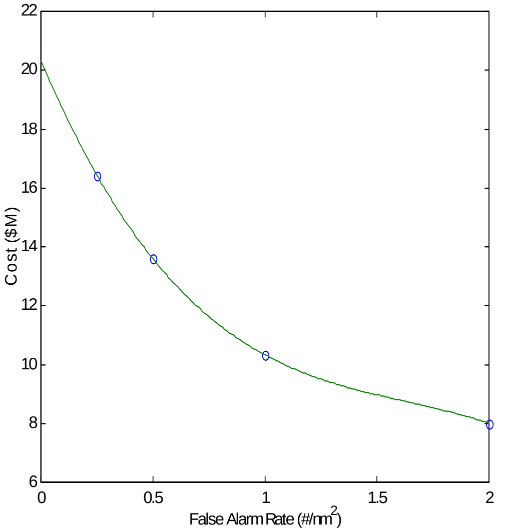


Figure 9: PBCM for False Alarm Rate

C. System S_1: Vehicle (Propulsion, Energy, Quieting, Margin, Integration)

As mentioned earlier in this Chapter, to generate expressions for T_c and T_{cf}, we must assume a specific classification CONOP. If we assume that S_1 takes the shortest route between contact locations and then executes a semi-circle of radius R_c about the contact location in an attempt to image the target, then an approximate expression for the time to classify is:

$$T_c = \frac{60 \cdot d_{mine}}{2000 \cdot V_{transit}} + \frac{60 \cdot \pi R_c}{2000 \cdot V_{class}} \; .$$

The time to classify is therefore a direct function of the maximum speed at which the vehicle can conduct the classification maneuver while holding the platform steady and quiet. We are assuming that this is a cost driver for a combination of subsystems, to include propulsion, quieting, margin, and integration; all of which we have cost estimates. With the nominal parameters as mentioned above, the following table represents the discrete data points from which the PBCM is derived, as a function of classification speed:

V_{class} (knots)	1	3	5	7
T_c (minutes)	9.17	4.77	389	3.513
Cost ($M)	5.0	7.574	8.190	9.191

The following MATLAB® code fits a second order polynomial to the data points for use in the cost constraints.

```
%  PBCM for C, Vehicle in support of time to classify.
x=[3.513,3.89,4.77,9.17]
y=[9.191,8.190,7.574,5.0]
p=polyfit(x,y,2)
x2=3.0:0.1:9.5;
y2=polyval(p,x2);
plot(x,y,'o',x2,y2)
xlabel('Time to Classify (min)')
ylabel('Cost ($M)')
```

```
» ccost
x =      3.513      3.89      4.77      9.17
y =      9.191      8.19      7.574        5
p =    0.11597    -2.1757     15.20.
```

Therefore, $p_{1,4}^{L} = 3.0$, $p_{1,4}^{U} = 9.17$, $p_{1,4}^{*} = 9.17$, and

$$h_{1,4}\left(p_{1,4}\right) = 0.11597\, p_{1,4}{}^{2} - 2.1757\, p_{1,4} + 15.204 .$$

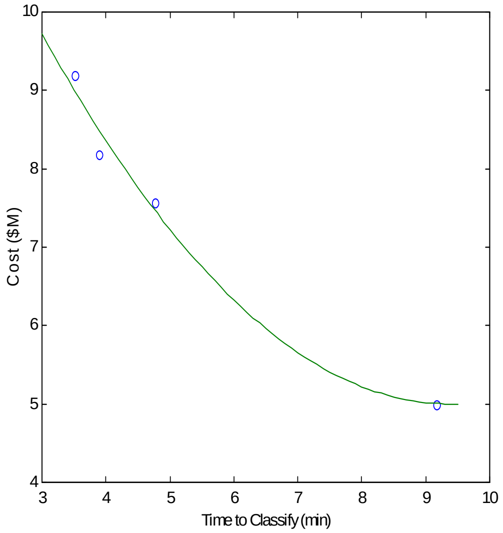


Figure 10: PBCM for Time to Classify

D. System S_1: Navigation

Maritime navigation systems are now truly COTS, and therefore low cost relative to other subsystems (note that costs are in $K, not $M). It is also difficult to sort out competing claims of accuracy, as performance is closely related to the CONOPS. Typical non-GPS, underwater maritime systems utilize an inertial navigation system augmented with a Doppler sonar velocity log (DSVL). The DSVL performance dominates over the long term, and needs to be updated periodically with a higher quality source, such as GPS. To obtain sufficiently high accuracy to enable re-acquisition of targets, we select a 30 nm run between GPS updates. The following table lists DSVL performance in terms of percent of distance traveled, derived one-sigma geodetic accuracy, and system costs[19]:

[19] Brokloff, N.A., maritime navigation systems specialist. Personal communication, June 1997.

Percent of Distance Traveled	0.15	0.1	0.08	0.07
σ (yards)	90	60	48	42
Cost ($K)	50	250	450	500

The following routine fits a second-order polynomial to the PBCM data above:

```
%  PBCM for D, Navigation subsystem in support of localization accuracy.
x=[90,60,48,42]
y=[0.050,0.250,0.450,0.550]
p=polyfit(x,y,2)
x2=40:1:90;
y2=polyval(p,x2);
plot(x,y,'o',x2,y2)
xlabel('One sigma accuracy (yards)')
ylabel('Cost ($M)')

» dcost
x =    90    60    48    42
y =         0.05         0.25         0.45         0.55
p =   0.00020618     -0.03776       1.7778
```

Therefore, $p_{1,5}^{L} = 42$, $p_{1,5}^{U} = 90$, $p_{1,5}^{*} = 90$, and

$$h_{1,5}\left(p_{1,5}\right) = (2.0618\mathrm{e} - 004)\, p_{1,5}{}^{2} - 0.03776\, p_{1,5} + 1.7778\,.$$

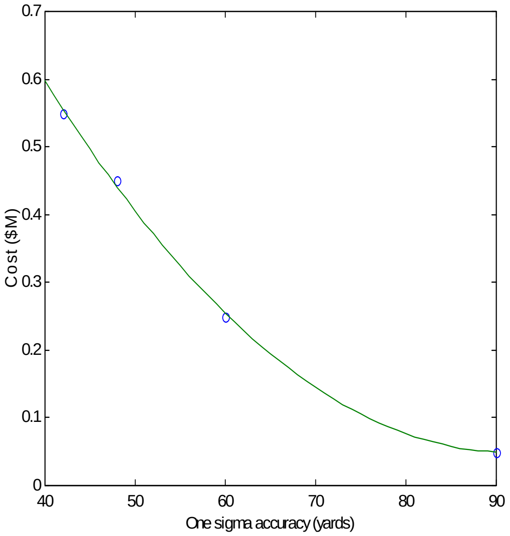


Figure 11: PBCM for Navigation Accuracy

E. System S_2: Sensors/Sonars

The neutralization system returns to the site localized by the reconnaissance system and attempts to re-acquire the target for neutralization activities. Depending on the quality of the localization activity, we should be able to get away with a much less expensive sonar than the long-range detection sonar used for reconnaissance to enhance the coverage rate. As in the case of navigation systems, these medium-performance sonars are available as NDI or COTS. One way to express the cost driving parameter is the range at which a reasonably high probability of re-acquiring the target can be assured, say 80%. Earlier in this chapter, a model for the interrelationship of localization accuracy, re-acquisition range, and probability of re-acquiring (localization) was developed under the assumption that the closer S_2 can be directed to the target location, the higher the resultant probability of reacquisition. Utilizing data from Benedict, the following PBCM data points are derived:

R_r (yards)	75	129	457	622
Cost ($M)	1.5	3.0	4.8	7.655

The following routine fits a third order polynomial to the data, and will form the cost model:

```
%  PBCM for E, Ahead looking sonar for re-acquisition
x=[129,457,622,75]
y=[3,4.8,7.655,1.5]
p=polyfit(x,y,3)
x2=75:5:675;
y2=polyval(p,x2);
plot(x,y,'o',x2,y2)
xlabel('Re-Acquisition Range (yards)')
ylabel('Cost ($M)')

» ecost
x =   129   457   622    75
y =     3          4.8         7.655           1.5
p =  1.5049e-007  -0.00015782     0.055167      -1.8133
```

Therefore, $p_{2,1}^{L} = 75$, $p_{2,1}^{U} = 700$, $p_{2,1}^{*} = 75$, and

$$\mathrm{h}_{2,1}\left(p_{2,1}\right) = (1.5049\mathrm{e}-007)\, p_{2,1}{}^{3} - (1.5782\mathrm{e}-004)\, p_{2,1}{}^{2} + 0.055167\, p_{2,1} - 1.8133\,.$$

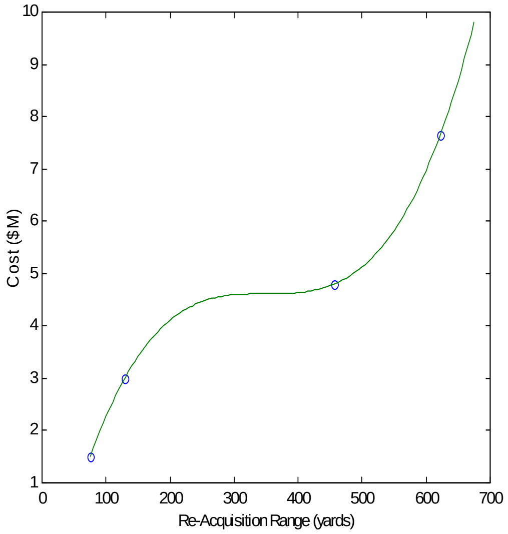


Figure 12: PBCM for Re-Acquisition Range

F. System S_2: Vehicle (Propulsion, Energy, Quieting, Margin, Integration)

As in the PBCM for classification time for system S_1, T_{pf} (time to prosecute a false target) is the cost driver for vehicle maneuverability required to cover the area around the localized false target holding the vehicle steady while providing the re-acquisition sonar multiple aspect angles on the target in order to confidently make a false target call. The CONOP we choose for this is to make a full circle of radius 70 yards about the indicated (from S_1) location, if the usual direct approach re-acquisition attempt is unsuccessful. This CONOP and associate PBCM has two important implications for how we view T_{pf} and subsequently, T_n.

- It ignores S_2 transit time to the target's designated location. This will be the same for all contacts provided by S_1 and shorter transit times would be a byproduct of improvements made to vehicle maneuverability driven by T_{pf}.

- It also ignores the true dependence of T_{pf} and T_n on the S_1 localization accuracy, σ. We have chosen to look at σ's direct effect on probability of localization, P_L only. Although this could be modeled in a more sophisticated simulation, it would be an equivalent bias on all three major additive terms of E_2, and therefore can be neglected for this analysis. Hence our formulation of T_{pf} and T_n are just the additional time it would take to make a non-contact call or neutralize the target, over and above the time required for nominal re-acquisition.

The following table lists vehicle costs as driven by false target prosecution performance:

Speed during maneuver	2	3	4	5	10
T_{pf} (min)	6.6	4.4	3.3	2.64	1.32
Cost ($M)	5.0	7.5	8.19	9.191	16.621

The following routine fits a third-order polynomial to the PBCM data above:

```
%  PBCM for F, Vehicle in support of time required to verify a false
target
x=[6.6,4.4,3.3,2.64,1.32]
y=[5.0,7.5,8.190,9.191,16.621]
p=polyfit(x,y,3)
x2=1.0:0.1:7;
y2=polyval(p,x2);
plot(x,y,'o',x2,y2)
xlabel('Time to Prosecute False Target(min)')
ylabel('Cost ($M)')

» fcost
x =  6.6          4.4          3.3         2.64         1.32
y =   5          7.5         8.19        9.191       16.621
p =     -0.28504       3.8462      -17.264       33.344
```

Therefore, $p_{2,2}^{L} = 1.0$, $p_{2,2}^{U} = 7.0$, $p_{2,2}^{*} = 6.6$, and

$$\mathrm{h}_{2,2}\left(p_{2,2}\right) = -0.28504\,p_{2,2}{}^{3} + 3.8462\,p_{2,2}{}^{2} - 17.264\,p_{2,2} + 33.344\,.$$

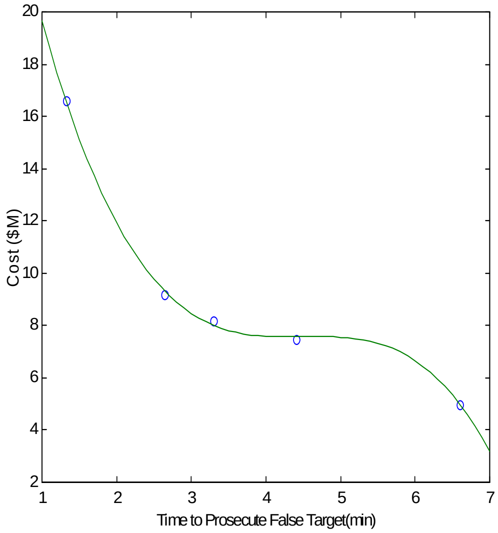


Figure 13: PBCM for Time to Prosecute False Target

G. System S_2: Neutralization

The neutralization threshold system performance and costs are patterned after the U.S. Navy's Mine Neutralization System (AN/SLQ-48) in which integrates a sonar, optical, and bomblet subsystems to re-acquire, identify, and neutralize targets passed to it from other systems. Although it takes time to launch, acquire, ID/neutralize, and recover this remotely operated vehicle, the ID/neutralize phase takes approximately 10 minutes. Unit cost for the AN/SLQ-48, which has been in production for some time, is down to approximately $500K. Assuming that the cost driver for the AN/SLQ-48 (which used NDI sonars/optics) was the neutralization component, and using a 10:1 rule of thumb for developmental vs. production costs, the threshold system development cost for a system was estimated as shown in the following table:

T_n (min)	10	8	7	5	3
Cost ($M)	5.3	6.0	7.0	10.0	15.0

The following routine fits a second-order polynomial to the PBCM data above:

```
%  PBCM for G, Neutralization subsystem
x=[10,8,7,5,3]
y=[5.3,6,7,10,15]
p=polyfit(x,y,2)
x2=3.0:0.1:10;
y2=polyval(p,x2);
plot(x,y,'o',x2,y2)
xlabel('Time to Neutralize (min)')
ylabel('Cost ($M)')
```

```
» gcost
x =    10     8     7     5     3
y =    5.3            6            7           10           15
p =    0.21024      -4.1096       25.397
```

Therefore, $p_{2,3}^{L} = 3.0$, $p_{2,3}^{U} = 10.0$, $p_{2,3}^{*} = 10.0$, and

$$\mathrm{h}_{2,3}\left(p_{2,3}\right) = \; 0.21024\, p_{2,3}{}^{2} - \; 4.1096\, p_{2,3} + 25.397\,.$$

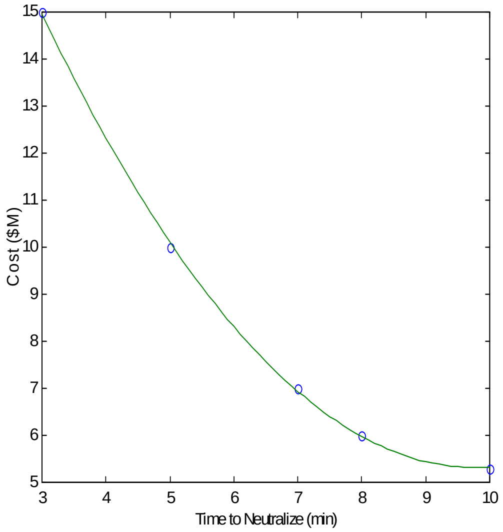


Figure 14: PBCM for Time to Neutralize

MCM System of Systems Model Summary

Maximize $S = \{S_1, \ldots, S_n\}$ system of systems performance (minimize time) subject to technology, cost, and performance threshold constraints:

Minimize:

$$E(\mathbf{p}_1, \mathbf{p}_2) = E_1(\mathbf{p}_1) + E_2(\mathbf{p}_1, \mathbf{p}_2)$$

$$= \frac{\mathrm{S}_{\text{minefield}}}{60}\left[\frac{24 \cdot 60}{p_{1,1}} + \lambda \cdot p_{1,2}\, p_{1,4} \mathrm{P}_{\mathrm{d}} + \left(2 p_{1,4} - \mathrm{T}_{\text{transit}}\right)\left((1 - p_{1,2})\mathrm{P}_{\mathrm{d}} \lambda + p_{1,3} + \mathrm{P}_{\mathrm{d}} \lambda_{ft}\right)\right]$$

$$+ \frac{\mathrm{S}_{\text{minefield}}}{60}\left[\mathrm{P}_{\mathrm{d}}\, p_{1,2}\, p_{2,3} \lambda\, \mathrm{e}^{\frac{-p_{1,5}}{4.481 p_{2,1}}} + \left(1 - \mathrm{e}^{\frac{-p_{1,5}}{4.481 p_{2,1}}}\right) \mathrm{P}_{\mathrm{d}}\, p_{1,2}\, p_{2,2} \lambda + (1 - p_{1,2})(p_{1,3} + \mathrm{P}_{\mathrm{d}} \lambda_{ft}) p_{2,2}\right]$$

subject to:

$$(3,0.9,0.25,3.0,42)^T \le \mathbf{p}_1 \le (100,0.98,2.0,9.17,90)^T, (75,1.0,3.0)^T \le \mathbf{p}_2 \le (700,7.0,10.0)^T$$

$$C(\mathbf{p}_1, \mathbf{p}_2) \le C^U = \text{costfactor} \cdot C^* \text{ and } \mathrm{q}(\mathbf{p}_1, \mathbf{p}_1) \ge \mathrm{q}_{\mathrm{T}} = 0.846$$

where:

$$\begin{aligned} C(\mathbf{p}_1, \mathbf{p}_2) = {} & (4.5034\mathrm{e}-005)\, p_{1,1}^{\;3} - 0.0053861 p_{1,1}^{\;2} + 0.21593\, p_{1,1} + 1.3342 \\ & + 283.4646 p_{1,2}^{\;2} - 507.63784 p_{1,2} + 227.4598 \\ & - 2.0484\, p_{1,3}^{\;3} + 9.9873 p_{1,3}^{\;2} - 17.942\, p_{1,3} + 20.322 \\ & + 0.11597 p_{1,4}^{\;2} - 2.1757\, p_{1,4} + 15.204 \\ & + (2.0618\mathrm{e}-004) p_{1,5}^{\;2} - 0.03776\, p_{1,5} + 1.7778 \\ & + (1.5049\mathrm{e}-007)\, p_{2,1}^{\;3} - (1.5782\mathrm{e}-004) p_{2,1}^{\;2} + 0.055167\, p_{2,1} - 1.8133 \\ & + -0.28504\, p_{2,2}^{\;3} + 3.8462 p_{2,2}^{\;2} - 17.264\, p_{2,2} + 33.344 \\ & + 0.21024 p_{2,3}^{\;2} - 4.1096\, p_{2,3} + 25.397 \end{aligned}$$

C*=threshold system cost=28.066

$$\mathrm{q}(\mathbf{p}_1, \mathbf{p}_1) = p = \mathrm{P}_{\mathrm{d}} \mathrm{P}_{\mathrm{c}} \mathrm{P}_{\mathrm{L}} = \mathrm{P}_{\mathrm{d}}\, p_{1,2}\, \mathrm{e}^{\frac{-p_{1,5}}{4.481 p_{2,1}}}$$

Figure 15 Summary of MCM System of Systems Optimization Problem

CHAPTER 6

OPTIMIZATION APPROACH

In this chapter, we investigate some alternative optimization methods that will be applicable to the MCM system of systems upgrade problem formulated in Chapter 5, and reformulate the general system of systems performance model (Chapter 4) to take advantage of the most promising approach.

Applicable Methods

Referring to the discussion in Chapter 4, the general system of systems upgrade optimization problem takes the following form:

Max $E = \mathrm{g}(\mathbf{m}, E_1,\ldots,E_n) = G(\mathbf{m},\mathbf{p_1},\ldots,\mathbf{p_n})$ where $E_i = \mathrm{f}_i(\mathbf{m},\mathbf{p}_1,\ldots,\mathbf{p}_n)$,

and subject to the following constraints:

	number of inequalities:
$\mathbf{0} \le \mathbf{m}^L \le \mathbf{m} \le \mathbf{m}^{\mathbf{U}}$	*n*
$\mathbf{0} \le \mathbf{p}_i^L \le \mathbf{p}_i \le \mathbf{p}_i^U$	*r*
$C \le C^U$ where	*1*
$c_i = \mathrm{h}_i(\mathbf{p}_i)$, $\mathbf{c} = \{c_1,\ldots,c_n\}$, and $C = \mathbf{mc}^T$.	
[$\mathrm{q}(\mathbf{m},\mathbf{p_1},\mathbf{p_2},\ldots,\mathbf{p_n}) \ge \mathrm{q_T}$, a potential secondary MOE threshold function constraint]	*1*

All quantities are real-valued, with the additional constraint that elements of *n*-vector **m** are integers. Each $\mathbf{p}_i$ has dimension r_i and $r = \sum_{i=1}^{n} r_i$.

For the purpose of this study, we will consider the case where there is only one of each type of system in the system of systems. In this case, $\mathbf{m} = (1, \ldots, 1)$, and that **m** is to remain fixed. We can then ignore the difficulty of having to solve for the integer-valued vector, **m,** which would result in an integer programming problem. Then what we have is a nonlinear, multi-dimensional optimization problem with a generally nonlinear constraint set.

We first consider the case in which the objective function, *G*, can be evaluated without error. Then, we can look to the deterministic domain of nonlinear programming and utilize conventional search techniques to solve the particular problem.

For the special case where all the constraints are linear, one of the most common and effective nonlinear programming algorithms is known as the Davidon-Davies method[20] or the Davidon-Fletcher-Powell method with linear constraints[21]. The Davidon-Davies method is a gradient search method that can handle linear equality and inequality constraints. The method projects the current solution estimate along the path of steepest descent until it intersects one (or more) hyperplanes formed by the inequality constraints. Then that constraint is said to be active, it is added to the equality constraint set, and the next iterate is restricted to move along the corresponding new intersecting hyperplane surface until a convergence criterion for a minimum is achieved. Checks are also done at each iteration to consider relaxing an equality constraint that has been established prematurely.

[20] Himmelblau, D.M., (1972). Applied Nonlinear Programming, McGraw-Hill, New York, pp. 261-266.

[21] Walsh, G.R., (1975). Methods of Optimization, John Wiley & Sons, London, pp.155-161.

Some fully general nonlinear programming algorithms restructure the problem or locally linearize the nonlinear constraint set so that this method can be used as the core search algorithm. For example, if there are a small number of nonlinear constraints, it is feasible to augment the objective function with penalty functions (described in more detail later in this chapter) to incorporate the active nonlinear constraints into the objective function so that unconstrained or simple projection methods will then be applicable. This is not generally done because it comes at a cost of ill-conditioning the problem, which must be traded against the gain in algorithm simplicity.

MATLAB® is a widely used, sophisticated mathematical problem-solving commercial software product that utilizes the method of sequential quadratic programming (SQP)[22] to solve the fully general nonlinear programming problem in which both objective and constraint functions can be nonlinear. The particular routine is called "CONSTR" and is contained in the "Optimization Toolbox". Basically, the method formulates a sequence of quadratic programming (QP) subproblems based on a quadratic approximation of the Lagrangian function and by linearizing the nonlinear constraints about the current iterate. The simpler QP subproblem (quadratic function with linear constraints) is solved by using an active set projection method[23] very similar to the Davidon-Davies Method in order to provide a search direction for a line search procedure that provides the next iterate. The original nonlinear function and constraint sets are then approximated about the new iterate and the sequence is repeated until convergence criteria are satisfied.

[22] Gill, P.E., Murray, W., Wright, M.H., (1981). Practical Optimization, Academic Press, London, Chap. 6.
[23] Ibid, Chapter 5.

Appendix A contains specific MATLAB® code necessary to solve the MCM system of systems problem formulation and a simple illustrative problem, while results are shown in Chapter 7. The user can call CONSTR with or without supplying an explicit gradient function. Since our ultimate objective is to utilize a simulation for function evaluations, we are interested in solving the problem without an explicit gradient function. CONSTR will then approximate the gradient at each iteration at the expense of more function evaluations. The most efficient classical gradient approximation is that of forward differencing, an approximation that requires as many function evaluations as the dimensionality, *r*, of the independent variable vector. To enhance search algorithm convergence properties, it is generally advisable to switch over to a central difference formula as the step size reduces near the solution[24], but at the expense of requiring *2r* function evaluations per step.

As mentioned above, obtaining a closed form, deterministic expression for the system of systems' MOE objective function is not always feasible. A growing number of application areas rely on stochastic modeling and simulation to predict system of systems performance under certain conditions of interest. Therefore, future practical implementations of this approach for the system of systems upgrade problem will include utilization of simulation to evaluate the objective function—an extension that will put a premium on minimizing the search algorithm's function evaluations. Simulation will generally be of the Monte Carlo type, hence there will be random error associated with

[24] Fletcher, R., (1981). Practical Methods of Optimization, John Wiley & Sons, Chichester, Scotland, p.108.

the function evaluation. The simulation then produces a corrupted realization of the objective function of the form:

$$y(\mathbf{p_1},\ldots,\mathbf{p_n}) = G(\mathbf{p_1},\ldots,\mathbf{p_n}) + \omega\,,$$

where ω represents simulation noise. This stochastic nature of the function evaluations puts us into the realm of stochastic optimization—to which classical optimization methods are not directly applicable. Moreover, not only will the gradient of y be unavailable explicitly, classical approximations of gradients become extremely costly to compute since each function evaluation represents a simulation run. Until recently, finite-difference-based gradient search stochastic approximation procedures that are adaptations of deterministic algorithms have been most widely used for this type of optimization. A major drawback of these methods is that the number of function evaluations required at each step is linear in the dimension of the search parameter vector.[25] Since we envision eventually using large scale system of systems simulations with tens of parameters, a much more efficient method is desirable. The recently developed Simultaneous Perturbation Stochastic Approximation (SPSA) Method[26] is the most efficient estimator in this domain with respect to function evaluations per iteration, and its first and second order versions have been adapted here for use in solving the MCM system of systems problem described in Chapter 5. Since it will be the central algorithm we use for stochastic optimization here, it is necessary to examine it in some detail.

[25] Glynn, P.W., (1989). "Optimization of Stochastic Systems via Simulation", *Proceedings of the 1989 Winter Simulation Conference*, ed. E.A. MacNair, K.J. Musselman, P. Heidelberger, IEEE, Piscataway, NJ, 90-105.

Constrained Simultaneous Perturbation Stochastic Approximation (SPSA) Method

As described in Fu and Hill[27], the first order SPSA method is a type of gradient search method that requires only two function evaluations per iteration. The current solution estimate is perturbed in all elements simultaneously in a sort of central difference fashion rather than one component at a time which is generally done in order to estimate the partial derivatives that comprise the gradient vector. Now, SPSA per se is an unconstrained optimization algorithm, and very little work has yet to be done to extend it to more commonly occurring constrained problems such as the system of systems upgrade problem structure formulated here. But it has been shown[28] that substituting the SPSA-generated gradient estimate for the gradient estimate in a projection-based constrained steepest descent algorithm results in convergence to a Kuhn-Tucker point—meaning that SPSA can be used to enhance efficiency of such methods as the basic Davidon-Davies method described above. Most of these methods "project" from the current iterate along an estimate of the gradient until they intersect one or more constraint surfaces, which are then "followed" (always reducing the objective function and updating the active constraint set) until convergence criteria are satisfied.

[26] Spall, J.C. (1992), "Multivariate Stochastic Approximation Using a Simultaneous Perturbation Gradient Approximation," IEEE Transactions on Automatic Control, vol. 37, pp. 332-341.

[27] Hill, S.D. and Fu, M.C., (1997). "Optimization of Discrete Event Systems via Simultaneous Perturbation Stochastic Approximation", *Transactions of the Institute of Industrial Engineers, Special Issue on Operations, Engineering, and Simulation, Vol. 29, Issue #3, pp.233-243*.

[28] Sadegh, P. (1997). "Constrained Optimization via Stochastic Approximation with a Simultaneous Perturbation Gradient Approximation", *Automatica*, vol. 33, 1997, pp. 889-892.

Although Sadegh[28] established the theoretical foundation for constrained SPSA, he did not present a workable algorithm nor does he address convergence properties relative to other first order approaches. One option for implementing a general constrained SPSA algorithm is to modify a tried and true optimization code such as MATLAB®'s implementation of the SQP method to embed the SPSA gradient and iterate update equations. This was judged to be impractical for the scope of this dissertation due to code complexity. There are a myriad of complex sub-algorithms necessary to ensure robustness and wide applicability of commercial code. There are also significant issues regarding determination of the active constraint set in the context of a stochastic optimization domain. Indeed, practical implementations of constrained SPSA have been limited to constraint sets of the simple bound type. SPSA has been used to optimize some rather simple constrained problems utilizing discrete event simulations very successfully[27] and more significantly, a 168-dimensional air-traffic control flight delay problem[29]. In order to discuss practical considerations any further, it is necessary to explicitly consider the SPSA algorithm.

The following unconstrained SPSA algorithm description is adapted from its author's implementation guide.[30] The adaptation utilizes the notation developed here for the system of systems upgrade problem and facilitates sharing of code that invokes the MATLAB® CONSTR function mentioned above for efficiency comparison purposes on the deterministic formulation developed in Chapter 5. Note that the method is also

[29] Kleinman, N.L., Hill, S.D., and Ilenda, V.A., (1997). "SPSA/SIMMOD Optimization of Air Traffic Delay Cost", *Transportation Science*, to appear.

[30] Spall, J.C., (1996). "Implementation of the Simultaneous Perturbation Algorithm for Stochastic Optimization", unpublished.

applicable to the deterministic case in which ω=0. As with MATLAB®, the SPSA convention is to minimize the objective function, so that we seek to minimize $-G(\mathbf{p_1},\ldots,\mathbf{p_n})$. Hence we would define $y(\mathbf{p_1},\ldots,\mathbf{p_n}) = -G(\mathbf{p_1},\ldots,\mathbf{p_n}) + \omega$.

Let $\mathbf{x} = \{\mathbf{p_1},\ldots,\mathbf{p_n}\}$, the *r*-dimensional MOP vector for the full system of systems. SPSA iteratively produces a sequence of estimates, $\hat{\mathbf{x}}_0, \hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2, \ldots,$ where $\hat{\mathbf{x}}_k$ will converge to the optimum value as *k* gets large. Implementation steps for the algorithm are as follows:

1. Initialization

   a) Set counter index *k=0.*

   b) Set initial estimate $\hat{\mathbf{x}}_0 = \{\mathbf{p}_1^L,\ldots,\mathbf{p}_n^L\}$, the threshold system of system's MOPs.

   c) Choose the SPSA algorithm gain coefficients, a, c, A, α, γ, in accordance with guidance and experience[25-28].

2. Generate *k-th* iteration's simultaneous perturbation vector.

   a) Generate the *r*-dimensional random vector, Δ_k, where each of the *r* components of Δ_k is generated from a Bernoulli ± 1 distribution with probability 0.5 for each ± 1 outcome.

3. Objective function evaluations during *k-th* iteration.

   Obtain two measurements of the objective function, $y(\mathbf{x})$, as follows:

   a) Let: $c_k = c(k+1)^{-\gamma}$, $a_k = a(A+k+1)^{-\alpha}$

   $$\mathbf{x}_k^+ = \hat{\mathbf{x}}_k + c_k\Delta_k, \quad \mathbf{x}_k^- = \hat{\mathbf{x}}_k - c_k\Delta_k$$

b) Evaluate the objective function twice, based upon the above simultaneous perturbation about the current MOP iterate, $\hat{\mathbf{x}}_k$:

$$y_k^+ = y\left(\mathbf{x}_k^+\right) \text{ and } y_k^- = y\left(\mathbf{x}_k^-\right)$$

4. <u>Gradient approximation during the *k-th* iteration</u>

Compute the simultaneous perturbation approximation to the *r*-dimensional gradient:

$$\hat{g}_k(\hat{\mathbf{x}}_k) = \begin{bmatrix} \dfrac{y_k^+ \text{ - } y_k^-}{2\, c_k\, \Delta_{k1}} \\ \vdots \\ \dfrac{y_k^+ \text{ - } y_k^-}{2\, c_k\, \Delta_{kr}} \end{bmatrix},$$ where Δ_{ki} is the *i-th* component of the Δ_k vector. This gradient approximation contrasts with the component-by-component perturbation in standard forward difference gradient approximations which result in *r* function evaluations at each iteration.

5. <u>Update the MOP iterate, $\hat{\mathbf{x}}_k$</u> .

Utilizing the gain sequence and the gradient approximation, compute:

$\hat{\mathbf{x}}_{k+1}^+ = \hat{\mathbf{x}}_k - a_k \hat{g}\left(\hat{\mathbf{x}}_k\right)$ and $y_{k+1} = y\left(\hat{\mathbf{x}}_{k+1}\right)$.

6. <u>Continue or terminate the search</u>.

Return to Step 2, replacing *k* with *k+1* or terminate if one of the following termination criteria are satisfied:

a) $\left|\hat{\mathbf{x}}_{k+1}^+ - \hat{\mathbf{x}}_k\right| \le x_{tol}$, where x_{tol} is the MOP convergence criterion

b) $\left|y_{k+1} - y_k\right| \le y_{tol}$, where y_{tol} is the MOE convergence criterion

c) *k+1=k_{max}*, the maximum allowable number of iterations.

The above algorithm summary only describes unconstrained SPSA. As mentioned above, Sadegh[28] has extended the original SPSA theoretical convergence results to constrained optimization problems. It was proven that a projection-type adaptation of SPSA will converge to a Kuhn-Tucker point asymptotically when optimizing over nonlinear inequality constraints that form a non-empty, bounded set and the constraint functions are continuously differentiable. A further condition he adds is that function evaluations at points where the constraints are violated are not feasible. By projection algorithm, he means the following. Let $P(\mathbf{x})$ be the projection function to the nearest point to **x** on the constraint set, utilizing the usual Euclidean norm. Then the projection algorithm has the general form

$$\hat{\mathbf{x}}_{k+1}^{+} = P\left(\hat{\mathbf{x}}_k - a_k \hat{g}\left(\hat{\mathbf{x}}_k\right)\right).$$

Intuitively, we "project" from the current iterate $\hat{\mathbf{x}}_k$ to the nearest border of the constraint set. Practically speaking, this means that $\hat{\mathbf{x}}_{k+1}^{+}$ must satisfy exactly one or more of the constraint equations, which are then said to be "active". Once the active set of constraints is determined, the projection algorithm then must consider those constraints as equalities for the purpose of computing $\hat{\mathbf{x}}_{k+1}^{+}$. The active set is modified as iterations proceed, and the effect is that the iterates "follow the border" as the objective function continues to improve. Unfortunately, this projection along the SPSA gradient approximation to the nearest point on the constraint set is not easy to compute for the general case where the constraint set is formed by nonlinear inequalities. Mature, classical optimization codes such as MATLAB®'s CONSTR function contain extensive, robust code to achieve the projection. Published implementations of constrained SPSA to date have avoided such complexities and have been applied only to problems with simple bound constraints of the form $\mathbf{x}^L \le \mathbf{x} \le \mathbf{x}^U$.

Note that our system of systems problem formulation is dominated by these simple bound constraints on MOPs, with the exceptions being the cost constraint and the secondary MOE constraint. In the next section, we will show how the model can be reformulated using penalty functions to eliminate these complex constraints. To finish off our discussion of constrained SPSA, the following three adaptations to the algorithm described above are made:

1. As done in Fu and Hill[25], when computing $\mathbf{x}_k^+ = \hat{\mathbf{x}}_k + c_k \Delta_k$, $\mathbf{x}_k^- = \hat{\mathbf{x}}_k - c_k \Delta_k$, and $\hat{\mathbf{x}}_{k+1}^+ = \hat{\mathbf{x}}_k - a_k \hat{g}\left(\hat{\mathbf{x}}_k\right)$ in Steps 3 and 5 above, project them to the nearest feasible point on the constraint set prior to evaluating the objective function. This is particularly critical for the system of systems upgrade application because many of the MOPs are probabilities or other physical quantities that have no practical meaning outside of the constraint set. In the case of simple bound constraints, this is trivial to implement.
2. As suggested by Spall[31], incorporate a “blocking function” to accelerate and ensure monotonic convergence of the SPSA algorithm. Inspection of the gradient approximation (Step 4) reveals that it should not be expected to be a particularly good approximation to the true objective function gradient in the sense that the more expensive forward or central difference formulae are constructed. Although convergence is guaranteed, the poor quality of the SPSA gradient means that the

[31] Spall, J.C., (1997). Personal communication.

sequence of iterates, $\{\hat{\mathbf{x}}_k\}$, does not necessarily produce a monotonically increasing objective function sequence. The blocking function consists of rejecting an iterate that degrades the objective function by more than a specified small amount and the next iteration is begun with the current iterate as its starting point; i.e., $\hat{\mathbf{x}}_{k+1} = \hat{\mathbf{x}}_k$.

3. When the SPSA iterates get near the solution, they will tend to "jitter" about the optimum. When using a large, fixed number of iterations to obtain a solution, the average of the last several iterates will provide a better estimate of the optimum than simply the last iterate.[32]

A MATLAB® implementation of this constrained SPSA algorithm is included as Appendix B, with results from a simple, illustrative problem—provided only to establish confidence in the author's implementation.

As will be shown in Chapter 7, this first order SPSA algorithm exhibited poor convergence on the MCM system of systems problem. This is attributed to inherent difficulties that a first order process encounters when the problem has large scaling differences in the components being estimated—and we have about two orders of magnitude variation in this problem. Recently, a second order version of SPSA has been developed, which emulates the convergence acceleration and scaling invariant properties of deterministic Newton-Raphson algorithms[33]. This algorithm (called "2SPSA") requires five function evaluations per iteration, but produced much better results than the

[32] Heydon, B., (1997). Personal communication.

[33] Spall, J.C., (1997). "Accelerated Second-Order Stochastic Optimization Using Only Function Measurements", *Proceedings of the 31st Conference on Information Sciences and Systems*, 19-21 March 1997, Baltimore, MD.

first order version (hereafter referred to as "1SPSA"). Results will be discussed in Chapter 7 and 8, with code and detailed results presented in Appendices D-F.

Model Reformulation via Penalty Function Method

As mentioned above, we are motivated to simplify the constraint set of our system of systems upgrade optimization problem to the point where the only constraints remaining are those of the simple bound type. Two classes of methods were considered: Lagrangian multiplier and penalty function methods, which turn out to be very similar in practice. The penalty function approach was selected as most appropriate for this application due to its conceptual simplicity and ease of implementation. Although attractive at first glance, Lagrangian multiplier methods[34] have drawbacks that inhibit a robust, practical implementation. For example, the resulting objective function can be unpredictably unbounded below and the more robust augmented Lagrangian[35] function methods require an outside iterative loop to solve for the Lagrangian multipliers, thereby greatly increasing the complexity and number of function evaluations. A straightforward Lagrangian multiplier method was successfully implemented on the same 3 dimensional constrained test problem described in Appendix D (formulated for SPSA checkout), but would not converge when applied to the full MCM system of systems problem, indicating a modified objective function that is unbounded below—a condition that will arise unpredictably in large dimensional problems.

[34] Cooper, L. and Steinberg, D., (1970), Introduction to Methods of Optimization, W.B. Saunders Company, Philadelphia, PA, pp. 290-295.

[35] Gill, P.E., Murray, W., Wright, M.H., (1981). Practical Optimization, Academic Press, London, Chap. 6

To reformulate the MCM problem using a penalty function, note that both the cost and quality constraints should be active. That is, the optimal solution will turn out to cost the maximum allowable amount, and the performance/quality MOE will be minimally satisfied. (This was borne out in the results using CONSTR, and is a good "sanity check" on the model formulation.) Continuing with the notation established in this section, our optimization problem has the following streamlined form, now assuming equality constraints:

$$\begin{aligned}&\text{Max } G(\mathbf{x}), \text{subject to}:\\&C(\mathbf{x}) = c^U\\&q(\mathbf{x}) = \mathrm{q}_{\mathrm{T}}\\&\mathbf{x}^L \le \mathbf{x} \le \mathbf{x}^U\end{aligned}$$

Following the MATLAB®/SPSA minimization convention and re-ordering:

$$\begin{aligned}&\text{Min - } G(\mathbf{x}), \text{subject to}:\\&C(\mathbf{x}) - c^U = 0\\&q(\mathbf{x}) - \mathrm{q}_{\mathrm{T}} = 0\\&\mathbf{x}^L \le \mathbf{x} \le \mathbf{x}^U\end{aligned}$$

There is a wide variety of practical penalty function methods available[36], most of which utilize variants of the quadratic and absolute value penalty functions:

$$\mathrm{F}_Q(\mathbf{x}) = -\mathrm{G}(\mathbf{x}) + A_1\left(C(\mathbf{x}) - c^U\right)^2 + A_2\left(q(\mathbf{x}) - \mathrm{q}_{\mathrm{T}}\right)^2 \text{ and}$$

$$\mathrm{F}_A(\mathbf{x}) = -\mathrm{G}(\mathbf{x}) + A_1\left|C(\mathbf{x}) - c^U\right| + A_2\left|q(\mathbf{x}) - \mathrm{q}_{\mathrm{T}}\right|.$$

[36] Himmelblau, Chapter 7.

Note that the effect of the penalty function augmentation is to create a function which will have a local minimum that is "near" the constrained optimum of G(**x**) for sufficiently large gains, A_1 and A_2. The penalty terms provide a positive penalty for increased constraint violation, which of course will never be numerically satisfied exactly. In practice, the gains, A_1 and A_2, on the penalty terms must be large enough so that the effective penalty due to the equality constraints is significant relative to the value of the original objective function, G(**x**), at its optimum point. If the gains are too small, then objective function convergence will occur at a point that does not satisfy the constraints very closely. Conversely, if the gains are too large, the algorithm-derived gradient estimates will be extremely large and variable for small changes in the constraint violation penalties, making for an inevitably ill-conditioned problem. This is mitigated by the absolute value penalty function relative to the quadratic penalty function, making it workable for the SPSA algorithm variants which have an inherently poor gradient estimate anyway.

Since the optimum values of the original problem are not generally known, the usual approach is to solve a sequence of problems, with gradually increasing gains until the desired degree of constraint satisfaction is achieved. Another approach is to construct a gain function that increases as a function of the number of iterations, approaching an asymptotic limit. Figure 16 plots an asymptotic penalty function gain function that (starts at 1 and grows to 11 after 500 iterations) that showed some degree of utility for solving the MCM problem with SPSA:

$$\mathrm{F}_A(\mathbf{x}) = -\mathrm{G}(\mathbf{x}) + \alpha_1(k)\left|C(\mathbf{x}) - c^U\right| + \alpha_2(k)\left|q(\mathbf{x}) - \mathrm{q}_\mathrm{T}\right|$$

where:

k = iteration number
$k_{\max}$ = iteration at which full gain is achieved

$$\alpha_{\mathrm{i}}(k) = \begin{cases} \dfrac{A_i}{2}\sin\left[\left(\dfrac{k-1}{k_{\max}}\right)\pi - \dfrac{\pi}{2}\right] + \dfrac{A_i}{2} + 1 & \text{for } k < k_{\max} \\ A_i & \text{for } k \geq k_{\max} \end{cases}$$

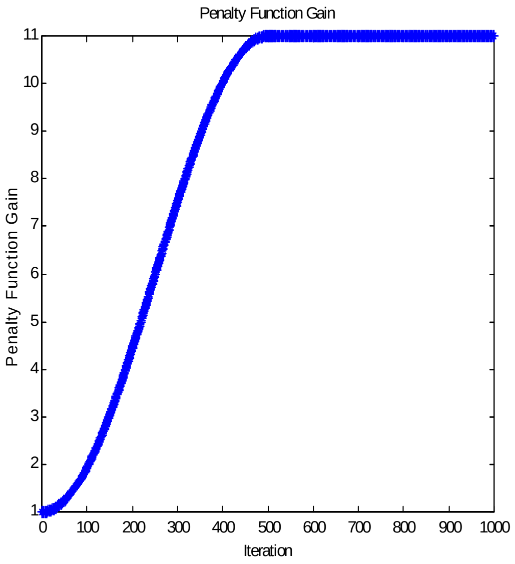


Figure 16   Asymptotic Penalty Gain Function

For the MCM problem we do know the optimum before we start, and can therefore estimate appropriate gains a priori, avoiding the need to solve for the gains in an expensive outer optimization loop. The overall MOE minimum is on the order of 20 hours. Therefore, A_1 and A_2 should be chosen so that their term's contribution to the penalty is on the order of about 10% of the optimal MOE value when their constraints are "close enough". E.g., if we want the cost constraint to be satisfied to approximately 0.1 and the quality constraint to 0.001, then A_1=20 and A_2=2,000 produce approximately 2 hours of penalty each, a sufficient amount to drive an optimization algorithm to satisfy the original constraints when utilizing the absolute value penalty function. Several additional orders of magnitude are necessary to produce a similar effect from the quadratic penalty function.

There are distinct advantages and disadvantages to the two penalty terms above, as indicated in the following table:

	Advantages	**Disadvantages**
Quadratic Penalty Function	Differentiable Initial estimate need not be close to optimum for convergence	Arbitrarily large gains required for equivalence to original problem Adversely affects algorithms with poor gradient estimators.
Absolute Value Penalty Function	Gains need not be arbitrarily large to ensure equivalence to original problem—avoids inevitable ill-conditioning	Not differentiable Weaker penalty leaves local minima so that initial estimate must be close to optimum for convergence

Many strategies of gains, penalty functions, and algorithm parameters were attempted and results catalogued in the course of this study. Results reported in Chapter 7 will reflect the finding that quadratic penalty function worked best with CONSTR and the absolute value penalty function worked best with SPSA, in spite of the theoretical SPSA assumption that the loss function is differentiable. The extremely large gains necessary to ensure the transformed quadratic penalty function formulation is equivalent to the original problem poses severe stability problems for SPSA, which generates a poor gradient estimator in order to conserve function evaluations.

Limitations and Scope of Research

Unfortunately, there are several areas which can prevent efficient solution of specific problems, and therefore limit generalization of results. Some of those potential issues and mitigation approaches are as follows:

- Recall that we had side-stepped the force structure issue, represented by the integer-valued vector **m**. In a domain where the nominal numbers for each system exceeds some moderate number, say 10, then it is probably reasonable to solve the continuous case and take the nearest integer values—keeping in mind that the goal is really to achieve some substantive improvement over the existing system of systems, and the decision-maker will realize that there are sources of error in the process that would dominate this level of approximation. On the other hand, if it is expected that there would be only a small number of each system, say 1-5, then we could put an exhaustive search loop around the constrained optimization algorithm. It is expected that this latter situation is more widely applicable.
- Non-differentiability of the objective function when using simulation can violate convergence proof assumptions of certain algorithms, such as SPSA. Since the simulations are representations of physical systems, they are generally well-behaved as long as functionality is preserved and the optimization is over technical performance measures vice scenario or CONOPS parameters. Use of the absolute

value penalty function creates a non-differentiable objective function, though this did not prevent SPSA convergence in practice. Force level optimization when considering small numbers of systems may also generate discontinuities. However, an exhaustive search method would not require the objective function to be well-behaved. The Taguchi method for obtaining robust designs is a sort of exhaustive search over discrete parameters and seems applicable to the integer-valued force level problem, especially in the face of an ill-behaved objective function.[37,38]

- Local vs. Global optimum. If $E = \mathrm{g}\left(\mathbf{m}, E_1, \ldots, E_n\right) = G(\mathbf{m}, \mathbf{p_1}, \ldots, \mathbf{p_n})$ is a concave function over a convex constraint set and the "optimum" point satisfies the Kuhn-Tucker conditions, then it is a global minimum solution. Unfortunately, these are difficult conditions to verify, especially if we are ultimately dealing with function evaluations provided by simulation or a complicated multivariate expression. Initial formulation of a closed-form analytic model should aid in generating initial conditions that are sufficiently "close" to the optimum. Moreover, if we keep in mind what we are trying to accomplish—the upgrading of a system of systems—then we should not be terribly concerned that our solution might not be at a global maximum, but rather be "satisficed" to find an upgrade suite that represents some significant improvement over the present situation as indicated by the objective function improvement over the threshold or nominal system of systems MOE. Convergence to a local vice global maximum did indeed occur for the MCM system of systems, and is

[37] Lee, K-H, Eom, I-S, Park, G-J, Lee, W-I, (1996). "Robust Design for Unconstrained Optimization Problems Using the Taguchi Method", *AIAA Journal*, Vol.34, No.5, 1059-1063.

[38] Phadke, M.S., (1989). Quality Engineering Using Robust Design, Prentice-Hall, Englewood Cliffs, NJ

discussed in Chapter 7. In Chapter 8, a strategy for generating good initial MOP estimates is postulated and demonstrated for the simulation case.

- Choosing test problems with interesting initial conditions and constraint sets. It is possible that an ill-conceived problem formulation could yield search results that consistently gravitate towards just one or two parameters. To demonstrate the method, an attempt has been made to create the MCM system of systems model that basically already "in balance". As will be seen in some of the results in Chapters 7 and 8, this has the drawback of creating a rather "shallow" objective function, making it a difficult problem for optimization algorithms.

The severity of the issues mentioned above will vary from problem to problem, and of course become more challenging as we migrate towards larger, more realistic problems. Therefore, the plan of research was to work from the well-understood deterministic domain towards the stochastic optimization domain motivated by the desire to utilize M&S for system of systems MOE evaluations. The sequence of optimizations that will be discussed in Chapters 7 and 8 for the MCM problem are:

1. Constrained SQP Optimization
2. Single system vs. system of systems optimization
3. Constrained SQP optimization via penalty function methods
4. First order constrained SPSA optimization via penalty function method
5. Second order constrained SPSA optimization via penalty function method
6. Second order constrained SPSA optimization via penalty function and simulation

CHAPTER 7

PHASE I RESULTS: CLOSED FORM OBJECTIVE FUNCTION

This Chapter presents the results of optimizing the closed form representation of the MCM system of systems model developed in Chapter 5 with the sequence of methods as described in Chapter 6. For ease of reference, the MCM system of system MOP definitions are repeated here:

$x(1) = p_{1,1} = A =$ System S1 area coverage rate during detection pass (nm^2 / day)

$x(2) = p_{1,2} = \mathrm{P_c} =$ Probability of correctly classifying a detection as mine - like
or nonmine like, at range $\mathrm{R_c}$

$x(3) = p_{1,3} = \mathrm{P_{fa}} =$ Detection false alarm rate (false alarms / nm^2)

$x(4) = p_{1,4} = \mathrm{T_c} =$ Time required to classify a mine (min)

$x(5) = p_{1,5} = \sigma =$ Standard deviation of minelike object localization error (yards)

$x(6) = p_{2,1} = \mathrm{R_r} =$ Contact localization error standoff which yields an 80% probability
of re - acquisition

$x(7) = p_{2,2} = \mathrm{T_{pf}} =$ Time spent prosecuting a non - mine classified as a mine or unsuccessfully
attempting to re - acquire a correctly classified mine (min)

$x(8) = p_{2,3} = \mathrm{T_n} =$ Time spent neutralizing (prosecuting) a classified mine (min)

The system of systems constrained MOE optimization has been solved for an increasing sequence of multipliers (“costfactor”) on the cost of the threshold system, denoted by C^*, which happens to be \$28.066M. This provides the decisionmaker with information to apply the CAIV approach to system upgrade or initial design. Plots are provided so that one can visualize the top level MOE improvement and corresponding MOP requirements as the system of systems cost upper bound is allowed to increase.

