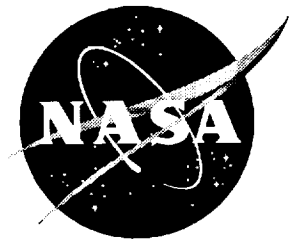
NASA Contractor Report 4542

Simulation Reduction Using the Taguchi Method

F. Mistree, U. Lautenschlager,
S. O. Erikstad, and J. K. Allen

CONTRACT NAG 9-616
October 1993

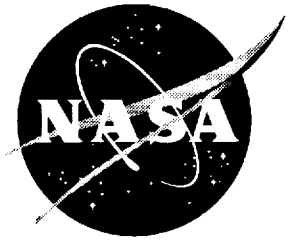


NASA Contractor Report 4542

Simulation Reduction Using the Taguchi Method

F. Mistree, U. Lautenschlager,
S. O. Erikstad, and J. K. Allen
University of Houston
Houston, Texas

Prepared for
Johnson Space Center
under Grant NAG 9-616



National Aeronautics
and Space Administration

Performance
Analysis Branch

1993

PREFACE

This is the final report from Team "A" concerning NASA Grant NAG 9-616 to the University of Houston. Our goal for this project was to develop a method to reduce the number of necessary simulations of real-life scenarios of rocket trajectories and to document the feasibility of this method in terms of confidence and reliability. This was to be accomplished by the use of the Taguchi techniques that form the core of Robust Design.

This project would not have come about without Ivan Johnson and Mike Tigges of the NASA Johnson Space Center. For this, we are extremely grateful.

Bert Bras, Ravi Reddy, and Janet Allen generously contributed their expertise and assistance. Thank you.

Dr. J. Sobieski of the NASA Langley Research Center offered many valuable suggestions for the improvement of this manuscript.

The work described in this report was supported in part by funds provided by NASA Grant NAG 9-616. The NSF equipment grant 8806811 is also gratefully acknowledged.

Farrokh Mistree
Systems Design Laboratory
Department of Mechanical Engineering
University of Houston
Houston, Texas 77204-4792

ABSTRACT

A large amount of engineering effort is consumed in conducting experiments to obtain information needed for making design decisions. Efficiency in generating such information is key to meeting market windows, keeping development and manufacturing costs low, and having high-quality products.

The principal focus of this project is to develop and implement applications of Taguchi's quality engineering techniques. In particular, we show how these techniques are applied for reducing the number of experiments for trajectory simulation of the LifeSat space vehicle. Orthogonal arrays are used to study many parameters simultaneously with a minimum of time and resources. Taguchi's signal-to-noise ratio is being employed to measure quality. A compromise Decision Support Problem and Robust Design are applied to demonstrate how quality is designed into a product in the early stages of designing.

CONTENTS

FIGURES

TABLES

CHAPTER 1

EFFICIENT EXPERIMENTATION FOR LIFESAT TRAJECTORY SIMULATION

The motivation for this study, namely NASA's request to substitute for Monte Carlo analysis a more efficient tool in order to reduce the number of simulations for design of space vehicle trajectories, is discussed in Section 1.1. Specifically, the trajectory of the LifeSat space vehicle is presented with a description of the problem requirements. Noise factors are the source of deviations from the desired landing target when the vehicle is deorbited. In Section 1.2, we introduce uncertainty in system performance due to these noise factors and show how their distributions are described. Statistical confidence about the magnitude of variation has to be established.

Our frame of reference combines the philosophy of Taguchi's quality engineering techniques and the compromise Decision Support Problem, DSP. The signal-to-noise ratio is a measure of quality. Orthogonal arrays are the basic tools for simulation reduction, while maintaining or improving quality. We further introduce a method for improvement of design quality when trade-offs between multiple objectives is required. This is shown in Section 1.3.

The main focus of the study and goals is discussed in Section 1.4. Also, a strategy for solution is provided in Section 1.5. The organization of this report is included in the last section, Section 1.6.

1.1 BACKGROUND AND MOTIVATION

A great deal of engineering effort is consumed in conducting experiments to obtain information needed to guide decisions related to a particular product. Efficiency in generating such information is the key to meeting market windows, keeping development and manufacturing costs low, and creating high-quality products.

The performance of complex engineering systems and the quality of products or processes generally depend on many factors. Taguchi separates these factors into two main groups: control factors and noise factors [Ross, 1988]. Control factors are those factors which are set by the manufacturer; noise factors are those factors over which the manufacturer has no direct control but which vary in the environment of the system or product [Phadke, 1989].

Frequently it is necessary to perform a statistical analysis on the response of a system to investigate the system performance due to the factors. The response of a system, or a function (linear or nonlinear) is typically quantified by the output and information about the numerical statistics.

In the following section, we introduce the simulation of the LifeSat space vehicle trajectory which is employed by NASA JSC [McCleary, 1991]. In this report we are concerned with problems involved in this simulation and suggest ways to overcome them.

1.1.1 Simulation of the Trajectory of the LifeSat Space Vehicle

NASA uses a program called *Simulation and Optimization of Rocket Trajectories* , SORT [McCleary, 1991] for the computation of trajectory aspects for the design of space vehicles. One important functional requirement of the design is its ability to follow a predetermined trajectory. For instance, it is necessary for the space vehicle to land within a desired target area. Naturally deviations from this trajectory will occur due to dispersions in vehicle and environmental parameters; e.g., initial state, atmospheric conditions, winds, vehicle and parachute aerodynamics, and vehicle weight. These deviations may make the achievement of the design goals uncertain. Simulation results provide an indication as to how the trajectory is influenced by changes in actual flight and operating conditions due to uncertainties.

The SORT program uses Monte Carlo simulations [Press et al., 1990] to obtain simulations of real-life scenarios of rocket trajectories. Different trajectory parameters—e.g., the initial state and various atmospheric and aerodynamic parameters—are varied (dispersed), and the simulation is carried out over the range of dispersed parameter values. Monte Carlo analysis capabilities implemented into SORT [McCleary, 1991] provide tools to disperse several state, environmental, and vehicle parameters. Furthermore, post-processing tools are available and statistical data concerning the individual factor contributions on the dispersions are generated.

One example, the case studied here in which the LifeSat space vehicle is deorbited, is given in a memo from McDonnell Douglas [McCleary, 1991]. We define the following nomenclature:

deorbit	- the process of leaving space and returning to Earth,
trim burn	- rocket burn to change position and velocity of vehicle,

entry interface	- altitude where vehicle enters atmosphere of Earth,
g-load	- a measure of acceleration in multiples of the acceleration of gravity,
peak	- maximum of a particular parameter,
Mach number	- ratio of velocity and speed of sound.

The LifeSat vehicle is a satellite which weighs approximately 1560 kg. In the satellite, experiments on small animals (mice) will be performed in space. In order to evaluate the results of the experiments, the animals must be brought back alive in order to evaluate the results of the experiments. There are several possibilities to deorbit the vehicle. These have being investigated and are explained in the problem description, as follows:

> "A deorbit burn is used in order to deorbit the LifeSat vehicle from space. After this burn it must be [decided] whether or not a correction of the vehicle's state vector is necessary. Three scenarios are employed to obtain the information if
>
> - no state vector correction is necessary after deorbit burn,
> - a navigational update and trim burn at 4000 km altitude is required, or
> - a trim burn at entry interface (EI) at 121 km altitude is necessary.
>
> A Monte Carlo analysis has been performed in order to determine whether a trim burn is needed and the necessary g-load to land within the desired target area. The nominal targeted landing position is the missile range in White Sands, New Mexico, which is at 33.6 degrees (deg) geodetic latitude and -106.65 deg longitude. The available impact area ranges from 33.4 deg to 33.8 deg geodetic latitude and [from] -106.7 to -106.55 deg longitude. For the three mentioned scenarios, five EI scenarios are analyzed. They are characterized by the nominal peak g-load which can range from 11 to 15. Each simulation is based on the initial state at entry interface. The Monte Carlo results of one analysis run are each based on 1109 dispersed cases in order to obtain the desired confidence of 95% and reliability of 99.73%."

In Figure 1.1, three different flight conditions of the vehicle during trajectory simulation are presented. These conditions are

- vehicle at EI state at 121 km altitude,
- vehicle after drogue chute deployment at Mach number M = 1.5, and
- vehicle after main parachute deployment at 3 km altitude.

After leaving the entry interface the major part of the flight is flown without parachutes. Because of aerodynamic drag forces the vehicle slows down and the drogue chutes are deployed at a height of approximately 20 km. At this point, the vehicle now has an approximately vertical trajectory (Sec. 3.3.2) and is very sensitive to winds. At 3 km altitude, the main chutes are deployed and the vehicle is supposed to land close to the desired target area.

Currently, the trajectory of the LifeSat vehicle is simulated with Monte Carlo simulation. In the following section, we briefly discuss the Monte Carlo simulation technique for trajectory evaluation and introduce alternate approaches to Monte Carlo simulation.

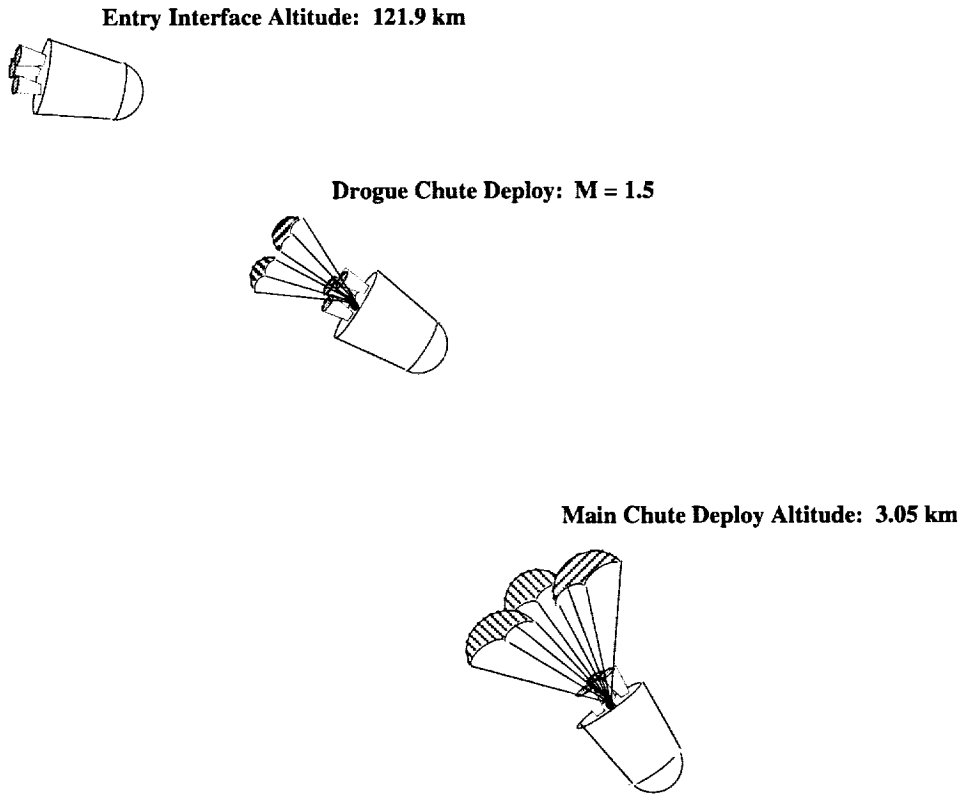


Figure 1.1 - Flight Conditions

1.1.2 Introduction to Monte Carlo Simulation and Alternate Approaches

In the Monte Carlo Simulation [Press et al., 1990], a random number generator is used to simulate a large number of combinations of noise factors and parameters within tolerances. The value of the response is computed for each testing condition and the mean and the variance of the response are then calculated.

For the optimization of rocket trajectories, NASA JSC employs Monte Carlo simulation in order to evaluate the system performance under uncertain conditions. To obtain accurate estimates of mean and variance of the system performance, Monte Carlo simulation requires evaluation of the response of the system under a large number of testing conditions (1109). Hence, only a limited number of scenarios can be examined completely. However, changes in the vehicle's proposed dimensions during design require repeated simulations. This can be very expensive, if

- the simulation requires a great deal of computational time, and
- if we also want to compare many combinations of control factors and different designs.

In order to reduce the large number of Monte Carlo simulations, studies have been undertaken to choose dispersed input factors systematically [Bogner, 1989]. Bogner presents an alternate approach to Monte Carlo simulation [Bogner, 1989]. In this approach different methods to scatter input parameters are considered in order to obtain more information about operating conditions further away from the nominal mean. Since with Monte Carlo simulation most output values are placed near the mean, the idea is to place more samples of the input parameters in regions with higher σ-levels (multiple of standard deviation σ) to obtain a better sampling of the output. Four different methods were examined for the systematic dispersion (scattering) of parameters. These methods are as follows:

- scatter systematically throughout volume of input space,
- scatter uniformly but randomly throughout volume,
- scatter uniformly but randomly in radius/likelihood of occurrence, and
- scatter uniformly in each σ-level for randomly generated points.

In the first and third cases, major problems occur for higher dimensional problems. The second case has a majority of outputs placed outside of the region of interest which is ($\mu \pm 3\sigma$). The fourth case is chosen because the scattering is uniform in σ-level regardless of dimension. Each output is weighted according to the probability of the input vector so that the calculated statistics are still valid.

Conclusions are drawn from a six-dimensional function which is assumed to be representative for lower and higher dimensional cases. For comparison with Monte Carlo technique, a sample size of 1000 is chosen. Another comparison is done for a problem concerning the Aeroassist Flight Experiment, a NASA spacecraft which is scheduled for launch in 1994. In this example, eight runs are made with 20,000 samples each. Although that study was developed to provide a better estimation of operating conditions further away from the mean, the savings of time and resources are low. Before we describe our approach to the problem we first introduce stochastic uncertainty and probability distributions. The effects of uncertainty or noise determine the deviation from the desired system performance.

1.2 AN INTRODUCTION TO STOCHASTIC UNCERTAINTY AND PROBABILITY DISTRIBUTIONS

Stochastic uncertainty in engineering systems occurs during the design process of these systems and during the system performance. It arises from a lack of knowledge of the exact value of a design or environmental parameter due to a process beyond the control of a designer. However, from past observations, the parameters may be known to lie in a range. In other words, a designer might have a statistically significant sample set which enables that designer to represent the parameter with a probability density distribution [Sokal and Rohlf, 1981]. In this study, we model uncertainty of parameters with *uniform* distributions and *Gaussian* or *normal* distributions. Parameters which have one of these distributions are also called dispersed parameters.

A parameter is *uniformly distributed* if any number in a specified range is just as likely as any other. In other words, a uniformly distributed parameter is a random number with

equal or uniform probability distribution. Engineering specifications are generally written as $\mu \pm \Delta_0$, where μ represents the mean value of a parameter and Δ_0 is the tolerance. The probability distribution of a uniformly distributed parameter is shown in Figure 1.2.

The probability distribution, also called probability density function p(x), of a uniformly distributed parameter x is defined as

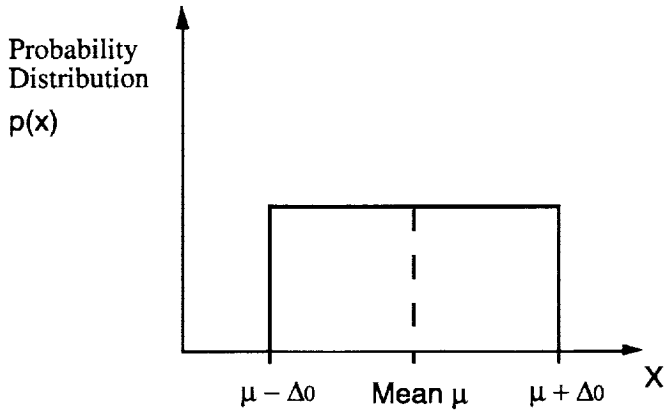


Figure 1.2 - Uniform Probability Distribution

$$p(x) = \begin{cases} \dfrac{1}{2\Delta_0} & \mu - \Delta_0 \le x \le \mu + \Delta_0 \\ 0 & \text{otherwise} \end{cases} , \qquad (1.1)$$

We assume that the total probability is one in a parameter range from $\mu - \Delta_0$ to $\mu + \Delta_0$.

A parameter is *normally distributed* (Gaussian distribution) if its normal probability density function can be represented by the expression

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}} , \qquad (1.2)$$

where μ is the mean and σ is the standard deviation. μ and σ determine the shape and location of the distribution. In Figure 1.3, we present three different probability density functions for normal distributions. The two left curves differ only in the standard deviation. The right curve has a smaller standard deviation than the others.

The value p(x) represents the probability density of x. The curve is symmetrical around the mean and the area below the curve is distributed as

$\mu \pm \sigma$ contains 68.26% of the area,
$\mu \pm 2\sigma$ contains 95.46% of the area, and
$\mu \pm 3\sigma$ contains 99.73% of the area.

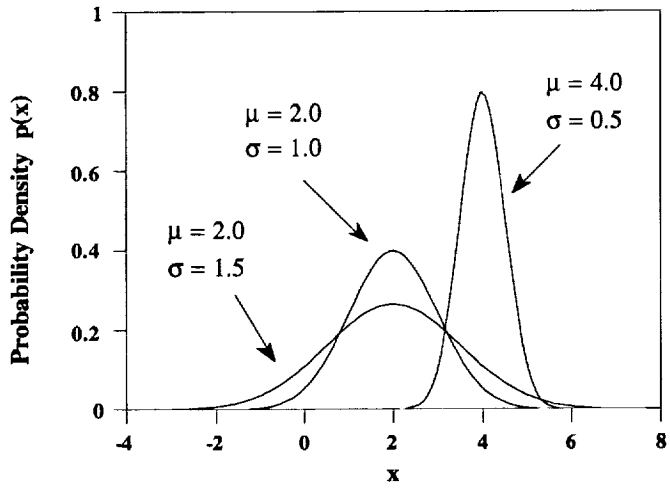


Figure 1.3 - Illustration of Normal Probability Density Function

Transformation of these values provides information about the σ value for a given percentage of area; e.g.,

50% of the area falls between $\mu \pm 0.674\ \sigma$,
95% of the area falls between $\mu \pm 1.960\ \sigma$, and
99% of the area falls between $\mu \pm 2.576\ \sigma$.

For calculation of these quantities see [Sokal and Rohlf, 1981]. It is common engineering practice that the range between $\mu \pm 3\sigma$ is considered to be the tolerance range for a manufacturing part or a parameter dimension.

We observe that uncertainty due to dispersed system and environmental parameters can be modeled by a uniform or normal distribution. To be able to make design decisions it is necessary to incorporate this uncertainty in the design model; for example, see [Zhou, 1992]. This improves our understanding of the state of a design by including the performance of the product within its environment. Taguchi's quality engineering and robust design offer a useful method to evaluate the performance of a system when uncertainty is present and to measure the quality of the design state. In the next section, we introduce this method and explain Taguchi's philosophy. The compromise DSP (**D**ecision **S**upport **P**roblem) is introduced as a model for decision support in design. We are

interested in modeling quality characteristics into a design model to provide a means for making better decisions.

1.3 OUR FRAME OF REFERENCE

1.3.1 The Taguchi Method and Robust Design

Products have characteristics that describe their performance relative to customer requirements or expectations [Ross, 1988]. Characteristics such as economy, weight of a component, or the durability of a switch concern the customer. The quality of a product/process is measured in terms of these characteristics. Typically, the quality is also measured throughout its life cycle. The *ideal* quality a customer can expect is that every product delivers the *target* performance each time the product is used under all intended operating conditions and throughout its intended life and that there will be no harmful side effects [Phadke, 1989]. Following Taguchi [Taguchi, 1987] we measure the quality of a product in terms of the total loss to society due to functional variation and harmful side effects. The ideal quality loss is zero.

The challenge for a designer to design high-quality products is obvious. Driven by the need to compete on price and performance, quality-conscious designers are increasingly aware of the need to improve products and processes [Roy, 1990]. Delivering a high-quality product at low cost is an interdisciplinary problem involving engineering, economics, statistics, and management [Phadke, 1989]. In the cost of a product, we must consider the operating cost, the manufacturing cost, and the cost of new product development. A high-quality product has low costs in all three categories. Robust design is a systematic method for keeping the producer's cost low while delivering a high-quality product and keeping the operating cost low. Taguchi espoused an excellent philosophy for quality control in the manufacturing industries [Roy, 1990]. His philosophy is founded on three very simple and fundamental concepts. These concepts are stated in Roy as follows [Roy, 1990]:

- Quality should be designed into the product and not inspected into it.
- Quality is best achieved by minimizing the deviation from the target. The product should be designed so that it is immune to uncontrollable environmental factors.
- The cost of quality should be measured as a function of deviation from the standard and the losses should be measured system-wide.

Quality concepts are based on the philosophy of prevention. The product design must be so robust that it is immune to the influence of uncontrolled environmental factors which are known as *noise factors* .[Roy, 1990]. We want a leading indicator of quality by which we can evaluate the effect of changing a particular design parameter on the product's performance. This indicator is called *signal-to-noise ratio.* It isolates the sensitivity of the system's performance to noise factors and converts a set of observations into a single number.

A product under investigation may exhibit a distribution which has a mean value that differs from the target value. The first step towards improving quality is to achieve a

distribution as close to the target value as possible. Efficient experimentation is required to find dependable information with minimum time and resources about the design parameters [Phadke, 1989]. Taguchi designs experiments using *orthogonal arrays* which make the design of experiments easy and consistent. The power of orthogonal arrays is their ability to evaluate several factors with a minimum number of experiments.

The signal-to-noise ratio is being employed to measure quality and orthogonal arrays to study many design parameters simultaneously with a minimum amount of time and resources. We have integrated these into the compromise DSP in order to support design decisions in the early stages of design. The compromise DSP is introduced in the following section.

1.3.2 The Compromise DSP

Compromise DSPs (**D**ecision **S**upport **P**roblem) are used to model engineering decisions involving multiple trade-offs [Mistree et al., 1992]. The compromise DSP is a hybrid multiobjective programming model [Bascaran et al., 1987; Karandikar and Mistree, 1991; Mistree et al., 1992]. It incorporates concepts from both traditional Mathematical Programming and Goal Programming (GP). The compromise DSP is a major component of the DSIDES (**D**ecision **S**upport **I**n the **D**esign of **E**ngineering **S**ystems) system [Mistree and Bras, 1992]. It is similar to GP in that the multiple objectives are formulated as system goals involving both system and deviation variables and the deviation function is solely a function of the goal deviation variables. This is in contrast to traditional Mathematical Programming where multiple objectives are modeled as a weighted function of the system variables only. The concept of system constraints is retained from the traditional constrained optimization formulation. However, unlike traditional Mathematical Programming and GP, the compromise DSP places special emphasis on the bounds of the system variables. In the compromise DSP, contrary to the GP formulation in which everything is converted into goals, constraints and bounds are handled separately from the system goals. In the compromise formulation, the set of system constraints and bounds defines the *feasible design space* and the sets of system goals define the *aspiration space*. For feasibility, the system constraints and bounds must be satisfied. A *satisficing* solution then is that feasible point which achieves the system goals as far as possible. The solution to this problem represents a trade-off between that which is desired (as modeled by the aspiration space) and that which can be achieved (as modeled by the design space). The compromise DSP is stated as follows:

GIVEN	An **alternative** that is to be improved through **modification**. **Assumptions** used to model the domain of interest. The **system parameters.** The **goals** for the design.
FIND	The values of the independent **system variables** (they describe the physical attributes of an artifact). The values of the **deviation variables** (they indicate the extent to which the goals are achieved).
SATISFY	The **system constraints** that must be satisfied for the solution to be feasible.

The **system goals** that must achieve a specified target value as far as possible.
The **lower and upper bounds** on the system variables.
Bounds on the deviation variables.

MINIMIZE The **deviation function** which is a measure of the deviation of the system performance from that implied by the set of goals and associated priority levels or relative weights.

A compromise DSP for the LifeSat model is developed in Chapter 5. The goals are based on the quality characteristics of meeting the target and having a robust design against noise factors.

1.4 FOCUS AND GOAL

Our principal focus in this study is to develop and implement applications of Taguchi's techniques for quality engineering and to establish its validity for systems which require a high level of reliability in performance. With this approach we want to reduce the number of simulations for LifeSat trajectory. The main difference in this study to standard applications of Taguchi's techniques is the different level on which we are approaching the LifeSat simulation problem. The focus is on noise factors and their contributions to performance variation for *one* particular design of the vehicle. A robust design study and quality improvement using the compromise DSP are on a higher level, where we are interested in the design of the vehicle in particular. The questions which are worthy of investigation are as follows:

- Is it possible to reduce a large number of Monte Carlo simulations by using Orthogonal Array experiments?
- What is the statistical confidence level to which the results from Monte Carlo simulations and orthogonal array based simulations are equivalent?
- How can we identify factor contributions to variation in system performance?
- Are we able to improve the quality of a design by using the signal-to-noise ratio?
- Is it possible to predict the factor levels for best system performance using robust design?
- What are the advantages and limitations of the approach?

Our principal goal is to find answers to these questions. In order to achieve this goal, our specific subgoals are as follows:

- to develop an understanding of Taguchi's quality technique and what is involved with it,
- to develop and implement a simulation model to study the effects of the technique in simulation, and
- to develop tools which support human decision making.

In Figure 1.4, we present a diagram of modules involved in the simulation of the LifeSat trajectory using orthogonal arrays. These modules can be classified as modules used to perform the simulation and post-processing tools.

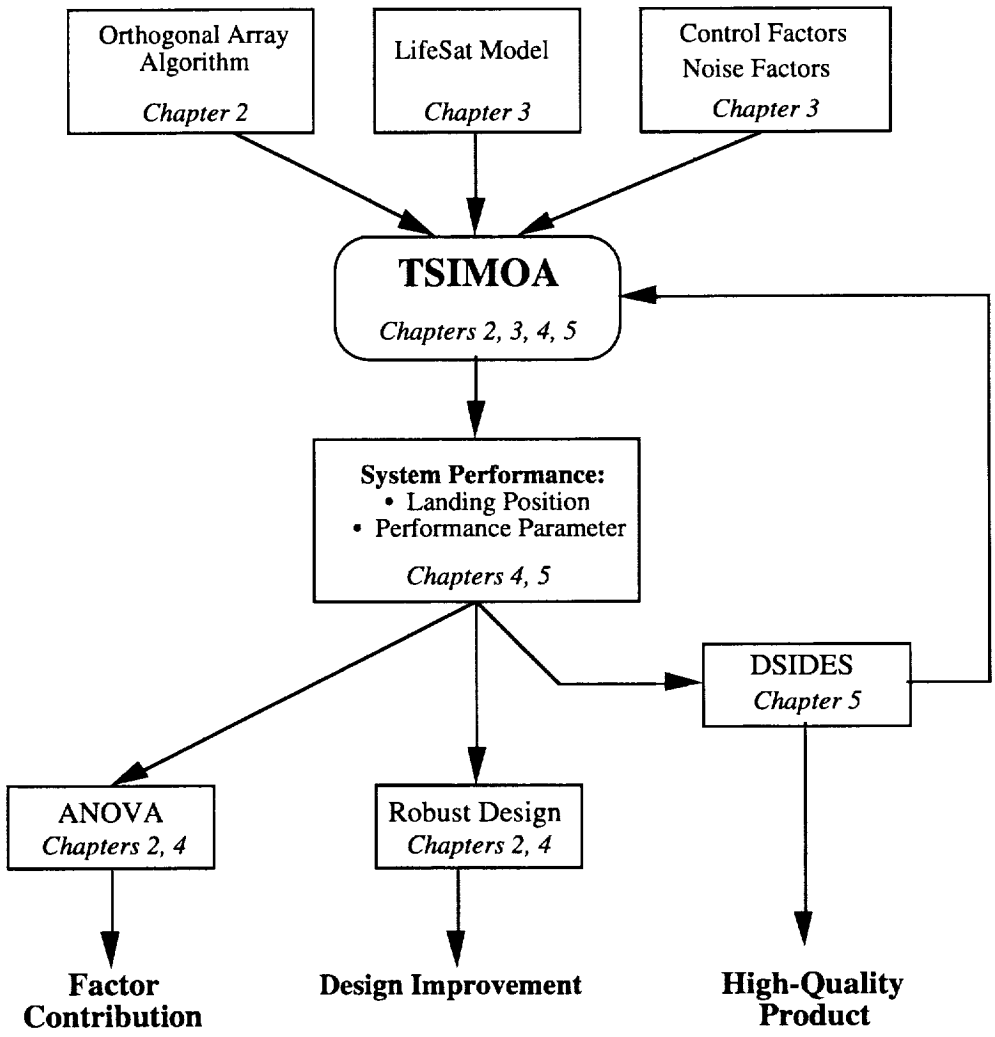


Figure 1.4 - Trajectory Simulation Modules

An algorithm to create orthogonal arrays is one module for the simulation program TSIMOA (**T**rajectory **SIM**ulation with **O**rthogonal **A**rrays). This is discussed in Chapter 2. In order to perform simulations with orthogonal arrays, we need information about factors and their dispersions which is provided in Chapter 3 along with the LifeSat model. The model is implemented in TSIMOA and the factor information is used as input data. The theory for quality engineering is explained in Chapter 2 and the LifeSat model is discussed in Chapter 3. This theory is applied to the LifeSat model as described in Chapters 4 and 5 to obtain and evaluate simulation results using TSIMOA. The simulation results are then post-processed by using analysis of variance (ANOVA), robust design, or a compromise DSP. The implementation of DSIDES provides a capable tool for design improvements. As a result, we expect to obtain a high-quality product.

Because we are driven by the need to compete in the market, a product must be developed with respect to quality, cost, and time. It is important to make this product's performance as close to the desired performance as possible. This work provides another contribution and ideas for quality design.

1.5 STRATEGY FOR SOLUTION AND VALIDATION PHILOSOPHY

The *strategy for solution* involves the following:

- Identifying control factors, noise factors/dispersed parameters, and factor levels in the case study.
- Identifying and applying the best suited orthogonal array.
- Running various scenarios and calculating statistical quantities to demonstrate the efficiency of orthogonal array based simulation.
- Determining factor effects using ANOVA.
- Applying techniques involved in quality engineering for a robust design study and formulating and solving a compromise DSP to support the designers of the LifeSat vehicle.

The preceding strategy for the LifeSat model (Chapter 3) is applied in detail in Chapters 4 and 5. The theory of this approach is explained in Chapter 2.

Our *validation philosophy* is based on one case study: the trajectory simulation of the LifeSat space vehicle. We use this case study throughout the report to explain and validate our approach and work. The complexity of this model is high, although, as shown in Chapter 3, it is only based on a few equations. Model implementation is validated by comparison of input and output with documented values. We further validate the approach of using orthogonal arrays for simulation by

- comparing Monte Carlo simulation and orthogonal array based simulation results,
- repeating the simulations with different factor levels and different experiments with these factors, and
- doing statistical calculations and tests to establish confidence.

Before we proceed to describe Taguchi's quality design using orthogonal arrays, we describe the organization of this report in the next section.

1.6 ORGANIZATION OF THIS REPORT

Our work is focused on the simulation of the LifeSat trajectory and is heavily influenced by quality engineering techniques developed by Dr. Genichi Taguchi. The contents of each chapter is briefly described below.

In Chapter 1, the motivation for this study is presented; namely, the problem of reducing the number of simulations of the LifeSat trajectory. Stochastic uncertainty is described

and two probability distributions are introduced. A brief description of Monte Carlo simulation is given. Our frame of reference—namely, quality design and the compromise DSP—are introduced. We define our goals along with the tasks necessary for achieving these goals and the main questions to be answered. Furthermore, the validation philosophy is documented.

In Chapter 2, a review of quality design developed by Taguchi is presented. Factors are classified and the signal-to-noise ratio is introduced. Simulation is identified as a method to obtain information about system performance. A simple example is used to obtain an understanding of several simulation techniques. The core of this chapter—namely, orthogonal arrays—is introduced. The capabilities and properties of orthogonal arrays are discussed. This discussion includes the statistical analysis of results by Analysis of Variance (ANOVA).

In Chapter 3, the analysis model for the trajectory of the LifeSat space vehicle is developed. It is the main case study in this report. We discuss the presence of noise in the model and describe the noise in terms of statistics. We identify a simplified model with test cases and implement this model. Validation of the implemented model with trajectory properties is then presented.

In Chapter 4, we present a comparison of results based on Monte Carlo simulation and simulation using orthogonal arrays. With three different scenarios we establish the foundations for the validation of model simulation based on orthogonal arrays. Techniques introduced in Chapter 2 are applied to model quality into the design of the LifeSat vehicle. In particular, we analyze the contributions of noise factors to the variation in system performance and use the signal-to-noise ratio for a robust design study to identify the best factor level settings.

In Chapter 5, we present the compromise DSP . A compromise DSP of the LifeSat model is developed, implemented, and solved. The signal-to-noise ratio is one of the goals to be achieved while satisfying the system constraints. With a compromise DSP we obtain a better design. Validation and observations based on the results are then documented.

In Chapter 6, we present a critical evaluation and conclusions of the presented work. These includs the goals and achievements of this study. Limitations of this work are identified, and suggestions for future work and closing remarks are given.

PAGE _____ INTENTIONALLY BLANK

CHAPTER 2

ON QUALITY DESIGN AND SIMULATION REDUCTION

Quality is measured in terms of the total loss to society due to functional variation in performance and harmful side effects. Quality is best achieved by minimizing the deviation from the target. We quantify *quality loss* and determine the factors which influence this loss. These factors, which cannot be controlled by a designer, are called *noise factors* The fundamental principle of robust design is to improve the quality of a product by minimizing the effects of the causes of variation without eliminating those causes. Efficient experimentation is necessary to find dependable information about design parameters. The information should be obtained with minimum time and resources. Estimated effects of parameters must be valid even when other parameters are changed. Employing the signal-to-noise ratio to measure quality and *orthogonal arrays* to study many parameters simultaneously are the keys to high quality and robust design.

Since variation in the product performance is similar to quality loss, *analysis of variance* (ANOVA) will be the statistical method used to interpret experimental data and factor effects. ANOVA is a statistically based decision tool for detecting differences in average performance of groups of items tested [Ross, 1988; Roy, 1990].

We use Taguchi's quality technique to lay the foundation for simulation reduction. Three techniques for the simulation of noise factors are introduced. Simulation based on orthogonal arrays is the concept that will be investigated in this report.

2.1 QUALITY, QUALITY LOSS, AND ROBUST DESIGN

Phadke [Phadke, 1989], following Taguchi [Taguchi, 1987], measures the quality of a product in terms of the total loss to society due to functional variation and harmful side effects. Under ideal conditions, the loss would be zero; that is, the greater the loss, the lower the quality. In the following sections, we discuss how we can quantify this loss (Sec. 2.1.1), factors that influence this loss (Sec. 2.1.2), and how we can avoid quality loss (Sec. 2.1.3).

2.1.1 The Quality Loss Function

How can we measure quality loss? Often quality is measured in terms of the fraction of the total number of units that are defective. This is referred to as fraction defective. However, this implies that all units which are within the tolerances of the requirements are equally good (Fig. 2.1(a)). In reality, a product that is exactly on target gives the best performance. As the product's response deviates from the target its quality becomes progressively worse. Therefore, we should not be focusing on meeting the tolerances but on meeting the target.

Taguchi defines the quality loss for not being on target by means of the quadratic quality loss function [Taguchi, 1987; Phadke, 1989]:

$$L(y) = k\,(y - m)^2 \quad , \tag{2.1}$$

where

y	is the quality characteristic of a product/process,
m	is the target value for y, and
k	is a constant called the quality loss coefficient.

The quality loss function is graphically shown in Figure 2.1. For maximum quality the loss must be zero; that is, the greater the loss, the lower the quality.

In Figure 2.1, notice that at y = m the loss is zero and so is the slope of the loss function. In the upper picture, the loss is zero within the range of $m \pm \Delta_0$. In the bottom picture, the loss increases slowly near m but more rapidly farther from m. Qualitatively this is the kind of behavior desired, and Equation (2.1) is the simplest mathematical equation exhibiting this behavior [Phadke, 1989]. The constant k needs to be determined so that Equation (2.1) best approximates the actual loss in the region of interest.

A convenient way to determine the constant k is to determine first the functional limits for the value of y. Let $m \pm \Delta_0$ be the landing range for a space vehicle; e.g., the LifeSat. Suppose the cost (loss) of losing or repairing the vehicle is A_0 when the vehicle lands outside the available area. By substitution into Equation (2.1), we obtain

$$k = \frac{A_0}{\Delta_0^2} . \qquad (2.2)$$

With the substitution of Equation (2.2) into Equation (2.1) we are able to calculate the quality loss for a given value of y. More on the determination of k can be found in [Phadke, 1989].

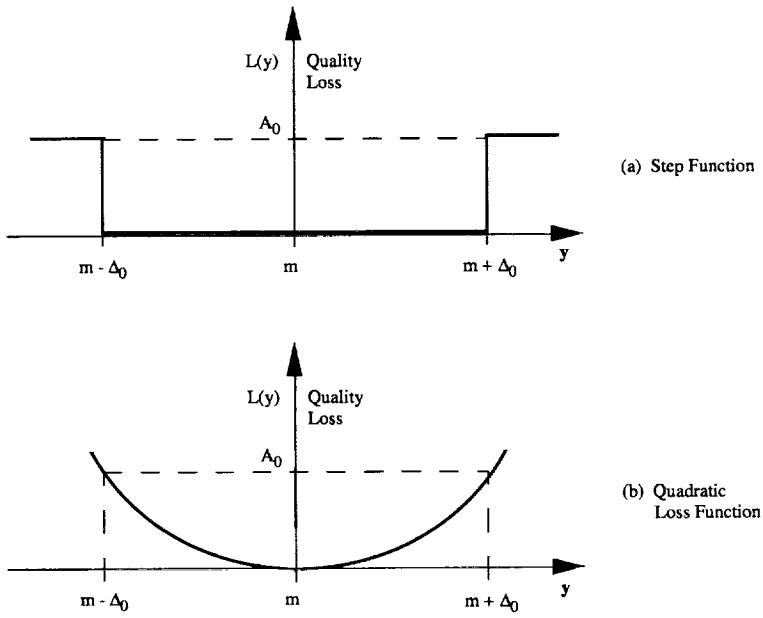


Figure 2.1 – Quality Loss Function [Phadke, 1989]

2.1.2 Factors Affecting Quality

What are the factors influencing the quality of a product or process? In Figure 2.2, a frequently used block diagram is given. It is called a P-diagram where the P stands for either product or process.

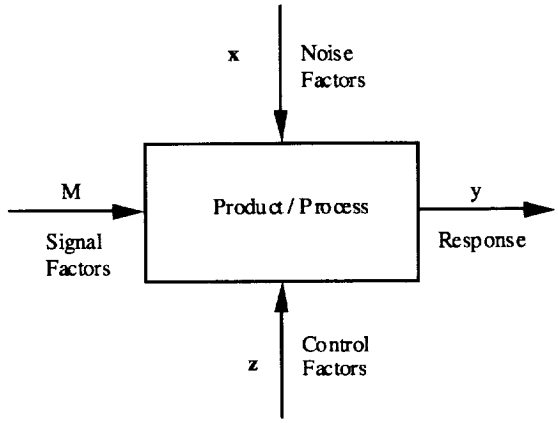


Figure 2.2 – Block Diagram of a Product/Process – P-Diagram [Phadke, 1989]

A number of parameters (also known as factors) influence the quality characteristic, y, of a system. Three types are distinguished:

- *Signal Factors (**M**)* – These are parameters set by the user or operator of the product to express the intended value for the response of the product. In other words, signal factors are the targets to be achieved.

- *Noise Factors (**x**)* – Certain parameters cannot be controlled by a designer and are called noise factors. Parameters whose settings (called levels) are difficult or expensive to control are also considered noise factors. The levels of noise factors change from process to process, from one environment to another, and from time to time. Often only the statistical characteristics (such as the mean and variance) of noise factors can be known or specified and the actual values in specific situations cannot be known. The noise factors cause the response y to deviate from the target specified by the signal factor M and lead to quality loss.

- *Control Factors (**z**)* – These are parameters that can be specified freely by a designer. In fact it is a designer's responsibility to determine the best values for these parameters. Each control value can take multiple values, which are called levels. Phadke refers to control factors which affect manufacturing cost as tolerance factors.

