Uncertainty Optimization Applied to the Monte Carlo Analysis of Planetary Entry Trajectories

A Thesis

Presented to

The Faculty of the School of Aerospace Engineering

By

David Wesley Way

In Partial Fulfillment of the Requirements for

The Degree of Doctor of Philosophy

In the School of Aerospace Engineering

Georgia Institute of Technology

Atlanta, GA

July 3, 2001

THESIS APPROVAL

Uncertainty Optimization Applied to the Monte Carlo Analysis of Planetary Entry Trajectories

Approved July 3, 2001:

John R. Olds, Chairman

Eric N. Johnson

Richard W. Powell

Panagiotis Tsiotras

Gerald D. Walberg

DEDICATION

To Lillian Frances

ACKNOWLEDGEMENTS

This research could not have been completed without the help and support of a large number of people. While many people deserve recognition, space prevents me from listing them all. I would, however, like to extend my sincerest gratitude to everyone who has helped me along the way. I would also like to give special thanks to a few people who's support has been invaluable.

First, I would like to thank my advisor, John Olds, for his leadership and for the opportunity that he has given me. Thank you for taking me under your wing and teaching me the art of our craft. You have been a great example to me.

Next, I would like to thank the rest of my committee: Eric Johnson, Dick Powell, Panagiotis Tsiotras, and Jerry Walberg. Thank you for your guidance and for taking time out your very busy careers to assist me in this effort. I greatly appreciate it.

I would also like to thank the people of the Vehicle Analysis Branch for the experience that they have given me. Thank you for sponsoring my research. Thank you to all of you who have helped me out and answered my questions. Thanks especially to Roger Lepsch, Prasun Desai, Scott Striepe, Eric Queen, Mary Kae Lockwood, and Dick Powell. I look forward to working with you all when I graduate.

I could not have completed this research without the help of my fellow students. Thank you to all the members of SSDL. Thank you to Laura Ledsinger and Irene Budianto for paving the way. Thanks to David Garza, Dave McCormick, and John Bradford for keeping me on track. Thank you for your friendship.

Finally, I would like to thank all of my family for their love and support. I would like to thank especially my parents and grandparents for all they have given me. You have given me so much. Thank you just for being there for me. Thank you most of all to my wife, Danielle, for her patience and thoughtfulness. The more difficult my undertaking, the more you have strived to ease my load. Thank you so much. I could not have done this without you.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

$\tilde{f}$	Augmented objective function
$\nabla\tilde{f}$	Gradient of augmented objective function
$\tilde{y}$	Metamodel response
ℓ	Number of equality constraints
f	Objective function
$\hat{x}, \hat{y}, \hat{z}$	Spatial coordinates, GCE frame
λ	Lagrange multiplier
θ	Longitude or rotation angle
ε	Reference ellipsoid eccentricity
μ	Uncertainty mean
a	Semi-major axis or LCG multiplier
b	Semi-minor axis or scale parameter
b_0, b_i, b_{ij}	Metamodel coefficients
c	Ellipse surface coefficient or LCG increment
F	Cumulative distribution function
g	Inequality constraint
h	Equality constraint
H	Hessian
J	Jacobian
k	Shape parameter
L	Latitude or lower triangular matrix
m	Number of inequality constraints or LCG modulus
n	Number of design variables
p	Probability

P	Permutation matrix
r	Radius
R	Range from target (miss-distance)
$\mathrm{r_e}$	Reference ellipsoid equatorial radius
$\mathrm{r_p}$	Reference ellipsoid polar radius
s^2	Sample variance
U	Upper triangular matrix
w	Weighting factor
x	Design variable
X	Design variables, including Lagrange multipliers
x^0	Design variable, baseline value
Σ	Covariance matrix
ρ	Correlation coefficient
σ	Uncertainty standard deviation

NOMENCLATURE

AFT	Atmospheric Flight Team
BVN	Bivariate Normal
CCD	Central Composite Design
cdf	Cumulative Distribution Function
CFD	Computational Fluid Dynamics
cpq	Cost-Plus-Quadratic
cr	Cross-range
DOE	Design Of Experiments
DOF	Degrees of Freedom
dr	Down-range
EDL	Entry, Descent, and Landing
ERC	Earth Return Capsule
GA	Genetic Algorithm
GCE	Geocentric Equatorial reference frame
GRAM	Global Reference Atmospheric Model
IMU	Inertial Measuring Unit
K-R	Kinderman-Ramage
K-S	Kolmogorov-Smirnov
LaRC	Langley Research Center
LAURA	Langley Aerothermodynamic Upwind Relaxation Algorithm
LCG	Linear, Congruential Generator
Mars-GRAM	Mars Global Reference Atmospheric Model
MAV	Mars Ascent Vehicle
mcp	Monte Carlo POST
MCS	Monte Carlo Simulation
METEOR	Multiple Experiment Transporter to Earth Orbit and Return

MPP	Most Probable Point
MSP	Mars Surveyor Program
MTV	Mars Transfer Vehicle
NASA	National Aeronautics and Space Administration
NPSOL	Nonlinear Programming SOLver
OEC	Overall Evaluation Criteria
OVAAT	One-Variable-At-A-Time
pdf	Probability Distribution Function
Perl	Practical Extraction and Reporting Language
PLU	Permutation, Lower triangular, Upper triangular
POST	Program to Optimize Simulated Trajectories
RM	Recovery Module
RMS	Root Mean Square
RSE	Response Surface Equation
RSM	Response Surface Methodology
SGI	Silicon Graphics, Inc.
SRC	Sample Return Capsule
UTTR	Utah Test and Training Range

SUMMARY

Motivation

"Once the cost of opening up the space frontier is no longer prohibitive, we could explore some of our neighboring planetary bodies. We could send people and equipment to Mars … One day, we could open up exploration throughout our solar system. Then, we could answer the questions that people have been asking about the distant planets since they were first discovered … It is our destiny." – Daniel S. Goldin, NASA Administrator (1997)

Recently, strong evidence of liquid water under the surface of Mars and a meteorite that might contain ancient microbes have renewed interest in Mars exploration. With this renewed interest, NASA plans to send spacecraft to Mars every opportunity (approximately every 26 months). These future spacecraft will return higher-resolution images, make precision landings, engage in longer-ranging surface maneuvers, and even return Martian soil and rock samples to Earth.

Future robotic missions (such as the Mars Surveyor Program 2007 smart lander) and any human missions to Mars will require precise entries to ensure safe landings near science objectives and pre-deployed assets. Potential sources of water and other interesting geographic features are often located near hazards, such as within craters or along canyon walls. In order for more accurate landings to be made, spacecraft entering the Martian atmosphere need to use lift to actively control the entry. This active guidance results in much smaller landing footprints (the probabilistic area that includes nearly all possible landing locations).

Planning for these missions will depend heavily on Monte Carlo analysis. Monte Carlo trajectory simulations have been used with a high degree of success in recent planetary exploration missions. These analyses ascertain the impact of off-nominal conditions during a flight and account for uncertainty. Uncertainties generally stem from limitations in manufacturing tolerances, measurement capabilities, analysis accuracies, and environmental unknowns. Thousands of off-nominal trajectories are simulated by randomly dispersing uncertainty variables and collecting statistics on forecast variables.

The dependability of Monte Carlo forecasts, however, is limited by the accuracy and completeness of the assumed uncertainties. This is because Monte Carlo analysis is a forward driven problem; beginning with the input uncertainties and proceeding to the forecasts outputs. It lacks a mechanism to affect or alter the uncertainties based on the forecast results. If the results are unacceptable, the current practice is to use an iterative, trial-and-error approach to reconcile discrepancies.

Therefore, an improvement to the Monte Carlo analysis is needed that will allow the problem to be worked in reverse. In this way, the largest allowable dispersions that achieve the required mission objectives can be determined quantitatively. This is necessary to:

(1) speed the design process by eliminating the trial-and-error approach;

(2) assist decision-makers by providing additional insight into program risk and uncertainty;

(3) provide system requirements that are justifiable; and

(4) remove some of the subjectivity in uncertainty extrema inherent in the current Monte Carlo process.