Constrained SQP Optimization

The baseline results are obtained through utilization of MATLAB®'s constrained sequential quadratic programming (SQP) algorithm, described in Chapter 6. Complete MATLAB® code and results for the MCM system of systems optimization is provided in Appendix A. Figure 17 below summarizes those results. The first two plots present the top-level MOE (*E*=time to complete minefield clearance) as a function of increasing cost factor and dollar cost upper bounds. The next two plots present the corresponding optimal MOPs as a function of increasing cost factor. The MOPs are normalized to their upper and lower bounds, with zero corresponding to their threshold system values and one corresponding to their technology limitations. Several significant insights can be obtained from examination of these plots:

- The system of systems MOE improves steadily to an asymptotic lower bound as the cost limit increases. Due to the imposed technology constraints, after a certain point no amount of money will enhance system performance.

- At the other extreme, if at least $1.25 \cdot C^*$ isn't spent, the quality constraint cannot be met and even a very slow system cannot be achieved.

- A subjective "knee of the curve" can be observed to occur somewhere around 1.8 times the threshold system cost (about $50M), after which he rate of MOE improvement significantly decreases.

- The component systems' MOP requirements can be determined from these plots (numerical results are in Appendix A), also as a function of cost factor. One can see which MOPs become stressed (i.e., move away from their threshold system values) and approach their technology constraint limits as the cost constraint is relaxed. Of course, this behavior is dependent upon the PBCM function developed for each MOP in Chapter 5, as well as their significance relative to the objective function and quality constraint.

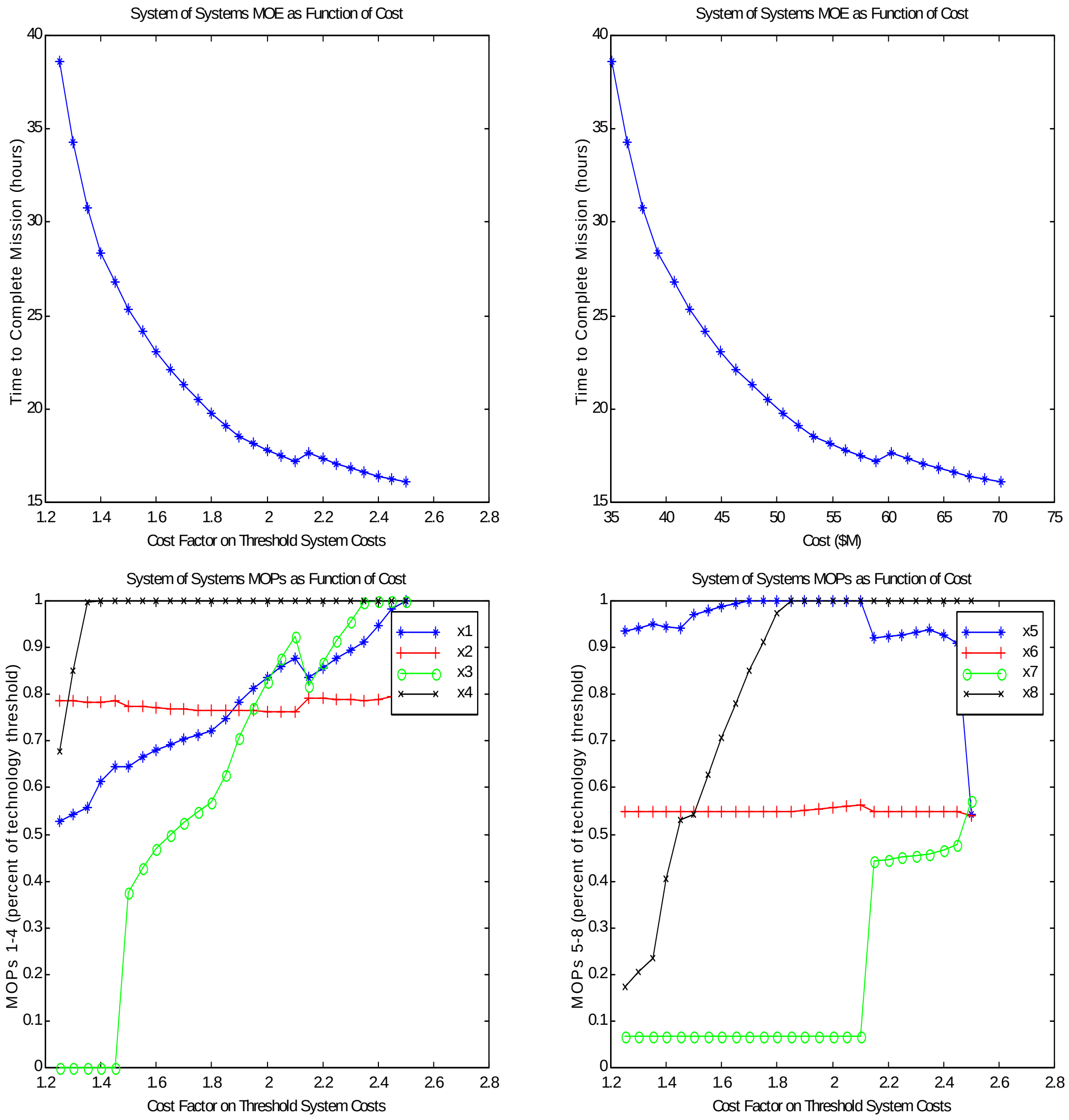


Figure 17: Constrained SQP Optimization Results for S

Note the hump in the MOE plot at costfactor=2.15, which also corresponds to a step change in the optimal value for x(7). This step change is due the existence of an objective function local minimum created by the unusually flat character of the PBCM for $x(7) = p_{2,2} = \mathrm{T}_{\mathrm{pf}}$ as illustrated in Figure 12. When the cost constraint is relaxed sufficiently to generate movement in x(7), the search algorithm moves from x(7)=7.0 to about 4, thereby using up all the additional allowable cost and then some. This additional cost allocated to x(7) forces cutbacks in other parameters, which turn out to be x(1) and x(3) as can also be seen in Figure 17. This translates into reduced overall performance that must correspond to a local, rather than global minimum—since the solution found at costfactor=2.1 is clearly superior and still meets the constraints.

The results can be used to design a specific cost-constrained upgrade to the threshold system of systems. For example, if the allowable cost constraint is twice that of the threshold system, then selecting $\mathbf{p}_1 = [85.3, 0.961, 0.55, 3.0, 42.0]$ and $\mathbf{p}_2 = [423.2, 7.0, 3.0]$ yields $E = 17.768$, with clearance rate q=0.846 at a cost of $56.132M. This is a substantial enhancement to the threshold system represented by $\mathbf{p}_1 = [10.0, 0.9, 2.0, 9.17, 90.0]$ and $\mathbf{p}_2 = [75.0, 6.6, 10.0]$ that results in an overarching MOE of $E = 93.33$ hours with clearance rate of only q=0.620 at a cost of $28.066M. Since CONSTR would not converge for costfactors less than 1.25, the analysis indicates that a system that satisfies the stringent requirement for 84.6% clearance will cost at least 25% more than a system of systems composed of the threshold component systems, but would take 38.38 hours to complete the clearance mission with a single pass from each system.

Single System Vs. System Of Systems Optimization

The claim is often made that optimization of each component system does not guarantee overall system of systems optimization[39]. However, an analysis is rarely available or offered to quantify the suboptimality of single system optimization. Here, we have optimized S_1 and S_2 separately, and compare the results with those just presented. It is not a straightforward process, due to the wide range of assumptions that can be made regarding the complementary system. To organize the analysis, two distinct single system approaches and sets of assumptions are hypothesized, analyzed, and tabular results offered in Figures 18 and 19. The two classes of single system developer assumptions regarding the other components to the system of systems are as follows:

1. Each system developer assumes the other is conducting a rigorous requirements optimization analysis and each has insight into the other's process and results.

2. Each system developer has little or no insight into the other's requirements analysis and will therefore assume that they will be interfacing with a system that is either very good or is marginally effective with regards to those MOPs that are known to interact between systems. This is referred to as "better" or "worse" than the baseline optimization results.

[39] Eisner, H., Marciniak, J., and McMillan, R., "Computer-Aided System of Systems (S2) Engineering", *Proceedings of the 1991 IEEE International Conference on Systems, Man, and Cybernetics*, 13-16 October 1991, University of Virginia, Charlottesville, VA.

It should be noted that both approaches should still offer more insight than would come from simple, single system (a.k.a. "stovepipe") analyses, which do not explicitly recognize that each system is part of a system of systems with quantifiable MOEs, quality constraints, technology constraints, and PBCMs for critical MOPs.

Referring to Figure 18, the first set of results is the baseline SQP optimization that is simply repeated from the previous section. The second set of results is obtained from optimizing S_1 and S_2 separately, but utilizing "perfect knowledge" about the baseline system of system optimization results. This is accomplished by holding the other system's baseline optimization results fixed when optimizing each system one at a time, subject to the stated cost and quality constraints. S_1 depends on the S_2 MOP $x(6) = p_{2,1} = \mathrm{R}_\mathrm{r}$ and S_2 depends on the S_1 MOPs $x(2) = p_{1,2} = \mathrm{P}_\mathrm{c}$, $x(3) = p_{1,3} = \mathrm{P}_\mathrm{fa}$, and $x(5) = p_{1,5} = \sigma$. (MATLAB® code and complete results are listed and plotted in Appendix B.) The resulting system MOEs are then combined to form the system of systems MOE, $E = E_1 + E_2$. The difference with the system of systems baseline optimization is shown as a percentage of the baseline. This provides insight as to the inherent suboptimality of performing a sequential optimization, even when the "right answer" to the full system of systems problem is known. The suboptimality of single system optimization is quite significant for cost factors below 1.5, but not otherwise. It is interesting to note that the local minima "hump" that is so noticeable in the baseline results is not apparent in the separately optimized results. The quality constraint that $q(\mathbf{x}) \geq 0.846$ was active for each costfactor value, and is therefore omitted from the table.

Next, the results of optimizing each system with good, but imperfect knowledge concerning the other's capability are shown in Figure 18 as a more realistic implementation of having knowledge of the other system's optimization process and results. S_1 is optimized assuming $x(6) = 500$, which is at the "knee-of-the-curve" for S_2's re-acquisition range PBCM (Figure 12). A similar inspection and assessment of the S_1 MOP PBCMs for $x(2), x(3), x(5)$ (see Figures 8,9,11) was done and $x(2) = 0.955, x(3) = 1.0, x(5) = 42$ were selected. The resulting recombined MOEs exhibit similar behavior in that the suboptimality is marked for small values of the cost constraint factor, which diminishes as the cost constraint is relaxed. Note that the recombined system of systems clearance rate is slightly better than the constrained value of 0.846, indicating a tradeoff of timeliness for a very modest gain in quality.

	System of Systems Optimization			Perfect Knowledge Single System Optimization				Imperfect Knowledge Single System Optimization				
costfactor	E	Cost	q	E1	E2	E	% Delta	E1	E2	E	% Delta	q
1.3	34.32	36.49	0.846	27.88	9.71	37.59	9.5%	27.29	13.16	40.44	17.8%	0.849
1.4	28.38	39.29	0.846	22.50	8.61	31.11	9.6%	22.06	10.65	32.71	15.3%	0.849
1.5	25.35	42.10	0.846	18.32	7.64	25.96	2.4%	18.03	9.31	27.34	7.8%	0.849
1.6	23.07	44.91	0.846	16.42	6.82	23.24	0.7%	16.27	8.26	24.53	6.3%	0.849
1.7	21.27	47.71	0.846	15.17	6.10	21.26	0.0%	15.08	7.38	22.46	5.6%	0.849
1.8	19.74	50.52	0.846	14.38	5.43	19.81	0.3%	14.33	6.60	20.92	6.0%	0.849
1.9	18.55	53.33	0.846	13.81	4.79	18.60	0.3%	13.77	5.89	19.66	6.0%	0.849
2.0	17.77	56.13	0.846	13.36	4.56	17.91	0.8%	13.33	5.24	18.56	4.5%	0.850
2.1	17.18	58.94	0.846	12.98	4.53	17.51	1.9%	12.95	4.63	17.58	2.3%	0.850
2.2	17.35	61.75	0.846	12.65	4.54	17.19	-0.9%	12.63	4.58	17.21	-0.8%	0.851
2.3	16.88	64.55	0.846	12.37	4.44	16.80	-0.5%	12.35	4.56	16.91	0.2%	0.851
2.4	16.40	67.36	0.846	12.11	4.37	16.48	0.5%	12.09	4.91	17.00	3.7%	0.851
2.5	16.12	70.17	0.846	11.89	4.38	16.26	0.9%	11.88	4.41	16.28	1.0%	0.851

Figure 18: Single System Optimization Results

Results from implementing the second approach are shown in Figure 19, again alongside the baseline results. We first optimize both systems separately, with the assumption that the other system's MOPs will be quite good, or "better" than the optimal constrained system of systems baseline results. Specifically, for the MOPs that are effective at the system interfaces, $x(6) = 550, x(2) = 0.975, x(3) = 0.50, x(5) = 42$ were selected. The resulting set of MOEs are a little bit better than the "perfect knowledge set", but the combined clearance rate, q, was up to 5% worse—a very significant number in the mine countermeasures domain.

What has happened is that by assuming the other system is being developed for high performance, one "under-engineers" his own system at the interface, and will naturally spend the remaining allowable funds to enhance his own single system MOE, E_i. Note that in this model, the interfacing MOPs have a first-order effect on the quality constraint, q. Hence the MOE times are very good, but the clearance rate is degraded as each system developer assumes that another system will take up the slack to produce an acceptable clearance rate. This is probably the most typical situation, because each system by itself will look good (in this case, fast) but the more complex combined system of systems quality MOE is either not addressed or is viewed as the responsibility of some other, overarching authority to assess the system of systems effectiveness.

Conversely, if one assumes that the other system's development is not performance-driven, the result is to "over-engineer" the system at the interface and since resources are constrained, this forces degradation in the single system MOE. This effect can be seen in the second set of single system optimization results in Figure 19 which were obtained with $x(6) = 300, x(2) = 0.975, x(3) = 0.50, x(5) = 52$. The clearance rate is significantly improved at the expense of a significant degradation in the system of systems performance MOE. Here, each system developer has assumed that his system must carry the load to maintain an adequate clearance rate and therefore relaxes parameters that most directly affect timeliness.

	System of Systems Optimization			Single System Optimization: Assume Other System is Better					Single System Optimization: Assume Other System is Worse				
costfactor	E	Cost	q	E1	E2	E	% Delta	q	E1	E2	E	% Delta	q
1.3	34.32	36.49	0.846	26.99	9.50	36.49	6.3%	0.823	29.23	13.29	42.52	23.9%	0.860
1.4	28.38	39.29	0.846	21.84	8.44	30.28	6.7%	0.823	23.45	11.55	35.00	23.3%	0.860
1.5	25.35	42.10	0.846	17.91	7.54	25.45	0.4%	0.824	19.09	9.92	29.01	14.4%	0.860
1.6	23.07	44.91	0.846	16.19	6.75	22.95	-0.6%	0.823	16.76	8.76	25.52	10.6%	0.860
1.7	21.27	47.71	0.846	15.04	6.04	21.08	-0.9%	0.822	15.35	7.81	23.16	8.9%	0.860
1.8	19.74	50.52	0.846	14.30	5.39	19.69	-0.3%	0.836	14.50	6.98	21.48	8.8%	0.860
1.9	18.55	53.33	0.846	13.75	4.77	18.52	-0.2%	0.820	13.90	6.24	20.14	8.6%	0.860
2.0	17.77	56.13	0.846	13.31	4.57	17.88	0.6%	0.845	13.43	5.56	18.99	6.9%	0.861
2.1	17.18	58.94	0.846	12.94	4.56	17.50	1.9%	0.847	13.04	4.94	17.97	4.6%	0.861
2.2	17.35	61.75	0.846	12.62	4.55	17.17	-1.0%	0.845	12.70	4.60	17.30	-0.2%	0.862
2.3	16.88	64.55	0.846	12.34	4.46	16.80	-0.5%	0.816	12.41	4.59	16.99	0.7%	0.864
2.4	16.40	67.36	0.846	12.09	4.41	16.50	0.6%	0.815	12.15	4.57	16.71	1.9%	0.864
2.5	16.12	70.17	0.846	11.87	4.38	16.26	0.8%	0.813	11.91	4.53	16.44	2.0%	0.864

Figure 19: Single System Optimization Assuming Better/Worse Other System Performance

As mentioned above, these analyses are each done assuming the same set of constraints and combining the separately obtained results to obtain the system of systems performance. This is in itself a very optimistic assumption in that typical stovepipe developments would not be that well-coordinated in their assumptions. For example, the reconnaissance system manager might be working towards spending no more than twice his threshold system cost whereas the clearance system manager might be limited to 1.5.

Worse, one system manager may not even be cost constrained, but seeking to maximize performance to the limits of technology. We have quantified significant suboptimalities in single system optimization even under self-consistent system of systems constraints on cost, technology, and the quality MOE. These models and analysis could be used to predict the effects of these other disparate approaches on system of system effectiveness.

In summary, this single system vs. system of systems optimization comparison illustrates what might be called common sense to the acquisition executive. If we are not resource constrained, then the correct course of action is simply to optimize each component system for performance without regard to cost—and it doesn't matter if this is implemented separately or as a system of systems. This has been the situation in Defense for many decades. But as the cost constraint is tightened it becomes increasingly important to consider the full impact of design decisions on the whole to get the most performance per unit dollar. Our results in this regard vividly illustrate the maxim,

"We are short of money, therefore we must think."

Constrained SQP Optimization Via Penalty Function Method

The penalty function method described in Chapter 6 was applied to our MCM problem in order to incorporate the nonlinear constraints into the objective function. Appendix C contains MATLAB® code and results from implementing the method utilizing the CONSTR optimization algorithm. The plots that constitute Figure 20 demonstrate well-behaved but suboptimal results relative to the baseline. These results were obtained utilizing the quadratic penalty function, which was found to be less sensitive to initial conditions than the absolute value penalty function whose results are also displayed. The quadratic penalty function formulation converged when initialized with the threshold system MOPs, whereas the absolute value penalty function would not converge unless it was initialized much closer to the optimal values—in this case, random perturbations of +/-20% from the baseline solution MOPs were sufficiently close to ensure convergence, though not always to a local minimum equivalent to the original problem. Also, CONSTR took anywhere from 2-20 times the number of iterations required for the baseline to converge within the selected nominal termination criteria. The degree of suboptimality can be seen explicitly in the Figure 21 table and the plot in Figure 25. Note that as the cost constraint is relaxed, the agreement converges.

Keep in mind that the penalty function approach on the closed form MCM problem formulation is used simply to establish and understand its nominal behavior prior to attempting stochastic optimization using simulation. Therefore, these results provide a best case limit for subsequent SPSA implementations. The dramatic increase in the number of required iterations is an indication that the transformed objective function is much “shallower”, primarily due to an effective relaxation of the constraint requirements—which is quite realistic from a management point of view. Therefore, the resulting minimum is nearly achieved in a rather large region rather than a single crisp point, implying that there should be some implications for flexibility in generating system requirements for the “optimal” CAIV sequence.

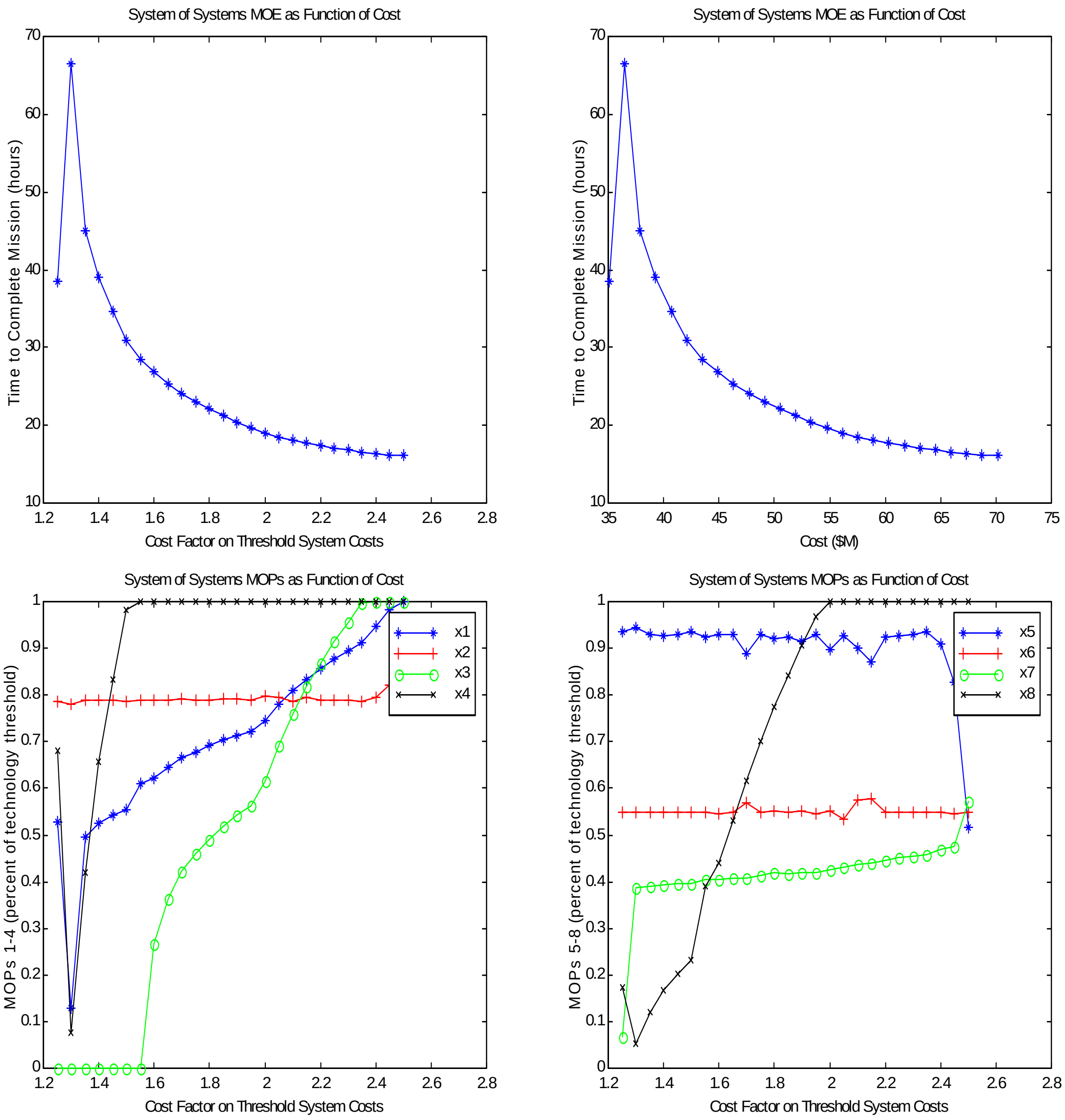
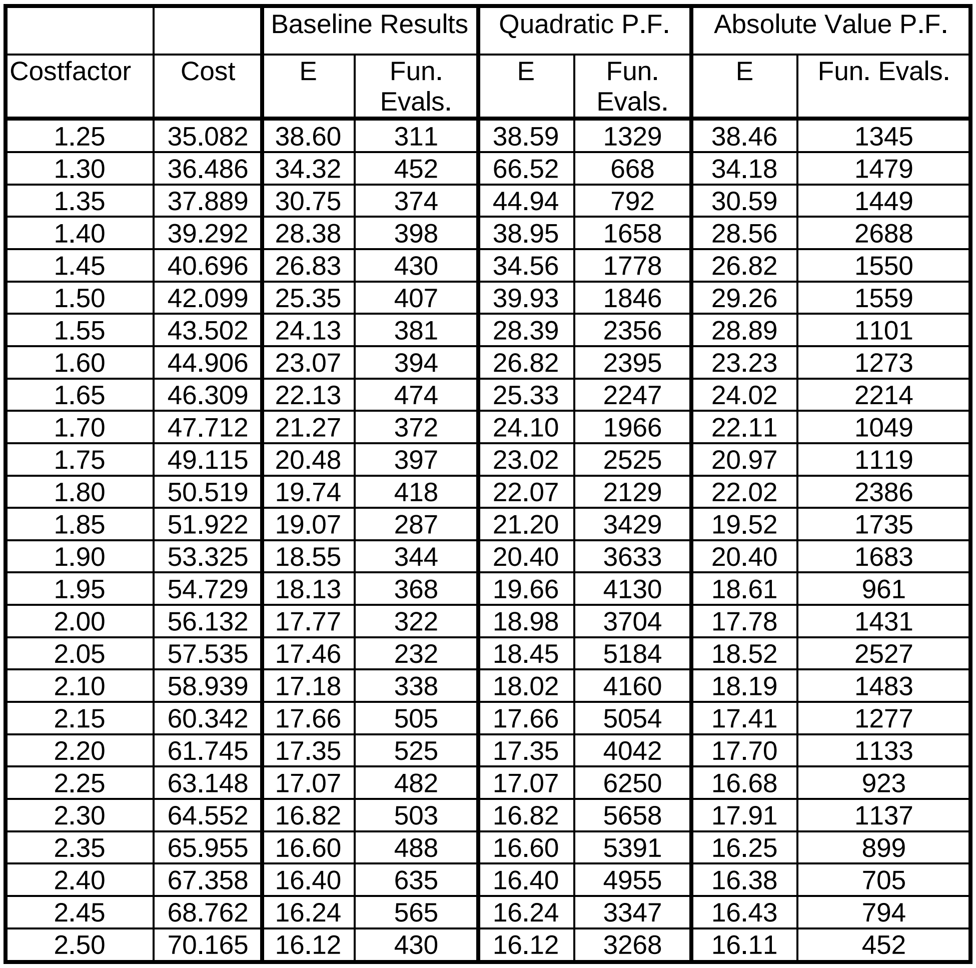


Figure 20: Constrained SQP Optimization (Penalty Function) Results for S



		Baseline Results		Quadratic P.F.		Absolute Value P.F.	
Costfactor	Cost	E	Fun. Evals.	E	Fun. Evals.	E	Fun. Evals.
1.25	35.082	38.60	311	38.59	1329	38.46	1345
1.30	36.486	34.32	452	66.52	668	34.18	1479
1.35	37.889	30.75	374	44.94	792	30.59	1449
1.40	39.292	28.38	398	38.95	1658	28.56	2688
1.45	40.696	26.83	430	34.56	1778	26.82	1550
1.50	42.099	25.35	407	39.93	1846	29.26	1559
1.55	43.502	24.13	381	28.39	2356	28.89	1101
1.60	44.906	23.07	394	26.82	2395	23.23	1273
1.65	46.309	22.13	474	25.33	2247	24.02	2214
1.70	47.712	21.27	372	24.10	1966	22.11	1049
1.75	49.115	20.48	397	23.02	2525	20.97	1119
1.80	50.519	19.74	418	22.07	2129	22.02	2386
1.85	51.922	19.07	287	21.20	3429	19.52	1735
1.90	53.325	18.55	344	20.40	3633	20.40	1683
1.95	54.729	18.13	368	19.66	4130	18.61	961
2.00	56.132	17.77	322	18.98	3704	17.78	1431
2.05	57.535	17.46	232	18.45	5184	18.52	2527
2.10	58.939	17.18	338	18.02	4160	18.19	1483
2.15	60.342	17.66	505	17.66	5054	17.41	1277
2.20	61.745	17.35	525	17.35	4042	17.70	1133
2.25	63.148	17.07	482	17.07	6250	16.68	923
2.30	64.552	16.82	503	16.82	5658	17.91	1137
2.35	65.955	16.60	488	16.60	5391	16.25	899
2.40	67.358	16.40	635	16.40	4955	16.38	705
2.45	68.762	16.24	565	16.24	3347	16.43	794
2.50	70.165	16.12	430	16.12	3268	16.11	452

Figure 21: Penalty Function vs. Baseline Numerical Comparison

First Order Constrained SPSA Optimization Via Penalty Function Method

Designed for stochastic optimization, the 1SPSA algorithm is itself a stochastic process due to the nature of generating the gradient approximation, as described in Chapter 6. To better understand this behavior, a simple three-dimensional constrained optimization problem was posed and solved with 1SPSA. The problem description and results summary are shown in Appendix D. The points of interest are that the algorithm always found the solution exactly, while on the average taking about the same number of iterations (9.8 vs. 9) as did CONSTR, which was also implemented as a control.

A variety of penalty functions and algorithm parameters were investigated and applied with 1SPSA on the closed form MCM problem formulation, with limited success. The wide range in parameter values at or near the optimal values cannot be handled well by a first order algorithm. The best results were obtained with the absolute value penalty function, starting each solution by randomly perturbing the baseline solution by 20%—an initial uncertainty level that a domain-knowledgeable system developer can probably guess at for their system. Figure 22 displays the usual plots for a 2500 iteration (5000 function evaluations) implementation, and Appendix D has the detailed output, including the SPSA algorithm control parameters that were utilized. While the top level MOE plot is not too noisy, there are some shortcomings (1) the resulting MOP plots are quite scattered, and (2) the clearance rate constraint is not well-satisfied, with q ranging from 0.810 to 0.866, and (3) the cost constraint is not well-satisfied, with the “solution” cost exceeding the upper bound by up to $5M. Regarding the first criticism, recall that we would like to use the MOP solutions to specify requirements for the component systems. Since this behavior is so apparently erratic, the algorithm is unsatisfactory for this purpose, on this problem. Actually, the problem with satisfying the clearance rate and cost constraints is more severe, in that the iterates do not "solve" the constrained problem.

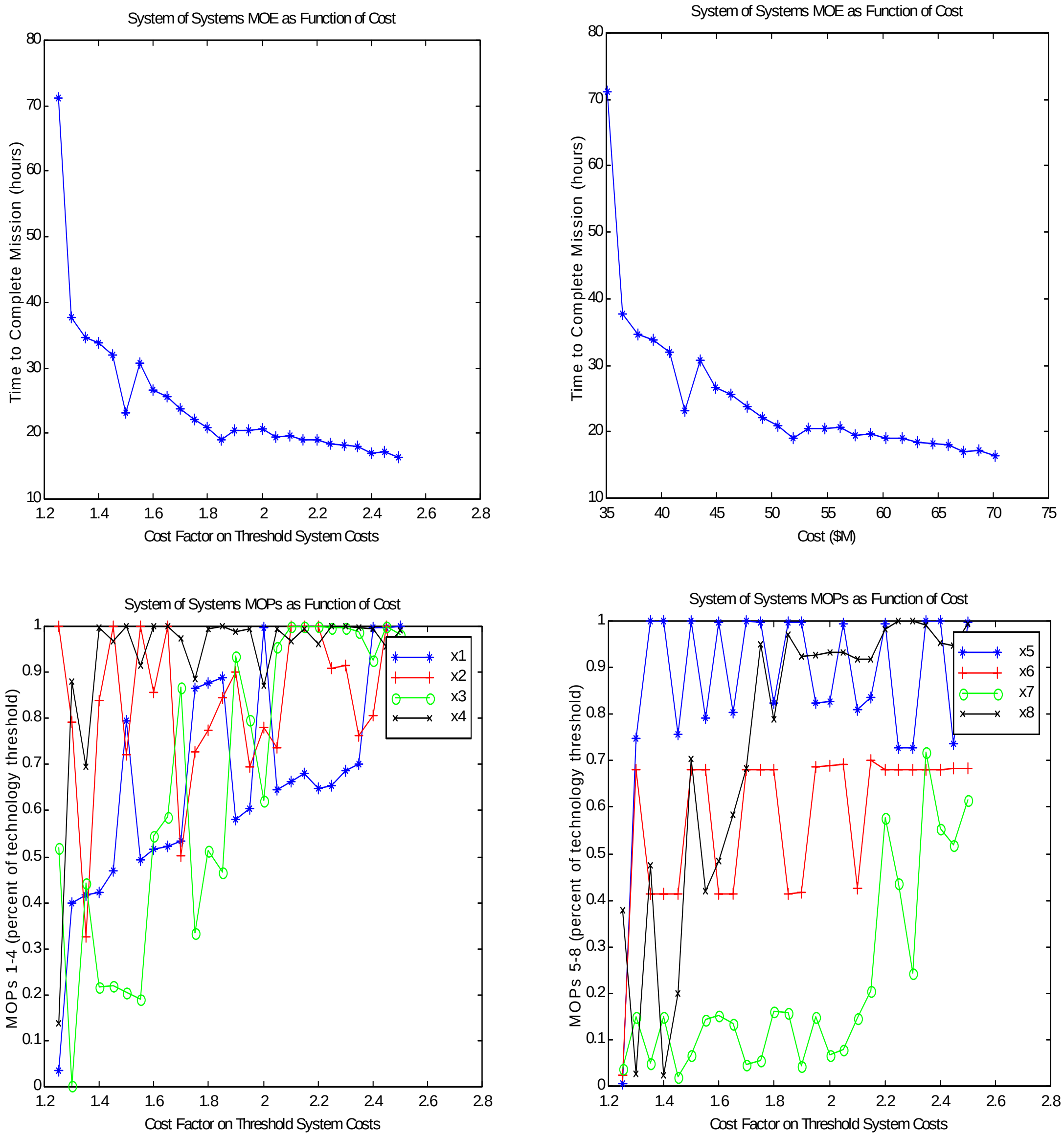


Figure 22: 1SPSA Results--5000 function evaluations

Second Order Constrained SPSA Optimization Via Penalty Function Method

To address the shortfalls of 1SPSA on this difficult problem, second order SPSA was implemented as mentioned in Chapter 6. As can be seen by inspection of the complete code and results listed in Appendix E, it can generate solutions that satisfy the full set of constraints, and approaches the MOE optimum values produced by CONSTR on the penalty function reformulation. Excellent satisfaction of both cost and quality constraints were obtained, unlike 1SPSA. Unfortunately, it still suffers from apparently erratic MOP estimates, which would be difficult to translate into system requirements. A wide variety of penalty function weightings, algorithm control parameters, blocking, Hessian averaging, and solution averaging techniques were applied in an attempt to obtain more stable MOP estimates with limited success. Figure 23 illustrates the results obtained with the same number of function evaluations (5000, with 1000 iterations) on the absolute value penalty function, with initial conditions randomly perturbed 20% from the baseline solution.

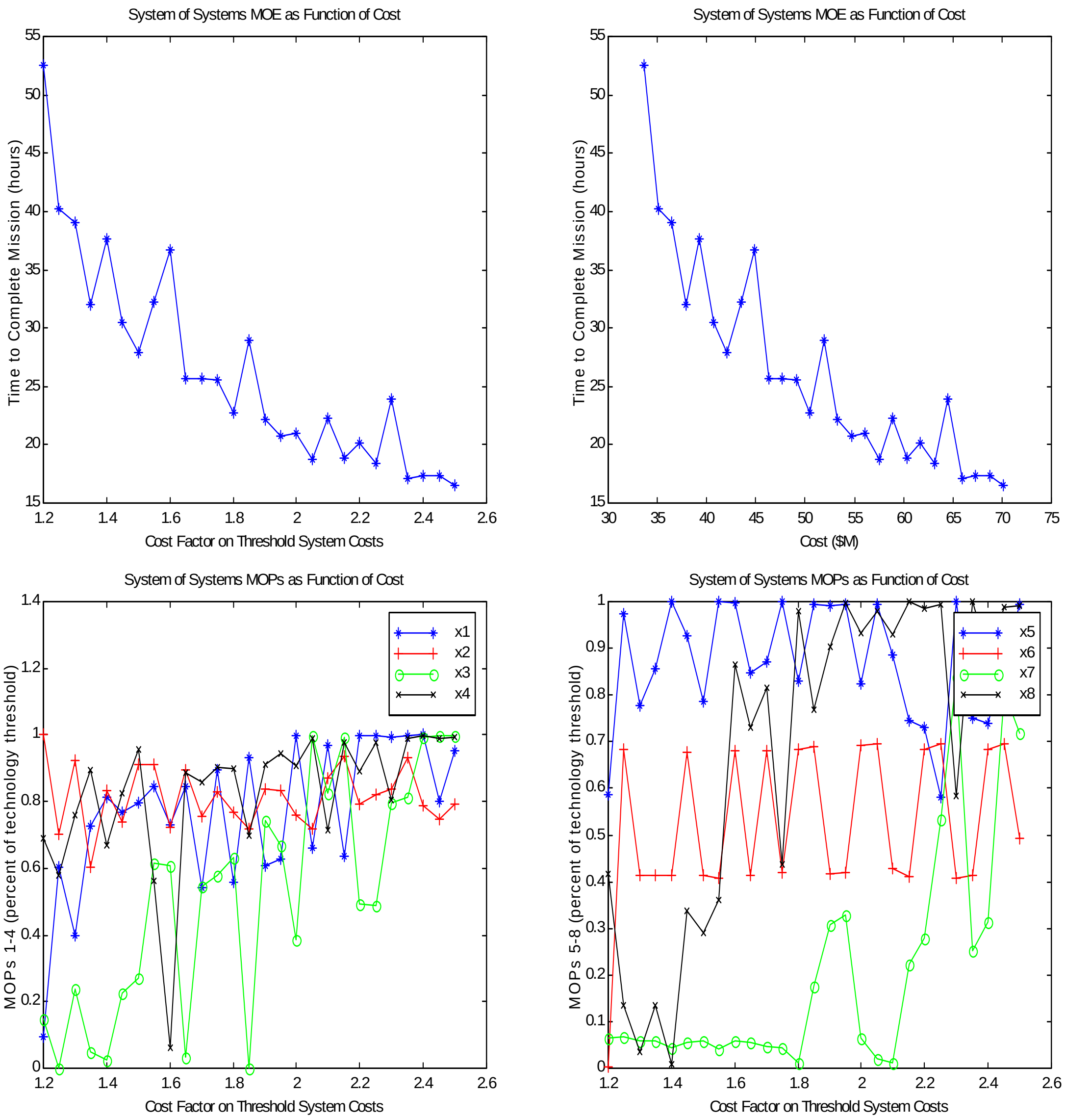


Figure 23: 2SPSA Results--5000 function evaluations

To better understand this erratic behavior, and to determine the best the algorithm (which is itself stochastic in nature) can produce on this problem, it was re-solved by initializing the algorithm with the baseline solution obtained from MATLAB®'s CONSTR. The following results were obtained, also with 5000 function evaluations (Figure 24):

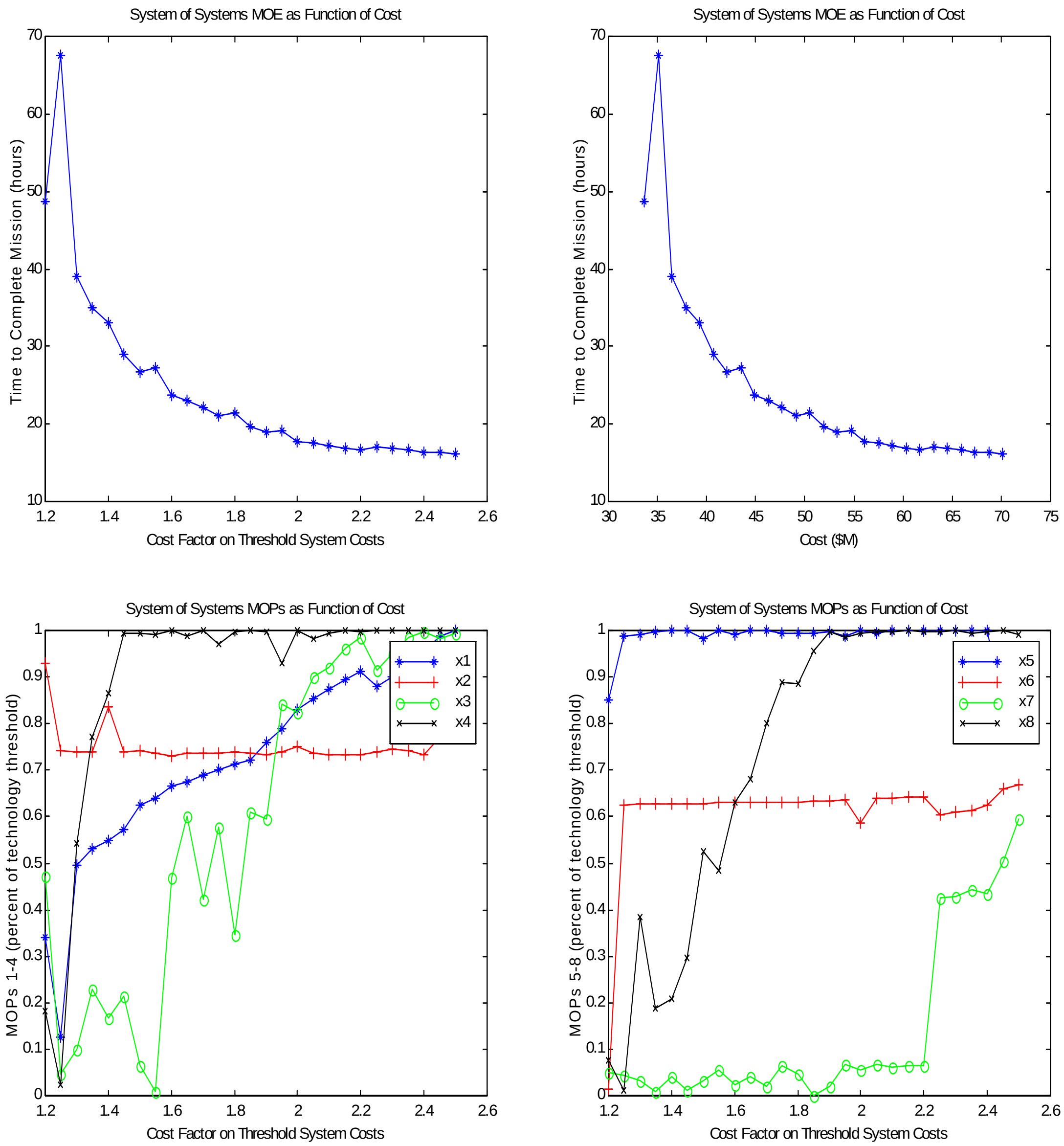


Figure 24: 2SPSA Initialized at Baseline Optimum

These results are excellent, and indicate that 2SPSA is stable and can produce good MOP results if it is initialized "close enough" to the optimum. It also confirms the previous finding that the transformed objective function is quite shallow. Of course, this is difficult to achieve when one is approaching a new problem for the first time. It should be noted that similar well-behaved results were obtained when the initial values were only 10% from the optimum, but getting that close to the optimum is probably impossible for a system architect to achieve, a priori. Also, recalling our experience with the penalty function control cases, the absolute value penalty function may be converging to nearby local minima that aren't optimal solutions to the original problem. This effect may actually account for some of the "noise" in Figure 24 rather than having been generated from the inherent algorithm uncertainty.

Figure 25 plots the MOEs obtained from each of the four algorithms that have been applied. There is good agreement with all four algorithms, even the two SPSA algorithms, which is encouraging as preparation for the stochastic simulation situation.

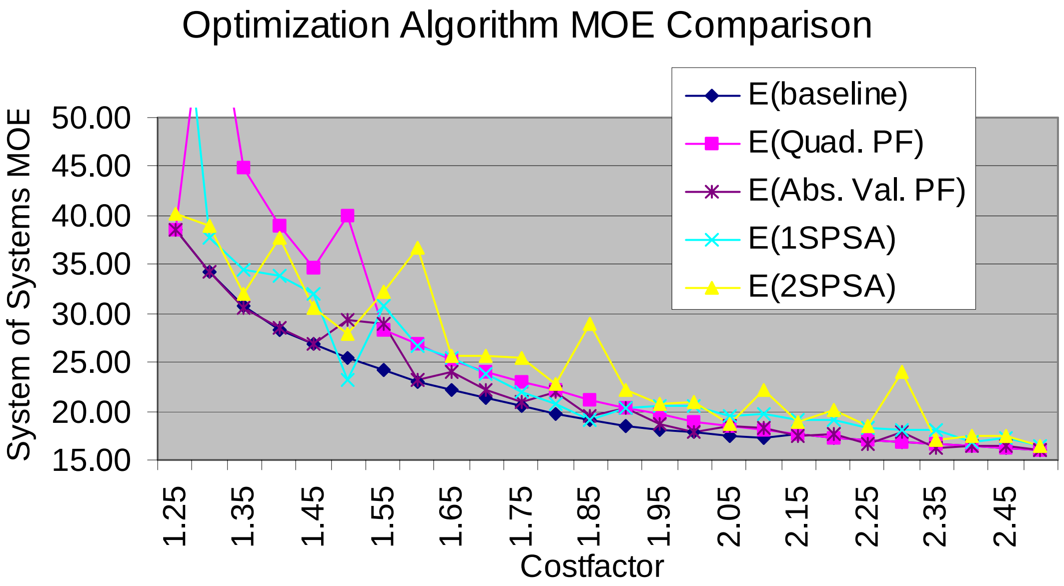


Figure 25: Comparison of System of Systems MOE Results

Although the results look good at the MOE level, we have to look carefully at the degree to which the constraints have been satisfied with 1SPSA and 2SPSA. Figure 26 offers a table of results that compares not only the final MOE values, but the cost and quality values resulting from each optimization. The cost upper bound is in the second column, and each algorithm is trying to produce q(x)=0.846—which was exactly satisfied with both the baseline and quadratic penalty function methods solved with CONSTR.

	Cost	Baseline Results		Quadratic P.F.		1SPSA: 5000 Fn. Evals			2SPSA: 5000 Fn. Evals.		
Costfactor	Upper Bnd.	E	Fun. Evals.	E	Fun. Evals.	E	q(x)	Cost	E	q(x)	Cost
1.25	35.082	38.60	311	38.59	1329	71.155	0.705	34.986	40.25	0.844	35.109
1.30	36.486	34.32	452	66.52	668	37.667	0.846	40.349	39.05	0.846	36.483
1.35	37.889	30.75	374	44.94	792	34.544	0.810	37.804	32.06	0.826	38.17
1.40	39.292	28.38	398	38.95	1658	33.915	0.846	41.980	37.63	0.846	39.292
1.45	40.696	26.83	430	34.56	1778	32.065	0.851	40.746	30.48	0.846	40.697
1.50	42.099	25.35	407	39.93	1846	23.24	0.846	46.095	27.90	0.846	42.105
1.55	43.502	24.13	381	28.39	2356	30.662	0.862	43.686	32.27	0.851	43.414
1.60	44.906	23.07	394	26.82	2395	26.708	0.848	45.738	36.75	0.846	44.906
1.65	46.309	22.13	474	25.33	2247	25.501	0.852	47.260	25.67	0.846	46.311
1.70	47.712	21.27	372	24.10	1966	23.766	0.830	47.672	25.62	0.846	47.712
1.75	49.115	20.48	397	23.02	2525	22.057	0.846	51.587	25.51	0.846	49.116
1.80	50.519	19.74	418	22.07	2129	20.791	0.846	54.596	22.77	0.846	50.518
1.85	51.922	19.07	287	21.20	3429	18.993	0.847	57.732	28.97	0.846	51.925
1.90	53.325	18.55	344	20.40	3633	20.388	0.851	53.791	22.18	0.846	53.322
1.95	54.729	18.13	368	19.66	4130	20.457	0.841	54.733	20.78	0.846	54.728
2.00	56.132	17.77	322	18.98	3704	20.579	0.847	57.189	20.97	0.846	56.134
2.05	57.535	17.46	232	18.45	5184	19.423	0.847	57.575	18.78	0.846	57.534
2.10	58.939	17.18	338	18.02	4160	19.642	0.853	59.005	22.23	0.846	58.939
2.15	60.342	17.66	505	17.66	5054	19.033	0.863	60.448	18.91	0.846	60.346
2.20	61.745	17.35	525	17.35	4042	19.042	0.866	61.897	20.12	0.846	61.746
2.25	63.148	17.07	482	17.07	6250	18.342	0.854	61.312	18.42	0.846	63.15
2.30	64.552	16.82	503	16.82	5658	18.121	0.855	61.619	23.95	0.846	64.552
2.35	65.955	16.60	488	16.60	5391	18.063	0.849	64.211	17.07	0.846	65.954
2.40	67.358	16.40	635	16.40	4955	16.936	0.852	67.868	17.40	0.846	67.358
2.45	68.762	16.24	565	16.24	3347	17.283	0.861	69.190	17.39	0.846	68.761
2.50	70.165	16.12	430	16.12	3268	16.377	0.850	70.334	16.52	0.846	70.166

Figure 26: Comparison of Optimization Algorithm MOE, Cost, and q(x) Results

To better compare 1SPSA vs. 2SPSA satisfaction of the constraints, Figures 27 and 28 are plots that compare both algorithms' final clearance rate and cost, respectively. While 2SPSA provides excellent results in this regard, 1SPSA does not, and therefore, almost none of its results are truly useful.