The LifeSat tasks can easily be stated in terms of quality and the factors used in the P-diagram as follows:

> The *quality characteristic* is the ability to follow the desired trajectory and land at the desired target. The *response* of the system is the actual trajectory including the deviation from the target. The *noise factors* are the environmental and vehicle parameters. *Control factors* are specified by the designer; hence, they are related to the vehicle like mass, velocity, dimensions, or coefficients. In this case, no *signal factors* are applied.

In the following section, we answer the question of how to avoid quality loss. We introduce the *signal-to-noise ratio* and the ideas of *robust design.*

2.1.3 Signal-to-Noise Ratio and Robust Design

How can we avoid quality loss? Taguchi has developed a ***signal-to-noise ratio*** (S/N) as a predictor of quality loss after making certain simple adjustments to the system's function [Taguchi, 1987; Phadke, 1989]. This ratio isolates the sensitivity of the system's function to noise factors and converts a set of observations into a single number. It is used as the objective function to be maximized in Robust Design [Phadke, 1989].

Three possible categories of quality characteristics exist. These are

- smaller is better; e.g., minimum shrinkage in a cast iron cylinder
- nominal is better; e.g., dimensions of a part with small variance
- larger is better; e.g., achieve the highest efficiency

We focus on the second type (nominal is better) because this category most accurately represents the LifeSat. For a "nominal is better" problem the quality has to meet a certain target, Y_0, and the a quality characteristic, η, is defined by [Roy, 1990] as

$$\eta = -10 \log_{10} (MSD) , \tag{2.3}$$

where η is expressed in decibels (dB) and MSD, the Mean Squared Deviation, is calculated from

$$MSD = ((Y_1 - Y_0)^2 + (Y_2 - Y_0)^2 + ... + (Y_N - Y_0)^2)/N . \tag{2.4}$$

In Equation (2.4), Y_i is the system response and N is the number of trials. There are other formulations of the signal-to-noise ratio, which can be found in [Taguchi, 1987; Phadke, 1989; Ross, 1988].

The conversion to the signal-to-noise ratio can be viewed as a scale transformation for convenience of better manipulation. It offers an objective way to look at two characteristics; namely, variation and average (mean) value. Analysis using the signal-to-noise ratio has two main advantages.

- It provides a guidance to a selection of the optimum level based on least variation around the target and also on the average value closest to the target.
- It offers objective comparison of two sets of experimental data with respect to variation around the target and the deviation of the average from the target value.

For the ***robust design*** of a product, two steps are required.

- Maximize the signal-to-noise ratio η. During this step, the levels of control factors to maximize η are selected while ignoring the mean.
- Adjust the mean on target. For this step, a control factor is used to bring the mean on target without changing η.

This is clarified in Figure 2.3, where we have given the target to be achieved and the tolerances around the target. These tolerances define a probability distribution. The larger the tolerances, the more products will satisfy these tolerances. Robust design is concerned with aligning the peak of the bell-shaped quality distribution with the targeted quality; that is, reducing the bias (Fig. 2.3). Furthermore, by increasing the signal-to-noise ratio, the bell shape becomes thinner. This means that the variation in quality is reduced and fewer products will be outside the tolerances set on the targeted quality. The robustness of the quality is increased in this way. Maximization of the signal-to-noise ratio and reduction of the bias form the core of robust design.

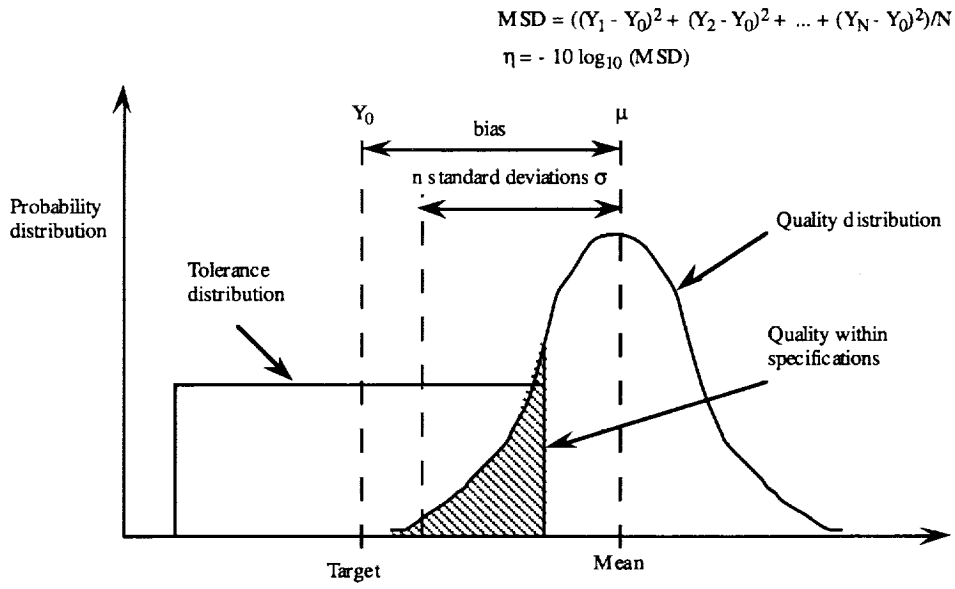


Figure 2.3 – A Graphical Representation of Robust Design

In the LifeSat example, we are concerned with the shaded area in Figure 2.3 which represents the distribution of the landing position within the specified tolerances of the target. For spaceflight applications, 99.73% of the area under the quality distribution must be within the specifications. This is similar to using three standard deviations on both sides of the mean (Sec. 1.2). Following the robust design approach, we want to decrease the variation around the mean of the landing position and simultaneously minimize the bias between this mean and the target.

Further information on quality loss and signal-to-noise ratios can be found in [Phadke, 1989; Taguchi, 1987; Ross, 1988; Roy, 1990]. Suh [Suh, 1990] discusses how to apply statistical methods and Taguchi's approach in the selection of design parameters for satisfying functional requirements. A maximization of the signal-to-noise ratio reduces the information content of a design in keeping with Suh's second design axiom [Suh, 1990].

In his book, Phadke describes Robust Design as a method to improve the quality of a product. The fundamental principle of Robust Design is to improve quality of a product by minimizing the effect of the causes of variation without eliminating the causes. In Robust Design, two important tasks need to be performed [Phadke, 1989].

- Measurement of quality during design and development. We want a leading indicator of quality by which we can evaluate the effect of changing a particular design parameter on the product's performance.

- Efficient experimentation to find dependable information about design parameters. It is critical to obtain dependable information about the design parameters so that design changes during manufacturing and customer use can

be avoided. Also, the information should be obtained with minimum time and resources.

We have mentioned the specification of having 99.73% of all missions land within the specified target range. How do we get the information about the deviation from the target and the variations due to the noise factors? We have to perform experiments which provide the necessary results and information. In the LifeSat example, it is impossible to do experiments with the actual system in the design phase. Therefore, a model has to be developed for the trajectory from entry interface until landing. In the following section, we give an overview about simulation involving noise factors based on traditional methods. In Section 2.3, we introduce orthogonal arrays—the core of Taguchi's experimental design technique—and apply this technique in Section 2.4 for simulation based on orthogonal arrays.

2.2 AN OVERVIEW ABOUT SIMULATION

In this section, we first define the necessary terminology used and discuss advantages and disadvantages of simulation. Let us assume there are k noise factors denoted by x_1, x_2, ..., x_k. These noise factors are specified by the mean and the standard deviation. How can we evaluate the mean and the variance in the output response of our system? Three common methods for these evaluations are

- the Monte Carlo simulation,
- the Taylor series expansion, and
- a simulation based on orthogonal arrays.

We introduce the ideas of Monte Carlo simulation (Sec. 2.2.1) and Taylor series expansion (Sec. 2.2.2) as two methods to simulate noise. An example is used to demonstrate limitations of these techniques.

2.2.1 Terminology and Characteristics of Simulation

The following definitions are used:

Simulation – is defined as a technique used to derive solutions when analytical or numerical solutions methods break down or are impractical to employ. It is used very often to gain an understanding of a *system*, rather than to obtain a solution in the mathematical sense [Donaghey, 1989].

A **system** – is a set of objects united by some form of interaction or interdependence that performs a specified set of tasks. The system performs a function or process which results in an output due to some input [Donaghey, 1989].

A real system placed in its real environment usually represents a complex situation; e.g., space stations, ships, or airplanes. The scientist or the engineer who wishes to study a real system must make many concessions to reality to perform some analysis on the system. In practice, we never analyze a real system but only an abstraction of it. A **model** is

an abstract description of the real world giving an approximate representation of more complex functions of physical systems [Papalambros and Wilde, 1988].
Deterministic models - are models that will give a unique set of outputs for a given set of inputs.

Nondeterministic or **stochastic models** – are models in which relationships depend on distributed parameters. Outputs for a given set of inputs can be predicted only in a probabilistic context.

The model of a real system is simulated to study its behavior. Simulation has both advantages and disadvantages. The advantages of simulation are as follows:

- Most complex real world systems with stochastic elements cannot be accurately described by a mathematical model, which can be evaluated analytically. Thus, a simulation is often the only type of investigation possible.
- Simulation allows us to estimate the performance of an existing system under some projected set of operating conditions.
- Alternative proposed system designs can be compared via simulation to see which design best meets a specified requirement.
- In a simulation, we can maintain much better control over experimental conditions than would be possible when experimenting with the system itself.
- Simulation allows us to study a system along a time frame.

The disadvantages of simulation are as follows:

- Simulation models are often expensive and time consuming to develop.
- On each run a stochastic simulation model produces only estimates of a model's true characteristics for a particular set of input parameters. Thus, several independent runs of the model will probably be required for each set of independent input parameters to be studied.
- The large volume of numbers produced by a simulation study often creates a tendency to place greater confidence in study results than is justified.

Dynamic systems are often analyzed by simulation since, as mentioned above, it may be difficult to obtain the solution analytically. In the LifeSat model which will be developed in Chapter 3, we are interested in the system response or output. One output is the landing position. For the simulation we need to know the position, the velocity, and the acceleration of the vehicle. But the position (**x**) is a function of the velocity (**v**), acceleration (**a**), and time (t), and the velocity is again dependent on acceleration and time. Acceleration is dependent on position and velocities. These relationships are shown in Equation (2.5) as

$$\mathbf{x} = f_1(\mathbf{v}, \mathbf{a}, t),$$
$$\mathbf{v} = f_2(\mathbf{a}, t), \text{ and} \qquad (2.5)$$

$$\mathbf{a} = f_3(\mathbf{x}, \mathbf{v}, t).$$

It is very difficult to obtain an analytical solution. Therefore, we numerically integrate the acceleration and velocity beginning at the initial state. In the beginning of the LifeSat simulation, we need an initial state that represents the position and the velocity. The acceleration is then calculated from these values, and velocity and position are updated or integrated using small time steps. This time step has to be adjusted to the dynamics of the system. After each iteration we obtain an approximation of the real system.

There exist several numerical ways to do the integration of a function g(x). The simplest one is the Euler technique [Bronstein and Semendjajew, 1987]. The slope of a function y(x) is defined as

$$\frac{dy}{dx} = y' = g(x) \quad , \tag{2.6}$$

and the new function values are calculated as

$$y_1 = y_0 + (x_1 - x_0)g(x_0) = y_0 + \Delta x g(x_0) \ ,$$

$$y_2 = y_1 + (x_2 - x_1)g(x_1) = y_1 + \Delta x g(x_1) \quad ,$$

$$\vdots$$

and

$$y_k = y_{k-1} + \Delta x g(x_{k-1}) \ . \tag{2.7}$$

Other existing integration methods, like the Runge-Kutta [Bronstein and Semendjajew, 1987] method, are more accurate but require more information. We therefore first study the differences in the output by employing different time steps for the Euler technique. The larger the time step, the fewer the iterations we need for the simulation starting at the initial state and stopping when a termination criterion is satisfied. The smaller the time step, the higher the accuracy until numerical round-off errors influence the result. This trade-off depends on the model behavior.

In this section, we have discussed advantages and disadvantages of simulation. We have identified the necessity for simulation of the LifeSat model and have introduced the Euler technique to be employed for numerical integration. To estimate the effects of noise factors, a simulation is necessary to investigate the system. In the following section, we introduce the Monte Carlo simulation and discuss some of its properties using a simple example.

2.2.2 Introduction to Monte Carlo Simulation

In the Monte Carlo simulation, a random number generator, which is usually provided by most computers, is used to simulate a large number of combinations of noise factors and

parameters with tolerances. These combinations are called testing conditions. The value of the response is computed for each testing condition, and the mean and the variance of the response are then calculated. For obtaining accurate estimates of mean and variance, the Monte Carlo method requires evaluation of the response under a large number of testing conditions. This is very expensive if the simulation requires a large amount of computational time and if we also want to compare many combinations of control factor levels. This is a disadvantage (Sec. 2.2.1).

The following example provides some insight into the behavior of a Monte Carlo simulation. We calculate the statistics of mean and standard deviation of a function F(**x**) in order to have mathematical values for comparison. A function F(**x**) consists of the sum of six squared variables $x_1, x_2, ..., x_6$. For each of the six input variables, x_i, we assume a normal distribution with a mean of $\mu_i = 0$ and a standard deviation of $\sigma_i = 1$. This means, most variable values will be placed near the mean value of zero. The function is given as

$$F(\mathbf{x}) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 + x_6^2 \ . \qquad (2.8)$$

The function is useful to study the behavior for simulation when we do not have a normally distributed output. The reason for the choice of this function is the fact that we have an analytical solution for the mean and the standard deviation. As documented in [Sokal and Rohlf, 1981; Donaghey, 1989], this function has a chi-square distribution with a mean of $\mu = 6$, and a standard deviation of $\sigma = \sqrt{12} = 3.464$. The function output is skewed to the right, which means that there is a tail on the right side of the output distribution. The distribution has no negative values and begins at zero. It is shown in Figure 2.6 later in this section. If all factors, x_i, are set to their mean values, the function output is zero and does not represent the analytical mean at all.

We have done six Monte Carlo simulations with 1000 function evaluations each. One function output value is one *sample*; hence, we have a sample size of 1000 each. The six random variables, x_i, for one function evaluation are called *error vectors*. This error vector is generated by the computer which follows a special sequence based on an initial seed value. If we change this seed value, we obtain different random numbers in a run.

We are interested in the mean μ and the standard deviation σ of the function output and also in the convergence of these values to the analytical solutions. We expect a better estimation with increasing sample size.

In Figure 2.4, the mean value for the function F is plotted for six runs. We identify six lines which represent the different runs. For each sample the new mean is calculated. When the number of samples is below 100, there is large difference between the different lines. If we do not consider the first few simulations, the lines vary between a mean value of 5.2 and 7.3. In the range from 100 to 200 numbers of samples, the mean varies between 5.7 and 7.0 for different lines. The changes for one line can be large within a small difference of sample size. When the sample size further increases, the mean value converges. After 1000 function evaluations, the mean values for different runs have not exactly converged to the analytical mean value of six. The total difference between the highest and the lowest mean estimation is nearly 0.5, which is approximately 10% of the mean. We also observe the trend that a larger number of runs estimates a higher mean

value. Five runs are above a value of six; one run has approached six. The different behavior of each line demonstrates the different choice of seed numbers for each run.

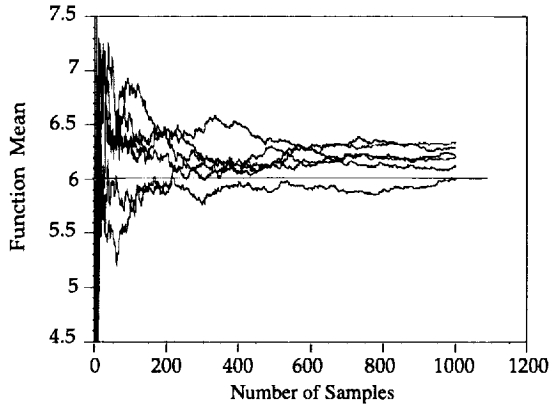


Figure 2.4 - Function Mean Value for Monte Carlo Simulation

The behavior of the standard deviation for the function is shown in Figure 2.5. As for the mean, there is large variation for the standard deviation during the first 200 samples. Also, the convergence for more samples is similar. After about 400 samples the changes within each run are small, whereas the differences in two runs still can be large. At the final sample size of 1000, the values for the standard deviation have *approached* its theoretical value of 3.464, but the differences for several runs remain.

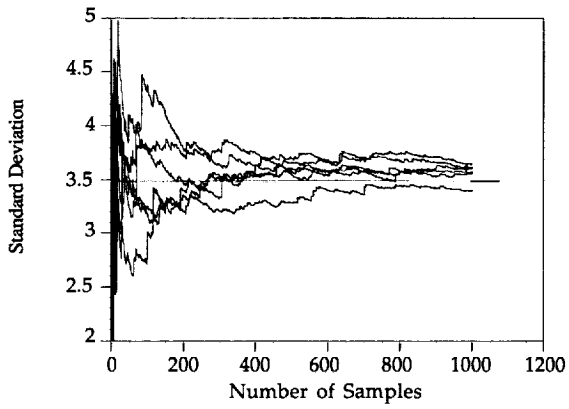


Figure 2.5 - Function Standard Deviation for Monte Carlo Simulation

In Figure 2.6, we present two *histograms* to show the *frequency* of function output values for two different runs. The frequency represents the number of function values falling into the interval 0-1, 1-2, 2-3, and so on. Because of the quadratic function, we have no negative values but theoretically any positive value is possible. The total number of frequencies equals 1000. Both pictures of Figure 2.6 have some differences, although the overall shape is similar. In the left picture, we observe at least one function value greater than 28; whereas, in the right picture, there is no value larger than 24. This again demonstrates the differences in different sample populations and also shows the skewed distribution of the function.

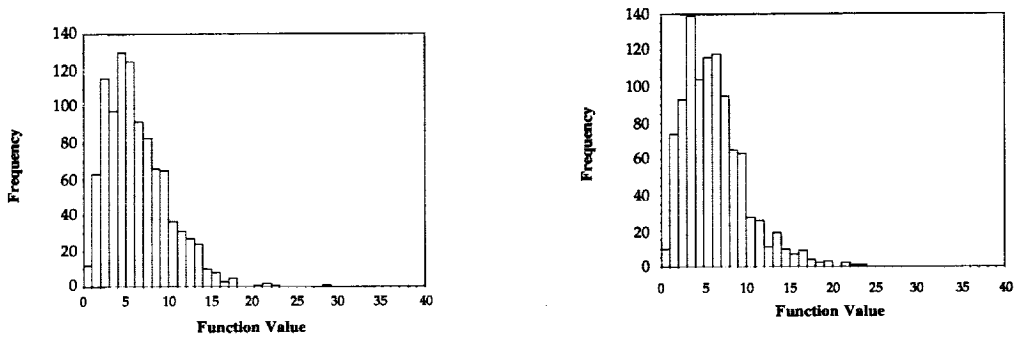


Figure 2.6 - Histograms for Two Monte Carlo Simulations

From the observations made in the simulated function, F, we draw two important conclusions about Monte Carlo simulation.

- Small variations or convergence for statistics does not guarantee that the population mean and standard deviation are indeed approached.
- Several independent runs result in different solutions. Therefore, a sample size of 1000 may not be enough to get a "good" estimation (this of course is dependent on the system or functional relations and on the number of factors involved) of the statistics. Statistical estimates are usually combined with a statistical confidence interval.

In the next section, we briefly explain a noise factor simulation method based on Taylor series expansion.

2.2.3 Simulation Based on Taylor Series Expansion

According to Phadke [Phadke, 1989], in the Taylor series expansion method the mean response is estimated by setting each noise factor equal to its nominal (mean) value. To estimate the variance of the response, we need to find the derivative of the response with respect to each noise factor. Let F denote the function and $\sigma_1^2, \sigma_2^2, ..., \sigma_k^2$ denote the variances of the k noise factors. The variance of F obtained by first-order Taylor series expansion is then calculated by [Phadke, 1989]

$$\sigma_F^2 = \sum_{i=1}^{k} \left(\frac{\partial F}{\partial x_i} \right)^2 \sigma_i^2 \quad . \tag{2.9}$$

The derivatives used in Equation (2.9) can be evaluated mathematically or numerically. The equation, which is based on first-order Taylor series expansion, gives quite accurate estimates of variance when the correlations among noise factors are negligible and the tolerances are small so that interactions among the noise factors and the higher order terms can be neglected. Otherwise higher order Taylor series expansions must be used, which makes the equation for evaluating the variance of the response complicated and computationally expensive. Thus, Equation (2.9) does not always give an accurate estimate of variance. Another disadvantage is that we do not obtain an estimation for the mean. In Section 2.2.3, we have seen that the mean of a function is not necessarily obtained if the factors are set to their nominal values.

We want to use the same function, F, used in Equation (2.8) in Section 2.2.3 and to apply the Taylor series expansion technique. We calculate all the derivatives as

$$\frac{\partial F}{\partial x_i} = 2x_i \quad . \tag{2.10}$$

Substituting x_i with the mean value $\mu_i = 0$ and using $\sigma_i = 1$, we obtain

$$\sigma_F^2 = 4 \sum_{i=1}^{k} x_i^2 \sigma_i^2 = 0.0 \quad . \tag{2.11}$$

The estimation of the variance does not give a satisfying value. For the function F at the given means and variances for the noise factors, neither the mean nor the variance is obtained. The reason for this is easy to understand when we recall the function again. All factors are squared; hence, the function value for all factors at $x_i = 0.0$ is a local minimum for the function. All first-order derivatives are zero and, by using Equation (2.11), a variance of zero is calculated. We agree that this example does not show qualities of this noise factor simulation method, but identifying limitations is important to avoid mistakes.

This is one case where the method fails; however, we find examples where the method is applied successfully. One examples is shown in Ullman [Ullman, 1992] for the design of a tank to hold liquid. Not only is the Taylor series expansion used but the methods of robust design are also applied in a deterministic way.

In the following section, we introduce orthogonal arrays and their properties. They provide the basis for simulation based on orthogonal arrays and, hence, a way to reduce the large number of simulations as required for Monte Carlo simulation. We also overcome problems identified for Taylor series expansion.

2.3 INTRODUCTION TO ORTHOGONAL ARRAYS

The technique of defining and investigating the possible conditions in an experiment involving multiple factors is known as the design of experiments. The concept of experimental design involving multiple factors was first introduced by Sir Ronald Fisher nearly 70 years ago. The technique is known as a *factorial design of experiments*. It is also called a matrix experiment. In Section 2.3.1, we provide some background about factorial design with factorial experiments. The concepts of orthogonal arrays are explained in Section 2.3.2, and an algorithm to create three-level orthogonal arrays is developed in Section 2.3.3.

2.3.1 Factor Experiments

If all possible combinations of the given set of factors are considered, we call it a *full factorial design*. For example, in an experiment where five factors are involved—each factor on three levels—the total number of combinations will be $3^5 = 243$. A factor level is the setting of this factor to a specific value. A full factorial design with two factors, A and B, at two levels, 1 and 2, is shown in Table 2.1.

Table 2.1 - Full Factorial Experiment with Two Factors at Two Levels

	Factor and Factor Level	
Experiment	A	B
1	1	1
2	1	2
3	2	1
4	2	2

We have four possible combinations (experiments) for two factors and two levels which are

A_1B_1, A_1B_2, A_2B_1, and A_2B_2.

With three factors A, B, and C at two levels, we have eight (2^3) combinations; namely,

$A_1B_1C_1$, $A_1B_2C_1$, $A_2B_1C_1$, $A_2B_2C_1$,

$A_1B_1C_2$, $A_1B_2C_2$, $A_2B_1C_2$, and $A_2B_2C_2$.

The increase of possible combinations is exponential and, for engineering problems involving many factors, the number of possible combinations is extremely large. The question is, How can an engineer efficiently investigate these design factors? To reduce the number of experiments to a practical level, a smaller set from all possibilities is selected. The technique of selecting a limited number of experiments to produce the most information is known as fractional factorial experiment or fractional factorial design. In

Ross [Ross, 1988], some efficient test strategies for fractional factorial experiments are presented.

Taguchi [Taguchi, 1987] has established Orthogonal Arrays (OAs) to describe a large number of experimental situations. Orthogonal arrays have been investigated by many other researchers; e.g., Kempthorne, Addelman, and Seiden [Kempthorne, 1979; Addelman, 1962; Seiden, 1954]. Taguchi's approach complements three important areas. First, Taguchi clearly defined a set of orthogonal arrays, each of which can be used for many situations. Second, he has devised a standard method for analyzing the results. Third, he has developed a graphical tool, called *linear graphs*, to represent interactions between pairs of columns in an orthogonal array. Roy [Roy, 1990] says, *"Taguchi's approach of combining standard experimental design techniques and analysis methods produces consistency and reproducibility rarely found in any other statistical method."*

The real power of an OA is its ability to evaluate several factors with a minimum of tests/ trials or experiments. Much information is obtained from a few tests. In Table 2.2, the number of all possible combinations for two- and three-level factors is compared to the number of experiments in Taguchi's orthogonal arrays.

Table 2.2 - Full Factorial Design Comparison with Taguchi Design

		Factorial Design	Taguchi Design
Factors	Levels	Number of Experiments/Trials	
3	2	8 (2^3)	4
7	2	128 (2^7)	8
15	2	32,768 (2^{15})	16
4	3	81 (3^4)	9
13	3	271,594,323 (3^{13})	27

Orthogonal arrays are denoted by L_x, where x represents the number of trials. For example, in the L_8 array with a maximum number of seven factors at two levels, only eight of the possible 128 combination are required. The number of columns and trials in an orthogonal array depends on the degrees of freedom, as described in Section 2.5.1. The properties of orthogonal arrays are explained in Section 2.3.2.

2.3.2 The Concept of Orthogonality in Orthogonal Arrays

Orthogonal arrays are special matrices used for matrix experiments or fractional factorial designs. They allow the effects of several parameters to be determined efficiently. Taguchi has introduced many standard orthogonal arrays for matrix experiments; e.g., L_9. This standard orthogonal array is shown below. For each of the four factors A, B, C, and D of L_9 there are three levels (1, 2, and 3) that are represented in the columns.

Table 2.3 - Standard Orthogonal Array L_9

Experiment	A	B	C	D
1	1	1	1	1
2	1	2	2	2
3	1	3	3	3
4	2	1	2	3
5	2	2	3	1
6	2	3	1	2
7	3	1	3	2
8	3	2	1	3
9	3	3	2	1

Orthogonality is interpreted in the combinatorial sense. For any pair of columns, all combinations of factor levels occur an equal number of times. This is called the *balancing property* [Phadke, 1989]. For instance in the orthogonal array L_9 for each pair of columns, there exist 3 x 3 = 9 possible combinations of factor levels. Any two columns of L_9 have nine combinatorial levels—namely, (1,1), (1,2), (1,3), (2,1), (2,2), (2,3), (3,1), (3,2), and (3,3)—but each combination only appears once [Suh, 1991]. Six different combinations of two columns can be formed (AB, AC, ..., CD) from the four columns (A, B, C, and D), which all have the same balancing property.

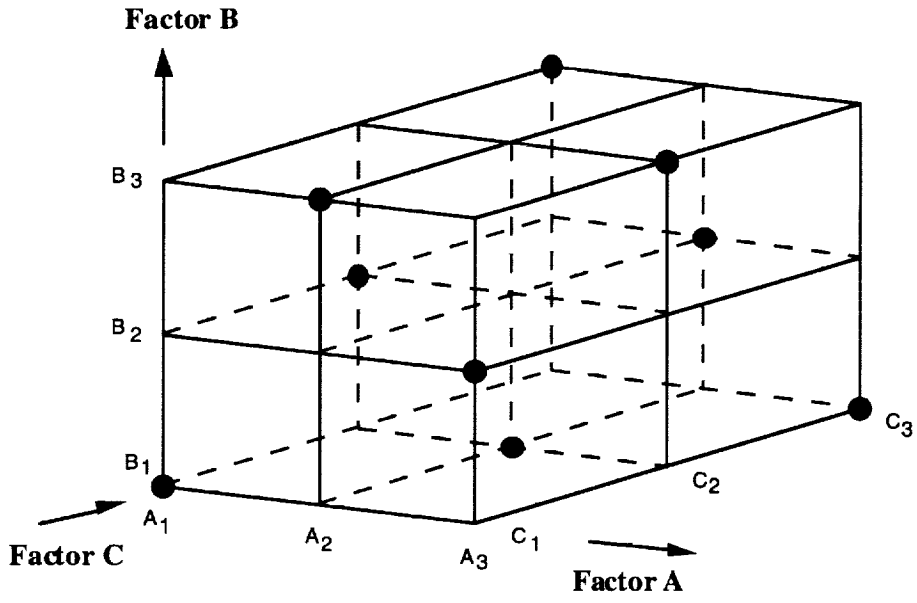


Figure 2.7 - Balancing Property in Orthogonal Arrays

The balancing property or orthogonality is demonstrated in an example where we have three factors A, B, and C with three levels each. In this case, the factors are assigned to Columns 1, 2, and 3 in the orthogonal array L_9 since we have only three factors and one column is therefore empty. Each dot in Figure 2.7 represents one combination of the nine experiments; hence, we have nine dots. The total number of all combinations is 27. When we observe the picture, we identify that every plane contains exactly three dots. Because of the balancing property, we have the minimum number of combinations necessary for three factors with three levels to obtain information about factor effects.

In the following, we summarize the main benefits by using orthogonal arrays:

- Conclusions derived from the experiments are valid over the entire experimental region spanned by the control factors and their settings.
- There is a large saving of experimental effort and, therefore, a reduction of computational time for simulations.
- Data analysis can be done easily.
- Orthogonal arrays and their experiments are designed deterministically, not randomly.

Orthogonal arrays are usually selected from existing standard orthogonal arrays. For use on a computer, it is convenient to create the orthogonal arrays when they are needed as discussed below.

2.3.3 An Automated Way for Three-Level Orthogonal Array Creation

To implement orthogonal arrays on the computer, we have developed an algorithm to create orthogonal arrays for three-level parameters. These are the arrays L_9, L_{27}, and L_{81} which are used for 4, 13, and 40 parameters, respectively. The basic information is provided by Latin Squares which present the smallest orthogonal entities. When there is a complete orthogonal system of (n-1) Latin Squares, each Latin Square of dimensions n x n, denoted by $L_1, L_2, ..., L_{n-1}$, it is possible to construct an orthogonal array of size n^r (r = 2, 3, ...) and number of columns $(n^r - 1)/n - 1$. The number r represents the number of trials. For more detail, see [Taguchi, 1987]. In constructing an orthogonal array for a k-level system, Bose and Bush [Bose and Bush, 1952] have shown a method of using a matrix whose elements are taken cyclically. Masuyama [Masuyama, 1957] has used the theory of algebraic numbers for the first time in constructing orthogonal arrays. The three mentioned arrays L_9, L_{27}, and L_{81} are basically built up from two Latin Squares of dimension 3 x 3. L_9 is created from the Latin-Squares, L_{27} from L_9, and L_{81} from L_{27}.

Latin Square 1		
1	2	3
2	3	1
3	1	2

Latin Square 2		
1	3	2
2	1	3
3	2	1

Figure 2.8 - Latin Squares for Three Levels

In Table 2.3, the orthogonal array L_9 is divided into three blocks from which the higher orthogonal array (L_{27}) is created. Each block represents one first column, A, with the same number. The three other columns B, C, and D contain: an equal value within one row in Block 1, the values of Latin Square 1 (Fig. 2.8) in Block 2, and the values of Latin Square 2 (Fig. 2.8) in Block 3.

Exp.	A	B	C	D	
1	1	1	1	1	
2	1	2	2	2	**Block 1**
3	1	3	3	3	
4	2	1	2	3	
5	2	2	3	1	**Block 2**
6	2	3	1	2	
7	3	1	3	2	
8	3	2	1	3	**Block 3**
9	3	3	2	1	

Figure 2.9 - Orthogonal Array L_9 Divided in Three Blocks

Generally, we take each column from the smaller array to create the new array with three times more experiments than in the old array and three plus one times more columns. When we create L_{27} from L_9, the array size changes from 9 to 27 experiments and from 4 to 13 columns.

The rules for the creation of a new three-level orthogonal array are always the same and are given in four steps. See Figure 2.10 where the rows of orthogonal array L_9 are shown in "bold" in columns B, E, H, and K of the third block, represented by experiments 19 to 27. These rows are also found in the first two blocks (experiments 1 to 9 and 10 to 18) of the same columns.

The four steps to create a new orthogonal array are given as follows:

Step 1
Starting at column 1 in the new array, we create three new blocks by assigning Level 1 to the first third of rows (Block 1), Level 2 to the second third of rows (Block 2), and Level 3 to the last third (Block 3).

Step 2
In Block 1, Columns 2 to 4 are created by taking the first column of the old array (which has the same length as one new block) and assigning the same level in one row to all three columns. Columns 5 to 7 are then created by using the second column of the old array. This continues until all columns in Block 1 are filled.

Step 3
In Block 2, Columns 2 to 4 are created by taking the first column of the old array and by assigning a row of Latin Square 1 starting with the old value. This step is repeated until all columns in Block 2 are filled.

Step 4
Block 3 is created similarly to Block 2, but instead of Latin Square 1 we use Latin Square 2.

The algorithm presented by the steps is implemented in a software subroutine as shown in Appendix A. Depending on the given number of parameters, the subroutine selects the suitable orthogonal array and assigns the factor values corresponding to the levels 1, 2, and 3.

Exp.	A	B	C	D	E	F	G	H	I	J	K	L	M
1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	1	1	1	1	2	2	2	2	2	2	2	2	2
3	1	1	1	1	3	3	3	3	3	3	3	3	3
4	1	2	2	2	1	1	1	2	2	2	3	3	3
5	1	2	2	2	2	2	2	3	3	3	1	1	1
6	1	2	2	2	3	3	3	1	1	1	2	2	2
7	1	3	3	3	1	1	1	3	3	3	2	2	2
8	1	3	3	3	2	2	2	1	1	1	3	3	3
9	1	3	3	3	3	3	3	2	2	2	1	1	1
10	2	1	2	3	1	2	3	1	2	3	1	2	3
11	2	1	2	3	2	3	1	2	3	1	2	3	1
12	2	1	2	3	3	1	2	3	1	2	3	1	2
13	2	2	3	1	1	2	3	2	3	1	3	1	2
14	2	2	3	1	2	3	1	3	1	2	1	2	3
15	2	2	3	1	3	1	2	1	2	3	2	3	1
16	2	3	1	2	1	2	3	3	1	2	2	3	1
17	2	3	1	2	2	3	1	1	2	3	3	1	2
18	2	3	1	2	3	1	2	2	3	1	1	2	3
19	3	**1**	3	2	**1**	3	2	**1**	3	2	**1**	3	2
20	3	**1**	3	2	**2**	1	3	**2**	1	3	**2**	1	3
21	3	**1**	3	2	**3**	2	1	**3**	2	1	**3**	2	1
22	3	**2**	1	3	**1**	3	2	**2**	1	3	**3**	2	1
23	3	**2**	1	3	**2**	1	3	**3**	2	1	**1**	3	2
24	3	**2**	1	3	**3**	2	1	**1**	3	2	**2**	1	3
25	3	**3**	2	1	**1**	3	2	**3**	2	1	**2**	1	3
26	3	**3**	2	1	**2**	1	3	**1**	3	2	**3**	2	1
27	3	**3**	2	1	**3**	2	1	**2**	1	3	**1**	3	2

Figure 2.10 - Orthogonal Array L_{27} with 13 Factors (A - M) on Three Levels

We have seen the orthogonal arrays L_9 and L_{27} in this section. The L_{81} and all other standard orthogonal arrays can be found in [Taguchi, 1987; Phadke, 1989]. Having identified the properties of orthogonal arrays and implemented an algorithm to create them (restricted to three-level OAs), we now have the background to introduce simulation that is based on orthogonal arrays.

2.4 SIMULATION BASED ON ORTHOGONAL ARRAYS

In this section, we introduce the simulation based on orthogonal arrays. We follow the suggestions of Taguchi and Phadke for the selection of factor levels. In Section 2.4.1, we introduce the basic methods for performing simulations with orthogonal arrays, which is demonstrated with an example in Section 2.4.2.

2.4.1 Simulation of Variation in Noise Factors Based on Orthogonal Arrays

Taguchi has proposed to sample the domain of noise factors using orthogonal arrays. If we use two- or three-level noise variables, the following level values are suggested in [Phadke, 1989]:

- For a normally distributed two-level variable x_i, having the mean μ_i and the variance σ_i^2, we choose the levels to be $\mu_i - \sigma_i$ and $\mu_i + \sigma_i$.

- For a three-level variable, we choose the levels to be $\mu_i - \sqrt{1.5}\sigma_i$, μ_i, and $\mu_i + \sqrt{1.5}\sigma_i$, having also the mean μ_i and the variance σ_i^2.

A selection of three levels rather than two levels for the noise factors gives a variance estimate of higher accuracy. However, the use of two-level orthogonal arrays leads to a smaller number of simulations and, hence, a time and cost savings. For space missions we are most concerned about the accuracy and confidence; therefore, we chose at least three levels for each noise variable. The choice of more levels is preferred since the number of orthogonal array simulations will still be very small compared to the number of Monte Carlo simulations. As explained in Section 2.3.3, an algorithm was developed that will create three-dimensional arrays with up to 40 variables (L_{81}). There are not many standard orthogonal arrays available in the literature which cover more than three levels and a large number of variables. In the simulation of the LifeSat model, we have used the three-level array L_{27} for up to 13 variables.

In the literature, we have not found suggestions for the factor levels for uniformly distributed variables with the mean μ and the tolerance Δ. We choose the three levels to be $\mu - \Delta$, μ , and $\mu + \Delta$, respectively, since these are levels are often chosen in orthogonal array experiments [Phadke, 1989; Suh, 1991].

Despite the suggested factor levels we should investigate and simulate the model with other levels. With the selections of different factor levels we have the opportunity to place points closer or further away from the mean. We can have a representation where more samples fall into the 1σ-level or more fall into the 3σ-level of the input factors.

Simulation with orthogonal arrays seems to be simple and is only dependent on the right choice of factor levels. As an example of the principal technique, we can use the six-dimensional function $F(\mathbf{x})$ already examined in Section 2.2.2 to demonstrate simulation with orthogonal arrays.

2.4.2 An Example for Orthogonal Array Based Simulation

For the following example we want to recall the function F(**x**) of Section 2.2.2 that is defined as

$$F(\mathbf{x}) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 + x_6^2 \ ,$$

with a mean of zero and a standard deviation of one for each factor. We have six variables and decide to select three levels for each variable. Therefore, the orthogonal array L_{27} is the smallest available array to be used for the simulation. Since this array has 13 columns (recall Section 2.3.2), we choose the last six columns to assign the variables for these columns with the corresponding levels.

All variables are normally distributed; hence, as suggested in Section 2.4.1, the levels for the variables are selected to be $\mu + \sqrt{1.5}\sigma$, μ, and $\mu - \sqrt{1.5}\sigma$. In Figure 2.11, we present the results of 27 experiments for the function value and the function mean which is calculated after each experiment.

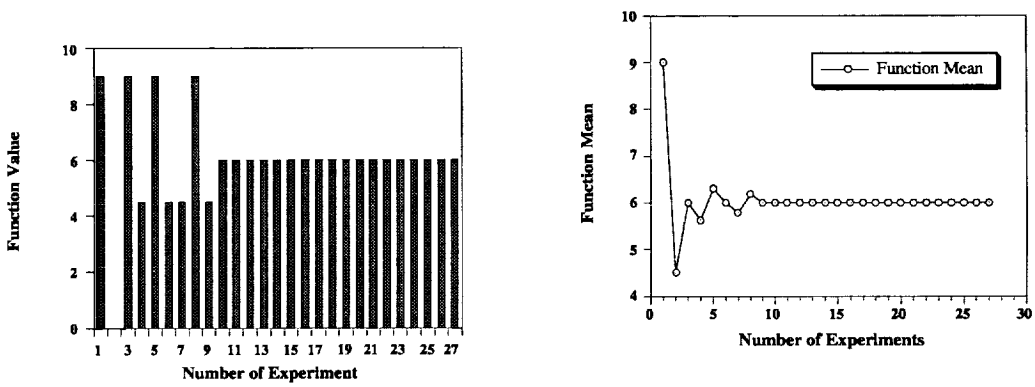


Figure 2.11 - Function Values and Mean Using Orthogonal Arrays

According to the last six columns of orthogonal array L_{27}, all factors are at level 1 in the first experiment. The function value becomes nine. The second experiment results in a function value of zero since all factors are at level 2, which means a value of zero for each variable x_i. The other results can be verified with the L_{27}. Obviously, the output remains the same at a value of six from the 10^{th} to the 27^{th} experiment. The reason for this behavior can be found in the symmetry of the function and in the orthogonal array itself. During the first nine experiments, there are only a few variations in the settings. From experiment 10 to experiment 27 each level occurs exactly two times. Since all variables have the same values at the same levels, changes in levels make no difference from one experiment to another. Due to the quadratic function, the lower and the upper level will have no different effect on the function value.