Objectives

The proposed research includes optimizing the uncertainties in the Monte Carlo analysis of spacecraft landing footprints. This approach is based on the assumptions that: (1) the engineer can control dispersions of uncertainty variables by altering their probability density ($\pm 3\sigma$ extrema) and (2) that there exists a real cost to changing any extremum from the baseline.

First, a proof-of-concept problem was used to evaluate the feasibility of the optimization method. A simple test problem was chosen so that it could be exhaustively examined with minimal computational expense. The problem was limited to only two design variables to facilitate visualization of the design space.

Next, the optimization method was further demonstrated on the Mars Surveyor Program 2001 Lander. The nominal trajectory for this problem was obtained from the Langley Research Center (LaRC) Vehicle Analysis Branch. The purpose of this example was to demonstrate that the methodology developed during the proof-of-concept could be applied to solve larger, more complicated, "real-world" problems.

Methodology

A metamodel is used to first write polynomial expressions for the semi-major and semi-minor axes of the landing footprint as functions of the independent uncertainty extrema. The coefficients of the metamodel are determined by performing experiments, where each experiment consists of performing a Monte Carlo analysis, constructing a footprint, and recording the size of the footprint semi-major and semi-minor axes. The metamodel is used in a constrained optimization procedure to minimize a cost-tolerance function. Figure 5 shows the methodology flowchart.

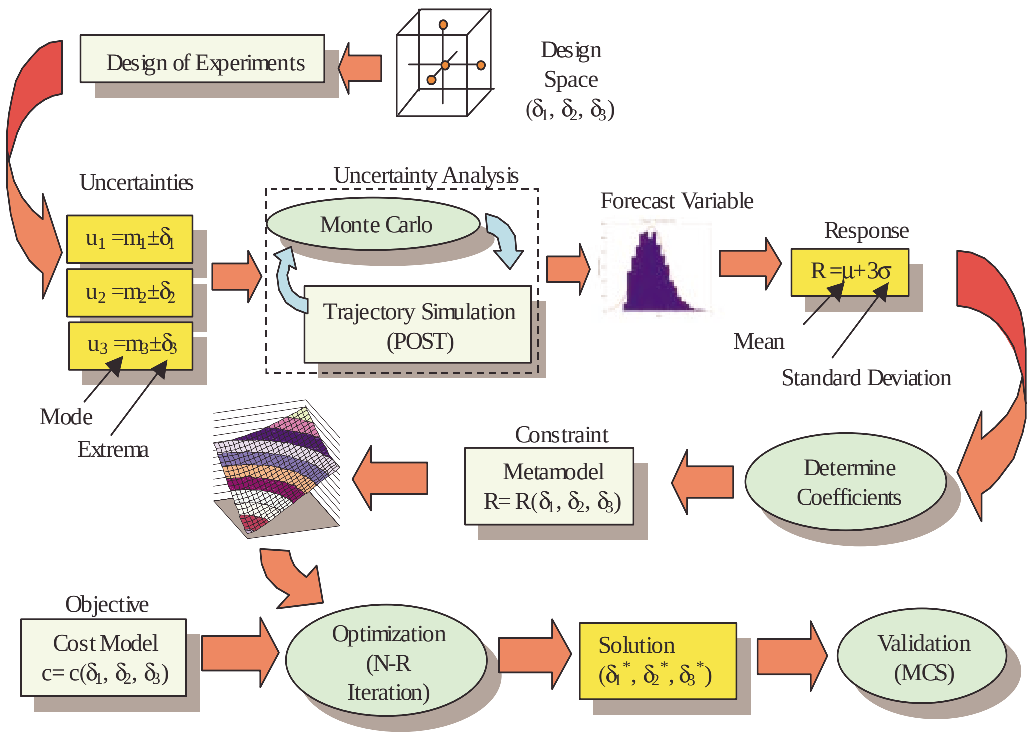


Figure 5: Methodology Flowchart.

<u>Optimization Procedure</u>

The following nine steps outline the optimization procedure.

1) Prepare the nominal trajectory simulation. Model the Entry, Descent, and Landing (EDL) sequence, obtain aerodynamic and mass properties, and optimize the trajectory.

2) Construct the Monte Carlo simulation. First, identify the uncertainties by name, nominal value, distribution, and extrema (minimum and maximum). Second, identify where each uncertainty will go in the POST input deck. This is done by placing markers (e.g., ****name****) in place of values in the nominal input deck.

3) Select the number of simulations, *s*, from Table 13 that gives the desired confidence interval at the desired confidence level. These confidence intervals are only applicable to footprints that assume bivariate normal distributions of down-range and cross-range. Table 13 is repeated here for convenience.

Table 13: Confidence Interval Ratios for Ellipse Semi-axes.

Conf.	*s*=2000	*s*=4000	*s*=6000	*s*=8000	*s*=10000
90%	0.975 - 1.027	0.982 - 1.019	0.985 - 1.015	0.987 - 1.013	0.989 - 1.012
95%	0.970 - 1.032	0.979 - 1.022	0.982 - 1.018	0.985 - 1.016	0.986 - 1.014
99%	0.961 - 1.042	0.972 - 1.030	0.977 - 1.024	0.980 - 1.021	0.982 - 1.019

4) Run the baseline Monte Carlo analysis using the initial guesses for each uncertainty extrema. Plot and evaluate the footprint and compare it to the target.

5) Run experiments to determine the ellipse surface metamodel coefficients, $b_{i,j}$. The first experiment is run with all design variables (uncertainty extrema), x_i, set to zero. Experiments are also run for each design variable where every other design variable is set to zero. Since there are n design variables and s simulations per experiment (runs per Monte Carlo), then n+1 experiments (Monte Carlo analyses) and $s(n+1)$ individual trajectory simulations are required.

6) Solve the 2-by-2 system of equations, Equation 78, for the Lagrange multipliers, λ_0 and λ_1. The partial derivatives in the Jacobian are given by Equation 79. The constraints, h_k, are given by Equation 77. Once the solution is found, substitute the Lagrange multipliers into Equation 74.

$$\begin{bmatrix} \dfrac{\partial h_0}{\partial \lambda_0} & \dfrac{\partial h_0}{\partial \lambda_1} \\ \dfrac{\partial h_1}{\partial \lambda_0} & \dfrac{\partial h_1}{\partial \lambda_1} \end{bmatrix} \begin{bmatrix} \Delta\lambda_0 \\ \Delta\lambda_1 \end{bmatrix} = \begin{bmatrix} -h_0 \\ -h_1 \end{bmatrix} \tag{78}$$

$$\frac{\partial h_k}{\partial \lambda_j} = \sum_{i=1}^{n} -2\left(-w_i x_i^0\right)^{\frac{2}{3}} \left\{ 3\left(\lambda_0 b_{i,o} + \lambda_1 b_{i,1} - \frac{1}{3} w_i \left[\frac{1}{x_i^0} \right]^2 \right) \right\}^{-\frac{5}{3}} b_{i,j} b_{i,k} \tag{79}$$

$$h_k = b_{0,k} + \sum_{i=1}^{n} b_{i,k} \left\{ \frac{-\frac{1}{3} w_i x_i^0}{\lambda_0 b_{i,o} + \lambda_1 b_{i,1} - \frac{1}{3} w_i \left[\frac{1}{x_i^0} \right]^2} \right\}^{\frac{2}{3}} - R_k^2 = 0 \qquad k = 1,2 \qquad \textbf{(77)}$$

$$x_i = \left\{ \frac{-\frac{1}{3} w_i x_i^0}{\lambda_0 b_{i,o} + \lambda_1 b_{i,1} - \frac{1}{3} w_i \left[\frac{1}{x_i^0} \right]^2} \right\}^{\frac{1}{3}} \qquad \textbf{(74)}$$

7) Validate the solution. Use the optimum set of extrema to run a single Monte Carlo analysis at the solution point. Plot and evaluate the footprint and compare it to the target.