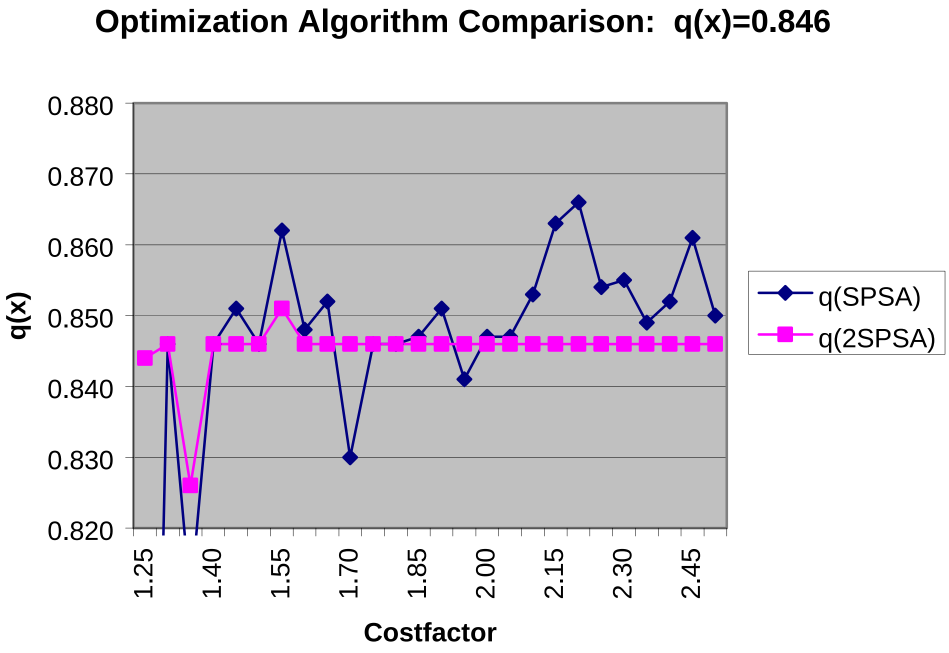


Figure 27: Comparison of 1SPSA vs. 2SPSA Clearance Rate Results

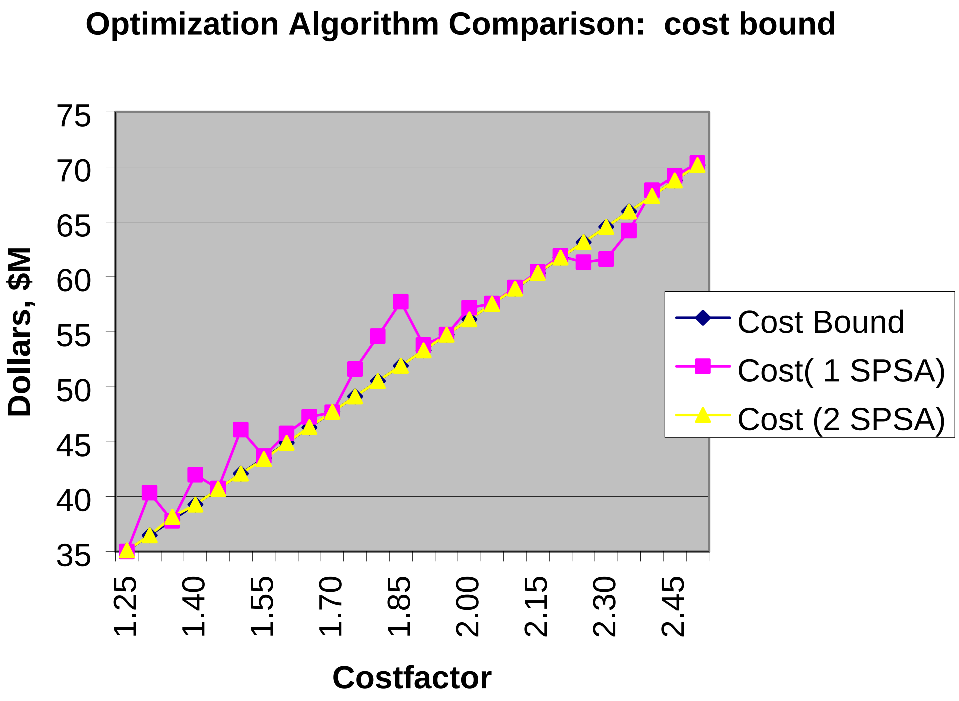


Figure 28: Comparison of 1SPSA vs. 2SPSA System of Systems Cost Results

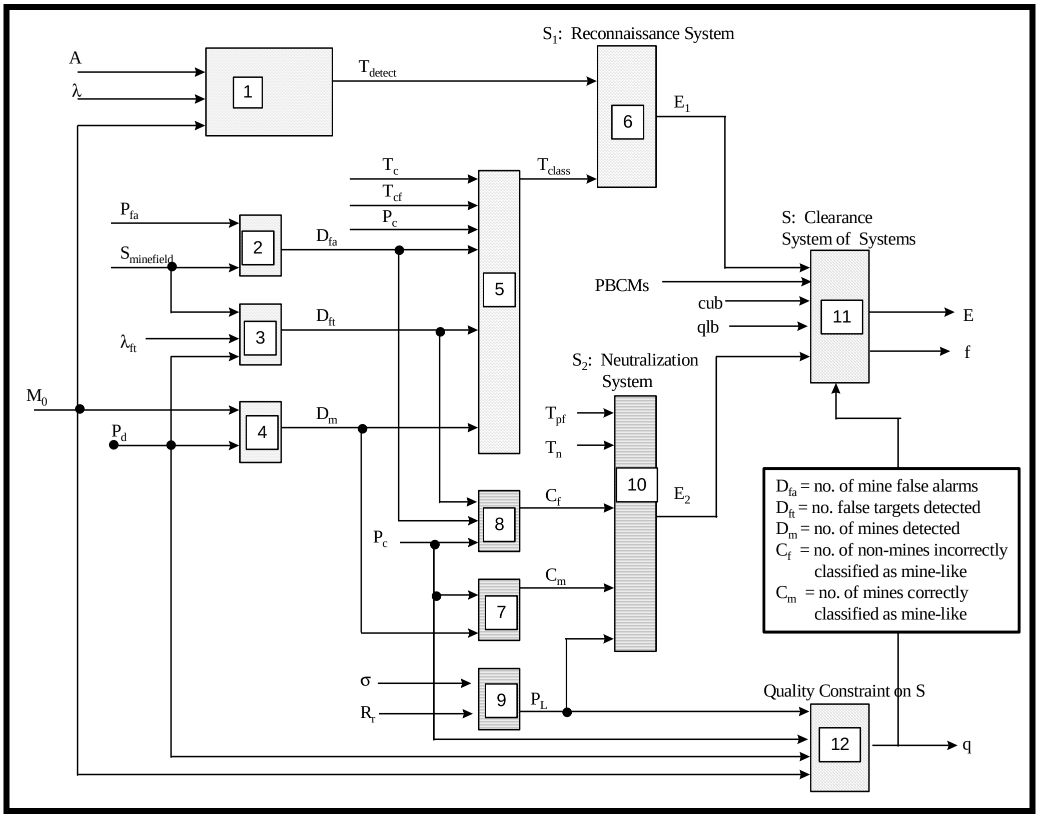
A final note regarding the nature of stochastic optimization is in order. When performing a stochastic optimization, each point on the CAIV plot is a random realization resulting from both the stochastic nature of the underlying function as well as that of the algorithm itself. Therefore, to obtain useable MOP estimates, many runs at each costfactor should be done and the most self-consistent composite set of runs should be selected as representative of solving the full range of the CAIV problem. This was done to obtain results in Chapter 8 that utilize the simulation with 2SPSA.

CHAPTER 8

PHASE II RESULTS: SIMULATION OBJECTIVE FUNCTION

Simulation Description

The MCM system of systems model was implemented as a simulation, patterned directly after the parameter dependency diagram developed in Chapter 5. It contains 12 functional blocks, as shown below (Figure 29), with parameters defined explicitly in Chapter 5.



S1: Reconnaissance System
A
λ
1
Tdetect
6
E1
Tc
Tcf
Pc
Tclass
Pfa
2
Dfa
Sminefield
5
S: Clearance
System of Systems
PBCMs
cub
qlb
11
E
f
3
Dft
λft
S2: Neutralization
System
M0
4
Dm
Pd
Tpf
Tn
10
E2
Cf
8
Pc
Cm
7
σ
9
PL
Rr
Quality Constraint on S
12
q
Dfa = no. of mine false alarms
Dft = no. false targets detected
Dm = no. of mines detected
Cf = no. of non-mines incorrectly classified as mine-like
Cm = no. of mines correctly classified as mine-like

Figure 29 MCM Simulation Block Diagram

The simulation was implemented as a MATLAB® function which produces one Monte Carlo realization of *E* and q with each function call. The simulation randomly generates the specified events in accordance with the MOPs. For example, looking at Block 4, if there are 100 mines in the minefield (i.e., M_0=100) and P_d=0.90, then the number of detected mines (D_m) is generated simply as 100 Bernoulli success/failure trials with probability of success equal to 0.90. The randomly generated D_m is then passed to Block 7, which in turn similarly generates the number of correctly classified mines, and so on. Eventually, the MOEs for that realization are produced and the resulting penalty function evaluation is returned by the simulation function MCMSIM after calculating the resultant system cost.

Second Order Constrained SPSA Optimization Via Penalty Function And Simulation

The resulting penalty function values are noisy and the amount of noise directly affects 2SPSA parameter selection. To characterize the noise, an auxilliary program was written that takes *x* as input, calls MCMSIM many times and generates statistics. For example, at the optimum for costfactor=2.0 and using the absolute value penalty function with gains A_1=3 and A_2=100, the standard deviation for $\mathrm{F_A}(\mathbf{x})$ and *E* are 2.3 and 0.38, respectively. Unfortunately, these modest gains for the cost and quality penalties did not produce acceptably close convergence. The values used for the previous analyses (A_1=50, A_2=1000) were used instead, with better convergence properties—but at the cost of greatly increased noise on the penalty function $\mathrm{F_A}(\mathbf{x})$: a mean and sigma values of 46 and 21 at costfactor equal to 2.0. This level of noise is extremely difficult to deal with, and is an artificiality of the penalty function method. A quick analysis of the quadratic penalty function indicates that the resulting noise on $F_Q(\mathbf{x})$ is unacceptably high when gains guaranteed to produce constraint agreement are used.

In stochastic optimization, the issue arises as to what the final answer is to the problem at hand. Specifically, what are the values for the MOEs *E* and q that are associated with the solution vector ***x*** (MOPs)? Since MCMSIM produces a random realization of the objective function, it must be called many times and results averaged to generate expected values for *E* and q. The 2SPSA code was therefore augmented to average 100 function calls to generate the average MOEs that are displayed.

To characterize the effects this substantial noise would have on 2SPSA, the 2SPSA code was initialized with the analytic model's optimal results for each value of the costfactor and optimized using the simulation as the objective function. The results are displayed in Appendix F and indicate that the algorithm is stable and would converge (or at least not wander far from) the optimal solution were it initialized sufficiently "close" to the optimum. Another insight is that the problem from this type of control run is that the final MOE values are relatively insensitive to values of ***x****(2)* and ***x****(3)* but very sensitive to ***x****(4).* As in the CONSTR baseline results, the value of ***x****(4)* took a step change at costfactor=2.5, where the problem changes character from a flat objective function to one that is much sharper. Also interesting is the level of noise in satisfaction of the clearance rate constraint that can be observed in the summary table in Appendix F. The noise is consistent with the characterization mentioned above, giving an indicator of the problem difficulty and noise in the simulation.

As in Chapter 7, initial values for ***x*** were chosen as random 20% perturbations from the optimal values as determined by the baseline (noise-free, analytic model) CONSTR results. Many runs were conducted at 1000 iterations, with some run out to 2000 and 4000 iterations without significantly enhancing the results. Figure 30 displays composite results of six sets of 1000-iteration runs that best satisfy the cost and clearance rate constraints. They were obtained with 2SPSA algorithm parameters α=0.602, A=50, c=0.02, a=10, blocking the use of any iterate that increased the current value of the objective function by more than 3, and averaging the last three unblocked iterates to obtain the final estimate. Although the MOE results are well-behaved, the MOP estimates are still noisy, exhibiting the same character as those obtained with the algorithm initialized at the optimum.

Figure 31 displays a comparison of the 2SPSA simulation MOE results against the various methods applied to the analytic model. Also included are 1 sigma bars on the average MOE obtained with the 2SPSA simulation results, as discussed above. In the MOE domain, the 1000-iteration SPSA simulation results compare favorably with the deterministic results.

The cost constraint agreement is excellent, diverging generally less than $10K. Figure 32 below shows the degree of convergence to the clearance rate constraint of 0.846 and the +/- one sigma bounds obtained numerically. This too shows excellent agreement as did the 2SPSA results obtained on the analytic model.

Practicality questions remain, however, as (1) we have used knowledge of the analytic model results to select the initial iterate and (2) the MOP estimates are too irregular to confidently assign system design parameters in a CAIV approach to the system of systems upgrade problem, as a small change in the cost constraint would produce unreasonably irregular changes in the corresponding MOPs.

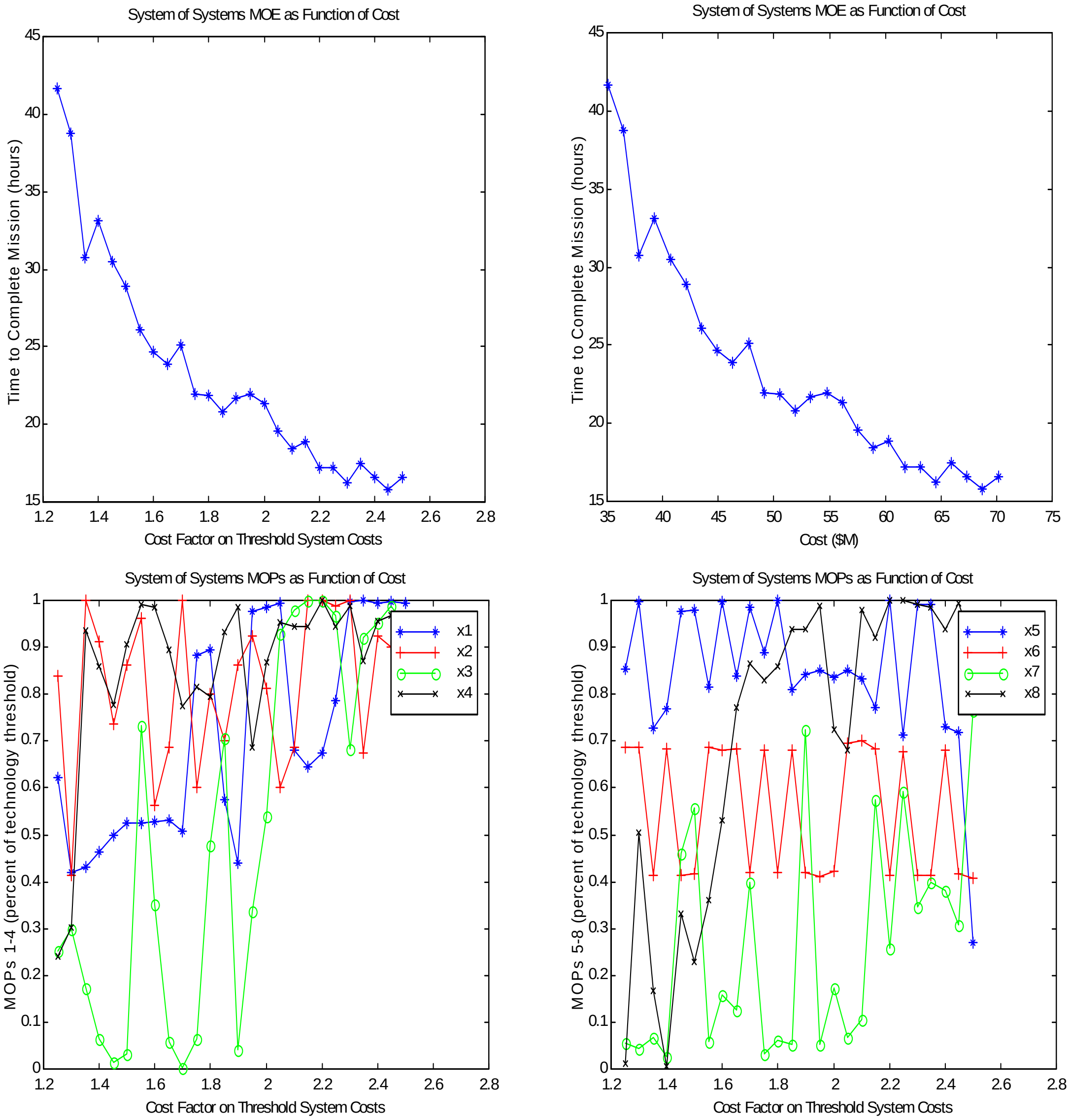


Figure 30: 2SPSA Simulation Results--5000 Function Evaluations

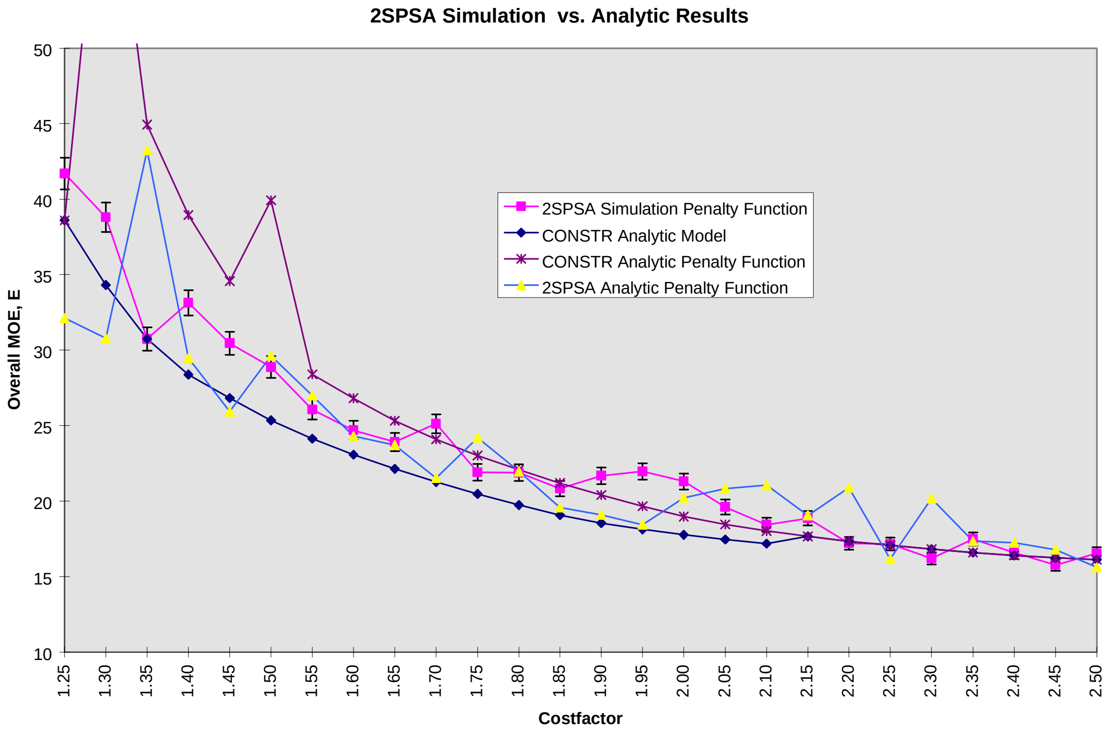


Figure 31: 2SPSA Simulation vs. Analytic Model Results

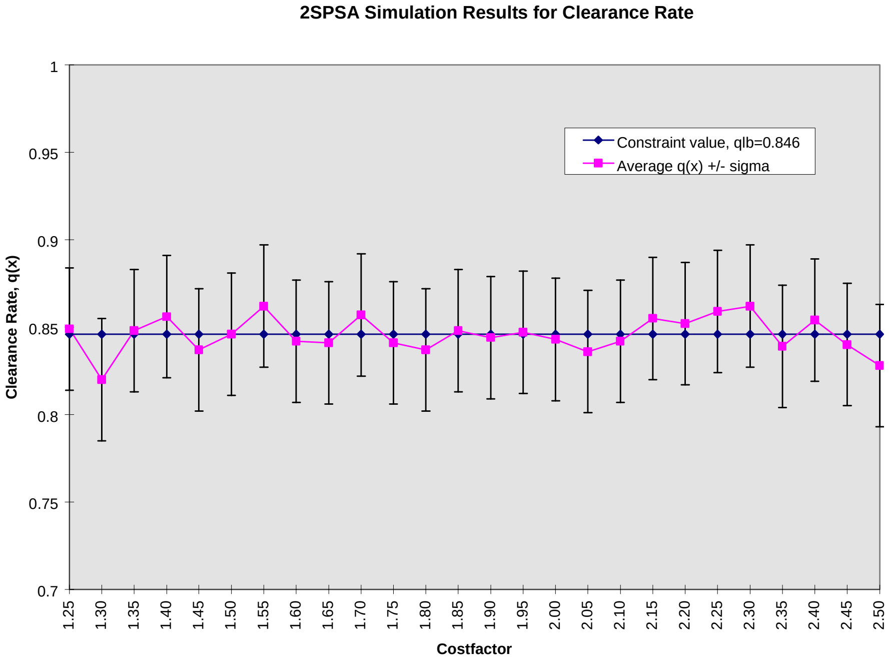


Figure 32: 2SPSA Simulation Results for Clearance Rate

Practical Selection of Initial MOP Estimates and Final Results

The question arises as to how, in the absence of prior analytic results, one would select an initial set of MOP values for the simulation optimization. Recall that for the algorithm comparisons above, we randomly perturbed the true optimum by 20% to initialize each 1SPSA and 2SPSA run. One "prior-free" way to do this is to first estimate the minimum value of the costfactor multiplier that would represent essentially no cost constraint—and therefore the optimization solution vector would move to the technology upper bound. In the absence of understanding the sensitivity of the objective function to each MOP, we could then assume that the optimal MOP values would tend to be proportional to the costfactor across its interval constraints. Therefore, when costfactor=1, we assume the optimum would be the threshold value, $\mathbf{x}^*$ and then linearly ramp up (or down) to its technology constraint as the costfactor constraint is relaxed to its asymptotic value. In our MCM problem notation with the asymptotic costfactor equal to 2.5,

$$\mathbf{x_0}(j) = \mathbf{x}^L(j) + \left(\frac{\text{costfactor - 1}}{1.5}\right)\left(\mathbf{x}^U(j) - \mathbf{x}^L(j)\right) \quad \text{for } j \text{ such that } \mathbf{x}^*(j) = \mathbf{x}^L(j)$$

$$\mathbf{x_0}(j) = \mathbf{x}^U(j) - \left(\frac{\text{costfactor - 1}}{1.5}\right)\left(\mathbf{x}^U(j) - \mathbf{x}^L(j)\right) \quad \text{for } j \text{ such that } \mathbf{x}^*(j) = \mathbf{x}^U(j)$$

This approach was used with 1000 iterations with marginally satisfactory results. Although it produced more stable MOP estimates, the MOEs were consistently worse than the random initialization scheme that was chosen to (1) provide a common challenge to all the algorithms, (2) represent some of the arbitrariness of most initial management decisions regarding MOPs, and (3) be certain of being “close enough” to the optimum to expect algorithm convergence.

To enhance results and their practical utility, two additional measures were taken. First, the number of iterations was increased to 2000 and a composite profile selected from the results of six separate runs at each costfactor value using the criteria of creating as smooth an overall MOE curve as possible—i.e., no prior information was used from the analytic baseline results. Figure 33 displays the MOE and MOPs normalized to their interval constraints in the same manner as previously presented results. The MOE estimates are very well-behaved as the costfactor constraint increases, and the MOP estimates are erratic yet still patterned.

Secondly, to enhance the practical utility of the MOP results, they were interpolated utilizing second to sixth order polynomials, as appropriate. The 2SPSA-generated MOP estimates and the polynomial interpolated curves are shown in their natural units. The solid red line in Figure 33 is the result of averaging the MOE realizations from 100 executions of the simulation function MCMSIM for each interpolated MOP vector as a function of the costfactor constraint. These final results are excellent and demonstrate practical utility for use in specifying component system MOPs.

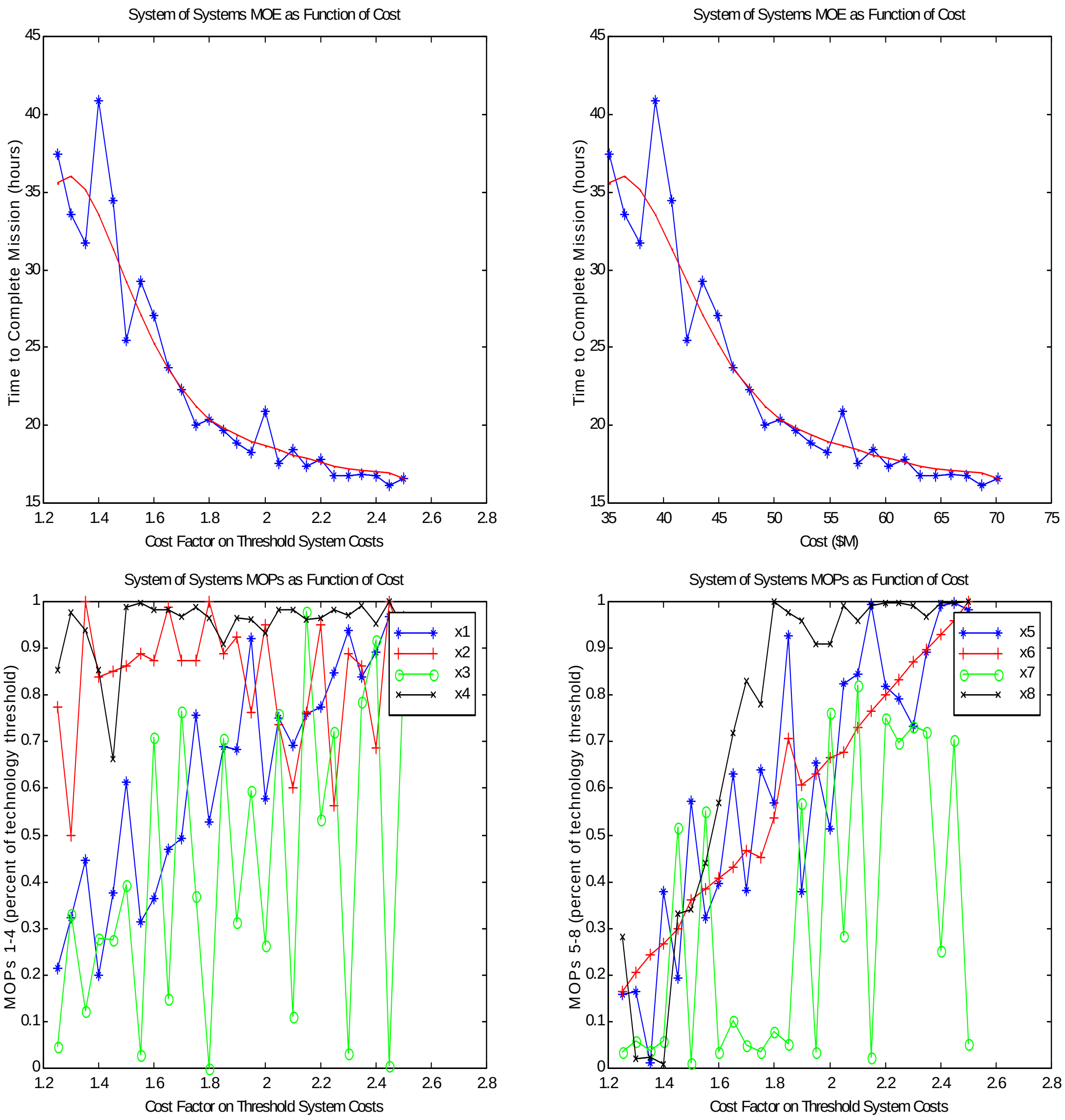


Figure 33: 2SPSA Simulation Results (MOE and Normalized MOPs)--2000 Iterations with Ramp Interpolation Initialization

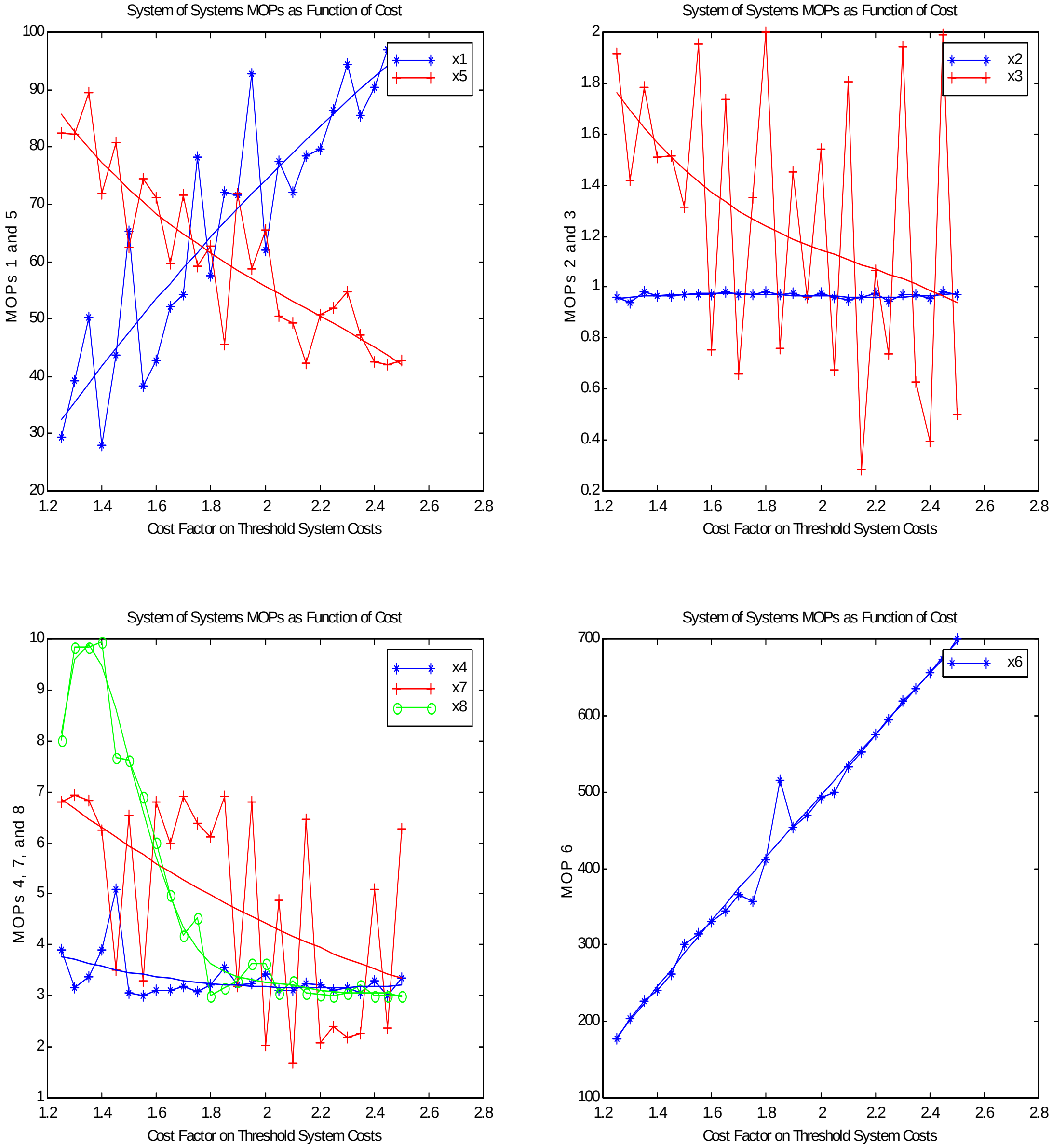


Figure 34: 2SPSA Simulation Results (MOP Values)--2000 Iterations with Ramp Interpolation Initialization

Figure 35 compares these results to the baseline analytic results in the overall MOE domain. The interpolated simulation results are very smooth, approximating the baseline results curve. Of course, we always have to make sure that the quality and cost constraints are reasonably well satisfied, which is indeed the case, as shown in Figures 36 and 37. Actually, the interpolated MOP values result in underspending the cost constraint by as much as $4M (about 6%), though the raw 2SPSA results are much closer.

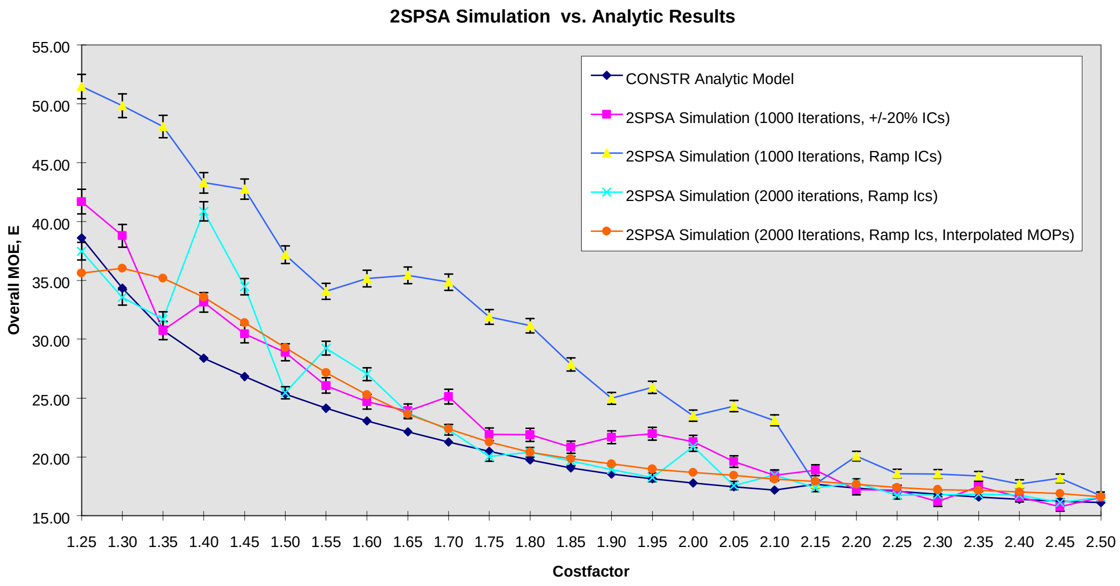


Figure 35: 2SPSA Simulation vs. Analytic Model Results

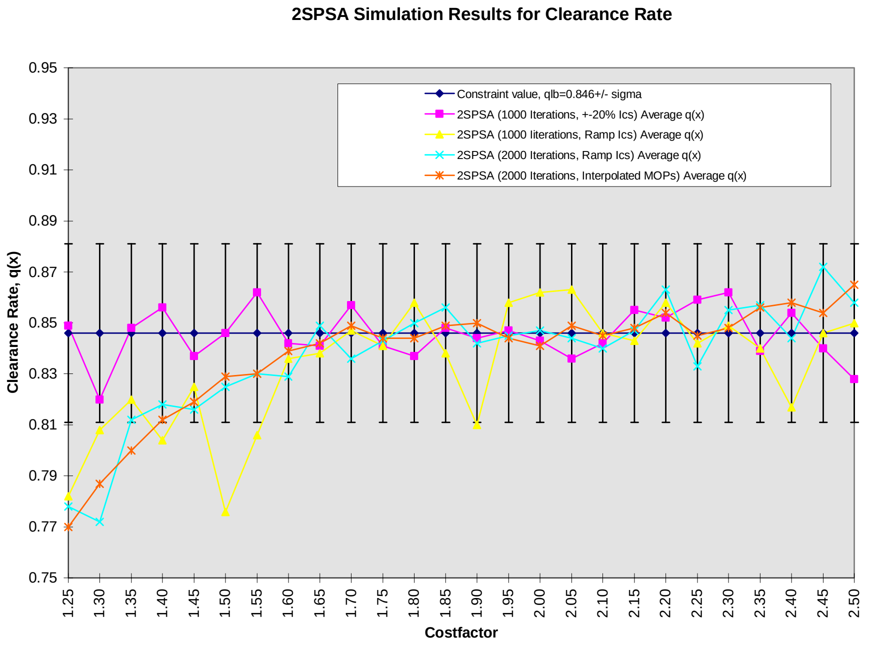


Figure 36: 2SPSA Simulation Clearance Rate Results

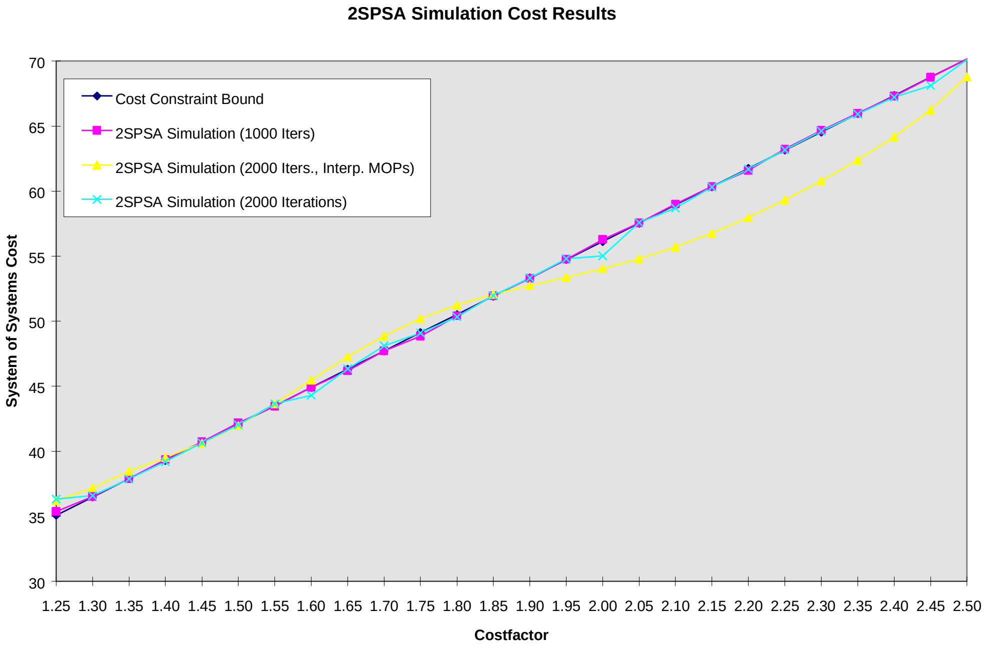


Figure 37: 2SPSA Simulation Cost Results

Also of interest is the distribution of costs between the optimized systems, S_1 and S_2. Figure 38 displays the resultant cost factor for each system as a function of the overall system of systems costfactor constraint. The distribution of costs varies as the costfactor constraint is relaxed and the impact of certain parameters are automatically traded off.

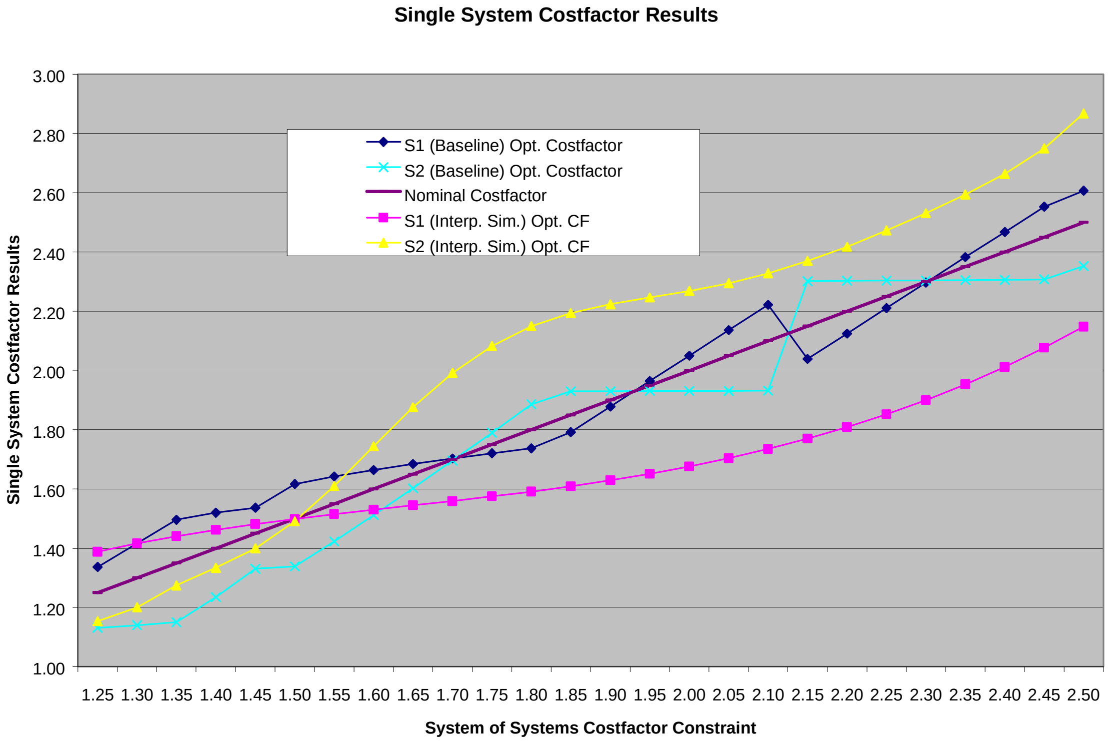


Figure 38: 2SPSA Simulation Costfactor Results by Individual System

Finally, referring back to the baseline analytic results from Chapter 7, we found that at a representative costfactor constraint value of 2.0, CONSTR produced $\mathbf{p}_1 = [85.3, 0.961, 0.55, 3.0, 42.0]$ and $\mathbf{p}_2 = [423.2, 7.0, 3.0]$ yielding $E = 17.8$, with clearance rate q=0.846 at a cost of $56.1M. Our final results with utilizing nonlinear, constrained, stochastic optimization with the simulation as the means to evaluate the objective function, produced $\mathbf{p}_1 = [74.3,0.965,1.1,3.2,55.8]$ and $\mathbf{p}_2 = [495.6,4.4,3.3]$ yielding $E = 18.7$, with clearance rate q=0.841 at a cost of $54.0M. The overall MOE is about 5% worse for $2M less cost and a very slight decrease in clearance rate of 0.005.

These runs have highlighted two fundamental difficulties created by the penalty function approach to constrained stochastic optimization: (1) sufficiently large penalty gains to guarantee constraint agreement also makes the transformed objective function very “flat” and (2) the penalty function multiplies the effect of the simulation noise to the point where it makes convergence very difficult. These factors should motivate further research in more direct methods for constrained stochastic optimization to enhance the likelihood of successful utilization of advanced M&S to support system of systems acquisition decisions.

CHAPTER 9

VERIFICATION AND VALIDATION ISSUES

This Chapter concerns how Verification, Validation and Accreditation (VV&A) for the system of systems upgrade process would be approached when extending the methodology beyond a proof of principle demonstration. Implementation of these VV&A approaches on a practical problem of significant scope and fidelity is beyond the scope of the dissertation effort. The discussion is included here for purposes of scoping the practical aspects of gaining widespread acceptance for a newly proposed process.

A commonly accepted set of definitions for VV&A of models and simulation (M&S) are as follows[40]:

> ***Verification**: The process of determining that a model implementation accurately represents the developer's conceptual description and specifications.*
>
> ***Validation:** The process of determining the degree to which a model is an accurate representation of the real world from the perspective of the intended uses of the model.*

[40] Williams, M.L. and Sikora, J., "SIMVAL Minisymposium—A Report", *Bulletin of Military Operations Research,* Vol. 24, No. 2, June 1991.

***Accreditation**: An official determination that a model is acceptable for a specific purpose.*

There are two parts to verification: *logical verification* that ensures that the basic equations and algorithms are correct, while *code verification* checks whether these representations or abstractions of the real world are correctly implemented in the computer.

Note that both V&V are considered to be processes. Since models and simulations are abstractions of the real world, complete validation is considered to be achievable for only the most simple of models. Hence it is almost never considered appropriate to refer to a complex M&S as “validated”. This is where accreditation comes in, representing a decision by an authoritative body that a given level of validation is sufficient to support a given decision process or application.

There is a large body of literature on the V&V of software systems, and a lesser extent on the VV&A of M&S[41]. This dissertation has examined a constrained optimization process that utilizes both an analytic model and a simulation to investigate upgrading a representative systems of systems. The issues in verifying and validating a <u>process</u> such as this is not unlike those associated with an expert system. An expert system implements an acknowledged process, although this process is not an expert system per se, due to the lack of an acknowledged knowledge base. Practical implementation of this process in its mature form would generally utilize previously developed models and simulations, each of which presumably brings its own VV&A

approach arising from the single system development which it supports. VV&A of the selected models and simulations is not part of the verifying and validating the process per se, but would be expected to have already been accomplished on the core models and simulations that would map component system MOPs to system of systems MOEs. The discussion below is therefore adapted from literature on V&V of expert systems, which seem more applicable to V&V of a new process.

What is Important and Why

When applied to a software system, validation means building the right system, and verification means building the system right. For a process, validation means implementation of the right process for the problem, whereas verification means implementing the intended, self-consistent and repeatable process. Therefore, although verification is a necessary condition, we are ultimately concerned with validation, since a highly efficient implementation of an invalid (or inappropriate) process would be useless. In the context of the proposed process, we will not be solving the system of systems upgrade problem directly, but an abstraction of it—and we are therefore justifying the level of abstraction through the validation process. Specific concerns that are applicable to the validation of the proposed process[42]:

[41] Pace, D.K., “Issues in Validating Simulations with Heterogeneous Levels of Fidelity”, *Second High Fidelity Modeling and Simulation Workshop in the DIS Environment*, Johns Hopkins University, 23 March 1995.

[42] O’Keefe, R.M, Balci, O., and Smith, E.P., “Validating Expert System Performance”, *IEEE Expert*, Winter 1987, pp. 81-87.

- **What to validate.** Candidate items to validate include (1) intermediate results, (2) the final result, and/or (3) the reasoning of the process. O'Keefe et al, recommend the reasoning process as the overall validation standard. His logic is that although a poor reasoning process can produce a correct result on a trial problem, it is unlikely to be successfully scaled up to a problem with full complexity. Also, it may be impossible to classify the final result as decidedly right or wrong. Domain experts can be assembled to classify the intermediate results, final result, and process reasoning into categories such as: ideal, acceptable, sub-optimal, and unacceptable.
- **What to validate against.** The process can be validated against known results or against domain expert opinion as mentioned above. Known results can be generated on carefully structured test cases, in particular system of systems models with closed form expressions for the component system MOEs and the overall MOE. Domain expert opinion could be obtained by examination of "successful" Cost and Operational Effectiveness Assessment (COEA) studies, but there is no guarantee that they are correct—indeed, the operating hypothesis is that the proposed system of systems upgrade process would be superior to the typical COEA approach that concentrates on just one system.
- **What to validate with.** For expert systems, validation efforts generally concentrate on comparisons against documented test cases. But here, the sample of possible COEA test cases will be very small, and as mentioned above, would not produce a range of suggested enhancements across the full system of systems, but their solutions would necessarily be confined to one system. The process constraints could be set up

to force enhancements to only one component system, or the system of systems could be composed of the degenerate case of one system, but this would not be a comprehensive validation check case. O'Keefe et al, recommend testing against a small number of complex cases and asking a panel of experts to assess how well the process handles them.

- **When to validate.** No agreement exists on this in the context of expert system validation. Initial validation of the process reasoning seems adequate, with subsequent re-validation as the process applications are extended.
- **Controlling the cost of validation.** This is generally measured in terms of time and the cost to obtain applicable data, which can be substantial for a large system of systems.
- **Controlling bias.** Several categories of bias must be guarded against, even when using experts in the validation process, such as:
    1. bias for or against a quantitative optimization approach
    2. developer bias in selection of test cases/scenarios that would favor his/her particular system's potential contribution to the whole
    3. bias for or against a centralized authority imposing system of systems perspectives or requirements on system developers

Selection Strategies For Ultimate Design Of The Process VV&A Approach(es)

Validation of the reasoning process would necessarily be qualitative. Potential qualitative approaches include:

- **Face validation**. Experts in the problem domain (i.e., mine countermeasures, COEA studies, stochastic optimization, etc.) subjectively compare the process performance against their expert experience, assessing the process results at face value with regard to prescribed acceptable performance range.
- **Predictive validation.** This approach utilizes historic test cases with known results and assessments of the errors previously obtained. Here, it will be necessary to find a test case where multiple systems have been upgraded in an attempt to enhance a system of systems. Again, the assessment will be against some acceptable performance range standard.
- **Field tests.** This is a long term approach that would be used only after initial validation, in an attempt to widen the validated domain.
- **Sensitivity analysis.** This is performed by systematically changing model parameters and constraint sets and observing impact on the solutions. This is a particularly useful approach in this application, as the available test cases may only be special cases of the process in that generally only one component system of a particular system of systems is considered for upgrading or replacement.

This area of validation of M&S and expert systems is one in which there is no consensus on approach. My initial judgment is that a combination of face validation, predictive validation and sensitivity analysis will be appropriate.

Perspectives on the Limitations of V&V Approaches

There are going to be limits and concerns related to any approach to verification and validation. One of the most important is related to completeness and complexity[43]. This is of particular concern here, as we are expanding the scope of systems engineering into the system of systems domain. At the system of systems level, the relationship of component systems' MOPs to the overarching MOE becomes increasingly non-intuitive for all but the most significant parameters, as the complexity of the system of systems grows. Therefore, the value of expert opinion wanes as the completeness of the representation or abstraction increases.

Secondly, there is the problem of acquiring sufficient data, or test cases to perform the qualitative validation described above. Since the whole approach represents a whole new way at looking at the system of systems upgrade process, it may be that the accumulated experience of available test cases will serve simply as knowledge acquisition for expert face validation.

Finally, and perhaps most significantly, we don't even know how well the current acquisition decision process "works" in a cost-constrained environment. We do know that in the absence of cost constraints, it doesn't matter whether one optimizes at the system of system of systems level. We also know that those two factors are relatively

[43] Rosness, R., "Limits to Analysis and Verification", Verification and Validation of Complex Systems: Human Factors Issues , Springer-Verlag, 1992, Wise, J.A., Hopkin, V.D., and Stager, P., Eds.

new concerns in DoD acquisition policy and implementation: (1) concern with cost rather than performance as the independent variable in trade studies, and (2) the system of systems perspective. These new factors make it unlikely that an applicable process baseline could ever be established.

CHAPTER 10

SUMMARY AND CONCLUSIONS

A systematic approach to considering how best to upgrade specific, complex systems of systems has been postulated and demonstrated. The process treats cost as the independent variable and seeks to find the "best" point design for upgrading a particular system of systems, subject to stated cost and technology constraints, relative to an overarching measure of effectiveness. The design requirements so generated represent an improved system of systems that may involve upgrading all component systems simultaneously, not just one at a time. Extension of the typical COEA/AOA approach to the system of systems environment would hypothesize a manageable suite of point designs and assess them against a common metric generally functionalized by cost. In contrast, this approach automatically generates a continuum of "optimal" designs not only functionalized by total system of systems cost, but also taking into account realistic technology constraints as well.

The process has been demonstrated on a naval mine countermeasures system of systems representation of sufficient complexity and detail to demonstrate the feasibility of the approach. This proof of principle demonstration features a constrained, nonlinear optimization algorithm whose objective function is a closed-form representation of the primary system of systems MOE, with constraints represented by functionalized Performance Based Cost Models, technology-driven bounds on system MOPs, and a secondary system of systems MOE. Various optimization approaches have been demonstrated and differences quantified, including the suboptimality of considering just one system at a time. Due to the nature of complex system of systems interactions, implementation of this optimization technique on problems of national interest will require M&S to represent the mapping of system measures of performance to single system MOEs and on to the overarching system of systems MOE. A stochastic simulation of the MCM system of systems was therefore implemented and optimized utilizing a constrained variant of the Stochastic Perturbation Simultaneous Approximation method.

In short, a disciplined, quantitative approach to developing system of systems upgrade options for very complex engineering situations has been developed and demonstrated. Application of this approach which can result in more effective and comprehensive systems acquisition and technology investment strategies, with the secondary benefit that the process can be used as a framework to determine how to utilize campaign-level M&S to support acquisition decisions.

Conclusions

The approach and methodology supports the quantitative elements of the System of Systems (S2) Engineering Process, especially those related to system performance optimization and development of transition or upgrade alternatives. Inclusion of PBCMs and associated overall cost constraints enables the decisionmaker to proceed based upon cost as the independent variable considerations. General methodology was developed and feasibility has been demonstrated through a proof of principle analysis of a naval mine countermeasures system of systems with realistic performance based cost models and technology constraints, and utilizing both classical and stochastic optimization algorithms.