The mean curve for this simulation is the right picture in Figure 2.11 which remains at a value of six from the ninth experiment on. Due to the balancing properties of orthogonal arrays, the function mean is estimated exactly. This might be accidental and has to be investigated with further simulations.

The behavior of the standard deviation is closely related to the function values. In the first case, the first function evaluation gives the highest possible output value of nine (in the first and third case) at the chosen settings, and the second experiment gives the lowest one, which is zero, the standard deviation has its highest value after two experiments. Each additional experiment then contributes to a decrease in the standard deviation. The differences between the first and the third case, which presents the lower curve in Figure 2.12, is extremely small due to the choice of the quadratic function.

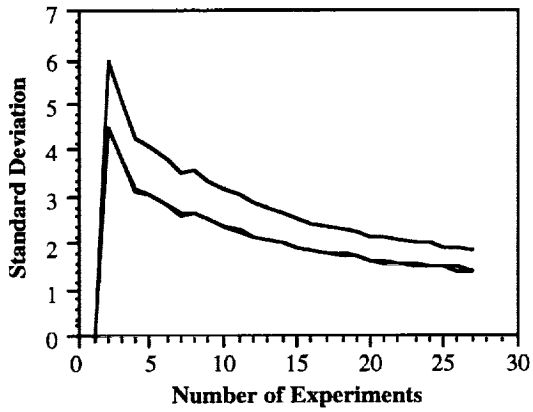


Figure 2.12 - Standard Deviation for Orthogonal Arrays

From the results we can see that use of an orthogonal array to estimate the function mean provides a good estimation even for this function with a skewed output. Due to the combinations in the orthogonal array, the standard deviation is underestimated and far away from the analytical one or the one calculated with a Monte Carlo simulation. It is much better than the Taylor series expansion method.

Generally, the use of orthogonal arrays seems to be applicable to simulate variations of a system output due to variations of the input parameters and other noise. The results are encouraging but further studies are necessary to see

- if the results are system dependent, and
- if we can identify the right settings of factor levels.

Analysis of variance is introduced in the following section. ANOVA provides a tool to analyze the results obtained from the simulation and to determine factor contributions in the variation of the output.

2.5 INTRODUCTION TO ANALYSIS OF VARIANCE

In the preceding, a full factorial design was replaced by a less expensive and faster method: the partial or fractional factorial design. Taguchi has developed orthogonal arrays for his factorial design. Partial experiments are only a sample of the full experiment. Hence, the analysis of the partial experiments not only has to include estimations of the statistics but also an analysis about the confidence of these results.

Analysis of variance (ANOVA) is routinely used and provides a measurement for confidence with respect to factor influences. We use it as the statistical method to interpret experimental data. This technique rather determines the variability (variance) of the data than analyzing the data itself. Some basic information about analysis of variance can be found in many statistic books; e.g., in [Dunn and Clark, 1987; Sokal and Rohlf, 1981; Casella and Betget, 1990]. Analysis of variance using orthogonal arrays is explained in greater detail in [Taguchi, 1987; Roy, 1990; Ross, 1988].

The analysis of the results based on experiments with orthogonal arrays is primarily to answer the following three questions:

- What is the optimum condition with respect to quality?
- Which factors contribute to the results and by how much?
- What will be the expected result at the optimum condition?

In our study, we are most concerned about the first two questions. When we deal with noise factors we can calculate the contribution of the factors to the output response or performance. Although ANOVA is basically developed to study control factors, we apply the same technique on a lower level to study the effects and contribution of noise factors to the variation of the output.

ANOVA is an analysis tool for the variance of controllable (signal) and uncontrollable (noise) factors. By understanding the source and magnitude of variance, better operating conditions can be predicted. But can we also predict worse conditions; e.g., the nominal output is on target but the variance is higher compared to other settings? ANOVA is a decision tool for detecting differences in the performance of a system.

2.5.1 ANOVA Notations

There are many quantities calculated in the analysis of variance, such as sum of squares, degrees of freedom, or variance ratio. All of these quantities are organized in a standard tabular format. We use the following notation:

SS = Sum of squares (factor, error, total),
SS' = Pure sum of squares,
f = Degrees of freedom,
V = Variance/mean squares,
F = Variance ratio,
T = Total sum (of results),
P = Percent contribution,
CF = Correction factor,
N = Number of experiments/trials.

2.5.2 ANOVA Terms, Equations, and Example

Sum of Squares

The sum of squares is a measure of the deviation from the mean value of the data. The effects of factors and interactions can be estimated by summing the squares of the effects at various levels and averaging over the number of degrees of freedom. The sum will be the total variation. Thus, the total sum of squares is calculated from

$$SS_T = \sum_{i=1}^{N} (y_i - \overline{T})^2 \quad , \tag{2.12}$$

where $\overline{T} = T / N$ is the mean value of the total result T and N is the number of experiments. This expression is mathematically equivalent to

$$SS_T = \sum_{i=1}^{N} y_i^2 - \frac{T^2}{N} = \sum_{i=1}^{N} y_i^2 - CF \quad . \tag{2.13}$$

Note that the expression T^2/N is called the correction factor CF. It presents a correction in the magnitude of the sum of squares of the mean.

When we perform experiments with orthogonal arrays, we have an equal number *n* of experiments at each of the *k* levels (1, 2, ...*k*) for one factor. The sum of squares—e.g., for factor A on levels A_i—is the sum of all level variations and can be calculated as

$$SS_A = \sum_{i=1}^{k_A} \frac{A_i^2}{n_A} - CF \quad , \tag{2.14}$$

and since n_A is constant, it becomes

$$SS_A = \frac{A_1^2 + A_2^2 + \ldots + A_{k_A}^2}{n_A} - CF \quad . \tag{2.15}$$

In the same way, we calculate the sum of squares for each of the involved factors and use the notation SS_A, SS_B, .. , SS_i for i factors A, B, .. , etc.

The error sum of squares can now be calculated as

$$SS_e = SS_T - \sum_i SS_i \quad , \tag{2.16}$$

which is the total sum of squares minus the sum of squares of all factors and interactions. We use a simple example to clarify Equations (2.12 to 2.16). In Table 2.4, the orthogonal array L_8 is presented which has seven columns and eight experiments (trials).

Table 2.4 - Orthogonal Array L_8

	Column no.						
Trial no.	**1**	**2**	**3**	**4**	**5**	**6**	**7**
1	1	1	1	1	1	1	1
2	1	1	1	2	2	2	2
3	1	2	2	1	1	2	2
4	1	2	2	2	2	1	1
5	2	1	2	1	2	1	2
6	2	1	2	2	1	2	1
7	2	2	1	1	2	2	1
8	2	2	1	2	1	1	2

From this array we use the first three columns to study the effects of factors A, B, and C. In Table 2.5, these three columns along with results y are shown. Each factor has two levels (k = 2), and the number of experiments at one level is four (n_i = 4). The y data values assigned to the last column are artificially chosen.

Table 2.5 - Two-Level Experiment Analysis with Three Factors

	A	**B**	**C**	
	Column no.			
Trial no.	**1**	**2**	**3**	**y data**
1	1	1	1	7
2	1	1	1	4
3	1	2	2	8
4	1	2	2	10
5	2	1	2	3
6	2	1	2	5
7	2	2	1	9
8	2	2	1	5

Using this table, we calculate the correction factor for the total sum of squares and the factor sum of squares. The error sum of squares is calculated last. These calculations are as follows:

$$CF = (7+4+8+10+3+5+9+5)^2/8 = \frac{51^2}{8} = 325.125,$$

$$SS_T = 7^2+4^2+8^2+10^2+3^2+5^2+9^2+5^2 - 325.125 = 43.875,$$

$$SS_A = \frac{(7+4+8+10)^2+(3+5+9+5)^2}{4} - 325.125$$
$$= 331.25 - 325.125 = 6.125,$$

$$SS_B = \frac{(7+4+3+5)^2+(8+10+9+5)^2}{4} - 325.125$$
$$= 346.25 - 325.125 = 21.125, \text{ and}$$

$$SS_C = \frac{(7+4+9+5)^2+(8+10+3+5)^2}{4} - 325.125$$
$$= 325.25 - 325.125 = 0.125.$$

Using Equation (2.16), we obtain the value for the error variation as

$$SS_e = SS_T - SS_A - SS_B - SS_C$$
$$= 43.875 - 6.125 - 21.125 - 0.125 = 16.5.$$

From the results we see that factor B has the highest influence on the variance of the output. Factor C has almost no influence on the variation. Also, a large variation is assigned to the error. Note that this example is not a typical textbook example since the error term is large. We have used only three of seven possible columns of OA L_8. If we use more columns to study interactions between factors, the error term will decrease. At the moment we have neglected these interactions.

Degrees of Freedom

A degree of freedom (DOF) in a statistical sense is associated with each piece of information that is estimated from the data. We are interested in independent comparisons between factor levels. A three-level factor contributes two degrees of freedom because we are interested in two comparisons. If we take one factor at any level A_1, we want to know the change in response compared to level A_2 and A_3. In general, the number of degrees of freedom associated with a factor is equal to one less than the number of levels for this factor. This concept of independent comparisons also applies to the degrees of freedom associated with the error estimate. If we have five experiments or data points, point 1 can be compared to point 2, point 2 to point 3, point 3 to point 4, and point 4 to point 5. There are four independent comparisons in this data set. A comparison of points 1 and 3 is not independent since it depends on the comparison of points 1 to 2.

The total number of degrees of freedom of a result T is calculated as

$$f_T = (\text{total number of experiments}) - 1. \tag{2.17}$$

The OA L_8 as shown in Table 2.3, for example, with three two-level columns has a total of seven DOF or one for each column. When we assign only three factors to the columns as in Table 2.4, the degree of freedom for the error is defined as

$$\begin{aligned} f_e &= f_T - f_A - f_B - f_C \\ &= 7 - 1 - 1 - 1 = 4 \quad , \end{aligned} \tag{2.18}$$

where

f_A = number of levels of factor A - 1 = 1,
f_B = number of levels of factor B - 1 = 1, and
f_C = number of levels of factor C - 1 = 1.

Hence, for an orthogonal array with eight experiments we cannot have more than seven columns or factors. With seven factors assigned, there is no DOF left to estimate the error.

Variance

The variance of each factor is calculated as sum of squares of this factor divided by degrees of freedom for this factor. For two factors A and B, we have

$$\begin{aligned} V_A &= SS_A/f_A \quad \text{(for factor A)}, \\ V_B &= SS_B/f_B \quad \text{(for factor B), and} \\ V_e &= SS_e/f_e \quad \text{(for error).} \end{aligned} \tag{2.19}$$

Variance Ratio and Pure Sum of Squares

The variance ratio F is the variance of the factor divided by the error variance. It is also called the F-test and was named by Sir Ronald Fisher who invented the ANOVA method. This tool provides a decision at some confidence level if a factor contributes to the sum of squares. This ratio is used to measure the significance of the investigated factor with respect to the variance of all factors included in the error term. For factors A, B, and the error, the F values are calculated as

$$\begin{aligned} F_A &= V_A/V_e \, , \\ F_B &= V_B/V_e \, , \text{ and} \\ F_e &= V_e/V_e = 1. \end{aligned} \tag{2.20}$$

Statisticians have worked out the expected distribution of this statistic, which is called the F-distribution. The F value obtained is then compared with the values from the F-tables for a given or desired level of significance. If the computed F value is less than the value determined from the F-tables at the selected level of significance, the factor does not contribute to the sum of squares within the confidence level. These tables are available in most handbooks of statistics; e.g., in [Roy, 1990; Ross, 1988]. A more detailed description can be found in [Sokal and Rohlf, 1981].

A simple example is used to illustrate the F value. If we substitute V_A by SS_A/f_A and assume f_A to be four, we obtain

$$F_A = SS_A/4V_e .$$

If the effect of A is negligible, SS_A should be about four times V_e. It also tells us that there are about four error variances in S_A. Therefore, the true effect of A, which is the pure sum of squares S_A', can be estimated from

$$SS_A' = SS_A - 4V_e .$$

In general, the pure sum of squares is the sum minus the degrees of freedom times the error variance. Factors A, B, and the error are calculated as

$$\begin{aligned} SS_A' &= SS_A - f_A V_e , \\ SS_B' &= SS_B - f_B V_e , \text{ and} \\ SS_e' &= SS_e + (f_A + f_B)V_e . \end{aligned} \tag{2.21}$$

Percent Contribution

If we want to know how much the variation SS_T is caused by the effect of one factor, we divide the pure sum of squares by the total sum of squares SS_T. The factor P gives the contribution in percent and is obtained by

$$\begin{aligned} P_A &= 100\, SS_A'/SS_T , \\ P_B &= 100\, SS_B'/SS_T , \text{ and} \\ P_e &= 100\, SS_e'/SS_T . \end{aligned} \tag{2.22}$$

These quantities are organized in an ANOVA table. A typical table is presented in Table 2.6. The values are calculated from the example shown in Table 2.5, where the sum of squares is explained.

Table 2.6 - Typical Analysis of Variance Table

Source	f	SS	V	F	SS'	P
A	1	6.125	6.125	1.485	2	4.56
B	1	21.125	21.125	5.121	17*	38.75
C	1	0.125	0.125	0.030*		
Error	4	16.5	4.125	1		
Total	7	43.875	-	43.875		100%

*At least 97.5% confidence.

Only factor B contributes to the variance of the output with a confidence level of at least 97.5%. The variation due to factors A and C is small compared to the error. Their confidence level is below 90% and is not listed. In Table 2.5, we see that each factor has a DOF of one while the error has four DOF. Therefore, only for the error the variance differs from the sum of squares. The calculated F value has significance only for factor B as stated above. The error is too large compared to the contribution of factor C. Hence, we do not calculate the SS'_C.

These equations and the accompanying notation provide basic information about analysis of variance and how it is applied to orthogonal arrays. More information can be found in the references mentioned.

2.6 WHAT HAS BEEN PRESENTED AND WHAT IS NEXT?

In this chapter we have presented the following information and concepts:

- Quality characteristics of a product/process and how to measure quality loss due to the effect of noise factors.
- Orthogonal arrays and orthogonal array based simulation as a technique to get dependable information about factors with less effort, cost, and time.
- A statistical method to interpret experimental data and factor effects.

We have identified our approach. We next need a simulation model to apply the suggested simulation technique. Given the background derived in this chapter, the answers to the questions posed in Section 1.4 become clearer. Is simulation based on orthogonal arrays an appropriate alternative to Monte Carlo simulation? The search for an answer to this question involves many tasks with respect to quality, confidence, available information, and design decisions. In the next chapter, we develop the analysis model for the LifeSat vehicle in order to perform the necessary tasks using this model.

PAGE _____ INTENTIONALLY BLANK

CHAPTER 3

THE LIFESAT SPACE VEHICLE MODEL

Our main case study, as described in Section 1.1.1, is the trajectory simulation of the LifeSat space vehicle. The vehicle is supposed to follow a predetermined trajectory defined by the initial state at entry interface and to land within the desired target area. In Section 2.1.2, we have described the classification of factors as signal, control, and noise factors. Noise factors will cause a deviation from the trajectory and, hence, a deviation form the target. In this chapter, the LifeSat model is derived in detail. The analysis model is discussed in Section 3.1 including all mathematical relationships concerning gravity, aerodynamic forces, atmosphere density, etc. In Section 3.2, we identify the noise factors and their dispersions. We separate them into factors with uniform and normal distributions. The implemented model is validated in Section 3.3. Next this model will be used in Chapter 4 to answer the questions posed in Section 1.4.

3.1 THE ANALYSIS MODEL FOR THE LIFESAT SPACE VEHICLE

To simulate the trajectory of the LifeSat space vehicle, we need an analytical model that describes the forces applied to the vehicle. We further have to model the atmospheric density and also to determine the performance parameters used to evaluate the trajectory. This is done in Sections 3.1.2 and 3.1.4. Information about the LifeSat model is obtained from [Tigges, 1992]. In Section 3.1.1, the nomenclature used in the equations is given; and in Section 3.1.3, we explain the coordinate systems used in our simulations.

3.1.1 Nomenclature

In the LifeSat model, the following nomenclature is used for variables, parameters, and constants where vectors are shown "bold":

A_{ref}	reference area (m^2)
c_d	drag coefficient
c_l	lift coefficient
$\mathbf{e}_r$	unit vector for relative velocity
$\mathbf{e}_R$	unit vector for position
$\mathbf{e}_d$	unit vector for drag
$\mathbf{e}_l$	unit vector for lift
$\mathbf{F}_{TOTAL}$	vector of total force applied to the vehicle (N)
$\mathbf{F}_{GRAVITY}$	vector of gravity force (N)
$\mathbf{F}_{AERO}$	vector of aerodynamic forces (N)
g	acceleration of gravity (m/s^2)
h	height, altitude (m)
h_0	standard height (m)
h_s	reference height (m)
J_i	Performance parameter
m	vehicle mass (kg)
M_e	mass of the Earth (kg)
M	mole mass (kg/kJ)
r, R	distance or radius from coordinate origin (m)
R_0	Earth radius (m)
v_r	magnitude of relative velocity (m/s)
$\mathbf{V}_r$	vector of relative velocity (m/s)
κ	gravitational constant ($m^3/(kgs^2)$)
μ	product of the gravity constant κ, and the mass of the Earth M_e (m^3/s^2)
ρ_0	standard air density (kg/m^3)
ρ	air density (kg/m^3)
φ, Θ	angles in polar coordinate system
θ	rotation angle (rad)
$\dot{\theta}$	angle of velocity (rad/s).

3.1.2 A Model of Gravitational and Aerodynamic Forces

The LifeSat space vehicle is modeled by simple functional relationships. The motion of the vehicle is only driven by forces not coming from the vehicle itself. Two forces are identified; namely,

- gravity, and
- aerodynamic force.

The applied force on the vehicle is a combination of the gravitational force and aerodynamic force. In vector form, the equation for the total force is given by

$$\mathbf{F}_{TOTAL} = \mathbf{F}_{GRAVITY} + \mathbf{F}_{AERO} \ , \tag{3.1}$$

where

$$\mathbf{F}_{GRAVITY} \cong -m\frac{\mu}{R^2}\mathbf{e}_R \ . \tag{3.2}$$

Gravity has the opposite direction from the radius vector, pointing from the coordinate origin (coordinate systems are explained in Section 3.1.3) to the vehicle's center of mass. The Cartesian coordinate system used is shown in Figure 3.1 (Sec. 3.1.3). In Equation (3.2), we have $\mu = gR_0^2 = 3.98*10^{14}\ m^3/s^2$, the product of acceleration of gravity and radius of the Earth squared. For our model we use values of

- $g = 9.81\ m/s^2$, and
- $R_0 = 6370$ km.

The constant μ is also the product of gravitation constant κ and the mass of the Earth M_e; namely, $\mu = \kappa M_e$. The *gravitational force* decreases proportionally to the radius squared.

The *aerodynamic force* is the sum of drag forces and lift forces, both having different direction vectors. The direction of the drag force is opposite to the velocity. In Equation (3.3), the product $0.5\rho v_r^2$ is called the dynamic pressure and is a measure of pressure due to velocity. The aerodynamic force is calculated from

$$\mathbf{F}_{AERO} = \frac{1}{2}\rho v_r^2 A_{ref}\left[c_d\mathbf{e}_d + c_l\mathbf{e}_l\right] \ . \tag{3.3}$$

In Equation (3.3), $\mathbf{e}_d$ and $\mathbf{e}_l$ are unit vectors for drag and lift and are defined as

$$\mathbf{e}_r = -\mathbf{e}_d = \frac{\mathbf{V}_r}{|\mathbf{V}_r|} \ , \tag{3.4}$$

and

$$\mathbf{e}_l = \mathbf{e}_1\cos\theta + \mathbf{e}_2\sin\theta \ . \tag{3.5}$$

The vehicle is stabilized through rotation. The vectors $\mathbf{e}_1$ and $\mathbf{e}_2$ are two orthogonal vectors of the vehicle's coordinate system. The third orthogonal component is the

velocity vector $\mathbf{e}_r$. In Equation (3.5), θ is the rotation angle. The nominal rotation rate is given in Equation (3.6) as

$$\theta = \theta_0 + \dot{\theta}\Delta t \quad , \tag{3.6}$$

where θ_0 is the initial value of the rotation angle and the angle of velocity, $\dot{\theta}$, for the vehicle rotation is given by

$$\dot{\theta} = \frac{25°}{\text{sec}} \quad . \tag{3.7}$$

In Equations (3.2) and (3.3), μ is the product of natural constants. The parameters m, c_d, c_l, and A_{ref} are vehicle parameters. The relative velocity v_r and the radius R are calculated from state vectors $\mathbf{V}_r$, $\mathbf{e}_R$ for speed and position. To model the trajectory we need coordinate systems which describe both the initial state and the state during the flight simulation. This is explained in Section 3.1.3.

3.1.3 Coordinate Systems

The state vector of the vehicle is used to describe the position and velocity of the vehicle. Since we are interested in the state relative to the Earth, we use coordinate systems with origins fixed at the center of the Earth. For this purpos,e we use the *Cartesian* and the *polar* coordinate systems. The coordinate systems are shown in Figure 3.1. The Cartesian coordinate system is defined by the orthogonal components x, y, and z; the radius r and the angles φ and Θ define the polar coordinate system. If the radius r is on the Earth's surface, it is denoted by R_0.

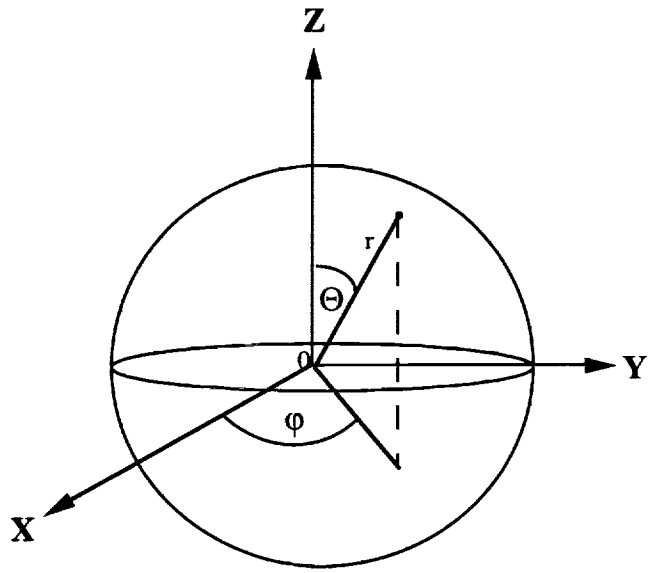


Figure 3.1 - Cartesian and Polar Coordinate System

Given a point at distance r from the origin, we measure the angle φ between the x-axis and the projection of r into the x-y plane. The angle Θ is measured between the z-axis and r. Additional information can be found in [Bronstein and Semendjajew, 1987].

The conversion of polar coordinates into Cartesian coordinates is given by the following equations as

$$x = r \cos\varphi \sin\Theta ,$$

$$y = r \sin\varphi \sin\Theta , \text{ and}$$

$$z = r \cos\Theta . \tag{3.8}$$

The conversion from Cartesian to polar coordinates is calculated from

$$r = \sqrt{x^2 + y^2 + z^2} ,$$

$$\varphi = \arctan (y/x) , \text{ and}$$

$$\Theta = \arctan \frac{\sqrt{x^2 + y^2}}{z} = \arccos \frac{z}{\sqrt{x^2 + y^2 + z^2}} . \tag{3.9}$$

There are several options that will characterize the position and velocity state vector for the trajectory of the entering vehicle. Some of these possibilities are listed in Table 3.1 where each group represents a different alternative with respect to position and velocity [McCleary, 1991].

Table 3.1 - State Descriptions for Position and Velocity

POSITION	VELOCITY
Longitude Geodetic latitude Altitude	Inertial velocity magnitude Inertial flight path angle Inertial azimuth
Longitude Declination Radius magnitude	Relative velocity magnitude Relative flight path angle Relative azimuth
Longitude Geocentric latitude Altitude	x, y, z components
x, y, z components	

The initial state is characterized by initial positions and initial velocity using one of the options of Table 3.1. In tables obtained from NASA [McCleary, 1991], the entry interface state or initial state is defined by

- longitude (deg), geodetic latitude (deg), and altitude (m) for position, and
- inertial velocity (m/s), inertial flight path angle (deg), and inertial azimuth (deg) for velocity.

Therefore, we use the same descriptors for the initial state in our model. Calculations are performed most easily in the Cartesian coordinate system; hence, we make a coordinate system transformation. Note that there should be no confusion about the use of the words *inertial* and *initial*. The inertial state for velocity is transformed into the initial state in the fixed Cartesian coordinate system. Therefore we, prefer to use *initial*.

The value of *longitude* corresponds exactly with the value of φ in the polar coordinate system. The *geodetic latitude* has a value of 0 deg at the equator of the Earth and 90 deg at the North Pole. The value can be transformed into Θ using $\Theta = 90$ deg minus geodetic latitude. The *altitude* is the difference between the radius r and radius of the surface of the Earth; hence, $r = R_0$ plus altitude.

To make a transformation of the velocity state vector we first must explain the terminology for flight path angle and azimuth. As can be seen in Figure 3.2, the flight path angle is the angle γ between the velocity vector and a parallel to the surface line.

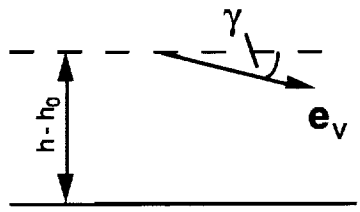


Figure 3.2 - Illustration of the Flight Path Angle γ

The angle azimuth α, which is depicted in Figure 3.3, is the angle which is measured between the longitude and a projection of the velocity vector onto the surface plane or a plane that is parallel to it.

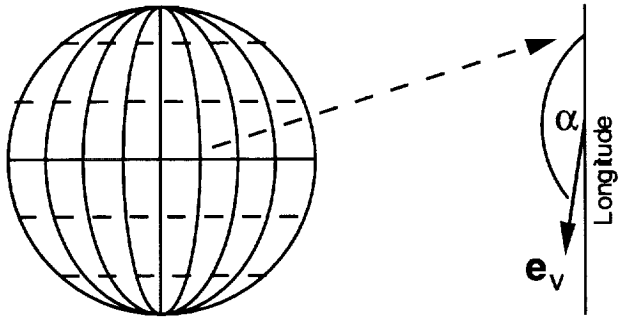


Figure 3.3 - Illustration of the Azimuth Angle α

The magnitude of the relative velocity v_r can be transformed into components of the velocity in the Cartesian coordinate system v_x, v_y, and v_z using angles γ and α and the angles φ and Θ of the polar coordinate system. In Equation (3.10), the relationships are described as

$$v_x = v_r\left[\cos\varphi(\sin\Theta\sin\gamma - \cos\Theta\cos\gamma\cos\alpha) + \sin\varphi\cos\gamma\sin\alpha\right] ,$$

$$v_y = v_r\left[\sin\varphi(\sin\Theta\sin\gamma - \cos\Theta\cos\gamma\cos\alpha) - \cos\varphi\cos\gamma\sin\alpha\right] , \text{ and}$$

$$v_z = v_r\left[\sin\Theta\cos\gamma\cos\alpha + \cos\Theta\sin\gamma\right] . \qquad (3.10)$$

When we know the x, y, and z components for position and velocity, it is much simpler to calculate the trajectory of the vehicle. At the end of the flight we are interested in the position given in longitude and geodetic latitude. For this purpose, we re-transform the position components back into these parameters.

Another parameter of Equation (3.2) has to be calculated; namely, the atmospheric density. The model of this parameter is explained in Section 3.1.4, and performance parameters for the vehicle are introduced in Section 3.1.5.

3.1.4 The Atmosphere Model

The atmospheric density is one parameter which has not yet been described and modeled. It is a necessary parameter for the calculation of the aerodynamic forces and largely determines the trajectory of the entering vehicle.

Originally, the ***atmosphere*** model is generated by the **G**lobal **R**eference **A**tmosphere **M**odel (GRAM-88) to be used for the LifeSat model. This model is a function of time, altitude, longitude, date, and solar activity. Planetary atmospheres can be assumed as a first approximation to have exponential density-altitude distributions. The exponential atmosphere, as shown in Equation (3.11), is a simplified approximation. The ratio of actual density ρ to standard density ρ_0 is

$$\frac{\rho}{\rho_0} = e^{-\beta \Delta h} \quad , \tag{3.11}$$

where $\beta = -(1/\rho)(d\rho/dh) = Mg/RT_0$. The integration of this differential equation then results in Equation (3.11). We use the following parameters and constants for the calculation:

- the mole mass M of air, which is approximately M = 29 kg/kmole,
- the universal gas-constant defined as R = 8.314 Joule/(mole K), and
- the reference temperature of T_0 = 293 K.

We calculate the standard density ρ_0 from $\rho_0 = Mp_0/(RT_0) = 1.2$ kg/m^3, where $p_0 = 1.013$ bar is the standard pressure. Using the previous g value, we obtain $\beta = 0.0001168$ m^{-1}. If we denote $h_s = 1/\beta = 8.563$ km as the reference height and reformulate Equation (3.11), we obtain Equation (3.12) as

$$\rho \approx \rho_0 e^{\frac{-(h-h_0)}{h_s}} \quad . \tag{3.12}$$

This equation is based on the assumption that g is constant over the whole range from h_0 to h. In fact this does not hold true. At a height of approximately 120 km above the surface (altitude), which we use as the height of the entry interface of the vehicle, g is about 3.5% smaller than the initial value at the surface. If we compare this to documented standard values in engineering handbooks (e.g., [Marks, 1986]), we have some deviations in our calculated values which are caused by approximations. For example, the standard density is documented as $\rho_0 = 1.225$ kg/m^3 and β as 0.0001396 m^{-1}. These differences will not affect our study since our aim is to show the working of a method and not accuracy in modeling.

3.1.5 Performance Parameter

Along with path values, several so-called ***performance parameters*** can be calculated which provide some information about the impacts on the vehicle. These parameters are

- J_1 = Peak (g - load/acceleration),
- J_2 = Peak ($0.5\rho v^2$; dynamic pressure),
- J_3 = Peak (heat-rate),
- J_4 = Δ-Range - target, and
- J_5 = Mach @ chute deploy.

Of primary interest are forces and the dynamic pressure. Acceleration, rather than forces, provides a value independent from the mass and is used as a measure of performance. After the vehicle has entered the atmosphere, gravity accelerates the vehicle towards the Earth. At the height of the entry interface at about 120 km altitude, the atmospheric density is approximately 8.65×10^{-7} kg/m^3 which is very low. The acceleration of gravity is about 9.45 m/s^2. The density increases along the trajectory with decreasing altitude of

the vehicle. The vehicle enters the atmosphere with an initial speed of nearly 10 km/s, but increasing atmospheric density slows the speed. The acceleration is also called g-load as a multiple of the acceleration of gravity. During the simulation of the model, the magnitude of the g-load is calculated along the path and the maximum value is stored. The maximum or ***peak g-load*** is denoted as *performance parameter* J_1.

The dynamic pressure, *performance parameter* J_2, is calculated using the product of density and the speed squared as $J_2 = 0.5\rho v_r^2$. It is a measure of the pressure due to velocity but is independent from the vehicle parameters. Again we calculate this parameter along the trajectory and store the largest value, the ***peak dynamic pressure***. Obviously, there is a strong relationship between J_1 and J_2 since both are part of Equation (3.2).

Performance parameter J_3, the ***heat rate***, is computed and as before the maximum value can be stored. The heat rate is measured as the peak heating of the vehicle per square meter per second.

The last two performance parameters, which are denoted by J_4 and J_5, are the ***Mach number*** at drogue chute deploy and the ***Δ-range*** of the target area. The Δ-range represents the deviation from target position measured in the directions of geodetic latitude and longitude.

All of these performance parameters indicate a designer's interest in the landing position of the vehicle and also other impacts on the vehicle. In the design of the vehicle, these factors have to be taken into account.

In the design of the vehicle, there are also tolerances on the dimensions and parameters of the vehicle which have to be taken into account. Vehicle tolerances, such as tolerances on the mass or the drag coefficient, cause a variation in performance and trajectory and, hence, a loss in quality when the target is not achieved. These tolerances, also called dispersions, are noise factors according to the notation of Section 2.1.2. The exact value of each vehicle parameter is not in control of the designer. Besides vehicle tolerances/ dispersions there are environmental dispersions. Atmospheric conditions vary and also the initial state of the vehicle varies within some tolerances. In Section 3.2, all of the different dispersion affecting the performance and the trajectory of the vehicle are explained in detail.

3.2 DISTRIBUTIONS OF THE LIFESAT MODEL PARAMETERS

In Section 1.3, we have explained the distribution of many natural parameters which can either be assumed to be uniformly distributed or normally distributed. In the following, we also use the word *dispersion* similar to distribution since this is the terminology used by NASA [McCleary, 1991]. The LifeSat vehicle model includes six dispersed vehicle and environmental parameters. In general, each variable/parameter can be assumed to be dispersed. All input parameters used in the model are assumed to be either normally or uniformly distributed. These are the following parameters:

- vehicle mass,
- angle of attack,

- initial state,
- aerodynamic coefficients,
- atmospheric density, and
- winds.

The preceding six dispersed parameters basically represent groups of parameters. The initial state, for example, is based on another six dispersed parameters: three for the initial position and three for the initial velocity.

Some of the parameters are dependent of the values of other parameters and have to be dispersed with respect to these values. The vehicle mass contains only one parameter whereas the aerodynamic coefficients imply coefficients for lift and drag, and for vehicle and parachutes. Hence, the final number of dispersed parameters will be greater than six.

As described in Section 1.3, a dispersed parameter can be represented by its mean μ and the standard deviation σ. For our purpose we use the 3σ value because this is widely accepted to be the value for tolerances. It is common practice that this value is given in percent of the mean value. If the mean value is $\mu = 0$, we need the absolute magnitude of the dispersion.

In Section 3.2.1, we explain the dispersion of uniformly distributed parameters; and in Section 3.2.2, we explain the normally distributed parameters.

3.2.1 Uniformly Distributed Parameters

According to the convention of Section 1.3, the distribution of uniformly distributed parameters is described by the mean μ and the tolerance Δ. The parameter can have values between $\mu \pm \Delta$. The uniformly dispersed parameters in the model are

- the vehicle mass,
- winds, and
- the angle of attack.

VEHICLE MASS – is uniformly dispersed around its mean value. There is no correlation or dependency with other parameters. Mean and tolerance are

μ = 1560.357 kg = 3440 lb, and

Δ = 78.018 kg = 172 lb, which is equivalent to 5%.

A higher mass than the mean value increases the gravity forces on the vehicle. Therefore, we expect a shorter flight time and higher values for performance parameter J_1, J_2, and J_3. The landing position for the latitude will be closer to the initial value at entry interface.

WINDS – are assumed to be uniformly dispersed and have a major effect when parachutes are deployed. They cause a shift of the vehicle parallel to the surface in a mainly east-west-direction or vice versa. In our simulation, we have neglected this factor and, hence, are not concerned about the variations of wind speed and direction. In Section 3.3,

we explain in more detail the choice of parameters for the simplified implemented model. Mean and tolerance are not of interest. There are some consequences following this exclusion which are explained in Section 3.3, too.

ANGLE OF ATTACK α – is uniformly dispersed around its mean value. The angle is assumed to be constant during the duration of the flight. Mean and tolerance are

μ = 0 deg, and
Δ = 5 deg.

The aerodynamic vehicle coefficients c_d and c_l are dependent on this angle whereas all other aerodynamic coefficients are independent of the angle. The coefficients c_d and c_l are normally dispersed; they will be explained in the following section. In Table 3.2, all related values between angle of attack and aerodynamic coefficients can be found. The listed values are the mean values of the normally distributed aerodynamic coefficients.

Table 3.2 - Angle of Attack Related Aerodynamic Coefficients

AERODYNAMIC COEFFICIENT	ANGLE OF ATTACK α (deg)		
	-5.0	0.0	+5.0
Vehicle C_d	0.67068	0.66512	0.67068
Vehicle C_l	-0.04204	0.0	0.04204
Drogue Chute C_d	0.55	0.55	0.55
Main Chute C_d	0.8	0.8	0.8

If we assume a linear relationship between mean values of the aerodynamic vehicle coefficients c_d and c_l and the angle of attack α, we model them as

$$c_d = | 0.66512 + 0.001112\,\alpha | \qquad (3.13)$$

and

$$c_l = 0.008408\,\alpha\ . \qquad (3.14)$$

The angle of attack is uniformly dispersed, and the calculated aerodynamic coefficients of the vehicle at any value are the mean values for their normal distribution. In Figure 3.4, we can see three curves for the normally dispersed drag coefficients for angle of attack values of α = -5, 0, and 5 deg.

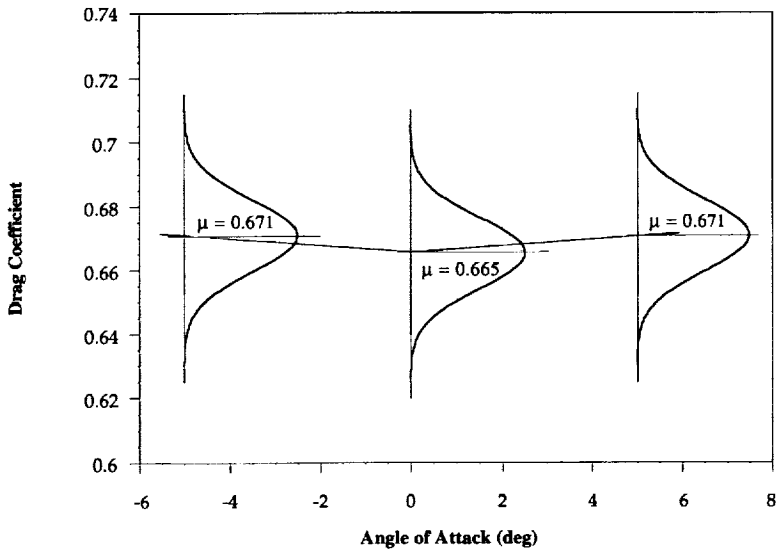


Figure 3.4 - Relation of Angle of Attack and Drag Coefficient

The distributions are probability density curves, and the height does not correspond to a value of the angle of attack. As the mean values of the aerodynamic coefficients vary, the value for their standard deviation also varies according to the mean. In Section 3.2.2, the 3σ value is defined as 5% of the mean value.

3.2.2 Normally Distributed Parameters

The distribution of normally distributed parameters is described by the mean μ and the standard deviation σ. The parameter can have any value, but the probability decreases further away from the mean. As shown in Section 1.3, 99.73% of the area under the probability density curve is within a range of $\mu \pm 3\sigma$.

The normally dispersed parameters in the model are

- the aerodynamic coefficients and reference areas,
- the atmosphere, and
- the six components of the initial state vector.

AERODYNAMIC COEFFICIENTS – The c_d values are normally dispersed. In Section 3.2.1, we have seen that they are related to the angle of attack and are not independent. For the vehicle and the parachutes we have different values for c_d. In the model, only drag coefficients are dispersed.

The vehicle's ***reference area*** is not dispersed, but the reference area for the parachutes is dispersed with 1% around the mean value. The values for the mean and 3σ for drag coefficients reference areas (shown below) are separated for vehicle and parachutes.

The *vehicle* drag coefficient is dispersed by

μ = c_d as evaluated for a given angle of attack, and

3σ = 5% of the mean μ.