8) Evaluate whether the new footprint is sufficiently close to the desired ellipse. If it is sufficiently close, continue. If not, set the initial design variable guesses to the optimized design variables from step (6). Repeat steps (5) through (8).

9) Determine if the optimized uncertainties are physically and economically achievable. If they are achievable, end. If not, reevaluate the uncertainty weightings, w_i. Repeat steps (6) through (9).

<u>Accomplishments</u>

This research includes the following ten specific accomplishments.

1) By employing an optimization methodology, it has been shown that is possible to control the size of the landing footprint and establish optimal tolerances for mission uncertainties.

2) A simplified ellipse surface metamodel was developed, which is equivalent to a response surface without linear or cross terms. This simplified model is enabling

for realistic problems because the computational expense associated with response surface methodology is prohibitive for problems with more than a few uncertainties.

3) A cost-plus-quadratic objective function was formulated that provides a more stable numerical problem because of the following two desirable characteristics: (1) the cost becomes infinite as the uncertainty is reduced to zero, and (2) the unconstrained minimum occurs when the uncertainties are set to their initial values.

4) Equations were presented for three possible objective functions: the reciprocal model, the minimum-distance model, and the cost-plus-quadratic model. These different models allow the user to tailor the method to a particular problem.

5) A technique for solving a constrained optimization with many design variables was explained. This approach included the classical methods of Lagrange multipliers, Newton-Raphson iteration, and LU decomposition.

6) Five methods of constructing footprints were described in detail. Advantages and disadvantages of each were discussed.

7) The bivariate normal (BVN) method of constructing a footprint ellipse was recommended as the most general, elegant, and statistically sound method. A procedure for constructing the ellipse and measuring the semi-major and semi-minor axes was presented. The use of this method also allows the calculation of confidence intervals on the predicted size of the ellipse.

8) A confidence interval on the size of the BVN footprint was derived, based on the confidence interval for the sample variance. Because the confidence interval is a function of sample size, it may be used to determine, *a priori*, how many simulations must be performed to achieve a desired accuracy in the size of the footprint.

9) Two random number generators were evaluated for their appropriateness in simulating random processes. These functions were compared against each other in a series of seven tests. A head-to-head comparison in the actual optimization process showed that both functions produced similar results.

10) The Monte Carlo analysis was automated with the Monte Carlo POST (mcp) program. This program, written in Perl, orchestrates the flow of information needed to perform Monte Carlo simulations. Mcp also has the capability to run multiple simulations in parallel on multi-processor computers.

CHAPTER 1

INTRODUCTION

1.1 History and Exploration of Mars

"Perhaps it is appropriate Mars was named for the Roman god of war, because it consistently has savaged earthlings' attempts to conquer it."
– Steven Siceloff, Florida Today (2001)

Mars has intrigued man since before recorded history. In some of the earliest writings of Mars (nearly 1600 BC), the ancient Babylonians described its red color and the way it appeared to wander among the other stars. In fact, the origin of the word *planet* is derived from Greek meaning "wanderer". The ancient Egyptians also wrote of Mars. Their name for the planet was *Har Descher*, meaning "the red one". This reddish color reminded many ancient cultures of the fire and blood of war. The Greeks, therefore, named the planet for their god of war, Ares. The Romans followed suit and named the planet for their god of war, Mars. [1]

In mythology, Mars (Ares) was a bloodthirsty warrior, driven by rage and a savage love for violence. [2] He was said to have enjoyed the very noise of battle. Rushing headlong into battle, he was followed by the twins, Phobos ("Fear") and Deimos ("Panic"). Mars was also said to have fathered Romulus and Remus, the founders of the Rome. In (pre-Christian) Roman religion, Mars held a very prominent position, second only to his father, Jupiter. [3] The month March was named in honor of Mars, as was the day Tuesday. For example, the French word for Tuesday is *Mardi*, which is derived from Mars. Mars is often portrayed as a warrior in full battle armor, wearing a crested helmet, and carrying a shield and spear. His shield and spear form the astronomer's symbol for the planet, ♂.

The ancient peoples, however, could only observe Mars (the planet) with the naked eye. Galileo Galilei was the first to look upon Mars with the aid of a telescope, which he invented in 1609. [4] His observations showed that Mars was a sphere, like Earth. In 1877 Giovanni Schiaparelli published the first modern map of the features of Mars, using names from history and mythology. His map became famous when he described the thin lines he observed as *canali*. [1] Figure 1 shows Schiaparelli's 1888 map.

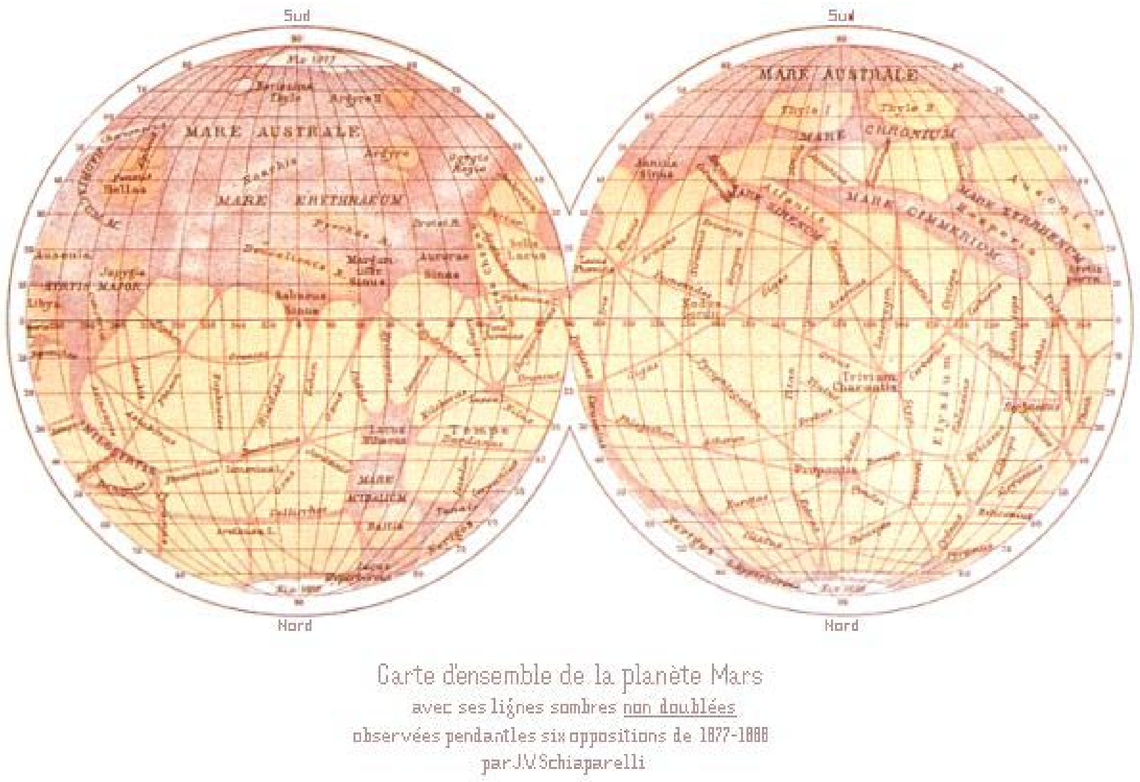


Figure 1: Schiaparelli's Map of Mars (1888).

In 1965, the spacecraft Mariner 4 took the first close-up pictures of Mars when it flew within 10,000 km of the surface. In 1971, Mariner 9 was the first spacecraft to orbit Mars (or any other planet besides Earth). [5] July 20, 1976, Viking 1 landed on Mars and took the first pictures of the Martian landscape. It was soon followed by Viking 2. Each of the Viking missions was actually two spacecraft in one, an orbiter and a lander. The

Viking missions were highly successful. However, it would be another twenty years before another spacecraft successfully completed a mission to the red planet. Between 1982 and 1996, four spacecraft failed in their attempts. [5]

On July 4, 1997, Mars Pathfinder ended the slump and delivered the six-wheeled Sojourner rover safely to the Ares Vallis flood plain. In September of that year, the Mars Global Surveyor entered Martian orbit and began aerobraking (a technique of repeatedly dipping into the atmosphere to lower the orbit). The pictures being sent back from Mars Global Surveyor have 10 to 100 times better resolution than any previous images.