In quantifying the sub-optimality of single-system optimization relative to simultaneously optimizing the entire system of systems, several significant insights were obtained and verified by examining some reasonable assumptions that might be held by component systems' management concerning the concurrent system engineering processes of other systems. For example, if a system engineer assumes that the other component systems are being developed for high performance, he will "under-engineer" his own system with respect to interfacing parameters and will tend to allocate his resources to enhancement of his single system MOE. Conversely, if a system engineer assumes that the other systems are not performance-driven, the result is to "over-engineer" his system at the interface and since resources are constrained, this forces degradation in his single system MOE. In both cases, the overall system of systems is sub-optimal, because all system engineers are making the same erroneous assumptions. These effects are accentuated with restrictive cost constraints and become insignificant as overall cost constraints are relaxed to the point where the most advanced technology is affordable for all system components.

Due to the complex interaction between systems (including multiple units of similar systems), closed form analytic expressions for system of systems performance are giving way to simulation-based representations. Therefore, we developed an efficient methodology that can utilize stochastic simulation to evaluate the system of system's measure of effectiveness. Transformation of the constrained nonlinear stochastic optimization problem formulation to include only interval constraints enabled the application of a straightforward constrained optimization projection adaptation of the second-order SPSA algorithm. Although feasible, the inherent ill-conditioning of the selected penalty function approach and the number of simulation-based functions evaluations still required for convergence limits practical, large scale applications and therefore motivates future research to enhance efficiency.

Future Research

Several interesting avenues for further research have become apparent in working through the general system of systems optimization approach and the proof of principle demonstration. The following areas need to be addressed to varying degrees in order to implement the methodology on systems of systems of national interest, scope and complexity:

1. Nonlinear constrained stochastic optimization. Efficient methods for nonlinear constrained stochastic optimization will be necessary to utilize simulations of reasonable fidelity to evaluate system of systems MOEs as part of optimization objective functions. The penalty function approach utilized here is a simple, brute force method that "works", but brings with it certain ill-conditioning that should be avoided. An approach that incorporates the efficient function evaluation properties of 2SPSA into a classical nonlinear programming algorithm such as SQP should be effective.
2. Incorporation of performance based cost modeling into system engineering. The development of a PBCM should be an integral part of the system engineering process, as they are necessary for any reasonable CAIV approach, including this one. Although research into methods and standards for software development cost estimating is active, similar efforts should be initiated for appropriate categories of

full systems, and incorporated into system engineering standards efforts. Furthermore, the modeling and simulation community will increasingly seek to integrate PBCMs into campaign-level simulations to reflect the cost of acquiring and using certain systems of systems. Therefore, a PBCM should be a requirement for all acquisition programs.

3. Extension to force level analysis. Although the general formulation included notation for force level analysis, only one system of each type was treated by the optimization approach utilized in this dissertation. Due to the small numbers of systems that generally make up a system of systems, an integer programming approach tailored to this problem structure is necessary to extend the analysis to determine how many systems of each type are appropriate to optimize the system of systems MOE.
4. Faster than real-time simulations. The trend in modeling and simulation is towards full-fidelity, “physics-based” models with graphical interfaces that provide realistic visualization to enable training and face validation by subject matter experts. The associated level of fidelity comes at the expense of execution speed that is at odds with the needs of quantitative analyses that generally utilize Monte Carlo implementations to enhance stochastic simulation sample size—and clearly at odds with the approach utilized here that may need several thousand simulation-generated function evaluations to optimize a system of system at just one cost constraint value. Research into methods of quantifying the effects of lowering the model fidelity on our system of systems optimization method is necessary to complement the research into efficient constrained nonlinear stochastic optimization algorithms.

APPENDIX A

CONSTRAINED SQP OPTIMIZATION MATLAB® CODE, EXAMPLE, AND RESULTS

```
%  MCM System of Systems
      %  CONSTR w/o Explicit Gradient
      %
      %  Filename:  \matlab\dissertation\MCM32.m
      %  Output file initialization
fid=fopen('c:\matlab\dissertation\mcm32 output.doc','w');
fprintf(fid,'output from execution of MCM32.m\n\n');
      %================================================================================
      %
      %        MCM Initialization
      %
      %================================================================================
      %
      % x(1)=p11=  S1 area coverage rate (nm^2/day)
      % x(2)=p12=  S1 classification probability
      % x(3)=p13=  S1 FAR (#/nm^2)
      % x(4)=p14=  S1 time to classify (min)
      % x(5)=p15=  S1 navigation accuracy (yards)
      % x(6)=p21=  S2 Re-acquisition range (yards)
      % x(7)=p22=  S2 time to prosecute a false target (min)
      % x(8)=p23=  S3 time to neutralize (min)
      pd=0.90;     %S1 detection system probability of detection
x0=[10,0.9,2.0,9.17,90,75,6.6,10.0]; %Might need to start at feasible point.  This won't
meet q(x) constraint.
fprintf(fid,'x0=\n');fprintf(fid,'%10.3f',x0);fprintf(fid,'\n');
vlb=[10.0,0.9,0.25,3.0,42,75,1.0,3.0];              % lower bound constraint on x
vub=[100,0.98,2.0,9.17,90,700,7.0,10.0];            % upper bound constraint on x
fprintf(fid,'vlb=\n');fprintf(fid,'%10.3f',vlb);fprintf(fid,'\n');
fprintf(fid,'vub=\n');fprintf(fid,'%10.3f',vub);fprintf(fid,'\n\n');
%  Compute px1, polynomial fit cost function parameters for x(1)
x1=[10,57,82,94];
y1=[3,4.483,7.655,11.445];
px1=polyfit(x1,y1,3);
%  Compute px2, polynomial fit cost function parameters for x(2)
x1=[0.9,0.93,0.96,0.98];
%y1=[3,4.483,7.655,11.445];%original PBCM for Pc
y1=[.2,.5,1.4,2.2];  %revised PBCM that reflects COTS/NDI development
px2=polyfit(x1,y1,2);
%x2=0.9:.01:1.0;
%  Compute px3, polynomial fit cost function parameters for x(3)
x1=[2,1,0.5,0.25];
y1=[8,10.319,13.592,16.429];
px3=polyfit(x1,y1,3);
%  Compute px4, polynomial fit cost function parameters for x(4)
x1=[3.513,3.89,4.77,9.17];
y1=[9.191,8.190,7.574,5.0];
px4=polyfit(x1,y1,2);
%  Compute px5, polynomial fit cost function parameters for x(5)
x1=[90,60,48,42];
y1=[0.050,0.250,0.450,0.550];
px5=polyfit(x1,y1,2);
%  Compute px6, polynomial fit cost function parameters for x(6)
x1=[129,457,622,75];
y1=[3,4.8,7.655,1.5];
px6=polyfit(x1,y1,3);
%  Compute px7, polynomial fit cost function parameters for x(7)
x1=[6.6,4.4,3.3,2.64,1.32];
y1=[5.0,7.5,8.190,9.191,16.621];
px7=polyfit(x1,y1,3);
%  Compute px8, polynomial fit cost function parameters for x(8)
x1=[10,8,7,5,3];
y1=[5.3,6,7,10,15];
```

```
px8=polyfit(x1,y1,2);
%compute threshold system cost
cost0=polyval(px1,x0(1))+polyval(px2,x0(2))+polyval(px3,x0(3))+polyval(px4,x0(4))+polyval(
px5,x0(5))+polyval(px6,x0(6))+polyval(px7,x0(7))+polyval(px8,x0(8))
costfactor=1.5;  %nominal value
%Insert costfactor loop
i=0
for costfactor=1.25:0.05:2.5 %Note problem is infeasible with costfactor<1.25
   i=i+1
   costfactor
   fprintf(fid,'\n');
cub=costfactor*cost0         %system of systems cost constraint
qlb=0.846              %quality constraint lower bound--remember that constraints are less-
thans, not gt's.
fprintf(fid,'Run with Costfactor = ');fprintf(fid,'%10.3f',costfactor);fprintf(fid,'
cub=');fprintf(fid,'%10.3f',cub);fprintf(fid,'   qlb=');fprintf(fid,'%10.3f\n',qlb);
sminefield=20;         %minefield area, (nm^2)
m0=100;                %number of mines in minefield, initially
lambda=m0/sminefield; %mine density, (#/nm^2)
dmine=600;             %average distance between mines, (yards)
vtransit=7;            %S1 vehicle transit speed (knots)
ttransit=dmine/(2000*vtransit);    %transit time during classification (hours)
lambdaft=1.0;          %false target density (#/nm^2)
% Define parameters for the objective function
p1=pd;
p2=lambda;
p3=lambdaft;
p4=ttransit;
p5=sminefield;
p6=cub;
p7=qlb;
[f,g]=mcmfun(x0,p1,p2,p3,p4,p5,p6,p7)
fprintf(fid,'Initial Values\n');
x=x0;
      cost =
polyval(px1,x(1))+polyval(px2,x(2))+polyval(px3,x(3))+polyval(px4,x(4))+polyval(px5,x(5))+
polyval(px6,x(6))+polyval(px7,x(7))+polyval(px8,x(8))
      q=p1*x(2)*exp(-x(5)/(4.481*x(6)));
      f1 = (p5/60)*(24*60/x(1) + p2*x(2)*x(4)*p1 + (2*x(4)-p4)*((1-x(2))*p1*p2 + x(3) +
p1*p3));
      f2 = (p5/60)*(p1*x(2)*x(8)*p2*exp(-x(5)/(4.481*x(6)))+(1-exp(-
x(5)/(4.481*x(6))))*p1*x(2)*x(7)*p2 + (1-x(2))*(x(3)+p1*p3)*x(7));
      E=f1+f2;
      q=p1*x0(2)*exp(-x0(5)/(4.481*x0(6)));
fprintf(fid,'f=');fprintf(fid,'%10.3f',f);fprintf(fid,'   E=');fprintf(fid,'%10.3f',E);
fprintf(fid,'   cost=');fprintf(fid,'%10.3f',cost);fprintf(fid,'
q=');fprintf(fid,'%10.3f\n',q);
%  Print out threshold system values for f and constraints, g.
      %===============================================================================
      %
      %        CONSTR Initialization and Call
      %
      %===============================================================================
      % Modify x0 so that it satisfies initial quality constraint, g(2)
%x0(2)=0.96
%x0(5)=45
%x0(6)=600
[f,g]=mcmfun(x0,p1,p2,p3,p4,p5,p6,p7)
%  Print out initial system values for f and constraints, g.
grad=[];          % need to set to null matrix in order to pass p1....p7 to mcmfun
options(1)=1;     % print output table
%options(2)=1e-5;  % relax x termination criteria
%options(3)=1e-3;  % relax f termination criteria
%options(4)=1e-5;  % relax constraint violation limits
options(9) = 0;   % if =1, check analytic gradient
[x,options]=constr('mcmfun',x0,options,vlb,vub,'mcmgrad',p1,p2,p3,p4,p5,p6,p7)
[f,g]=mcmfun(x,p1,p2,p3,p4,p5,p6,p7)
fprintf(fid,'Final Values\n');
cost =
polyval(px1,x(1))+polyval(px2,x(2))+polyval(px3,x(3))+polyval(px4,x(4))+polyval(px5,x(5))+
polyval(px6,x(6))+polyval(px7,x(7))+polyval(px8,x(8))
q=p1*x(2)*exp(-x(5)/(4.481*x(6)))
f1 = (p5/60)*(24*60/x(1) + p2*x(2)*x(4)*p1 + (2*x(4)-p4)*((1-x(2))*p1*p2 + x(3) + p1*p3));
f2 = (p5/60)*(p1*x(2)*x(8)*p2*exp(-x(5)/(4.481*x(6)))+(1-exp(-
x(5)/(4.481*x(6))))*p1*x(2)*x(7)*p2 + (1-x(2))*(x(3)+p1*p3)*x(7));
```

```
E=f1+f2
fprintf(fid,'Total time, E=');fprintf(fid,'%10.3f',E);
fprintf(fid,'       cost=');fprintf(fid,'%10.3f',cost);fprintf(fid,'
q=');fprintf(fid,'%10.3f\n',q);
fprintf(fid,'x=');fprintf(fid,'%10.3f',x);fprintf(fid,'\n');
fprintf(fid,'Function evaluations= ');fprintf(fid,'%8.0f\n',options(10));
cf(i)=costfactor; fval(i)=f;
costval(i)=cub;fevals(i)=options(10);costi(i)=cost;qfinal(i)=q;
z1(i)=x(1);z2(i)=x(2);z3(i)=x(3);z4(i)=x(4);z5(i)=x(5);z6(i)=x(6);z7(i)=x(7);z8(i)=x(8);
z1(i)=abs((z1(i)-x0(1))/(vlb(1)-vub(1)));
z2(i)=abs((z2(i)-x0(2))/(vlb(2)-vub(2)));
z3(i)=abs((z3(i)-x0(3))/(vlb(3)-vub(3)));
z4(i)=abs((z4(i)-x0(4))/(vlb(4)-vub(4)));
z5(i)=abs((z5(i)-x0(5))/(vlb(5)-vub(5)));
z6(i)=abs((z6(i)-x0(6))/(vlb(6)-vub(6)));
z7(i)=abs((z7(i)-x0(7))/(vlb(7)-vub(7)));
z8(i)=abs((z8(i)-x0(8))/(vlb(8)-vub(8)));
end
%  Plot option 1:  Plot System of systems MOE as CAIV
figure
plot(cf,fval,'-*b')
title('System of Systems MOE as Function of Cost')
xlabel('Cost Factor on Threshold System Costs')
ylabel('Time to Complete Mission (hours)')
figure
plot(costval,fval,'-*b')
title('System of Systems MOE as Function of Cost')
xlabel('Cost ($M)')
ylabel('Time to Complete Mission (hours)')
%
%
%  Plot option 2:  Plot MOPs as CAIV
figure
plot(cf,z1,'-b*',cf,z2,'-r+',cf,z3,'-go',cf,z4,'-kx')
legend('x1','x2','x3','x4')
title('System of Systems MOPs as Function of Cost')
xlabel('Cost Factor on Threshold System Costs')
ylabel('MOPs 1-4 (percent of technology threshold)')
figure
plot(cf,z5,'-b*',cf,z6,'-r+',cf,z7,'-go',cf,z8,'-kx')
legend('x5','x6','x7','x8')
title('System of Systems MOPs as Function of Cost')
xlabel('Cost Factor on Threshold System Costs')
ylabel('MOPs 5-8 (percent of technology threshold)')
%  print table of results to file
fprintf(fid,'\n');
fprintf(fid,'cost factor');fprintf(fid,'   cub');fprintf(fid,'          E');fprintf(fid,'
cost');fprintf(fid,'        qfinal');fprintf(fid,'      fun. evals\n');
for j=1:i

fprintf(fid,'%10.2f',cf(j));fprintf(fid,'%10.3f',costval(j));fprintf(fid,'%10.3f',fval(j))
;fprintf(fid,'%10.3f',costi(j));fprintf(fid,'%10.3f',qfinal(j));fprintf(fid,'%10.0f\n',fev
als(j));
end
status=fclose(fid)
```

```
function [f,g] = mcmfun(x,p1,p2,p3,p4,p5,p6,p7)
%updated Px2, 7/3/97
px1=[4.503408803940725e-005,   -5.386095666335044e-003,    2.159330101073730e-001,
       1.334245377520354e+000];
px2=[2.834645669291690e+002,   -5.076377952756583e+002,    2.274598425197177e+002];
px3=[-2.048380952380911e+000,    9.987333333333214e+000,   -1.794233333333325e+001,
       2.032238095238094e+001];
px4=[1.159691730856429e-001,   -2.175732453433467e+000,    1.520381256718985e+001];
px5=[2.061825086032983e-004,   -3.775958229500408e-002,    1.777803488786043e+000];
px6=[1.504875482450802e-007,   -1.578229837871938e-004,    5.516694369186904e-002,  -
       1.813253427503106e+000];
px7=[ -2.850358103957624e-001,    3.846213159671302e+000,   -1.726423877731832e+001,
       3.334408692656030e+001];
px8=[2.102445277065673e-001,   -4.109593768487483e+000,    2.539723920331297e+001];
f1 = (p5/60)*(24*60/x(1) + p2*x(2)*x(4)*p1 + (2*x(4)-p4)*((1-x(2))*p1*p2 + x(3) + p1*p3));
f2 = (p5/60)*(p1*x(2)*x(8)*p2*exp(-x(5)/(4.481*x(6)))+(1-exp(-
x(5)/(4.481*x(6))))*p1*x(2)*x(7)*p2 + (1-x(2))*(x(3)+p1*p3)*x(7));
```

```
f=f1+f2;
% evaluate cost constraint
g(1) =
polyval(px1,x(1))+polyval(px2,x(2))+polyval(px3,x(3))+polyval(px4,x(4))+polyval(px5,x(5))+
polyval(px6,x(6))+polyval(px7,x(7))+polyval(px8,x(8))-p6;
% evaluate negative of quality constraint
g(2) = -p1*x(2)*exp(-x(5)/(4.481*x(6)))+p7;
%p1=pd
%p2=lambda
%p3=lambdaft
%p4=ttransit
%p5=sminefield
%p6=cub
%p7=qlb
```

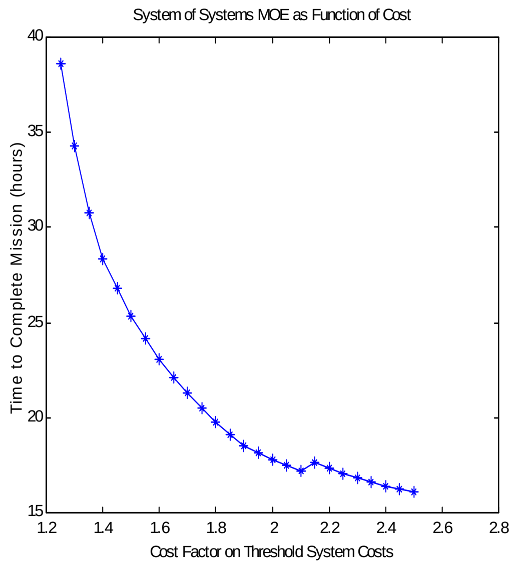
```
function [df,dg] = mcmgrad(x,p1,p2,p3,p4,p5,p6,p7)
df = [];
dg = [];
```

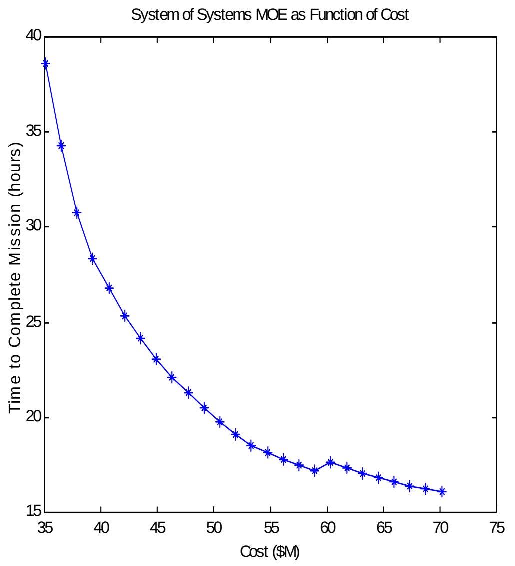
```
09/10/97 4:58 PM      Baseline 9-10-97.doc
output from execution of MCM32.m.

x0=
   10.000    0.900    2.000    9.170   90.000   75.000    6.600   10.000
vlb=
   10.000    0.900    0.250    3.000   42.000   75.000    1.000    3.000
vub=
  100.000    0.980    2.000    9.170   90.000  700.000    7.000   10.000


Run with Costfactor =     1.250   cub=    35.082   qlb=     0.846
Initial Values
f=    93.871   E=    93.871   cost=    28.066      q=     0.620
Final Values
Total time, E=    38.597       cost=    35.082      q=     0.846
x=   57.379    0.963    2.000    4.983   45.108  417.391    7.000    8.793
Function evaluations=      311

Run with Costfactor =     1.300   cub=    36.486   qlb=     0.846
Initial Values
f=    93.871   E=    93.871   cost=    28.066      q=     0.620
Final Values
Total time, E=    34.316       cost=    36.486      q=     0.846
x=   58.915    0.963    2.000    3.915   44.820  417.374    7.000    8.555
Function evaluations=      452

Run with Costfactor =     1.350   cub=    37.889   qlb=     0.846
Initial Values
f=    93.871   E=    93.871   cost=    28.066      q=     0.620
Final Values
Total time, E=    30.747       cost=    37.889      q=     0.846
x=   60.058    0.963    2.000    3.017   44.418  417.371    7.000    8.355
Function evaluations=      374

Run with Costfactor =     1.400   cub=    39.292   qlb=     0.846
Initial Values
f=    93.871   E=    93.871   cost=    28.066      q=     0.620
Final Values
Total time, E=    28.375       cost=    39.292      q=     0.846
x=   65.201    0.963    2.000    3.000   44.632  417.375    7.000    7.162
Function evaluations=      398

Run with Costfactor =     1.450   cub=    40.696   qlb=     0.846
Initial Values
f=    93.871   E=    93.871   cost=    28.066      q=     0.620
Final Values
```

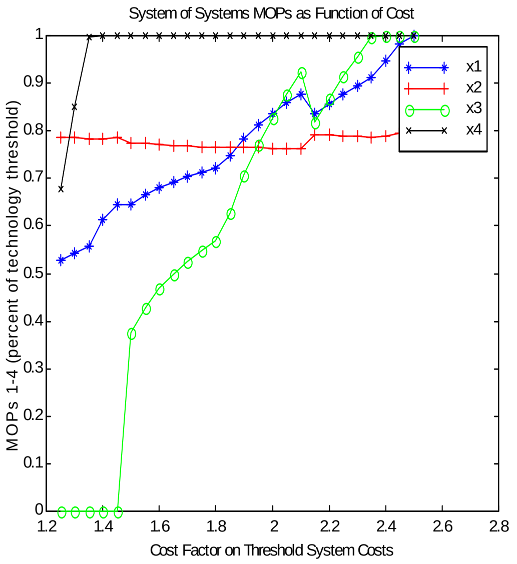
```
Total time, E=    26.831       cost=    40.696      q=     0.846
x=    67.996     0.963     2.000     3.000    44.802   417.387     7.000     6.282
Function evaluations=      430

Run with Costfactor =      1.500   cub=    42.099   qlb=     0.846
Initial Values
f=    93.871   E=    93.871   cost=    28.066      q=     0.620
Final Values
Total time, E=    25.349       cost=    42.099      q=     0.846
x=    68.194     0.962     1.342     3.000    43.341   417.304     7.000     6.213
Function evaluations=      407

Run with Costfactor =      1.550   cub=    43.502   qlb=     0.846
Initial Values
f=    93.871   E=    93.871   cost=    28.066      q=     0.620
Final Values
Total time, E=    24.131       cost=    43.502      q=     0.846
x=    69.825     0.962     1.249     3.000    42.959   417.274     7.000     5.603
Function evaluations=      381

Run with Costfactor =      1.600   cub=    44.906   qlb=     0.846
Initial Values
f=    93.871   E=    93.871   cost=    28.066      q=     0.620
Final Values
Total time, E=    23.073       cost=    44.906      q=     0.846
x=    71.172     0.962     1.181     3.000    42.596   417.237     7.000     5.045
Function evaluations=      394

Run with Costfactor =      1.650   cub=    46.309   qlb=     0.846
Initial Values
f=    93.871   E=    93.871   cost=    28.066      q=     0.620
Final Values
Total time, E=    22.129       cost=    46.309      q=     0.846
x=    72.316     0.961     1.126     3.000    42.245   417.195     7.000     4.532
Function evaluations=      474

Run with Costfactor =      1.700   cub=    47.712   qlb=     0.846
Initial Values
f=    93.871   E=    93.871   cost=    28.066      q=     0.620
Final Values
Total time, E=    21.270       cost=    47.712      q=     0.846
x=    73.310     0.961     1.080     3.000    42.000   417.254     7.000     4.055
Function evaluations=      372

Run with Costfactor =      1.750   cub=    49.115   qlb=     0.846
Initial Values
f=    93.871   E=    93.871   cost=    28.066      q=     0.620
Final Values
Total time, E=    20.478       cost=    49.115      q=     0.846
x=    74.189     0.961     1.040     3.000    42.000   417.519     7.000     3.607
Function evaluations=      397

Run with Costfactor =      1.800   cub=    50.519   qlb=     0.846
Initial Values
f=    93.871   E=    93.871   cost=    28.066      q=     0.620
Final Values
Total time, E=    19.740       cost=    50.519      q=     0.846
x=    74.979     0.961     1.004     3.000    42.000   417.757     7.000     3.184
Function evaluations=      418

Run with Costfactor =      1.850   cub=    51.922   qlb=     0.846
Initial Values
f=    93.871   E=    93.871   cost=    28.066      q=     0.620
Final Values
Total time, E=    19.071       cost=    51.922      q=     0.846
x=    77.328     0.961     0.901     3.000    42.000   418.623     7.000     3.000
Function evaluations=      287
```

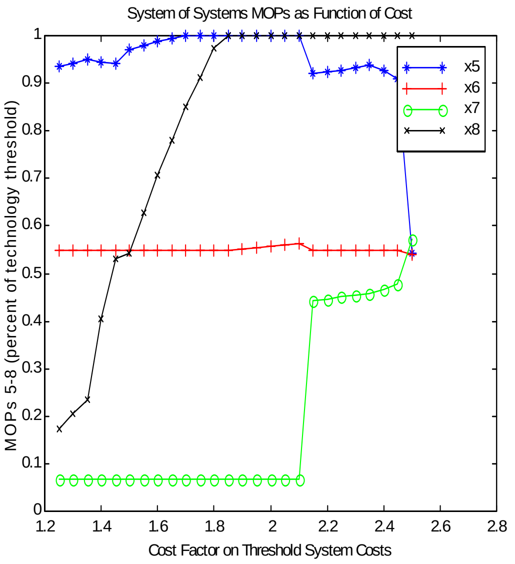
```
Run with Costfactor =     1.900   cub=    53.325   qlb=     0.846
Initial Values
f=    93.871   E=    93.871   cost=    28.066      q=     0.620
Final Values
Total time, E=    18.549       cost=    53.325      q=     0.846
x=    80.440     0.961     0.765     3.000    42.000   420.090     7.000     3.000
Function evaluations=      344

Run with Costfactor =     1.950   cub=    54.729   qlb=     0.846
Initial Values
f=    93.871   E=    93.871   cost=    28.066      q=     0.620
Final Values
Total time, E=    18.125       cost=    54.729      q=     0.846
x=    83.033     0.961     0.651     3.000    42.000   421.627     7.000     3.000
Function evaluations=      368

Run with Costfactor =     2.000   cub=    56.132   qlb=     0.846
Initial Values
f=    93.871   E=    93.871   cost=    28.066      q=     0.620
Final Values
Total time, E=    17.768       cost=    56.132      q=     0.846
x=    85.261     0.961     0.552     3.000    42.000   423.201     7.000     3.000
Function evaluations=      322

Run with Costfactor =     2.050   cub=    57.535   qlb=     0.846
Initial Values
f=    93.871   E=    93.871   cost=    28.066      q=     0.620
Final Values
Total time, E=    17.458       cost=    57.535      q=     0.846
x=    87.217     0.961     0.463     3.000    42.000   424.790     7.000     3.000
Function evaluations=      323

Run with Costfactor =     2.100   cub=    58.939   qlb=     0.846
Initial Values
f=    93.871   E=    93.871   cost=    28.066      q=     0.620
Final Values
Total time, E=    17.183       cost=    58.939      q=     0.846
x=    88.965     0.961     0.383     3.000    42.000   426.399     7.000     3.000
Function evaluations=      338

Run with Costfactor =     2.150   cub=    60.342   qlb=     0.846
Initial Values
f=    93.871   E=    93.871   cost=    28.066      q=     0.620
Final Values
Total time, E=    17.658       cost=    60.342      q=     0.846
x=    85.153     0.963     0.570     3.000    45.839   417.389     3.948     3.000
Function evaluations=      505

Run with Costfactor =     2.200   cub=    61.745   qlb=     0.846
Initial Values
f=    93.871   E=    93.871   cost=    28.066      q=     0.620
Final Values
Total time, E=    17.346       cost=    61.745      q=     0.846
x=    87.149     0.963     0.481     3.000    45.694   417.395     3.921     3.000
Function evaluations=      525

Run with Costfactor =     2.250   cub=    63.148   qlb=     0.846
Initial Values
f=    93.871   E=    93.871   cost=    28.066      q=     0.620
Final Values
Total time, E=    17.070       cost=    63.148      q=     0.846
x=    88.941     0.963     0.400     3.000    45.508   417.412     3.894     3.000
Function evaluations=      482

Run with Costfactor =     2.300   cub=    64.552   qlb=     0.846
Initial Values
```

```
f=    93.871   E=    93.871   cost=    28.066      q=     0.620
Final Values
Total time, E=    16.822       cost=    64.552      q=     0.846
x=    90.539     0.963     0.326     3.000    45.239   417.380     3.876     3.000
Function evaluations=      503

Run with Costfactor =      2.350   cub=    65.955   qlb=     0.846
Initial Values
f=    93.871   E=    93.871   cost=    28.066      q=     0.620
Final Values
Total time, E=    16.596       cost=    65.955      q=     0.846
x=    92.010     0.963     0.257     3.000    44.947   417.360     3.857     3.000
Function evaluations=      488

Run with Costfactor =      2.400   cub=    67.358   qlb=     0.846
Initial Values
f=    93.871   E=    93.871   cost=    28.066      q=     0.620
Final Values
Total time, E=    16.399       cost=    67.358      q=     0.846
x=    95.302     0.963     0.250     3.000    45.572   417.461     3.797     3.000
Function evaluations=      635

Run with Costfactor =      2.450   cub=    68.762   qlb=     0.846
Initial Values
f=    93.871   E=    93.871   cost=    28.066      q=     0.620
Final Values
Total time, E=    16.236       cost=    68.762      q=     0.846
x=    98.430     0.964     0.250     3.000    46.343   417.398     3.737     3.000
Function evaluations=      565

Run with Costfactor =      2.500   cub=    70.165   qlb=     0.846
Initial Values
f=    93.871   E=    93.871   cost=    28.066      q=     0.620
Final Values
Total time, E=    16.122       cost=    70.165      q=     0.846
x=   100.000     0.973     0.250     3.000    64.008   412.202     3.162     3.000
Function evaluations=      430

cost factor   cub         E       cost       qfinal     fun. evals
     1.25    35.082    38.597    35.082     0.846       311
     1.30    36.486    34.316    36.486     0.846       452
     1.35    37.889    30.747    37.889     0.846       374
     1.40    39.292    28.375    39.292     0.846       398
     1.45    40.696    26.831    40.696     0.846       430
     1.50    42.099    25.349    42.099     0.846       407
     1.55    43.502    24.131    43.502     0.846       381
     1.60    44.906    23.073    44.906     0.846       394
     1.65    46.309    22.129    46.309     0.846       474
     1.70    47.712    21.270    47.712     0.846       372
     1.75    49.115    20.478    49.115     0.846       397
     1.80    50.519    19.740    50.519     0.846       418
     1.85    51.922    19.071    51.922     0.846       287
     1.90    53.325    18.549    53.325     0.846       344
     1.95    54.729    18.125    54.729     0.846       368
     2.00    56.132    17.768    56.132     0.846       322
     2.05    57.535    17.458    57.535     0.846       323
     2.10    58.939    17.183    58.939     0.846       338
     2.15    60.342    17.658    60.342     0.846       505
     2.20    61.745    17.346    61.745     0.846       525
     2.25    63.148    17.070    63.148     0.846       482
     2.30    64.552    16.822    64.552     0.846       503
     2.35    65.955    16.596    65.955     0.846       488
     2.40    67.358    16.399    67.358     0.846       635
     2.45    68.762    16.236    68.762     0.846       565
     2.50    70.165    16.122    70.165     0.846       430
```


System of Systems MOE as Function of Cost
Time to Complete Mission (hours)
40
35
30
25
20
15
1.2
1.4
1.6
1.8
2
2.2
2.4
2.6
2.8
Cost Factor on Threshold System Costs


System of Systems MOE as Function of Cost
Time to Complete Mission (hours)
40
35
30
25
20
15
35
40
45
50
55
60
65
70
75
Cost ($M)


System of Systems MOPs as Function of Cost
MOPs 1-4 (percent of technology threshold)
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
1.2
1.4
1.6
1.8
2
2.2
2.4
2.6
2.8
Cost Factor on Threshold System Costs
x1
x2
x3
x4


System of Systems MOPs as Function of Cost
MOPs 5-8 (percent of technology threshold)
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
1.2
1.4
1.6
1.8
2
2.2
2.4
2.6
2.8
Cost Factor on Threshold System Costs
x5
x6
x7
x8

APPENDIX B

SINGLE SYSTEM CONSTRAINED SQP OPTIMIZATION CODE AND RESULTS

MCM System S1—Single System Optimization Code:

```
       %  MCM System One of Two--Single System Optimization
      %  CONSTR w/o Explicit Gradient
      %
      %  Filename:  \matlab\dissertation\MCM32s1.m
      %  Output file initialization
fid=fopen('c:\matlab\dissertation\mcm32s1 output.doc','w');
fprintf(fid,'output from execution of MCM32s1.m\n\n');
      %================================================================================
      %
      %        MCM Initialization--System S1 only
      %
      %================================================================================
      %
      % x(1)=p11=  S1 area coverage rate (nm^2/day)
      % x(2)=p12=  S1 classification probability
      % x(3)=p13=  S1 FAR (#/nm^2)
      % x(4)=p14=  S1 time to classify (min)
      % x(5)=p15=  S1 navigation accuracy (yards)
s2x6(1)=417.352;s2x6(2)=417.320;s2x6(3)=417.268;s2x6(4)=417.201;s2x6(5)=418.311;
s2x6(6)=419.238;s2x6(7)=419.491;s2x6(8)=420.166;s2x6(9)=420.781;s2x6(10)=421.338;
s2x6(11)=421.861;s2x6(12)=422.339;s2x6(13)=422.792;s2x6(14)=424.387;s2x6(15)=426.622;
s2x6(16)=428.734;s2x6(17)=430.719;s2x6(18)=432.599;s2x6(19)=434.390;s2x6(20)=418.950;
s2x6(21)=419.408;s2x6(22)=419.837;s2x6(23)=420.239;s2x6(24)=420.626;s2x6(25)=421.301;
s2x6(26)=421.916;s2x6(27)=416.997;
pd=0.90;     %S1 detection system probability of detection
x0=[10,0.9,2.0,9.17,90]; %Might need to start at feasible point.  This won't meet q(x)
constraint.
fprintf(fid,'x0=\n');fprintf(fid,'%10.3f',x0);fprintf(fid,'\n');
vlb=[10.0,0.9,0.25,3.0,42];           % lower bound constraint on x
vub=[100,0.98,2.0,9.17,90];           % upper bound constraint on x
fprintf(fid,'vlb=\n');fprintf(fid,'%10.3f',vlb);fprintf(fid,'\n');
fprintf(fid,'vub=\n');fprintf(fid,'%10.3f',vub);fprintf(fid,'\n\n');
%  Compute px1, polynomial fit cost function parameters for x(1)
x1=[10,57,82,94];
y1=[3,4.483,7.655,11.445];
px1=polyfit(x1,y1,3);
%  Compute px2, polynomial fit cost function parameters for x(2)
x1=[0.9,0.93,0.96,0.98];
y1=[.2,.5,1.4,2.2];  %revised PBCM that reflects COTS/NDI development
px2=polyfit(x1,y1,2);
%x2=0.9:.01:1.0;
%  Compute px3, polynomial fit cost function parameters for x(3)
x1=[2,1,0.5,0.25];
y1=[8,10.319,13.592,16.429];
px3=polyfit(x1,y1,3);
%  Compute px4, polynomial fit cost function parameters for x(4)
x1=[3.513,3.89,4.77,9.17];
y1=[9.191,8.190,7.574,5.0];
px4=polyfit(x1,y1,2);
%  Compute px5, polynomial fit cost function parameters for x(5)
x1=[90,60,48,42];
y1=[0.050,0.250,0.450,0.550];
px5=polyfit(x1,y1,2);
%compute threshold system cost
cost0=polyval(px1,x0(1))+polyval(px2,x0(2))+polyval(px3,x0(3))+polyval(px4,x0(4))+polyval(
px5,x0(5));
costfactor=1.5;  %nominal value
%Insert costfactor loop
i=0
for costfactor=1.2:0.05:2.5 %Note problem is infeasible with costfactor<1.25
   i=i+1
   x6=s2x6(i)
   x6=300%Hardwired to "reasonable", knee-of-the-curve guess
   costfactor
   cub=costfactor*cost0         %system of systems cost constraint
   qlb=0.846   %quality constraint lower bound
   fprintf(fid,'\n');
   fprintf(fid,'Run with Costfactor = ');
   fprintf(fid,'%10.3f',costfactor);fprintf(fid,'   cub=');
   fprintf(fid,'%10.3f',cub);fprintf(fid,'   qlb=');fprintf(fid,'%10.3f\n',qlb);
sminefield=20;        %minefield area, (nm^2)
m0=100;               %number of mines in minefield, initially
lambda=m0/sminefield; %mine density, (#/nm^2)
dmine=600;            %average distance between mines, (yards)
vtransit=7;           %S1 vehicle transit speed (knots)
ttransit=dmine/(2000*vtransit);    %transit time during classification (hours)
lambdaft=1.0;         %false target density (#/nm^2)
% Define parameters for the objective function
p1=pd;
p2=lambda;
p3=lambdaft;
p4=ttransit;
p5=sminefield;
p6=cub;
p7=qlb;
[f,g]=mcmfuns1(x0,p1,p2,p3,p4,p5,p6,p7,x6)
fprintf(fid,'Initial Values\n');
x=x0;
cost=polyval(px1,x(1))+polyval(px2,x(2))+polyval(px3,x(3))+polyval(px4,x(4))+polyval(px5,x
(5))
q=p1*x(2)*exp(-x(5)/(4.481*x6));
f1 = (p5/60)*(24*60/x(1) + p2*x(2)*x(4)*p1 + (2*x(4)-p4)*((1-x(2))*p1*p2 + x(3) + p1*p3));
E=f1;
q=p1*x0(2)*exp(-x0(5)/(4.481*x6));
fprintf(fid,'f=');fprintf(fid,'%10.3f',f);fprintf(fid,'   E=');fprintf(fid,'%10.3f',E);
fprintf(fid,'   cost=');fprintf(fid,'%10.3f',cost);fprintf(fid,'
q=');fprintf(fid,'%10.3f\n',q);
%  Print out threshold system values for f and constraints, g.
      %================================================================================
      %
      %        CONSTR Initialization and Call
      %
      %================================================================================
      % Modify x0 so that it satisfies initial quality constraint, g(2)
       %x0(2)=0.96
       %x0(5)=45
       %x0(6)=600
[f,g]=mcmfuns1(x0,p1,p2,p3,p4,p5,p6,p7,x6)
%  Print out initial system values for f and constraints, g.
grad=[];          % need to set to null matrix in order to pass p1....p7 to mcmfun
options(1)=1;     % print output table
%options(2)=1e-5;  % relax x termination criteria
%options(3)=1e-3;  % relax f termination criteria
%options(4)=1e-5;  % relax constraint violation limits
options(9) = 0;   % if =1, check analytic gradient
[x,options]=constr('mcmfuns1',x0,options,vlb,vub,'mcmgrads1',p1,p2,p3,p4,p5,p6,p7,x6)
[f,g]=mcmfuns1(x,p1,p2,p3,p4,p5,p6,p7,x6)
fprintf(fid,'Final Values\n');
cost =
polyval(px1,x(1))+polyval(px2,x(2))+polyval(px3,x(3))+polyval(px4,x(4))+polyval(px5,x(5))
q=p1*x(2)*exp(-x(5)/(4.481*x6))
f1 = (p5/60)*(24*60/x(1) + p2*x(2)*x(4)*p1 + (2*x(4)-p4)*((1-x(2))*p1*p2 + x(3) + p1*p3));
E=f1
fprintf(fid,'S1 Recon Time, E=');fprintf(fid,'%10.3f',E);
fprintf(fid,'       cost=');fprintf(fid,'%10.3f',cost);fprintf(fid,'
q=');fprintf(fid,'%10.3f\n',q);
fprintf(fid,'x=');fprintf(fid,'%10.3f',x);fprintf(fid,'\n');
fprintf(fid,'Function evaluations= ');fprintf(fid,'%8.0f\n',options(10));
cf(i)=costfactor; fval(i)=f;
costval(i)=cub;fevals(i)=options(10);costi(i)=cost;qfinal(i)=q;
z1(i)=x(1);z2(i)=x(2);z3(i)=x(3);z4(i)=x(4);z5(i)=x(5);
z1(i)=abs((z1(i)-x0(1))/(vlb(1)-vub(1)));
z2(i)=abs((z2(i)-x0(2))/(vlb(2)-vub(2)));
z3(i)=abs((z3(i)-x0(3))/(vlb(3)-vub(3)));
z4(i)=abs((z4(i)-x0(4))/(vlb(4)-vub(4)));
z5(i)=abs((z5(i)-x0(5))/(vlb(5)-vub(5)));
end
%  Plot option 1:  Plot System of systems MOE as CAIV
figure
```

```
plot(cf,fval,'-*b')
title('System S1 MOE as Function of Cost')
xlabel('Cost Factor on Threshold System Costs')
ylabel('Time to Complete Mission (hours)')
figure
plot(costval,fval,'-*b')
title('System S1 MOE as Function of Cost')
xlabel('Cost ($M)')
ylabel('Time to Complete Mission (hours)')
%
%
%  Plot option 2:  Plot MOPs as CAIV
figure
plot(cf,z1,'-b*',cf,z2,'-r+',cf,z3,'-go',cf,z4,'-kx',cf,z5,'-m.')
legend('x1','x2','x3','x4','x5')
title('System S1 MOPs as Function of Cost')
xlabel('Cost Factor on Threshold System Costs')
ylabel('MOPs 1-5 (percent of technology threshold)')
%  print table of results to file
fprintf(fid,'\n');
fprintf(fid,'cost factor');fprintf(fid,'   cub');fprintf(fid,'          E');fprintf(fid,'
cost');fprintf(fid,'        qfinal');fprintf(fid,'      fun. evals\n');
for j=1:i
fprintf(fid,'%10.2f',cf(j));fprintf(fid,'%10.3f',costval(j));
fprintf(fid,'%10.3f',fval(j));fprintf(fid,'%10.3f',costi(j));
fprintf(fid,'%10.3f',qfinal(j));fprintf(fid,'%10.0f\n',fevals(j));
end
status=fclose(fid)
```

```
function [f,g] = mcmfun(x,p1,p2,p3,p4,p5,p6,p7,x6)
% Function to just optimize system S1 alone, with S2's parameter x6 passed by argument
%updated Px2, 7/3/97
px1=[4.503408803940725e-005,   -5.386095666335044e-003,    2.159330101073730e-001,
       1.334245377520354e+000];
px2=[2.834645669291690e+002,   -5.076377952756583e+002,    2.274598425197177e+002];
px3=[-2.048380952380911e+000,    9.987333333333214e+000,   -1.794233333333325e+001,
       2.032238095238094e+001];
px4=[1.159691730856429e-001,   -2.175732453433467e+000,    1.520381256718985e+001];
px5=[2.061825086032983e-004,   -3.775958229500408e-002,    1.777803488786043e+000];
f1 = (p5/60)*(24*60/x(1) + p2*x(2)*x(4)*p1 + (2*x(4)-p4)*((1-x(2))*p1*p2 + x(3) + p1*p3));
f=f1;
% evaluate cost constraint
g(1) =
polyval(px1,x(1))+polyval(px2,x(2))+polyval(px3,x(3))+polyval(px4,x(4))+polyval(px5,x(5))-
p6;
% evaluate negative of quality constraint
g(2) = -p1*x(2)*exp(-x(5)/(4.481*x6))+p7;
%p1=pd
%p2=lambda
%p3=lambdaft
%p4=ttransit
%p5=sminefield
%p6=cub
%p7=qlb
```

S1 Optimization Results with Imperfect Knowledge of S2:

```
output from execution of MCM32s1.m
S1 Imperfect.doc   09/14/97 8:19 PM

x0=
   10.000    0.900    2.000    9.170   90.000
vlb=
   10.000    0.900    0.250    3.000   42.000
vub=
  100.000    0.980    2.000    9.170   90.000


Run with Costfactor =      1.250   cub=    20.307   qlb=     0.846
Initial Values
f=    80.811   E=    80.811   cost=    16.246     q=    0.778
```

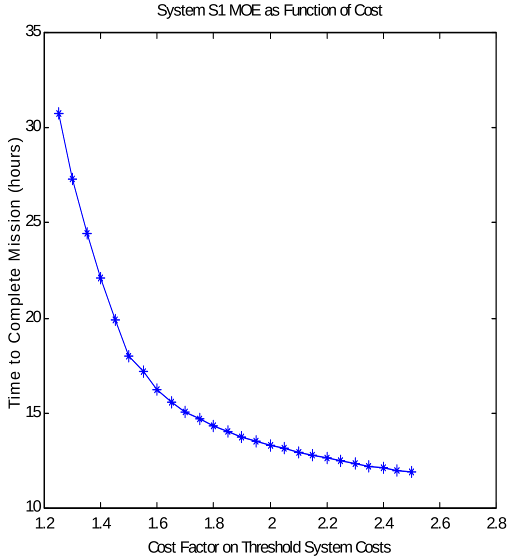
```
Final Values
S1 Recon Time, E=    30.766       cost=    20.307      q=     0.846
x=    55.007     0.963     2.000     6.329    53.739
Function evaluations=      115

Run with Costfactor =      1.300   cub=    21.119   qlb=     0.846
Initial Values
f=    80.811    E=    80.811    cost=    16.246      q=     0.778
Final Values
S1 Recon Time, E=    27.286       cost=    21.119      q=     0.846
x=    56.693     0.963     2.000     5.406    53.912
Function evaluations=      175

Run with Costfactor =      1.350   cub=    21.932   qlb=     0.846
Initial Values
f=    80.811    E=    80.811    cost=    16.246      q=     0.778
Final Values
S1 Recon Time, E=    24.478       cost=    21.932      q=     0.846
x=    57.880     0.963     2.000     4.651    53.954
Function evaluations=      172

Run with Costfactor =      1.400   cub=    22.744   qlb=     0.846
Initial Values
f=    80.811    E=    80.811    cost=    16.246      q=     0.778
Final Values
S1 Recon Time, E=    22.061       cost=    22.744      q=     0.846
x=    58.808     0.963     2.000     3.996    53.912
Function evaluations=      148

Run with Costfactor =      1.450   cub=    23.556   qlb=     0.846
Initial Values
f=    80.811    E=    80.811    cost=    16.246      q=     0.778
Final Values
S1 Recon Time, E=    19.909       cost=    23.556      q=     0.846
x=    59.575     0.963     2.000     3.409    53.816
Function evaluations=      136

Run with Costfactor =      1.500   cub=    24.369   qlb=     0.846
Initial Values
f=    80.811    E=    80.811    cost=    16.246      q=     0.778
Final Values
S1 Recon Time, E=    18.028       cost=    24.369      q=     0.846
x=    63.112     0.963     2.000     3.000    54.410
Function evaluations=      133

Run with Costfactor =      1.550   cub=    25.181   qlb=     0.846
Initial Values
f=    80.811    E=    80.811    cost=    16.246      q=     0.778
Final Values
S1 Recon Time, E=    17.152       cost=    25.181      q=     0.846
x=    66.920     0.964     1.778     3.000    55.570
Function evaluations=      188

Run with Costfactor =      1.600   cub=    25.993   qlb=     0.846
Initial Values
f=    80.811    E=    80.811    cost=    16.246      q=     0.778
Final Values
S1 Recon Time, E=    16.266       cost=    25.993      q=     0.846
x=    67.955     0.964     1.387     3.000    55.945
Function evaluations=      174

Run with Costfactor =      1.650   cub=    26.805   qlb=     0.846
Initial Values
f=    80.811    E=    80.811    cost=    16.246      q=     0.778
Final Values
S1 Recon Time, E=    15.592       cost=    26.805      q=     0.846
x=    71.112     0.964     1.207     3.000    57.276
```

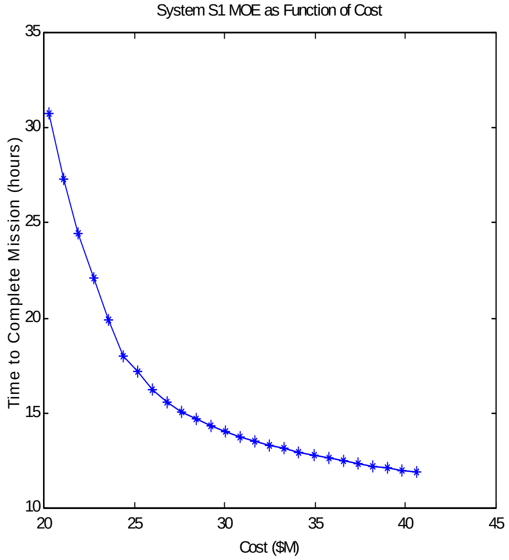
```
Function evaluations=      155

Run with Costfactor =      1.700   cub=    27.618   qlb=     0.846
Initial Values
f=    80.811   E=    80.811   cost=    16.246      q=     0.778
Final Values
S1 Recon Time, E=    15.081       cost=    27.618      q=     0.846
x=    73.868     0.965     1.078     3.000    58.688
Function evaluations=      187

Run with Costfactor =      1.750   cub=    28.430   qlb=     0.846
Initial Values
f=    80.811   E=    80.811   cost=    16.246      q=     0.778
Final Values
S1 Recon Time, E=    14.670       cost=    28.430      q=     0.846
x=    76.226     0.966     0.973     3.000    60.101
Function evaluations=      191

Run with Costfactor =      1.800   cub=    29.242   qlb=     0.846
Initial Values
f=    80.811   E=    80.811   cost=    16.246      q=     0.778
Final Values
S1 Recon Time, E=    14.326       cost=    29.242      q=     0.846
x=    78.274     0.966     0.885     3.000    61.499
Function evaluations=      183

Run with Costfactor =      1.850   cub=    30.055   qlb=     0.846
Initial Values
f=    80.811   E=    80.811   cost=    16.246      q=     0.778
Final Values
S1 Recon Time, E=    14.030       cost=    30.055      q=     0.846
x=    80.083     0.967     0.807     3.000    62.876
Function evaluations=      179

Run with Costfactor =      1.900   cub=    30.867   qlb=     0.846
Initial Values
f=    80.811   E=    80.811   cost=    16.246      q=     0.778
Final Values
S1 Recon Time, E=    13.769       cost=    30.867      q=     0.846
x=    81.703     0.967     0.736     3.000    64.224
Function evaluations=      214

Run with Costfactor =      1.950   cub=    31.679   qlb=     0.846
Initial Values
f=    80.811   E=    80.811   cost=    16.246      q=     0.778
Final Values
S1 Recon Time, E=    13.536       cost=    31.679      q=     0.846
x=    83.174     0.968     0.673     3.000    65.563
Function evaluations=      207

Run with Costfactor =      2.000   cub=    32.491   qlb=     0.846
Initial Values
f=    80.811   E=    80.811   cost=    16.246      q=     0.778
Final Values
S1 Recon Time, E=    13.325       cost=    32.491      q=     0.846
x=    84.519     0.968     0.614     3.000    66.873
Function evaluations=      172

Run with Costfactor =      2.050   cub=    33.304   qlb=     0.846
Initial Values
f=    80.811   E=    80.811   cost=    16.246      q=     0.778
Final Values
S1 Recon Time, E=    13.132       cost=    33.304      q=     0.846
x=    85.760     0.969     0.559     3.000    68.159
Function evaluations=      190

Run with Costfactor =      2.100   cub=    34.116   qlb=     0.846
```