The *parachute* parameter are dispersed as follows:

The drogue chute drag coefficient is dispersed by

μ = 0.8, and

3σ = 0.04 (equals 5% of the mean μ).

The drogue chute reference area is dispersed by

μ_A = 51.24 m^2, and

$3\sigma_A$ = 0.512 m^2 (equals 1% of the mean μ).

The main chute drag coefficient is dispersed by

μ = 0.55, and

3σ = 0.0275 (equals 5% of the mean μ).

The main chute reference area is dispersed by

μ_A = 4.1 m^2, and

$3\sigma_A$ = 0.041 m^2 (equals 1% of the mean μ).

In Table 3.3, the values for vehicle and chute dispersions of the drag coefficient and the reference area are summarized.

Table 3.3 - Vehicle and Chute Dispersions for Aerodynamic Coefficients and Reference Area

DISPERSION	DISPERSION MAGNITUDE	
	$c_d(3\sigma)$	Ref. Area (3σ, m^2)
Vehicle	c_d (5%)	-
Drogue Chute	0.0275 (5%)	0.041 (1%)
Main Chute	0.04 (5%)	0.512 (1%)

In Equation (3.3) of Section 3.1.2, we use only one value for the reference area and the drag coefficient. Therefore, we finally normalize the drag coefficients to obtain a single value. According to the different stages during the flight (no chutes, drogue chute, main chute) as described in Section 1.1.1, the coefficients for vehicle and parachutes are normalized to be

$$c_d = \frac{c_{d1}A_1 + c_{d2}A_2 + \ldots + c_{dn}A_n}{A_1 + A_2 + \ldots A_n} \quad , \tag{3.16}$$

where n is the number of drag coefficients and A_i is the reference areas.

ATMOSPHERE – is normally dispersed and characterized by the density ρ_0. Originally, the dispersed atmosphere is generated by the Global Reference Atmosphere Model (GRAM-88). This is a function of time, altitude, longitude, date, and solar activity. The exponential atmosphere, as shown in Equation (3.12), is an extremely simplified approximation. A table of pressure, density, or temperature at a given altitude can be found in Baumeister [Baumeister, 1986] which gives more accurate values. Due to this approximation, we assume 30% (equals 3σ) variation in the atmospheric density. The dispersion is approximated by a normal dispersion and the mean value is the standard density ρ_0 at surface. At a given altitude, the mean is the value calculated at this height which uses Equation (3.12) and standard density ρ_0. In Section 3.1.4, we have calculated the standard density to be $\rho_0 = 1.2$ kg/m^3. This value is used for the simulation.

We disperse the atmospheric density at surface by

μ = 1.2 kg/m^3, and
3σ = 0.36 kg/m^3, which is equivalent to 30%.

The density at a given altitude h is dispersed by

$$\mu = \rho_0 e^{\frac{-(h-h_0)}{h_s}} \quad \text{and}$$

$$3\sigma = 0.3\rho_0 e^{\frac{-(h-h_0)}{h_s}} \quad .$$

During the flight, the dispersion of the density does not change; only the new magnitude is calculated for each iteration in the simulation. The use of the GRAM-88 model for the simulation is much more complicated and is briefly explained below using NASA terminology. The dispersed atmospheric density is computed as

$$\rho_2' = A\frac{\overline{\rho}_2}{\overline{\rho}_1}\rho_1' + Bq_1 \quad , \tag{3.15}$$

where

$$A = \frac{\sigma_2}{\sigma_1} e^{\frac{-r}{L}} ,$$

$$B = \sigma_2\left[1 - e^{-2r/L}\right]^{\frac{1}{2}} ,$$

ρ_1' = dispersed density at previous location,
ρ_2' = dispersed density at current location,
$\overline{\rho}_1$ = mean density at previous location,
$\overline{\rho}_2$ = mean density at current location,
σ_1 = standard deviation about mean at previous location,
σ_2 = standard deviation about mean at current location,
q_1 = normally distributed random number,
r = distance traveled from previous location, and
L = correlation scale length.

Basically, the calculation starts at the initial density value and the dispersed density is updated at each location/position. By using statistical information from the previous and the current location, the new value of the dispersed density can be calculated. This procedure is obviously very complicated and requires much more knowledge than does the exponential density model. Therefore, we have chosen Equation (3.12) to model the atmospheric density. Since the density is one of the largest contributors to performance variations, the large variation of 30% is justified to represent the "real" situation.

The **STATE** of the vehicle is described by three parameters for the position and by three parameters for the velocity. We also need a total of six parameters to identify the initial state. The various possibilities for the description of the state are shown in Section 3.1.3. Originally, the dispersion of these six parameters is given in a covariance matrix which includes correlation of each parameter with every other parameter. A transformation of the coordinate system used in this matrix to the systems described in Section 3.1.3 is necessary. We simplify these dispersions by assuming no correlation among the parameters.

The *Initial Position* - is defined by the parameters longitude (deg), geodetic latitude (deg), and altitude (m). There are two possibilities to establish dispersion in the three parameters. First, we transform the parameters just mentioned into the x, y, z coordinate system and then disperse each component either by an absolute value or by using a percentage of the mean value for 3σ. Second, we disperse each of the initial parameters in the same way as before and then make the coordinate transformation. The first method has the advantage of being able to determine the exact dispersions in the Cartesian coordinate system. The second method seems to be better to study the effect of variations in the initial position since it is related to angles instead of distances. We have used both methods in our study and also have used different mean values for the initial position. For the first method, we used the following dispersions of the x, y, z components:

$\mu_x = x_{initial}$	$\mu_y = y_{initial}$	$\mu_z = z_{initial}$
$3\sigma_x =$ 15000 m	$3\sigma_y =$ 15000 m	$3\sigma_z =$ 15000 m.

In this case, the dispersions are quite large (Sec. 4.1). As will be shown in Section 4.1, this method is used for initial studies of the model. In a second case, as used in Section 4.2, we define smaller values to be

$\mu_x = x_{initial}$	$\mu_y = y_{initial}$	$\mu_z = z_{initial}$
$3\sigma_x =$ 6000 m	$3\sigma_y =$ 6000 m	$3\sigma_z =$ 6000 m.

In the last case, we use the second method and hold the value for the altitude constant. This is discussed in Section 4.3. The 3σ variation of the longitude and the geodetic latitude are assumed to be absolute values and their initial values are set to obtain a target value near -106.65 deg longitude and 33.6 deg geodetic latitude. Dispersions and their magnitudes are shown in Table 3.4.

Table 3.4 - Initial State Dispersions for Position

DISPERSION	DISPERSION MAGNITUDE	
	μ	3σ
Altitude	6491920 m	-
Longitude	-106.65 deg	0.01 deg
Geo. Latitude	44.3 deg	0.1 deg

The *Initial Velocity* – can be dispersed in the same way as the initial position. We either disperse the x, y, z components of the initial velocity or disperse the magnitude of the initial velocity and the inertial angles of flight path angle and azimuth. Using the first method we define

$\mu_x = v_{xinitial}$ $\quad \mu_y = v_{yinitial}$ $\quad \mu_z = v_{zinitial}$

$3\sigma_x =$ 2% of mean $\quad 3\sigma_y =$ 2% of mean $\quad 3\sigma_z =$ 2% of mean.

In the second method, we keep a 2% dispersion for the magnitude of velocity and also disperse the angles by absolute values as shown in Table 3.5

Table 3.5 - Initial State Dispersions for Velocity

DISPERSION	DISPERSION MAGNITUDE	
	μ	3σ
Velocity	9946.5 m/s	199.0 m/s (2%)
Flight Path Angle	-5.88 deg	0.1 deg
Azimuth	180.0 deg	0.1 deg

In the preceding sections, we explained the analysis model of the LifeSat vehicle in Section 3.1 and the parameter distributions in Section 3.2. For the implementation, we simplify the model based on some assumptions. In Section 3.3, we describe its implementation and validation.

3.3 MODEL IMPLEMENTATION AND VALIDATION

In this section, we explain the assumptions and simplifications used in the implemented LifeSat model. We select the most important parameters for dispersions and validate the implementation of the model by running a simulation and observing its behavior.

3.3.1 Implementation of a Simplified LifeSat Model

Our implementation of the model is driven by a trade-off between completeness and sufficiency. To simplify the model, we make the following assumptions:

- Winds only have an influence when parachutes are deployed in the last section of the flight. They cause a bias of the landing position in wind direction. The effect (bias in landing position) of winds is dominant but is neglected in the simplified model.

- Aerodynamic forces due to the lift coefficient, c_l, equal out during the flight because of constant rotation. Any variation in the landing position caused by this factor is very small.

- Coriolis forces, which are fictitious forces due to rotation of the Earth, are not applied. They cause a bias in longitude and have a small effect based on dispersed parameters.

Based on these assumptions, we determine the dispersed parameters for the study. A set of contributions of error sources/factors to the output dispersion of longitude and geodetic latitude is given in [McCleary, 1991]. We select the following nine dispersed parameters from this set in the order of highest contributions:

- atmospheric density,
- vehicle drag coefficient,
- initial state, and
- vehicle mass.

The order is based on the data provided in [McCleary, 1991]. As explained in Section 3.2, the initial state includes six dispersed parameters. To simplify the model, we neglect the influence of winds and, hence, we neglect all parameters involved for the parachutes. The lift coefficient is also neglected. Using the nine parameters, the model is sufficiently accurate to show the application of the quality methods as introduced in Chapter 2. The model with nine dispersed parameters is implemented for simulation.

To assure a proper implementation we validate the model by performing a simulation based on the mean values of all parameters. The validation is done in Section 3.3.2.

3.3.2 Validation of Model Implementation

To validate the simulation we use initial state values which are chosen to be close to those documented by NASA. We expect the flight-time to be in the range of 280 to 330 seconds. We also expect the landing position for the latitude near 33.6 deg [McCleary, 1991]. The complete initial state values are shown in Table 3.6. When a flight simulation is done using only the mean values of each dispersed parameter we will call it the *nominal flight*. In Chapter 4 a nominal flight is always simulated before parameters are dispersed.

Table 3.6 - Initial State for Nominal Flight Simulation

Longitude	-106.65 deg
Geodetic Latitude	44.3 deg
Altitude	121.92 km
Initial Velocity	9946.5 m/s
Initial Flight Path Angle	-5.88 deg
Initial Azimuth	180 deg

After the simulation is finished, we obtain data about the landing position, the final velocity, and flight parameters. Of course, the final altitude is zero. The longitude remains at its initial value because we have neglected winds and Coriolis force. The final parameter values for our simulation model are shown in Table 3.7.

Table 3.7 - Flight Parameters and Final State for Nominal Flight Simulation

Longitude	-106.65 deg
Geodetic Latitude	33.71 deg
Altitude	0.0 km
Velocity	112.8 m/s
Flight-Time	302 sec
J_1	130.57 m/s^2
J_2	92995.6 kg/(ms^2)

The flight-time is within the given range and the geodetic latitude is close to the target. For performance parameter J_1, we calculate the corresponding g-load as $J_1/g = 130.57/9.81 = 13.3$. This value also fits with the documented ones, which lie between 11 and 15. Values for performance parameter J_2 are not documented and can not be compared now. But we get a first estimate of its magnitude.

We obtain some information about the flight trajectory from the following figures. In Figure 3.5, the altitude versus geodetic latitude is shown. We identify a nearly linear decrease in altitude with geodetic latitude until a value of about 40 km. Then the vehicle nearly drops down compared to the distance already flown since entry interface.

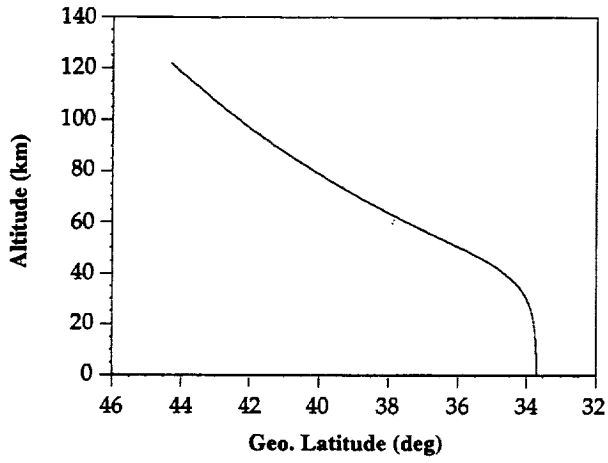


Figure 3.5 - Geodetic Latitude vs. Altitude

The increase in density seems to have a very high influence because all other parameters except gravity do not change during the flight. Since we have a major change in the trajectory at the "end" of the flight simulation, we are interested in when these changes occur. In Figure 3.6, the altitude and latitude are shown versus flight-time. The whole flight lasts about 300 sec; and although the altitude decreases gradually with each second, the geodetic latitude has almost reached its final state before half the time has passed.

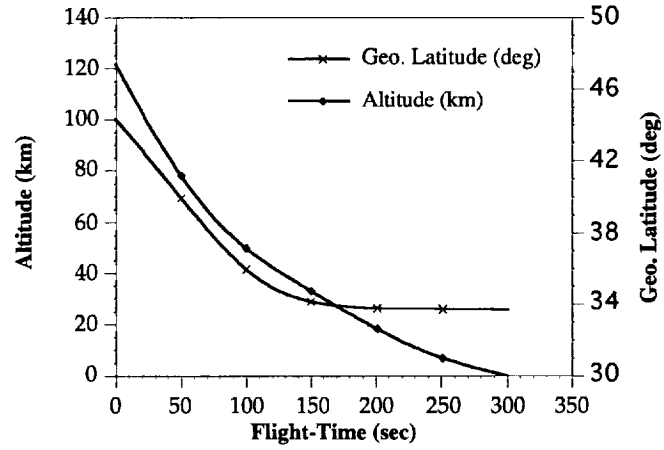


Figure 3.6 - Flight-Time vs. Altitude and Latitude

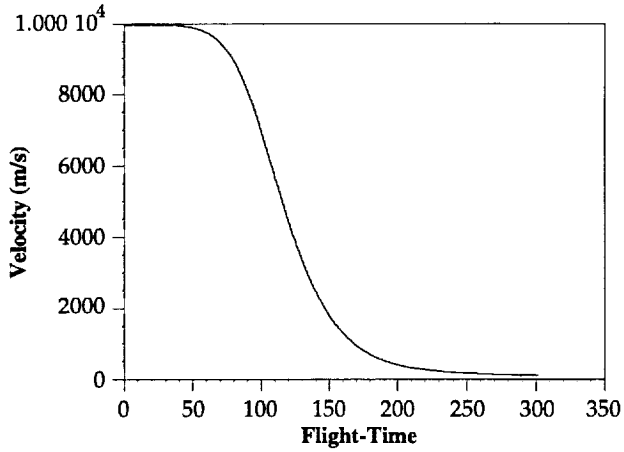


Figure 3.7 - Flight-Time vs. Velocity

Finally, in Figure 3.7 the velocity during the flight is depicted versus the flight-time. The main drop in velocity starts after about 70 sec. The velocity further decreases until grav-

ity and aerodynamic forces reach an equilibrium and, hence, it remains approximately constant at about 110 m/s, which is also the impact velocity on the surface. Of course this is a high value, but we also see that use of parachutes during the last 10 to 20 km altitude will have only a small influence on the landing position if we do not assume winds.

The output values for this nominal flight simulation make us confident that the model has been implemented properly. Small deviations from the documented values are caused by the model simplification.

3.4 WHAT HAS BEEN PRESENTED AND WHAT IS NEXT?

Now we have established the analytical model of the LifeSat vehicle and implemented it by choosing nine important dispersed parameters under some assumptions and simplifications. This model will be used in Chapter 4 to answer the questions posed at the end of Chapter 1 to satisfy our report goals. Simulation with orthogonal arrays is used to simulate the noise factors and quality techniques are applied to find a better design. In summary, we want to find answers to the following questions:

- Is the use of orthogonal arrays for simulation an alternative to Monte Carlo simulation?
- If it is, can we establish the same confidence level with a reduced number of simulations?
- Is the signal-to-noise ratio a measurement for quality and confidence?
- Using ANOVA, are we able to estimate the factor influences on output variations?

In Chapter 4, we perform simulations with the developed and implemented model for the LifeSat vehicle. We also try to find limitations and identify further investigations necessary to apply orthogonal arrays for simulations.

CHAPTER 4

ORTHOGONAL ARRAY BASED SIMULATION OF THE LIFESAT MODEL

In this chapter, we will discuss simulations of the LifeSat model that were developed in Chapter 3. Results of Monte Carlo simulations provide the baseline for our comparison of simulations with orthogonal arrays. A template, in which all required information for the simulation is presented, is formulated in Section 4.1.1. For each dispersed parameter, the mean and the standard deviation are given. The model is mainly examined by using orthogonal array based simulations as explained in Chapter 2. By using the statistics of the output, we are able to compare the different methods and to prove the accuracy of the use of orthogonal arrays. In the first study, we investigate model behavior and compare statistics. In the second study, we focus on the choice of different levels for the dispersed input parameters.

In the last section of this chapter, we focus on statistical confidence and robust design. We apply the analysis of variance technique to identify the most influential parameters on the variation of the output. The employment of the signal-to-noise ratio to improve the quality is demonstrated.

4.1 INITIAL LIFESAT MODEL SIMULATION

In this section, we first study the model behavior and make comparisons between Monte Carlo simulations and orthogonal array simulations. For each of the two methods we perform two simulation studies and compare the statistical values of mean and standard deviation for several system outputs. From the results we obtain a first estimation of the feasibility of the orthogonal array approach. We also obtain information about the savings of computational time. As the output of interest, the longitude and geodetic latitude are selected as representatives of the landing position (representing performance parameter J_5) and the maximum acceleration and the maximum dynamic pressure as performance parameters J_1 and J_2. A *footprint* is a scatter diagram of longitude and geodetic latitude which is used to show the landing position.

4.1.1 Template Development for Orthogonal Array

In Section 2.4, a simulation technique based on orthogonal arrays is presented. Especially the three-level orthogonal array is used and will be used further throughout this entire study. In Chapter 3, we discuss the LifeSat model and the vehicle and environmental parameter dispersions. From the given parameter dispersions in Section 3.2 we identify an orthogonal array and the levels for each factor. First, however, we want to recall the dispersed parameters for the implemented LifeSat model. These are

- initial state,
- atmospheric density,
- vehicle mass,
- angle of attack, and
- vehicle drag coefficient.

The initial state is described by three parameters for position and three parameters for velocity. Therefore, we have a total of ten dispersed parameters. We have seen the relation between the angle of attack and the vehicle drag coefficient in Section 3.2. If we refer to Figure 3.4 in Section 3.2.1, we see that the changes in the angle of attack (less than 1%) are small compared to the dispersion of the drag coefficient, c_d. For the use of orthogonal arrays we had to define nine levels for the drag coefficient: three for angle of attack α = -5 deg, three for α = 0 deg, and three for α = 5 deg. Since we want to use only three levels for each parameter, we calculate the arithmetic mean of the c_d for the angle of attack between α = 0 deg and α = 5 deg. Therefore, the average drag coefficient is calculated from

$$\bar{c}_d = \frac{c_d(\alpha = 0) + c_d(\alpha = 5)}{2} = \frac{0.665 + 0.671}{2} = 0.668 \quad . \tag{4.1}$$

This is the mean value for the drag coefficient we have used. Three levels are selected with respect to this value. This means that the final model for the use of orthogonal arrays has nine dispersed parameters each on three levels. The orthogonal array which fits this model is L_{27} with 27 experiments and 13 columns. We leave the first four columns empty and assign the dispersed parameters to the remaining nine columns.

Before doing this, the initial state is defined and coordinates are transformed. The initial state data are the necessary input for the simulation, and we choose settings for the velocity, geodetic latitude, and flight path angle to have the same magnitudes as shown in [McCleary, 1991]. When all parameters are set to their mean values, the landing position of this flight simulation is assumed to be close to the mean of all simulations when the parameters are dispersed. To obtain the mean values, we first determine the data for the initial velocity vector. Then the landing position for geodetic latitude is varied by modifying the initial latitude value. Longitude, altitude, and azimuth are the same as in the validation experiment in Section 3.3.2. In Table 4.1, all of the initial state parameter values are presented. The mean values for the longitude, altitude, and inertial azimuth are not varied throughout the several scenarios.

Table 4.1 - Initial State Data

Initial Velocity	9000 m/s
Longitude	-106.65 deg
Geodetic Latitude	43.0 deg
Altitude	121920 m
Inertial Flight Path Angle	-5.8 deg
Inertial Azimuth	180.0 deg

With a coordinate transformation to the Cartesian system, we obtain initial state parameters for position and velocity which are then dispersed as shown in Section 3.2.2. Please refer to Figure 2.10 in Section 2.3.3 where the OA L_{27} is shown. In the same order of the parameters in Table 4.2 we assign the x, y, z coordinates of position to the columns (5 to 7) with factors **E**, **F**, and **G**, respectively.

Table 4.2 - Parameter Statistics

Parameter	Mean μ	Std. Deviation σ
Position: x	-1360.4 km	5000 m
Position: y	-4548.8 km	5000 m
Position: z	4427.5 km	5000 m
Vehicle Mass	1560.4 kg	1.667%
Atm. Density	1.2 kg/m^3	10%
Drag Coefficient	0.668	1.667%
Speed: x	-1559.1 m/s	0.667%
Speed: y	-5213.2 m/s	0.667%
Speed: z	-7168.8 m/s	0.667%

The vehicle mass is assigned to column 8 with factor **H** in the same way the remaining parameters are assigned to columns 9 to 13. This choice is arbitrary and parameters can be assigned in other ways to the columns. For the orthogonal array experiments, the levels for all normally distributed parameters are chosen to be $\mu_i - \sqrt{1.5}\ \sigma_i$, μ_i, and $\mu_i + \sqrt{1.5}\ \sigma_i$, as suggested in Section 3.3.1.

The values shown in Table 4.2 represent the mean and the standard deviation for each dispersed parameter after coordinate transformation. The standard deviation for each position component is the absolute value of 5000 m. All other deviations are given in percentages of the mean values. Multiplying the standard deviation by three we obtain the 3σ values given in Section 3.2.2.

In Section 4.1.2, the model is first simulated using Monte Carlo simulation and a sample size of 1000 error vectors. The Monte Carlo results provide a good representation of the system range and, therefore, of the baseline for comparison. Two Monte Carlo runs with different seed numbers for the random number generator are performed to get two sets of data.

The study in Section 4.1. is used as follows:

- to understand the behavior of the model,
- to verify its implementation with dispersed parameters, and
- to understand the use of orthogonal arrays for simulation compared to Monte Carlo simulation.

In Section 4.1.3, we examine system performance based on the same simulations as in Section 4.1.2. The maximum acceleration and maximum dynamic pressure represent the performance characteristics. All results are then compared in Section 4.1.4 with respect to the means and standard deviations. In the last part of Section 4.1, we analyze in greater detail the statistics parameters for the geodetic latitude and show how the individual experiments of orthogonal arrays contribute to the output.

4.1.2 Initial Footprints for Monte Carlo and Orthogonal Array Simulations

In this first study, we present the behavior of the LifeSat model and verify its implementation by comparing the system's output response with the data obtained by Monte Carlo and orthogonal array simulation. The focus is on the dispersion of the geographical parameters longitude and geodetic latitude and on the dispersion of performance parameters J_1 and J_2, which are the maximum acceleration and the maximum dynamic pressure. The initial state is chosen in such a way that the selected parameter values result in a landing position which is close to the desired target area of -106.65 deg longitude and 33.6 deg geodetic latitude.

As mentioned before, a Monte Carlo simulation with 1000 samples is used to obtain a footprint of the landing range. We have done two Monte Carlo simulations with 1000 samples each by using different seed values for the random number generator and, hence, have obtained two data sets to calculate the statistics. The two footprints are shown in Figure 4.1.

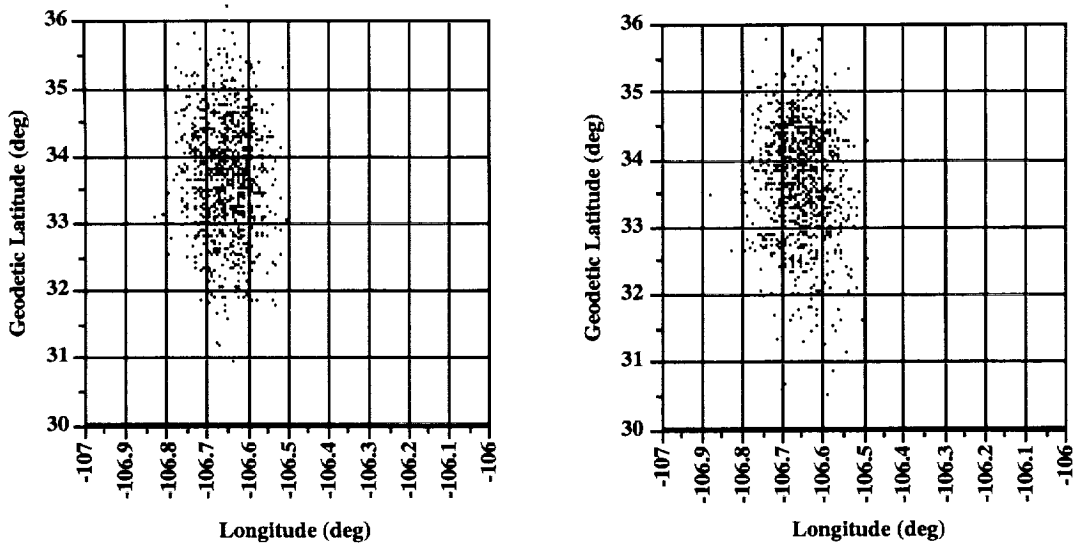


Figure 4.1 - Footprints of Two Monte Carlo Simulations Using 1000 Samples

Each point in Figure 4.1 represents the vehicle's landing position which is the output of one simulation of the flight trajectory. Most of the points are concentrated in an area from -106.52 deg to -106.78 deg longitude and from 31.7 deg to 35.5 deg geodetic latitude. In both pictures we find some differences in the position of some of the points, but this has to be the case since we have two different sample populations. The calculated statistics provide a measure to identify the differences. To obtain the dispersions in kilometers, a transformation of the coordinate system leads to the following values at a radius of 6370 km, which represents the Earth radius:

- 1 deg longitude $\hat{=}$ 111.2 km, and
- 1 deg geodetic latitude $\hat{=}$ 111.2 km.

Hence, most of the points are placed within a rectangle of the size 30 km x 420 km. This result represents a large variation around the target position. At this stage we do not know which of the factors has the highest contribution for the variation.

Having done two Monte Carlo simulations with a sample size of 1000, we now use the same initial parameter values and dispersions to simulate the trajectory with orthogonal arrays. Two footprints for the orthogonal array based simulation are shown in Figure 4.2. Because we are using the same scale for the presentation, it is easy to compare these footprints with the two footprints in Figure 4.1.

In the left picture, we have assigned the dispersed parameters to the orthogonal array columns as explained in Section 4.1.1. In the right picture, we then have changed the columns and therefore obtain a different set of factor combinations for 27 experiments. Because we still use the last nine columns, the first three experiments remain the same. In experiment 1, all parameters are at level 1; for experiment 2 at level 2; and for experiment 3 at level 3. All the other experiments have modified factor combinations.

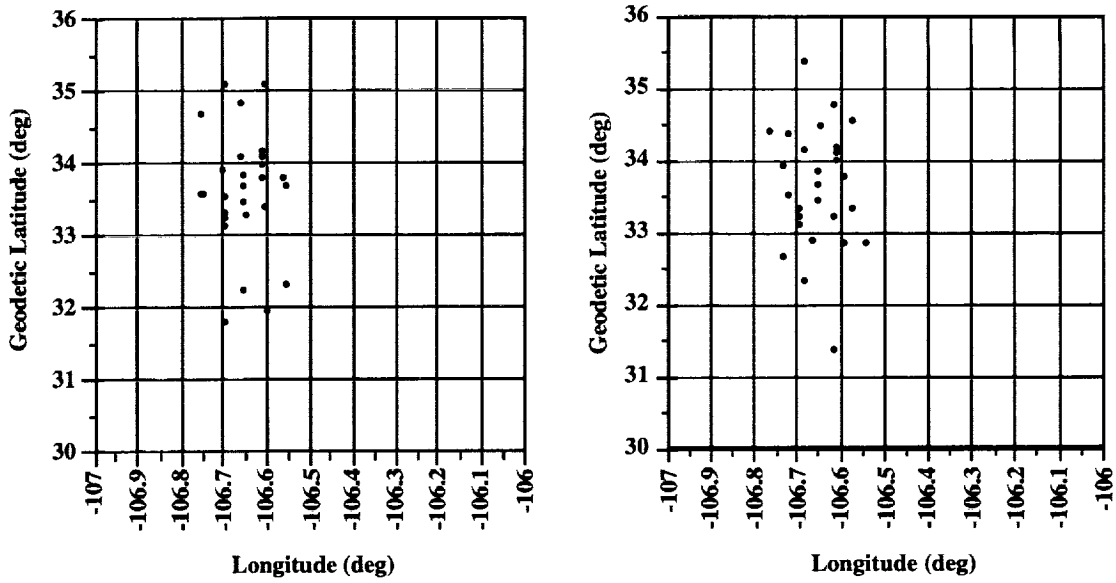


Figure 4.2 - Footprints for Orthogonal Array Based Simulation with 27 Experiments

In the footprints for the orthogonal array based simulations, we identify three main groups of points in the left picture of Figure 4.2. All points are placed at a longitude between -106.55 deg and -106.75 deg. Four points of one group are placed around 35 deg geodetic latitude, 19 points are placed around the mean (which is 33.6 deg for geodetic latitude and -106.65 for longitude), and the last four points are placed around 32 deg. In the right picture, we do not find these groups but two points which are on the extremes of the geodetic latitude. One is at a value of 35.4 deg, and one is at a value of 31.4 deg.

Obviously, the points of the footprints for orthogonal array based simulation do not have a distribution, which can be considered a normal distribution as it "seems" to be in the Monte Carlo simulation. In Table 4.3, we present the flight statistics for the landing range. The mean values and standard deviations for longitude and geodetic latitude are given in Table 4.3. The presented values are estimates of the real populations.

Table 4.3 - Landing Range Statistics

Method	Longitude Mean μ (deg)	Longitude Std. Deviation σ	Geo. Latitude Mean μ (deg)	Geo. Latitude Std. Deviation σ
Nominal	-106.650	-	33.625	-
Monte Carlo 1	-106.652	0.0541	33.591	0.808
Monte Carlo 2	-106.651	0.0582	33.581	0.825
Orth. Array 1	-106.650	0.0548	33.582	0.813
Orth. Array 2	-106.650	0.0547	33.582	0.813

The simulation with all parameters at their mean values is called the nominal flight and is shown in the first row. Since the nominal flight represents only one flight, there is no standard deviation value. We deal with a nonlinear model, therefore it is not possible to estimate the mean in advance as we have seen in Section 2.2.2. But we expect to be close to the real mean. In the following rows, the results for two Monte Carlo simulations (with 1000 samples) and two orthogonal array simulations (with 27 samples) are presented, denoted by numbers 1 and 2.

Comparing the mean values for the longitude first, we do not see any significant differences. The values for standard deviation range from 0.0541 deg to 0.0582 deg. The values for the orthogonal array are within this range and are almost the same. This standard deviation represents a value of approximately 6 km. For the geodetic latitude, there is a slight difference in the mean value between nominal flight and the other simulations. Now the standard deviation represents a value of approximately 90 km. This is extremely large because the available target area is only approximately 45 km in length in the direction of geodetic latitude. The values for orthogonal array simulation are again within the range of the Monte Carlo simulations.

For the verification of our model and comparison of the Monte Carlo and orthogonal array simulations, it is not important how large the variation in the landing range is. We are interested in the *difference of the output* when we use orthogonal arrays to represent the *variation* in the noise factors.

Besides the footprint we also want to know the magnitudes of the performance parameters. Outputs and statistics are presented in the next section.

4.1.3 System Performance for Monte Carlo and Orthogonal Array Simulations

Since the covered area of both footprints (Monte Carlo and Orthogonal Array) and also of the statistics are nearly the same, we expect the same coverage for other outputs. These are the performance parameters for maximum acceleration and dynamic pressure. We only present the result of the first orthogonal array based simulation, which appears in the left picture of Figure 4.2 and on the fourth row in Table 4.3. In Figure 4.3, the relation between the maximum acceleration and the geodetic latitude is depicted.

Obviously, there is a strong relationship between the geodetic latitude and the maximum acceleration. Within the range of the geodetic latitude the maximum acceleration varies between 120 m/s^2 and 160 m/s^2. This is approximately a g-load between 10 and 16. The relation between the two parameters is almost linear. We are not concerned about the grouping since the whole range is covered.

A plot of the maximum acceleration versus longitude is depicted in Figure 4.4. In this figure, we cannot identify any relationship between the parameters. Although there are a few more points in the middle, the distribution is more "uniform" than in the previous figure.

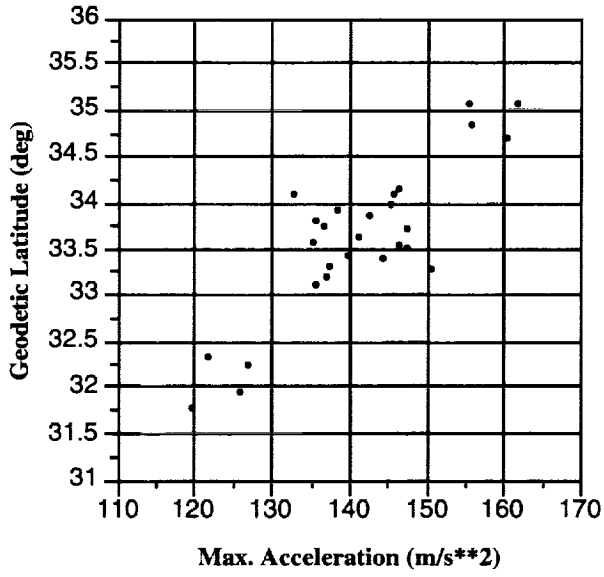


Figure 4.3 - Maximum Acceleration vs. Geodetic Latitude

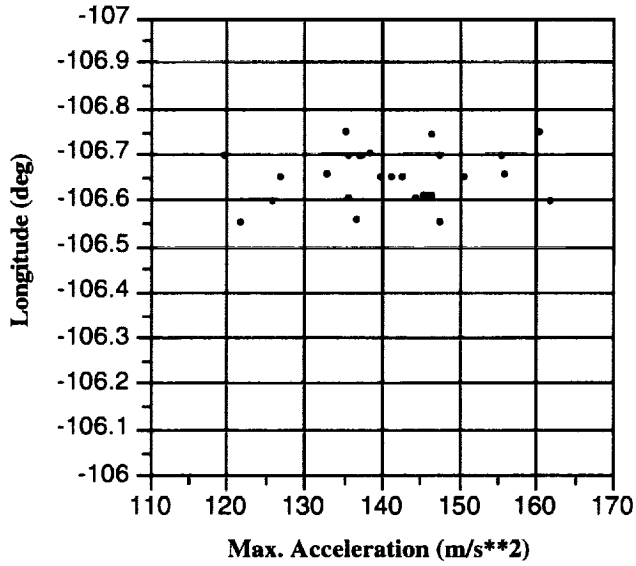


Figure 4.4 - Maximum Acceleration vs. Longitude

Another important performance parameter is the maximum dynamic pressure during flight. This parameter is the product of density and speed and is described in Section 3.1.5 by $0.5\rho v_r^2$. The product of dynamic pressure, reference area, and drag coefficient provides the value for aerodynamic force because we have neglected the lift-coefficient term. In Figure 4.5, we see the strong correlation among the two performance parameters J_1 and J_2, the maximum dynamic pressure, and the maximum acceleration.

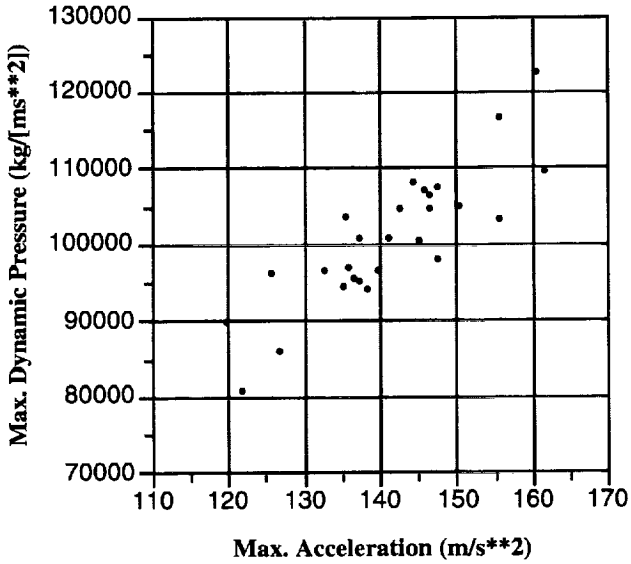


Figure 4.5 - Maximum Dynamic Pressure vs. Maximum Acceleration

The range of the dynamic pressure is between 80,000 kg/(ms^2) and approximately 120,000 kg/(ms^2) when we use the described columns of orthogonal arrays for the factors. The range with Monte Carlo simulation is expected to be larger as it was in Figure 4.1. Again we obtain an almost linear relationship between the parameters.

The mean values and standard deviations for the performance parameters are given in Table 4.4. The first row again represents the nominal flight, the next two rows the results of the Monte Carlo simulations, and the last two rows give the orthogonal array based simulations.

Table 4.4 - Statistics for Performance Parameter

Method	Max. Accel. Mean μ (deg)	Max. Accel. Std. Deviation σ	Max. Dyn. Pr. Mean μ (deg)	Max. Dyn. Pr. Std. Deviation σ
Nominal	141.569	-	100992.8	-
Monte Carlo 1	141.490	10.164	101092.0	7848.7
Monte Carlo 2	141.798	10.733	101289.0	8329.1
Orth. Array 1	141.455	10.355	100510.0	8452.9
Orth. Array 2	141.457	10.354	100511.0	8434.5

The presented statistical values in Table 4.4 have the same tendency for the different simulation runs as for the landing position. The values for orthogonal array simulation are within the range of the two Monte Carlo simulations for the maximum acceleration.

For the maximum dynamic pressure the mean values for orthogonal arrays are smaller and, although the standard deviation is larger, of the same magnitude.

4.1.4 Comparison of the Statistical Parameters

In the last two sections, we present two Monte Carlo simulations and two orthogonal array based simulations along with the nominal flight. Deviations between the mean values seemed to be very small and those between the standard deviation values not as big. A comparison of these deviations relative to the smallest value of all cases (including the nominal flight) provides some first information about the magnitude of the deviations in percent. We use Tables 4.3 and 4.4 for the calculations. In Table 4.5, the smallest value always equals 100% and is shown in "bold/italic" style. The other values (given in percent) represent the percentage difference to these values.

Table 4.5 - Relative Differences of Parameter Statistics for Nominal, Monte Carlo, and Orthogonal Array Simulation

Parameter	Nominal	Monte Carlo 1	Monte Carlo 2	Orthogonal Array 1	Orthogonal Array 2
Longitude μ	***-106.65 deg***	0.002%	0.001%	0.0%	0.0%
Longitude σ	-	***0.0541***	7.58%	1.29%	1.11%
G. Latitude μ	0.13%	0.03%	***33.581 deg***	0.003	0.003
G. Latitude σ	-	***0.8083***	2.1%	0.58%	0.58%
Acceleration μ	0.08%	0.02%	0.24%	***141.455 m/s^2***	0.001%
Acceleration σ	-	***10.164***	5.6%	1.9%	1.9%
D. Pressure μ	0.48%	0.6%	0.78%	***100,510 kg/(ms^2)***	0.001%
D. Pressure σ	-	***7848.7***	6.12%	7.7%	7.5%

The result of the first Monte Carlo simulation represents the smallest standard deviations for all observed parameters, whereas the second Monte Carlo simulation results in the highest standard deviations compared to the first one for almost all parameters. The differences of all parameter mean values are very small (from 0.0 to 0.78%). Larger differences only occur in the standard deviation of the dynamic pressure.