Proving once again that Mars missions are difficult, NASA lost two back-to-back missions in 1999, the Mars Climate Orbiter and the Mars Polar Lander. In fact, of 30 Mars missions launched before 1997 by the United States, the former Soviet Union, Russia, and Japan, only 40 percent have been successful. [6] Table 1 lists all the past missions to the red planet as well as those currently underway. [7] This table also indicates the general nature of any failures.

Table 1: Missions to Mars.

Mission	Country	Launch	Comments
Marsnik 1 (1960A)	USSR	10 Oct. 1960	Flyby (Launch Failure)
Marsnik 2 (1960B)	USSR	14 Oct. 1960	Flyby (Launch Failure)
Sputnik 22	USSR	24 Oct. 1962	Flyby (Launch Failure)
Mars 1	USSR	1 Nov. 1962	Flyby (Contact Lost)
Sputnik 24	USSR	4 Nov. 1962	Lander (Launch Failure)
Mariner 3	USA	5 Nov. 1964	Flyby (Improper Orbit)
Mariner 4	USA	28 Nov. 1964	Mars Flyby
Zond 2	USSR	30 Nov. 1964	Mars Flyby (Contact Lost)
Zond 3	USSR	18 July 1965	Lunar Flyby, Mars Test Vehicle
Mariner 6	USA	25 Feb. 1969	Mars Flyby
Mariner 7	USA	27 Mar. 1969	Mars Flyby
Mars 1969A	USSR	27 Mar. 1969	Orbiter (Launch Failure)
Mars 1969B	USSR	2 Apr. 1969	Orbiter (Launch Failure)
Mariner 8	USA	8 May 1971	Flyby (Launch Failure)
Cosmos 419	USSR	10 May 1971	Orbiter/Lander (Failed Insertion)
Mars 2	USSR	19 May 1971	Orbiter/Lander (Crash)
Mars 3	USSR	28 May 1971	Orbiter/Lander (Failed at 110 sec)
Mariner 9	USA	30 May 1971	Mars Orbiter
Mars 4	USSR	21 Jul. 1973	Orbiter (Failed orbit insertion)
Mars 5	USSR	25 Jul. 1973	Orbiter (Failed shortly after orbit)
Mars 6	USSR	5 Aug. 1973	Lander (Crash)
Mars 7	USSR	9 Aug. 1973	Lander (Missed Mars)
Viking 1	USA	20 Aug. 1975	Orbiter and Lander
Viking 2	USA	9 Sep. 1975	Orbiter and Lander
Phobos 1	USSR	7 Jul. 1988	Phobos Landers (Lost Contact)
Phobos 2	USSR	12 Jul. 1988	Phobos Landers (Power Failure)
Mars Observer	USA	25 Sep. 1992	Orbiter (Contact Lost)
Mars Global Surveyor	USA	07 Nov. 1996	Orbiter
Mars 96	Russia	16 Nov. 1996	Orbiter/Landers (Failed Insertion)
Mars Pathfinder	USA	04 Dec. 1996	Lander and Rover
Nozomi (Planet-B)	Japan	3 Jul. 1998	Orbiter (Delayed Arrival)
Mars Climate Orbiter	USA	11 Dec. 1998	Orbiter (Navigation Error)
Mars Polar Lander	USA	3 Jan. 1999	Lander (Lost Contact)
Deep Space 2 (DS2)	USA	3 Jan. 1999	Penetrators (Unknown Fate)
2001 Mars Odyssey	USA	7 Apr. 2001	Orbiter (Arrival 24 Oct. 2001)

Since the Mars Pathfinder mission, NASA has instituted a policy of sending spacecraft to Mars at every opportunity (approximately every 26 months). Following the losses of the Mars Climate Orbiter and the Mars Polar Lander, this pace slowed and the Mars Surveyor Program 2001 lander was cancelled. The 2001 Mars Odyssey Orbiter is currently en route to the planet and will begin aerobraking in October 2001.

1.2 The 4th Rock from the Sun

"NASA has made a startling discovery that points to the possibility that a primitive form of microscopic life may have existed on Mars more than three billion years ago." – Daniel S. Goldin, NASA Administrator (1996)

So what is known of the fourth planet from our Sun? In many ways, Mars is more like Earth than any other body in our solar system. It has mountains, volcanoes, canyons, valleys, polar ice caps (mostly dry ice), and dry riverbeds. It has seasons, an atmosphere, clouds, winds, and dust storms. [1]

Table 2 lists many of the physical characteristics of Mars and compares them with Earth. [5] Mars is smaller than Earth, it has approximately one-third the gravity, and it has two moons (Phobos: "Fear" and Deimos: "Panic"). Mars also has a very thin atmosphere, the approximate composition of which is: carbon-dioxide, 95%; nitrogen, 3%; argon, 1.5%; and trace amounts of oxygen and water. [8] The surface pressure on Mars is less than 1% that of Earth's atmosphere.

Table 2: Mars Physical Parameters.

Parameter	Mars	Earth
Mass (10^{24} kg)	0.64185	5.9736
Volume (10^{10} km^3)	16.318	108.321
Equatorial radius (km)	3397	6378.1
Polar radius (km)	3375	6356.8
Volumetric mean radius (km)	3390	6371.0
Eccentricity	0.00648	0.00335
Mean density (kg/m^3)	3933	5515
Surface gravity (m/s^2)	3.69	9.78
Escape velocity (km/s)	5.03	11.19
Gravitational parameter (10^6 km^3/s^2)	0.04283	0.3986
Solar irradiance (W/m^2)	589.2	1367.6
Black-body temperature (K)	210.1	254.3
J2 (10^{-6})	1960.45	1082.63
Natural satellites	Phobos, Deimos	Moon

Table 3 lists several of the orbital parameters associated with Mars. [5] Mars is 50% farther from the sun than Earth. As Earth and Mars circle the sun in their respective elliptical orbits, their closest point of approach (when they are both on the same side of the sun) is 34.5 million miles. When Earth and Mars are at their farthest distance (on opposite sides of the sun), they are separated by 247 million miles. It is interesting to note that Earth and Mars have nearly the same rotation rate (length of day) and tilt of their axes (which is responsible for the seasons). The Martian year is almost twice as long as Earth's.

Table 3: Mars Orbital Parameters.

Parameter	Mars	Earth
Semi-major axis (10^6 km)	227.92	149.60
Sidereal orbit period (days)	686.980	365.256
Perihelion (10^6 km)	206.62	147.09
Aphelion (10^6 km)	249.23	152.10
Mean solar distance (AU)	1.524	1.000
Synodic period (days)	779.94	-
Mean orbital velocity (km/s)	24.13	29.78
Max. orbital velocity (km/s)	26.50	30.29
Min. orbital velocity (km/s)	21.97	29.29
Orbit inclination (deg)	1.850	0.000
Orbit eccentricity	0.0935	0.0167
Sidereal rotation period (hrs)	24.6229	23.9345
Length of day (hrs)	24.6597	24.0000
Obliquity to orbit (deg)	25.19	23.45

So how would Mars appear to an explorer? An explorer standing on the surface would surely survey a breath-taking scene. Mars has reddish-orange sand, volcanic rocks, and an eerie sky, which has been described as butterscotch or salmon pink in color. The surface is an exceptionally dry, cold, and desolate place. The landscapes are magnificent; impressive cliffs, rolling sand dunes, majestic canyons, ancient impact craters, and looming volcanoes. In fact, Mars has both the highest mountain (Olympus Mons) and the deepest canyon (Valles Marineris) known in the solar system.