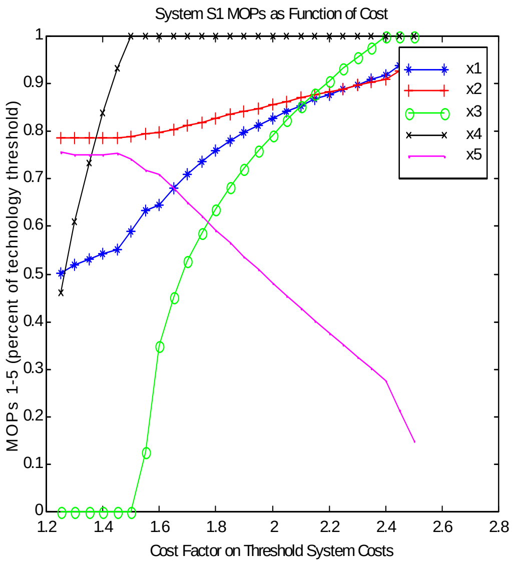
```
Initial Values
f=   80.811   E=   80.811   cost=   16.246     q=    0.778
Final Values
S1 Recon Time, E=   12.953      cost=   34.116     q=    0.846
x=   86.914    0.970    0.508    3.000   69.435
Function evaluations=     189

Run with Costfactor =     2.150   cub=   34.928   qlb=    0.846
Initial Values
f=   80.811   E=   80.811   cost=   16.246     q=    0.778
Final Values
S1 Recon Time, E=   12.787      cost=   34.928     q=    0.846
x=   87.991    0.970    0.460    3.000   70.690
Function evaluations=     199

Run with Costfactor =     2.200   cub=   35.741   qlb=    0.846
Initial Values
f=   80.811   E=   80.811   cost=   16.246     q=    0.778
Final Values
S1 Recon Time, E=   12.632      cost=   35.741     q=    0.846
x=   89.003    0.971    0.414    3.000   71.926
Function evaluations=     204

Run with Costfactor =     2.250   cub=   36.553   qlb=    0.846
Initial Values
f=   80.811   E=   80.811   cost=   16.246     q=    0.778
Final Values
S1 Recon Time, E=   12.486      cost=   36.553     q=    0.846
x=   89.957    0.971    0.370    3.000   73.145
Function evaluations=     212

Run with Costfactor =     2.300   cub=   37.365   qlb=    0.846
Initial Values
f=   80.811   E=   80.811   cost=   16.246     q=    0.778
Final Values
S1 Recon Time, E=   12.348      cost=   37.365     q=    0.846
x=   90.859    0.972    0.329    3.000   74.355
Function evaluations=     154

Run with Costfactor =     2.350   cub=   38.177   qlb=    0.846
Initial Values
f=   80.811   E=   80.811   cost=   16.246     q=    0.778
Final Values
S1 Recon Time, E=   12.218      cost=   38.177     q=    0.846
x=   91.716    0.972    0.289    3.000   75.546
Function evaluations=     190

Run with Costfactor =     2.400   cub=   38.990   qlb=    0.846
Initial Values
f=   80.811   E=   80.811   cost=   16.246     q=    0.778
Final Values
S1 Recon Time, E=   12.094      cost=   38.990     q=    0.846
x=   92.532    0.973    0.251    3.000   76.727
Function evaluations=     157

Run with Costfactor =     2.450   cub=   39.802   qlb=    0.846
Initial Values
f=   80.811   E=   80.811   cost=   16.246     q=    0.778
Final Values
S1 Recon Time, E=   11.979      cost=   39.802     q=    0.846
x=   94.477    0.974    0.250    3.000   79.711
Function evaluations=     123

Run with Costfactor =     2.500   cub=   40.614   qlb=    0.846
Initial Values
f=   80.811   E=   80.811   cost=   16.246     q=    0.778
Final Values
```

```
S1 Recon Time, E=    11.877       cost=    40.614      q=     0.846
x=    96.309     0.975     0.250     3.000    82.769
```

```
Function evaluations=      125
cost factor   cub          E      cost        qfinal      fun. evals
     1.25    20.307    30.766    20.307     0.846        115
     1.30    21.119    27.286    21.119     0.846        175
     1.35    21.932    24.478    21.932     0.846        172
     1.40    22.744    22.061    22.744     0.846        148
     1.45    23.556    19.909    23.556     0.846        136
     1.50    24.369    18.028    24.369     0.846        133
     1.55    25.181    17.152    25.181     0.846        188
     1.60    25.993    16.266    25.993     0.846        174
     1.65    26.805    15.592    26.805     0.846        155
     1.70    27.618    15.081    27.618     0.846        187
     1.75    28.430    14.670    28.430     0.846        191
     1.80    29.242    14.326    29.242     0.846        183
     1.85    30.055    14.030    30.055     0.846        179
     1.90    30.867    13.769    30.867     0.846        214
     1.95    31.679    13.536    31.679     0.846        207
     2.00    32.491    13.325    32.491     0.846        172
     2.05    33.304    13.132    33.304     0.846        190
     2.10    34.116    12.953    34.116     0.846        189
     2.15    34.928    12.787    34.928     0.846        199
     2.20    35.741    12.632    35.741     0.846        204
     2.25    36.553    12.486    36.553     0.846        212
     2.30    37.365    12.348    37.365     0.846        154
     2.35    38.177    12.218    38.177     0.846        190
     2.40    38.990    12.094    38.990     0.846        157
     2.45    39.802    11.979    39.802     0.846        123
     2.50    40.614    11.877    40.614     0.846        125
```







MCM System S2—Single System Optimization Code:

```
%  MCM System Two of Two--Single System Optimization
      %  CONSTR w/o Explicit Gradient
      %
      %  Filename:  \matlab\dissertation\MCM32s2.m
      %  Output file initialization
fid=fopen('c:\matlab\dissertation\mcm32s2 output.doc','w');
fprintf(fid,'output from execution of MCM32s2.m\n\n');
      %================================================================================
      %
      %       MCM Initialization
      %
      %================================================================================
      % x2=p12=  S1 classification probability
      % x3=p13=  S1 FAR (#/nm^2)
      % x5=p15=  S1 navigation accuracy (yards)
      % x(6)=p21=  S2 Re-acquisition range (yards)
      % x(7)=p22=  S2 time to prosecute a false target (min)
      % x(8)=p23=  S3 time to neutralize (min)
      pd=0.90;     %S1 detection system probability of detection
x0=[0,0,0,0,0,75,6.6,10.0]; %Might need to start at feasible point.  This won't meet q(x)
constraint.
fprintf(fid,'x0=\n');fprintf(fid,'%10.3f',x0);fprintf(fid,'\n');
vlb=[0,0,0,0,0,75,1.0,3.0];           % lower bound constraint on x
vub=[0,0,0,0,0,700,7.0,10.0];         % upper bound constraint on x
%Try to get away with not re-coding x6-x8, so set up dummy x1-x5 with zerio initial
conditions and bounds
fprintf(fid,'vlb=\n');fprintf(fid,'%10.3f',vlb);fprintf(fid,'\n');
fprintf(fid,'vub=\n');fprintf(fid,'%10.3f',vub);fprintf(fid,'\n\n');
%
s2x2(1)=.963;s2x2(2)=.963;s2x2(3)=.963;s2x2(4)=.963;s2x2(5)=.963;
s2x2(6)=.962;s2x2(7)=.962;s2x2(8)=.962;s2x2(9)=.961;s2x2(10)=.961;
s2x2(11)=.961;s2x2(12)=.961;s2x2(13)=.961;s2x2(14)=.961;s2x2(15)=.961;
s2x2(16)=.961;s2x2(17)=.961;s2x2(18)=.961;s2x2(19)=.963;s2x2(20)=.963;
s2x2(21)=.963;s2x2(22)=.963;s2x2(23)=.963;s2x2(24)=.963;s2x2(25)=.964;
s2x2(26)=.973;
s2x3(1)=2;s2x3(2)=2;s2x3(3)=2;s2x3(4)=2;s2x3(5)=2;
s2x3(6)=1.342;s2x3(7)=1.249;s2x3(8)=1.181;s2x3(9)=1.126;s2x3(10)=1.080;
s2x3(11)=1.040;s2x3(12)=1.004;s2x3(13)=0.901;s2x3(14)=0.765;s2x3(15)=0.651;
s2x3(16)=.552;s2x3(17)=.463;s2x3(18)=.383;s2x3(19)=.570;s2x3(20)=.481;
s2x3(21)=.400;s2x3(22)=.326;s2x3(23)=.257;s2x3(24)=.250;s2x3(25)=.250;
s2x3(26)=.250;
s2x5(1)=45.108;s2x5(2)=44.820;s2x5(3)=44.418;s2x5(4)=44.632;s2x5(5)=44.802;
s2x5(6)=43.341;s2x5(7)=43.502;s2x5(8)=42.596;s2x5(9)=42.245;s2x5(10)=42;
s2x5(11)=42;s2x5(12)=42;s2x5(13)=42;s2x5(14)=42;s2x5(15)=42;
s2x5(16)=42;s2x5(17)=42;s2x5(18)=42;s2x5(19)=45.839;s2x5(20)=45.694;
```

```
s2x5(21)=45.508;s2x5(22)=45.239;s2x5(23)=44.947;s2x5(24)=45.575;s2x5(25)=46.343;
s2x5(26)=64.008;
%  Compute px6, polynomial fit cost function parameters for x(6)
x1=[129,457,622,75];
y1=[3,4.8,7.655,1.5];
px6=polyfit(x1,y1,3);
%  Compute px7, polynomial fit cost function parameters for x(7)
x1=[6.6,4.4,3.3,2.64,1.32];
y1=[5.0,7.5,8.190,9.191,16.621];
px7=polyfit(x1,y1,3);
%  Compute px8, polynomial fit cost function parameters for x(8)
x1=[10,8,7,5,3];
y1=[5.3,6,7,10,15];
px8=polyfit(x1,y1,2);
%compute threshold system cost
%
cost0=polyval(px6,x0(6))+polyval(px7,x0(7))+polyval(px8,x0(8))
costfactor=1.5;  %nominal value
%Insert costfactor loop
i=0
for costfactor=1.25:0.05:2.5 %Note problem is infeasible with costfactor<1.25
   i=i+1
   x2=s2x2(i);x3=s2x3(i);x5=s2x5(i);
   x2=.958;x3=1.0;x5=55;
   costfactor
   cub=costfactor*cost0         %system of systems cost constraint
   qlb=0.846             %quality constraint lower bound
   fprintf(fid,'\n');
   fprintf(fid,'Run with Costfactor = ');fprintf(fid,'%10.3f',costfactor);fprintf(fid,'
cub=');fprintf(fid,'%10.3f',cub);fprintf(fid,'   qlb=');fprintf(fid,'%10.3f\n',qlb);
sminefield=20;        %minefield area, (nm^2)
m0=100;               %number of mines in minefield, initially
lambda=m0/sminefield; %mine density, (#/nm^2)
dmine=600;            %average distance between mines, (yards)
vtransit=7;           %S1 vehicle transit speed (knots)
ttransit=dmine/(2000*vtransit);    %transit time during classification (hours)
lambdaft=1.0;         %false target density (#/nm^2)
% Define parameters for the objective function
p1=pd;
p2=lambda;
p3=lambdaft;
p4=ttransit;
p5=sminefield;
p6=cub;
p7=qlb;
[f,g]=mcmfuns2(x0,p1,p2,p3,p4,p5,p6,p7,x2,x3,x5)
fprintf(fid,'Initial Values\n');
x=x0;
      cost = polyval(px6,x(6))+polyval(px7,x(7))+polyval(px8,x(8))
      q=p1*x2*exp(-x5/(4.481*x(6)));
      f2 = (p5/60)*(p1*x2*x(8)*p2*exp(-x5/(4.481*x(6))))+(1-exp(-
x5/(4.481*x(6))))*p1*x2*x(7)*p2 + (1-x2)*(x3+p1*p3)*x(7));
      E=f2;
      q=p1*x2*exp(-x5/(4.481*x0(6)));
fprintf(fid,'f=');fprintf(fid,'%10.3f',f);fprintf(fid,'   E=');
fprintf(fid,'%10.3f',E);fprintf(fid,'   cost=');
fprintf(fid,'%10.3f',cost);fprintf(fid,'      q=');fprintf(fid,'%10.3f\n',q);
%  Print out threshold system values for f and constraints, g.
      %===============================================================================
      %
      %        CONSTR Initialization and Call
      %
      %===============================================================================
[f,g]=mcmfuns2(x0,p1,p2,p3,p4,p5,p6,p7,x2,x3,x5)
%  Print out initial system values for f and constraints, g.
grad=[];          % need to set to null matrix in order to pass p1....p7 to mcmfun
options(1)=1;     % print output table
%options(2)=1e-5;  % relax x termination criteria
options(3)=1e-5;  % relax f termination criteria
%options(4)=1e-5;  % relax constraint violation limits
options(9) = 0;   % if =1, check analytic gradient
[x,options]=constr('mcmfuns2',x0,options,vlb,vub,'mcmgrads2',p1,p2,p3,p4,p5,p6,p7,x2,x3,x5
)
[f,g]=mcmfuns2(x,p1,p2,p3,p4,p5,p6,p7,x2,x3,x5)
fprintf(fid,'Final Values\n');
```

```
cost = polyval(px6,x(6))+polyval(px7,x(7))+polyval(px8,x(8))
q=p1*x2*exp(-x5/(4.481*x(6)))
f2 = (p5/60)*(p1*x2*x(8)*p2*exp(-x5/(4.481*x(6)))+(1-exp(-x5/(4.481*x(6))))*p1*x2*x(7)*p2
+ (1-x2)*(x3+p1*p3)*x(7));
E=f2
fprintf(fid,'S2 Clearance time, E=');fprintf(fid,'%10.3f',E);
fprintf(fid,'       cost=');fprintf(fid,'%10.3f',cost);fprintf(fid,'
q=');fprintf(fid,'%10.3f\n',q);
fprintf(fid,'x=');fprintf(fid,'%10.3f',x);fprintf(fid,'\n');
fprintf(fid,'Function evaluations= ');fprintf(fid,'%8.0f\n',options(10));
cf(i)=costfactor; fval(i)=f;
costval(i)=cub;fevals(i)=options(10);costi(i)=cost;qfinal(i)=q;
z6(i)=x(6);z7(i)=x(7);z8(i)=x(8);
z6(i)=abs((z6(i)-x0(6))/(vlb(6)-vub(6)));
z7(i)=abs((z7(i)-x0(7))/(vlb(7)-vub(7)));
z8(i)=abs((z8(i)-x0(8))/(vlb(8)-vub(8)));
end
%  Plot option 1:  Plot System of systems MOE as CAIV
figure
plot(cf,fval,'-*b')
title('System S2 MOE as Function of Cost')
xlabel('Cost Factor on Threshold System Costs')
ylabel('Time to Complete Mission (hours)')
figure
plot(costval,fval,'-*b')
title('System S2 MOE as Function of Cost')
xlabel('Cost ($M)')
ylabel('Time to Complete Mission (hours)')
%
%
%  Plot option 2:  Plot MOPs as CAIV
figure
plot(cf,z6,'-b*',cf,z7,'-r+',cf,z8,'-go')
legend('x6','x7','x8')
title('System S2 MOPs as Function of Cost')
xlabel('Cost Factor on Threshold System Costs')
ylabel('MOPs 6-8 (percent of technology threshold)')
%  print table of results to file
fprintf(fid,'\n');
fprintf(fid,'cost factor');fprintf(fid,'   cub');fprintf(fid,'          E');fprintf(fid,'
cost');fprintf(fid,'        qfinal');fprintf(fid,'      fun. evals\n');
for j=1:i
fprintf(fid,'%10.2f',cf(j));fprintf(fid,'%10.3f',costval(j));
fprintf(fid,'%10.3f',fval(j));fprintf(fid,'%10.3f',costi(j));
fprintf(fid,'%10.3f',qfinal(j));fprintf(fid,'%10.0f\n',fevals(j));
end
status=fclose(fid)


function [f,g] = mcmfuns2(x,p1,p2,p3,p4,p5,p6,p7,x2,x3,x5)
%MCM function for optimizing system S2 only
px6=[1.504875482450802e-007,   -1.578229837871938e-004,    5.516694369186904e-002,  -
       1.813253427503106e+000];
px7=[ -2.850358103957624e-001,    3.846213159671302e+000,   -1.726423877731832e+001,
       3.334408692656030e+001];
px8=[2.102445277065673e-001,   -4.109593768487483e+000,    2.539723920331297e+001];
f2 = (p5/60)*(p1*x2*x(8)*p2*exp(-x5/(4.481*x(6)))+(1-exp(-x5/(4.481*x(6))))*p1*x2*x(7)*p2
+ (1-x2)*(x3+p1*p3));
f=f2;
% evaluate cost constraint
g(1) = polyval(px6,x(6))+polyval(px7,x(7))+polyval(px8,x(8))-p6;
% evaluate negative of quality constraint
g(2) = -p1*x2*exp(-x5/(4.481*x(6)))+p7;
%p1=pd
%p2=lambda
%p3=lambdaft
%p4=ttransit
%p5=sminefield
%p6=cub
%p7=qlb
```

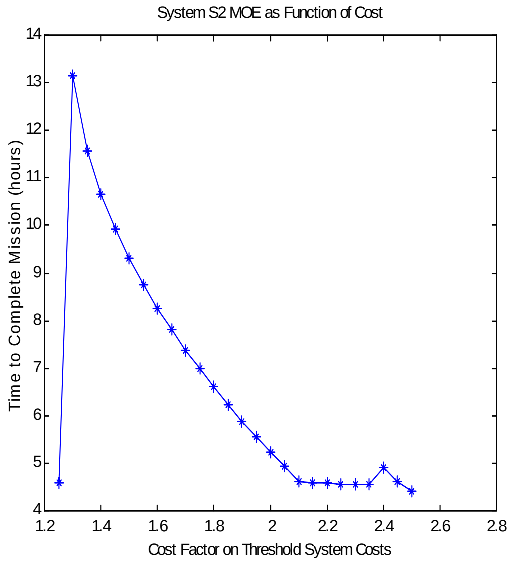
S2 Optimization Results with Imperfect Knowledge of S1:

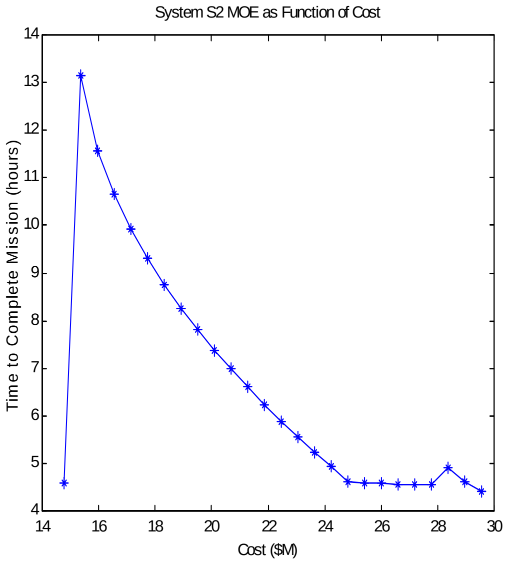
```
output from execution of MCM32s2.m
S2 Imperfect.doc      09/14/97 9:14 PM

x0=
    0.000     0.000     0.000     0.000     0.000    75.000     6.600    10.000
vlb=
    0.000     0.000     0.000     0.000     0.000    75.000     1.000     3.000
vub=
    0.000     0.000     0.000     0.000     0.000   700.000     7.000    10.000


Run with Costfactor =      1.250   cub=    14.775   qlb=     0.846
Initial Values
f=    13.941   E=    13.941   cost=    11.820      q=     0.759
Final Values
S2 Clearance time, E=     4.585       cost=    25.086      q=     0.846
x=     0.000     0.000     0.000     0.000     0.000   592.043     6.966     3.000
Function evaluations=     504

Run with Costfactor =      1.300   cub=    15.366   qlb=     0.846
Initial Values
f=    13.941   E=    13.941   cost=    11.820      q=     0.759
Final Values
S2 Clearance time, E=    13.155       cost=    15.366      q=     0.846
x=     0.000     0.000     0.000     0.000     0.000   592.043     7.000     9.077
Function evaluations=      91

Run with Costfactor =      1.350   cub=    15.957   qlb=     0.846
Initial Values
f=    13.941   E=    13.941   cost=    11.820      q=     0.759
Final Values
S2 Clearance time, E=    11.578       cost=    15.957      q=     0.846
x=     0.000     0.000     0.000     0.000     0.000   592.043     7.000     7.958
Function evaluations=      82

Run with Costfactor =      1.400   cub=    16.548   qlb=     0.846
Initial Values
f=    13.941   E=    13.941   cost=    11.820      q=     0.759
Final Values
S2 Clearance time, E=    10.653       cost=    16.548      q=     0.846
x=     0.000     0.000     0.000     0.000     0.000   592.043     7.000     7.302
Function evaluations=      82

Run with Costfactor =      1.450   cub=    17.139   qlb=     0.846
Initial Values
f=    13.941   E=    13.941   cost=    11.820      q=     0.759
Final Values
S2 Clearance time, E=     9.927       cost=    17.139      q=     0.846
x=     0.000     0.000     0.000     0.000     0.000   592.043     7.000     6.787
Function evaluations=      82

Run with Costfactor =      1.500   cub=    17.730   qlb=     0.846
Initial Values
f=    13.941   E=    13.941   cost=    11.820      q=     0.759
Final Values
S2 Clearance time, E=     9.308       cost=    17.730      q=     0.846
x=     0.000     0.000     0.000     0.000     0.000   592.043     7.000     6.349
Function evaluations=      73

Run with Costfactor =      1.550   cub=    18.321   qlb=     0.846
Initial Values
f=    13.941   E=    13.941   cost=    11.820      q=     0.759
Final Values
```

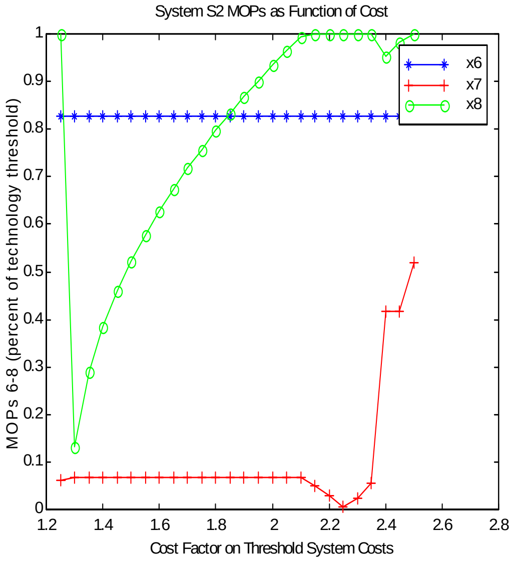
```
S2 Clearance time, E=    8.761      cost=    18.321      q=     0.846
x=    0.000     0.000     0.000     0.000     0.000   592.043     7.000     5.960
Function evaluations=       91

Run with Costfactor =      1.600   cub=    18.912   qlb=     0.846
Initial Values
f=    13.941   E=    13.941   cost=    11.820      q=     0.759
Final Values
S2 Clearance time, E=    8.264      cost=    18.912      q=     0.846
x=    0.000     0.000     0.000     0.000     0.000   592.043     7.000     5.608
Function evaluations=       82

Run with Costfactor =      1.650   cub=    19.503   qlb=     0.846
Initial Values
f=    13.941   E=    13.941   cost=    11.820      q=     0.759
Final Values
S2 Clearance time, E=    7.806      cost=    19.503      q=     0.846
x=    0.000     0.000     0.000     0.000     0.000   592.043     7.000     5.283
Function evaluations=       82

Run with Costfactor =      1.700   cub=    20.094   qlb=     0.846
Initial Values
f=    13.941   E=    13.941   cost=    11.820      q=     0.759
Final Values
S2 Clearance time, E=    7.379      cost=    20.094      q=     0.846
x=    0.000     0.000     0.000     0.000     0.000   592.043     7.000     4.980
Function evaluations=       91

Run with Costfactor =      1.750   cub=    20.685   qlb=     0.846
Initial Values
f=    13.941   E=    13.941   cost=    11.820      q=     0.759
Final Values
S2 Clearance time, E=    6.978      cost=    20.685      q=     0.846
x=    0.000     0.000     0.000     0.000     0.000   592.043     7.000     4.695
Function evaluations=       91

Run with Costfactor =      1.800   cub=    21.276   qlb=     0.846
Initial Values
f=    13.941   E=    13.941   cost=    11.820      q=     0.759
Final Values
S2 Clearance time, E=    6.597      cost=    21.276      q=     0.846
x=    0.000     0.000     0.000     0.000     0.000   592.043     7.000     4.426
Function evaluations=      100

Run with Costfactor =      1.850   cub=    21.867   qlb=     0.846
Initial Values
f=    13.941   E=    13.941   cost=    11.820      q=     0.759
Final Values
S2 Clearance time, E=    6.236      cost=    21.867      q=     0.846
x=    0.000     0.000     0.000     0.000     0.000   592.043     7.000     4.169
Function evaluations=      100

Run with Costfactor =      1.900   cub=    22.458   qlb=     0.846
Initial Values
f=    13.941   E=    13.941   cost=    11.820      q=     0.759
Final Values
S2 Clearance time, E=    5.889      cost=    22.458      q=     0.846
x=    0.000     0.000     0.000     0.000     0.000   592.043     7.000     3.924
Function evaluations=      100

Run with Costfactor =      1.950   cub=    23.049   qlb=     0.846
Initial Values
f=    13.941   E=    13.941   cost=    11.820      q=     0.759
Final Values
S2 Clearance time, E=    5.557      cost=    23.049      q=     0.846
x=    0.000     0.000     0.000     0.000     0.000   592.043     7.000     3.688
Function evaluations=      100
```

```
Run with Costfactor =     2.000   cub=    23.641   qlb=     0.846
Initial Values
f=    13.941   E=    13.941   cost=    11.820      q=     0.759
Final Values
S2 Clearance time, E=     5.238       cost=    23.641      q=     0.846
x=     0.000     0.000     0.000     0.000     0.000   592.043     7.000     3.461
Function evaluations=      100

Run with Costfactor =     2.050   cub=    24.232   qlb=     0.846
Initial Values
f=    13.941   E=    13.941   cost=    11.820      q=     0.759
Final Values
S2 Clearance time, E=     4.929       cost=    24.232      q=     0.846
x=     0.000     0.000     0.000     0.000     0.000   592.043     7.000     3.243
Function evaluations=      109

Run with Costfactor =     2.100   cub=    24.823   qlb=     0.846
Initial Values
f=    13.941   E=    13.941   cost=    11.820      q=     0.759
Final Values
S2 Clearance time, E=     4.630       cost=    24.823      q=     0.846
x=     0.000     0.000     0.000     0.000     0.000   592.043     7.000     3.031
Function evaluations=      109

Run with Costfactor =     2.150   cub=    25.414   qlb=     0.846
Initial Values
f=    13.941   E=    13.941   cost=    11.820      q=     0.759
Final Values
S2 Clearance time, E=     4.582       cost=    25.414      q=     0.846
x=     0.000     0.000     0.000     0.000     0.000   592.043     6.901     3.000
Function evaluations=      127

Run with Costfactor =     2.200   cub=    26.005   qlb=     0.846
Initial Values
f=    13.941   E=    13.941   cost=    11.820      q=     0.759
Final Values
S2 Clearance time, E=     4.575       cost=    26.005      q=     0.846
x=     0.000     0.000     0.000     0.000     0.000   592.043     6.774     3.000
Function evaluations=      127

Run with Costfactor =     2.250   cub=    26.596   qlb=     0.846
Initial Values
f=    13.941   E=    13.941   cost=    11.820      q=     0.759
Final Values
S2 Clearance time, E=     4.568       cost=    26.596      q=     0.846
x=     0.000     0.000     0.000     0.000     0.000   592.043     6.631     3.000
Function evaluations=      127

Run with Costfactor =     2.300   cub=    27.187   qlb=     0.846
Initial Values
f=    13.941   E=    13.941   cost=    11.820      q=     0.759
Final Values
S2 Clearance time, E=     4.560       cost=    27.187      q=     0.846
x=     0.000     0.000     0.000     0.000     0.000   592.043     6.465     3.000
Function evaluations=      127

Run with Costfactor =     2.350   cub=    27.778   qlb=     0.846
Initial Values
f=    13.941   E=    13.941   cost=    11.820      q=     0.759
Final Values
S2 Clearance time, E=     4.549       cost=    27.778      q=     0.846
x=     0.000     0.000     0.000     0.000     0.000   592.043     6.264     3.000
Function evaluations=      145

Run with Costfactor =     2.400   cub=    28.369   qlb=     0.846
Initial Values
```

```
f=    13.941   E=    13.941   cost=    11.820      q=     0.759
Final Values
S2 Clearance time, E=     4.906       cost=    28.369      q=     0.846
x=     0.000     0.000     0.000     0.000     0.000   592.043     4.102     3.331
Function evaluations=      157

Run with Costfactor =      2.450   cub=    28.960   qlb=     0.846
Initial Values
f=    13.941   E=    13.941   cost=    11.820      q=     0.759
Final Values
S2 Clearance time, E=     4.604       cost=    28.960      q=     0.846
x=     0.000     0.000     0.000     0.000     0.000   592.043     4.098     3.117
Function evaluations=      147

Run with Costfactor =      2.500   cub=    29.551   qlb=     0.846
Initial Values
f=    13.941   E=    13.941   cost=    11.820      q=     0.759
Final Values
S2 Clearance time, E=     4.407       cost=    29.551      q=     0.846
x=     0.000     0.000     0.000     0.000     0.000   592.043     3.479     3.000
Function evaluations=      118

cost factor   cub          E      cost        qfinal      fun. evals
     1.25    14.775     4.585    25.086     0.846       504
     1.30    15.366    13.155    15.366     0.846        91
     1.35    15.957    11.578    15.957     0.846        82
     1.40    16.548    10.653    16.548     0.846        82
     1.45    17.139     9.927    17.139     0.846       82
     1.50    17.730     9.308    17.730     0.846       73
     1.55    18.321     8.761    18.321     0.846       91
     1.60    18.912     8.264    18.912     0.846       82
     1.65    19.503     7.806    19.503     0.846       82
     1.70    20.094     7.379    20.094     0.846       91
     1.75    20.685     6.978    20.685     0.846       91
     1.80    21.276     6.597    21.276     0.846      100
     1.85    21.867     6.236    21.867     0.846      100
     1.90    22.458     5.889    22.458     0.846      100
     1.95    23.049     5.557    23.049     0.846      100
     2.00    23.641     5.238    23.641     0.846      100
     2.05    24.232     4.929    24.232     0.846      109
     2.10    24.823     4.630    24.823     0.846      109
     2.15    25.414     4.582    25.414     0.846      127
     2.20    26.005     4.575    26.005     0.846      127
     2.25    26.596     4.568    26.596     0.846      127
     2.30    27.187     4.560    27.187     0.846      127
     2.35    27.778     4.549    27.778     0.846      145
     2.40    28.369     4.906    28.369     0.846      157
     2.45    28.960     4.604    28.960     0.846      147
     2.50    29.551     4.407    29.551     0.846      118
```


System S2 MOE as Function of Cost
Time to Complete Mission (hours)
14
13
12
11
10
9
8
7
6
5
4
1.2
1.4
1.6
1.8
2
2.2
2.4
2.6
2.8
Cost Factor on Threshold System Costs


System S2 MOE as Function of Cost
Time to Complete Mission (hours)
14
13
12
11
10
9
8
7
6
5
4
14
16
18
20
22
24
26
28
30
Cost ($M)


System S2 MOPs as Function of Cost
MOPs 6-8 (percent of technology threshold)
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
1.2
1.4
1.6
1.8
2
2.2
2.4
2.6
2.8
Cost Factor on Threshold System Costs
x6
x7
x8

APPENDIX C

CONSTRAINED SQP OPTIMIZATION (PENALTY FUNCTION) MATLAB® CODE AND RESULTS

The same basic code it utilized for this method as that shown in Appendix A, with the exception of the objective function, which is listed below.

```
function [f,g] = mcmfunpf(x,p1,p2,p3,p4,p5,p6,p7)
%  Modified to have smaller Px2 on 7/3/97
a1=2000;  %  penalty function gains for absolute value PF
a2=2e6;
px1=[4.503408803940725e-005,   -5.386095666335044e-003,    2.159330101073730e-001,
       1.334245377520354e+000];
px2=[2.834645669291690e+002,   -5.076377952756583e+002,    2.274598425197177e+002];
px3=[-2.048380952380911e+000,    9.987333333333214e+000,   -1.794233333333325e+001,
       2.032238095238094e+001];
px4=[1.159691730856429e-001,   -2.175732453433467e+000,    1.520381256718985e+001];
px5=[2.061825086032983e-004,   -3.775958229500408e-002,    1.777803488786043e+000];
px6=[1.504875482450802e-007,   -1.578229837871938e-004,    5.516694369186904e-002,  -
       1.813253427503106e+000];
px7=[ -2.850358103957624e-001,    3.846213159671302e+000,   -1.726423877731832e+001,
       3.334408692656030e+001];
px8=[2.102445277065673e-001,   -4.109593768487483e+000,    2.539723920331297e+001];
f1 = (p5/60)*(24*60/x(1) + p2*x(2)*x(4)*p1 + (2*x(4)-p4)*((1-x(2))*p1*p2 + x(3) + p1*p3));
f2 = (p5/60)*(p1*x(2)*x(8)*p2*exp(-x(5)/(4.481*x(6)))+(1-exp(-
x(5)/(4.481*x(6))))*p1*x(2)*x(7)*p2 +        (1-x(2))*(x(3)+p1*p3)*x(7));
f=f1+f2;
% evaluate cost constraint
g1=polyval(px1,x(1))+polyval(px2,x(2))+polyval(px3,x(3))+polyval(px4,x(4))+polyval(px5,x(5
))+polyval     (px6,x(6))+polyval(px7,x(7))+polyval(px8,x(8))-p6;
% evaluate negative of quality constraint
g2 = -p1*x(2)*exp(-x(5)/(4.481*x(6)))+p7;
f=f1 + f2 + a1*g1*g1 + a2*g2*g2;
%f=f1+f2+a1*abs(g1)+a2*abs(g2);
g=[];
%p1=pd
%p2=lambda
%p3=lambdaft
%p4=ttransit
%p5=sminefield
%p6=cub
%p7=qlb
```

```
output from execution of MCM42pf.m
09/15/97 12:15 AM    MCM42PF Baseline 9-15-97.doc

x0=
   10.000    0.900    2.000    9.170   90.000   75.000    6.600   10.000
vlb=
   10.000    0.900    0.250    3.000   42.000   75.000    1.000    3.000
vub=
  100.000    0.980    2.000    9.170   90.000  700.000    7.000   10.000

Run with Costfactor =      1.250
Initial Values
Total time, f=200978.782
Costfactor =      1.250
cost=    28.066      q=    0.620
Final Values
Total time, f=    38.589
cost=    35.083      q=    0.846
x=    57.379    0.963    2.000    4.981    45.127   417.387    7.000    8.793
```

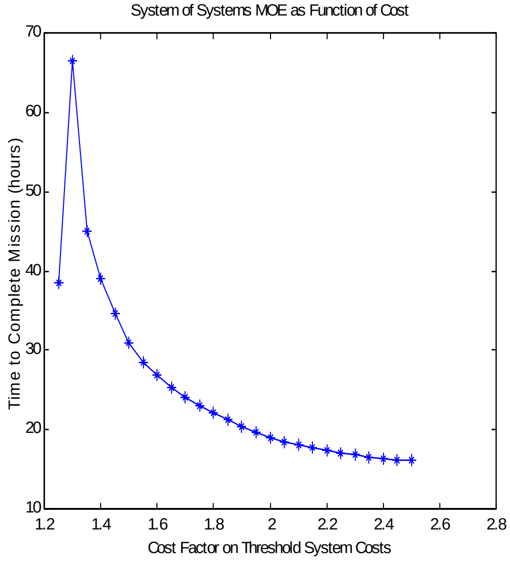
```
Function evaluations=     1329

Run with Costfactor =      1.300
Initial Values
Total time, f=244302.199
Costfactor =      1.300
cost=    28.066      q=     0.620
Final Values
Total time, f=    66.522
cost=    36.491      q=     0.846
x=    21.494     0.962     2.000     8.703    44.724   417.375     4.285     9.622
Function evaluations=      668

Run with Costfactor =      1.350
Initial Values
Total time, f=295502.600
Costfactor =      1.350
cost=    28.066      q=     0.620
Final Values
Total time, f=    44.935
cost=    37.890      q=     0.846
x=    54.484     0.963     2.000     6.584    45.440   417.385     4.259     9.150
Function evaluations=      792

Run with Costfactor =      1.400
Initial Values
Total time, f=354579.987
Costfactor =      1.400
cost=    28.066      q=     0.620
Final Values
Total time, f=    38.954
cost=    39.293      q=     0.846
x=    57.161     0.963     2.000     5.120    45.533   417.408     4.243     8.824
Function evaluations=     1658

Run with Costfactor =      1.450
Initial Values
Total time, f=421534.358
Costfactor =      1.450
cost=    28.066      q=     0.620
Final Values
Total time, f=    34.564
cost=    40.696      q=     0.846
x=    58.769     0.963     2.000     4.026    45.351   417.392     4.231     8.580
Function evaluations=     1778

Run with Costfactor =      1.500
Initial Values
Total time, f=496365.715
Costfactor =      1.500
cost=    28.066      q=     0.620
Final Values
Total time, f=    30.932
cost=    42.100      q=     0.846
x=    59.942     0.963     2.000     3.113    45.043   417.385     4.222     8.376
Function evaluations=     1846

Run with Costfactor =      1.550
Initial Values
Total time, f=579074.056
Costfactor =      1.550
cost=    28.066      q=     0.620
Final Values
Total time, f=    28.390
cost=    43.503      q=     0.846
x=    64.779     0.963     2.000     3.000    45.649   417.394     4.177     7.280
Function evaluations=     2356
```

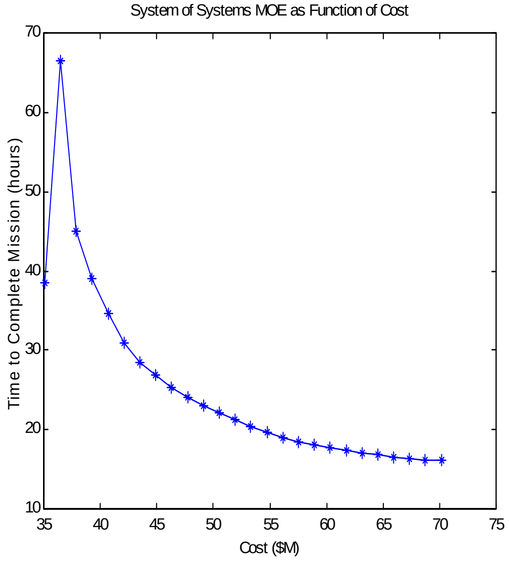
```
Run with Costfactor =     1.600
Initial Values
Total time, f=669659.382
Costfactor =     1.600
cost=   28.066     q=    0.620
Final Values
Total time, f=   26.816
cost=   44.906     q=    0.846
x=   65.900    0.963    1.533    3.000    45.352   416.227    4.173    6.928
Function evaluations=    2395

Run with Costfactor =     1.650
Initial Values
Total time, f=768121.693
Costfactor =     1.650
cost=   28.066     q=    0.620
Final Values
Total time, f=   25.329
cost=   46.309     q=    0.846
x=   68.013    0.963    1.366    3.000    45.410   417.394    4.154    6.276
Function evaluations=    2247

Run with Costfactor =     1.700
Initial Values
Total time, f=874460.989
Costfactor =     1.700
cost=   28.066     q=    0.620
Final Values
Total time, f=   24.096
cost=   47.712     q=    0.846
x=   69.876    0.963    1.260    3.000    47.367   429.961    4.147    5.688
Function evaluations=    1966

Run with Costfactor =     1.750
Initial Values
Total time, f=988677.270
Costfactor =     1.750
cost=   28.066     q=    0.620
Final Values
Total time, f=   23.020
cost=   49.116     q=    0.846
x=   71.052    0.963    1.197    3.000    45.337   417.504    4.123    5.098
Function evaluations=    2525

Run with Costfactor =     1.800
Initial Values
Total time, f=1110770.536
Costfactor =     1.800
cost=   28.066     q=    0.620
Final Values
Total time, f=   22.066
cost=   50.519     q=    0.846
x=   72.231    0.963    1.141    3.000    45.808   419.586    4.090    4.582
Function evaluations=    2129

Run with Costfactor =     1.850
Initial Values
Total time, f=1240740.786
Costfactor =     1.850
cost=   28.066     q=    0.620
Final Values
Total time, f=   21.200
cost=   51.922     q=    0.846
x=   73.220    0.963    1.094    3.000    45.681   417.212    4.100    4.099
Function evaluations=    3429
```

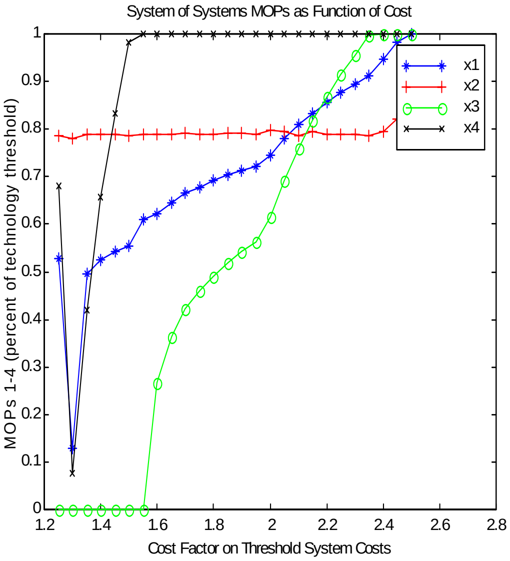
```
Run with Costfactor =     1.900
Initial Values
Total time, f=1378588.022
Costfactor =     1.900
cost=   28.066     q=    0.620
Final Values
Total time, f=   20.402
cost=   53.325     q=    0.846
x=   74.134    0.963    1.052    3.000   46.129  418.973    4.084    3.651
Function evaluations=    3633

Run with Costfactor =     1.950
Initial Values
Total time, f=1524312.242
Costfactor =     1.950
cost=   28.066     q=    0.620
Final Values
Total time, f=   19.660
cost=   54.729     q=    0.846
x=   74.887    0.963    1.015    3.000   45.445  416.088    4.080    3.225
Function evaluations=    4130

Run with Costfactor =     2.000
Initial Values
Total time, f=1677913.448
Costfactor =     2.000
cost=   28.066     q=    0.620
Final Values
Total time, f=   18.981
cost=   56.132     q=    0.846
x=   76.993    0.964    0.924    3.000   46.944  419.530    4.055    3.000
Function evaluations=    3704

Run with Costfactor =     2.050
Initial Values
Total time, f=1839391.638
Costfactor =     2.050
cost=   28.066     q=    0.620
Final Values
Total time, f=   18.448
cost=   57.535     q=    0.846
x=   80.227    0.964    0.787    3.000   45.544  408.849    4.012    3.000
Function evaluations=    5184

Run with Costfactor =     2.100
Initial Values
Total time, f=2008746.813
Costfactor =     2.100
cost=   28.066     q=    0.620
Final Values
Total time, f=   18.022
cost=   58.939     q=    0.846
x=   82.877    0.963    0.672    3.000   46.788  434.541    3.980    3.000
Function evaluations=    4160

Run with Costfactor =     2.150
Initial Values
Total time, f=2185978.973
Costfactor =     2.150
cost=   28.066     q=    0.620
Final Values
Total time, f=   17.661
cost=   60.342     q=    0.846
x=   85.025    0.964    0.567    3.000   48.223  435.763    3.962    3.000
Function evaluations=    5054

Run with Costfactor =     2.200
```

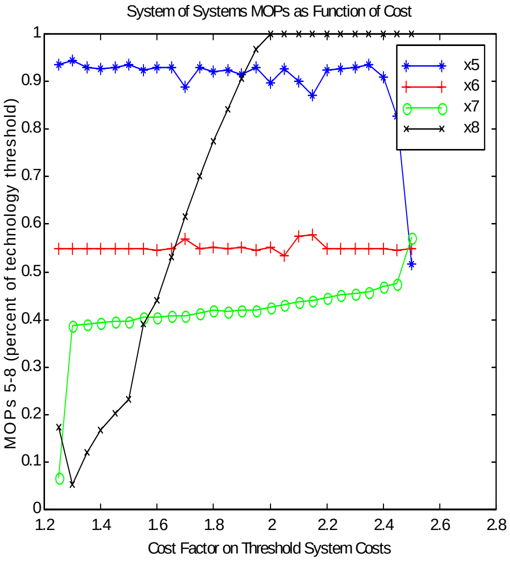
```
Initial Values
Total time, f=2371088.117
Costfactor =      2.200
cost=    28.066      q=     0.620
Final Values
Total time, f=    17.346
cost=    61.745      q=     0.846
x=    87.141     0.963     0.481     3.000     45.604    417.051     3.920     3.000
Function evaluations=     4042

Run with Costfactor =      2.250
Initial Values
Total time, f=2564074.247
Costfactor =      2.250
cost=    28.066      q=     0.620
Final Values
Total time, f=    17.070
cost=    63.148      q=     0.846
x=    88.943     0.963     0.401     3.000     45.574    417.080     3.887     3.000
Function evaluations=     6250

Run with Costfactor =      2.300
Initial Values
Total time, f=2764937.362
Costfactor =      2.300
cost=    28.066      q=     0.620
Final Values
Total time, f=    16.822
cost=    64.552      q=     0.846
x=    90.519     0.963     0.325     3.000     45.361    417.815     3.876     3.000
Function evaluations=     5658

Run with Costfactor =      2.350
Initial Values
Total time, f=2973677.461
Costfactor =      2.350
cost=    28.066      q=     0.620
Final Values
Total time, f=    16.596
cost=    65.955      q=     0.846
x=    92.032     0.963     0.257     3.000     45.133    417.483     3.856     3.000
Function evaluations=     5391

Run with Costfactor =      2.400
Initial Values
Total time, f=3190294.545
Costfactor =      2.400
cost=    28.066      q=     0.620
Final Values
Total time, f=    16.399
cost=    67.358      q=     0.846
x=    95.287     0.964     0.250     3.000     46.376    417.007     3.783     3.000
Function evaluations=     4955

Run with Costfactor =      2.450
Initial Values
Total time, f=3414788.615
Costfactor =      2.450
cost=    28.066      q=     0.620
Final Values
Total time, f=    16.237
cost=    68.762      q=     0.846
x=    98.415     0.966     0.250     3.000     50.354    415.269     3.745     3.000
Function evaluations=     3347

Run with Costfactor =      2.500
Initial Values
```

```
Total time, f=3647159.669
Costfactor =      2.500
cost=    28.066      q=     0.620
Final Values
Total time, f=    16.122
cost=    70.165      q=     0.846
x=   100.000     0.973     0.250     3.000    65.289   417.588     3.166     3.000
Function evaluations=     3268

cost factor  cost        MOE      function evals
     1.25    35.082    38.589  1329.000
     1.30    36.486    66.522   668.000
     1.35    37.889    44.935   792.000
     1.40    39.292    38.954  1658.000
     1.45    40.696    34.564  1778.000
     1.50    42.099    30.932  1846.000
     1.55    43.502    28.390  2356.000
     1.60    44.906    26.816  2395.000
     1.65    46.309    25.329  2247.000
     1.70    47.712    24.096  1966.000
     1.75    49.115    23.020  2525.000
     1.80    50.519    22.066  2129.000
     1.85    51.922    21.200  3429.000
     1.90    53.325    20.402  3633.000
     1.95    54.729    19.660  4130.000
     2.00    56.132    18.981  3704.000
     2.05    57.535    18.448  5184.000
     2.10    58.939    18.022  4160.000
     2.15    60.342    17.661  5054.000
     2.20    61.745    17.346  4042.000
     2.25    63.148    17.070  6250.000
     2.30    64.552    16.822  5658.000
     2.35    65.955    16.596  5391.000
     2.40    67.358    16.399  4955.000
     2.45    68.762    16.237  3347.000
     2.50    70.165    16.122  3268.000
```


System of Systems MOE as Function of Cost
Time to Complete Mission (hours)
10
20
30
40
50
60
70
1.2
1.4
1.6
1.8
2
2.2
2.4
2.6
2.8
Cost Factor on Threshold System Costs


System of Systems MOE as Function of Cost
Time to Complete Mission (hours)
10
20
30
40
50
60
70
35
40
45
50
55
60
65
70
75
Cost ($M)


System of Systems MOPs as Function of Cost
MOPs 1-4 (percent of technology threshold)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
1.2
1.4
1.6
1.8
2
2.2
2.4
2.6
2.8
Cost Factor on Threshold System Costs
x1
x2
x3
x4


System of Systems MOPs as Function of Cost
MOPs 5-8 (percent of technology threshold)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
1.2
1.4
1.6
1.8
2
2.2
2.4
2.6
2.8
Cost Factor on Threshold System Costs
x5
x6
x7
x8

APPENDIX D

FIRST ORDER CONSTRAINED SPSA MATLAB® CODE, EXAMPLE, AND RESULTS

The following three dimensional nonlinear programming problem was used to check out implementation of the constrained SPSA algorithm and code developed for this dissertation effort:

$$\begin{aligned}&\text{Min } z = 3x_1^{\;2} + 4x_2^{\;2},\\ &\text{subject to:}\\ &2x_1 - 3x_2 \le 10\\ &2 \le x_1 \le 4\\ &-6 \le x_2 \le -2\end{aligned}$$

Geometrically, z is an elliptic cone with its origin at (0,0). The interval constraints form a box with the linear constraint making an intersecting plane. The problem boils down to finding a point on section of the elliptic cone that lies inside the clipped box formed by the constraints. The correct solution is [2,-2], yielding z=28 as the constrained minimum. Since the objective function is simply a quadratic, the Lagrangian method was used in a straightforward manner to transform the problem:

$$\begin{aligned}&\text{Min F}(x_1, x_2, x_3) = 3x_1^{\;2} + 4x_2^{\;2} \text{ - } x_3(\text{-}2x_1 + 3x_2 + 10)\\ &\text{subject to:}\\ &2 \le x_1 \le 4\\ &-6 \le x_2 \le -2\\ &0 \le x_3, \qquad \text{where } x_3 \text{ is the Lagrangian multiplier}\end{aligned}$$