From the results in Table 4.5, we justify the following conclusions from comparisons with the smallest values of mean and standard deviation:

- The differences in the estimation of the mean values are very small since all deviations are smaller than 1%. The deviation is even smaller for orthogonal arrays than for the Monte Carlo simulation.

- In the estimation of the standard deviations, the differences between both Monte Carlo runs are great. Without establishing statistical confidence we cannot say which value is the "better" estimation.

- The differences in the statistics for the dynamic pressure are larger than for the other outputs.

In summary, in Sections 4.1.2 to 4.1.4 we have compared two Monte Carlo simulations with two orthogonal array based simulations and the nominal flight. This comparison was based on statistics for the landing range and two performance parameters. In the following section, we examine the relationship between the statistical parameters and the experiments in the orthogonal arrays.

4.1.5 Analysis of Orthogonal Array Experiments with Respect to the Geodetic Latitude

In Figure 4.2, we see that a change in the assigned orthogonal array columns for the parameters causes a change in the output response of the system. In both cases, we have two fractions of all possible combinations of a full factorial design (Sec. 2.3). In the following three figures (4.6 to 4.8), we compare the value for the geodetic latitude, mean, and standard deviation to see the differences. This output parameter is representative for all.

In the two histograms in Figure 4.6, geodetic latitude values are shown for each experiment. They are helpful to understand the curves of the mean and the standard deviation. We are able to find the exact number of an experiment which results in a larger deviation from the mean value in the output.

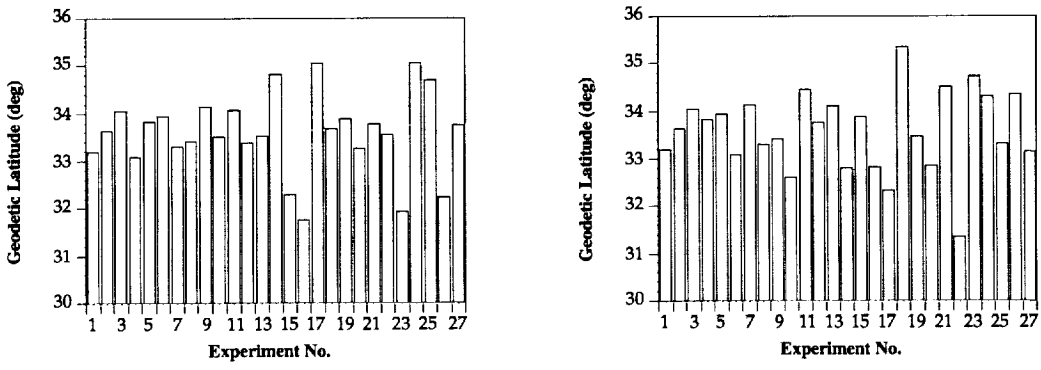


Figure 4.6 - Histogram for Geodetic Latitude

The left picture in Figure 4.6 represents the first simulation with orthogonal arrays; the right picture represents the simulation with changed columns. Since we use the last nine columns of the orthogonal array L_{27} (Fig. 2.10), the first three experiments have the same settings and result in the same output response. From the 4^{th} to the 27^{th} experiment, the

output for each experiment is different. Therefore, the mean values for the geodetic latitude as shown in Figure 4.7 differ from the fourth experiment.

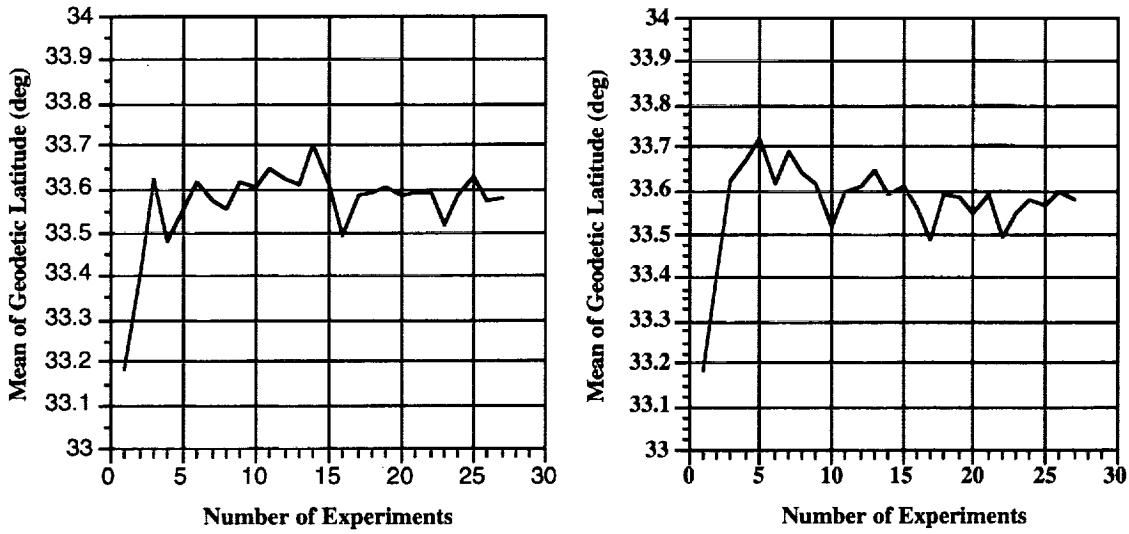


Figure 4.7 - Mean Value for Geodetic Latitude for each Experiment

As can be seen in Figure 4.7, we cannot talk about convergence for the mean value. For both cases the first three experiments have the same values, but experiment 4 gives different directions to the curve. Although the curves are different at experiment 27 the mean values are corresponding, which is quite surprising.

The behavior of the standard deviation for the geodetic latitude is shown in Figure 4.8. The first experiment must have a value of zero. With the following experiments the value increases.

The behavior can easily be understood when we look at values for the geodetic latitude for each experiment as shown the histograms in Figure 4.6. After the first three experiments in the left picture of Figure 4.8, the highest increase in the standard deviation is a result of experiments 14, 15, 16, and 17. Experiments 14 and 17 have a high value and experiments 15 and 16 have a low value relative to the mean. In the right picture, there are three smaller increases due to experiments 10 and 11, experiments 17 and 18, and experiments 22 and 23.

After 27 experiments the values for both cases are the same although we have different factor combinations and, hence, different outputs. In Tables 4.3 and 4.4, we have seen that we obtain approximately the same values for the longitude and for the two performance parameters.

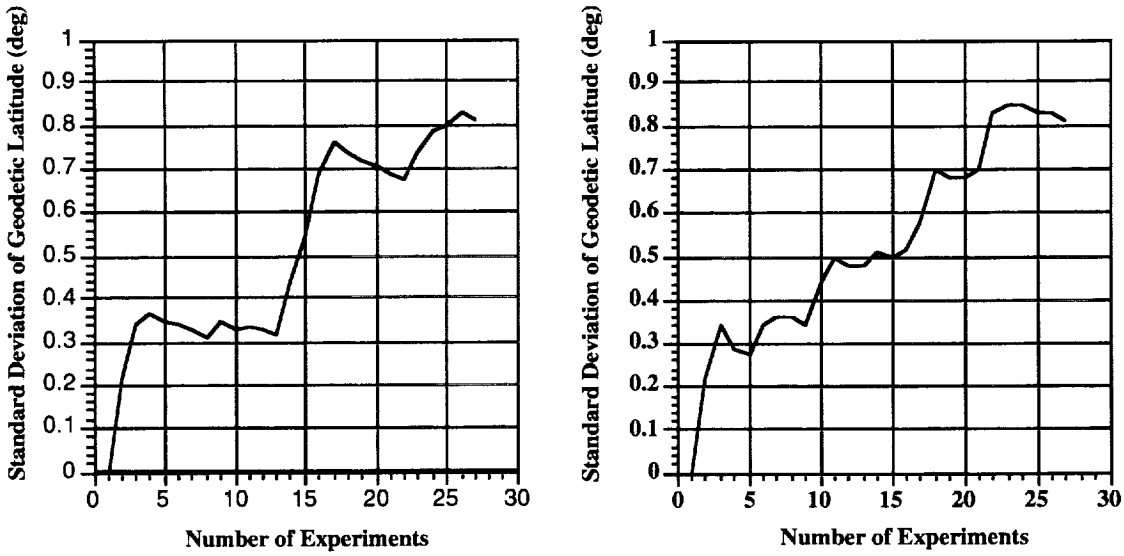


Figure 4.8 - Standard Deviation for Geodetic Latitude

From all the results presented we draw the following conclusions and answers to previously posed questions:

- Estimations of mean and standard deviation for output parameters using orthogonal arrays are obtained only after all the predetermined experiments are executed.
- There seems to be no such thing as convergence. But after all experiments are performed, the values obtained are very accurate compared to the Monte Carlo simulation and the nominal flight.
- The differences between the two Monte Carlo simulations are larger than the differences between the two orthogonal array simulations. The reason might be that one of the three levels for orthogonal arrays is the mean value of the parameter.
- We now expect orthogonal array based simulation to be a feasible approach compared to Monte Carlo simulation.

The next question to be answered is: How confident are we that the approach is feasible? This is discussed in the following section.

4.2 STATISTICAL CONFIDENCE OF ORTHOGONAL ARRAY BASED SIMULATION

The question arising now is: Do the statistical results from orthogonal array simulations represent the same population as the results obtained by Monte Carlo simulation?

To answer this question, we compare ten different runs of Monte Carlo simulation with ten runs of orthogonal array simulation. Each Monte Carlo result is based on a sample size of 1000 and the random number generator always has a different seed value. For orthogonal array simulation, we have 27 individual simulations. For the ten orthogonal array based simulation runs, factors are assigned to different columns in the OA L_{27}.

We choose values of the geodetic latitude for the test of confidence that are representative of all other parameters. For each of the ten runs (sample size 10), we calculate the mean and the standard deviation as before. An important theorem found in [Sokal and Rohlf, 1981] is: *The means of samples from a normally distributed population are themselves normally distributed, regardless of sample size n.* Thus, we note that the means of our ten samples are normally distributed. The same property is valid for the standard deviation and other statistics. The standard deviation of the means, standard deviations, etc. is known as *standard error* [Sokal and Rohlf, 1981]. More information about standard error is found in [Sokal and Rohlf, 1981]. For both simulation methods, we calculate the average of the mean and the standard deviation and also calculate the standard error for these statistics. The results for each run and calculations are shown in Table 4.6.

Table 4.6 - Distribution of Means and Standard Deviations for Ten Monte Carlo Simulation and Ten Orthogonal Array Simulation Runs

10 x Monte Carlo (1000)		10 x Ort. Array (27)		
Latitude (μ)	Latitude (σ)	Latitude (μ)	Latitude (σ)	
33.588	0.642	33.592	0.641	
33.603	0.678	33.593	0.633	
33.599	0.698	33.593	0.619	
33.591	0.654	33.590	0.671	
33.596	0.648	33.592	0.644	
33.588	0.663	33.591	0.642	
33.601	0.672	33.591	0.654	
33.606	0.657	33.590	0.661	
33.601	0.669	33.591	0.649	
33.574	0.660	33.592	0.637	
33.595	0.664	33.592	0.645	*Average*
0.0096	0.0161	0.0011	0.0147	*Standard Error*

The difference in the standard error for the means as shown in the last row of columns 1 and 3 is very large, but it is small compared to the standard error of the standard deviations σ with respect to the magnitude of the values. We mean that a standard error of 0.01 for a mean value of 33.6 is small compared to a mean of 0.66. For orthogonal array simulation, the standard error for the means is approximately ten times less than for Monte Carlo simulation. The average mean values are almost the same. But the difference of the standard error of standard deviations between the two methods is very small. The values in the average of the standard deviations have a small difference. In Figure 4.9, the distribution of the standard deviation is shown for the geodetic latitude obtained from the Monte Carlo simulation and orthogonal array simulation.

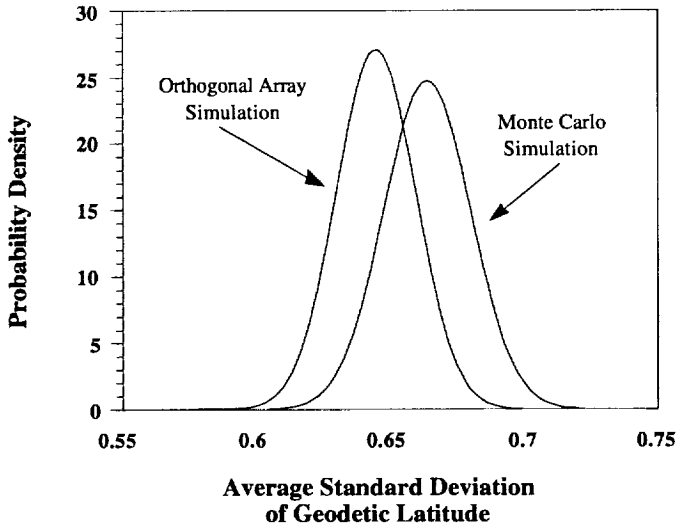


Figure 4.9 - Distribution of Average Standard Deviation for Ten Monte Carlo and Ten Orthogonal Array Simulations for the Geodetic Latitude

We observe that there is a difference between the two distributions. The orthogonal array simulations result in a smaller mean (average of all standard deviations) and a standard error. To verify the observed difference we need a statistical calculation. We want to answer the following question:

> *Is sample 1, with a mean of 0.664 and a standard deviation of 0.0161, from the same population as sample 2, with a mean of 0.645 and a standard deviation of 0.0147?*

A method to analyze the *difference between two sample means* is the *t-test.* The *t*-test is a traditional method to solve our problem [Sokal and Rohlf, 1981]. The quantity *t* has a distribution very similar to the standard normal distribution [Dunn and Clark, 1987]. It is described in more detail elsewhere [Dunn and Clark, 1987; Sokal and Rohlf, 1981]. When we have two equal sample sizes (ten in our case), we calculate the t value from

$$t = \frac{\overline{Y}_1 - \overline{Y}_2}{\sqrt{\frac{s_1^2 + s_2^2}{n-1}}} = \frac{0.664 - 0.645}{\sqrt{\frac{0.0161^2 + 0.0147^2}{9}}} = 2.61 \ . \qquad (4.2)$$

In Equation (4.2), $\overline{Y}_i$ is the means of the standard deviations and s_i is the standard errors. Comparing the value with the table values for nine degrees of freedom, we obtain values for standard deviation which are the same in at least 97.5% of all the cases. This is an important result since it says that we obtain a high confidence in the results of orthogonal array-based simulations compared to Monte Carlo simulation.

We perform two additional runs, one with Monte Carlo simulation and 10,000 samples, and one with orthogonal arrays and 81 experiments. For 81 experiments, we simply use nine columns of the OA L_{81} from a maximum of 40 columns. The results for these two simulations are shown below:

- Estimation of the geodetic latitude with 10,000 Monte Carlo simulations

  μ = 33.591 deg,
  σ = 0.659.

- Estimation of the geodetic latitude with 81 Orthogonal Array simulations

  μ = 33.598 deg,
  σ = 0.674.

For the Monte Carlo simulation, the mean is farther away from 33.6 deg and the standard deviation is lower than in the previous case. For orthogonal array simulations, the mean is closer to 33.6 deg and the standard deviation is higher than the previous average for Monte Carlo simulations. Thus, any simulation technique provides results within a certain confidence interval around the actual population mean. But simulation with orthogonal arrays has a smaller standard error that the Monte Carlo simulation.

For additional information many more samples have to be evaluated and different levels for orthogonal arrays have to be compared. Although the results are not exactly the same for both simulation techniques, we continue our study with the analysis of variance.

4.3 ANALYSIS OF VARIANCE FOR THE TRAJECTORY SIMULATION

In Section 2.5, we introduce analysis of variance as a tool for the analysis of experimental data. We are able to calculate factor effects on the variation of the data. We apply ANOVA to evaluate the contribution of noise factors to the performance of the LifeSat vehicle. Performance is determined by the landing position and by the performance parameters. We select the geodetic latitude to study the factor effects for a simulation, which will be used for a robust design example. In Sections 4.1 and 4.2, we disperse the x, y, z components for position and speed. Both times, although the mean is on target, there is a large variation around this target. When we reduce the tolerances for the initial position components from 5000 m to 2000 m, a reduced variation in the footprint (compare Figures 4.1 and 4.9) is obvious. We reduce the variation in the following sections of the initial position further by dispersing altitude, flight path angle and azimuth (Sec. 3.1.3). In Section 4.3.1, we calculate factor contributions with Monte Carlo simulation and do the same with orthogonal arrays and ANOVA in Section 4.3.2.

4.3.1 Factor Contributions with Monte Carlo Simulation

In the following example, we disperse the parameters of the initial state before we transform them into the Cartesian coordinate system for the simulation. In Table 4.7, we

present the new dispersions for the initial state and the dispersions for mass, density, and drag coefficient, which remain at their previous values.

Table 4.7 - Parameter Statistics

Parameter	Mean μ	Std. Deviation σ
Initial Velocity	9946.5 m/s	0.667%
Longitude	-106.65 deg	0.1 deg
Geodetic Latitude	44.2	0.05 deg
Vehicle Mass	1560.4 kg	1.667%
Density	1.2 kg/m^3	10%
Drag Coefficient	0.6679	1.667%
Altitude	121920.0 m	250.0 m
Flight Path Angle	- 5.88 deg	0.25%
Azimuth	180.0 deg	0.333%

In the following, we vary each factor alone by using 1000 sample points for Monte Carlo simulation. Recall Equation (2.12), where the sum of squares is calculated from

$$SS_T = \sum_{i=1}^{N} y_i^2 - \frac{T^2}{N} = \sum_{i=1}^{N} y_i^2 - CF \quad . \tag{4.3}$$

We are interested in the effect of each parameter on the variation of the output. By varying one parameter each time, we calculate a total sum of squares for the output variation that is due to only one factor. From the sum of all individual total sum of squares (SS_T), we calculate the individual factor contribution. Remember that we study the effects for the geodetic latitude. The parameters, the total sum of squares, and the percentage of contributions are shown in Table 4.8.

Note that only seven parameters are shown in Table 4.8. The sum of squares for azimuth and longitude has no significant value and is summarized in "other". This sum becomes part of the error term later in this section when we analyze the orthogonal array based results with ANOVA.

We observe that the variation of the density has the highest influence on the variation of the geodetic latitude with a 45% contribution. The next largest contributor is the velocity with 22%. Vehicle mass, altitude, and flight path angle have approximately equal influences of 5%. The drag coefficient has only an insignificant contribution. The sum of all individual sum of squares is $SS_T = 61.825$.

Table 4.8 - Factor Contribution to the Variation of Geodetic Latitude

Parameter	Total Sum of Squares (SS_T)	% Contribution
Initial Velocity	13.975	22.6
Latitude	3.366	5.44
Vehicle Mass	2.773	4.49
Vehicle Drag Coefficient	0.814	1.32
Density	27.901	45.1
Altitude	2.902	4.69
Flight Path Angle	2.765	4.47
Other	0.813	1.31

In another simulation we vary all parameters at the same time again with a sample size of 1000. The total sum of all factors varied at once is $SS_T = 48.66$. This is much less than the sum of all individual variations (SS_T). Thus, we should not study the effects of parameters by varying one parameter at a time.

4.3.2 Factor Contributions with ANOVA and Orthogonal Array Simulation

To obtain the presented Table 4.8 we do 1000 simulations for each factor, which is a total of 9000 for all nine factors. If we vary all factors at once, we still have to simulate 1000 sample points. The use of orthogonal arrays and ANOVA provides a capable tool with which to estimate the factor contributions with much less effort. Factors azimuth and longitude are pooled into the error since they have no significant contribution. With ANOVA we do only 27 experiments to estimate all of the factor effects. In Section 2.5, all equations (Equations (2.11) to (2.21)) for the analysis are presented. We have implemented these equations into a computer program. The results of implementation and the ANOVA table are shown in Table 4.9.

In Table 4.9, the data are described first. The ANOVA table includes seven factors with their contributions, the error term, and the total. We need to identify which factor number represents which parameter.

Factor 1 --> Density

Factor 2 --> Vehicle Mass
Factor 3 --> Drag Coefficient
Factor 4 --> Velocity
Factor 5 --> Flight Path angle
Factor 6 --> Altitude
Factor 7 --> Geodetic Latitude

Factors 8 and 9 are the longitude and the azimuth. Their contribution is *pooled* [Roy, 1990; Ross, 1988] into the error term since their contribution is insignificant. Each value for the sum of squares of the first seven factors is larger than the error sum of squares.

Table 4.9 - Results and ANOVA Table for the Variation of Geodetic Latitude

DESCRIPTION OF DATA

Number of points:	27
Mean:	33.5919
Standard deviation:	0.2269
Variance:	0.0515
ssTot:	1.3383
Sum:	906.9821
Sum of squares:	30468.6152

Interval mean ±1*st.dev.:	(33.3651, 33.8188)
Interval mean ±2*st.dev.:	(33.1382, 34.0457)
Interval mean ±3*st.dev.:	(32.9113, 34.2725)

ANOVA FOR THE DISPERSIONS

Source:	f	SS	V	F	% Perc.	Confidence
Factor 1:	2	0.6895	0.3447	211.8000	49.14%	100.00%
Factor 2:	2	0.1289	0.0645	39.6000	9.00%	100.00%
Factor 3:	2	0.0215	0.0107	6.6000	1.31%	98.83%
Factor 4:	2	0.3496	0.1748	107.4000	24.80%	100.00%
Factor 5:	2	0.0566	0.0283	17.4000	3.82%	99.97%
Factor 6:	2	0.0605	0.0303	18.6000	4.10%	99.98%
Factor 7:	2	0.0703	0.0352	21.6000	4.80%	99.99%
Factor 8:		Pooled				
Factor 9:		Pooled				
Error:	12	0.0195	0.0016			
Total:	26	1.3965				

Now we compare these data from the orthogonal array based simulation with the data obtained by Monte Carlo simulation, shown in Table 4.8. The results are similar but have some differences. The effects of density, velocity, flight path angle, altitude, geodetic latitude, and drag coefficient are approximately the equal for both simulation techniques. The largest difference is found in the contribution of the vehicle mass, which is twice as much for orthogonal array simulation as for Monte Carlo simulation. At this point we

cannot tell which estimation is more accurate. But we must remember the difference for Monte Carlo simulation in the dispersion of each factor individually and of all factors together. All seven factors have a confidence level of greater than 98%. This means we have a confidence of at least 98% that each of the factors is contributing to the variation in the geodetic latitude.

In summary we can say that ,by using orthogonal array based simulation and ANOVA, we obtain as much information with only 27 experiments as by using Monte Carlo simulation. The density has the largest contribution of approximately 50%, followed by the velocity with 25%. The third largest contributor is mass with 9%. The drag coefficient has the smallest effect with only 1.3%. If we group these results into three categories, we obtain the following:

- *Environmental dispersions* have 50% contribution.
- *Initial state dispersions* have 37% contribution.
- *Vehicle parameter dispersions* have 10% contribution.

All of the dispersions are not controllable by a designer. They are called noise factors as defined in Section 2.1. How we improve a design without controlling the noise is shown in the following section.

4.4 ROBUST DESIGN FOR THE TRAJECTORY SIMULATION

In Section 2.1, we define the quality characteristic for the LifeSat vehicle as the ability to follow the desired trajectory and land on target. The more we deviate from the target, the higher the quality loss (Sec. 2.1.1). Although the vehicle is designed to land on target, noise factors cause a variation in the landing position. The fundamental principle of robust design is to improve the quality of a product by minimizing the effect of the causes of variation without eliminating the causes [Phadke, 1989]. What we want is a measurement of quality during the design and development phase. We also want to obtain this particular information with a minimum of effort or time.

In this section, we use the signal-to-noise ratio η as the measurement for quality. This is a single number which represents the sensitivity of the system's function to noise factors. In a robust design project, the signal-to-noise ratio is maximized. We further show the difference in the use of the standard deviation.

4.4.1 Designing Experiments for Robust Design

Assume that a robust design problem for the LifeSat vehicle is available for which a designer has the three control factors. These control factors are the vehicle mass, initial velocity, and flight path angle. Assume further that the designer wants to find the best parameter values to make the vehicle land on target for average performance and also to obtain the least variation around this target. From previous information the designer knows that the mass can vary between 1460 kh and 1660 kg, the velocity between 9846.5

m/s and 10046.5 m/s, and the flight path angle between -5.78 deg and -5.98 deg. These values represent the bounds. The ranges are small but sufficient to show the method.

If we have three factors and select three levels for each factor, the best suitable orthogonal array is L_9. For the factor levels, we select the lower bound, upper bound, and middle value. In Table 4.10, these levels are presented for each factor.

Table 4.10 - Factor Levels for Robust Design Project

Factor	Level		
	1	2	3
Vehicle Mass (kg)	1460.0	1560.0	1660.0
Velocity (m/s)	9846.5	9946.5	10046.5
Flight Path Angle (deg)	-5.78	-5.88	-5.98

Assigning the three factors to the first three columns of L_9, we obtain nine experiments with factor combinations as shown in Table 4.11. Note that this is a change from previous studies where we use the last columns of the orthogonal array. This choice does not influence the principal results.

Table 4.11 - Experiments for Robust Design Project

Experiment	Mass (kg)	Velocity (m/s)	Flight Path Angle (deg)
1	1460.0	9846.5	-5.78
2	1460.0	9946.5	-5.88
3	1460.0	10046.5	-5.98
4	1560.0	9846.5	-5.88
5	1560.0	9946.5	-5.98
6	1560.0	10046.5	-5.78
7	1660.0	9846.5	-5.98
8	1660.0	9946.5	-5.78
9	1660.0	10046.5	-5.88

Each of these nine experiments is simulated using the same noise factor dispersion as shown in Table 4.7. Our control factors are noise factors, too. The nine experiments of L_9 are called inner array. We still use the orthogonal array L_{27} to represent and disperse the nine noise factors. This is called the outer array. To obtain one result for one experiment of L_9, we simulate 27 experiments of L_{27}.

4.4.2 Evaluation of Results from Robust Design Study Using Signal-to-Noise Ratio, Standard Deviation, and Mean

Previously, the target value for the geodetic latitude was 33.6 deg. We calculate the signal-to-noise ratios (SNs) for three new target values (T_1, T_2, T_3) for the geodetic latitude, which are

- T_1 = 33.7 deg,
- T_2 = 33.5 deg, and
- T_3 = 33.3 deg.

By Equation (2.4), the value of the signal-to-noise ratio depends on the target value. We calculate the mean and the standard deviation of the geodetic latitude, which are independent from the target. The results of the simulation of the experiments are shown in Table 4.12.

Table 4.12 - Experiments for Robust Design Project

Experiment	Geodetic Latitude				
	μ	σ	SN (T_1)	SN (T_2)	SN (T_3)
1	33.572	0.219	11.78	***12.58***	9.07
2	33.692	0.215	***13.16***	10.70	6.94
3	33.817	0.211	12.23	8.34	5.04
4	33.755	0.209	13.14	9.57	5.98
5	33.870	0.205	11.37	7.42	4.32
6	33.092	0.246	3.63	6.34	9.73
7	33.933	***0.200***	10.19	6.40	3.55
8	33.178	0.237	4.80	7.90	11.33
9	33.311	0.232	6.83	10.37	***12.49***

The range for the mean value varies from 33 deg to 34 deg. Note that each value is the average of the 27 experiments from the outer (noise) array. The standard deviation ranges between 0.2 and 0.25. The values of the signal-to-noise ratio depend on the target. The highlighted numbers represent the best designs with respect to the standard deviation and the signal-to-noise ratio. We observe that the smallest standard deviation does not correspond with the highest signal-to-noise ratios since the mean value is far away from one of the targets. For target T_1, experiment 2 has approximately the same signal-to-noise ratio as experiment 4. Although the mean value is closer at the target, the variation is larger around this mean compared to the variation in experiment 4. The signal-to-noise ratio combines the two measures of variation and meeting the target.

How can we now find a better design from this information? We compare three different strategies.

- Take the experiment with the smallest variation and adjust the mean to the desired target value.
- Find the factor levels with the least variation and run a verification test using these levels.
- Do the same as in the second strategy and then adjust the mean on target.

To adjust the mean on target, we use the initial value for the geodetic latitude and change it by the value of the bias between mean and target. We have not introduced the initial

geodetic latitude as a design parameter since we only expect a bias in the output of the geodetic latitude with a change of its initial value. We further use only T_1 and T_3 as the targets to be investigated.

To study the average output sensitivity of a factor on one level—e.g., on level 1—we calculate the mean of the response with all experiments when the factor is on level 1. The factor velocity is on level 1 at experiments 1, 4, and 7. The average of the geodetic latitude is (33.57 + 33.76 + 33.93)/3 = 33.75. Similarly, we calculate the mean values for levels 2 and 3 and for the other factors. The results are shown in Table 4.13 for the mean of the geodetic latitude, the standard deviation, and two of the signal-to-noise ratios. The three control factors are denoted by A, B, and C. We want to refer to the factor levels as A_1, for factor A on level 1, etc.

Table 4.13 - Level Means for System Performance

	Level Means for Latitude (deg)		
Factor	1	2	3
A: *Vehicle Mass (kg)*	33.694	33.572	33.474
B: *Velocity (m/s)*	33.753	33.58	33.407
C: *Flight Path Angle (deg)*	33.281	33.586	33.873

	Level Means for Standard Deviation		
Factor	1	2	3
A: *Vehicle Mass (kg)*	***0.215***	0.220	0.223
B: *Velocity (m/s)*	***0.209***	0.219	0.230
C: *Flight Path Angle (deg)*	0.234	0.219	***0.205***

	Level Means for SN Ratio (T_1)		
Factor	1	2	3
A: *Vehicle Mass (kg)*	***12.39***	9.38	7.27
B: *Velocity (m/s)*	***11.70***	9.78	7.57
C: *Flight Path Angle (deg)*	6.74	11.05	***11.26***

	Level Means for SN Ratio (T_3)		
Factor	1	2	3
A: *Vehicle Mass (kg)*	7.02	6.68	***9.12***
B: *Velocity (m/s)*	6.20	7.53	***9.09***
C: *Flight Path Angle (deg)*	***10.04***	8.47	4.30

When we choose T_1 as the target, from the first table, we observe that level 1 for A, level 1 for B, and level 2 for C have average values closest to the target of 33.7 deg. But none of these values is exactly on target. In the second table, we show the level means for the standard deviation. These level means are calculated as before using the corresponding values for each level and factor. Again we observe the smallest standard deviations for factors A_1, B_1, and C_3. This is also the case when we observe the third table for the signal-to-noise ratio of T_1. Thus we conclude that, for a target of T_1 = 33.7 deg, the factor combination A_1, B_1, and C_3 results in the best performance with respect to the geodetic latitude. But if the target changes to T_3 = 33.3 deg, each of the previous levels is on its lowest value for the signal-to-noise ratio. Now the factor combination A_3, B_3, and C_1 is expected to result in the best performance. Therefore, we have shown that a combined representation of deviation from the target *and* standard deviation is preferable. This is implied in the signal-to-noise ratio.

In Figures 4.10 and 4.11, the average sensitivity of geodetic latitude and its standard deviation is depicted. The values are the same as those in Table 4.13. Since level 2 for all factors corresponds to a design with target T = 33.6 deg, the average values for the geodetic latitude also approximate this target.

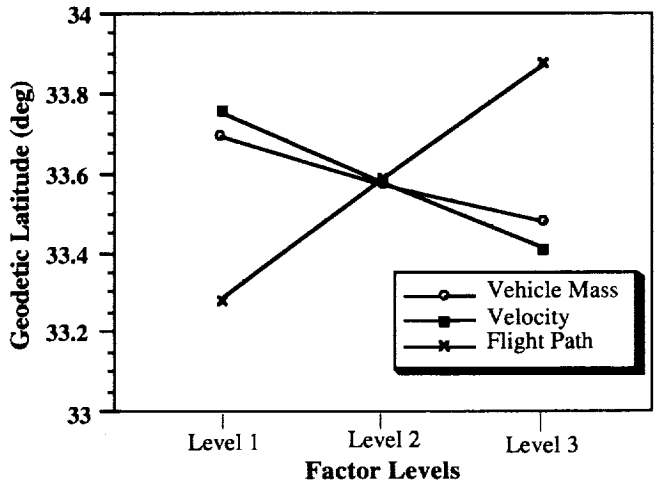


Figure 4.10 - Average Sensitivity of Geodetic Latitude to Factor Levels

As can be seen in Figure 4.10, the flight path angle has the highest differences between the three levels and opposite direction in the effect on the geodetic latitude. Mass and velocity have equal influence on the output. These results are different from the results obtained from the analysis of variance. Each level now is the mean value for the dispersion of the factor. Although the lines in Figure 4.10 are linear, we do not know the behavior between two levels.

We perform another analysis of variance for the mean of the geodetic latitude, the results of which are shown in Table 4.14. For the three factors mass (factor 1), velocity (factor 2), and flight path angle (factor 3), we obtain results corresponding to the trends shown in Figure 4.10

The vehicle mass contributes the smallest value (9%) to the variation. Approximately twice as much contributes the velocity, and the flight path angle is responsible for two-thirds of the total variation. These values confirm the magnitude of slopes of the curves in Figure 4.10, but we cannot identify the sign of the slope with ANOVA. The variation due to noise, which is contained in the error sum of squares, is small compared to the variation owing to factor level changes. With the application of ANOVA, we obtain more information about the sensitivity of the output than by calculating the average performance only. We depict the sensitivity of the standard deviation in Figure 4.11, which is the graphical representation of the second table in Table 4.13. The same could also be done for the signal-to-noise ratio.

Table 4.14 - ANOVA Table for the Variation of Geodetic Latitude

DESCRIPTION OF DATA

Number of points:	9
Mean:	33.5800
Standard deviation:	0.3123
Variance:	0.0975
ssTot:	0.7802
Sum:	302.2200
Sum of squares:	10149.3281

Interval mean ±1*st.dev.:	(33.2677, 33.8923)
Interval mean ±2*st.dev.:	(32.9554, 34.2046)
Interval mean ± 3*st.dev.:	(32.6431, 34.5169)

ANOVA FOR THE DISPERSIONS

Source:	f	SS	V	F	% Perc.	Confidence
Factor 1:	2	0.0723	0.0361	74.0000	9.14%	98.67%
Factor 2:	2	0.1797	0.0898	184.0000	22.90%	99.46%
Factor 3:	2	0.5273	0.2637	540.0000	67.46%	99.82%
Error:	2	0.0010	0.0005			
Total:	8	0.7803				

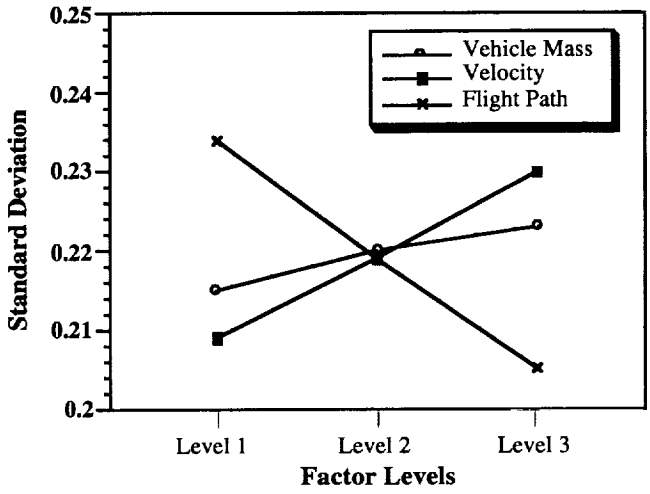


Figure 4.11 - Average Sensitivity of Standard Deviation to Factor Levels

In Figure 4.11, the average sensitivity to the standard deviation is shown. The flight path angle has the largest influence, and the smallest value for the standard deviation is obtained at level 3 (C_3). The slopes are opposite compared to their influence on the mean.

Taking the experiment with the smallest variation and adjusting the mean on target is one of the strategies to improve the design The experiment with the smallest standard deviation is experiment 7 with factor levels A_3, B_1, and C_3. As shown in Table 4.12, the deviation from the target mean of $T_1 = 33.7$ is (33.933 - 33.7) deg = 0.233 deg. We assume that the initial geodetic latitude behaves like a scaling factor since a change in this value results in an almost proportional change in the value for the final geodetic latitude. According to the deviation from the target, we change the initial geodetic latitude from 44.2 deg to (44.2 - 0.233) deg = 43.967 deg and perform one simulation, the verification experiment, again with factor levels A_3, B_1, and C_3. We obtain the following results which represent the mean, the standard deviation, and the signal-to-noise ratio:

μ = 33.6998 deg,
σ = 0.200, and
SN = 13.819.

This result has the mean on target ($T_1 = 33.7$ deg) and the standard deviation is unchanged. The signal-to-noise ratio is higher than all the previous ratios with a value of SN = 13.819; hence, this is an improved design with higher quality than before.

For the second strategy, we select the factor levels with the highest average signal-to-noise ratio. The selection of factor levels with the smallest average standard deviation is equivalent in this case. These factor levels are the ones marked in Table 4.13 (second and third table), which is level 1 for factor A (A_1) that has a signal-to-noise ratio of SN = 12.39 and a standard deviation of $\sigma = 0.215$. These average values are preferable with respect to our quality characteristics compared to lower values of the signal-to-noise ratio and higher values for the standard deviation on levels A_2, and A_3. Similarly, we select level 1 for factor B (B_1), and level 3 for factor C (C_3). For the verification experiment using the selected levels A_1, B_1, and C_3, we obtain the following results from the simulation with orthogonal arrays:

μ = 34.12 deg,
σ = 0.195, and
SN = 6.705.

The mean for the geodetic latitude has a value of $\mu = 34.12$ deg and is (34.12 - 33.7) deg = 0.42 deg away from the desired target. Thus, the signal-to-noise ratio is very low although the standard deviation is small. The combination of factor levels with the lowest average standard deviation has resulted in an even lower value of $\sigma = 0.195$. We see how important it is to verify the suggested factor levels. If we especially take the factor C at level C_3 with an average mean value for the geodetic latitude of $\mu = 33.87$ deg, this brings us to the new mean of $\mu = 34.12$ deg. Thus, level C_2 could have resulted in a better value. But now we need to adjust the mean on the target as suggested in the third

strategy, by changing the initial latitude according to the bias of 0.42 deg to (44.2 - 0.42) deg = 43.78 deg. We obtain the following values from the simulation:

μ = 33.70 deg,
σ = 0.1946, and
SN = 14.037.

This is obviously the best design we get from the analysis of only nine experiments. The settings of the parameters result in the smallest standard deviation, which is $\sigma = 0.1946$, and the highest signal-to-noise ratio with a value of SN = 14.04. Our most robust design to all the effects of the noise factors has the following values for the design parameters by using the suggested factor levels A_1, B_1, and C_3:

Vehicle mass	= 1460 kg,
Velocity	= 9846.5 m/s, and
Flight path angle	= -5.98 deg.

We recognize that we have to use the initial geodetic latitude as an adjustment factor to get the mean on target. Without the ability to adjust the mean on target, we would choose the factor levels which give the highest signal-to-noise ratio for one experiment. This is experiment 2 in Table 4.12 with a signal-to-noise ratio of SN = 13.16.

As mentioned, we obtain all of these results and information with only nine experiments. Of course, we need to perform 27 simulations in the outer array to simulate the effects of the noise factors. This is done for each of the nine experiments (Table 4.11) where we change the levels of design parameters. This is a total of 27 x 9 = 243 flight simulations, which is one quarter (25%) of the number of Monte Carlo simulations (if we use 1000 samples) necessary to evaluate only one design. The time saved with orthogonal arrays by using 27 experiments is larger than 95% of the time used for Monte Carlo simulations. Therefore, simulations based on orthogonal arrays seems to be qualified to substitute for Monte Carlo simulations.

4.5 WHAT HAS BEEN PRESENTED AND WHAT IS NEXT?

In Section 4.1, we compare the landing range with footprints obtained from Monte Carlo simulation and orthogonal array based simulation. We compare statistics and examine the influence of experiments in the orthogonal array. In Section 4.2, we assess the statistical confidence in the orthogonal array simulations. Analysis of variance is applied in Section 4.3, and results from Monte Carlo simulation are compared with those of orthogonal array simulation. With the robust design study in Section 4.4, we show how the robustness of a product is easily improved. The use of the signal-to-noise ratio is verified representing in a single number what is otherwise represented by the mean and the standard deviation.