While scientists have learned much about the planet, three major questions remain unanswered. First, why does Mars lack plate tectonics? Second, does liquid water exist on Mars? Third, has life ever existed on Mars? The answers to these questions are fundamentally important because they will help us understand more about our own planet.

1.3 Future Exploration of Mars

"Once the cost of opening up the space frontier is no longer prohibitive, we could explore some of our neighboring planetary bodies. We could send people and equipment to Mars … One day, we could open up exploration throughout our solar system. Then, we could answer the questions that people have been asking about the distant planets since they were first discovered ... It is our destiny." – Daniel S. Goldin, NASA Administrator (1997)

Recently, strong evidence of liquid water under the surface of Mars and a meteorite that might contain ancient microbes, have renewed interest in Mars exploration. NASA has resumed an active program of robotic spacecraft exploration of Mars. Gradually increasing mission complexity in a "stepping stone" approach, these spacecraft will return higher-resolution images, make precision landings, engage in longer-ranging surface maneuvers, and even return Martian soil and rock samples to Earth. [9] Planners hope to launch a sample return mission as early as 2009 [10], though it is more likely to be 2014.

Currently three spacecraft are scheduled for the 2003 opportunity. The first two are identical landers, which will carry Mars Exploration Rovers to separate sites in different regions of the planet. These rovers will be able to travel across 100 meters of rugged surface each Martian day (sol). The landing technique will be identical to the Pathfinder mission's direct entry. Airbags will cushion the landing following a descent on a parachute. When the spacecraft stops bouncing, the airbags will deflate and three petals will open up to expose the rover. [9]

The third mission planned for 2003 is a joint mission with the European Space Agency (ESA) and the Italian space agency. Called Mars Express, this orbiter/lander combination will explore the atmosphere and surface of Mars from a polar orbit, searching for sub-surface water. The small lander is named “Beagle 2” after Charles Darwin’s ship.

Only one mission, the Mars Reconnaissance Orbiter, is currently scheduled for the 2005 opportunity. This orbiter will measure Martian landscapes at 20- to 30-centimeter (8- to 12-inch) resolution. The best images currently from Mars Global Surveyor are in the 1- to 2-meter resolution range.

In 2007, NASA proposes to launch a “smart lander” with precision landing capability and hazard avoidance technology. This lander will deliver a long-range, long-duration rover that will pave the way for future sample-return missions. The mission is expected to explore very promising, but difficult-to-reach, scientific sites.

NASA [9] reports that current plans call for the first sample return mission to be launched in 2014, with the second in 2016. However, options currently under study could move the first sample return mission to as early as 2011.

> *“NASA’s robotic Mars exploration program has received a budgetary boost from the Bush Administration, perhaps moving closer the day when a human trek to the red planet becomes feasible.” – Leonard David, Space.com (2001)*

These future spacecraft may provide answers to fundamental questions concerning the presence of water and life on Mars. [5] The Mars Exploration Program’s overall science strategy is to “follow the water”. The four science goals that support this strategy are: (1) determine whether life does exist, or has ever existed, on Mars; (2) study the current climate of Mars; (3) study the geology of Mars; and (4) prepare for human exploration. [11]

So when will humans explore Mars? Currently no space agency has concrete plans for a manned mission to Mars. However, a human mission could fly within the next few decades. It is expected that human missions, whenever they occur, will be the beginning of a permanent presence on the red planet.

> *". . . if they decide to do so, our descendants can create a New Earth - perhaps a New Eden - on the next world outwards from the Sun." - Arthur C. Clarke, science fiction author*

Getting astronauts to the surface of Mars and returning them safely to Earth, however, is extremely difficult. It requires the development of many key technologies. One such technology is the development of precision landing capability. In order for more accurate landings to be made, spacecraft entering the planet's atmosphere need to use the atmosphere's lift and drag to guide the spacecraft towards to the intended target. This active control of the entry and computer targeting results in much smaller landing footprints (the probabilistic area that includes nearly all possible landing locations). [12].

This chapter introduces Mars and planetary exploration. It is in the context of this exploration that the research of this thesis is performed. The subject of this research is the Monte Carlo analysis of spacecraft trajectories (flight paths). In particular, this research studies the trajectories of spacecraft entering the Martian atmosphere and performing precision landings on its surface.

1.3.1 Thesis Organization

This thesis is organized into eleven chapters and two appendices:

Chapter 1 is the introduction. This chapter introduces planetary exploration, in a non-technical way, by discussing the history and characteristics of Mars as well as past, present, and planned future missions. The purpose of this chapter is to acquaint the unfamiliar reader with the context in which Monte Carlo analyses are performed.

Chapter 2 introduces the research problem. This chapter presents the motivation for the research, expresses research goals, establishes specific objectives for the research, and outlines the approach to the research. The purpose of this chapter is to introduce the reader to a shortcoming in the current engineering practice of conducting Monte Carlo simulations, establish a need to correct this shortcoming, and present a proposed solution.

Chapter 3 describes the current state-of-practice in Monte Carlo analyses. This chapter discusses seven missions for which Monte Carlo simulation results have been published: METEOR, Pathfinder, Stardust, Mars Surveyor 2001, Genesis, MUSES-C, and Mars Ascent Vehicle. The purpose of this chapter is to provide a brief background into how the analysis is performed, how many uncertainties are modeled, and how many simulations are run.

Chapter 4 describes the state-of-the-art in design optimization methodologies. This chapter discusses multi-objective function optimization, design variable screening, unconstrained optimization, metamodeling, and probabilistic methods. The purpose of this chapter is to provide background into optimization methodologies and discuss some of the available techniques.

Chapter 5 reviews random number generators. This chapter discusses randomness, how random numbers are generated, and provides tests for random number generators. The purpose of this chapter is to answer whether the random number generator used in this research is appropriate in this application.

Chapter 6 discusses landing footprints. This chapter explains five methods for constructing landing footprints: the 3-sigma radius, Rayleigh, Weibull, 3-sigma down-range and cross-range, and bivariate normal methods. The purpose of this chapter is to examine how the footprints are constructed, what the relevant assumptions are, and what probability is associated with landing within the footprint. This chapter also discusses how to choose an appropriate number of simulations when using the BVN footprint method.

Chapter 7 explains the mechanics of how trajectory simulations were performed for this research. This chapter discusses POST, mcp, and automation scripts. The purpose of this chapter is to present the computer programs needed to perform a large number of Monte Carlo simulations.

Chapter 8 develops the mathematical procedure for solving the optimization problem. This chapter discusses the objective function, constraints, method of Lagrange multipliers, and Newton-Raphson iteration. The purpose of this chapter is to apply the general techniques explained in Appendix A to this specific problem.

Chapter 9 presents the proof-of-concept problem. This chapter defines the simple problem and shows the results. The purpose of this chapter is to validate the methodology.

Chapter 10 presents the Mars Surveyor 2001 Lander example. This chapter describes the mission, the simulations, and the results. The purpose of this chapter is to demonstrate the methodology on a “real world” problem.

Chapter 11 gives the conclusions. This chapter summarizes the method, presents lessons learned, and lists the accomplishments of this research. The purpose of this chapter is to present the findings of this research.

Appendix A provides a review of relevant mathematics. This chapter discusses the method of Lagrange multipliers, Newton-Raphson iterations, PLU matrix decomposition, and Cholesky factoring. All of these techniques are used in this research. The purpose of this chapter is to provide an explanation of these mathematical procedures.

Appendix B provides Perl source code used in the Monte Carlo POST (mcp) program. Appendix B.1 is the Perl module for mcp. Appendix B.2 is the executive program for mcp. Appendix B.3 is the random number module.

CHAPTER 2

RESEARCH SUMMARY

2.1 Motivation

Monte Carlo trajectory simulations have been used with a high degree of success in recent planetary exploration missions [13, 14, 15, 16, 17, 18, 19, 20]. These Monte Carlo analyses use a simple yet powerful method to ascertain the impact of off-nominal conditions during a flight and account for uncertainty. These uncertainties generally stem from limitations in manufacturing tolerances, measurement capabilities, analysis accuracies, and environmental unknowns. Thousands of contingency (off-nominal) trajectories are simulated by randomly dispersing uncertainty variables from their most likely values and collecting statistics on various forecast variables (uncertain outcomes or performance variations).