With initial condition vector [3,-4,0], 1SPSA took an average of 9.8 iterations to achieve the exact solution, with an absolute blocking criteria applied. Without blocking, the average number of iterations was 13.4. Other initial conditions were investigated, with similar results. The following code was used to solve this initial problem, and served as the basis for algorithm/code development for the MCM system of systems problem.

```
%  Filename:  \lumanrr1\matlab\prob43xx.m
%  Output file initialization
fid=fopen('/home/lumanrr1/matlab/output.txt','w');
fprintf(fid,'output from execution of prob43.m\n\n');
%======================================================================%
%              SPSA Initialization
%
%======================================================================
x0=[3,-4,0]
fprintf(fid,'x0=');
fprintf(fid,'%8.3f  %8.3f   %8.3f\n',x0);
vlb=[2,-6,0]           % lower bound constraint on x
vub=[4,-2,1000]               % upper bound constraint on x
x=x0
n=100                  % maximum allowable number of iterations
blocks=0               % number of blocked iterations
tolf=0.001             % convergence tolerances
tolx=0.00001
p=3                    % dimension of x
[f,g]=funa4(x)         % evaluate f right away
%
A=20                   % Spall cookbook, about 5% of max iterations
alpha=0.602            % Spall cookbook values
gamma=0.101
c=0.5                  % Value for c used by Ilenda was 0.05.  Critical parameter.
%                        May need to scale whole problem's x so that the optimal
%                        values lie in a similar range.
a=0.078                % Select a consistent with expectation of about 0.1 movement
%                        in x during initial iterations.  Hence,
%                        [a/(A+1)^alpha]*20.0 = 0.1, solve for a.
fprintf(fid,'vlb=\n');fprintf(fid,'%8.3f',vlb);fprintf(fid,'\n');
fprintf(fid,'vub=\n');fprintf(fid,'%8.3f',vub);fprintf(fid,'\n');
fprintf(fid,'tolf=');fprintf(fid,'%10.6f\n',tolf);
fprintf(fid,'tolx=');fprintf(fid,'%10.6f\n',tolx);
fprintf(fid,'A=');fprintf(fid,'%8.3f',A);fprintf(fid,'
alpha=');fprintf(fid,'%8.3f\n',alpha);
fprintf(fid,'c=');fprintf(fid,'%8.3f',c);fprintf(fid,'   a=');fprintf(fid,'%8.3f\n',a);
%======================================================================
%
%              SPSA Loop
%
%======================================================================
for k=1:n
        k
        fprintf(fid,'\n');fprintf(fid,'====================================================
=\n');
        fprintf(fid,'iteration #');fprintf(fid,'%3.0f\n',k);
        a_k=a/(k+A)^alpha      % strongly dependent on a, A initialization
        c_k=c/k^gamma          % strongly dependednt on c:  small relative to x
        fprintf(fid,'a_k=');fprintf(fid,'%8.3f',a_k);fprintf(fid,'
c_k=');fprintf(fid,'%8.3f\n',c_k);
        for j=1:p
                delta(j)=2*round(rand(1))-1;
                delta(j);
        end
%       delta
        xplus=x+c_k*delta      % If the elements of x vary significantly in magnitude,
        xminus=x-c_k*delta     % then should either scale delta or x elements.
        fprintf(fid,'delta=');fprintf(fid,'%8.3f',delta);fprintf(fid,'\n');
        fprintf(fid,'xminus(raw)=');fprintf(fid,'%8.3f',xminus);fprintf(fid,'\n');
        fprintf(fid,'xplus(raw)=');fprintf(fid,'%8.3f',xplus);fprintf(fid,'\n');
                               %  Key is probably the jitter, not magnitude.
%       check for infeasible vectors
        for j=1:p
                if xplus(j)<vlb(j)
                xplus(j)=vlb(j);
                end
                if xminus(j)<vlb(j)
                xminus(j)=vlb(j);
                end
                if xplus(j)>vub(j)
                xplus(j)=vub(j);
                end
                if xminus(j)>vub(j)
                xminus(j)=vub(j);
                end
        end
        xminus,xplus
        fprintf(fid,'xminus(constr)=');fprintf(fid,'%8.3f',xminus);fprintf(fid,'\n');
        fprintf(fid,'xplus(constr)=');fprintf(fid,'%8.3f',xplus);fprintf(fid,'\n');
%
%       Update, checking for convergence
        [fplus,g]=funa4(xplus);
        [fminus,g]=funa4(xminus);
        fminus
        fplus
        ghat=(fplus-fminus)./(2*c_k*delta)
        fprintf(fid,'fminus=');fprintf(fid,'%8.3f',fminus);
        fprintf(fid,'    fplus=');fprintf(fid,'%8.3f\n',fplus);
        fprintf(fid,'ghat=');fprintf(fid,'%8.3f',ghat);fprintf(fid,'\n');
        xkp1=x-a_k*ghat
        fprintf(fid,'xkp1(raw)=');fprintf(fid,'%8.3f',xkp1);fprintf(fid,'\n');
%
%       check for infeasible updated estimate
        for j=1:p
                if xkp1(j)<vlb(j)
                xkp1(j)=vlb(j);
                end
                if xkp1(j)>vub(j)
                xkp1(j)=vub(j);
                end
        end
        xkp1
        fprintf(fid,'xkp1(constr)=');fprintf(fid,'%8.3f',xkp1);fprintf(fid,'\n');
        [fkp1,g]=funa4(xkp1);
        fkp1
        fprintf(fid,'f_k=');fprintf(fid,'%8.3f',f);fprintf(fid,'f_kp1=');
        fprintf(fid,'%8.3f\n',fkp1);
%
        fundiff=abs(fkp1-f);
        if fundiff < tolf
                fundiff
                xkp1
                fkp1
                k
                fprintf(fid,'Terminate:  fundiff=');fprintf(fid,'%12.8f\n',fundiff);
                return
        end
        xdifnorm=norm(x-xkp1);
        if xdifnorm < tolx
                xdifnorm
                xkp1
                fkp1
                k
                fprintf(fid,'Terminate:  xdifnorm=');fprintf(fid,'%12.8f\n',xdifnorm);
                return
        end
        if fkp1<f
                x=xkp1  % Accept new estimate.  Otherwise block the update passively.
                f=fkp1
        else
                blocks=blocks+1
                fprintf(fid,'Update Blocked.  Block number:');fprintf(fid,'%3.0\n',blocks)
        end
end

function [f,g] = funa4(x)
f = 3*(x(1))^2 + 4*(x(2))^2 + x(3)*(2*x(1)-3*x(2)-10); % LaGrangian
g=[] ;   % only constraints are upper and lower bounds
```

1SPSA Code for MCM System of Systems:

```
%  MCM System of Systems
      %  SPSA utilizing penalty function MCMFUNPF that was used with MCM42PF.m
      %  Except that the penalty functions have a gain that increases with k, iteration
number.
      %  Filename:  \matlab\dissertation\MCMSPSA.m
      %This file utilizes the standard scalar gain, a_k, which will not compensate for
scaling
      %differences in variables.
      %  Output file initialization
      clear
fid=fopen('c:\matlab\dissertation\output SPSA.doc','w');
fprintf(fid,'output from execution of MCMspsa.m, scalar gain a_k\n\n');
      %================================================================================
      %
      %        MCM Initialization
      %
      %================================================================================
      %
      % x(1)=p11=  S1 area coverage rate (nm^2/day)
      % x(2)=p12=  S1 classification probability
      % x(3)=p13=  S1 FAR (#/nm^2)
      % x(4)=p14=  S1 time to classify (min)
      % x(5)=p15=  S1 navigation accuracy (yards)
      % x(6)=p21=  S2 Re-acquisition range (yards)
      % x(7)=p22=  S2 time to prosecute a false target (min)
      % x(8)=p23=  S2 time to neutralize (min)
      pd=0.90;     %S1 detection system probability of detection
x0=[10;0.9;2.0;9.17;90;75;6.6;10.0]; %Might need to start at feasible point.  This won't
meet q(x) constraint.
      p=8;                    % dimension of x
xstar=x0   %this is the threshold system.  But too far from opimal for SPSA to start with.
x0=[70;.95;1.0;5.0;45;430;3.5;6]; %pretty near optimal for costfactor around 1.6.
xoptimal=[10.000     0.900     2.000     9.170    90.000    75.000     6.600    10.000;...
      57.379     0.963     2.000     4.983    45.108   417.391     7.000     8.793;...
      58.915     0.963     2.000     3.915    44.820   417.374     7.000     8.555;...
      60.058     0.963     2.000     3.017    44.418   417.371     7.000     8.355;...
      65.201     0.963     2.000     3.000    44.632   417.375     7.000     7.162;...
      67.996     0.963     2.000     3.000    44.802   417.387     7.000     6.282;...
      68.194     0.962     1.342     3.000    43.341   417.304     7.000     6.213;...
      69.825     0.962     1.249     3.000    42.959   417.274     7.000     5.603;...
      71.172     0.962     1.181     3.000    42.596   417.237     7.000     5.045;...
      72.316     0.961     1.126     3.000    42.245   417.195     7.000     4.532;...
      73.310     0.961     1.080     3.000    42.000   417.254     7.000     4.055;...
      74.189     0.961     1.040     3.000    42.000   417.519     7.000     3.607;...
      74.979     0.961     1.004     3.000    42.000   417.757     7.000     3.184;...
      77.328     0.961     0.901     3.000    42.000   418.623     7.000     3.000;...
      80.440     0.961     0.765     3.000    42.000   420.090     7.000     3.000;...
      83.033     0.961     0.651     3.000    42.000   421.627     7.000     3.000;...
      85.261     0.961     0.552     3.000    42.000   423.201     7.000     3.000;...
      87.217     0.961     0.463     3.000    42.000   424.790     7.000     3.000;...
      88.965     0.961     0.383     3.000    42.000   426.399     7.000     3.000;...
      85.153     0.963     0.570     3.000    45.839   417.389     3.948     3.000;...
      87.149     0.963     0.481     3.000    45.694   417.395     3.921     3.000;...
      88.941     0.963     0.400     3.000    45.508   417.412     3.894     3.000;...
      90.539     0.963     0.326     3.000    45.239   417.380     3.876     3.000;...
      92.010     0.963     0.257     3.000    44.947   417.360     3.857     3.000;...
      95.302     0.963     0.250     3.000    45.572   417.461     3.797     3.000;...
      98.430     0.964     0.250     3.000    46.343   417.398     3.737     3.000;...
      100.000     0.973     0.250     3.000    64.008   412.202     3.162     3.000];
fprintf(fid,'x0=\n');fprintf(fid,'%10.3f',x0);fprintf(fid,'\n');
vlb=[10.0;0.9;0.25;3.0;42;75;1.0;3.0];              % lower bound constraint on x
vub=[100;0.98;2.0;9.17;90;700;7.0;10.0];            % upper bound constraint on x
fprintf(fid,'vlb=\n');fprintf(fid,'%10.3f',vlb);fprintf(fid,'\n');
fprintf(fid,'vub=\n');fprintf(fid,'%10.3f',vub);fprintf(fid,'\n\n');
%  Compute px1, polynomial fit cost function parameters for x(1)
x1=[10,57,82,94];
y1=[3,4.483,7.655,11.445];
px1=polyfit(x1,y1,3);
%  Compute px2, polynomial fit cost function parameters for x(2)
x1=[0.9,0.93,0.96,0.98];
y1=[.2,.5,1.4,2.2];  %revised PBCM that reflects COTS/NDI development
px2=polyfit(x1,y1,2);
%  Compute px3, polynomial fit cost function parameters for x(3)
x1=[2,1,0.5,0.25];
y1=[8,10.319,13.592,16.429];
px3=polyfit(x1,y1,3);
%  Compute px4, polynomial fit cost function parameters for x(4)
x1=[3.513,3.89,4.77,9.17];
y1=[9.191,8.190,7.574,5.0];
px4=polyfit(x1,y1,2);
```

```
%  Compute px5, polynomial fit cost function parameters for x(5)
x1=[90,60,48,42];
y1=[0.050,0.250,0.450,0.550];
px5=polyfit(x1,y1,2);
%  Compute px6, polynomial fit cost function parameters for x(6)
x1=[129,457,622,75];
y1=[3,4.8,7.655,1.5];
px6=polyfit(x1,y1,3);
%  Compute px7, polynomial fit cost function parameters for x(7)
x1=[6.6,4.4,3.3,2.64,1.32];
y1=[5.0,7.5,8.190,9.191,16.621];
px7=polyfit(x1,y1,3);
%  Compute px8, polynomial fit cost function parameters for x(8)
x1=[10,8,7,5,3];
y1=[5.3,6,7,10,15];
px8=polyfit(x1,y1,2);
%compute threshold system cost
cost0=polyval(px1,xstar(1))+polyval(px2,xstar(2))+polyval(px3,xstar(3))+polyval(px4,xstar(
4))+polyval(px5,xstar(5))+polyval(px6,xstar(6))+polyval(px7,xstar(7))+polyval(px8,xstar(8)
)
%Insert costfactor loop
i=0;
for costfactor=1.25:0.05:2.5 %Note problem is infeasible with costfactor<1.25
   i=i+1;
   for j=1:p  %start "near the optimal solution"--perturb randomly by 20%
      del=2*round(rand)-1; %this is +/-1 random number
      x0(j)=(1+0.20*del)*xoptimal(i,j); % results in either 120% or 80% of optimal.
      if x0(j)<vlb(j);x0(j)=vlb(j);end  %make sure x0 is feasible
      if x0(j)>vub(j);x0(j)=vub(j);end
      end
costfactor
   fprintf(fid,'\n');
   cub=costfactor*cost0;         %system of systems cost constraint
   qlb=0.846;             %quality constraint lower bound
   fprintf(fid,'Run with Costfactor = ');fprintf(fid,'%10.3f',costfactor);fprintf(fid,'
cub=');fprintf(fid,'%10.3f',cub);fprintf(fid,'   qlb=');fprintf(fid,'%10.3f\n',qlb);
sminefield=20;        %minefield area, (nm^2)
m0=100;               %number of mines in minefield, initially
lambda=m0/sminefield; %mine density, (#/nm^2)
dmine=600;            %average distance between mines, (yards)
vtransit=7;           %S1 vehicle transit speed (knots)
ttransit=dmine/(2000*vtransit);    %transit time during classification (hours)
lambdaft=1.0;         %false target density (#/nm^2)
% Define parameters for the objective function
p1=pd;
p2=lambda;
p3=lambdaft;
p4=ttransit;
p5=sminefield;
p6=cub;
p7=qlb;
p8=1;  %Placeholder---p8 will be iteration number of SPSA algorithm.
[f,g]=mcmfunpfs(x0,p1,p2,p3,p4,p5,p6,p7,p8);
fprintf(fid,'Initial Values\n');
x=x0
cost=polyval(px1,x(1))+polyval(px2,x(2))+polyval(px3,x(3))+polyval(px4,x(4))+polyval(px5,x
       (5))+polyval(px6,x(6))+polyval(px7,x(7))+polyval(px8,x(8))
q=p1*x(2)*exp(-x(5)/(4.481*x(6)));
f1 = (p5/60)*(24*60/x(1) + p2*x(2)*x(4)*p1 + (2*x(4)-p4)*((1-x(2))*p1*p2 + x(3) + p1*p3));
f2 = (p5/60)*(p1*x(2)*x(8)*p2*exp(-x(5)/(4.481*x(6)))+(1-exp(-
       x(5)/(4.481*x(6))))*p1*x(2)*x(7)*p2 + (1-x(2))*(x(3)+p1*p3)*x(7));
E=f1+f2;
q=p1*x0(2)*exp(-x0(5)/(4.481*x0(6)));
fprintf(fid,'f=');fprintf(fid,'%10.3f',f);fprintf(fid,'   E=');fprintf(fid,'%10.3f',E);
fprintf(fid,'   cost=');fprintf(fid,'%10.3f',cost);fprintf(fid,'
q=');fprintf(fid,'%10.3f',q);
%  Print out threshold system values for f and constraints, g.
%==============================================================================
%
%              SPSA Initialization
%
%==============================================================================
n=2500;               % maximum allowable number of iterations
blocks=0;             % number of blocked iterations
tolf=0.000015;        % convergence tolerances
```

```
tolx=0.00001;
p=8;                   % dimension of x
%[f,g]=mcmfunpfs(x0,p1,p2,p3,p4,p5,p6,p7,p8)        % evaluate f right away
%
A=50;                  % Spall cookbook, about 5% of max iterations
alpha=0.602;           % Spall cookbook values
gamma=0.101;
c=0.002;                      % Value for c used by Ilenda was 0.05.  Critical parameter.
%                        May need to scale whole problem's x so that the optimal
%                        values lie in a similar range.
a=0.02;
%a=[.02,1e-4,.006,.02,.2,.4,.02,.02]                 % Select a consistent with
expectation of about 0.1 movement
%                        in x during initial iterations.  Hence,
%                        [a/(A+1)^alpha]*20.0 = 0.1, solve for a.
fprintf(fid,'   tolx=');fprintf(fid,'%10.6f',tolx);
fprintf(fid,'   tolf=');fprintf(fid,'%10.6f\n',tolf);
fprintf(fid,'A=');fprintf(fid,'%8.3f',A);fprintf(fid,'
alpha=');fprintf(fid,'%8.3f',alpha);
fprintf(fid,'    c=');fprintf(fid,'%8.3f',c);fprintf(fid,'
a=');fprintf(fid,'%8.3f\n',a);
%=================================================================================
%
%              SPSA Loop
%
%=================================================================================
for k=1:n;
   p8=k;
%fprintf(fid,'\n');fprintf(fid,'=================================================\n');
%fprintf(fid,'iteration #');fprintf(fid,'%3.0f\n',k);
a_k=a/(k+A)^alpha;     % strongly dependent on a, A initialization
c_k=c/k^gamma;         % strongly dependednt on c:  small relative to x
%fprintf(fid,'a_k=');fprintf(fid,'%8.3f',a_k);fprintf(fid,'
c_k=');fprintf(fid,'%8.3f\n',c_k);
delta=2*round(rand(p,1))-1;
   %   delta
       xplus=x+c_k*delta;     % If the elements of x vary significantly in magnitude,
       xminus=x-c_k*delta;    % then should either scale delta or x elements.
       %fprintf(fid,'delta=');fprintf(fid,'%8.3f',delta);fprintf(fid,'\n');
       %fprintf(fid,'xminus(raw)=');fprintf(fid,'%8.3f',xminus);fprintf(fid,'\n');
       %fprintf(fid,'xplus(raw)=');fprintf(fid,'%8.3f',xplus);fprintf(fid,'\n');
                              %  Key is probably the jitter, not magnitude.
%      check for infeasible vectors
for j=1:p;  %if on a border, pull back and adjust c_k component j.
      if xplus(j)<vlb(j);
         xplus(j)=vlb(j);
      end;
      if xminus(j)<vlb(j);
         xminus(j)=vlb(j);
      end;
      if xplus(j)>vub(j);
         xplus(j)=vub(j);
      end;
               if xminus(j)>vub(j);
         xminus(j)=vub(j);
               end;
end;
       %xminus,xplus
       %fprintf(fid,'xminus(constr)=');%fprintf(fid,'%8.3f',xminus);%fprintf(fid,'\n');
       %fprintf(fid,'xplus(constr)=');%fprintf(fid,'%8.3f',xplus);%fprintf(fid,'\n');
%
%      Update, checking for convergence
       [fplus,g]=mcmfunpfs(xplus,p1,p2,p3,p4,p5,p6,p7,p8);
       [fminus,g]=mcmfunpfs(xminus,p1,p2,p3,p4,p5,p6,p7,p8);
fminus;
fplus;
ghat=(fplus-fminus)./(2*c_k*delta);
      % xkp1(j)=x(j)-a_k(j)*ghat(j);   %vector gain to compensate for scaling
xkp1=x-a_k*ghat;        %scalar gain as standard SPSA
%fprintf(fid,'fminus=');fprintf(fid,'%8.3f',fminus);fprintf(fid,'
fplus=');fprintf(fid,'%8.3f\n',fplus);
%fprintf(fid,'ghat=');fprintf(fid,'%8.3f',ghat);fprintf(fid,'\n');
%fprintf(fid,'xkp1(raw)=');fprintf(fid,'%8.3f',xkp1);fprintf(fid,'\n');
%
%      check for infeasible updated estimate
```

```
        for j=1:p;
                if xkp1(j)<vlb(j);
                xkp1(j)=vlb(j);
                end;
                if xkp1(j)>vub(j);
                xkp1(j)=vub(j);
                end;
        end;
xkp1;
%fprintf(fid,'xkp1(constr)=');fprintf(fid,'%8.3f',xkp1);fprintf(fid,'\n');
[fkp1,g]=mcmfunpfs(xkp1,p1,p2,p3,p4,p5,p6,p7,p8);
%fprintf(fid,'f_k=');fprintf(fid,'%8.3f',f);fprintf(fid,'
f_kp1=');fprintf(fid,'%8.3f\n',fkp1);
   %
if fkp1<f+1;
        x=xkp1; % Accept new estimate.  Otherwise block the update passively.
        f=fkp1;
   else;
        blocks=blocks+1;
        %%fprintf(fid,'Update Blocked.  Block number:');%fprintf(fid,'%3.0\n',blocks)
   end;
end
mcmeval(x,p1,p2,p3,p4,p5,p6,p7)
[f,g]=mcmfunpfs(x,p1,p2,p3,p4,p5,p6,p7,p8)
fprintf(fid,'Final Values\n');
cost =
polyval(px1,x(1))+polyval(px2,x(2))+polyval(px3,x(3))+polyval(px4,x(4))+polyval(px5,x(5))+
polyval(px6,x(6))+polyval(px7,x(7))+polyval(px8,x(8))
q=p1*x(2)*exp(-x(5)/(4.481*x(6)))
f1 = (p5/60)*(24*60/x(1) + p2*x(2)*x(4)*p1 + (2*x(4)-p4)*((1-x(2))*p1*p2 + x(3) + p1*p3));
f2 = (p5/60)*(p1*x(2)*x(8)*p2*exp(-x(5)/(4.481*x(6)))+(1-exp(-
x(5)/(4.481*x(6))))*p1*x(2)*x(7)*p2 + (1-x(2))*(x(3)+p1*p3)*x(7));
E=f1+f2
fprintf(fid,'Total time, E=');fprintf(fid,'%10.3f',E);
fprintf(fid,'       cost=');fprintf(fid,'%10.3f',cost);fprintf(fid,'
q=');fprintf(fid,'%10.3f',q);
fprintf(fid,'   Function evaluations= ');fprintf(fid,'%8.0f',2*k);
fprintf(fid,'   Blocks=');fprintf(fid,'%8.0f\n',blocks);
fprintf(fid,'x=');fprintf(fid,'%10.3f',x);fprintf(fid,'\n');
cf(i)=costfactor; fval(i)=E; costval(i)=cub;fevals(i)=2*k; costi(i)=cost;
qfinal(i)=q;nblock(i)=blocks;
z1(i)=x(1);z2(i)=x(2);z3(i)=x(3);z4(i)=x(4);z5(i)=x(5);z6(i)=x(6);z7(i)=x(7);z8(i)=x(8);
z1(i)=abs((z1(i)-xstar(1))/(vlb(1)-vub(1)));
z2(i)=abs((z2(i)-xstar(2))/(vlb(2)-vub(2)));
z3(i)=abs((z3(i)-xstar(3))/(vlb(3)-vub(3)));
z4(i)=abs((z4(i)-xstar(4))/(vlb(4)-vub(4)));
z5(i)=abs((z5(i)-xstar(5))/(vlb(5)-vub(5)));
z6(i)=abs((z6(i)-xstar(6))/(vlb(6)-vub(6)));
z7(i)=abs((z7(i)-xstar(7))/(vlb(7)-vub(7)));
z8(i)=abs((z8(i)-xstar(8))/(vlb(8)-vub(8)));
end
%  Plot option 1:  Plot System of systems MOE as CAIV
figure
plot(cf,fval,'-*b')
title('System of Systems MOE as Function of Cost')
xlabel('Cost Factor on Threshold System Costs')
ylabel('Time to Complete Mission (hours)')
figure
plot(costval,fval,'-*b')
title('System of Systems MOE as Function of Cost')
xlabel('Cost ($M)')
ylabel('Time to Complete Mission (hours)')
%
%
%  Plot option 2:  Plot MOPs as CAIV
figure
plot(cf,z1,'-b*',cf,z2,'-r+',cf,z3,'-go',cf,z4,'-kx')
legend('x1','x2','x3','x4')
title('System of Systems MOPs as Function of Cost')
xlabel('Cost Factor on Threshold System Costs')
ylabel('MOPs 1-4 (percent of technology threshold)')
figure
plot(cf,z5,'-b*',cf,z6,'-r+',cf,z7,'-go',cf,z8,'-kx')
legend('x5','x6','x7','x8')
title('System of Systems MOPs as Function of Cost')
```

```
xlabel('Cost Factor on Threshold System Costs')
ylabel('MOPs 5-8 (percent of technology threshold)')
%  print table of results to file
fprintf(fid,'\n');
fprintf(fid,'cost factor');fprintf(fid,'   cub');fprintf(fid,'          E');fprintf(fid,'
cost');fprintf(fid,'        qfinal');fprintf(fid,'      fun. evals');fprintf(fid,'
#blocks\n');
for j=1:i
fprintf(fid,'%10.2f',cf(j));fprintf(fid,'%10.3f',costval(j));
fprintf(fid,'%10.3f',fval(j));fprintf(fid,'%10.3f',costi(j));
fprintf(fid,'%10.3f',qfinal(j));fprintf(fid,'%10.0f',fevals(j));
fprintf(fid,'%10.0f\n',nblock(j));
end
status=fclose(fid)
```

```
function [f,g] = mcmfunpfs(x,p1,p2,p3,p4,p5,p6,p7,p8)
%  squared penalty function version
kmax=600;
k=p8;              %k is the iteration number of the SPSA algorithm
A1=20;
A2=2000;
if k<kmax
   a1=0.5*(A1)*sin((k-1)*pi/kmax-pi/2)+A1/2+1;
   a2=0.5*(A2)*sin((k-1)*pi/kmax-pi/2)+A2/2+1;
else
   a1=A1;
   a2=A2;
end
px1=[4.503408803940725e-005,   -5.386095666335044e-003,    2.159330101073730e-001,
       1.334245377520354e+000];
px2=[2.834645669291690e+002,   -5.076377952756583e+002,    2.274598425197177e+002];
px3=[-2.048380952380911e+000,    9.987333333333214e+000,   -1.794233333333325e+001,
       2.032238095238094e+001];
px4=[1.159691730856429e-001,   -2.175732453433467e+000,    1.520381256718985e+001];
px5=[2.061825086032983e-004,   -3.775958229500408e-002,    1.777803488786043e+000];
px6=[1.504875482450802e-007,   -1.578229837871938e-004,    5.516694369186904e-002,  -
       1.813253427503106e+000];
px7=[ -2.850358103957624e-001,    3.846213159671302e+000,   -1.726423877731832e+001,
       3.334408692656030e+001];
px8=[2.102445277065673e-001,   -4.109593768487483e+000,    2.539723920331297e+001];
f1 = (p5/60)*(24*60/x(1) + p2*x(2)*x(4)*p1 + (2*x(4)-p4)*((1-x(2))*p1*p2 + x(3) + p1*p3));
f2 = (p5/60)*(p1*x(2)*x(8)*p2*exp(-x(5)/(4.481*x(6)))+(1-exp(-
x(5)/(4.481*x(6))))*p1*x(2)*x(7)*p2 + (1-x(2))*(x(3)+p1*p3)*x(7));
E=f1+f2;
% evaluate cost constraint
g1=polyval(px1,x(1))+polyval(px2,x(2))+polyval(px3,x(3))+polyval(px4,x(4))+polyval(px5,x(5
))+polyval     (px6,x(6))+polyval(px7,x(7))+polyval(px8,x(8))-p6;
% evaluate negative of quality constraint
g2 = -p1*x(2)*exp(-x(5)/(4.481*x(6)))+p7;
%f=f1 + f2 + a1*g1*g1 + a2*g2*g2;
%E,g1,g2,f
f=f1+f2+a1*abs(g1)+a2*abs(g2);
g=[];
%p1=pd
%p2=lambda
%p3=lambdaft
%p4=ttransit
%p5=sminefield
%p6=cub
%p7=qlb
%p8=k
```

09/11/97 6:48 PM   1SPSA Baseline 9-11-97.doc

Block criteria relaxed to 1.0. Came out pretty good.

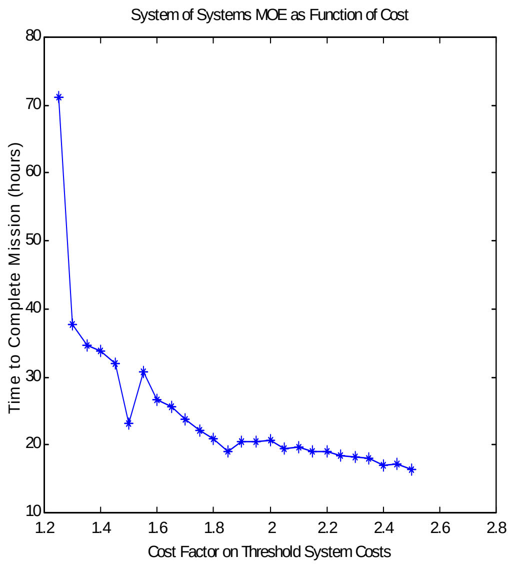
output from execution of mcmspsa2500Final.m, scalar gain a_k

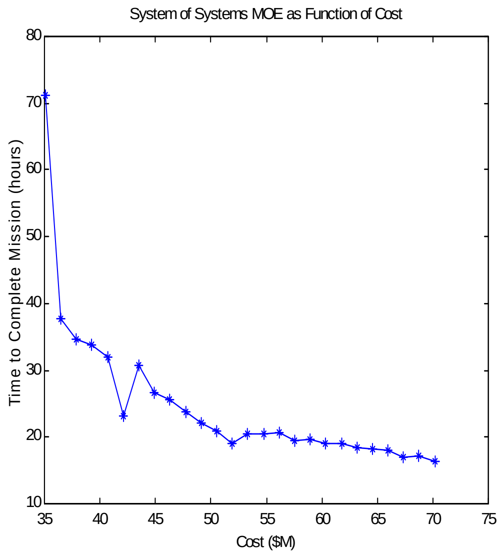
```
x0=
   70.000    0.950    1.000    5.000   45.000  430.000    3.500    6.000
vlb=
   10.000    0.900    0.250    3.000   42.000   75.000    1.000    3.000
vub=
  100.000    0.980    2.000    9.170   90.000  700.000    7.000   10.000


Run with Costfactor =     1.250   cub=    35.082   qlb=     0.846
Initial Values
f=    88.894   E=    83.317   cost=    29.646     q=     0.706   tolx=  0.000010   tolf=  0.000015
A=  50.000   alpha=   0.602   c=   0.002   a=   0.020
Final Values
Total time, E=    71.155       cost=    34.986     q=     0.705   Function evaluations=     5000   Blocks=    2419
x=    13.048     0.980     1.090     8.328    89.695    89.340     6.366     7.356

Run with Costfactor =     1.300   cub=    36.486   qlb=     0.846
Initial Values
f=    43.267   E=    38.874   cost=    40.863     q=     0.861   tolx=  0.000010   tolf=  0.000015
A=  50.000   alpha=   0.602   c=   0.002   a=   0.020
Final Values
Total time, E=    37.667       cost=    40.349     q=     0.846   Function evaluations=     5000   Blocks=    2422
x=    45.835     0.963     1.994     3.733    54.072   500.902     5.711     9.805

Run with Costfactor =     1.350   cub=    37.889   qlb=     0.846
Initial Values
f=    35.761   E=    35.349   cost=    38.290     q=     0.858   tolx=  0.000010   tolf=  0.000015
A=  50.000   alpha=   0.602   c=   0.002   a=   0.020
Final Values
Total time, E=    34.544       cost=    37.804     q=     0.810   Function evaluations=     5000   Blocks=    2460
x=    47.373     0.926     1.227     4.886    42.049   334.142     6.902     6.680

Run with Costfactor =     1.400   cub=    39.292   qlb=     0.846
Initial Values
f=    35.161   E=    33.700   cost=    40.694     q=     0.788   tolx=  0.000010   tolf=  0.000015
A=  50.000   alpha=   0.602   c=   0.002   a=   0.020
Final Values
Total time, E=    33.915       cost=    41.980     q=     0.846   Function evaluations=     5000   Blocks=    2418
x=    48.108     0.967     1.618     3.010    42.049   333.894     5.699     9.836

Run with Costfactor =     1.450   cub=    40.696   qlb=     0.846
Initial Values
f=    39.858   E=    34.258   cost=    35.160     q=     0.782   tolx=  0.000010   tolf=  0.000015
A=  50.000   alpha=   0.602   c=   0.002   a=   0.020
Final Values
Total time, E=    32.065       cost=    40.746     q=     0.851   Function evaluations=     5000   Blocks=    2453
x=    52.340     0.980     1.615     3.208    53.654   333.883     6.471     8.603

Run with Costfactor =     1.500   cub=    42.099   qlb=     0.846
Initial Values
f=    26.358   E=    23.208   cost=    45.198     q=     0.795   tolx=  0.000010   tolf=  0.000015
A=  50.000   alpha=   0.602   c=   0.002   a=   0.020
Final Values
Total time, E=    23.240       cost=    46.095     q=     0.846   Function evaluations=     5000   Blocks=    2439
x=    81.526     0.958     1.641     3.007    42.041   500.990     7.000     5.077

Run with Costfactor =     1.550   cub=    43.502   qlb=     0.846
Initial Values
f=    31.370   E=    31.285   cost=    43.433     q=     0.862   tolx=  0.000010   tolf=  0.000015
A=  50.000   alpha=   0.602   c=   0.002   a=   0.020
Final Values
Total time, E=    30.662       cost=    43.686     q=     0.862   Function evaluations=     5000   Blocks=    2488
x=    54.368     0.980     1.667     3.524    52.024   500.858     5.734     7.072

Run with Costfactor =     1.600   cub=    44.906   qlb=     0.846
Initial Values
f=    27.323   E=    26.697   cost=    44.338     q=     0.788   tolx=  0.000010   tolf=  0.000015
A=  50.000   alpha=   0.602   c=   0.002   a=   0.020
Final Values
Total time, E=    26.708       cost=    45.738     q=     0.848   Function evaluations=     5000   Blocks=    2432
x=    56.366     0.969     1.044     3.006    42.094   334.064     5.687     6.605
```

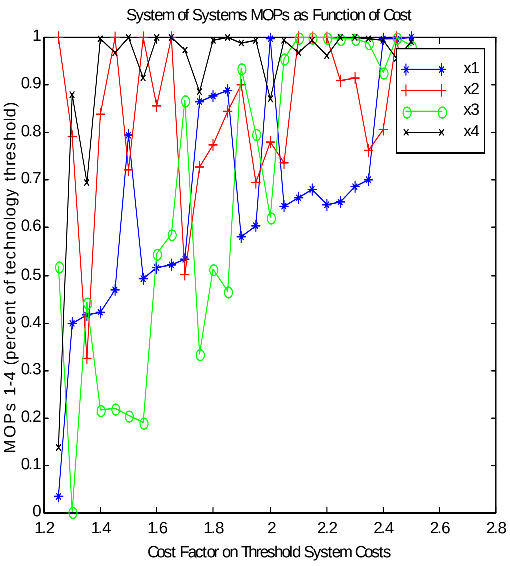
```
Run with Costfactor =     1.650   cub=    46.309   qlb=     0.846
Initial Values
f=    26.913   E=    26.584   cost=    45.986     q=     0.852   tolx=  0.000010   tolf=  0.000015
A=  50.000   alpha=   0.602    c=    0.002   a=   0.020
Final Values
Total time, E=    25.501       cost=    47.260      q=     0.852   Function evaluations=     5000   Blocks=     2438
x=    57.010     0.980     0.976     3.000    51.396   333.643     5.797     5.920

Run with Costfactor =     1.700   cub=    47.712   qlb=     0.846
Initial Values
f=    29.037   E=    24.617   cost=    43.344     q=     0.795   tolx=  0.000010   tolf=  0.000015
A=  50.000   alpha=   0.602    c=    0.002   a=   0.020
Final Values
Total time, E=    23.766       cost=    47.672      q=     0.830   Function evaluations=     5000   Blocks=     2471
x=    58.073     0.940     0.483     3.163    42.000   500.474     6.887     5.209

Run with Costfactor =     1.750   cub=    49.115   qlb=     0.846
Initial Values
f=    23.467   E=    21.615   cost=    50.917     q=     0.795   tolx=  0.000010   tolf=  0.000015
A=  50.000   alpha=   0.602    c=    0.002   a=   0.020
Final Values
Total time, E=    22.057       cost=    51.587      q=     0.846   Function evaluations=     5000   Blocks=     2434
x=    87.775     0.958     1.415     3.698    42.074   500.725     6.942     3.353

Run with Costfactor =     1.800   cub=    50.519   qlb=     0.846
Initial Values
f=    24.508   E=    19.979   cost=    54.993     q=     0.792   tolx=  0.000010   tolf=  0.000015
A=  50.000   alpha=   0.602    c=    0.002   a=   0.020
Final Values
Total time, E=    20.791       cost=    54.596      q=     0.846   Function evaluations=     5000   Blocks=     2429
x=    88.950     0.962     1.101     3.038    50.396   501.052     5.634     4.478

Run with Costfactor =     1.850   cub=    51.922   qlb=     0.846
Initial Values
f=    26.518   E=    17.885   cost=    60.544     q=     0.858   tolx=  0.000010   tolf=  0.000015
A=  50.000   alpha=   0.602    c=    0.002   a=   0.020
Final Values
Total time, E=    18.993       cost=    57.732      q=     0.847   Function evaluations=     5000   Blocks=     2429
x=    89.906     0.968     1.186     3.000    42.124   334.208     5.653     3.198

Run with Costfactor =     1.900   cub=    53.325   qlb=     0.846
Initial Values
f=    24.233   E=    21.072   cost=    50.176     q=     0.858   tolx=  0.000010   tolf=  0.000015
A=  50.000   alpha=   0.602    c=    0.002   a=   0.020
Final Values
Total time, E=    20.388       cost=    53.791      q=     0.851   Function evaluations=     5000   Blocks=     2450
x=    62.143     0.972     0.363     3.068    42.123   335.117     6.866     3.536

Run with Costfactor =     1.950   cub=    54.729   qlb=     0.846
Initial Values
f=    21.818   E=    20.607   cost=    53.571     q=     0.792   tolx=  0.000010   tolf=  0.000015
A=  50.000   alpha=   0.602    c=    0.002   a=   0.020
Final Values
Total time, E=    20.457       cost=    54.733      q=     0.841   Function evaluations=     5000   Blocks=     2460
x=    64.350     0.956     0.602     3.032    50.405   504.265     5.704     3.515

Run with Costfactor =     2.000   cub=    56.132   qlb=     0.846
Initial Values
f=    22.244   E=    19.815   cost=    58.544     q=     0.863   tolx=  0.000010   tolf=  0.000015
A=  50.000   alpha=   0.602    c=    0.002   a=   0.020
Final Values
Total time, E=    20.579       cost=    57.189      q=     0.847   Function evaluations=     5000   Blocks=     2444
x=    99.689     0.962     0.914     3.803    50.305   505.697     6.996     3.467

Run with Costfactor =     2.050   cub=    57.535   qlb=     0.846
Initial Values
f=    19.888   E=    19.686   cost=    57.718     q=     0.866   tolx=  0.000010   tolf=  0.000015
```

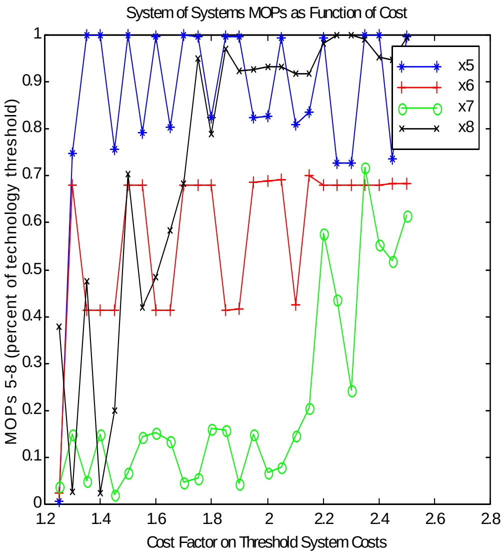
```
A=  50.000   alpha=   0.602    c=   0.002   a=   0.020
Final Values
Total time, E=    19.423       cost=    57.575      q=    0.847   Function evaluations=     5000   Blocks=    2473
x=    68.126     0.959     0.326     3.044    42.300   507.746     6.124     3.464

Run with Costfactor =      2.100   cub=    58.939   qlb=     0.846
Initial Values
f=    29.146   E=    21.381   cost=    51.182      q=    0.853   tolx=  0.000010   tolf=  0.000015
A=  50.000   alpha=   0.602    c=   0.002   a=   0.020
Final Values
Total time, E=    19.642       cost=    59.005      q=    0.853   Function evaluations=     5000   Blocks=    2389
x=    69.626     0.980     0.250     3.206    51.167   339.978     5.719     3.576

Run with Costfactor =      2.150   cub=    60.342   qlb=     0.846
Initial Values
f=    22.147   E=    19.227   cost=    57.476      q=    0.792   tolx=  0.000010   tolf=  0.000015
A=  50.000   alpha=   0.602    c=   0.002   a=   0.020
Final Values
Total time, E=    19.033       cost=    60.448      q=    0.863   Function evaluations=     5000   Blocks=    2429
x=    71.258     0.980     0.250     3.031    49.908   512.145     5.377     3.575

Run with Costfactor =      2.200   cub=    61.745   qlb=     0.846
Initial Values
f=    25.242   E=    21.096   cost=    57.619      q=    0.866   tolx=  0.000010   tolf=  0.000015
A=  50.000   alpha=   0.602    c=   0.002   a=   0.020
Final Values
Total time, E=    19.042       cost=    61.897      q=    0.866   Function evaluations=     5000   Blocks=    2448
x=    68.379     0.980     0.250     3.237    42.274   500.831     3.130     3.116

Run with Costfactor =      2.250   cub=    63.148   qlb=     0.846
Initial Values
f=    26.027   E=    19.099   cost=    56.276      q=    0.790   tolx=  0.000010   tolf=  0.000015
A=  50.000   alpha=   0.602    c=   0.002   a=   0.020
Final Values
Total time, E=    18.342       cost=    61.312      q=    0.854   Function evaluations=     5000   Blocks=    2391
x=    68.871     0.973     0.257     3.000    55.084   500.842     3.983     3.000

Run with Costfactor =      2.300   cub=    64.552   qlb=     0.846
Initial Values
f=    24.204   E=    19.125   cost=    59.488      q=    0.861   tolx=  0.000010   tolf=  0.000015
A=  50.000   alpha=   0.602    c=   0.002   a=   0.020
Final Values
Total time, E=    18.121       cost=    61.619      q=    0.855   Function evaluations=     5000   Blocks=    2395
x=    71.850     0.973     0.257     3.007    55.124   500.295     5.139     3.000

Run with Costfactor =      2.350   cub=    65.955   qlb=     0.846
Initial Values
f=    25.381   E=    19.502   cost=    60.127      q=    0.795   tolx=  0.000010   tolf=  0.000015
A=  50.000   alpha=   0.602    c=   0.002   a=   0.020
Final Values
Total time, E=    18.063       cost=    64.211      q=    0.849   Function evaluations=     5000   Blocks=    2389
x=    72.962     0.961     0.269     3.021    42.005   500.999     2.284     3.071

Run with Costfactor =      2.400   cub=    67.358   qlb=     0.846
Initial Values
f=    18.302   E=    16.198   cost=    69.412      q=    0.795   tolx=  0.000010   tolf=  0.000015
A=  50.000   alpha=   0.602    c=   0.002   a=   0.020
Final Values
Total time, E=    16.936       cost=    67.868      q=    0.852   Function evaluations=     5000   Blocks=    2457
x=    99.799     0.965     0.378     3.033    42.000   500.886     3.283     3.320

Run with Costfactor =      2.450   cub=    68.762   qlb=     0.846
Initial Values
f=    18.166   E=    16.985   cost=    67.637      q=    0.791   tolx=  0.000010   tolf=  0.000015
A=  50.000   alpha=   0.602    c=   0.002   a=   0.020
Final Values
Total time, E=    17.283       cost=    69.190      q=    0.861   Function evaluations=     5000   Blocks=    2465
```

```
x=    99.760     0.980     0.250     3.265    54.661   501.335     3.478     3.374

Run with Costfactor =      2.500   cub=    70.165   qlb=     0.846
Initial Values
f=    16.824   E=    16.191   cost=    69.584      q=     0.795   tolx=  0.000010   tolf=  0.000015
A=  50.000   alpha=   0.602    c=   0.002   a=   0.020
Final Values
Total time, E=    16.377       cost=    70.334      q=     0.850   Function evaluations=     5000   Blocks=    2467
x=    99.906     0.963     0.283     3.049    42.083   501.176     2.910     3.050

cost factor   cub          E      cost       qfinal      fun. evals  #blocks
     1.25    35.082    71.155    34.986     0.705      5000      2419
     1.30    36.486    37.667    40.349     0.846      5000      2422
     1.35    37.889    34.544    37.804     0.810      5000      2460
     1.40    39.292    33.915    41.980     0.846      5000      2418
     1.45    40.696    32.065    40.746     0.851      5000      2453
     1.50    42.099    23.240    46.095     0.846      5000      2439
     1.55    43.502    30.662    43.686     0.862      5000      2488
     1.60    44.906    26.708    45.738     0.848      5000      2432
     1.65    46.309    25.501    47.260     0.852      5000      2438
     1.70    47.712    23.766    47.672     0.830      5000      2471
     1.75    49.115    22.057    51.587     0.846      5000      2434
     1.80    50.519    20.791    54.596     0.846      5000      2429
     1.85    51.922    18.993    57.732     0.847      5000      2429
     1.90    53.325    20.388    53.791     0.851      5000      2450
     1.95    54.729    20.457    54.733     0.841      5000      2460
     2.00    56.132    20.579    57.189     0.847      5000      2444
     2.05    57.535    19.423    57.575     0.847      5000      2473
     2.10    58.939    19.642    59.005     0.853      5000      2389
     2.15    60.342    19.033    60.448     0.863      5000      2429
     2.20    61.745    19.042    61.897     0.866      5000      2448
     2.25    63.148    18.342    61.312     0.854      5000      2391
     2.30    64.552    18.121    61.619     0.855      5000      2395
     2.35    65.955    18.063    64.211     0.849      5000      2389
     2.40    67.358    16.936    67.868     0.852      5000      2457
     2.45    68.762    17.283    69.190     0.861      5000      2465
     2.50    70.165    16.377    70.334     0.850      5000      2467
```


System of Systems MOE as Function of Cost
Time to Complete Mission (hours)
Cost Factor on Threshold System Costs
10
20
30
40
50
60
70
80
1.2
1.4
1.6
1.8
2
2.2
2.4
2.6
2.8


System of Systems MOE as Function of Cost
Time to Complete Mission (hours)
Cost ($M)
10
20
30
40
50
60
70
80
35
40
45
50
55
60
65
70
75


System of Systems MOPs as Function of Cost
MOPs 1-4 (percent of technology threshold)
Cost Factor on Threshold System Costs
x1
x2
x3
x4
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
1.2
1.4
1.6
1.8
2
2.2
2.4
2.6
2.8


System of Systems MOPs as Function of Cost
MOPs 5-8 (percent of technology threshold)
Cost Factor on Threshold System Costs
x5
x6
x7
x8
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
1.2
1.4
1.6
1.8
2
2.2
2.4
2.6
2.8