In summary, we find the following answers to previous questions:

- The use of orthogonal arrays for simulation is an alternative to Monte Carlo simulation.

- We establish the confidence for a reduced number of simulations with statistical tests and comparison of results.

- The signal-to-noise ratio is a measurement for quality and robust design which enables a designer to improve a product with a small number of experiments.

- By using ANOVA, we are easily able to estimate the factor influences on output variations.

In the next chapter, we change the task of robust design to find the best factor levels. By using a compromise DSP, we obtain continuous values for the factors but not levels. This enables a designer to further improvements in the design.

CHAPTER 5

ROBUST DESIGN USING A COMPROMISE DSP

The compromise decision Support Problem, DSP, is a decision model that supports design decisions in the early stages. In this chapter, we develop the mathematical formulation for a compromise DSP for the LifeSat model. Multiple objectives are formulated and solutions are obtained for different priorities for the objectives. Exploration of design space is presented as a useful tool during the initial phase of designing. Four different scenarios are investigated. We use three different starting points to validate the solutions.

5.1 THE COMPROMISE DSP AND LEXICOGRAPHIC MINIMIZATION

Compromise DSPs are used to model engineering decisions involving multiple trade-offs. In this chapter, we show how to apply such decision models in robust design. We show how different scenarios are formulated and solved numerically. Further, we reflect on which quality aspects we want to focus in our design.

Given

An alternative to be improved through modification.
Assumptions used to model the domain of interest.
The system parameters:

n	number of system variables
p+q	number of system constraints
p	equality constraints
q	inequality constraints
m	number of system goals
$g_i(\underline{X})$	system constraint function: $g_i(\underline{X}) = C_i(\underline{X}) - D_i(\underline{X})$
$f_k(d_i)$	function of deviation variables to be minimized at priority level k for Preemptive case.

Find

X_i $\qquad i = 1, \ldots, n$

d_i^-, d_i^+ $\qquad i = 1, \ldots, m$

Satisfy

System constraints (linear, nonlinear)

$g_i(\underline{X}) = 0$; $\qquad i = 1, \ldots, p$

$g_i(\underline{X}) \geq 0$; $\qquad i = p+1, \ldots, p+q$

System goals (linear, nonlinear)

$A_i(\underline{X}) + d_i^- - d_i^+ = G_i$; $\qquad i = 1, \ldots, m$

Bounds

$X_i^{min} \leq X_i \leq X_i^{max}$; $\qquad i = 1, \ldots, n$

$d_i^- , d_i^+ \geq 0$; $\qquad i = 1, \ldots, m$

$(d_i^- \cdot d_i^+ = 0$; $\qquad i = 1, \ldots, m)$

Minimize

Preemptive (lexicographic minimum)

$\mathbf{Z} = [f_1(d_1^-, d_1^+), \ldots, f_k(d_i^-, d_i^+)]$

Figure 5.1 – Mathematical Form of a Compromise DSP

As stated in Chapter 1, the compromise DSP is a multiobjective programming model that we consider to be a hybrid formulation [Bascaran et al., 1987; Karandikar and Mistree, 1991; Mistree et al., 1992]. It incorporates concepts from both traditional Mathematical Programming and Goal Programming (GP). In the compromise formulation, the set of system constraints and bounds defines the *feasible design space* and the sets of system goals define the *aspiration space*. For feasibility, the system constraints and bounds must be satisfied. A *satisficing* solution, then, is that feasible point which achieves the system goals as far as possible. The solution to this problem represents a trade-off between that which is desired (as modeled by the aspiration space) and that which can be achieved (as modeled by the design space).

The mathematical formulation of a compromise DSP is presented in Figure 5.1.

Each goal A_i has two associated deviation variables d_i^- and d_i^+ which indicate the deviation from the target [Mistree et al., 1992]. The range of values of these deviation variables depends on the goal itself. The product constraint $d_i^+ \cdot d_i^- = 0$ ensures that at least one of the deviation variables for a particular goal will always be zero. If the problem is solved using a vertex solution scheme (as in the ALP-algorithm [Mistree et al., 1992]), then this condition is automatically satisfied. Goals are not equally important to a decision maker. To effect a solution on the basis of preference, the goals may be rank-ordered into priority levels. We should seek a solution which minimizes all unwanted deviations. There are various methods of measuring the effectiveness of the minimization of these unwanted deviations. The lexicographic minimum concept is necessary to the solution of our problem. The lexicographic minimum is defined as follows [Ignizio, 1985]:

LEXICOGRAPHIC MINIMUM Given an ordered array $\mathbf{f} = (f_1, f_2, \ldots, f_n)$ of non-negative elements f_k's, the solution given by $f^{(1)}$ is preferred to $f^{(2)}$ if

$$f_k^{(1)} < f_k^{(2)}$$

and all higher order elements (i.e. $f_1, \ldots, f_{k-1}$) are equal. If no other solution is preferred to **f**, then **f** is the lexicographic minimum.

As an example, consider two solutions, $\mathbf{f}^{(r)}$ and $\mathbf{f}^{(s)}$, where

$$\mathbf{f}^{(r)} = (0, 10, 400, 56)$$

and

$$\mathbf{f}^{(s)} = (0, 11, 12, 20).$$

In this example, note that $\mathbf{f}^{(r)}$ is preferred to $\mathbf{f}^{(s)}$. The value 10 corresponding to $\mathbf{f}^{(r)}$ is smaller than the value 11 corresponding to $\mathbf{f}^{(s)}$. Once a preference is established, then all higher order elements are assumed to be equivalent. Hence, the deviation function **Z** for the preemptive formulation is written as

$$\mathbf{Z} = [\, f_1(\mathbf{d}^-, \mathbf{d}^+), \ldots, f_k(\mathbf{d}^-, \mathbf{d}^+) \,] .$$

For a four-goal problem, the deviation function may look like

$$\mathbf{Z}(\mathbf{d}^-, \mathbf{d}^+) = [\,(d_1^- + d_2^-), (d_3^-), (d_4^+)\,]$$

In this case, three priority levels are considered. The deviation variables d_1^- and d_2^- have to be minimized preemptively before variable d_3^- is considered and so on. These priorities represent rank; that is, the preference of one goal over another. No conclusions can be drawn with respect to the amount by which one goal is preferred or is more important than another.

5.2 ROBUST DESIGN USING THE COMPROMISE DSP

Phadke [Phadke, 1989] provides guidelines for selecting quality characteristics, but like many others he uses only one characteristic per problem. We believe that it is difficult, if not impossible, to find one single metric for assessing the quality of a process (for example, [Karandikar and Mistree, 1992]). In our opinion, since there are multiple objectives to be satisfied in design, there must be multiple aspects to quality. Therefore, we assert that quality loss is dependent on a number of quality characteristics with different importances. Quality involves trade-off and the desired quality cannot always be achieved. Related to our view is a discussion by Otto and Antonsson [Otto and Antonsson, 1991] on the possibilities and drawbacks of the Taguchi method. They offer some extensions (e.g., the inclusion of design constraints). Otto and Antonsson also note that the Taguchi method is single-objective. The question is raised: how should a trade-off be handled? But it remains unanswered. We show how trade-off is handled, however; namely, through compromise DSPs [Mistree et al., 1992] as explained in this section.

If we have an analytical equation characterizing the relationship between the response and the noise and control factors, then we are able to determine the mean and variance of the response as functions of the mean and variances of the noise and control factors through Taylor series expansion (Sec. 2.2.3). An in-depth discussion about analytical determination is found in [Bras, 1992] with solutions to concurrent robust design. In [Lautenschlager et al., 1992; Bras, 1992], solution strategies to model quality into decision models are shown using a compromise DSP, Taguchi's quality design, and Suh's [Suh, 1991] design axioms. We want to incorporate the technique of robust design into a compromise DSP. Robust design involves values for the mean, the standard deviation, and the signal-to-noise ratio. Now the values for mean and standard deviation are determined through experiments and not through analytical relationships.

5.2.1 A Compromise DSP Formulation for the Robust Design of the LifeSat Trajectory

Problem Statement: Assume that we want to design a LifeSat vehicle that has the same control factors (design variables) as in Section 4.4.1. These control factors are vehicle mass (VM), initial velocity (IV), and flight path angle (FP). As before (Sec. 4.4.2), we want to find the best factor values to make the vehicle land on target, at 33.6 deg geodetic latitude, and to have the least variation around this target. By representing this in a value for the signal-to-noise ratio, it is desired that this value becomes 15 dB. It is also desired

to obtain a small value for the maximum acceleration. We would like to obtain a value of 50 m/s^2, but the maximum value should not be larger than 140 m/s^2. We loosen the bounds on the factors (Sec. 4.4.1) to have larger freedom for the design. The lower and upper bounds on mass are 1400 kg and 1800 kg, respectively. The bounds on velocity are 9.8 km/s and 10.2 km/s, and the bounds on the flight path angle are -6.2 deg and -5.5 deg. The initial geodetic latitude is kept constant at 44.2 deg.

We want to compare the signal-to-noise ratio on the one side to the mean and standard deviation on the other side in order to observe the influence on the results. The signal-to-noise ratio includes all quality characteristics in a single objective, while the mean on target and small standard deviation involve two objectives.

The first step in a compromise DSP is the word formulation in terms of the keywords introduced in Section 1.3.2. From the problem statement we obtain the following:

Given

- a model for the LifeSat trajectory simulation
- signal-to-noise ratio for geodetic latitude
- mean of geodetic latitude
- standard deviation for geodetic latitude
- mean value for maximum acceleration
- target values for mean, standard deviation, and signal-to-noise ratio
- target value for maximum acceleration
- upper and lower bounds on the design variables
- maximum acceleration limit

Find

- Independent system variables
  the value for initial velocity IV
  the value for vehicle mass VM
  the value for flight path angle FP

Satisfy

- System constraints
  the constraint on the maximum acceleration
- System goals
  meet target value for mean of geodetic latitude
  meet (low) target value of standard deviation
  meet (high) target value for signal-to-noise ratio
  meet (low) target value for maximum acceleration
- Bounds on system variables
  lower and upper bounds on all variables

Minimize

- the deviations from the target values

In order to obtain a mathematical formulation of the preceding problem, we need to derive the equations for the signal-to-noise ratio, mean, standard deviation, and maximum acceleration with respect to the information given.

To calculate the signal-to-noise ratio of the geodetic latitude, we need to do the simulations with orthogonal arrays and to calculate the Mean Squared Deviation as defined in

Equation (2.4) in Section 2.1.2. The signal-to-noise ratio depends on the dispersions of nine noise factors (Sec. 4.3.1) which are now fixed. Three factors are also our control factors F_i and, therefore, are independent system variables. We have a total of three factors; and the signal-to-noise ratio, denoted by SNGL, is defined by

$$SNGL = f_1\ (F_1, F_2, F_3) = 15.0 \tag{5.1}$$

From experience gained through previous studies, we set our target signal-to-noise ratio to $T_{SNGL} = 15$ dB. Equation (5.1) is modeled as a goal in the compromise DSP by

$$\frac{SNGL}{15.0} + d_1^- - d_1^+ = 1.0\ . \tag{5.2}$$

A similar functional relationship exist for the mean and the standard deviation. The mean for the geodetic latitude is obtained by

$$MGL = f_2\ (F_1, F_2, F_3) = 33.6, \tag{5.3}$$

and is formulated as a goal for the compromise DSP. The target for the mean of the geodetic latitude is $T_{MGL} = 33.6$ deg; hence, we get

$$\frac{MGL}{33.6} + d_2^- - d_2^+ = 1.0\ . \tag{5.4}$$

The standard deviation is obtained as

$$SDGL = f_3\ (F_1, F_2, F_3) = 0.0. \tag{5.5}$$

Our desired target is a standard deviation of zero, SDGL = 0.0. This goal for the compromise DSP is calculated from

$$SDGL + d_3^- - d_3^+ = 0.0\ . \tag{5.6}$$

The maximum acceleration MAXA is obtained as

$$MAXA = f_4\ (F_1, F_2, F_3) = 50.0. \tag{5.7}$$

Our desired target is a maximum acceleration of 50 m/s^2. This goal for the compromise DSP is calculated from

$$\frac{MAXA}{50.0} + d_4^- - d_4^+ = 1.0\ . \tag{5.8}$$

In order to prevent too much heating of the vehicle, a constraint on the maximum acceleration MACC is introduced. This constraint limits the maximum acceleration to a value of 140 m/s^2; therefore, we get

$$MACC = g_1\ (F_1, F_2, F_3) \leq 140.0. \tag{5.9}$$

The bounds on the system variables (control factors) are introduced in the problem statement. The preceding leads to the mathematical formulation of the problem as given in Figure 5.2.

Given

- a model of the LifeSat trajectory;
- signal-to-noise ratio for geodetic latitude; SNGL
- mean of geodetic latitude; MGL
- standard deviation for geodetic latitude; SDGL
- maximum acceleration; MAXA
- target for signal-to-noise ratio; $T_{SNGL} = 15.0$
- target for mean of geodetic latitude; $T_{MGL} = 33.6$
- target for standard deviation; $T_{SDGL} = 0.0$
- target for max. acceleration; $T_{MAXA} = 50.0$
- upper limit for max. acceleration; $MACC \leq 140.0$
- constants;
  - the initial latitude = 44.2 deg;
  - the dispersions of noise factors;

Find

- the values for unknown elements:
  the value for initial velocity IV
  the value for vehicle mass VM
  the value for flight path angle FP

Satisfy

- the constraint:

$$MACC / 140.0 \leq 1.0 \qquad (5.9)$$

- the signal-to-noise ratio goal:

$$SNGL / T_{SNGL} + d_1^- - d_1^+ = 1.0 \qquad (5.2)$$

- the mean on target goal:

$$MGL / T_{MGL} + d_2^- - d_2^+ = 1.0 \qquad (5.4)$$

- the standard deviation goal:

$$SDGL + d_3^- - d_3^+ = 0.0 \qquad (5.6)$$

- the maximum acceleration goal:

$$MAXA / T_{MAXA} + d_4^- - d_4^+ = 1.0 \qquad (5.8)$$

- the bounds:
  $9.8 \leq IV \leq 10.2$
  $1.4 \leq VM \leq 1.8$
  $-6.2 \leq FP \leq -5.5$

Minimize

the deviation function;

$$\mathbf{Z} = [\, f_1(\mathbf{d}^-, \mathbf{d}^+), \ldots, f_k(\mathbf{d}^-, \mathbf{d}^+) \,].$$

Figure 5.2 – Mathematical Formulation of a Compromise DSP for the Robust Design of the LifeSat Trajectory

The formulation in Figure 5.2 has four goals. This means we are interested in the effect of multiple quality characteristics of the process. The preemptive deviation function **Z** is not explicitly specified (yet) in Figure 5.2. In Section 5.3, we specify several deviation functions, each associated with a different design scenario.

In Figure 5.2, we show normalized goals. The goals are normalized by dividing them with their associated targets or vice versa in order to obtain values for the deviation variables ranging between zero and one. The normalization facilitates comparisons between the goals. The mathematical formulation given in Figure 5.2 is solved using the ALP-algorithm in DSIDES [Mistree et al., 1992].

Before we explore the behavior of the model, we will discuss another aspect of the design model. In this section, Equations (5.1) to (5.9) contain functions (f_i, g_1) which are not given as explicit algebraic expressions of the control factors F_1, F_2, and F_3. The nature of functions is discussed in greater detail in Papalambros [Papalambros and Wilde, 1991]. Not all factor dispersions of the nine noise factors are constant during the calculations; only six of these nine factors are assumed to be constants. As stated earlier, three of the noise factors are the system variables or control factors. These are the vehicle mass, the initial velocity, and the flight path angle. Their dispersions are given in percentage of the mean; therefore, the dispersions change with the mean values. Our goal is to meet the target values of mean, standard deviation, signal-to-noise ratio, and maximum acceleration. The calculations of these values are only estimates of the real population. We show in Sections 4.1 and 4.2 that the estimations based on Monte Carlo and orthogonal array simulation have only small differences. All the functions f_1 to f_4 are internally evaluated by the simulation model using orthogonal arrays. The function values are based on statistical calculations using the 27 experiments as before (Chapter 4). Therefore, we have a nonlinear programming problem which could also be nondeterministic. If we use a Monte Carlo simulation to evaluate the function values, these values do not change from run to run if all inputs are the same and the random number generator always starts with the same seed number. By using different seed numbers the function output would vary, even for the same input. This will cause difficulties for the solution of the problem. Using Equation (5.9) as an example, where $g_1 \leq 140.0$, g_1 evaluated with different seed numbers may have an estimated value of 139.0 the first time and a value of 141.0 the second time for the same input factors. The function will be nonsmooth, and we are not able to solve the problem with nonlinear programming techniques. If we use orthogonal arrays, all of the experiments are designed in advance and, therefore, the function output is a deterministic one for a given input. Of course this deterministic value is an estimation, but the behavior of the function is not influenced by this. Since the input variables are continuous, the output of functions f_1 to f_4, and g_1 is continuous and differentiable and can be solved with the ALP-algorithm. Any gradients calculated in this algorithm are based on the results of the simulation with 27 experiments for a function evaluation.

In the following section, we explore the LifeSat model. This provides information for the designer and helps to understand the model behavior.

5.2.2 Model Exploration

All nonlinear optimization algorithms are sensitive to the quality of the starting points. One optimization run can best identify one local minimum. We generate contour plots in order to obtain a better understanding of the behavior of the LifeSat model. These plots also serve to validate the results obtained by solving the model using the ALP-algorithm

[Mistree et al., 1992]. Each contour plot consists of 200 equally distributed points generated by the "XPLORE" feature within the DSIDES system. The "XPLORE" feature in DSIDES facilitates parametric studies of a design space. The time spent to evaluate these 200 points is approximately 10 minutes. Since we use the orthogonal array L_{27} for the simulation, a total of 200*27 = 5400 simulations has to be done. In Figure 5.2, we present two contour plots of velocity versus flight path angle. The vehicle mass is kept constant at a value of m = 1560 kg. The contours represent the values for the standard deviation in the left picture and the signal-to-noise ratio in the right picture.

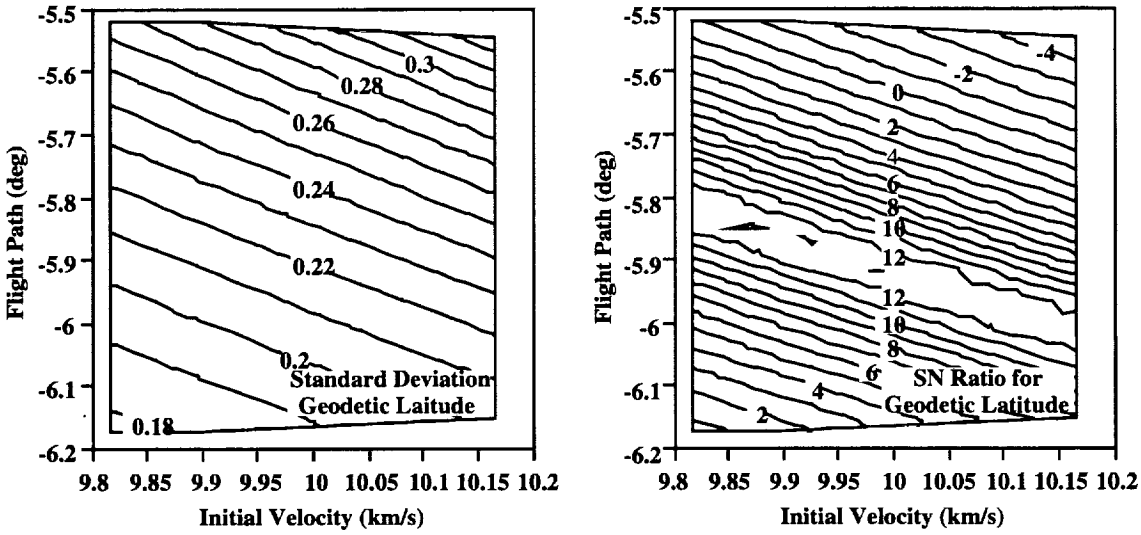


Figure 5.3 – Contour Plots of Initial Velocity vs. Standard Deviation and SN Ratio for Geodetic Latitude

In the left part of Figure 5.3, the standard deviation is depicted for initial velocity and for flight path angle. The value of mass is fixed to 1560 kg. We observe a linear relation between the three parameters (velocity, standard deviation, and flight path angle). For a speed below 9.9 km/s and a flight path angle below - 6.1 deg, we achieve a very small standard deviation of approximately $\sigma = 0.16$. For high values of speed and flight path angle, the standard deviation is approximately twice as high as the best value. Although we want a small variation of the output, we also want to achieve the target which is 33.6 deg for geodetic latitude. The values for the signal-to-noise ratio in the right portion of Figure 5.3 provide this information. We identify a band of high values for the signal-to-noise ratio which range from -6.0 deg to - 5.8 deg for the flight path angle and over the whole range of the velocity. Thus, we conclude that for landing on target we cannot obtain a standard deviation smaller than 0.2 for this particular value of mass. The signal-to-noise ratio will be larger than 12.0 because this is the value on the border of the band. The steepness cannot be identified but, from the trend, we suggest a step hill around the best values. When we have only two design variables—initial velocity and flight path angle—we have to assure that their mean values are within the band.

For the results shown in Figure 5.4, we keep the initial velocity constant at 9946.5 km/s and vary the vehicle mass and the flight path angle. We observe the same trends as in

Figure 5.3 when the mass is held constant. We identify a smaller range of the standard deviation, which varies between 0.18 and 0.28. The relative difference between the lower and the upper value of the mass is larger (25% difference) compared to the relative difference of the velocity values (4% difference). The sensitivity of the standard deviation and the signal-to-noise ratio to flight path angle is larger for a particular value of mass when compared to the velocity.

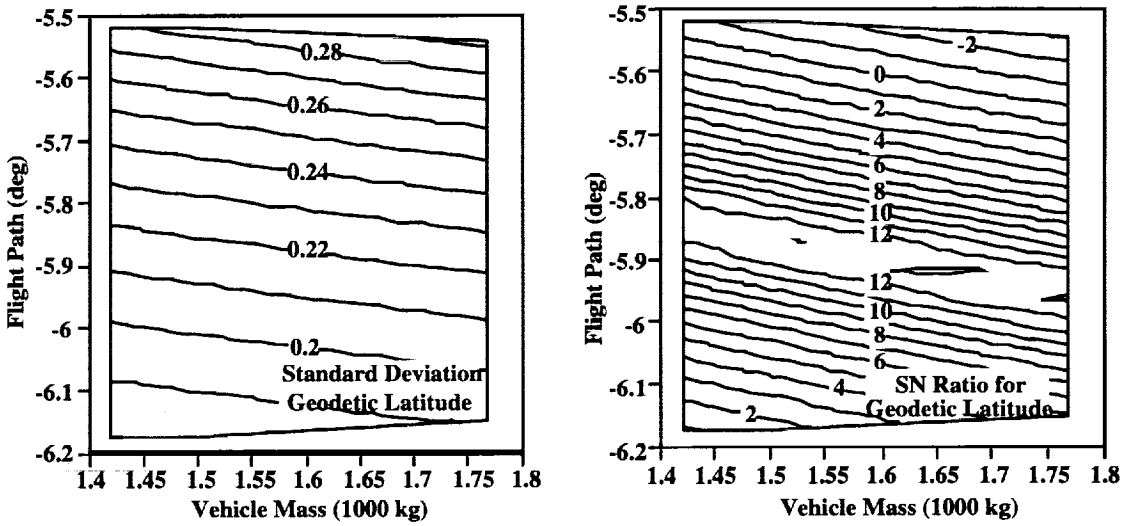


Figure 5.4 – Contour Plots of Vehicle Mass vs. Standard Deviation and SN Ratio for Geodetic Latitude

Use of the XPLORE feature for design space exploration is extensively demonstrated in [Smith, 1992]. We obtain important information within a short period of time. From Figures 5.3 and 5.4 we identify ranges of values for two system variables for good designs, which will be useful when we are choosing starting points for the optimization. The major observation we can glean from Figures 5.3 and 5.3 is that several equally good solutions exist within the identified band; hence, several different designs will satisfy our goals.

In the following section, we investigate several scenarios in order to solve the compromise DSP as stated in Section 5.2.1. Different scenarios means that we change the order of priority for the four goals. We use the observations from this section to explain the results.

5.2.3 Solving the Compromise DSP for Four Scenarios

In Section 5.2.1, we developed the mathematical formulation for the compromise DSP. We identify three design variable; namely, initial velocity, vehicle mass, and flight path angle. For solution validation, we use three different starting points. The first starting point is a point within the ranges for the design variables; the second and third points are points on the bounds of the variables. By changing the priority levels, we exercise the

behavior of the model for different priorities in the goals. Our highest priority is to get the mean of the geodetic latitude on the target. If this is achieved as much as possible, we want to minimize the standard deviation. Finally, the maximum acceleration should be minimized. The signal-to-noise ratio goal is not used because it is represented in the first two priorities. In the following table (Table 5.1), we present four different scenarios, each of which is characterized by the priority order of the deviation variables. For the first scenario according to the concept of lexicographic minimization, the deviation function is written as

$$Z(\mathbf{d}^-, \mathbf{d}^+) = [(d_2^- + d_2^+), (d_3^- + d_3^+), (d_4^- + d_4^+)].$$

Similarly, the deviation function is written for the remaining scenarios. All scenarios and corresponding priorities are shown in Table 5.2. In the second scenario, the signal-to-noise ratio goal has the highest priority, thus presenting our quality characteristic for Robust Design. Therefore, we assign the acceleration goal to the second priority. In the third scenario, we changed the priority order of mean and standard deviation goals. Finally, Scenario 4 represents the acceleration goal with the highest priority. All scenarios and corresponding priorities are shown in Table 5.1.

Table 5.1 - Scenarios for the Design of the LifeSat Vehicle

Scenario	Priority 1	Priority 2	Priority 3
1	$d_2^- + d_2^+$	$d_3^- + d_3^+$	$d_4^- + d_4^+$
2	$d_1^- + d_1^+$	$d_4^- + d_4^+$	$d_2^- + d_2^+$
3	$d_3^- + d_3^+$	$d_2^- + d_2^+$	$d_4^- + d_4^+$
4	$d_4^- + d_4^+$	$d_2^- + d_2^+$	$d_3^- + d_3^+$

As can be seen in Table 5.1, we assign every goal once to the highest priority and, hence, expect to identify designs which satisfy each of the goals. For each of the scenarios, we use three different starting points which are given in Table 5.2.

Table 5.2 - Starting Points for Design Variables

	Starting Point 1	Starting Point 2	Starting Point 3
Velocity (m/s)	9950.0	10,200.0	9800.0
Mass (kg)	1750.0	1400.0	1800.0
Flight Path Angle (deg)	- 5.90	- 6.20	- 5.50

In the following, we explain the results for each starting point and for all four scenarios. For the first starting point, the results are shown in Table 5.3. We present the final values for the three design variables and for the four goals. Although we use only three priority levels, the value of the fourth goal is still calculated. Refer to Table 5.2 to identify the priority for each presented goal.

Table 5.3 - Results with Starting Point 1

	Scenario 1	Scenario 2	Scenario 3	Scenario 4
Velocity (m/s)	9939.0	9830.0	10,044.1	10,200.0
Mass (kg)	1758.4	1787.3	1800.0	1800.0
Flight Path Angle (deg)	- 5.944	- 5.891	- 6.20	- 5.50
Mean μ (deg)	33.600	33.608	34.098	31.209
Std. Deviation σ	0.215	0.212	0.193	0.376
SN Ratio (dB)	13.18	13.29	5.422	- 7.684
Acceleration (m/s^2)	132.43	131.35	145.21*	94.81

*Constraint violation acceptable.

For scenario 1 as shown in Table 5.3, we find that in the solution the mean value for the geodetic latitude is exactly on target. The standard deviation is minimized on the second priority level and a value of $\sigma = 0.215$ is obtained. The values for the design variables have changed slightly compared to the initial values. For the second scenario where the signal-to-noise ratio has the highest priority, we have larger changes in the design variables. The speed is decreased and the mass is increased. Although the mean value is not exactly on target as before, the standard deviation could be decreased and the signal-to-noise ratio could be increased. These results are driven by the signal-to-noise ratio and are slightly better. In the third scenario, the standard deviation is minimized with highest priority. As we can see, this is achieved with a value of $\sigma = 0.193$ but the mean is far away from the target. Therefore, the signal-to-noise ratio has a small value. If we are only interested in minimizing the variation of an output this would be a good design. For the design variables, the velocity has increased and the mass is approaching the lower bound. The flight path angle is exactly on the lower bound. Compared to the contour plot in Figure 5.3, we should be able to obtain values for the smallest standard deviation of approximately $\sigma = 0.18$. Since the contour is very flat in this area, the convergence criteria for termination are satisfied. Hence, tighter stopping criteria would lead to a better solution. In scenario 4, the maximum acceleration is minimized. This means that we want to find a smaller value for the highest acceleration which occurs during the flight. This goal is achieved within one iteration for all three starting points. Velocity and mass are going to their upper bound, the flight path angle is going to the lower bound. Even starting on the opposite bounds leads to this solution. Therefore, the value of 94.8 m/s^2 is the smallest value for the maximum acceleration we obtain within the given bounds. This goal results in bad values for the other goals. When we are far away from the target, the standard deviation is extremely high; hence, the signal-to-noise ratio is very low. This is not a design that we really want to have.

In Table 5.4, we present the results for starting point 2. The results for our goals are approximately the same for all four scenarios as for starting point 1. We identify the major difference in the values for the mass for the first three scenarios. Because we started at the lower bound, the final values are still close to it. In scenario 2, the velocity remains on the upper bound and the results are not as good as before. We are fsrther away from the target and the standard deviation is higher. Scenario 4 has the same results as for starting point 1.

Table 5.4 - Results with Starting Point 2

	Scenario 1	Scenario 2	Scenario 3	Scenario 4
Velocity (m/s)	9952.1	10,200.0	10,020.0	10,200.0
Mass (kg)	1584.6	1414.6	1440.0	1800.0
Flight Path Angle (deg)	- 5.897	- 5.984	- 6.13	- 5.50
Mean μ (deg)	33.604	33.635	34.024	31.209
Std. Deviation σ	0.217	0.223	0.204	0.376
SN Ratio (dB)	13.081	12.76	6.512	- 7.684
Acceleration (m/s^2)	131.11	133.486	142.410*	94.81

* Constraint violation acceptable.

In Table 5.5, the results for the third starting point are presented. As for starting point 1, we obtain good results for our goals but different values for the design variables. Although the mass in scenario 3 is on the upper bound, we have a low value for the standard deviation. According to the contour plots in Figure 5.3 and the stopping criteria, this result is a solution since the velocity is on its lower bound.

Table 5.5 - Results with Starting Point 3

	Scenario 1	Scenario 2	Scenario 3	Scenario 4
Velocity (m/s)	9981.3	9800.0	9800.0	10,200.0
Mass (kg)	1800.0	1770.0	1800.0	1800.0
Flight Path Angle (deg)	- 5.981	- 5.856	- 6.062	- 5.50
Mean μ (deg)	33.600	33.617	34.097	31.209
Std. Deviation σ	0.215	0.212	0.190	0.376
SN Ratio (dB)	13.18	13.26	5.457	- 7.684
Acceleration (m/s^2)	133.38	130.58	141.2*	94.81

* Constraint violation acceptable

Our conclusions for the different scenarios and starting points are as follows:

- The two goals of getting the mean on target and minimizing the standard deviation in this order result in approximately the same solutions for the goals as using the single goal of maximizing the signal-to-noise ratio.
- No unique solution exists. For different starting points we obtain equally good results for the goals, but the values for the design variables are different and, hence, we have different designs.
- The order of priority for the quality aspects of mean and standard deviation has to be like that in scenario 1 to obtain a good design solution. As demonstrated in scenario 3, the opposite order results in a bad design which is reflected in the value of the signal-to-noise ratio.

In the following, we show two figures representing the values of the deviation function for the first three priorities (Fig. 5.5) and the values for the design variables (Fig.

5.6) for each iteration. This is done for three different starting points and for scenario 1 as shown in Table 5.1. Priority levels 1 and 2 represent our quality characteristics in scenario 1. The highest priority is to get the mean of the geodetic latitude on target, the second priority is the standard deviation goal, and the third priority is the acceleration goal.

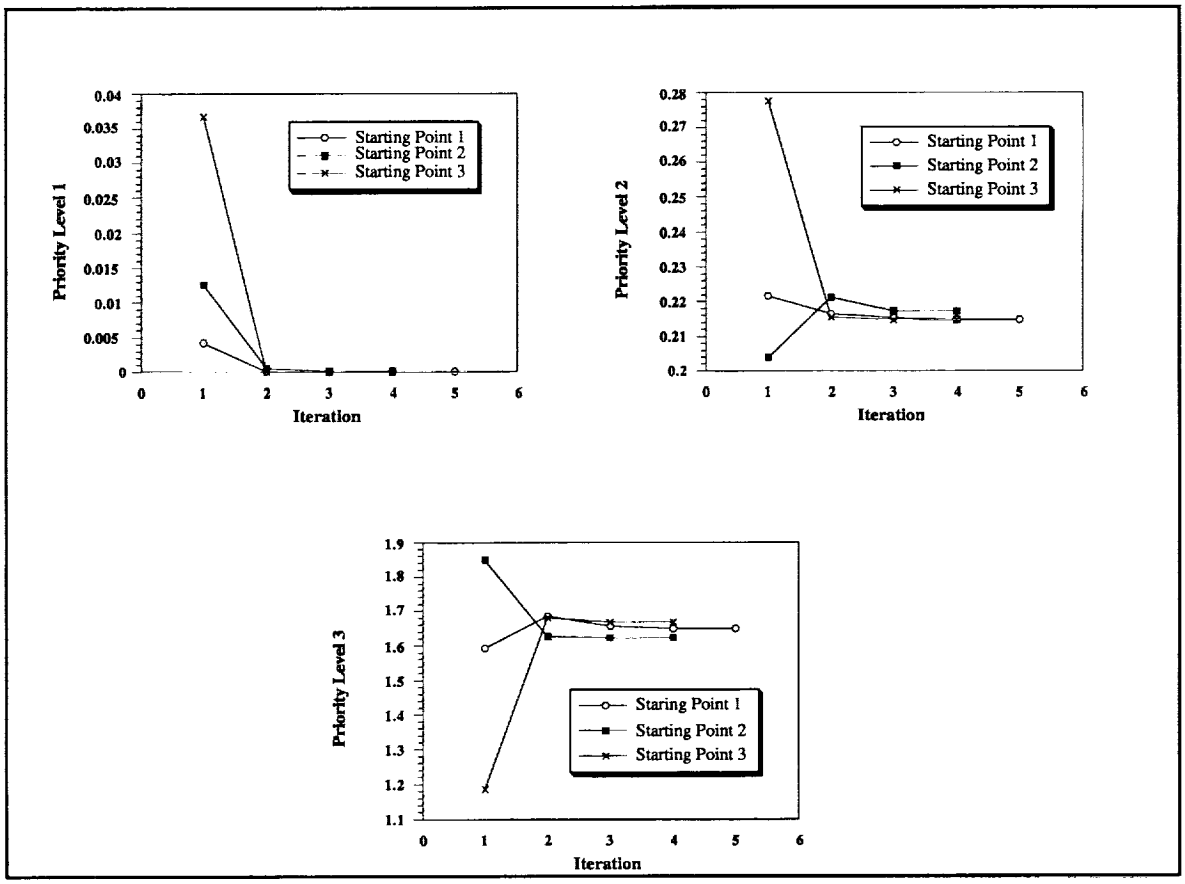


Figure 5.5 - Convergence of Deviation Function for Three Priority Levels

From Figure 5.5, we identify convergence for each of the priority levels to approximately the same solution for all three starting points. The number of iterations for starting points 2 and 3 is four, and for starting point 1 is five. Priority level 1 goes to zero; that means we are on target. On priority level 2, by minimizing the standard deviation we obtain a value around 0.215 (Tables 5,4, 5.5, and 5.6). The deviation function for the acceleration goal (priority level 3) has final values near 1.65; i.e., a value around 133 m/s^2 for acceleration. Convergence is achieved within four to five iterations; but the first iteration, which is driven by priority level 1, almost leads to the final solution.

As indicated earlier in this section, the design variables do not have the same solution values for different starting points. In the pictures of Figure 5.6, the complete range for the variables is used as scale.

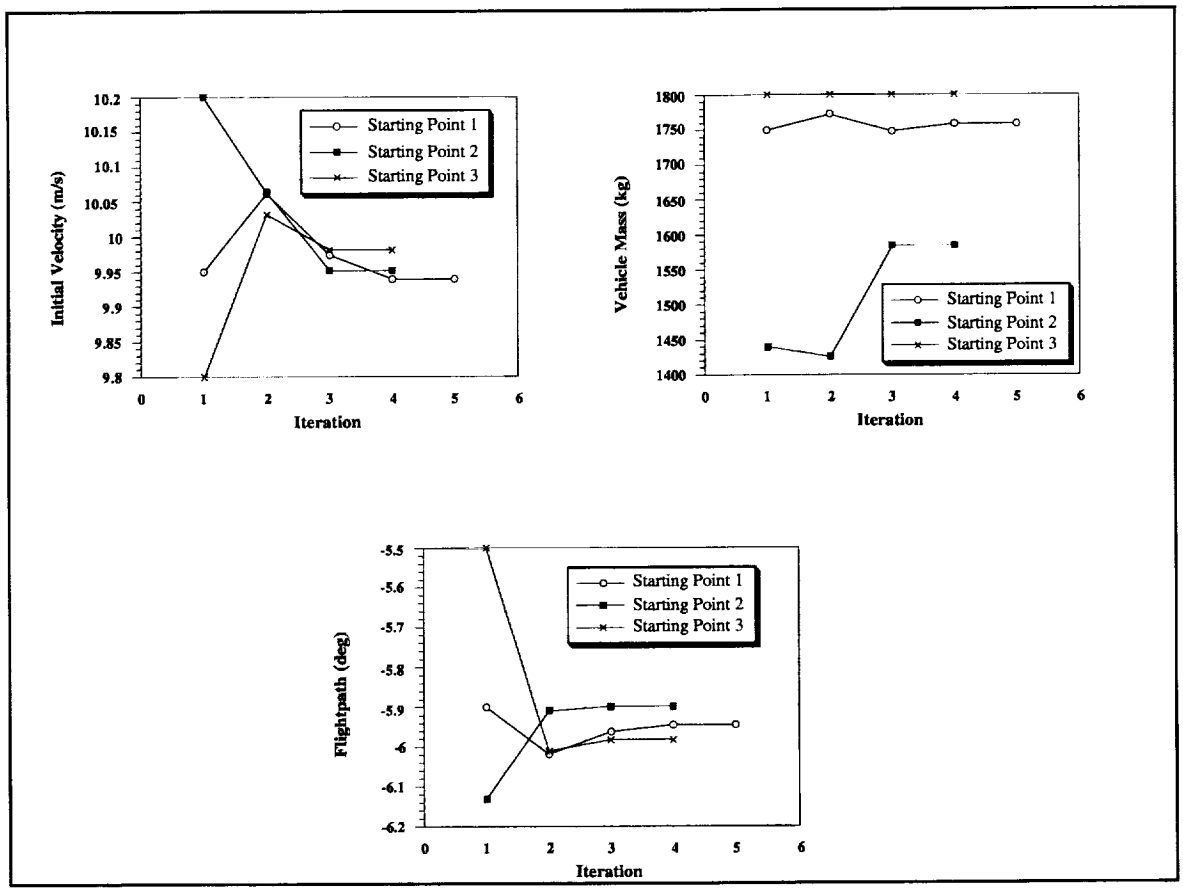


Figure 5.6 - Convergence of Design Variables for Three Different Starting Points

The initial velocity has different solution values but in the same area. The vehicle mass converges to three different final values covering a large range within the bounds. For starting point 3 the mass remains constant on the initial value. Only velocity and flight path angle vary in this case, but we still obtain a good design. In Table 5.6, we represent these three different but equivalent designs, which are reflected in a similar value for the signal-to-noise ratio. Values from 13.08 to 13.18 can be assumed to be quite similar.