Flight dynamics personnel have used Monte Carlo analysis throughout every stage of the design process for many planetary missions since its successful validation with the Mars Pathfinder landing in 1998. Even in the conceptual phase, when little definition exists, this procedure allows insight into the robustness of the design and the program risk (the risk of not meeting mission goals and objectives). For example, a particular Mars sample return mission, calling for an orbital rendezvous with an unguided launch vehicle, might be determined to be too risky when the probability of achieving the required orbit is analyzed.

Since the back-to-back failures of Mars Climate Orbiter and Mars Polar Lander in 1999, additional emphasis has been placed on managing risk and uncertainty in flight projects, clearly establishing Monte Carlo analysis as an important step in mission analysis. It is likely that Monte Carlo simulations will be used for all planetary missions in the near future.

Figure 2 illustrates the use of landing ellipses in previous work. This figure shows the Ares Vallis region of Mars chosen for the Pathfinder landing. The large ellipse is the 100-by-200 km target ellipse, determined by science objectives and landing requirements. Future missions, currently under design, require two orders of magnitude reductions in footprint size. The smaller ellipses are a sequence of navigation ellipses generated by the navigation team as the spacecraft neared entry. The shrinking of these ellipses is due to the improved knowledge of actual entry conditions gathered from navigational updates.

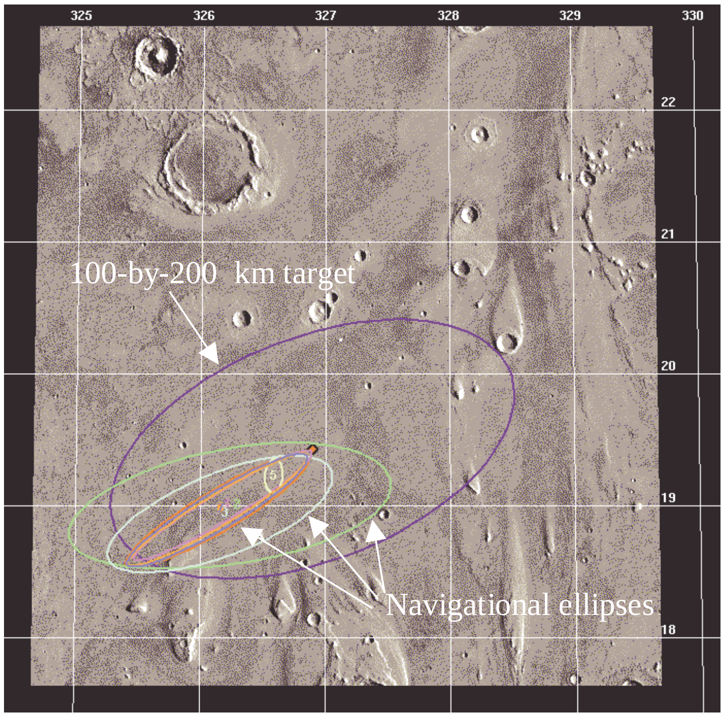


Figure 2: Pathfinder Landing Ellipse.

Future robotic missions (such as the Mars Surveyor Program 2007 smart lander) and any human missions to Mars will require precise entries to ensure safe landings near science objectives and pre-deployed assets. Potential sources of water and other interesting geographic features are often located near hazards, such as within craters or along canyon walls. Astronaut crews, debilitated by long transits in microgravity and physically unable to travel far distances on the surface, will likely need to land very close to life-sustaining resources. These missions will depend heavily on Monte Carlo analysis to ensure that aggressive landing footprint requirements are met. The penalty for not achieving the desired landing accuracy in these hazardous places could be a loss of mission or crew.

The dependability of Monte Carlo forecasts, however, is limited by the accuracy and completeness of the assumed uncertainties (inputs). Monte Carlo analysis is a forward driven problem; beginning with the input uncertainties and proceeding to the forecasts (outputs). By itself, it lacks a mechanism to affect or alter the uncertainties based on the forecast results. If the results are unacceptable (i.e., mission objectives are not met), the current practice is to use an iterative, trial-and-error approach to reconcile the discrepancy. This iteration involves collaboration between flight dynamics personnel and system and subsystem experts from other fields.

The experts are first asked to provide their best estimates for the uncertainties. For a normal (or Gaussian) distribution, these estimates are normally in the form of values for the mean and 99.74% confidence interval (±3 standard deviations). For other distributions, the extrema (minimum and maximum) and the most likely value (mode) are often given. The flight dynamics personnel then conduct the Monte Carlo analysis and evaluate confidence intervals in forecast attributes of the trajectory (such as the size of the landing footprint). If the results fail to meet mission objectives, the flight dynamics personnel again poll the experts – this time to determine if tighter tolerances are possible. Often the only uncertainties targeted in this manner, are those identified through a One-Variable-At-A-Time analyses as the most important drivers. The Monte Carlo process is then repeated with new uncertainty definitions.

An improvement to the Monte Carlo analysis is needed, one that allows the problem to be worked in reverse. A feedback mechanism is desired that would determine, quantitatively, the largest allowable dispersions that achieve the required mission objectives. Because multiple solutions may exist, a means of prioritizing or ranking feasible alternatives is also needed. Such a quantitative feedback mechanism could potentially:

(1) speed the design process by eliminating the trial-and-error approach;

(2) assist decision-makers by providing additional insight into program risk and uncertainty;

(3) provide traceable system requirements; and

(4) remove some of the subjectivity inherent in the current Monte Carlo process.

Optimization reduces design time by replacing the current iterative process. Additionally, once the design space is known (i.e., mapped by a response surface), additional optimizations and trade-studies may be performed at little computational expense – provided that the nominal trajectory is unchanged. If the nominal trajectory changes, the analysis must be repeated.

Feedback provides the decision-maker with cause and effect information. By changing the prioritization (weighted cost) of the design variables, the decision maker can quantitatively evaluate the relative cost of reducing the uncertainty in one system or sub-system over another. Alternatives are compared on equal ground, since mission goals are met precisely (within the limitations of the Monte Carlo simulation) in each case.

The optimum set of dispersion extrema, once established, translates immediately to system and sub-system requirements. These requirements may be established early in the design process and used to design, compare, and test components throughout the life of the program. This process establishes these requirements in a fashion that is easily documented.

The results of Monte Carlo analysis are dependent on the accuracy of the assumptions used in generating the dispersions. The subjective estimation of the mission uncertainties is typically delegated to system experts. As the design matures, and the dispersions are known with increasing accuracy, the Monte Carlo is run again with improved confidence in the results. In contrast, the solution of an optimization problem does not require *a priori* knowledge of the actual dispersions. Rather, the optimization problem establishes a benchmark, with success guaranteed if-and-only-if actual dispersions are reduced below the benchmark. The system experts, while not removed from the process, need only evaluate *a posteriori* whether the actual dispersions can or cannot meet the given requirement.

2.2 <u>Goals</u>

The primary goal of this research was to show that dispersion extrema in Monte Carlo analyses could be optimized to meet mission objectives (e.g., landing footprint constraints). Figure 3 compares landing footprints before and after optimization and shows the desired effect of tightening the uncertainty tolerances. This concept was to be proven using a simple two dimensional problem. Once verified, the method was to be demonstrated on a “real world” problem. This thesis sought to “close the loop” in the Monte Carlo process by providing a mechanism for feedback. The goals of this problem were as follows.