APPENDIX E

SECOND ORDER CONSTRAINED SPSA MATLAB® CODE AND RESULTS

```
%  MCM System of Systems
      %  SPSA utilizing penalty function MCMFUNPF that was used with MCM42PF.m
      %  Except that the penalty functions have a gain that increases with k, iteration
number.
      %  Filename:  \matlab\dissertation\MCMSPSA.m
      %This file implements constrained, second-order SPSA, with fixed number of
iterations.
      %  Output file initialization
clear
fid=fopen('c:\matlab\dissertation\output 2SPSA.doc','w');
fprintf(fid,'output from execution of MCM2spsaAvgFinal.m with stepped blocktol (0.2,0.1),
averaging of last 5 iterates\n\n');
      %================================================================================
      %
      %        MCM Initialization
      %
      %================================================================================
      %
      % x(1)=p11=  S1 area coverage rate (nm^2/day)
      % x(2)=p12=  S1 classification probability
      % x(3)=p13=  S1 FAR (#/nm^2)
      % x(4)=p14=  S1 time to classify (min)
      % x(5)=p15=  S1 navigation accuracy (yards)
      % x(6)=p21=  S2 Re-acquisition range (yards)
      % x(7)=p22=  S2 time to prosecute a false target (min)
      % x(8)=p23=  S2 time to neutralize (min)
      pd=0.90;     %S1 detection system probability of detection
      p=8;                    % dimension of x
      xstar=[10;0.9;2.0;9.17;90;75;6.6;10.0];
       %this is the threshold system.  But too far from opimal for SPSA to start with.
x0=[70;.95;1.0;5.0;45;430;3.5;6]; %pretty near optimal for costfactor around 1.6.
xoptimal=[10.000     0.900     2.000     9.170    90.000    75.000     6.600    10.000;...
      57.379     0.963     2.000     4.983    45.108   417.391     7.000     8.793;...
      58.915     0.963     2.000     3.915    44.820   417.374     7.000     8.555;...
      60.058     0.963     2.000     3.017    44.418   417.371     7.000     8.355;...
      65.201     0.963     2.000     3.000    44.632   417.375     7.000     7.162;...
      67.996     0.963     2.000     3.000    44.802   417.387     7.000     6.282;...
      68.194     0.962     1.342     3.000    43.341   417.304     7.000     6.213;...
      69.825     0.962     1.249     3.000    42.959   417.274     7.000     5.603;...
      71.172     0.962     1.181     3.000    42.596   417.237     7.000     5.045;...
      72.316     0.961     1.126     3.000    42.245   417.195     7.000     4.532;...
      73.310     0.961     1.080     3.000    42.000   417.254     7.000     4.055;...
      74.189     0.961     1.040     3.000    42.000   417.519     7.000     3.607;...
      74.979     0.961     1.004     3.000    42.000   417.757     7.000     3.184;...
      77.328     0.961     0.901     3.000    42.000   418.623     7.000     3.000;...
      80.440     0.961     0.765     3.000    42.000   420.090     7.000     3.000;...
      83.033     0.961     0.651     3.000    42.000   421.627     7.000     3.000;...
      85.261     0.961     0.552     3.000    42.000   423.201     7.000     3.000;...
      87.217     0.961     0.463     3.000    42.000   424.790     7.000     3.000;...
      88.965     0.961     0.383     3.000    42.000   426.399     7.000     3.000;...
      85.153     0.963     0.570     3.000    45.839   417.389     3.948     3.000;...
      87.149     0.963     0.481     3.000    45.694   417.395     3.921     3.000;...
      88.941     0.963     0.400     3.000    45.508   417.412     3.894     3.000;...
      90.539     0.963     0.326     3.000    45.239   417.380     3.876     3.000;...
      92.010     0.963     0.257     3.000    44.947   417.360     3.857     3.000;...
      95.302     0.963     0.250     3.000    45.572   417.461     3.797     3.000;...
      98.430     0.964     0.250     3.000    46.343   417.398     3.737     3.000;...
      100.000     0.973     0.250     3.000    64.008   412.202     3.162     3.000];
%fprintf(fid,'x0=\n');fprintf(fid,'%10.3f',xoptimal);fprintf(fid,'\n');
vlb=[10.0;0.9;0.25;3.0;42;75;1.0;3.0];              % lower bound constraint on x
vub=[100;0.98;2.0;9.17;90;700;7.0;10.0];            % upper bound constraint on x
fprintf(fid,'vlb=\n');fprintf(fid,'%10.3f',vlb);fprintf(fid,'\n');
fprintf(fid,'vub=\n');fprintf(fid,'%10.3f',vub);fprintf(fid,'\n\n');
%  Compute px1, polynomial fit cost function parameters for x(1)
x1=[10,57,82,94];
y1=[3,4.483,7.655,11.445];
px1=polyfit(x1,y1,3);
%  Compute px2, polynomial fit cost function parameters for x(2)
```

```
x1=[0.9,0.93,0.96,0.98];
%y1=[3,4.483,7.655,11.445];%original PBCM for Pc
y1=[.2,.5,1.4,2.2];  %revised PBCM that reflects COTS/NDI development
px2=polyfit(x1,y1,2);
%  Compute px3, polynomial fit cost function parameters for x(3)
x1=[2,1,0.5,0.25];
y1=[8,10.319,13.592,16.429];
px3=polyfit(x1,y1,3);
%  Compute px4, polynomial fit cost function parameters for x(4)
x1=[3.513,3.89,4.77,9.17];
y1=[9.191,8.190,7.574,5.0];
px4=polyfit(x1,y1,2);
%  Compute px5, polynomial fit cost function parameters for x(5)
x1=[90,60,48,42];
y1=[0.050,0.250,0.450,0.550];
px5=polyfit(x1,y1,2);
%  Compute px6, polynomial fit cost function parameters for x(6)
x1=[129,457,622,75];
y1=[3,4.8,7.655,1.5];
px6=polyfit(x1,y1,3);
%  Compute px7, polynomial fit cost function parameters for x(7)
x1=[6.6,4.4,3.3,2.64,1.32];
y1=[5.0,7.5,8.190,9.191,16.621];
px7=polyfit(x1,y1,3);
%  Compute px8, polynomial fit cost function parameters for x(8)
x1=[10,8,7,5,3];
y1=[5.3,6,7,10,15];
px8=polyfit(x1,y1,2);
%compute threshold system cost
cost0=polyval(px1,xstar(1))+polyval(px2,xstar(2))+polyval(px3,xstar(3))+polyval(px4,xstar(
       4))+polyval(px5,xstar(5))+polyval(px6,xstar(6))+polyval(px7,xstar(7))+polyval(px8,
       xstar(8))
%Insert costfactor loop
i=0;
for costfactor=1.2:0.05:2.5 %Note problem is infeasible with costfactor<1.25
   i=i+1;
      for j=1:p  %start 2SPSA "near the optimal solution"--perturb randomly by 20%
      del=2*round(rand)-1; %this is +/-1 random number
      x0(j)=(1+0.20*del)*xoptimal(i,j); % results in either 120% or 80% of optimal.
      if x0(j)<vlb(j);x0(j)=vlb(j);end  %make sure x0 is feasible
      if x0(j)>vub(j);x0(j)=vub(j);end
   end
costfactor
fprintf(fid,'\n');
cub=costfactor*cost0;         %system of systems cost constraint
qlb=0.846;             %quality constraint lower bound
fprintf(fid,'Run with Costfactor = ');fprintf(fid,'%10.3f',costfactor);
fprintf(fid,'   cub=');fprintf(fid,'%10.3f',cub);
fprintf(fid,'   qlb=');fprintf(fid,'%10.3f\n',qlb);
sminefield=20;        %minefield area, (nm^2)
m0=100;               %number of mines in minefield, initially
lambda=m0/sminefield; %mine density, (#/nm^2)
dmine=600;            %average distance between mines, (yards)
vtransit=7;           %S1 vehicle transit speed (knots)
ttransit=dmine/(2000*vtransit);    %transit time during classification (hours)
lambdaft=1.0;         %false target density (#/nm^2)
% Define parameters for the objective function
p1=pd;
p2=lambda;
p3=lambdaft;
p4=ttransit;
p5=sminefield;
p6=cub;
p7=qlb;
p8=1;  %Placeholder---p8 will be iteration number of SPSA algorithm.
[f,g]=mcmfunpf2s(x0,p1,p2,p3,p4,p5,p6,p7,p8);
fprintf(fid,'Initial Values\n');
fprintf(fid,'x0=');fprintf(fid,'%10.3f',x0);fprintf(fid,'\n');
x=x0;
cost=polyval(px1,x(1))+polyval(px2,x(2))+polyval(px3,x(3))+polyval(px4,x(4))+polyval(px5,x
      (5))+polyval(px6,x(6))+polyval(px7,x(7))+polyval(px8,x(8))
q=p1*x(2)*exp(-x(5)/(4.481*x(6)));
f1 = (p5/60)*(24*60/x(1) + p2*x(2)*x(4)*p1 + (2*x(4)-p4)*((1-x(2))*p1*p2 + x(3) + p1*p3));
f2 = (p5/60)*(p1*x(2)*x(8)*p2*exp(-x(5)/(4.481*x(6)))+(1-exp(-
       x(5)/(4.481*x(6))))*p1*x(2)*x(7)*p2 + (1-x(2))*(x(3)+p1*p3)*x(7));
```

```
E=f1+f2;
q=p1*x0(2)*exp(-x0(5)/(4.481*x0(6)));
fprintf(fid,'f=');fprintf(fid,'%10.3f',f);fprintf(fid,'   E=');fprintf(fid,'%10.3f',E);
fprintf(fid,'   cost=');fprintf(fid,'%10.3f',cost);fprintf(fid,'
q=');fprintf(fid,'%10.3f',q);
%  Print out threshold system values for f and constraints, g.
%===============================================================================
%
%              SPSA Initialization
%
%===============================================================================
for k1=1:50;s1(k1)=0;s2(k1)=0;s3(k1)=0;s4(k1)=0;s5(k1)=0;s6(k1)=0;s7(k1)=0;s8(k1)=0;end
n=1000;               % maximum allowable number of iterations
N=0;
blocks=0;             % number of blocked iterations
tolf=0.00005;         % convergence tolerances
tolx=0.00001;
A=10;                 % Spall cookbook, about 5% of max iterations
alpha=0.602;          % Spall cookbook values
gamma=0.101;
c=0.005;                      % Value for c used by Ilenda was 0.05.  Critical parameter.
%                       May need to scale whole problem's x so that the optimal
%                       values lie in a similar range.
a=50;
fprintf(fid,'   tolx=');fprintf(fid,'%10.6f',tolx);
fprintf(fid,'   tolf=');fprintf(fid,'%10.6f\n',tolf);
fprintf(fid,'A=');fprintf(fid,'%8.3f',A);fprintf(fid,'alpha=');
fprintf(fid,'%8.3f',alpha);fprintf(fid,'    c=');fprintf(fid,'%8.3f',c);
fprintf(fid,'   a=');fprintf(fid,'%8.3f\n',a);
%2SPSA Initialization
%number of individual gradient/Hessian estimates to be averaged at each iter.
gH_avg=3;
%rand('seed',31415297) %this makes each run the same, if fixed number of iterations.
%randn('seed',111113)
%
%
%the loop 1:n does 2SPSA following guidelines in Spall, 1997 CISS.
%
%lines below initialize various recuresions for the gradient/Hess. averaging
%and for final error reporting based on the average of the solutions for
%"cases" replications.
meanHbar=0;
errthetaH=0;
errtheta=0;
losstheta=0;
lossthetaH=0;
theta_0=x0;
[f,g]=mcmfunpf2s(theta_0,p1,p2,p3,p4,p5,p6,p7,p8)           % evaluate f right away
%DUMMY STATEMENT FOR SETTING DIMENSIONS OF Hhat (AVOIDS OCCASIONAL
%ERROR MESSAGES)
Hhat=eye(p);
theta=theta_0;
thetaH=theta;
Hbar=eye(p);
%================================================================================
%
%              SPSA Loop
%
%================================================================================
k1=0;  %counter for last fifty iterations
for k=1:n;
   p8=k;
%fprintf(fid,'\n');fprintf(fid,'==================================================\n');
%fprintf(fid,'iteration #');fprintf(fid,'%3.0f\n',k);
ak=a/(k+A)^alpha;
ck=c/k^gamma;
ghatinput=0;
Hhatinput=0;
% GENERATION OF AVERAGED GRADIENT AND HESSIAN (NO AVERAGING IF gH_avg=1)
    for m=1:gH_avg
      delta=2*round(rand(p,1))-1;
      thetaplus=thetaH+ck*delta;
      thetaminus=thetaH-ck*delta;
      % check for infeasible vectors
      for j=1:p;
```

```
         if thetaplus(j)<vlb(j);
            thetaplus(j)=vlb(j);
         end;
         if thetaminus(j)<vlb(j);
            thetaminus(j)=vlb(j);
         end;
         if thetaplus(j)>vub(j);
            thetaplus(j)=vub(j);
         end;
         if thetaminus(j)>vub(j);
            thetaminus(j)=vub(j);
         end;
      end;
      [yplus,g]=mcmfunpf2s(thetaplus,p1,p2,p3,p4,p5,p6,p7,p8);
      [yminus,g]=mcmfunpf2s(thetaminus,p1,p2,p3,p4,p5,p6,p7,p8);
      ghat=(yplus-yminus)./(2*ck*delta);
% GENERATE THE HESSIAN UPDATE
      deltatilda=2*round(rand(p,1))-1;
      thetaplustilda=thetaplus+ck*deltatilda;
      thetaminustilda=thetaminus+ck*deltatilda;
       %       check for infeasible vectors
       for j=1:p;
          if thetaplustilda(j)<vlb(j);
             thetaplustilda(j)=vlb(j);
          end;
          if thetaminustilda(j)<vlb(j);
             thetaminustilda(j)=vlb(j);
          end;
          if thetaplustilda(j)>vub(j);
             thetaplustilda(j)=vub(j);
          end;
          if thetaminustilda(j)>vub(j);
             thetaminustilda(j)=vub(j);
          end;
    end;
% LOSS FUNCTION CALLS
      [yplustilda,g]=mcmfunpf2s(thetaplustilda,p1,p2,p3,p4,p5,p6,p7,p8);
      [yminustilda,g]=mcmfunpf2s(thetaminustilda,p1,p2,p3,p4,p5,p6,p7,p8);
      ghatplus=(yplustilda-yplus)./(ck*deltatilda);
      ghatminus=(yminustilda-yminus)./(ck*deltatilda);
% STATEMENT PROVIDING AN AVERAGE OF SP GRAD. APPROXS. PER ITERATION
      ghatinput=((m-1)/m)*ghatinput+ghat/m;
      deltaghat=ghatplus-ghatminus;
      for j=1:p
         Hhat(:,j)=deltaghat(j)./(2*ck*delta);
      end
      Hhat=.5*(Hhat+Hhat');
      Hhatinput=((m-1)/m)*Hhatinput+Hhat/m;
    end
    Hbar=((k-N)/(k-N+1))*Hbar+Hhatinput/(k-N+1);
%   THE THETA UPDATE (FORM BELOW USES NAIVE DIRECT HESSIAN INVERSE FORM;
%   LARGER-SCALE IMPLEMENTATIONS SHOULD USE MORE NUMERICALLY EFFICIENT APPROACH
%   SUCH AS GAUSSIAN ELIMINATION TO AVOID DIRECT COMPUTATION OF HESSIAN INVERSE)
    thetaHlag=thetaH;
    Hbarbar=sqrtm(Hbar*Hbar)+.000001*eye(p)/k;
    update=inv(Hbarbar)*ghatinput;
    thetaH=thetaH-ak*update;
    %   check for infeasible updated estimate
       for j=1:p;
               if thetaH(j)<vlb(j);
               thetaH(j)=vlb(j);
               end;
               if thetaH(j)>vub(j);
               thetaH(j)=vub(j);
               end;
   end;
   thetaH;
   xkp1=thetaH;x=thetaHlag;
   [fkp1,g]=mcmfunpf2s(xkp1,p1,p2,p3,p4,p5,p6,p7,p8);
   %Blocking test on f.
   if k<200; blocktol=0.2; else blocktol=0.1;end;
   if fkp1<f+blocktol;
       x=xkp1; % Accept new estimate.  Otherwise block the update passively.
       f=fkp1;
       k1=k1+1;  %store last 10 iterates for averaging
       if k1<6;
       s1(k1)=x(1);s2(k1)=x(2);s3(k1)=x(3);s4(k1)=x(4);s5(k1)=x(5);s6(k1)=x(6);s7(k1)=x(7
              );s8(k1)=x(8);
      else
        for j=1:4
       s1(j)=s1(j+1);s2(j)=s2(j+1);s3(j)=s3(j+1);s4(j)=s4(j+1);s5(j)=s5(j+1);
              s6(j)=s6(j+1);s7(j)=s7(j+1);s8(j)=s8(j+1);
       s1(5)=x(1);s2(5)=x(2);s3(5)=x(3);s4(5)=x(4);s5(5)=x(5);s6(5)=x(6);
              s7(5)=x(7);s8(5)=x(8);
         end
      end
   else;
      blocks=blocks+1;
      thetaH=thetaHlag;
   end;
end
   %average the last 5 unblocked iterates to get final estimate
   if k1>5; k1=5; end
   sum1=0;sum2=0;sum3=0;sum4=0;sum5=0;sum6=0;sum7=0;sum8=0;
   for j=1:k1
       sum1=sum1+s1(j);sum2=sum2+s2(j);sum3=sum3+s3(j);sum4=sum4+s4(j);
       sum5=sum5+s5(j);sum6=sum6+s6(j);sum7=sum7+s7(j);sum8=sum8+s8(j);
   end
x(1)=sum1/k1;x(2)=sum2/k1;x(3)=sum3/k1;x(4)=sum4/k1;x(5)=sum5/k1;
       x(6)=sum6/k1;x(7)=sum7/k1;x(8)=sum8/k1;
x  %x is now average over last 5 unblocked iterates, or all iterates if less than 5 are
blocked.
[f,g]=mcmfunpf2s(x,p1,p2,p3,p4,p5,p6,p7,p8)
fprintf(fid,'Final Values\n');
cost=polyval(px1,x(1))+polyval(px2,x(2))+polyval(px3,x(3))+polyval(px4,x(4))+polyval(px5,x
       (5))+polyval(px6,x(6))+polyval(px7,x(7))+polyval(px8,x(8))
q=p1*x(2)*exp(-x(5)/(4.481*x(6)))
f1 = (p5/60)*(24*60/x(1) + p2*x(2)*x(4)*p1 + (2*x(4)-p4)*((1-x(2))*p1*p2 + x(3) + p1*p3));
f2 = (p5/60)*(p1*x(2)*x(8)*p2*exp(-x(5)/(4.481*x(6)))+(1-exp(-
       x(5)/(4.481*x(6))))*p1*x(2)*x(7)*p2 + (1-x(2))*(x(3)+p1*p3)*x(7));
E=f1+f2
fprintf(fid,'Total time, E=');fprintf(fid,'%10.3f',E);
fprintf(fid,'       cost=');fprintf(fid,'%10.3f',cost);
fprintf(fid,'      q=');fprintf(fid,'%10.3f',q);
fprintf(fid,'   Function evaluations= ');fprintf(fid,'%8.0f',5*k);
fprintf(fid,'   Blocks=');fprintf(fid,'%8.0f\n',blocks);
fprintf(fid,'x=');fprintf(fid,'%10.3f',x);fprintf(fid,'\n');
cf(i)=costfactor; fval(i)=E; costval(i)=cub;fevals(i)=5*k; costi(i)=cost;
qfinal(i)=q;nblock(i)=blocks;
z1(i)=x(1);z2(i)=x(2);z3(i)=x(3);z4(i)=x(4);z5(i)=x(5);z6(i)=x(6);z7(i)=x(7);z8(i)=x(8);
z1(i)=abs((z1(i)-xstar(1))/(vlb(1)-vub(1)));
z2(i)=abs((z2(i)-xstar(2))/(vlb(2)-vub(2)));
z3(i)=abs((z3(i)-xstar(3))/(vlb(3)-vub(3)));
z4(i)=abs((z4(i)-xstar(4))/(vlb(4)-vub(4)));
z5(i)=abs((z5(i)-xstar(5))/(vlb(5)-vub(5)));
z6(i)=abs((z6(i)-xstar(6))/(vlb(6)-vub(6)));
z7(i)=abs((z7(i)-xstar(7))/(vlb(7)-vub(7)));
z8(i)=abs((z8(i)-xstar(8))/(vlb(8)-vub(8)));
end
%  Plot option 1:  Plot System of systems MOE as CAIV
figure
plot(cf,fval,'-*b')
title('System of Systems MOE as Function of Cost')
xlabel('Cost Factor on Threshold System Costs')
ylabel('Time to Complete Mission (hours)')
figure
plot(costval,fval,'-*b')
title('System of Systems MOE as Function of Cost')
xlabel('Cost ($M)')
ylabel('Time to Complete Mission (hours)')
%
%
%  Plot option 2:  Plot MOPs as CAIV
figure
plot(cf,z1,'-b*',cf,z2,'-r+',cf,z3,'-go',cf,z4,'-kx')
legend('x1','x2','x3','x4')
title('System of Systems MOPs as Function of Cost')
xlabel('Cost Factor on Threshold System Costs')
ylabel('MOPs 1-4 (percent of technology threshold)')
figure
```

```
plot(cf,z5,'-b*',cf,z6,'-r+',cf,z7,'-go',cf,z8,'-kx')
legend('x5','x6','x7','x8')
title('System of Systems MOPs as Function of Cost')
xlabel('Cost Factor on Threshold System Costs')
ylabel('MOPs 5-8 (percent of technology threshold)')
%  print table of results to file
fprintf(fid,'\n');
fprintf(fid,'cost factor');fprintf(fid,'   cub');fprintf(fid,'          E');fprintf(fid,'
cost');fprintf(fid,'        qfinal');fprintf(fid,'      fun. evals');fprintf(fid,'
#blocks\n');
for j=1:i
       fprintf(fid,'%10.2f',cf(j));fprintf(fid,'%10.3f',costval(j));
       fprintf(fid,'%10.3f',fval(j));fprintf(fid,'%10.3f',costi(j));
       fprintf(fid,'%10.3f',qfinal(j));fprintf(fid,'%10.0f',fevals(j));
       fprintf(fid,'%10.0f\n',nblock(j));
end
status=fclose(fid)
```

```
function [f,g] = mcmfunpf2s(x,p1,p2,p3,p4,p5,p6,p7,p8)
A1=50;
A2=1e3;
kmax=600;
k=p8;               %k is the iteration number of the SPSA algorithm
if k<kmax
   a1=0.5*(A1)*sin((k-1)*pi/kmax-pi/2)+A1/2+1;
   a2=0.5*(A2)*sin((k-1)*pi/kmax-pi/2)+A2/2+1;
else
   a1=A1;
   a2=A2;
end
px1=[4.503408803940725e-005,   -5.386095666335044e-003,    2.159330101073730e-001,
       1.334245377520354e+000];
px2=[2.834645669291690e+002,   -5.076377952756583e+002,    2.274598425197177e+002];
px3=[-2.048380952380911e+000,    9.987333333333214e+000,   -1.794233333333325e+001,
       2.032238095238094e+001];
px4=[1.159691730856429e-001,   -2.175732453433467e+000,    1.520381256718985e+001];
px5=[2.061825086032983e-004,   -3.775958229500408e-002,    1.777803488786043e+000];
px6=[1.504875482450802e-007,   -1.578229837871938e-004,    5.516694369186904e-002,  -
       1.813253427503106e+000];
px7=[ -2.850358103957624e-001,    3.846213159671302e+000,   -1.726423877731832e+001,
       3.334408692656030e+001];
px8=[2.102445277065673e-001,   -4.109593768487483e+000,    2.539723920331297e+001];
f1 = (p5/60)*(24*60/x(1) + p2*x(2)*x(4)*p1 + (2*x(4)-p4)*((1-x(2))*p1*p2 + x(3) + p1*p3));
f2 = (p5/60)*(p1*x(2)*x(8)*p2*exp(-x(5)/(4.481*x(6)))+(1-exp(-
x(5)/(4.481*x(6))))*p1*x(2)*x(7)*p2 + (1-x(2))*(x(3)+p1*p3)*x(7));
E=f1+f2;
% evaluate cost constraint
g1=polyval(px1,x(1))+polyval(px2,x(2))+polyval(px3,x(3))+polyval(px4,x(4))+polyval(px5,x(5
))+polyval     (px6,x(6))+polyval(px7,x(7))+polyval(px8,x(8))-p6;
% evaluate negative of quality constraint
g2 = -p1*x(2)*exp(-x(5)/(4.481*x(6)))+p7;
%E,g1,g2,f
f=f1+f2+A1*abs(g1)+A2*abs(g2);
%f=f1+f2+A1*g1*g1+A2*g2*g2;
g=[];
%p1=pd
%p2=lambda
%p3=lambdaft
%p4=ttransit
%p5=sminefield
%p6=cub
%p7=qlb
%p8=k
```

output from execution of MCM2spsaAvgFinal.m with stepped blocktol (0.2,0.1), averaging of last 5 iterates
09/14/97 3:04 PM   2SPSA Baseline C 9-14-97.doc

```
vlb=
   10.000    0.900    0.250    3.000   42.000   75.000    1.000    3.000
vub=
  100.000    0.980    2.000    9.170   90.000  700.000    7.000   10.000
```

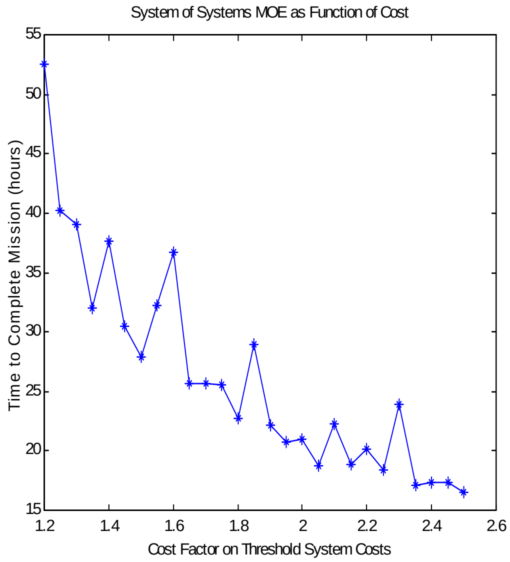
```
Run with Costfactor =     1.200   cub=    33.679   qlb=     0.846
Initial Values
x0=    10.000     0.900     1.600     9.170    72.000    75.000     5.280     8.000
f=   362.523   E=    88.904   cost=    32.051      q=     0.654   tolx=  0.000010   tolf=  0.000050
A=  10.000   alpha=   0.602    c=   0.005    a=  50.000
Final Values
Total time, E=    52.545       cost=    33.695      q=     0.737   Function evaluations=     5000   Blocks=     978
x=    18.548     0.980     1.739     4.921    61.902    76.895     6.994     7.087

Run with Costfactor =     1.250   cub=    35.082   qlb=     0.846
Initial Values
x0=    68.855     0.980     2.000     5.980    42.000   500.869     5.600    10.000
f=   315.989   E=    42.326   cost=    40.163      q=     0.866   tolx=  0.000010   tolf=  0.000050
A=  10.000   alpha=   0.602    c=   0.005    a=  50.000
Final Values
Total time, E=    40.251       cost=    35.109      q=     0.844   Function evaluations=     5000   Blocks=     980
x=    64.176     0.956     1.999     5.600    43.213   502.870     7.000     9.053

Run with Costfactor =     1.300   cub=    36.486   qlb=     0.846
Initial Values
x0=    47.132     0.900     1.600     3.132    53.784   333.899     7.000    10.000
f=   110.994   E=    34.470   cost=    36.247      q=     0.781   tolx=  0.000010   tolf=  0.000050
A=  10.000   alpha=   0.602    c=   0.005    a=  50.000
Final Values
Total time, E=    39.047       cost=    36.483      q=     0.846   Function evaluations=     5000   Blocks=     917
x=    45.805     0.974     1.585     4.488    52.753   333.880     6.956     9.758

Run with Costfactor =     1.350   cub=    37.889   qlb=     0.846
Initial Values
x0=    72.070     0.900     2.000     3.000    53.302   333.897     5.600     6.684
f=   356.863   E=    26.876   cost=    43.202      q=     0.782   tolx=  0.000010   tolf=  0.000050
A=  10.000   alpha=   0.602    c=   0.005    a=  50.000
Final Values
Total time, E=    32.061       cost=    38.170      q=     0.826   Function evaluations=     5000   Blocks=     990
x=    75.543     0.948     1.911     3.656    48.934   333.321     6.960     9.061

Run with Costfactor =     1.400   cub=    39.292   qlb=     0.846
Initial Values
x0=    78.241     0.980     1.600     3.000    42.000   333.900     5.600     8.594
f=   352.992   E=    28.293   cost=    45.555      q=     0.858   tolx=  0.000010   tolf=  0.000050
A=  10.000   alpha=   0.602    c=   0.005    a=  50.000
Final Values
Total time, E=    37.628       cost=    39.292      q=     0.846   Function evaluations=     5000   Blocks=     676
x=    83.273     0.967     1.956     5.039    42.065   333.152     6.858     9.937

Run with Costfactor =     1.450   cub=    40.696   qlb=     0.846
Initial Values
x0=    81.595     0.900     1.600     3.600    42.000   500.864     5.600     7.538
f=   279.433   E=    28.376   cost=    44.696      q=     0.795   tolx=  0.000010   tolf=  0.000050
A=  10.000   alpha=   0.602    c=   0.005    a=  50.000
Final Values
Total time, E=    30.482       cost=    40.697      q=     0.846   Function evaluations=     5000   Blocks=     603
x=    79.004     0.959     1.608     4.091    45.503   497.790     6.941     7.646

Run with Costfactor =     1.500   cub=    42.099   qlb=     0.846
Initial Values
x0=    81.833     0.980     1.610     3.000    52.009   333.843     7.000     7.456
f=    74.647   E=    26.493   cost=    42.945      q=     0.852   tolx=  0.000010   tolf=  0.000050
A=  10.000   alpha=   0.602    c=   0.005    a=  50.000
Final Values
Total time, E=    27.900       cost=    42.105      q=     0.846   Function evaluations=     5000   Blocks=     569
x=    81.656     0.973     1.522     3.270    52.242   333.536     6.959     7.974

Run with Costfactor =     1.550   cub=    43.502   qlb=     0.846
Initial Values
x0=    83.790     0.980     1.499     3.000    42.000   333.819     5.600     6.724
```

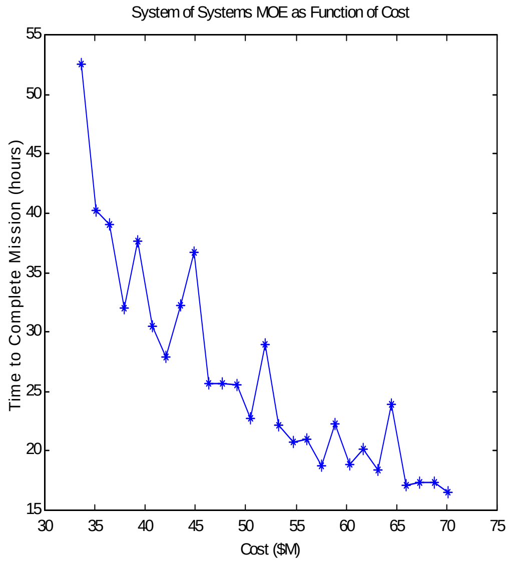
```
f=  295.128   E=   25.008   cost=   48.673     q=    0.858   tolx=  0.000010   tolf=  0.000050
A=  10.000   alpha=   0.602   c=   0.005   a=  50.000
Final Values
Total time, E=    32.265       cost=    43.414     q=    0.851   Function evaluations=     5000   Blocks=     983
x=    86.095     0.973     0.921     5.707    42.042   329.523     6.847     7.476

Run with Costfactor =      1.600   cub=    44.906   qlb=     0.846
Initial Values
x0=    85.406     0.900     0.945     3.600    42.000   500.684     7.000     4.036
f=  291.830   E=   21.908   cost=   49.284     q=    0.795   tolx=  0.000010   tolf=  0.000050
A=  10.000   alpha=   0.602   c=   0.005   a=  50.000
Final Values
Total time, E=    36.752       cost=    44.906     q=    0.846   Function evaluations=     5000   Blocks=     884
x=    75.642     0.958     0.935     8.797    42.201   500.654     6.944     3.941

Run with Costfactor =      1.650   cub=    46.309   qlb=     0.846
Initial Values
x0=    86.779     0.900     0.901     3.600    50.694   333.756     7.000     3.626
f=  280.516   E=   21.228   cost=   50.235     q=    0.783   tolx=  0.000010   tolf=  0.000050
A=  10.000   alpha=   0.602   c=   0.005   a=  50.000
Final Values
Total time, E=    25.668       cost=    46.311     q=    0.846   Function evaluations=     5000   Blocks=     699
x=    86.141     0.972     1.941     3.702    49.255   332.978     6.943     4.892

Run with Costfactor =      1.700   cub=    47.712   qlb=     0.846
Initial Values
x0=    58.648     0.980     1.296     3.000    50.400   500.705     5.600     4.866
f=  105.715   E=   24.393   cost=   49.010     q=    0.862   tolx=  0.000010   tolf=  0.000050
A=  10.000   alpha=   0.602   c=   0.005   a=  50.000
Final Values
Total time, E=    25.618       cost=    47.712     q=    0.846   Function evaluations=     5000   Blocks=     822
x=    58.614     0.960     1.043     3.867    48.174   500.120     6.313     4.302

Run with Costfactor =      1.750   cub=    49.115   qlb=     0.846
Initial Values
x0=    89.027     0.980     1.248     3.000    42.000   334.015     5.600     4.328
f=  327.869   E=   20.740   cost=   55.026     q=    0.858   tolx=  0.000010   tolf=  0.000050
A=  10.000   alpha=   0.602   c=   0.005   a=  50.000
Final Values
Total time, E=    25.512       cost=    49.116     q=    0.846   Function evaluations=     5000   Blocks=     673
x=    90.694     0.966     0.988     3.587    42.052   337.661     6.335     6.942

Run with Costfactor =      1.800   cub=    50.519   qlb=     0.846
Initial Values
x0=    59.983     0.980     0.803     3.600    50.400   501.308     7.000     3.000
f=   47.558   E=   22.192   cost=   50.697     q=    0.862   tolx=  0.000010   tolf=  0.000050
A=  10.000   alpha=   0.602   c=   0.005   a=  50.000
Final Values
Total time, E=    22.765       cost=    50.518     q=    0.846   Function evaluations=     5000   Blocks=     885
x=    60.105     0.961     0.896     3.630    50.130   501.648     6.663     3.148

Run with Costfactor =      1.850   cub=    51.922   qlb=     0.846
Initial Values
x0=    92.794     0.900     1.081     3.000    42.000   502.348     5.600     3.600
f=  331.992   E=   19.330   cost=   57.156     q=    0.795   tolx=  0.000010   tolf=  0.000050
A=  10.000   alpha=   0.602   c=   0.005   a=  50.000
Final Values
Total time, E=    28.971       cost=    51.925     q=    0.846   Function evaluations=     5000   Blocks=     973
x=    93.851     0.958     2.000     4.860    42.234   505.560     5.539     4.632

Run with Costfactor =      1.900   cub=    53.325   qlb=     0.846
Initial Values
x0=    64.352     0.980     0.612     3.600    42.000   336.072     5.600     3.600
f=   84.933   E=   22.002   cost=   54.349     q=    0.858   tolx=  0.000010   tolf=  0.000050
A=  10.000   alpha=   0.602   c=   0.005   a=  50.000
Final Values
Total time, E=    22.177       cost=    53.322     q=    0.846   Function evaluations=     5000   Blocks=     735
```

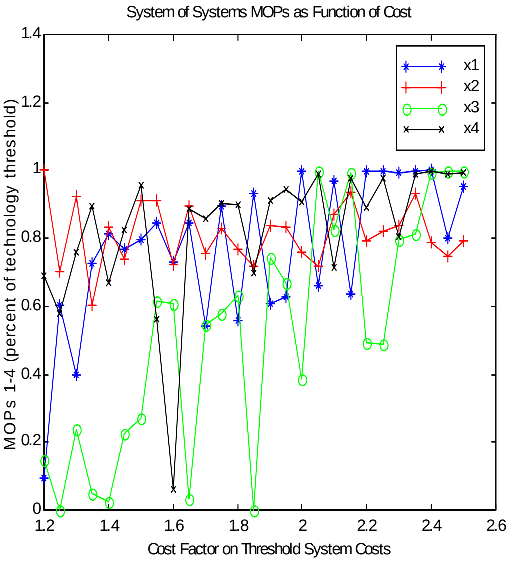
```
x=   64.517    0.967    0.700    3.554   42.470  336.011    4.754    3.679

Run with Costfactor =     1.950   cub=   54.729   qlb=    0.846
Initial Values
x0=   66.426    0.900    0.781    3.000   42.000  337.302    7.000    3.000
f=  326.228  E=   20.098  cost=   49.770     q=    0.788  tolx= 0.000010  tolf= 0.000050
A=  10.000  alpha=  0.602   c=  0.005  a=  50.000
Final Values
Total time, E=   20.777      cost=   54.728     q=    0.846  Function evaluations=    5000  Blocks=    903
x=   66.356    0.967    0.832    3.343   42.248  337.055    4.621    3.016

Run with Costfactor =     2.000   cub=   56.132   qlb=    0.846
Initial Values
x0=  100.000    0.900    0.662    3.600   50.400  507.841    7.000    3.000
f=  223.718  E=   18.994  cost=   59.152     q=    0.792  tolx= 0.000010  tolf= 0.000050
A=  10.000  alpha=  0.602   c=  0.005  a=  50.000
Final Values
Total time, E=   20.968      cost=   56.134     q=    0.846  Function evaluations=    5000  Blocks=    936
x=   99.862    0.961    1.325    3.573   50.507  508.225    6.995    3.479

Run with Costfactor =     2.050   cub=   57.535   qlb=    0.846
Initial Values
x0=   69.774    0.900    0.370    3.000   42.000  509.748    7.000    3.000
f=  232.757  E=   18.790  cost=   54.271     q=    0.795  tolx= 0.000010  tolf= 0.000050
A=  10.000  alpha=  0.602   c=  0.005  a=  50.000
Final Values
Total time, E=   18.779      cost=   57.534     q=    0.846  Function evaluations=    5000  Blocks=    958
x=   69.611    0.958    0.256    3.077   42.334  509.696    6.724    3.145

Run with Costfactor =     2.100   cub=   58.939   qlb=    0.846
Initial Values
x0=  100.000    0.900    0.460    3.600   50.400  341.119    5.600    3.000
f=  351.322  E=   18.395  cost=   64.352     q=    0.784  tolx= 0.000010  tolf= 0.000050
A=  10.000  alpha=  0.602   c=  0.005  a=  50.000
Final Values
Total time, E=   22.231      cost=   58.939     q=    0.846  Function evaluations=    5000  Blocks=    943
x=   97.030    0.970    0.552    4.772   47.497  343.415    6.535    3.494

Run with Costfactor =     2.150   cub=   60.342   qlb=    0.846
Initial Values
x0=   68.122    0.980    0.684    3.600   55.007  333.911    4.738    3.600
f=  329.078  E=   21.734  cost=   54.278     q=    0.850  tolx= 0.000010  tolf= 0.000050
A=  10.000  alpha=  0.602   c=  0.005  a=  50.000
Final Values
Total time, E=   18.912      cost=   60.346     q=    0.846  Function evaluations=    5000  Blocks=    974
x=   67.364    0.975    0.261    3.131   54.230  332.306    5.260    3.006

Run with Costfactor =     2.200   cub=   61.745   qlb=    0.846
Initial Values
x0=  100.000    0.980    0.577    3.000   54.833  500.874    3.137    3.600
f=  243.127  E=   17.628  cost=   65.961     q=    0.861  tolx= 0.000010  tolf= 0.000050
A=  10.000  alpha=  0.602   c=  0.005  a=  50.000
Final Values
Total time, E=   20.120      cost=   61.746     q=    0.846  Function evaluations=    5000  Blocks=    812
x=   99.685    0.963    1.137    3.661   54.956  501.232    4.931    3.103

Run with Costfactor =     2.250   cub=   63.148   qlb=    0.846
Initial Values
x0=  100.000    0.900    0.320    3.000   54.610  500.894    3.115    3.000
f=  326.717  E=   16.347  cost=   68.246     q=    0.791  tolx= 0.000010  tolf= 0.000050
A=  10.000  alpha=  0.602   c=  0.005  a=  50.000
Final Values
Total time, E=   18.415      cost=   63.150     q=    0.846  Function evaluations=    5000  Blocks=    777
x=   99.864    0.966    1.145    3.136   62.152  509.218    3.395    3.041

Run with Costfactor =     2.300   cub=   64.552   qlb=    0.846
Initial Values
```

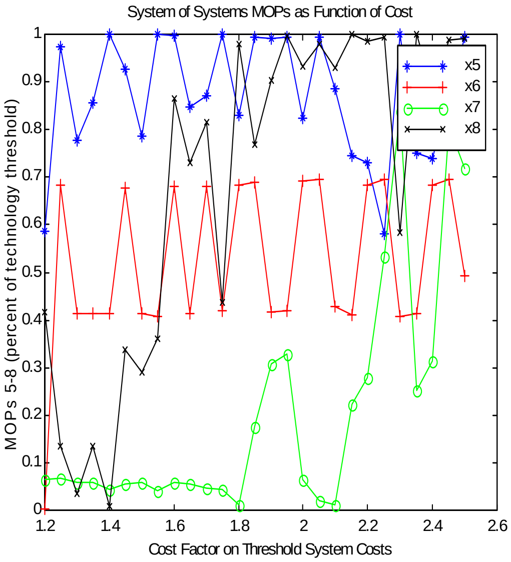
```
x0=   100.000     0.900     0.261     3.000    42.000   333.904     4.651     3.600
f=   166.315   E=    17.128   cost=    66.367      q=     0.788   tolx=  0.000010   tolf=  0.000050
A=  10.000   alpha=   0.602    c=   0.005   a=  50.000
Final Values
Total time, E=    23.948       cost=    64.552      q=     0.846   Function evaluations=     5000   Blocks=     894
x=    99.555     0.967     0.606     4.199    42.004   330.610     1.567     5.916

Run with Costfactor =      2.350   cub=    65.955   qlb=     0.846
Initial Values
x0=   100.000     0.900     0.250     3.000    53.936   333.888     4.628     3.000
f=   179.673   E=    16.332   cost=    67.928      q=     0.781   tolx=  0.000010   tolf=  0.000050
A=  10.000   alpha=   0.602    c=   0.005   a=  50.000
Final Values
Total time, E=    17.068       cost=    65.954      q=     0.846   Function evaluations=     5000   Blocks=     916
x=    99.874     0.975     0.577     3.058    53.951   333.661     5.078     3.008

Run with Costfactor =      2.400   cub=    67.358   qlb=     0.846
Initial Values
x0=   100.000     0.980     0.300     3.000    54.686   500.953     4.556     3.600
f=    72.965   E=    17.134   cost=    68.179      q=     0.861   tolx=  0.000010   tolf=  0.000050
A=  10.000   alpha=   0.602    c=   0.005   a=  50.000
Final Values
Total time, E=    17.395       cost=    67.358      q=     0.846   Function evaluations=     5000   Blocks=     712
x=    99.979     0.963     0.265     3.009    54.520   501.143     4.722     3.806

Run with Costfactor =      2.450   cub=    68.762   qlb=     0.846
Initial Values
x0=    78.744     0.900     0.300     3.600    55.612   500.878     4.484     3.600
f=   607.749   E=    19.961   cost=    58.122      q=     0.790   tolx=  0.000010   tolf=  0.000050
A=  10.000   alpha=   0.602    c=   0.005   a=  50.000
Final Values
Total time, E=    17.390       cost=    68.761      q=     0.846   Function evaluations=     5000   Blocks=     952
x=    82.212     0.960     0.255     3.071    48.147   509.330     1.764     3.075

Run with Costfactor =      2.500   cub=    70.165   qlb=     0.846
Initial Values
x0=    80.000     0.980     0.300     3.000    76.810   329.762     3.794     3.000
f=   423.314   E=    17.471   cost=    62.222      q=     0.837   tolx=  0.000010   tolf=  0.000050
A=  10.000   alpha=   0.602    c=   0.005   a=  50.000
Final Values
Total time, E=    16.516       cost=    70.166      q=     0.846   Function evaluations=     5000   Blocks=     929
x=    95.704     0.963     0.256     3.051    42.267   382.521     2.287     3.058

cost factor   cub          E       cost        qfinal      fun. evals  #blocks
     1.20    33.679    52.545    33.695    0.737      5000       978
     1.25    35.082    40.251    35.109    0.844      5000       980
     1.30    36.486    39.047    36.483    0.846      5000       917
     1.35    37.889    32.061    38.170    0.826      5000       990
     1.40    39.292    37.628    39.292    0.846      5000       676
     1.45    40.696    30.482    40.697    0.846      5000       603
     1.50    42.099    27.900    42.105    0.846      5000       569
     1.55    43.502    32.265    43.414    0.851      5000       983
     1.60    44.906    36.752    44.906    0.846      5000       884
     1.65    46.309    25.668    46.311    0.846      5000       699
     1.70    47.712    25.618    47.712    0.846      5000       822
     1.75    49.115    25.512    49.116    0.846      5000       673
     1.80    50.519    22.765    50.518    0.846      5000       885
     1.85    51.922    28.971    51.925    0.846      5000       973
     1.90    53.325    22.177    53.322    0.846      5000       735
     1.95    54.729    20.777    54.728    0.846      5000       903
     2.00    56.132    20.968    56.134    0.846      5000       936
     2.05    57.535    18.779    57.534    0.846      5000       958
     2.10    58.939    22.231    58.939    0.846      5000       943
     2.15    60.342    18.912    60.346    0.846      5000       974
     2.20    61.745    20.120    61.746    0.846      5000       812
     2.25    63.148    18.415    63.150    0.846      5000       777
     2.30    64.552    23.948    64.552    0.846      5000       894
```

2.35	65.955	17.068	65.954	0.846	5000	916
2.40	67.358	17.395	67.358	0.846	5000	712
2.45	68.762	17.390	68.761	0.846	5000	952
2.50	70.165	16.516	70.166	0.846	5000	929









APPENDIX F

MCM SYSTEM OF SYSTEMS MATLAB® SIMULATION CODE AND RESULTS

```
function
[E,cost,q,f,g]=mcmsim(x,m0,lambda,sminefield,pd,lambdaft,dmine,vtransit,cub,qlb,k)

p11=x(1);p12=x(2);p13=x(3);p14=x(4);p15=x(5);p21=x(6);p22=x(7);p23=x(8);
Tdetect=Block1(m0,lambda,p11);
Dfa=Block2(sminefield,p13);
Dft=Block3(pd,lambdaft,sminefield);
Dm=Block4(pd,m0);
[Tclass,Cm,Cf]=Block578(Dm,Dfa,Dft,p12,p14,dmine,vtransit);
E1=Block6(Tdetect,Tclass);
PL=Block9(p15,p21);
[E2,Nreacq,Nmiss]=Block10(PL,Cm,Cf,p23,p22);
E=Block12(E1,E2);
q=Block11(Nreacq,m0);
%
A1=3;
A2=100;
       px1=[4.503408803940725e-005,   -5.386095666335044e-003,    2.159330101073730e-001,
       1.334245377520354e+000];
px2=[2.834645669291690e+002,   -5.076377952756583e+002,    2.274598425197177e+002];
       px3=[-2.048380952380911e+000,    9.987333333333214e+000,   -
       1.794233333333325e+001,  2.032238095238094e+001];
px4=[1.159691730856429e-001,   -2.175732453433467e+000,    1.520381256718985e+001];
px5=[2.061825086032983e-004,   -3.775958229500408e-002,    1.777803488786043e+000];
       px6=[1.504875482450802e-007,   -1.578229837871938e-004,    5.516694369186904e-002,
       -1.813253427503106e+000];
       px7=[ -2.850358103957624e-001,    3.846213159671302e+000,   -
       1.726423877731832e+001,    3.334408692656030e+001];
px8=[2.102445277065673e-001,   -4.109593768487483e+000,    2.539723920331297e+001];
cost=polyval(px1,x(1))+polyval(px2,x(2))+polyval(px3,x(3))+polyval(px4,x(4))+polyval(px5,x
       (5))+polyv     al(px6,x(6))+polyval(px7,x(7))+polyval(px8,x(8));
% evaluate negative of quality constraint
g1=cost-cub;
g2 = -q+qlb;
%E,g1,g2,f
f=E+A1*abs(g1)+A2*abs(g2);
%f=f1+f2+A1*g1*g1+A2*g2*g2;
g=[];
```

```
function Tdetect=Block1(m0,lambda,p11)
Tdetect=24*m0/(lambda*p11);
```

```
function Dfa=Block2(sminefield,p13)
area=round(sminefield);
Dfa=success(p13,area);
```

```
function Dft=Block3(pd,lambdaft,sminefield)
F0=round(lambdaft*sminefield);
Dft=success(pd,F0);
```

```
function Dm=Block4(pd,m0)
Dm=success(pd,m0);
```

```
function [Tclass,Cm,Cf]=Block578(Dm,Dfa,Dft,p12,p14,dmine,vtransit)
% Number of classifications of detected mines
Cm=success(p12,Dm);
% Number of incorrect classifications of detected mines
Cf=Dm-Cm;
% Total time to classify
Tc=p14;
Tcf=2*Tc-60*dmine/(vtransit*2000);
Tclass=(1/60)*(Tc*Cm+Cf*Tcf+Tcf*(Dfa+Dft));
```

```
function E1=Block6(Tdetect,Tclass)
E1=Tdetect+Tclass;
```

```
function PL=Block9(p15,p21)
PL=exp(-p15/(4.481*p21));
```

```
function [E2,Nreacq,Nmiss]=Block10(PL,Cm,Cf,p23,p22)
% # of re-acquired MLOs
Nreacq=success(PL,Cm);
% # of not re-acquired MLOs
```

```
Nmiss=Cm-Nreacq;
E2=(1/60)*(Nreacq*p23+Nmiss*p22+Cf*p22);
```

```
function q=Block11(Nreacq,m0)
q=Nreacq/m0;
```

```
function E=Block12(E1,E2)
E=E1+E2;
```

2SPSA Simulation Optimization Results with Perfect Initial Conditions:

output from execution of MCM2spsaAvgsim.m with blocktol=5, averages last 3 iterates, gains A1=50,A2=1000 an initial x=optimum
09/21/97 11:37 AM   2SPSAsim 1000 x0=optimal.doc

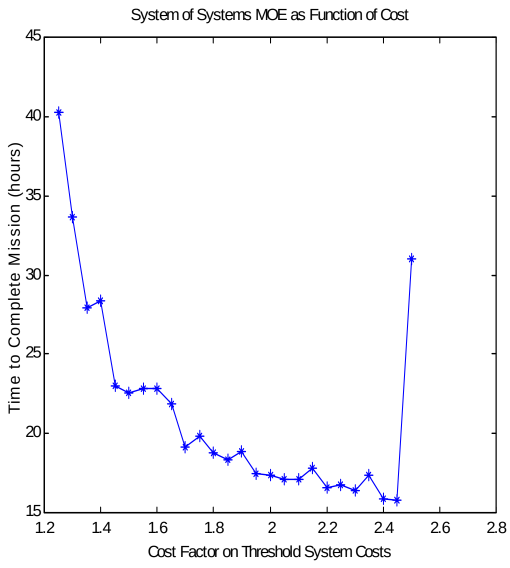
```
vlb=
   10.000     0.900     0.250     3.000    42.000    75.000     1.000     3.000
vub=
  100.000     0.980     2.000     9.170    90.000   700.000     7.000    10.000


Run with Costfactor =      1.250   cub=    35.082   qlb=     0.846
Initial Values
x0=    57.379     0.963     2.000     4.983    45.108   417.391     7.000     8.793
f=    68.400   E=    34.290   cost=    35.085      q=     0.880A=  50.000   alpha=   0.602    c=   0.020    a=  10.000
Final Values
Total time, E=    38.766       cost=    35.076      q=     0.810   Function evaluations=     5000   Blocks=     885
x=    56.037     0.941     1.352     6.954    44.677   416.403     6.750     9.367
Average total time, Eavg=    40.212   Esigma=     0.969   Average q=     0.827   qsigma=     0.042


Run with Costfactor =      1.300   cub=    36.486   qlb=     0.846
Initial Values
x0=    58.915     0.963     2.000     3.915    44.820   417.374     7.000     8.555
f=    65.324   E=    28.959   cost=    36.493      q=     0.810A=  50.000   alpha=   0.602    c=   0.020    a=  10.000
Final Values
Total time, E=    34.542       cost=    36.406      q=     0.920   Function evaluations=     5000   Blocks=     862
x=    58.836     0.967     1.988     5.253    45.256   418.608     6.759     8.459
Average total time, Eavg=    33.629   Esigma=     0.706   Average q=     0.844   qsigma=     0.036


Run with Costfactor =      1.350   cub=    37.889   qlb=     0.846
Initial Values
x0=    60.058     0.963     2.000     3.017    44.418   417.371     7.000     8.355
f=    31.878   E=    27.108   cost=    37.904      q=     0.850A=  50.000   alpha=   0.602    c=   0.020    a=  10.000
Final Values
Total time, E=    28.537       cost=    37.860      q=     0.830   Function evaluations=     5000   Blocks=     923
x=    59.196     0.954     1.683     3.307    44.472   417.321     6.984     8.331
Average total time, Eavg=    27.964   Esigma=     0.578   Average q=     0.833   qsigma=     0.034


Run with Costfactor =      1.400   cub=    39.292   qlb=     0.846
Initial Values
x0=    65.201     0.963     2.000     3.000    44.632   417.375     7.000     7.162
f=    80.883   E=    24.304   cost=    39.304      q=     0.790A=  50.000   alpha=   0.602    c=   0.020    a=  10.000
Final Values
Total time, E=    28.144       cost=    39.406      q=     0.750   Function evaluations=     5000   Blocks=     938
x=    63.202     0.901     1.899     4.643    42.301   417.354     6.584     6.025
Average total time, Eavg=    28.355   Esigma=     0.579   Average q=     0.792   qsigma=     0.040


Run with Costfactor =      1.450   cub=    40.696   qlb=     0.846
Initial Values
x0=    67.996     0.963     2.000     3.000    44.802   417.387     7.000     6.282
```