Table 5.6 - Comparison of Equivalent Designs

	Design 1	Design 2	Design 3
Velocity (m/s)	9939.0	9952.1	9981.3
Mass (kg)	1758.4	1584.6	1800.0
Flight Path Angle (deg)	- 5.944	- 5.897	- 5.981
Signal-to-Noise Ratio	13.18	13.08	13.18

The depicted figures represent only scenario 1. For all other scenarios we obtain approximately the same results in the sense that we need only a few iterations for convergence but obtain different designs for different starting points. This is exactly what we expected from the contour plots, where we have identified a band with equally high values for the

signal-to-noise ratio. We could obtain these values with many different combinations of the design variables.

5.3 WHAT HAS BEEN PRESENTED AND WHAT IS NEXT?

In this chapter, we introduce the compromise DSP as a tool to support design decisions in the early stages of design. We present the word formulation and derive the mathematical formulation of a compromise DSP for the LifeSat vehicle. Our goal is to find dimensions of the design variables which give a robust design of the vehicle on highest priority. We compare the signal-to-noise ratio goal with a two goal formulation of mean on target goal and standard deviation goal. We study the model behavior by minimizing the standard deviation and by minimizing the maximum acceleration. Before solving the problem, we explore the design space for the standard deviation and the signal-to-noise ratio. Observations made from the contour plots are helpful to explain and validate results. Design space exploration is also useful when we deal with highly nonlinear problems. We draw the following conclusions from this study using the compromise DSP:

- The use of the "XPLORE" feature in DSIDES is a helpful tool to identify the behavior of the design space. Many conclusions can be made from contour plots which are obtained in a small amount of time (10 to 15 minutes for 200 points).

- The LifeSat vehicle model has multiple solutions for the goals. We identify some of these solution within a few iterations. The time spent is approximately 2 to 3 minutes, and the obtained solution is satisficing with respect to goals.

- For robust design goals, the use of the signal-to-noise ratio is as good as using two goals for mean and standard deviation. Compared to Section 4.4.2, we are not dependent on factor levels but find the best values for the design variables from a continuous range.

- For future applications, more design constraints could be introduced into the model to make it more realistic. By having multiple goals and constraints, the compromise DSP is a capable tool to support required design decisions .

In the last chapter, we review the presented work, critically evaluate it, and draw conclusions. All questions posed in Chapter 1 are answered, and a discussion of future research areas is presented.

CHAPTER 6

CLOSURE

The principal goal of this work is identified in Chapter 1. We review this goal along with the questions posed in Section 1.4. The current work, as presented in this report, is critically evaluated to see if these questions are answered. We address advantages and limitations of the proposed approach using orthogonal arrays for simulation purposes. The discussion of areas for future work is followed by closing remarks.

6.1 REVIEW OF THE GOAL FOR THE REPORT, RELATED QUESTIONS, AND CONCLUSIONS

We identify the principal goal of this work along with the focus to develop and implement applications of Taguchi's quality engineering in Section 1.4. Several questions worthy of investigation are posed in Section 1.4 that concern the capability of the application and confidence in the results. The presented work is critically evaluated to see if the questions are answered in this report.

- ***Is it possible to reduce through substitution a large number of Monte Carlo simulations by Orthogonal Array experiments?***

  In Chapter 1, we introduce the need for simulation reduction by substitution of Monte Carlo simulation with a more efficient technique. Orthogonal arrays bthat are discussed in Section 2.3 are identified to fulfill this task. Aspects of how to use orthogonal arrays for simulation are described in Section 2.4. Based on simulation results obtained in Section 2.4.2 and Chapter 4, we answer the question with a clear *yes*.

- ***What is the statistical confidence level that the results from Monte Carlo simulations and orthogonal array based simulations are equal?***

  Statistical confidence is established in Section 4.2.3. Based on statistical tests, we identify that results obtained for both simulation methods are equal in 97.5% of the cases. Furthermore, we establish confidence through comparison of statistics parameters, such as mean and standard deviation in Sections 4.1 and 4.2. We admit that many additional tests are available to obtain confidence in the results, but this is beyond the scope of this work.

- ***How can we identify factor contributions to variation in system performance?***

  The method of evaluating factor effects is analysis of variance (ANOVA) as explained in Section 2.5. The use of ANOVA provides a measure for the factor contributions and a level of confidence for these contributions. ANOVA is applied in Section 4.3.2. Through comparison with results from Monte Carlo simulations in Section 4.3.1, we validate the use of ANOVA to obtain information about factor contributions while varying all factors simultaneously. We identify the contributions of environmental, initial state, and vehicle parameter dispersions to the variation of the geodetic latitude. In Section 4.4.2, we validate results about the sensitivity of the geodetic latitude to factor levels with ANOVA.

- ***Are we able to improve the quality of a design by using the signal-to-noise ratio?***

  The signal-to-noise ratio is being introduced in Section 2.1. Quality characteristics such as meeting the target and making the product insensitive to noise are employed in a single number. In Section 4.4.2, we discuss in detail how the signal-

to-noise ratio is used to obtain a robust design. Only if both quality characteristics are satisfied is the value for the signal-to-noise ratio sufficiently high. In Chapter 5, the signal-to-noise ratio is one of the main goals in a compromise DSP. In this chapter, we use the signal-to-noise ratio as a continuous measure for quality compared to average quality in Section 4.4.2 with traditional robust design. From all results (Chapters 4 and 5), we clearly obtain an indication for improved quality of designs.

- ***Is it possible to predict the factor levels for best system performance by using robust design?***

  This question is being answered in Section 4.4.2. By using only nine experiments along with the L_9 orthogonal array, we show with a simple example how to calculate average system performance for each factor on each level. From average performance we identify the factor levels which we believe will result in better system performance. The verification experiment has to be performed to verify the proposed factor levels. This question is closely related to the previous question and, since we are able to predict the factor levels, the next step is again the employment of the compromise DSP as shown in Chapter 5.

- ***What are the advantages and limitations of the presented approach?***

  The new philosophy involved in this approach of employing Taguchi's quality engineering techniques is the use of orthogonal arrays to simulate the effects of noise factors. Without having an inner array to evaluate control factor settings, we evaluate one particular design and only the effects of noise factors. This particular design for the LifeSat vehicle is evaluated by simulating the trajectory and is characterized by the variation in landing position and performance parameters. We show that using information obtained from 27 simulations based on orthogonal arrays requires 40 times less simulations than Monte Carlo simulation technique and maintains accuracy. We get a feeling for the time savings if we look at the robust design formulation of the trajectory using the compromise DSP. In the following, we present the average time spent to simulate the trajectory:

  - 2 to 3 seconds for 27 simulations of the trajectory.
  - 3 to 5 minutes to perform complete robust design study with three to four system variables using the compromise DSP.
  - 10 to 15 minutes for design space exploration (as shown in Section 5.2) to detect best candidate starting points for optimization and information about the design space.

  If we do the calculations when the trajectory is simulated with Monte Carlo simulation and 1106 (Chapter 1) samples, the same robust design study would take approximately 3.5 hours as compared with 3 to 5 minutes. The estimated time savings is approximately 97.5%.

  One limitation of our approach at this stage is the dependency of factor levels on the observed system. In Section 4.2.2, we show how the estimated values for the

standard deviations of all landing position and performance parameters change with the σ-levels of the factors. Another limitation involves the determination of factor levels. If factors are dependent on each other, we have to modify columns of the standard orthogonal arrays in order to represent these dependencies. Recall the dependency of angle of attack and drag coefficient as explained in Section 3.2. Only because of small changes in the angle of attack are we able to assign three levels to the drag coefficients. For larger changes, a total of nine levels for the angle of attack is required: three for each level of the angle of attack.

We observe that all of the questions posed in Chapter 1 are answered through the current work. Our conclusion is that using orthogonal arrays for the simulation is a valuable and time saving method which obtains more accurate results than the Monte Carlo simulation. The performance of a design can be evaluated within a short period of time and can be improved by using design models; e.g., a compromise DSP. By using the signal-to-noise ratio, we have a single number indicator for the quality of the design. The analysis of results can be done easily with ANOVA.

We hope that we have opened another domain of application for Taguchi's quality engineering and for what is involved with the use of orthogonal arrays. A domain dependent insight and understanding into the new philosophy are gained while doing this work. Some possible areas of future work are outlined in the next section.

6.2 FUTURE WORK

This work covers only a small part of what is possible that is being investigated. A large body of future work is identified. In the near future, the focus will be on issues related to the current LifeSat model. For long-term research, the focus is more general. The following issues need to be addressed:

- *Orthogonal Arrays.* We have focused only on three-level orthogonal arrays. Do we obtain results with a higher confidence when we use four- or five-level arrays? The question to be posed is the following: Are simulations of noise factors sufficient with the use of only three levels? But what is our choice for the factor levels with more levels? Can we weigh the output proportionate to the probability of input parameter values? Should we use the same levels (three) but higher order arrays L_{27} --> L_{81} to represent more factor combinations? These are issues that concern orthogonal arrays and factor levels.

- *Model Refinement.* In the work presented here, we have not focused on meeting the available impact area, although we met the target and minimized the variation around this target. Especially, tolerances on initial state parameters and the atmospheric density have to be adjusted. Not all results correspond to the documented ones [McCleary, 1991]. Therefore, the model can be refined.

- *Implementation of Axiomatic Design.* A connection among Taguchi's quality design, the compromise DSP, and Suh's design axioms [Suh, 1991] has already been shown in [Lautenschlager et al., 1992; Bras, 1992]. When the signal-to-noise ratio and variance are calculated analytically, Bras [Bras, 1992] shows how we can obtain a robust design involving parameter design and tolerance design. For

simulation models like the LifeSat model, this combination of several design tools or design models is desirable.

- *Industrial Implementation.* The work has to be applied in design practice. Further comparative case studies—e.g., with nonlinear models—are essential to validate this approach. A computer environment needs to be developed in which we can select suitable orthogonal arrays. Existing capabilities of DSIDES to support human design decisions need to be used extensively in order to obtain solutions to real-life problems when a trade-off between multiple objectives has to be included.

6.3 CLOSING REMARKS

Finally, the main emphasis of research and development has been to develop tools that can be used to find answers to a set of specifications and requirements. We suggest that in the future we should foster the recognition and employment of quality engineering tools for the whole life-cycle of a product.

REFERENCES

Addelman, S. (1962). "Orthogonal Main Effect Plans for Assymetrical Factorial Experiments" *Technometrics*, Vol. 4, pp. 21-46.

Bascaran, E., F. Mistree and R.B. Bannerot (1987). "Compromise: An Effective Approach for Solving Multi-objective Thermal Design Problems," *Engineering Optimization,* Vol. 12, No. 3, pp. 175-189.

Baumeister, T. (1986). *Marks' Standard Handbook for Mechanical Engineers,* McGraw Hill, New York.

Bogner, A. (1989). *Statistical Estimation for Nonlinear Systems.* Charles Stark Draper Laboratory.

Bose, R.C. and K.A. Bush (1952). "Orthogonal Arrays of Strength Two and Three," *Annals of Mathematical Statistics,* Vol. 23, pp. 508-524.

Bras, B.A. (1992). *Foundations for Designing Decision-Based Design Processes,* Ph.D. Dissertation, University of Houston, Houston, TX.

Bras, B.A. and F. Mistree (1991). "Designing Design Processes in Decision-Based Concurrent Engineering," Proceedings SAE Aerotech '91, SAE Publication SP-886, Paper No. 912209, Long Beach, CA, SAE International, pp. 15-36.

Bras, B.A. (1992) *Foundations for Designing Decision-Based Design Processes,* Ph.D. Dissertation, University of Houston, Houston, TX.

Bronstein, I.N. and K.A. Semendjajew (1987). *Taschenbuch der Mathematik,* Vetlag Harri Deutsch, Frankfurt/Main.

Casella, G. and R.L. Betget (1990). *Statistical Inference,* Wadsworth & Brooks/Cole Statistics/Probability Series, Wadsworth & Brooks/Cole Advanced Books & Software, Pacific Grove, California.

Dey, A. (1985). *Orthogonal Fractional Factorial Design,* Halsted Press, New York.

Donaghey, C.E. (1989). *Digital Simulation.* University of Houston, Industrial Engineering Dept.

Dunn, O.J. and V.A. Clark (1987). *Applied Statistics: Analysis of Variance and Regression,* Wiley Series in Probability and Mathematical Statistics, John Wiley & Sons, NY.

Ignizio, J.P. (1985). *Introduction to Linear Goal Programming,* Quantitative Applications in the Social Sciences, J.L. Sullivan and R.G. Niemi Ed., Sage University Papers, Beverly Hills, CA.

Karandikar, H. and F. Mistree (1991). "Designing Composite Material Pressure Vessel for Manufacture: A Case Study for Concurrent Engineering," Vol. 18, No. 4, 1991, pp.

235-262.

Kempthorne, O. (1979). *The Design and Analysis of Experiments,* Robert E. Krieger Publishing Co., NY.

Lautenschlager, U., S.O. Erikstad, B. Bras and F. Mistree (1992). "Quality Design Decision Models," Fourth National Symposium on Concurrent Engineering, Washington, DC, pp. 423-441.

Marks, L.S., (1951) *Mechanical Engineers' Handbook*, Fifth Edition, McGraw-Hill Book Company, Inc., NY.

Masuyama, M. (1957). "On Different Sets for Constructing Orthogonal Arrays of Index Two and of Strength Two," *Rep. Statist. Appl. Res. Un. Jap. Sci. Eng.*, No. 32, pp. 27-34.

McCleary (1991). *Entry Monte Carlo Analysis Capability Using SORT.* McDonnell Douglas Space Systems Co.

Mistree, F., O.F. Hughes and B.A. Bras (1992). "The Compromise Decision Support Problem and the Adaptive Linear Programming Algorithm," in *Structural Optimization: Status and Promise,* M.P. Kamat Ed., AIAA, Washington, D.C.

Mistree, F., O.F. Hughes and H.B. Phuoc (1981). "An Optimization Method for the Design of Large, Highly Constrained, Complex Systems," *Engineering Optimization,* Vol. 5, No. 3, pp. 141-144.

Otto, K.N. and E.K. Antonsson (1991). "Extensions to the Taguchi Method of Product Design", *Third International Conference on Design Theory and Methodology,* L.A. Stauffer Ed., Miami, Florida, American Society of Mechanical Engineers, pp. 21-30.

Papalambros, P. and D. Wilde (1988). *Principles of Optimal Design.* Cambridge University Press, Cambridge.

Phadke, M.S. (1989). *Quality Engineering using Robust Design,* Prentice Hall, Englewood Cliffs, NJ.

Plackett, R.L. and J.P. Burman "The Design of Optimal Multifactorial Experiments," *Biometrika,* Vol. 33, pp. 305-325.

Press, W.H., B.P. Flannery, S.A. Teukolsky and W.T. Vetterling, (1988) *Numerical Recipes in C,* Cambridge University Press.

Rice, J.R. (1983). "Numerical Methods. Software and Analysis." McGraw-Hill, NY.

Ross, P.J. (1988). *Taguchi Techniques for Quality Engineering.* McGraw Hill, NY.

Roy, R. (1990). *A Primer on the Taguchi Method,* Competitive Manufacturing Series, Van Nostrand Reinhold, NY.

Seiden, E. (1954). "On the Problem of Constructing Orthogonal Arrays," *Annals of Mathematical Statistics.* Vol. 25, pp. 151-156.

Smith, W.F. (1992). *The Modeling and Exploration of Ship Systems in the early Stages of Decision-Based Design,* Ph.D. Dissertation, University of Houston, Houston, TX.

Sokal, R. and F.J. Rohlf (1981). *Biometrv,* W.H. Freeman & Co., San Francisco, CA.

Suh, N.P. (1990). *The Principles of Design,* Oxford University Press, NY.

Taguchi, G. (1987). *System of Experimental Design,* Kraus International Publications, NY.

Taguchi, G., E.A. Elsayed and T. Hsiang (1989). *Quality Engineering in Production Systems,* McGraw-Hill, New York, NY.

Tigges, M.A. (1992). *Life Sat Monte Carlo Dispersions and Simple Functional Relations.* NASA JSC, Informal Report.

Ullman, D.G. (1992). *The Mechanical Design Process,* McGraw-Hill, New York.

Zhou, Q-J., J.K. Allen and F. Mistree (1992) "Decisions Under Uncertainty: The Fuzzy Compromise Decision Support Problem", *Engineering, Optimization,* Vol. 20, pp. 21-43.

Appendix A

Flightsimulation Program Source Code

The simulation routines for the deorbiting LifeSat vehicle are included in this appendix. The flightsimulation consists of the main program (spaceflight) and several subroutines in FORTRAN. In a data-file (flightsim.dat), required flight data have to be provided by the user. The user can choose between flightsimulation using Monte Carlo simulation or using orthogonal array based simulation. The program creates output-files for results about position, performance parameters, and statistics. These results can be further analyzed by ANOVA programs, as provided in Appendix C.

```
      PROGRAM SPACEFLIGHT
C*********************************************************************
C
C       NAME:  SPACEFLIGHT
C
C       PURPOSE:  LIFE SAT TRAJECTORY SIMULATION WITH DISPERSED PARAMETERS
C
C       PROGRAMMER:  UWE LAUTENSCHLAGER
C
C       DATE:  2 MAY, 1992
C
C*********************************************************************
C
C       ARGUMENTS:
C
C*********************************************************************
C
C       LOCAL VARIABLES:
C
      CHARACTER*1 TAB
      INTEGER ZAEHL, RUNS, NDESV, IFLAG, NSKIP, COL(40), FLIGHTTIME
      INTEGER*4 IDUM
      REAL ACCMAX, ACCX, ACCY, ACCZ, AREF(3),
     &    AOT, AOTTOL, AOTDIS,
     &    CD(3), CDTOL(3), CDDIS(3),
     &    DENSINIT, DENS, DENSTOL, DENSDIS, DELTAT, DYNMAX,
     &    MASS, MASSTOL, MASSDIS,
     &    MEAN(5), VAR(5), STDEV(5),
     &    LRX, LRY, LRZ, LNX, LNY, LNZ,
     &    DESPX, DESPY, DESPZ, DEPOSX, DEPOSY, DEPOSZ,
     &    RADINIT, RADNEW, RADIUS, RANGE(2), DELRANGE(2), ROTAT,
     &    HS, HINIT, PI, EXPONENT,
     &    MUE, TETNEW, PHINEW, FLIPA, AZIMU, LONGIT, LATIT,
     &    LONGITOL, LATITOL, RADITOL, SPEEDTOL, AZIMUTOL, FLIPATOL,
     &    LONGIDIS, LATIDIS, RADIS,  SPEEDIS,AZIMUDIS, FLIPADIS,
     &    POSXDIS, POSYDIS, POSZDIS, SPEED, SPEEDINIT,
     &    VELX, VELY, VELZ,
     &    SPEEDX, SPEEDY, SPEEDZ,
     &    OA(81,40), LOWER(40), MIDDLE(40), UPPER(40),
     &    XDUM
      INTEGER NUOUT, NUINP
C
      COMMON /DISTRIB/IDUM
      COMMON /PERFORM/ ACCMAX, DYNMAX
C*********************************************************************
C
      NUINP = 10
      NUOUT = 13
C
      TAB = CHAR(9)
C
      IDUM = 1791
C
      OPEN(UNIT=NUINP, FILE = 'flightsim.dat',STATUS='OLD')
C
C -------------------------------------------------------
C
```

```
C    Skip first 5 lines of header information
C
     DO 10 J = 1,5
      READ (NUINP,*)
  10 CONTINUE
C
C ---------------------------------------------------------
C
C    Read flags and parameters from data file
C
     READ(NUINP,*) IFLAG
     READ(NUINP,*) NDESV
     READ(NUINP,*) NSKIP, RUNS
     READ(NUINP,*) SPEEDINIT
     READ(NUINP,*) MASS
     READ(NUINP,*) FLIPA, AZIMU
     READ(NUINP,*) LONGIT, LATIT
C
     DO 11 J = 1,6
      READ (NUINP,*)
  11 CONTINUE
C
     READ(NUINP,*) LONGITOL, COL(1)
     READ(NUINP,*) LATITOL,  COL(2)
     READ(NUINP,*) RADITOL,  COL(3)
     READ(NUINP,*) MASSTOL,  COL(4)
     READ(NUINP,*) DENSTOL,  COL(5)
     READ(NUINP,*) CDTOL(1), COL(6)
     READ(NUINP,*) FLIPATOL, COL(7)
     READ(NUINP,*) AZIMUTOL, COL(8)
     READ(NUINP,*) SPEEDTOL, COL(9)
C
     CLOSE(NUINP)
C

     IF (IFLAG.EQ.1) THEN
       IF (NDESV.LE.4) THEN
         RUNS = 10
       ELSE IF (NDESV.LE.13.AND.NDESV.GT.4) THEN
         RUNS = 28
       ELSE IF (NDESV.LE.40.AND.NDESV.GT.13) THEN
         RUNS = 82
       END IF
     END IF
C
C*********************************************************************
     IF (IFLAG.EQ.0) THEN
       OPEN (UNIT = 1, FILE = 'ranstat.out', STATUS = 'UNKNOWN')
       OPEN (UNIT = 2, FILE = 'ranpoint.out', STATUS = 'UNKNOWN')
     ELSE IF (IFLAG.EQ.1) THEN
       OPEN (UNIT = 1, FILE = 'ortstat.out', STATUS = 'UNKNOWN')
       OPEN (UNIT = 2, FILE = 'ortpoint.out', STATUS = 'UNKNOWN')
     END IF
       OPEN (UNIT = 3, FILE = 'altitude.out', STATUS = 'UNKNOWN')
C
     DO 20 J = 1, NSKIP
       XDUM = SRAN(IDUM)
```

```
 20  CONTINUE
C
C*********************************************************************
C
C  INITIALIZATION OF PARAMETERS AND VARIABLES
C
    PI = 3.1415927
C
C  TIME STEPS IN SECONDS FOR INTEGRATION
C
    DELTAT = 1.0
C
C  1. GRAVITATIONAL PARAMETERS
C
    RADINIT = 6370000.0
    RADNEW  = RADINIT + 121920.0
    MUE = 9.81*RADINIT**2
C
C  2. ATMOSPHERE
C
    DENSINIT = 1.2
    HS   = 8620.7
    HINIT   =  RADINIT
C
C  3. VEHICLE
C
C     MASS    = 1560.357
    AREF(1) = 3.3
    AREF(2) = 9.0898
    AREF(3) = 113.469
    AOT     = 0.0
    CL      = 0.0
    ROTAT   = 0.0

C
    FLIPA  = FLIPA*PI/180.0
    AZIMU  = AZIMU*PI/180.0
C
C*********************************************************************
C  DEFINE TOLERANCES OR 1-SIGMA LEVELS FOR ALL PARAMETERS
C
C  1. TOLERANCE IN MASS EQUALS 5%, UNIFORM
C
    MASSTOL = MASSTOL*MASS
C
C  2. 3-SIGMA DISPERSION IN DENSITY EQUALS 30%, NORMAL
C
    DENSTOL = DENSTOL* DENSINIT/3.0
C
C  3. TOLERANCE IN ANGLE OF ATTACK EQUALS 5%, UNIFORM
C
    AOTTOL = 5.0
C
C  4. 3-SIGMA DISPERSION OF VEHICLE Cd EQUALS 5%, BUT DEPENDS ON
C     THE MEAN AT THE GIVEN ANGLE OF ATTACK; CHUTE COEFFICIENTS
C     MEANS ARE CONSTANT, NORMAL
C
```

```
C     CDTOL(2) = 0.0275/3.0
C     CDTOL(3) = 0.04/3.0
C
C  5. 3-SIGMA DISPERSIONS OF REF. AREA OF CHUTES EQUAL 1%, NORMAL
C
C     AREFTOL(2) = 0.4418/3.0
C     AREFTOL(3) = 5.515/3.0
C
C  6. THE THE INITIAL STATE IS VARIED BY 5% OF THE INITIAL VALUES, NORMAL
C
      LONGITOL = LONGITOL
      LATITOL  = LATITOL
      RADITOL  = RADITOL
C
C  7. 3-SIGMA DISPERSIONS OF SPEED DIRECTIONS EQUAL 2% OF ANGLES, NORMAL
C
      FLIPATOL = FLIPATOL*FLIPA/3.0
      AZIMUTOL = AZIMUTOL*AZIMU/3.0
      SPEEDTOL = SPEEDTOL*SPEEDINIT/3.0
C
C**********************************************************************
C  IF IFLAG = 1, WHICH MEANS ORTHOGONAL ARRAYS ARE APPLIED, WE
C  DETERMINE LOWER, MIDDLE, AND UPPER LEVEL FOR EACH DISPERSED VARIABLE
C  FOR UNIFORM: LOWER = MEAN - TOL., MIDDLE = MEAN, UPPER = MEAN + TOL.
C  FOR NORMAL: LOWER = MEAN - SQRT(1.5)*SIGMA, MIDDLE = MEAN, UPPER = ...
C
      IF (IFLAG.EQ.1) THEN
C
C  1. DEFINE INITIAL POSITIONS; NORMALLY
C
       RADIUS = RADNEW
C
      CALL LEVELSNOR (LONGIT, LONGITOL, LOWER(COL(1)),
     &           MIDDLE (COL(1)), UPPER(COL(1)))
      CALL LEVELSNOR (LATIT,  LATITOL, LOWER(COL(2)),
     &           MIDDLE (COL(2)), UPPER(COL(2)))
      CALL LEVELSNOR (RADIUS,  RADITOL, LOWER(COL(3)),
     &           MIDDLE (COL(3)), UPPER(COL(3)))
C
C  2. DEFINE MASS; UNIFORMALLY
C
      CALL LEVELSUNI (MASS, MASSTOL, LOWER(COL(4)),
     &           MIDDLE(COL(4)), UPPER(COL(4)))
C
C  3. DEFINE DENSITY; NORMALLY
C
      CALL LEVELSNOR (DENSINIT, DENSTOL, LOWER(COL(5)),
     &           MIDDLE(COL(5)), UPPER(COL(5)))
C
C  4. DEFINE CD FOR ANGLE OF ATTACK; UNIFORMALLY
C
         CD(1) = (0.66512 + 0.67068)/2.0
         CDTOL(1) = CDTOL(1)*CD(1)/3.0
C
C  5. DEFINE DRAG-COEFFICIENT, NORMALLY
C
      CALL LEVELSNOR (CD(1), CDTOL(1), LOWER(COL(6)),
```

```
     &          MIDDLE(COL(6)), UPPER(COL(6)))
C
C  6. DEFINE INITIAL SPEED, NORMALLY
C
      CALL LEVELSNOR (FLIPA, FLIPATOL, LOWER(COL(7)),
     &          MIDDLE(COL(7)), UPPER(COL(7)))
      CALL LEVELSNOR (AZIMU, AZIMUTOL, LOWER(COL(8)),
     &          MIDDLE(COL(8)), UPPER(COL(8)))
      CALL LEVELSNOR (SPEEDINIT, SPEEDTOL, LOWER(COL(9)),
     &          MIDDLE(COL(9)), UPPER(COL(9)))
C
      CALL ORTARRAY (NDESV, LOWER, MIDDLE, UPPER, OA)
C
C
      END IF
C
C  MAIN LOOP FOR ALL SIMULATIONS
C
      ZAEHL = 0
C
      DO WHILE (ZAEHL.LE.(RUNS-1))
C
        ZAEHL = ZAEHL + 1
C
      RADIUS = RADNEW
C
      FLIGHTTIME = 0
C
      ACCMAX = 0.0
      DYNMAX = 0.0
C
      LRX = 0.0
      LRY = 0.0
      LRZ = 0.0
      LNX = 0.0
      LNY = 0.0
      LNZ = 0.0
C
      IF ((IFLAG.EQ.1).AND.(ZAEHL.GT.1)) THEN
      CALL POSIT (OA(ZAEHL-1,COL(1)), OA(ZAEHL-1,COL(2)),
     &  OA(ZAEHL-1,COL(3)), POSXDIS, POSYDIS, POSZDIS)
        MASSDIS = OA(ZAEHL-1,COL(4))
        DENSDIS = OA(ZAEHL-1,COL(5))
        CDDIS(1) = OA(ZAEHL-1,COL(6))
      CALL VELOC (OA(ZAEHL-1,COL(1)), OA(ZAEHL-1,COL(2)),
     &    OA(ZAEHL-1,COL(7)), OA(ZAEHL-1,COL(8)), OA(ZAEHL-1,COL(9)),
     &    SPEEDX, SPEEDY, SPEEDZ)
        AOTDIS  = AOT
C
      ELSE IF (IFLAG.EQ.0) THEN
C
C*********************************************************************
C
C  1. DISPERSE INITIAL POSITIONS NORMALLY
C
      CALL DISPERSNOR (LONGIT, LONGITOL, LONGIDIS)
      CALL DISPERSNOR (LATIT,  LATITOL,  LATIDIS)
```

```
      CALL DISPERSNOR (RADIUS, RADITOL,  RADIS)
C
      CALL POSIT (LONGIDIS, LATIDIS, RADIS, POSXDIS, POSYDIS, POSZDIS)
C
C  2. DISPERSE MASS UNIFORMALLY
C
      CALL DISPERSUNI (MASS, MASSTOL, MASSDIS)
C
C  3. DISPERSE DENSITY NORMALLY
C
      CALL DISPERSNOR (DENSINIT, DENSTOL, DENSDIS)
C
C  4. DISPERSE ANGLE OF ATTACK UNIFORMALLY
C
      CALL DISPERSUNI (AOT, AOTTOL, AOTDIS)
C
C  5. MODEL VEHICLE CD LINEAR TO ANGLE OF ATTACK
C    DISPERSE VEHICLE CD RELATED TO ANGLE OF ATTACK
C
      IF (AOTDIS.GE.AOT) THEN
        CD(1) = 0.66512 + 0.001112*AOTDIS
      ELSE
        CD(1) = 0.67068 - 0.001112*(5.0 - AOTDIS)
      END IF
C
      CDTOL (1)= 0.05*CD(1)/3.0
      CALL DISPERSNOR (CD(1), CDTOL (1), CDDIS(1))
C
C  6.  DISPERSE INITIAL SPEED NORMALLY
C
      CALL DISPERSNOR (FLIPA, FLIPATOL, FLIPADIS)
      CALL DISPERSNOR (AZIMU, AZIMUTOL, AZIMUDIS)
      CALL DISPERSNOR (SPEEDINIT, SPEEDTOL, SPEEDIS)
C
      CALL VELOC (LONGIDIS, LATIDIS, FLIPADIS, AZIMUDIS, SPEEDIS,
     &          SPEEDX, SPEEDY, SPEEDZ)

C
      END IF
C
C  MAKE A DRY RUN FOR THE MEAN AT FIRST
C
      IF (ZAEHL.EQ.1) THEN
      CALL POSIT (LONGIT, LATIT, RADNEW, POSXDIS, POSYDIS, POSZDIS)
      CALL VELOC (LONGIT, LATIT, FLIPA, AZIMU, SPEEDINIT,
     &          SPEEDX, SPEEDY, SPEEDZ)
        DENSDIS = DENSINIT
        MASSDIS = MASS
        CDDIS(1) = (0.66512 + 0.67068)/2.0
        AOTDIS  = AOT
      END IF
C
      VELX = SPEEDX
      VELY = SPEEDY
      VELZ = SPEEDZ
C
C  CALCULATE INITIAL POSITION STATE PARAMETERS IN POLAR COORDINATES
```

```
C
      CALL POLAR (POSXDIS, POSYDIS, POSZDIS, RADIUS, TETNEW, PHINEW)
C
C  MAIN LOOP FOR ITERATIONS AND ONE SIMULATION
C
      DO WHILE (RADIUS.GE.RADINIT)
C
      FLIGHTTIME = FLIGHTTIME + 1
C
      CALL  DIRECTION
     &     (SPEEDX, SPEEDY, SPEEDZ, LRX, LRY, LRZ, LNX, LNY, LNZ)
C
C  CALCULATION OF DENSITY
C
      EXPONENT = (HINIT - RADIUS)/HS
      DENS = DENSDIS*EXP(EXPONENT)
C
      SPEED = SQRT(SPEEDX**2 + SPEEDY**2 + SPEEDZ**2)
C
      CALL FORCES
     & (RADIUS, DENS, SPEED, MUE, MASSDIS, LRX, LRY, LRZ,
     &  LNX, LNY, LNZ, TETNEW, PHINEW, CDDIS(1), CL, AREF,
     &  ACCX, ACCY, ACCZ)
C
C  INTEGRATION OF FOR NEW POSITION AND VELOCITY
C
      DESPX  = ACCX*DELTAT
      DESPY  = ACCY*DELTAT
      DESPZ  = ACCZ*DELTAT
      DEPOSX = 0.5*ACCX*DELTAT**2
      DEPOSY = 0.5*ACCY*DELTAT**2
      DEPOSZ = 0.5*ACCZ*DELTAT**2
C
      POSXDIS = POSXDIS + DEPOSX + SPEEDX*DELTAT
      POSYDIS = POSYDIS + DEPOSY + SPEEDY*DELTAT
      POSZDIS = POSZDIS + DEPOSZ + SPEEDZ*DELTAT
      SPEEDX  = SPEEDX + DESPX
      SPEEDY  = SPEEDY + DESPY
      SPEEDZ  = SPEEDZ + DESPZ
C
      ROTAT = ROTAT + 25.0*DELTAT
C
C  NEW POSITION IN POLAR COORDINATES
C
      CALL POLAR (POSXDIS, POSYDIS, POSZDIS, RADIUS, TETNEW, PHINEW)
C
      THETA = 90 - TETNEW*180.0/PI
      PHI   = PHINEW*180.0/PI
      RANGE(1) = PHI
      RANGE(2) = THETA
C
C  WRITE FLIGHT DATA INTO FILE
C
C     WRITE(3,915) FLIGHTTIME,TAB,RADIUS-RADINIT,TAB,THETA,TAB,SPEED
C
      IF (RADIUS.GE.0.7E+07) GOTO 9998
      IF (RADIUS.LE.RADINIT) GOTO 9999
```

```
C
      END DO
C
C  CALCULATE FLIGHT STATISTICS: MEAN VALUE OF ANGLE, AND SIGMA FOR
C  DELTA RANGE, AND MAX ACCELERATION AND DYN. PRESSURE
C
9998  WRITE(*,*) 'SUCCESSFUL LANDING ON PLUTO'
C
9999  IF (ZAEHL.EQ.1) THEN
        WRITE (*,*) ' MEAN-LONGITUDE =', RANGE(1)
        WRITE (*,*) ' MEAN-LATITUDE  =', RANGE(2)
      END IF
C
      NOPO = ZAEHL - 1
C
      IF (NOPO.EQ.1) THEN
        MEAN(1)  = RANGE(1)
        STDEV(1) = 0
        VAR(1)   = 0
        MEAN(2)  = RANGE(2)
        STDEV(2) = 0
        VAR(2)   = 0
        MEAN(3)  = ACCMAX
        STDEV(3) = 0
        VAR(3)   = 0
        MEAN(4)  = DYNMAX
        STDEV(4) = 0
        VAR(4)   = 0
      ELSE IF (NOPO.GT.1) THEN
        CALL STATISTICS(RANGE(1), NOPO, MEAN(1), STDEV(1), VAR(1))
        CALL STATISTICS(RANGE(2), NOPO, MEAN(2), STDEV(2), VAR(2))
        CALL STATISTICS(ACCMAX,   NOPO, MEAN(3), STDEV(3), VAR(3))
        CALL STATISTICS(DYNMAX,   NOPO, MEAN(4), STDEV(4), VAR(4))
      END IF
C
      DELRANGE(1) = RANGE(1) - MEAN(1)
      DELRANGE(2) = RANGE(2) - MEAN(2)
C
C     WRITE(*,*)  'Max Accel. =',ACCMAX, '  Max. Dyn.Pressure =',DYNMAX
C     WRITE(*,*)  'Flighttime =', FLIGHTTIME
      WRITE(2,906)  ZAEHL, TAB, RANGE(1), TAB, RANGE(2), TAB, ACCMAX,
     &          TAB, DYNMAX
      IF (ZAEHL.EQ.RUNS) THEN
C       WRITE(*,910)
C       WRITE(*,911) ZAEHL,TAB,MEAN(1),TAB,VAR(1),TAB,STDEV(1)
C       WRITE(*,911) ZAEHL,TAB,MEAN(2),TAB,VAR(2),TAB,STDEV(2)
      END IF
      WRITE(1,913) ZAEHL,TAB,MEAN(1),TAB,STDEV(1),TAB,MEAN(2),TAB,
     &  STDEV(2),TAB,MEAN(3),TAB,STDEV(3),TAB,MEAN(4),TAB,STDEV(4)
C     WRITE(*,*) '************************************************',
C    &          '**************************'
C
C  END OF ITERATION LOOP
C
      END DO
C
C  END OF SIMULATION FOR ONE RUN
```

```
C
C
 906 FORMAT(1X, I4, A1, G13.7, A1, G16.10, A1, G13.7, A1, G13.7)
C
 910 FORMAT(1X, ' CASE    MEAN    VARIANCE    STANDARD DEVIATION')
 911 FORMAT(1X, I4, A1, G13.7, A1, G10.4, A1, G10.4)
C
 913 FORMAT(1X,I4,A1,G12.6,A1,G12.6,A1,G12.6,A1,G12.6,
   &     A1,G12.6,A1,G12.6,A1,G12.6,A1,G12.6)
C
 915 FORMAT(1X,I4,A1,G12.6,A1,G12.6,A1,G12.6)
C
    CLOSE (UNIT = 1)
    CLOSE (UNIT = 2)
    CLOSE (UNIT = 3)
C
    END

C
C**********************************************************************
    SUBROUTINE DIRECTION
   &        (SPEEDX, SPEEDY, SPEEDZ, LRX, LRY, LRZ, LNX, LNY, LNZ)
C
C  THE PURPOSE FO THIS SUBROUTINE IS TO CALCULATE THE DIRECTION
C  VECTORS FOR THE DRAG-COEFFICIENT CD AND THE LIFT-COEFFICIENT CL.
C  THE LIFT-COEFF. IS ASSUMED TO BE NORMAL TO CD.
C
    REAL SPEEDX, SPEEDY, SPEEDZ, LRX, LRY, LRZ, LNX, LNY, LNZ
    REAL X, Y, Z, MAGNIT
C
    X = SPEEDX
    Y = SPEEDY
    Z = SPEEDZ
    MAGNIT = SQRT(X**2 + Y**2 + Z**2)
    IF (MAGNIT.EQ.0) MAGNIT = 1.0
C
    LRX = X/MAGNIT
    LRY = Y/MAGNIT
    LRZ = Z/MAGNIT
C
    LNX = 0.0
    LNY = 0.0
    LNZ = 0.0
C
    RETURN
    END
C**********************************************************************


C**********************************************************************
    SUBROUTINE FORCES
   & (RADIUS, DENS, SPEED, MUE, MASS, LRX, LRY, LRZ,
   &  LNX, LNY, LNZ, TETNEW, PHINEW, CD, CL, AREF, ACCX, ACCY, ACCZ)
C
    REAL RADIUS, DENS, SPEED, ACCX, ACCY, ACCZ, LRX, LRY, LRZ,
   &    LNX, LNY, LNZ, MUE, CD, CL, AREF(3), MASS, PHINEW, TETNEW
C
```

```
      COMMON /PERFORM/ ACCMAX, DYNMAX
C
C  THE PURPOSE OF THIS SUBROUTINE IS TO CALCULATE THE MAGNITUDES OF
C  THE GRAVITATIONAL, AND THE AERODYNAMIC FORCES.  THE VALUES FOR
C  THE ACCELERATIONS ARE RETURNED.
C
      REAL  FORGRAV, AERO, AERODR, AEROLIF, ACC, DYN, DYNMAX, ACCMAX,
     &      GRX, GRY, GRZ, ADRX, ADRY, ADRZ, ALIFX, ALIFY, ALIFZ,
     &      AX, AY, AZ
C
C**********************************************************************
C
C  MAGNITUDES OF FORCES
C
C  1.  GRAVITATIONAL FORCE
C
      FORGRAV = MASS*MUE/RADIUS**2
C
C  2.  AERODYNAMIC FORCES
C
      DYN = 0.5*DENS*SPEED**2
      AERO = DYN*AREF(1)
      AERODR = AERO*CD
      AEROLIF= AERO*CL
C
C
C
C  DIRECTION OF FORCES
C
C  1.  GRAVITATIONAL FORCE
C
      GRX = - FORGRAV*COS(PHINEW)*SIN(TETNEW)
      GRY = - FORGRAV*SIN(PHINEW)*SIN(TETNEW)
      GRZ = - FORGRAV*COS(TETNEW)
C
C  2.  AERODYNAMIC FORCES
C
C
      ADRX = - AERODR*LRX
      ADRY = - AERODR*LRY
      ADRZ = - AERODR*LRZ
      ALIFX = AEROLIF*LNX
      ALIFY = AEROLIF*LNY
      ALIFZ = AEROLIF*LNZ
C
      AX = ADRX + ALIFX
      AY = ADRY + ALIFY
      AZ = ADRZ + ALIFZ
C
C  TOTAL ACCELERATIONS
C
      ACCX = (GRX + AX)/MASS
      ACCY = (GRY + AY)/MASS
      ACCZ = (GRZ + AZ)/MASS
      ACC  = SQRT(ACCX**2 + ACCY**2 + ACCZ**2)
C
      IF (ACC.GT.ACCMAX) ACCMAX = ACC
```

```
      IF (DYN.GT.DYNMAX) DYNMAX = DYN
C
      RETURN
      END
C*********************************************************************
      SUBROUTINE DISPERSUNI (MEAN, DELTA, DISPERS)
C
C  WHEN THIS SUBROUTINE IS CALLED, A RANDOM NUMBER AROUND THE MEAN
C  "MEAN"+- THE TOLERANCE "DELTA" IS GENERATED.  THE RANDOM NUMBER IS
C  GENERATED IN THE FUNCTION "SRAN", FROM WHICH A NUMBER BETWEEN 0 - 1
C  IS OBTAINED.
C
      REAL MEAN, DELTA, DISPERS
      COMMON /DISTRIB/IDUM
      INTEGER*4 IDUM
      REAL WERT
C
      WERT = SRAN(IDUM)
C
      DISPERS = MEAN + DELTA*(2.0*WERT - 1.0)
      RETURN
      END

C*********************************************************************
      SUBROUTINE DISPERSNOR (MEAN, SIGMA, DISPERS)
C
C  IN THIS SUBROUTINE A NORMALLY DISTRIBUTED RANDOM NUMBER IS GENERATED
C  AROUND THE MEAN "PARAM" WITH STANDARD DEVIATION OF SIGMA.

      REAL MEAN, SIGMA, DISPERS
      COMMON /DISTRIB/IDUM
      INTEGER*4 IDUM
      REAL WERT, V1, V2, R, FAC, GSET, GASDEV
      DATA ISET/0/
C
      IF (ISET.EQ.0) THEN
1       V1 = 2.0*SRAN(IDUM) - 1.0
        V2 = 2.0*SRAN(IDUM) - 1.0
        R = V1**2 + V2**2
        IF (R.GE.1.0.OR.R.EQ.0.) GOTO 1
        FAC = SQRT (-2.0*LOG(R)/R)
        GSET = V1*FAC
        GASDEV = V2*FAC
        ISET = 1
      ELSE
        GASDEV = GSET
        ISET = 0
      ENDIF
      WERT = GASDEV
C
      DISPERS = MEAN + SIGMA*WERT
      RETURN
      END

C*********************************************************************

      SUBROUTINE STATISTICS (VALUE, NOPO, MEAN, STDEV, VAR)
```

```
      REAL VALUE, MEAN, STDEV, VAR
      INTEGER NOPO
C
C  THIS STATISTICS ROUTINE USES THE OLD VALUES OF MEAN, STANDARD
C  DEVIATION, AND VARIANCE AND UPDATES THE VALUES WITH THE NEW POINT
C
      MEAN = (MEAN*(NOPO - 1) + VALUE)/NOPO
      VAR  = (VAR*(NOPO - 2) + (VALUE - MEAN)**2)/(NOPO - 1)
      STDEV = SQRT(VAR)
      RETURN
      END

C+
C*********************************************************************
C
C Real Function SRAN
C
C Purpose:  Return a pseudo-random number between 0.0 and 1.0
C
C Reference:
C         Numerical Recipes in FORTRAN
C         by Press et. al
C
C---------------------------------------------------------------------
C Arguments      Name    Type  Description
C ---------      ----    ----  -----------
C Input:         none
C
C Output:        none
C
C Input/Output:  IDUMS    int   dummy seed number
C
C---------------------------------------------------------------------
C Common Blocks:  RANCOM
C
C Include Files:  none
C
C Calls to:       none
C---------------------------------------------------------------------
C Development History
C
C Author:  Ravi P. Reddy
C Date:    February 5, 1991
C
C Modifications:
C
C*********************************************************************
C-
C
      REAL FUNCTION SRAN (IDUMS)
C
      REAL RM
      INTEGER M, IA, IC
C
C     PARAMETER(M = 6075,  IA = 106, IC = 1283, RM = 1./M)
C     PARAMETER(M = 7875,  IA = 211, IC = 1663, RM = 1./M)
C     PARAMETER(M = 7875,  IA = 421, IC = 1663, RM = 1./M)
```

```
      PARAMETER(M = 11979, IA = 430, IC = 2531, RM = 1./M)
C
      INTEGER IDUMS, J, IR, ISEED
      COMMON/RANCOM/ ISEED, IR(97)
C
      INTEGER IFF
      DATA IFF/0/
C
C-----------------------------------------------------
C
C     Initialization
C
      IF((IFF.EQ.0).OR.(IDUMS.LT.0))THEN
       IFF=1
       IF(IDUMS.LT.0)IDUMS = -IDUMS
       IF(IDUMS .GT. IC) IDUMS = 1111
       IDUMS=MOD(IC-IDUMS,M)
C
C     Warming up
C
       IDUMS=MOD(IA*IDUMS+IC,M)
       IDUMS=MOD(IA*IDUMS+IC,M)
       IDUMS=MOD(IA*IDUMS+IC,M)
C
C     Load the shuffling deck of numbers
C
       DO 100 J=1,97
        IDUMS=MOD(IA*IDUMS+IC,M)
        IR(J)=IDUMS
100     CONTINUE
C
       IDUMS=MOD(IA*IDUMS+IC,M)
       ISEED=IDUMS
      ENDIF
C-----------------------------------------------------
C
C  Normal execution
C
      J=1+(97*ISEED)/M
      IF(J.GT.97.OR.J.LT.1)WRITE(2,*)' ERROR IN SRAN'
      IDUMS=IR(J)
      SRAN=IDUMS*RM
      ISEED=MOD(IA*IDUMS+IC,M)
      IR(J)=MOD(IA*ISEED+IC,M)
C
C
      RETURN
C
C**********************************************************************
C     End of Real Function SRAN
C**********************************************************************
C
      END
C
      SUBROUTINE ORTARRAY (NDESV, LOWER, MIDDLE, UPPER, OA)
C**********************************************************************
C
```

```
C       NAME: CREATE_ORT_ARRAY
C
C       PURPOSE: CREATION OF 3-LEVEL ORTHOGONAL ARRAYS FROM
C                LATIN-SQUARES WITH 9, 27 OR 81 EXPERIMENTS
C
C       PROGRAMMER: UWE LAUTENSCHLAGER
C
C       DATE: 26 MARCH, 1992
C
C**********************************************************************
C
C       ARGUMENTS:
C
C**********************************************************************
C
C   VARIABLES:
C
    INTEGER I, J, COUNT, OLDROW, OLDCOL, LATSQ1(3,3), LATSQ2(3,3),
   &      NDESV
    REAL   ORTAR9(9,4), ORTAR27(27,13), ORTAR81(81,40), ORTAR(81,40),
   &       OLD(81,40), LOWER(40), MIDDLE(40), UPPER(40), OA(81,40)
C
C**********************************************************************
C
C       CREATE ARRAY FOR LATIN-SQUARE 1 (1,2,3)
C
    LATSQ1(1,1) = 1
    LATSQ1(1,2) = 2
    LATSQ1(1,3) = 3
    LATSQ1(2,1) = 2
    LATSQ1(2,2) = 3
    LATSQ1(2,3) = 1
    LATSQ1(3,1) = 3
    LATSQ1(3,2) = 1
    LATSQ1(3,3) = 2
C
C**********************************************************************
C
C       CREATE ARRAY FOR LATIN-SQUARE 2 (3,2,1)
C
    LATSQ2(1,1) = 1
    LATSQ2(1,2) = 3
    LATSQ2(1,3) = 2
    LATSQ2(2,1) = 2
    LATSQ2(2,2) = 1
    LATSQ2(2,3) = 3
    LATSQ2(3,1) = 3
    LATSQ2(3,2) = 2
    LATSQ2(3,3) = 1
C
C**********************************************************************
C       WE NEED THE FOLLOWING INFORMATION ABOUT THE VARIABLES:
C       NDESV, LOWER AND UPPER BOUND, TOLERANCES
C
C**********************************************************************
C       CREATE ORTHOGONAL ARRAYS FROM LATIN SQUARES
C       THE RULES ARE AS FOLLOWS:
```

```
C
C       0. TAKE THE COLUMN FROM THE SMALLER ARRAY TO CREATE 2 NEW
C         COLUMNS AND 2X NEW ROWS
C       1. BLOCK1(1. 1/3 ROWS): TAKE OLD VALUE 2X FOR NEW COLUMNS
C       2. BLOCK2(2. 1/3 ROWS): TAKE OLD VALUE, USE LATSQ1 FOR NEW COLUMNS
C       3. BLOCK2(3. 1/3 ROWS): TAKE OLD VALUE, USE LATSQ2 FOR NEW COLUMNS
C       4. COLUMN1: DEVIDE EXPERIMENTS INTO GROUPS OF 1,2,3
C
   OLD(1,1) = 1
   OLD(2,1) = 2
   OLD(3,1) = 3
   OLDROW = 3
   OLDCOL = 1
C
C**************************************************************************
C
C
   DO 100 COUNT = 1, 3
   DO 20 I = 1, OLDROW
     DO 20 J = 1, OLDCOL
C
C       BLOCK1
C
       IF (OLD(I,J).EQ.1) THEN
        ORTAR(I,J*3-1) = 1
        ORTAR(I,J*3)   = 1
        ORTAR(I,J*3+1)  = 1
       ELSE IF(OLD(I,J).EQ.2) THEN
        ORTAR(I,J*3-1) = 2
        ORTAR(I,J*3)   = 2
        ORTAR(I,J*3+1)  = 2
       ELSE IF(OLD(I,J).EQ.3) THEN
        ORTAR(I,J*3-1) = 3
        ORTAR(I,J*3)   = 3
        ORTAR(I,J*3+1)  = 3
       END IF
C
        ORTAR(I,1) = 1
C
C       BLOCK2
C
       IF (OLD(I,J).EQ.1) THEN
        ORTAR(I+OLDROW,J*3-1) = LATSQ1(1,1)
        ORTAR(I+OLDROW,J*3)   = LATSQ1(1,2)
        ORTAR(I+OLDROW,J*3+1) = LATSQ1(1,3)
       ELSE IF(OLD(I,J).EQ.2) THEN
        ORTAR(I+OLDROW,J*3-1) = LATSQ1(2,1)
        ORTAR(I+OLDROW,J*3)   = LATSQ1(2,2)
        ORTAR(I+OLDROW,J*3+1) = LATSQ1(2,3)
       ELSE IF(OLD(I,J).EQ.3) THEN
        ORTAR(I+OLDROW,J*3-1) = LATSQ1(3,1)
        ORTAR(I+OLDROW,J*3)   = LATSQ1(3,2)
        ORTAR(I+OLDROW,J*3+1) = LATSQ1(3,3)
       END IF
C
        ORTAR(I+OLDROW,1) = 2
C
```

```
C        BLOCK3
C
         IF (OLD(I,J).EQ.1) THEN
           ORTAR(I+2*OLDROW,J*3-1) = LATSQ2(1,1)
           ORTAR(I+2*OLDROW,J*3)   = LATSQ2(1,2)
           ORTAR(I+2*OLDROW,J*3+1) = LATSQ2(1,3)
         ELSE IF(OLD(I,J).EQ.2) THEN
           ORTAR(I+2*OLDROW,J*3-1) = LATSQ2(2,1)
           ORTAR(I+2*OLDROW,J*3)   = LATSQ2(2,2)
           ORTAR(I+2*OLDROW,J*3+1) = LATSQ2(2,3)
         ELSE IF(OLD(I,J).EQ.3) THEN
           ORTAR(I+2*OLDROW,J*3-1) = LATSQ2(3,1)
           ORTAR(I+2*OLDROW,J*3)   = LATSQ2(3,2)
           ORTAR(I+2*OLDROW,J*3+1) = LATSQ2(3,3)
         END IF
C
         ORTAR(I+2*OLDROW,1) = 3
C
 20   CONTINUE
C
     OLDROW = 3*OLDROW
     OLDCOL = 3*OLDCOL + 1
C
     DO 30 I = 1, OLDROW
       DO 30 J = 1, OLDCOL
         IF (COUNT.EQ.1) THEN
           ORTAR9(I,J)  = ORTAR(I,J)
           OLD(I,J)     = ORTAR(I,J)
         ELSE IF (COUNT.EQ.2) THEN
           ORTAR27(I,J) = ORTAR(I,J)
           OLD(I,J)     = ORTAR(I,J)
         ELSE IF (COUNT.EQ.3) THEN
           ORTAR81(I,J) = ORTAR(I,J)
         END IF
C
 30   CONTINUE
C
 100  CONTINUE
C
C
C*********************************************************************
C
C        ASSIGN VARIABLE VALUES TO ORTHOGONAL ARRAYS
C
     OLDCOL = NDESV
     IF (NDESV.LE.4) THEN
       CASE = 1
       OLDROW = 9
     ELSE IF ((NDESV.GT.4).AND.(NDESV.LE.13)) THEN
       CASE = 2
       OLDROW = 27
     ELSE
       CASE = 3
       OLDROW = 81
     END IF
C
     DO 110 I = 1, OLDROW
```

```
      DO 110 J = 1, OLDCOL

        IF (CASE.EQ.1) THEN
          ORTAR(I,J) = ORTAR9(I,4 - OLDCOL + J)
        ELSE IF (CASE.EQ.2) THEN
          ORTAR(I,J) = ORTAR27(I,13 - OLDCOL + J)
        ELSE IF (CASE.EQ.3) THEN
          ORTAR(I,J) = ORTAR81(I,40 - OLDCOL + J)
        ENDIF
C
         IF (ORTAR(I,J).EQ.1) THEN
          ORTAR(I,J) = LOWER(J)
         ELSE IF (ORTAR(I,J).EQ.2) THEN
          ORTAR(I,J) = MIDDLE(J)
         ELSE
          ORTAR(I,J) = UPPER(J)
         END IF
C
 110  CONTINUE
C
      OPEN (UNIT = 7, FILE = 'experi.check', STATUS = 'UNKNOWN')
C
      WRITE(7,901) NDESV, OLDROW
      WRITE(7,903) (I, I = 1, NDESV)
      WRITE(7,904) ('-', I = 1, 12*NDESV + 6)
C
      DO 120 I = 1, OLDROW
        WRITE(7,902) I, (ORTAR(I,OLDCOL+J-NDESV), J = 1, NDESV)
        DO 130 J = 1, NDESV
          OA(I,J) = ORTAR(I,OLDCOL+J-NDESV)
 130    CONTINUE
 120  CONTINUE
C
 901  FORMAT(1X, 'NUMBER OF VARIABLES: 'I2,3X,'NUMBER OF EXPERIMENTS: ',
     &      I2,/)
 902  FORMAT(2X, I3, 5X, 40(1X,G11.5))
 903  FORMAT(1X,'EXP.',40I12,/)
 904  FORMAT(1X, 172A1)
C
      CLOSE (UNIT = 7)
C
      RETURN
      END

      SUBROUTINE LEVELSNOR (MEAN, SIGMA, LOWER, MIDDLE, UPPER)
C**********************************************************************
C   PROVIDES VALUES FOR LOWER, MIDDLE, AND UPPER BOUND; NORMAL DISTR.
C**********************************************************************
C
C        ARGUMENTS:
C
C**********************************************************************
C
C        LOCAL VARIABLES:
C
      REAL    MEAN, SIGMA, LOWER, MIDDLE, UPPER
C
```

```
C*********************************************************************
C
C  SPECIFY LOWER, MIDDLE AND UPPER VALUES
C
      LOWER  = MEAN - SQRT(1.5)*SIGMA
      MIDDLE = MEAN
      UPPER  = MEAN + SQRT(1.5)*SIGMA
C
      RETURN
      END

      SUBROUTINE LEVELSUNI (MEAN, DELTA, LOWER, MIDDLE, UPPER)
C*********************************************************************
C    PROVIDES VALUES FOR LOWER, MIDDLE, AND UPPER BOUND; UNIFORM DISTR.
C*********************************************************************
C
C         ARGUMENTS:
C
C*********************************************************************
C
C         LOCAL VARIABLES:
C
      REAL    MEAN, DELTA, LOWER, MIDDLE, UPPER
C
C*********************************************************************
C
C  SPECIFY LOWER, MIDDLE AND UPPER VALUES
C
      LOWER  = MEAN - DELTA
      MIDDLE = MEAN
      UPPER  = MEAN + DELTA
C
      RETURN
      END

      SUBROUTINE POLAR (POSX, POSY, POSZ, RAD, TET, PHI)
C*********************************************************************
C    CALCULATES POSITION STATE PARAMETERS IN POLAR COORDINATES
C*********************************************************************
C
C         ARGUMENTS:
C
C*********************************************************************
C
C         LOCAL VARIABLES:
C
      REAL    POSX, POSY, POSZ, RAD, TET, PHI, ARGU, PI
C
C*********************************************************************
C
C  SPECIFY LOWER, MIDDLE AND UPPER VALUES
C
      PI = 3.1415927
C
C  CALCULATION OF RADIUS
C
      RAD = SQRT(POSX**2 + POSY**2 + POSZ**2)
```

```
C
C  CALCULATION OF PHI (PHI = ARCTAN(Y/X))
C
    IF (POSX.NE.0) THEN
      ARGU = POSY/POSX
      PHI = ATAN(ARGU)
    ELSE IF (POSY.GT.0) THEN
         PHI = PI/2.0
       ELSE
         PHI = PI*1.5
    END IF
C
    IF (POSX.LT.0) THEN
      IF (POSY.GE.0) THEN
        PHI = PHI + PI
      ELSE IF (POSY.LT.0) THEN
        PHI = PHI - PI
      END IF
    END IF
C
C  CALCULATION OF THETA (THETA = ARCCOS(Z/R)
C
    TET = ACOS(POSZ/RAD)
C
    RETURN
    END

    SUBROUTINE POSIT (LONGIT, LATIT, RADIUS, POSX, POSY, POSZ)
C*********************************************************************
C   CALCULATES POSITION STATE PARAMETERS IN POLAR COORDINATES
C*********************************************************************
C
C       ARGUMENTS:
C
C*********************************************************************
C
C       LOCAL VARIABLES:
C
    REAL    RADIUS, LONGIT, LATIT, POSX, POSY, POSZ, PI, PHI, TET
C
C*********************************************************************
C
    PI = 3.1415927
C
    PHI = LONGIT*PI/180.0
    TET = PI/2.0 - LATIT*PI/180.0
C
    POSX = RADIUS*COS(PHI)*SIN(TET)
    POSY = RADIUS*SIN(PHI)*SIN(TET)
    POSZ = RADIUS*COS(TET)
C
    RETURN
    END

    SUBROUTINE VELOC (LONGIT, LATIT, FLIPA, AZIMU, SPEED,
   &              VELX, VELY, VELZ)
C*********************************************************************
```

```
C    CALCULATES VELOCITY STATE PARAMETERS IN X, Y, Z COORDINATES
C*********************************************************************
C
C         ARGUMENTS:
C
C*********************************************************************
C
C         LOCAL VARIABLES:
C
      REAL    LONGIT, LATIT, FLIPA, AZIMU, SPEED, PI, PHI, TET,
     &        VELU, VELV, VELW, VELX, VELY, VELZ
C
C*********************************************************************
C
      PI = 3.1415927
C
      PHI = LONGIT*PI/180.0
      TET = PI/2.0 - LATIT*PI/180.0
C
      VELU = SPEED*COS(FLIPA)*COS(AZIMU)
      VELV = SPEED*COS(FLIPA)*SIN(AZIMU)
      VELW = SPEED*SIN(FLIPA)
C
      VELX = COS(PHI)*(-VELU*COS(TET) + VELW*SIN(TET)) + VELV*SIN(PHI)
      VELY = SIN(PHI)*(-VELU*COS(TET) + VELW*SIN(TET)) - VELV*COS(PHI)
      VELZ = VELU*SIN(TET) + VELW*COS(TET)
C
      RETURN
      END
```