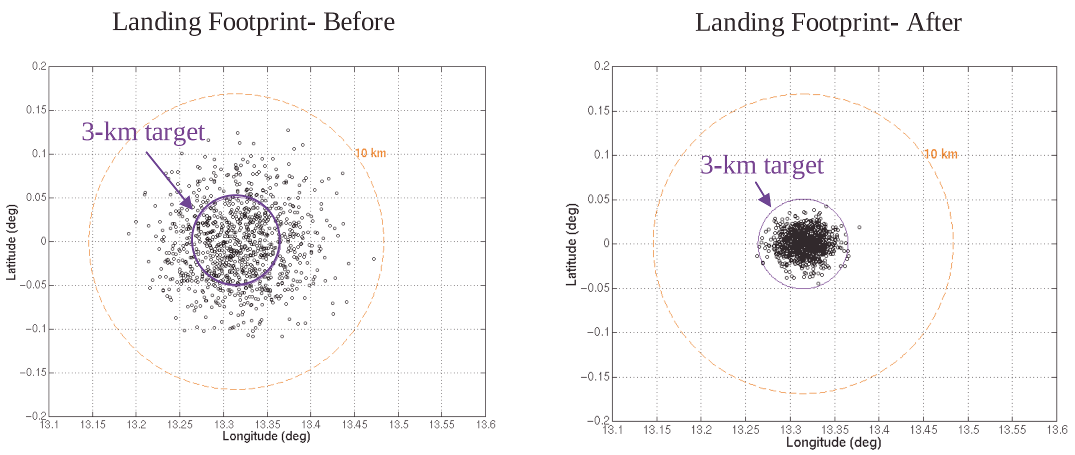


Figure 3: Example Footprint Reduction.

1) Go beyond the current state-of-practice in Monte Carlo analysis of planetary entry trajectories. Show that is possible to control the size of the landing footprint and establish tolerances for mission uncertainties by developing an optimization

methodology. In a sense, this goal is to perform the Monte Carlo analysis backwards: beginning with a desired output and proceeding to the required inputs.

2) Develop a simplified metamodel that has fewer terms than a response surface. This simplified model is required to enable solutions of typical "real world" problems. This is because the computational expense associated with a standard response surface is prohibitive, even for problems with more than a just a few uncertainties.

3) Formulate an objective function that provides for a stable numerical problem. Two desirable objective function characteristics are: (1) the cost becomes infinite as the uncertainty is reduced to zero, and (2) the unconstrained minimum occurs when the uncertainties are set to their initial values.

4) Present several possible objective functions. Different models allow the user to tailor the method to a particular problem.

5) Outline a technique for solving a constrained optimization with many design variables. Write a numerical solver to perform this optimization.

6) Recommend a method of constructing a footprint ellipse that is general, elegant, and statistically sound. Present a procedure for constructing the ellipse, measuring the semi-major and semi-minor axes, and determining the probability of landing within the ellipse.

7) Describe several methods of constructing footprints. These descriptions should include construction techniques, assumptions, and probabilities. These methods may create either circular or elliptical footprints. Discuss the advantages and disadvantages of each.

8) Determine, *a priori*, how many simulations must be performed to achieve a given accuracy in the size of the footprint.

9) Evaluate the appropriateness of using the "ran1" and the Perl "rand" built-in random number generators. Compare the "randomness" of these functions against each other and the Matlab built-in random number generator in statistical tests.

10) Automate the Monte Carlo process such that multiple simulations may be run in parallel on multi-processor computers.

2.3 Objectives

2.3.1 Proof-of-Concept

A proof-of-concept problem was used to evaluate the feasibility of the optimization method. A very simple test problem was chosen so that it could be exhaustively examined with minimal computational expense. The problem was limited to only two design variables to facilitate visualization of the design space. The objectives of this problem were as follows.

1) Successfully locate the minimum cost extrema that satisfy a 3-km landing footprint. Evaluate the appropriateness of the metamodeling technique by comparing three forms of the model based on the quality of their fit: the standard response surface, the squared response surface, and the ellipse surface. Verify each solution several times, calculate the average error (difference from the desired miss-distance), and compare with the other solutions.

2) Provide a visualization of the design space by plotting the results of a gridsearch. Compare the resultant surface with the simplified ellipse surface. This method is very important because it provides a means for extending the methodology to problems with larger numbers of design variables due to the minimal number of function evaluations required.

3) Determine the appropriateness of approximating the range with a metamodel. The development of the ellipse surface suggests that it may be more natural to fit the square of the range, rather than the range itself.

2.3.2 "Real World" Example

The optimization method was demonstrated on a "real-world" problem. This problem had a larger number of uncertainties (twenty-seven) and a more complicated EDL sequence. This EDL simulation included both a guidance algorithm and a lifting entry trajectory. This problem was modeled after an actual proposed flight project, the Mars Surveyor Program 2001 Lander. The objectives of this problem were as follows.

1) Demonstrate that the solution procedure developed in the proof-of-concept can be applied to “real world” problems. Choose a problem with a large number of design variables, such that it can not be solved using a response surface metamodel. Several iterations may be necessary.

2) Evaluate the numerical stability of the problem when using more than one constraint. Use the method of Lagrange multipliers with substitution and compare the solutions obtained by the different objective functions: reciprocal model, minimum-distance model, and cost-plus-quadratic model.

3) Evaluate the appropriateness of the bivariate normal (BVN) footprint assumption. The use of this method assumes that normal distributions are appropriate approximations to the down-range and cross-range distributions. This is important because the BVN method is the only method that allows the calculation of a confidence interval.

2.4 Approach

Figure 4 outlines the steps taken in this research. The first step was to become familiar with current practices and the Monte Carlo analysis. This step was completed through hands-on experience working with the Vehicle Analysis Branch at the NASA Langley Research Center and extended over a period of three summers.