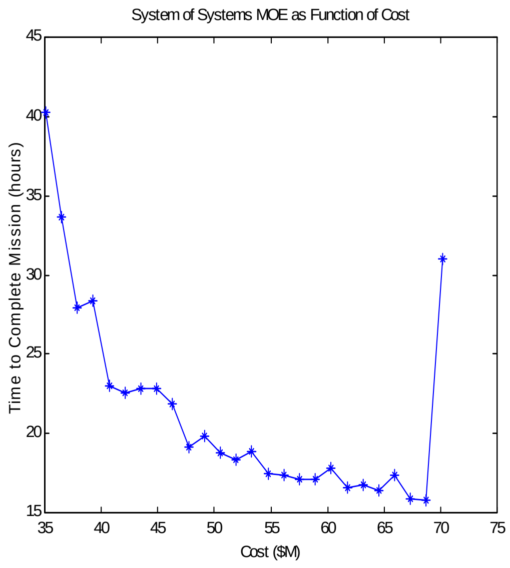
```
f=   49.092   E=    22.714   cost=    40.703     q=     0.820A=  50.000   alpha=  0.602    c=   0.020    a=  10.000
Final Values
Total time, E=    23.039       cost=    40.715      q=     0.830   Function evaluations=     5000   Blocks=     907
x=    69.049     0.920     1.984     3.346    46.114   417.417     6.986     5.420
Average total time, Eavg=    23.023   Esigma=     0.434   Average q=     0.816   qsigma=     0.037


Run with Costfactor =      1.500   cub=    42.099   qlb=     0.846
Initial Values
x0=    68.194     0.962     1.342     3.000    43.341   417.304     7.000     6.213
f=   27.022   E=    22.943   cost=    42.097     q=     0.850A=  50.000   alpha=  0.602    c=   0.020    a=  10.000
Final Values
Total time, E=    22.659       cost=    42.113      q=     0.870   Function evaluations=     5000   Blocks=     934
x=    70.984     0.979     1.998     3.001    42.885   418.882     7.000     6.080
Average total time, Eavg=    22.591   Esigma=     0.440   Average q=     0.858   qsigma=     0.037


Run with Costfactor =      1.550   cub=    43.502   qlb=     0.846
Initial Values
x0=    69.825     0.962     1.249     3.000    42.959   417.274     7.000     5.603
f=   26.120   E=    21.874   cost=    43.507     q=     0.850A=  50.000   alpha=  0.602    c=   0.020    a=  10.000
Final Values
Total time, E=    23.389       cost=    43.552      q=     0.900   Function evaluations=     5000   Blocks=     886
x=    71.513     0.977     1.948     3.058    42.124   416.226     6.677     6.115
Average total time, Eavg=    22.801   Esigma=     0.437   Average q=     0.861   qsigma=     0.031


Run with Costfactor =      1.600   cub=    44.906   qlb=     0.846
Initial Values
x0=    71.172     0.962     1.181     3.000    42.596   417.237     7.000     5.045
f=   36.120   E=    21.518   cost=    44.918     q=     0.860A=  50.000   alpha=  0.602    c=   0.020    a=  10.000
Final Values
Total time, E=    22.408       cost=    44.876      q=     0.840   Function evaluations=     5000   Blocks=     909
x=    72.903     0.979     1.594     3.390    42.230   418.895     6.669     5.651
Average total time, Eavg=    22.869   Esigma=     0.493   Average q=     0.859   qsigma=     0.034


Run with Costfactor =      1.650   cub=    46.309   qlb=     0.846
Initial Values
x0=    72.316     0.961     1.126     3.000    42.245   417.195     7.000     4.532
f=   25.770   E=    20.841   cost=    46.290     q=     0.850A=  50.000   alpha=  0.602    c=   0.020    a=  10.000
Final Values
Total time, E=    22.202       cost=    46.300      q=     0.740   Function evaluations=     5000   Blocks=     856
x=    72.953     0.930     0.936     3.317    42.428   417.224     6.682     4.992
Average total time, Eavg=    21.856   Esigma=     0.461   Average q=     0.823   qsigma=     0.043


Run with Costfactor =      1.700   cub=    47.712   qlb=     0.846
Initial Values
x0=    73.310     0.961     1.080     3.000    42.000   417.254     7.000     4.055
f=   36.061   E=    19.319   cost=    47.697     q=     0.830A=  50.000   alpha=  0.602    c=   0.020    a=  10.000
Final Values
Total time, E=    19.120       cost=    47.617      q=     0.840   Function evaluations=     5000   Blocks=     910
x=    73.339     0.950     1.841     3.043    42.795   418.061     6.831     3.585
Average total time, Eavg=    19.105   Esigma=     0.367   Average q=     0.837   qsigma=     0.037


Run with Costfactor =      1.750   cub=    49.115   qlb=     0.846
Initial Values
x0=    74.189     0.961     1.040     3.000    42.000   417.519     7.000     3.607
f=   44.120   E=    19.431   cost=    49.102     q=     0.870A=  50.000   alpha=  0.602    c=   0.020    a=  10.000
Final Values
Total time, E=    19.982       cost=    49.161      q=     0.860   Function evaluations=     5000   Blocks=     856
x=    74.626     0.977     1.210     3.263    42.528   417.385     6.799     3.884
Average total time, Eavg=    19.851   Esigma=     0.423   Average q=     0.855   qsigma=     0.034
```

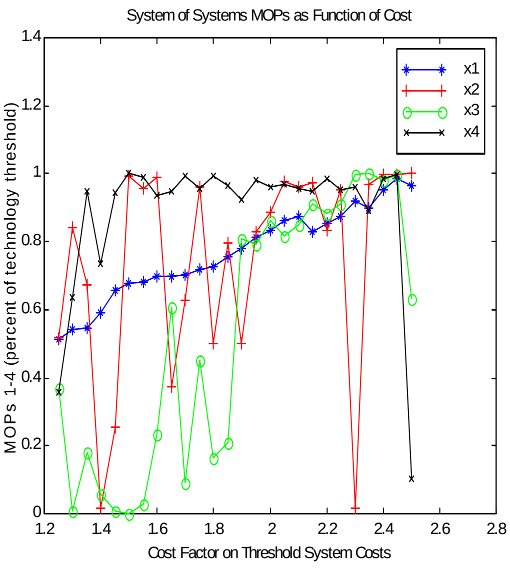
```
Run with Costfactor =     1.800   cub=    50.519   qlb=     0.846
Initial Values
x0=    74.979     0.961     1.004     3.000    42.000   417.757     7.000     3.184
f=   43.092   E=    18.569   cost=    50.508      q=     0.870A=  50.000   alpha=  0.602    c=   0.020    a=  10.000
Final Values
Total time, E=    18.429       cost=    50.667      q=     0.820   Function evaluations=     5000   Blocks=     893
x=    75.439     0.940     1.715     3.049    42.450   417.358     6.103     3.453
Average total time, Eavg=    18.746   Esigma=     0.376   Average q=     0.832   qsigma=     0.037


Run with Costfactor =     1.850   cub=    51.922   qlb=     0.846
Initial Values
x0=    77.328     0.961     0.901     3.000    42.000   418.623     7.000     3.000
f=   43.706   E=    17.160   cost=    51.911      q=     0.820A=  50.000   alpha=  0.602    c=   0.020    a=  10.000
Final Values
Total time, E=    18.529       cost=    51.909      q=     0.890   Function evaluations=     5000   Blocks=     888
x=    77.849     0.964     1.635     3.208    42.881   418.669     6.479     3.029
Average total time, Eavg=    18.332   Esigma=     0.343   Average q=     0.852   qsigma=     0.033


Run with Costfactor =     1.900   cub=    53.325   qlb=     0.846
Initial Values
x0=    80.440     0.961     0.765     3.000    42.000   420.090     7.000     3.000
f=   21.814   E=    17.393   cost=    53.317      q=     0.850A=  50.000   alpha=  0.602    c=   0.020    a=  10.000
Final Values
Total time, E=    18.850       cost=    53.308      q=     0.810   Function evaluations=     5000   Blocks=     904
x=    80.105     0.940     0.584     3.458    42.623   419.849     6.733     3.438
Average total time, Eavg=    18.893   Esigma=     0.410   Average q=     0.826   qsigma=     0.036


Run with Costfactor =     1.950   cub=    54.729   qlb=     0.846
Initial Values
x0=    83.033     0.961     0.651     3.000    42.000   421.627     7.000     3.000
f=   43.357   E=    17.078   cost=    54.723      q=     0.820A=  50.000   alpha=  0.602    c=   0.020    a=  10.000
Final Values
Total time, E=    17.601       cost=    54.759      q=     0.820   Function evaluations=     5000   Blocks=     893
x=    83.277     0.966     0.612     3.110    42.588   421.281     6.968     3.190
Average total time, Eavg=    17.482   Esigma=     0.360   Average q=     0.859   qsigma=     0.035


Run with Costfactor =     2.000   cub=    56.132   qlb=     0.846
Initial Values
x0=    85.261     0.961     0.552     3.000    42.000   423.201     7.000     3.000
f=   30.685   E=    16.451   cost=    56.127      q=     0.860A=  50.000   alpha=  0.602    c=   0.020    a=  10.000
Final Values
Total time, E=    17.278       cost=    56.211      q=     0.820   Function evaluations=     5000   Blocks=     908
x=    84.923     0.971     0.493     3.244    42.517   423.708     6.999     3.142
Average total time, Eavg=    17.389   Esigma=     0.403   Average q=     0.855   qsigma=     0.031


Run with Costfactor =     2.050   cub=    57.535   qlb=     0.846
Initial Values
x0=    87.217     0.961     0.463     3.000    42.000   424.790     7.000     3.000
f=   30.414   E=    16.284   cost=    57.538      q=     0.860A=  50.000   alpha=  0.602    c=   0.020    a=  10.000
Final Values
Total time, E=    16.536       cost=    57.356      q=     0.880   Function evaluations=     5000   Blocks=     918
x=    87.659     0.978     0.568     3.188    42.387   424.798     6.877     3.144
Average total time, Eavg=    17.143   Esigma=     0.398   Average q=     0.862   qsigma=     0.034


Run with Costfactor =     2.100   cub=    58.939   qlb=     0.846
Initial Values
x0=    88.965     0.961     0.383     3.000    42.000   426.399     7.000     3.000
f=   42.309   E=    16.183   cost=    58.941      q=     0.820A=  50.000   alpha=  0.602    c=   0.020    a=  10.000
Final Values
Total time, E=    16.663       cost=    59.030      q=     0.820   Function evaluations=     5000   Blocks=     941
x=    88.815     0.977     0.512     3.255    42.271   426.596     6.651     3.154
```

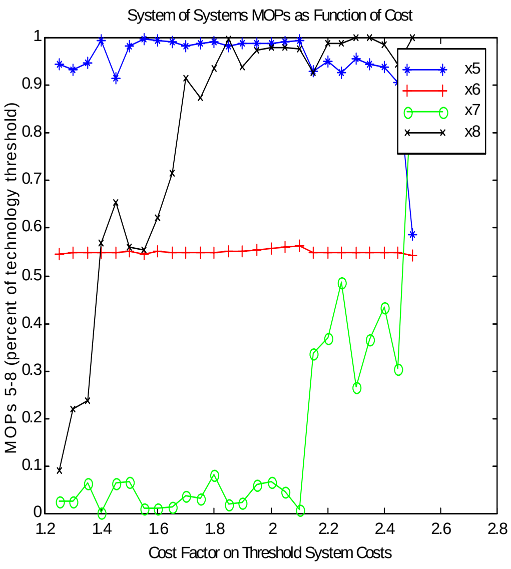
```
Average total time, Eavg=    17.132   Esigma=     0.405   Average q=     0.861   qsigma=     0.032


Run with Costfactor =      2.150   cub=    60.342   qlb=     0.846
Initial Values
x0=    85.153     0.963     0.570     3.000    45.839   417.389     3.948     3.000
f=    30.843   E=    16.304   cost=    60.331      q=     0.860A=  50.000   alpha=   0.602    c=   0.020   a=  10.000
Final Values
Total time, E=    17.923       cost=    60.532      q=     0.880   Function evaluations=     5000   Blocks=     870
x=    84.689     0.978     0.409     3.319    45.397   417.880     4.576     3.506
Average total time, Eavg=    17.816   Esigma=     0.403   Average q=     0.859   qsigma=     0.035


Run with Costfactor =      2.200   cub=    61.745   qlb=     0.846
Initial Values
x0=    87.149     0.963     0.481     3.000    45.694   417.395     3.921     3.000
f=    32.601   E=    16.065   cost=    61.734      q=     0.830A=  50.000   alpha=   0.602    c=   0.020   a=  10.000
Final Values
Total time, E=    16.071       cost=    61.688      q=     0.820   Function evaluations=     5000   Blocks=     922
x=    86.919     0.967     0.453     3.103    44.451   418.260     4.383     3.085
Average total time, Eavg=    16.563   Esigma=     0.376   Average q=     0.849   qsigma=     0.034


Run with Costfactor =      2.250   cub=    63.148   qlb=     0.846
Initial Values
x0=    88.941     0.963     0.400     3.000    45.508   417.412     3.894     3.000
f=    20.122   E=    16.036   cost=    63.147      q=     0.850A=  50.000   alpha=   0.602    c=   0.020   a=  10.000
Final Values
Total time, E=    17.174       cost=    62.948      q=     0.890   Function evaluations=     5000   Blocks=     924
x=    88.578     0.976     0.407     3.287    45.502   417.372     3.688     3.077
Average total time, Eavg=    16.772   Esigma=     0.362   Average q=     0.857   qsigma=     0.034


Run with Costfactor =      2.300   cub=    64.552   qlb=     0.846
Initial Values
x0=    90.539     0.963     0.326     3.000    45.239   417.380     3.876     3.000
f=    40.080   E=    15.853   cost=    64.547      q=     0.870A=  50.000   alpha=   0.602    c=   0.020   a=  10.000
Final Values
Total time, E=    15.952       cost=    64.590      q=     0.810   Function evaluations=     5000   Blocks=     940
x=    92.654     0.901     0.252     3.231    44.103   417.150     4.998     3.000
Average total time, Eavg=    16.444   Esigma=     0.367   Average q=     0.795   qsigma=     0.035


Run with Costfactor =      2.350   cub=    65.955   qlb=     0.846
Initial Values
x0=    92.010     0.963     0.257     3.000    44.947   417.360     3.857     3.000
f=    40.045   E=    15.895   cost=    65.958      q=     0.870A=  50.000   alpha=   0.602    c=   0.020   a=  10.000
Final Values
Total time, E=    17.042       cost=    65.224      q=     0.870   Function evaluations=     5000   Blocks=     938
x=    90.813     0.977     0.250     3.672    44.740   417.499     4.396     3.006
Average total time, Eavg=    17.359   Esigma=     0.397   Average q=     0.865   qsigma=     0.036


Run with Costfactor =      2.400   cub=    67.358   qlb=     0.846
Initial Values
x0=    95.302     0.963     0.250     3.000    45.572   417.461     3.797     3.000
f=    21.907   E=    15.559   cost=    67.351      q=     0.840A=  50.000   alpha=   0.602    c=   0.020   a=  10.000
Final Values
Total time, E=    15.782       cost=    67.321      q=     0.890   Function evaluations=     5000   Blocks=     932
x=    95.686     0.980     0.282     3.089    44.954   417.610     3.999     3.113
Average total time, Eavg=    15.838   Esigma=     0.287   Average q=     0.859   qsigma=     0.032


Run with Costfactor =      2.450   cub=    68.762   qlb=     0.846
Initial Values
x0=    98.430     0.964     0.250     3.000    46.343   417.398     3.737     3.000
f=    70.632   E=    15.828   cost=    68.778      q=     0.900A=  50.000   alpha=   0.602    c=   0.020   a=  10.000
```

```
Final Values
Total time, E=    16.469       cost=    68.433      q=     0.850   Function evaluations=     5000   Blocks=     944
x=    98.813     0.980     0.251     3.004    46.567   417.829     4.772     3.394
Average total time, Eavg=    15.802   Esigma=     0.388   Average q=     0.858   qsigma=     0.040


Run with Costfactor =      2.500   cub=    70.165   qlb=     0.846
Initial Values
x0=   100.000     0.973     0.250     3.000    64.008   412.202     3.162     3.000
f=    69.488   E=    15.144   cost=    70.158      q=     0.900A=  50.000   alpha=   0.602    c=   0.020   a=  10.000
Final Values
Total time, E=    30.983       cost=    70.135      q=     0.900   Function evaluations=     5000   Blocks=     925
x=    96.936     0.980     0.895     8.545    61.870   414.896     1.010     3.005
Average total time, Eavg=    31.005   Esigma=     0.815   Average q=     0.852   qsigma=     0.038


cost factor   cub       Avg E      cost     Avg qfinal      fun. evals  #blocks
     1.25    35.082    40.212    35.076     0.827      5000       885
     1.30    36.486    33.629    36.406     0.844      5000       862
     1.35    37.889    27.964    37.860     0.833      5000       923
     1.40    39.292    28.355    39.406     0.792      5000       938
     1.45    40.696    23.023    40.715     0.816      5000       907
     1.50    42.099    22.591    42.113     0.858      5000       934
     1.55    43.502    22.801    43.552     0.861      5000       886
     1.60    44.906    22.869    44.876     0.859      5000       909
     1.65    46.309    21.856    46.300     0.823      5000       856
     1.70    47.712    19.105    47.617     0.837      5000       910
     1.75    49.115    19.851    49.161     0.855      5000       856
     1.80    50.519    18.746    50.667     0.832      5000       893
     1.85    51.922    18.332    51.909     0.852      5000       888
     1.90    53.325    18.893    53.308     0.826      5000       904
     1.95    54.729    17.482    54.759     0.859      5000       893
     2.00    56.132    17.389    56.211     0.855      5000       908
     2.05    57.535    17.143    57.356     0.862      5000       918
     2.10    58.939    17.132    59.030     0.861      5000       941
     2.15    60.342    17.816    60.532     0.859      5000       870
     2.20    61.745    16.563    61.688     0.849      5000       922
     2.25    63.148    16.772    62.948     0.857      5000       924
     2.30    64.552    16.444    64.590     0.795      5000       940
     2.35    65.955    17.359    65.224     0.865      5000       938
     2.40    67.358    15.838    67.321     0.859      5000       932
     2.45    68.762    15.802    68.433     0.858      5000       944
     2.50    70.165    31.005    70.135     0.852      5000       925
```









2SPSA Simulation Composite 2000-Iteration Optimization Results:

COMPOSITE 2000 ITERATION RUN RESULTS:
File “composite Ramp 2000 output 10-8.doc”

output from execution of MCM2spsaAvgsim.m with blocktol=3/1, averages last 3 iterates, gains A1=50,A2=1000, ramp for x0 proportional to costfactor

```
vlb=
   10.000     0.900     0.250     3.000    42.000    75.000     1.000     3.000
vub=
  100.000     0.980     2.000     9.170    90.000   700.000     7.000    10.000


Run with Costfactor =      1.250   cub=    35.082   qlb=     0.846
Initial Values
```

```
x0=   25.000    0.913    1.708    8.142   82.000   179.167    6.000    8.833
f=   69.228   E=   53.049   cost=   34.223     q=    0.710A=  50.000   alpha=  0.602    c=  0.020   a=  10.000
Final Values
Total time, E=   38.113      cost=   36.307     q=    0.840   Function evaluations=   10000   Blocks=    1775
x=   29.280    0.962    1.916    3.900   82.457   177.871    6.805    8.020
Average total time, Eavg=   37.379   Esigma=    0.664   Average q=    0.780   qsigma=    0.047


Run with Costfactor =     1.300   cub=   36.486   qlb=    0.846
Initial Values
x0=   28.000    0.916    1.650    7.936   80.400   200.000    5.800    8.600
f=   67.004   E=   49.470   cost=   35.174     q=    0.710A=  50.000   alpha=  0.602    c=  0.020   a=  10.000
Final Values
Total time, E=   34.616      cost=   36.595     q=    0.810   Function evaluations=   10000   Blocks=    1700
x=   39.126    0.940    1.418    3.152   82.181   204.082    6.949    9.847
Average total time, Eavg=   33.591   Esigma=    0.691   Average q=    0.774   qsigma=    0.042

Run with Costfactor =     1.350   cub=   37.889   qlb=    0.846
Initial Values
x0=   31.000    0.919    1.592    7.730   78.800   220.833    5.600    8.367
f=   61.319   E=   48.010   cost=   35.986     q=    0.770A=  50.000   alpha=  0.602    c=  0.020   a=  10.000
Final Values
Total time, E=   31.892      cost=   37.862     q=    0.840   Function evaluations=   10000   Blocks=    1627
x=   50.156    0.980    1.786    3.381   89.427   226.955    6.835    9.838
Average total time, Eavg=   31.579   Esigma=    0.726   Average q=    0.806   qsigma=    0.041

Run with Costfactor =     1.400   cub=   39.292   qlb=    0.846
Initial Values
x0=   34.000    0.921    1.533    7.525   77.200   241.667    5.400    8.133
f=   58.045   E=   45.637   cost=   36.690     q=    0.800A=  50.000   alpha=  0.602    c=  0.020   a=  10.000
Final Values
Total time, E=   40.198      cost=   39.219     q=    0.810   Function evaluations=   10000   Blocks=    1680
x=   28.007    0.967    1.511    3.904   71.816   241.400    6.242    9.940
Average total time, Eavg=   40.951   Esigma=    0.712   Average q=    0.823   qsigma=    0.041

Run with Costfactor =     1.450   cub=   40.696   qlb=    0.846
Initial Values
x0=   37.000    0.924    1.475    7.319   75.600   262.500    5.200    7.900
f=   59.454   E=   42.716   cost=   37.317     q=    0.780A=  50.000   alpha=  0.602    c=  0.020   a=  10.000
Final Values
Total time, E=   35.251      cost=   40.624     q=    0.870   Function evaluations=   10000   Blocks=    1532
x=   43.661    0.968    1.517    5.078   80.725   261.692    3.505    7.672
Average total time, Eavg=   34.528   Esigma=    0.657   Average q=    0.814   qsigma=    0.038

Run with Costfactor =     1.500   cub=   42.099   qlb=    0.846
Initial Values
x0=   40.000    0.927    1.417    7.113   74.000   283.333    5.000    7.667
f=   59.358   E=   41.156   cost=   37.898     q=    0.790A=  50.000   alpha=  0.602    c=  0.020   a=  10.000
Final Values
Total time, E=   26.304      cost=   42.038     q=    0.890   Function evaluations=   10000   Blocks=    1461
x=   65.215    0.969    1.313    3.068   62.550   300.401    6.533    7.626
Average total time, Eavg=   25.557   Esigma=    0.554   Average q=    0.832   qsigma=    0.038


Run with Costfactor =     1.550   cub=   43.502   qlb=    0.846
Initial Values
x0=   43.000    0.929    1.358    6.908   72.400   304.167    4.800    7.433
f=   56.217   E=   39.510   cost=   38.467     q=    0.830A=  50.000   alpha=  0.602    c=  0.020   a=  10.000
Final Values
Total time, E=   28.408      cost=   43.655     q=    0.790   Function evaluations=   10000   Blocks=    1526
x=   38.211    0.971    1.950    3.010   74.526   314.854    3.284    6.915
Average total time, Eavg=   29.256   Esigma=    0.447   Average q=    0.834   qsigma=    0.036


Run with Costfactor =     1.600   cub=   44.906   qlb=    0.846
Initial Values
x0=   46.000    0.932    1.300    6.702   70.800   325.000    4.600    7.200
```

```
f=   57.677   E=    37.520   cost=    39.053      q=     0.820A=  50.000   alpha=   0.602    c=   0.020    a=  10.000
Final Values
Total time, E=    26.660       cost=    44.286      q=     0.810   Function evaluations=    10000    Blocks=    1585
x=    42.734     0.970     0.756     3.104    71.039   330.399     6.814     6.011
Average total time, Eavg=    27.056   Esigma=     0.502   Average q=     0.835   qsigma=     0.038

Run with Costfactor =      1.650   cub=    46.309   qlb=     0.846
Initial Values
x0=    49.000     0.935     1.242     6.496    69.200   345.833     4.400     6.967
f=   57.509   E=    36.051   cost=    39.689      q=     0.830A=  50.000   alpha=   0.602    c=   0.020    a=  10.000
Final Values
Total time, E=    23.609       cost=    46.372      q=     0.790   Function evaluations=    10000    Blocks=    1399
x=    52.165     0.979     1.738     3.104    59.748   343.736     5.978     4.973
Average total time, Eavg=    23.739   Esigma=     0.421   Average q=     0.845   qsigma=     0.035

Run with Costfactor =      1.700   cub=    47.712   qlb=     0.846
Initial Values
x0=    52.000     0.937     1.183     6.291    67.600   366.667     4.200     6.733
f=   62.633   E=    35.316   cost=    40.407      q=     0.900A=  50.000   alpha=   0.602    c=   0.020    a=  10.000
Final Values
Total time, E=    22.254       cost=    48.101      q=     0.830   Function evaluations=    10000    Blocks=    1630
x=    54.244     0.970     0.658     3.195    71.728   366.155     6.904     4.200
Average total time, Eavg=    22.233   Esigma=     0.452   Average q=     0.834   qsigma=     0.036

Run with Costfactor =      1.750   cub=    49.115   qlb=     0.846
Initial Values
x0=    55.000     0.940     1.125     6.085    66.000   387.500     4.000     6.500
f=   58.240   E=    34.004   cost=    41.237      q=     0.840A=  50.000   alpha=   0.602    c=   0.020    a=  10.000
Final Values
Total time, E=    20.055       cost=    49.102      q=     0.810   Function evaluations=    10000    Blocks=    1560
x=    78.208     0.970     1.352     3.074    59.283   357.190     6.395     4.538
Average total time, Eavg=    20.004   Esigma=     0.351   Average q=     0.842   qsigma=     0.033

Run with Costfactor =      1.800   cub=    50.519   qlb=     0.846
Initial Values
x0=    58.000     0.943     1.067     5.879    64.400   408.333     3.800     6.267
f=   62.264   E=    31.743   cost=    42.212      q=     0.790A=  50.000   alpha=   0.602    c=   0.020    a=  10.000
Final Values
Total time, E=    19.954       cost=    50.334      q=     0.830   Function evaluations=    10000    Blocks=    1481
x=    57.487     0.980     1.998     3.222    62.682   410.679     6.129     3.001
Average total time, Eavg=    20.406   Esigma=     0.341   Average q=     0.847   qsigma=     0.036

Run with Costfactor =      1.850   cub=    51.922   qlb=     0.846
Initial Values
x0=    61.000     0.945     1.008     5.674    62.800   429.167     3.600     6.033
f=   63.178   E=    28.899   cost=    43.362      q=     0.760A=  50.000   alpha=   0.602    c=   0.020    a=  10.000
Final Values
Total time, E=    19.793       cost=    51.961      q=     0.870   Function evaluations=    10000    Blocks=    1503
x=    72.033     0.971     0.762     3.555    45.498   515.835     6.921     3.158
Average total time, Eavg=    19.584   Esigma=     0.406   Average q=     0.855   qsigma=     0.035


Run with Costfactor =      1.900   cub=    53.325   qlb=     0.846
Initial Values
x0=    64.000     0.948     0.950     5.468    61.200   450.000     3.400     5.800
f=   62.783   E=    29.370   cost=    44.721      q=     0.770A=  50.000   alpha=   0.602    c=   0.020    a=  10.000
Final Values
Total time, E=    18.940       cost=    53.336      q=     0.840   Function evaluations=    10000    Blocks=    1353
x=    71.521     0.974     1.449     3.219    71.836   454.461     3.184     3.293
Average total time, Eavg=    18.949   Esigma=     0.337   Average q=     0.843   qsigma=     0.034

Run with Costfactor =      1.950   cub=    54.729   qlb=     0.846
Initial Values
x0=    67.000     0.951     0.892     5.262    59.600   470.833     3.200     5.567
f=   54.525   E=    27.694   cost=    46.318      q=     0.830A=  50.000   alpha=   0.602    c=   0.020    a=  10.000
Final Values
Total time, E=    18.566       cost=    54.787      q=     0.860   Function evaluations=    10000    Blocks=    1533
```

```
x=   92.837    0.961    0.959    3.237   58.676   469.751    6.808    3.630
Average total time, Eavg=   18.194   Esigma=    0.418   Average q=    0.838   qsigma=    0.041

Run with Costfactor =     2.000   cub=   56.132   qlb=    0.846
Initial Values
x0=   70.000    0.953    0.833    5.057   58.000   491.667    3.000    5.333
f=   59.620   E=   25.184   cost=   48.187     q=    0.740A=  50.000   alpha=  0.602    c=   0.020   a=  10.000
Final Values
Total time, E=   20.360      cost=   55.028     q=    0.790   Function evaluations=   10000   Blocks=    1463
x=   61.926    0.976    1.539    3.416   65.406   491.500    2.033    3.632
Average total time, Eavg=   20.861   Esigma=    0.363   Average q=    0.847   qsigma=    0.035

Run with Costfactor =     2.050   cub=   57.535   qlb=    0.846
Initial Values
x0=   73.000    0.956    0.775    4.851   56.400   512.500    2.800    5.100
f=   47.599   E=   25.464   cost=   50.357     q=    0.840A=  50.000   alpha=  0.602    c=   0.020   a=  10.000
Final Values
Total time, E=   17.734      cost=   57.559     q=    0.860   Function evaluations=   10000   Blocks=    1502
x=   77.539    0.959    0.673    3.112   50.409   499.238    4.888    3.069
Average total time, Eavg=   17.561   Esigma=    0.347   Average q=    0.835   qsigma=    0.035

Run with Costfactor =     2.100   cub=   58.939   qlb=    0.846
Initial Values
x0=   76.000    0.959    0.717    4.645   54.800   533.333    2.600    4.867
f=   42.745   E=   23.915   cost=   52.862     q=    0.840A=  50.000   alpha=  0.602    c=   0.020   a=  10.000
Final Values
Total time, E=   18.328      cost=   58.691     q=    0.840   Function evaluations=   10000   Blocks=    1513
x=   72.180    0.948    1.804    3.104   49.428   532.036    1.678    3.296
Average total time, Eavg=   18.406   Esigma=    0.348   Average q=    0.837   qsigma=    0.039

Run with Costfactor =     2.150   cub=   60.342   qlb=    0.846
Initial Values
x0=   79.000    0.961    0.658    4.440   53.200   554.167    2.400    4.633
f=   36.838   E=   22.407   cost=   55.732     q=    0.840A=  50.000   alpha=  0.602    c=   0.020   a=  10.000
Final Values
Total time, E=   16.943      cost=   60.324     q=    0.790   Function evaluations=   10000   Blocks=    1635
x=   78.434    0.961    0.285    3.241   42.279   552.864    6.461    3.061
Average total time, Eavg=   17.419   Esigma=    0.351   Average q=    0.851   qsigma=    0.031

Run with Costfactor =     2.200   cub=   61.745   qlb=    0.846
Initial Values
x0=   82.000    0.964    0.600    4.234   51.600   575.000    2.200    4.400
f=   33.543   E=   21.904   cost=   58.999     q=    0.880A=  50.000   alpha=  0.602    c=   0.020   a=  10.000
Final Values
Total time, E=   17.365      cost=   61.710     q=    0.790   Function evaluations=   10000   Blocks=    1487
x=   79.608    0.976    1.067    3.209   50.661   575.874    2.091    3.030
Average total time, Eavg=   17.805   Esigma=    0.313   Average q=    0.863   qsigma=    0.033

Run with Costfactor =     2.250   cub=   63.148   qlb=    0.846
Initial Values
x0=   85.000    0.967    0.542    4.028   50.000   595.833    2.000    4.167
f=   25.047   E=   20.086   cost=   62.695     q=    0.810A=  50.000   alpha=  0.602    c=   0.020   a=  10.000
Final Values
Total time, E=   16.664      cost=   63.173     q=    0.800   Function evaluations=   10000   Blocks=    1559
x=   86.327    0.945    0.737    3.117   51.930   595.164    2.408    3.014
Average total time, Eavg=   16.730   Esigma=    0.331   Average q=    0.833   qsigma=    0.040

Run with Costfactor =     2.300   cub=   64.552   qlb=    0.846
Initial Values
x0=   88.000    0.969    0.483    3.823   48.400   616.667    1.800    3.933
f=   34.051   E=   19.754   cost=   66.851     q=    0.920A=  50.000   alpha=  0.602    c=   0.020   a=  10.000
Final Values
Total time, E=   16.615      cost=   64.612     q=    0.810   Function evaluations=   10000   Blocks=    1522
x=   94.376    0.971    1.942    3.173   54.769   618.622    2.197    3.064
Average total time, Eavg=   16.787   Esigma=    0.310   Average q=    0.858   qsigma=    0.034

Run with Costfactor =     2.350   cub=   65.955   qlb=    0.846
```

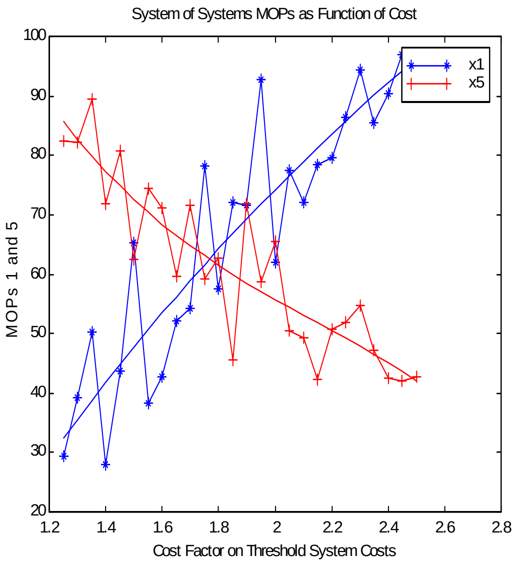
```
Initial Values
x0=    91.000     0.972     0.425     3.617    46.800   637.500     1.600     3.700
f=   35.316   E=    18.085   cost=    71.499      q=     0.840A=  50.000   alpha=   0.602    c=   0.020    a=  10.000
Final Values
Total time, E=    16.573       cost=    65.923      q=     0.890   Function evaluations=    10000   Blocks=    1528
x=    85.410     0.969     0.627     3.050    47.148   635.212     2.278     3.225
Average total time, Eavg=    16.786   Esigma=     0.347   Average q=     0.858   qsigma=     0.035

Run with Costfactor =      2.400   cub=    67.358   qlb=     0.846
Initial Values
x0=    94.000     0.975     0.367     3.411    45.200   658.333     1.400     3.467
f=   46.281   E=    16.746   cost=    76.670      q=     0.830A=  50.000   alpha=   0.602    c=   0.020    a=  10.000
Final Values
Total time, E=    17.330       cost=    67.238      q=     0.820   Function evaluations=    10000   Blocks=    1466
x=    90.360     0.955     0.393     3.290    42.463   656.124     5.081     3.013
Average total time, Eavg=    16.750   Esigma=     0.341   Average q=     0.849   qsigma=     0.035

Run with Costfactor =      2.450   cub=    68.762   qlb=     0.846
Initial Values
x0=    97.000     0.977     0.308     3.206    43.600   679.167     1.200     3.233
f=   59.280   E=    15.777   cost=    82.396      q=     0.820A=  50.000   alpha=   0.602    c=   0.020    a=  10.000
Final Values
Total time, E=    16.558       cost=    68.115      q=     0.890   Function evaluations=    10000   Blocks=    1519
x=    97.019     0.980     1.991     3.000    42.092   674.479     2.377     3.018
Average total time, Eavg=    16.166   Esigma=     0.326   Average q=     0.871   qsigma=     0.036

Run with Costfactor =      2.500   cub=    70.165   qlb=     0.846
Initial Values
x0=   100.000     0.980     0.250     3.000    42.000   700.000     1.000     3.000
f=   75.675   E=    14.443   cost=    88.709      q=     0.790A=  50.000   alpha=   0.602    c=   0.020    a=  10.000
Final Values
Total time, E=    16.610       cost=    70.097      q=     0.830   Function evaluations=    10000   Blocks=    1496
x=    97.207     0.973     0.500     3.341    42.846   699.810     6.281     3.000
Average total time, Eavg=    16.593   Esigma=     0.415   Average q=     0.864   qsigma=     0.030
```

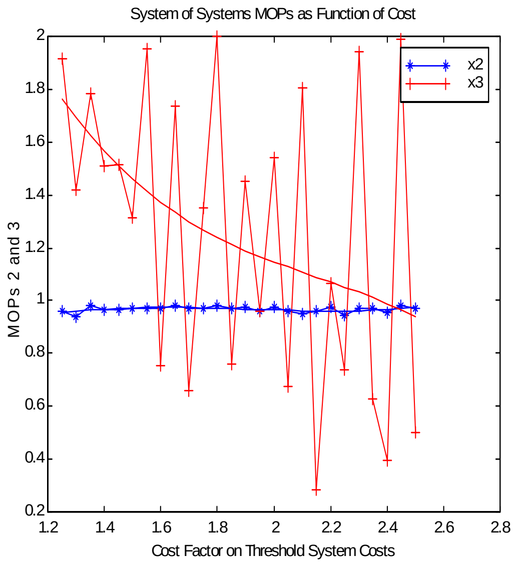
Note: Last 3 columns are from interpolating MOP values and feeding back into the simulation.

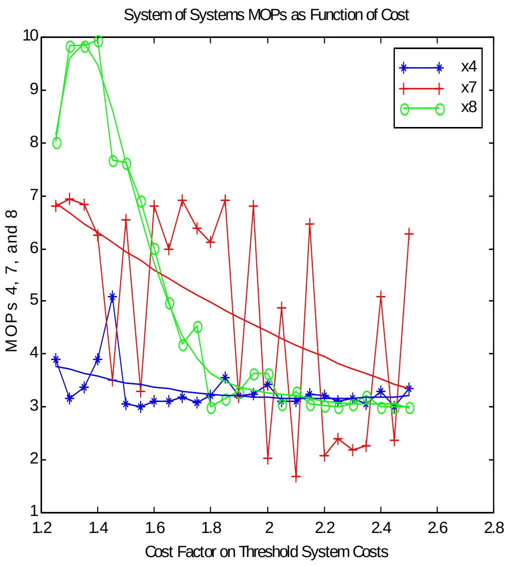
cost factor	cub	Avg. E	Cost	Avg. Qfinal	function evals.	# Blocks	Eavg (interp)	Cost (interp)	qavg (interp)	S1 c.f .(interp)	S2 c.f. (interp)
1.25	35.082	37.478	36.326	0.778	5000	1775	35.618	36.188	0.770	1.388	1.154
1.30	36.486	33.549	36.596	0.772	5000	1700	36.023	37.196	0.787	1.416	1.201
1.35	37.889	31.707	37.865	0.812	5000	1627	35.171	38.468	0.800	1.441	1.275
1.40	39.292	40.865	39.201	0.818	5000	1680	33.573	39.523	0.812	1.462	1.334
1.45	40.696	34.474	40.644	0.816	5000	1532	31.398	40.616	0.819	1.482	1.400
1.50	42.099	25.459	42.027	0.825	5000	1461	29.276	41.982	0.829	1.499	1.491
1.55	43.502	29.235	43.651	0.830	5000	1526	27.147	43.646	0.830	1.515	1.610
1.60	44.906	27.040	44.301	0.829	5000	1585	25.271	45.472	0.839	1.530	1.744
1.65	46.309	23.734	46.349	0.849	5000	1399	23.628	47.269	0.842	1.545	1.876
1.70	47.712	22.299	48.116	0.836	5000	1630	22.377	48.876	0.849	1.559	1.992
1.75	49.115	20.023	49.097	0.843	5000	1560	21.268	50.207	0.844	1.575	2.083
1.80	50.519	20.405	50.336	0.850	5000	1481	20.396	51.256	0.844	1.591	2.150
1.85	51.922	19.657	51.962	0.856	5000	1503	19.849	52.076	0.849	1.609	2.194
1.90	53.325	18.911	53.329	0.842	5000	1353	19.406	52.752	0.850	1.629	2.224
1.95	54.729	18.263	54.801	0.845	5000	1533	18.959	53.380	0.844	1.651	2.247
2.00	56.132	20.888	55.021	0.847	5000	1463	18.681	54.043	0.841	1.676	2.269
2.05	57.535	17.575	57.569	0.844	5000	1502	18.424	54.806	0.849	1.704	2.295
2.10	58.939	18.459	58.699	0.840	5000	1513	18.107	55.708	0.845	1.735	2.328
2.15	60.342	17.384	60.310	0.847	5000	1635	17.928	56.766	0.848	1.770	2.370
2.20	61.745	17.789	61.695	0.863	5000	1487	17.679	57.975	0.854	1.809	2.418
2.25	63.148	16.740	63.173	0.833	5000	1559	17.404	59.321	0.845	1.852	2.473
2.30	64.552	16.769	64.618	0.855	5000	1522	17.231	60.793	0.848	1.900	2.531
2.35	65.955	16.810	65.918	0.857	5000	1528	17.138	62.396	0.856	1.953	2.594
2.40	67.358	16.745	67.225	0.844	5000	1466	17.021	64.176	0.858	2.012	2.664
2.45	68.762	16.130	68.116	0.872	5000	1519	16.902	66.243	0.854	2.077	2.750
2.50	70.165	16.588	70.100	0.858	5000	1496	16.615	68.800	0.865	2.148	2.868

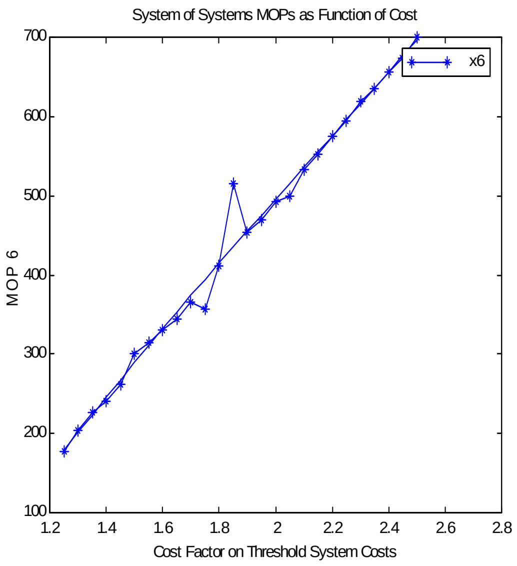
Interpolated MOP values, x1-x8

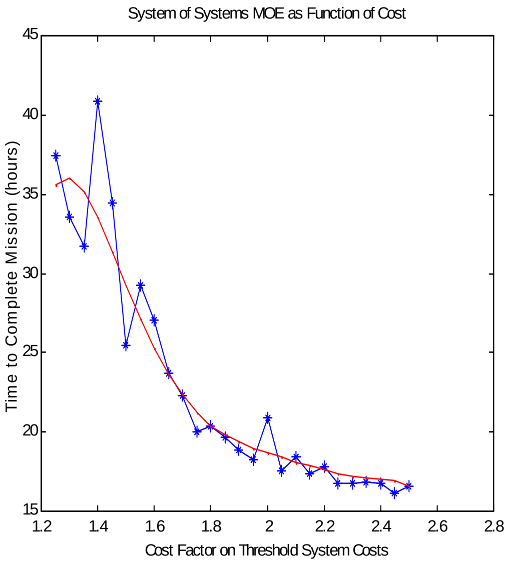
1.25	32.382	0.953	1.763	3.779	85.656	178.329	6.855	8.155
1.30	35.572	0.959	1.691	3.706	82.681	201.225	6.663	9.602
1.35	38.699	0.964	1.626	3.639	79.896	223.780	6.476	9.895
1.40	41.766	0.968	1.565	3.576	77.288	246.016	6.292	9.463
1.45	44.774	0.971	1.510	3.520	74.847	267.953	6.113	8.630
1.50	47.724	0.972	1.460	3.468	72.562	289.613	5.939	7.630
1.55	50.616	0.974	1.414	3.420	70.421	311.016	5.768	6.624
1.60	53.453	0.974	1.372	3.378	68.412	332.183	5.602	5.714
1.65	56.235	0.974	1.334	3.340	66.525	353.136	5.441	4.953
1.70	58.963	0.973	1.300	3.306	64.748	373.896	5.283	4.362
1.75	61.639	0.972	1.268	3.276	63.071	394.483	5.130	3.935
1.80	64.263	0.971	1.240	3.251	61.481	414.919	4.981	3.648
1.85	66.837	0.970	1.214	3.229	59.967	435.224	4.836	3.472
1.90	69.362	0.968	1.190	3.210	58.519	455.420	4.696	3.372
1.95	71.839	0.966	1.167	3.195	57.124	475.528	4.560	3.318
2.00	74.270	0.965	1.146	3.184	55.772	495.568	4.429	3.283
2.05	76.654	0.964	1.126	3.175	54.451	515.562	4.301	3.249
2.10	78.994	0.962	1.107	3.169	53.150	535.530	4.178	3.209
2.15	81.291	0.962	1.089	3.166	51.857	555.494	4.060	3.161
2.20	83.546	0.961	1.070	3.166	50.563	575.475	3.945	3.112
2.25	85.759	0.962	1.051	3.167	49.254	595.494	3.835	3.073
2.30	87.932	0.963	1.031	3.172	47.920	615.571	3.729	3.054
2.35	90.066	0.964	1.011	3.178	46.549	635.728	3.628	3.056
2.40	92.163	0.967	0.989	3.186	45.131	655.986	3.530	3.071
2.45	94.223	0.970	0.966	3.195	43.654	676.366	3.438	3.066
2.50	96.248	0.975	0.940	3.206	42.106	696.889	3.349	2.979

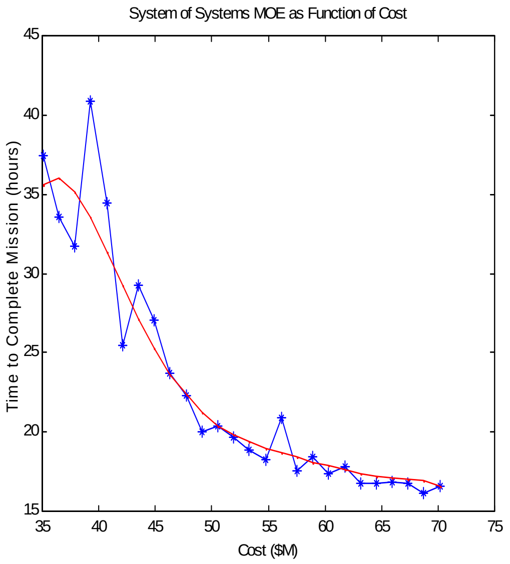


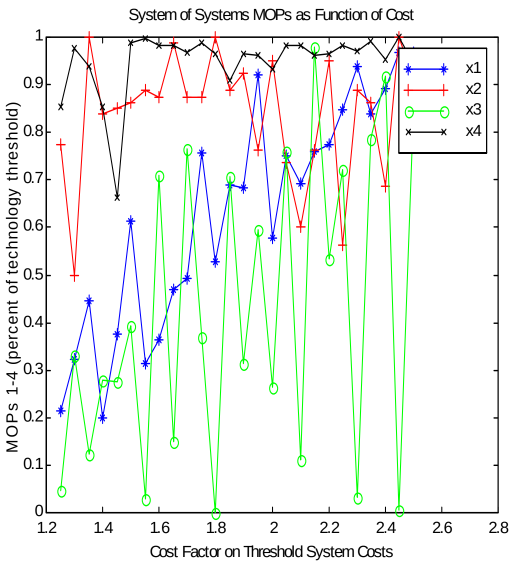



System of Systems MOPs as Function of Cost
MOPs 4, 7, and 8
Cost Factor on Threshold System Costs
x4
x7
x8
10
9
8
7
6
5
4
3
2
1
1.2
1.4
1.6
1.8
2
2.2
2.4
2.6
2.8

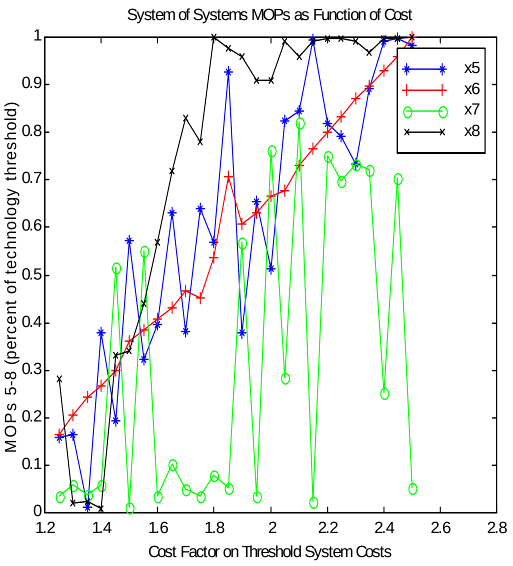

System of Systems MOPs as Function of Cost
MOP 6
Cost Factor on Threshold System Costs
x6
700
600
500
400
300
200
100
1.2
1.4
1.6
1.8
2
2.2
2.4
2.6
2.8


System of Systems MOE as Function of Cost
Time to Complete Mission (hours)
Cost Factor on Threshold System Costs
45
40
35
30
25
20
15
1.2
1.4
1.6
1.8
2
2.2
2.4
2.6
2.8


System of Systems MOE as Function of Cost
Time to Complete Mission (hours)
Cost ($M)
45
40
35
30
25
20
15
35
40
45
50
55
60
65
70
75

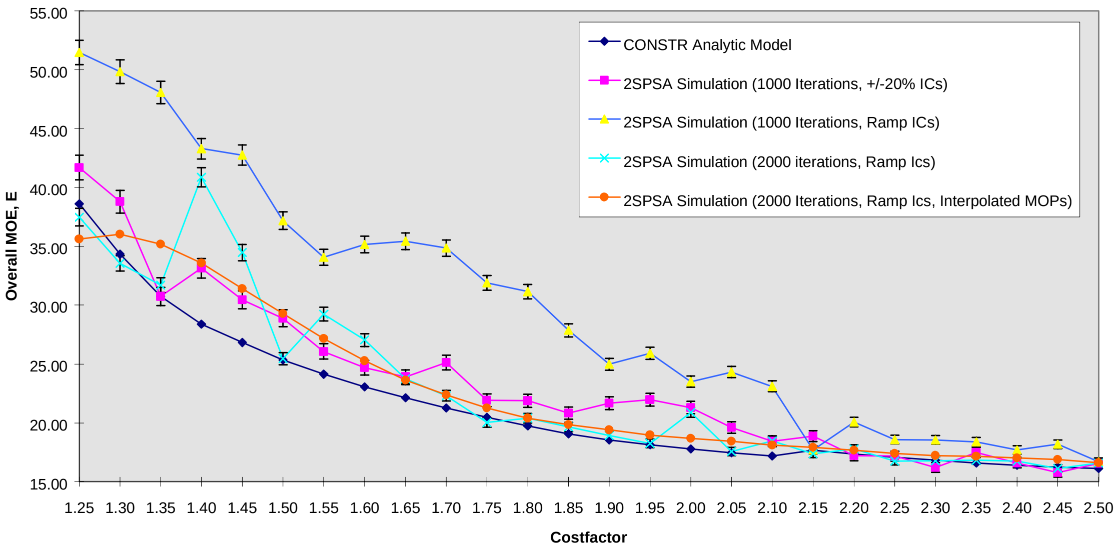


System of Systems MOPs as Function of Cost



System of Systems MOPs as Function of Cost

2SPSA Simulation vs. Analytic Results



Overall MOE, E
Costfactor
CONSTR Analytic Model
2SPSA Simulation (1000 Iterations, +/-20% ICs)
2SPSA Simulation (1000 Iterations, Ramp ICs)
2SPSA Simulation (2000 iterations, Ramp Ics)
2SPSA Simulation (2000 Iterations, Ramp Ics, Interpolated MOPs)