Flightsimulation Data-file: flightsim.dat

```
DATA FILE FOR FLIGHTSIMULATION
Created by : UWE LAUTENSCHLAGER
Date :  10 JUNE 1992
Comment: GOOD LUCK
-----------------------------------------------------------------
1                       : '0' for random Nunber or '1' for Orthogonal Arrays
13                      : number of design variables, or number of columns
37571   200             : # of skipped random numbers, # of dispersed runs
9946.5                  :  initial speed(m/s) (9946.5)
1560.0          :  mass (1560)
-5.8814 180.0           : inertial flightpath(deg), inertial azimuth (deg) -5.8814
-106.65 44.2    : longitude (deg), geodetic latitude (deg) 44.314
-----------------------------------------------------------------
Please specify the all the tolerances for the parameters.  Enter
either the 3-sigma value in % of the mean for normally dispersed
parameters or the tolerance in % for uniformly dispersed parameters.
In the second column, enter the column to be used in the OA
-----------------------------------------------------------------
0.1   9                 : absolute tolerance in (deg) for longitude
0.05   7        : absolute tolerance in (deg) for latitude
250.0   6       : absolute tolerance in (m) for radius
0.05   2        : mass tolerance (0.05)
0.1   1                 : density 3-sigma (0.3)
0.15   3                : drag-coefficent 3-sigma (0.05)
0.0075   5      : flightpath 3-sigma
0.01   8                : azimuth 3-sigma
0.02   4                : initial velocity 3-sigma

Please note that dispersions are 1-2 % for the initial state parameters.
```

Appendix B

Flightsimulation Program Source Code

This appendix contains a short description of the program ANOVA.

ANOVA

General Description:

The program ANOVA is a *prototypical* tool to perform analysis of variance on experimental results obtained by using orthogonal arrays.

File Structure:

This program requires two input files:

- One file which contains the standard orthogonal array corresponding to the OA used in the experimentation. There are currently three OA files implemented: oaL8.dat, oaL9.dat, and oaL27.dat, which correspond to an L_8, L_9, and L_{27} matrix experiment, respectively.

- One file which contains the result file from the experiment. The current choice is the output file from the XPLORE facility in DSIDES, the standard output file from the simulation program FLUG, and a general one column result file. The format of this file will be discussed later.

The program prints the result from the anova calculation both to the screen and to a file called "anova.out".

Assumptions:

The results on which the analysis of variance are performed are assumed to be obtained by using experiments prescribed by a suitable orthogonal array. The following assumptions are made:

- The number of variables in the experiment should be less than the number of columns in the OA, minus one. This is because we want to dedicate at least one column in the OA to estimate the error term.

- The variables should be assigned to the first columns of the OA. (The theory would work just as well if the variables were assigned to the last columns—in the program we have just made a choice.)

The code:

The ANOVA program is programmed in ANSI C:

The code consists of three files:

anova.c contains the main routine and subroutine specific to this program (reading the files, calculating the anova table for the OA, etc.)

anovaLib.c Contains various subroutines which are called from anova.c (initialization of vectors and matrices, beta- and gamma-functions, etc.)

anova.h is a headerfile which is included in both of the other files and consists mainly of function declarations.

At SDL the program was compiled by:

sdl> **acc anova.c anovaLib.c -o anova -lm -g**

The experiment result file:

As mentioned earlier, the program is made to read three different types of result files. But it can relatively easily be changed to read and result file.

E.g., the standard output file from FLUG consists of five columns. The first column is the experiment number, the second and third columns are the longitudinal and latitudinal landing position, respectively, and the fourth and fifth columns are other output data that we do not use in the anova calculation.

```
1    -106.6500    33.60182571    130.6447    92639.22
2    -106.9348    33.52516174    130.2403    89573.64
3    -106.6500    33.58375921    130.7231    89907.05
4    -106.3649 .....
.....
```

In the file reading routine, which in this case is found in the subroutines redResult() in "anova.c", the results are read into the vectors res.Long and res.Lat, respectively. The other information in the file, which is not used for the anova calculations, is just read into dummy variables.

```
    if ((fpl=fopen(fileName, "r"))==NULL)  {
        printf("\nError reading file %s".fileName);
        exit (1);
    }
.....
    for (i=0;i<*noExp;i++){
        fscanf(fp1,"%d",&dummyI);
        fscanf(fp1,"%f",resLong+i);
        fscanf(fpl,"%f",resLat+i);
.....
        fscanf(fpl,"%f",&dummyF);
        fscanf(fpl,"%f",dummyF);
    }
```

```
        fclose(fpl);
```

For another format on the result file, this reading sequence can be changed by just adding or deleting *fscanf* sentences.

The orthogonal array file:

The orthogonal array file will provide the program with the level settings for the orthogonal experiment used to produce the results. The file looks like (oaL27.dat):

```
3    13   27

1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0
1.0  1.0  1.0  2.0  2.0  2.0  2.0  2.0  2.0  2.0  2.0  2.0  2.0
1.0  1.0  1.0  3.0  3.0  3.0  3.0  3.0  3.0  3.0  3.0  3.0  3.0
1.0  2.0  2.0  2.0  1.0  1.0  1.0  2.0  2.0  ...
```

The first three numbers are the "particulars" for this OA. In this case, they represent three levels, a maximum of 13 variables, and 27 experiments, respectively.

Appendix C

ANOVA Program Source Code

This appendix contains the source code for the ANOVA program. The code is written in ANSI C. The code consists of three files: *anova.c* includes the main routine and the ANOVA subroutines, *anovaLib.c* includes several library routines for the program, and *anova.h* is a headerfile where constants, functions, include-files, etc. are declared.

```
/*-------------------------------------------------------------
                        anova.h

Name:   Stein Ove Erikstad
Date:   June 1992
Source:

The source code of the program is in tree files:

        anova.h         header file for the program
        anova.c         the main function, and some program specific
                        functions
        anovaLib.c      some library function used by the program


This program calculates a ANOVA table from an orthogonal array.

The program requires two input files:

        oa.out -        which contains the orthogonal array to be used
        ortpoint.out - which contains the results from the experiments
                referred to in oa.out

-------------------------------------------------------------------------*/

/*--------------------- Functions in anova.c ----------------------*/

float **readOA();
void readResult();
void readExplore();
void readGenResFile();
void anova();
void printAnovaTable();
int pool();
void confIntMean();

/*--------------------- Functions in anovaLib.c ----------------------*/

float *vector();
int *vectorI();
float **matrix();
void freeVector();
void freeIVector();
void freeMatrix();
void errMess();
void descrData();
void writeMatrix();
float gammaln();
float betacf();
float fvalue();
float betai();
```

```
/*-------------------------------------------------------------
                        anova.c

Name:   Stein Ove Erikstad
Date:   June 1992
Source:

The source code of the program is in tree files:

        anova.h         header file for the program
        anova.c         the main function, and some program specific
                        functions
        anovaLib.c      some library function used by the program


This program calculates a ANOVA table from an orthogonal array.

The program requires two input files:

        oa.out -        which contains the orthogonal array to be used
        ortpoint.out - which contains the results from the experiments
                referred to in oa.out

-------------------------------------------------------------------------*/

/*-------------------- Functions in anova.c ----------------------*/

float **readOA();
void readResult();
void readExplore();
void readGenResFile();
void anova();
void printAnovaTable();
int pool();
void confIntMean();

/*-------------------- Functions in anovaLib.c ----------------------*/

float *vector();
int *vectorI();
float **matrix();
void freeVector();
void freeIVector();
void freeMatrix();
void errMess();
void descrData();
void writeMatrix();
float gammaln();
float betacf();
float fvalue();
float betai();
```

```
/*-------------------------------------------------------
        anovaLib.c
-------------------------------------------------------*/
#include <stdio.h>
#include <math.h>
#include "anova.h"

#define ITMAX 200
#define EPS 3.0e-7

/*-------------------------------------------------------
            MATRIX
        matrix allocates storage to a matrix with float
        elements, and with size nrow, ncol. The function
        returns a pointer to an array of pointers to the rows

        Name:   Stein Ove Erikstad
        Date:   May 1992
        Source: Based on function in Numerical Recipies in C
-------------------------------------------------------*/

float **matrix(nrow,ncol)
        int nrow, ncol;
{
        int i;
        float **m;

        m=(float **) malloc((unsigned) (nrow+1)*sizeof(float*));
        if (!m) errMess("Allocation error in matrix()");

        for (i=0;i<=nrow;i++) {
                m[i]=(float *) malloc((unsigned) (ncol+1)*sizeof(float));
                if (!m[i]) errMess("Allocation failure in matrix()");
        }
        return m;
}

/*-------------------------------------------------------
            ERRMESS
        errMess writes an error message given in errText
        to stderr.

        Name:   Stein Ove Erikstad
        Date:   May 1992
        Source: Based on function in Numerical Recipies in C
-------------------------------------------------------*/

void errMess(errText)
        char errText[];
{
        fprintf(stderr,"A run-time error is discovered....\n");
        fprintf(stderr,"%s\n",errText);
        fprintf(stderr,"...aborting system....\n");
        exit(1);
}
/*-------------------------------------------------------
            VECTOR
```

```
        vector allocates storage to a vector with float
        elements, and with size nelem. The function
        returns a pointer to the first element

        Name:   Stein Ove Erikstad
        Date:   May 1992
        Source: Based on function in Numerical Recipies in C
-------------------------------------------------------*/
float *vector(nelem)
        int nelem;
{
        float *v;
        v=(float *) malloc((unsigned) (nelem+1)*sizeof(float));
        if (!v) errMess("Allocation failure in vector()");
        return v;
}
int *vectorI(nelem)
        int nelem;
{
        int *v;
        v=(int *) malloc((unsigned) (nelem+1)*sizeof(int));
        if (!v) errMess("Allocation failure in vectorI()");
        return v;
}
/*-------------------------------------------------------
            FREE
        vector frees storage to a vector with float
        elements, and with size nelem.

        Name:   Stein Ove Erikstad
        Date:   July 1992
        Source: Based on function in Numerical Recipies in C
-------------------------------------------------------*/
void freeVector(v,nelem)
        float *v;
        int nelem;
{
        free((char*)v);
}
void freeIVector(v,nelem)
        int *v,nelem;
{
        free((char*)v);
}
void freeMatrix(m,nrow,ncol)
        float **m;
        int nrow,ncol;
{
        int i;

        for (i=nrow;i>=0;i--) free((char*)m[i]);
        free((char*)m);
}

/*-------------------------------------------------------
                WRITEMATRIX
        writeMatrix prints a matrix row by row from the standard
```

```
        output. Each row element is separated by a whitespace,
        and each row is terminated by CR.

        Input:  m       adress of upper left corner of matrix
                nrow    no. of rows
                ncol    no. of columns

        Name:   Stein Ove Erikstad
        Date:   May 1992
        Source: None
-------------------------------------------------------------*/
void writeMatrix(m,nrow,ncol)
        float **m;
        int nrow,ncol;
{
        int i,j;

        for (i=0;i<nrow;i++) {
                printf("\n");
                for (j=0;j<ncol;j++) {
                        printf("\t%5.2f",*(*(m+i)+j));
                }
        }
        return;
}

/*-------------------------------------------------------------
                DESCRDATA
        descrData receives an array of data, and returns
        a statistical description of these data: mean,
        variance, etc.

        Name:   Stein Ove Erikstad
        Date:   May 1992
        Source: Based on moment() in Numerical Recipies in C
-------------------------------------------------------------*/
void descrData(data,n,sum,sqsum,mean,adev,var,writeFlag,fp)
        int n,writeFlag;
        float data[],*sum,*sqsum,*mean,*adev,*var;
        FILE *fp;
{
        int j;
        float s,p,sn;

        *adev=(*var)=(*sum)=(*sqsum)=0.0;

        if (n<=1) errMess("No. of data pts. must be at least two");
        for (j=0;j<n;j++) {
                *sum += data[j];
                *sqsum += data[j]*data[j];
        }
        *mean=*sum/n;
        for (j=0;j<n;j++) {
                *adev += fabs(s=data[j]-(*mean));
                *var += (p=s*s);
        }
        *var /= (n-1);
```

```
        sn=10.0*log10(((*mean)*(*mean))/(*var));

        if (writeFlag) {
                fprintf(fp,"\n\n\t\tDESCRIPTION OF DATA");
                fprintf(fp,"\n\t\t------------------");
                fprintf(fp,"\n\n\tNumber of points:\t%12d",n);
                fprintf(fp,"\n\tMean:\t\t\t%12.4f",*mean);
                fprintf(fp,"\n\tStandard deviation:\t%12.4f",sqrt(*var));
                fprintf(fp,"\n\tVariance:\t\t%12.4f",*var);
                fprintf(fp,"\n\tSum:\t\t\t%12.4f",*sum);
                fprintf(fp,"\n\tSum of squares:\t\t%12.4f",*sqsum);
                fprintf(fp,"\n\tSignal-to-Noise ratio:\t%12.4f",sn);
                fprintf(fp,"\n");
                for (j=1;j<=3;j++) fprintf(fp,
                        "\n\tInterval mean +/- %d*st.dev.:\t(%8.4f,%8.4f)",
                        j,*mean-j*sqrt(*var),*mean+j*sqrt(*var));
        }
        return;
}

/*#############################################################
                        MYFUNC.C
        contains several useful functions, which is
        commonly used in other routines.  The functions
        are:

                - the gamma function
                - the continued fraction beta function
                - the incomplete beta function
                - F probability distribution function
-------------------------------------------------------------*/


/*-------------------------------------------------
                        GAMMA FUNCTION
        This function take a value xx, xx>1, and returns
        the natural logaritm of the gamme function of
        the value.

        Name:   Stein Ove Erikstad
        Date:   June 1992
        Source: Numerical Recipies in C, p.169
-------------------------------------------------*/
float gammaln(xx)
        float xx;
{
        double x,temp,ser;
        static double coeff[6]={76.18009173,-86.50532033,
                24.01409822,-1.231739516,
                0.120858003e-2,-0.536382e-5};
        int j;

        x=xx-1.0;
        temp=x+5.5;
        temp-=(x+0.5)*log(temp);
        ser=1.0;
```

```
        for(j=0;j<=5;j++) {
                x+=1.0;
                ser+=coeff[j]/x;
        }
        return -temp+log(2.50662827465*ser);
}

/*-------------------------------------------------------
        CONTINUED FRACTION BETA FUNCTION
        used in betai

        Name:   Stein Ove Erikstad
        Date:   June 20 1992
        Source: Numerical Recipies in C, p.180
-------------------------------------------------------*/
float betacf(a,b,x)
        float a,b,x;
{
        float qap,qam,qab,em,tem,d;
        float bz, bm=1.0,bp,bpp;
        float az=1.0,am=1.0,ap,app,aold;
        int m;

        qab=a+b;
        qap=a+1.0;
        qam=a-1.0;
        bz=1.0-qab*x/qap;
        for (m=1;m<=ITMAX;m++) {
                em=(float) m;
                tem=em+em;
                d=em*(b-em)*x/((qam+tem)*(a+tem));
                ap=az+d*am;
                bp=bz+d*bm;
                d = -(a+em)*(qab+em)*x/((qap+tem)*(a+tem));
                app=ap+d*az;
                bpp=bp+d*bz;
                aold=az;
                am=ap/bpp;
                bm=bp/bpp;
                az=app/bpp;
                bz=1.0;
                if (fabs(az-aold) < (EPS*fabs(az))) return az;
        }
        errMess("a or b too big, or ITMAX too small in BETACF");
}

/*-------------------------------------------------------
                        INCOMPLETE BETA FUNCTION

        Name:   Stein Ove Erikstad
        Date:   June 1992
        Source: Numerical Recipies in C
-------------------------------------------------------*/
float betai(a,b,x)
        float a,b,x;
{
        float bt;
```

```
        if (x<0.0 || x>1.0) errMess("Bad x in routine BETAI");
        if (x==0.0||x==1.0) bt=0.0;
        else
                bt=exp(gammaln(a+b)-gammaln(a)-gammaln(b)+a*log(x)+b*log(1.0-x));
        if (x < (a+1.0)/(a+b+2.0))
                return bt*betacf(a,b,x)/a;
        else
                return 1.0-bt*betacf(b,a,1.0-x)/b;
}

/*-------------------------------------------------------
                        F PROBABILITY FUNCTION

        Takes the degrees of freedom, df1 and df2, and the
        confidence level alpha, and returns the F value (if
        the F value is >= 1.0).

        The function is based on a serch using the incomplete
        beta function. The algoritm searches first upwards from
        the initial value of f (1.0) until a upper limit of f
        is found. Then the f value is stepwise bracketed by
        comparing the output from the betai-function with the
        desired f-value. The procedure stops when the difference
        between the desired and calculated aplha value is less
        than the accuracy level

        Name:   Stein Ove Erikstad
        Date:   June 1992
        Source:
-------------------------------------------------------------*/
float fvalue(df1,df2,alpha)
        float df1,df2,alpha;
{
        float f,fmin1,fmin2,accuracy=0.0005,step=5.0,a;
        int it=0,itmax=20;

        f=fmin1=fmin2=1.0;
        a=betai(df2*0.5,df1*0.5,df2/(df2+df1*f));

        if (a<alpha)
                printf("\nF<1.0, a=%8.3f",a);
        else {
                while((a>alpha)&&(it<itmax)) {
                                it++;
                                f += it*step;
                                a=betai(df2*0.5,df1*0.5,df2/(df2+df1*f));
                }
                fmin1=f;
                while((abs(alpha-a) > accuracy)&&(it<itmax)) {
                                it++;
                                if (a > alpha)
                                        f+=0.5*fabs(fmin1-fmin2);
                                else
                                        f-=0.5*fabs(fmin1-fmin2);
                                fmin2=fmin1;
                                fmin1=f;
```

```
                    a=betai(df2*0.5,df1*0.5,df2/(df2+df1*f));
                }
        }
        return f;
}
```

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	October 1993	Contractor Report

4. TITLE AND SUBTITLE
Simulation Reduction Using the Taguchi Method

5. FUNDING NUMBERS
NAG 9-616

6. AUTHOR(S)
Farrokh Mistree, Ume Lautenschlager, Stein Owe Erikstad,
Janet K. Allen

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
University of Houston
480 Calhoun
Houston, Texas

8. PERFORMING ORGANIZATION REPORT NUMBER
S-734

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Lyndon B. Johnson Space Center
Houston, TX 77058

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
NASA CR 4542

11. SUPPLEMENTARY NOTES
Farrokh Mistree, Ume Lautenschlager, Stein Owe Erikstad, Janet K. Allen, University of Houston

12a. DISTRIBUTION / AVAILABILITY STATEMENT
National Technical Information Service
5285 Port Royal Road
Springfield, VA 22161
(703) 487-4600

Subject Category: 66

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

A large amount of engineering effort is consumed in conducting experiments to obtain information needed for making design decisions. Efficiency in generating such information is the key to meeting market windows, keeping development and manufacturing costs low, and having high-quality products.

The principal focus of this project is to develop and implement applications of Taguchi's quality engineering techniques. In particular, we show how these techniques are applied to reduce the number of experiments for trajectory simulation of the LifeSat space vehicle. Orthogonal arrays are used to study many parameters simultaneously with a minimum of time and resources. Taguchi's signal-to-noise ratio is being employed to measure quality. A comprise Decision Support Problem and Robust Design are applied to demonstrate how quality is designed into a product in the early stages of designing.

14. SUBJECT TERMS
decision making; quality control; trajectory analysis; simulation; Monte Carlo Method; Taguchi Method; orthogonal function

15. NUMBER OF PAGES
162

16. PRICE CODE

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

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18
298-102