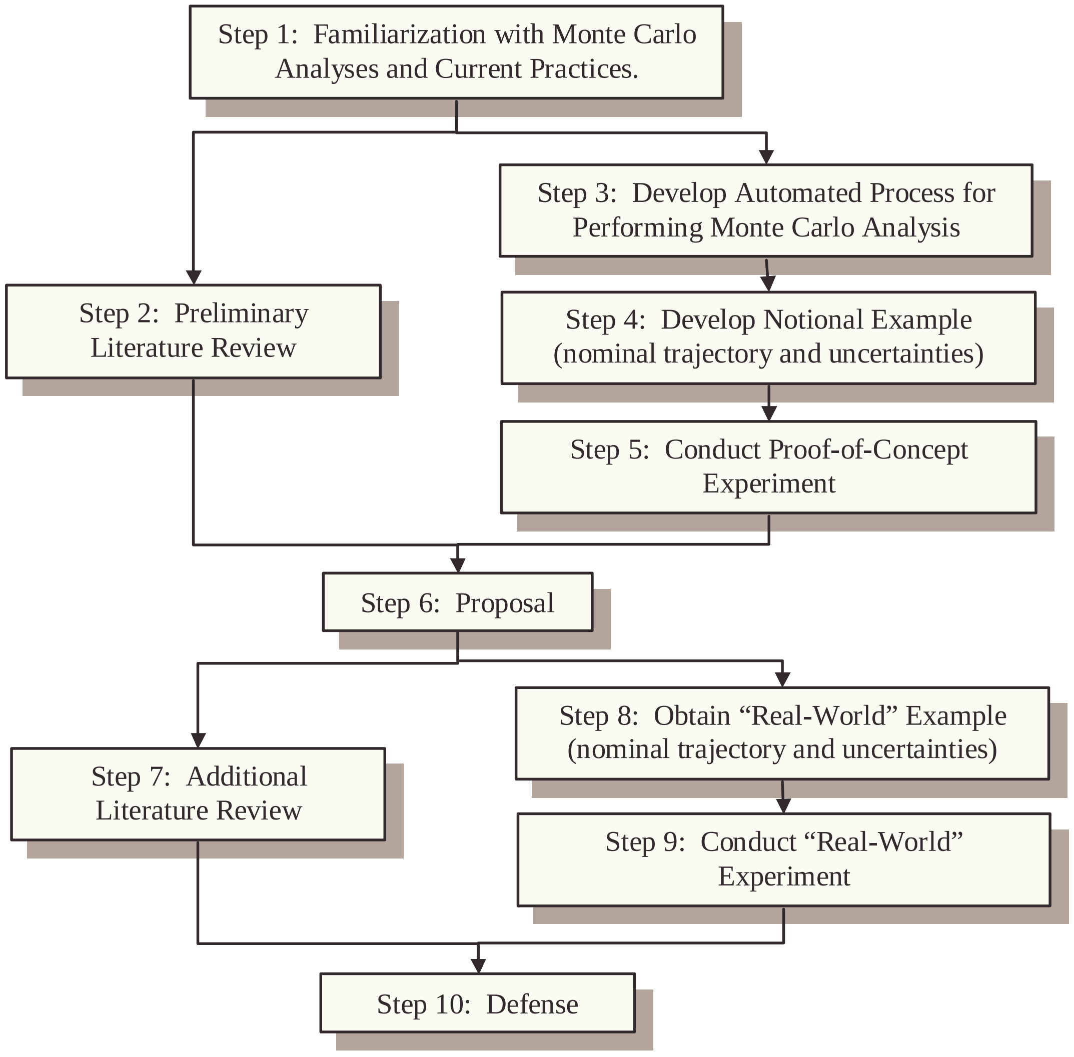

Step 1: Familiarization with Monte Carlo Analyses and Current Practices.
Step 2: Preliminary Literature Review
Step 3: Develop Automated Process for Performing Monte Carlo Analysis
Step 4: Develop Notional Example (nominal trajectory and uncertainties)
Step 5: Conduct Proof-of-Concept Experiment
Step 6: Proposal
Step 7: Additional Literature Review
Step 8: Obtain "Real-World" Example (nominal trajectory and uncertainties)
Step 9: Conduct "Real-World" Experiment
Step 10: Defense

Figure 4: Research Approach.

Next, a literature review was conducted in parallel with a proof-of-concept experiment. The proof-of-concept was designed to be simple, yet still representative of larger problems. Because many Monte Carlo analyses would be needed, a program was developed to automate the process.

These steps led to the thesis proposal where the results of the proof-of-concept were presented to the thesis committee. Once the thesis topic was approved, the method was applied to a "real world" example. Further literature review was conducted as necessary.

The problem chosen for the "real world" example was the Mars Surveyor Program 2001 Lander. The nominal trajectory for this problem was obtained from the Langley

Research Center (LaRC) Vehicle Analysis Branch. The purpose of this example was to demonstrate that the methodology developed during the proof-of-concept could be applied to solve larger, more complicated, problems.

2.4.1 Solution Methodology

This research includes optimizing the uncertainties in the Monte Carlo analysis of spacecraft landing footprints. This approach is based on the assumptions that: (1) the engineer can control dispersions of uncertainty variables by altering their extrema ($\pm 3\sigma$ tolerances) and (2) that there exists a real cost to changing any extremum from the baseline. It follows then, that if the uncertainty dispersions are controllable, then so are the forecast variables. Therefore, a non-unique set of uncertainty extrema exists that results in any desired forecast performance. Additionally, any number of feasible sets of extrema may be ranked according to their associated costs.

A metamodel is used to first write polynomial expressions for the semi-major and semi-minor axes of the landing footprint as functions of the independent uncertainty extrema. In general, any forecast variable may be used in a constraint. The coefficients of the metamodel are determined by performing experiments, where each experiment has been chosen from a design of experiments. Each experiment consists of performing a Monte Carlo analysis, constructing a footprint, and recording the size of the footprint.

An objective function is written for the cost as a function of the uncertainty extrema. An optimization is then performed that minimizes the cost subject to the constraint that the landing footprint is a specified size.

Figure 5 shows the methodology flowchart. The uncertainty design space is sampled according to a design of experiments. A Monte Carlo analysis is performed for each of these uncertainty definitions. Forecast variable data is collected from each of the simulations and used to form a response statistic. The responses are regressed to create a metamodel of the design space. The metamodel is used in a constrained optimization procedure to minimize a cost-tolerance function. The solution of this optimization problem is the desired minimum-cost uncertainty settings, which satisfy the response constraints.

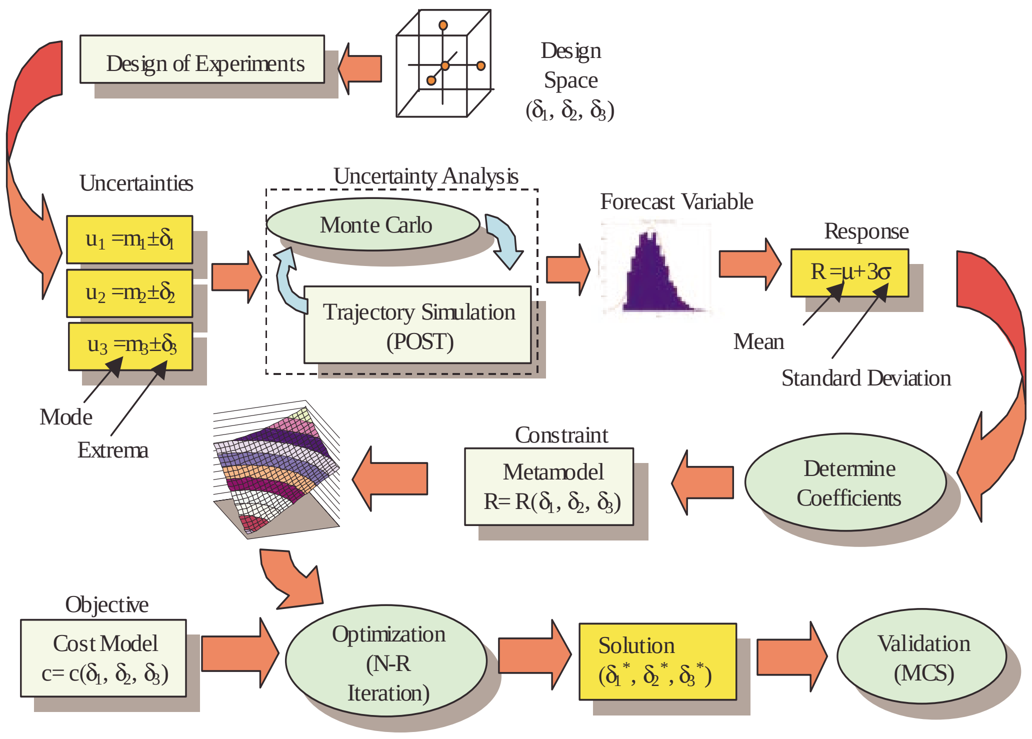


Figure 5: Methodology Flowchart.

Finally, one or more validation runs are performed using the optimum extrema to ensure that the desired outcome is obtained. An example for a Mars entry trajectory would be to determine the allowable dispersions that minimize cost while ensuring (with a 99.5% confidence) that the spacecraft lands within 10 km of the intended landing site.

CHAPTER 3

MONTE CARLO IN TRAJECTORY ANALYSIS

Many recent flight programs have benefited from Monte Carlo simulations. This chapter describes the current state-of-practice in Monte Carlo analyses and discusses seven missions for which Monte Carlo simulation results have been published. These missions are: METEOR, Mars Pathfinder, Stardust, Mars Surveyor Program 2001, Genesis, MUSES-C, and Mars Ascent Vehicle.

This list of missions is not intended to be all-inclusive, but rather represents a wide cross-section of spacecraft and mission requirements. These programs, which include landers, orbiters, and ascent vehicles, illustrate the many ways in which Monte Carlo analyses influence the design of spacecraft. These programs further illustrate: (1) the usefulness of the Monte Carlo technique; (2) typical values for number of simulations and number of uncertainties; (3) typical landing ellipse (footprint) sizes; and (4) some of the limitations of the method.

Several of the missions described here fall under the jurisdiction of NASA's Discovery Program. Discovery is an ongoing program that is intended to offer frequent, high quality science missions through NASA's vision of "Better, Faster, Cheaper". It seeks to keep both performance high (by using new technologies) and expenses low (by setting a cost cap of $299 million for an entire mission). This program represents a dramatic departure from past missions that were very large in scope, and cost billions of dollars. Eight Discovery Missions have been chosen to date: Near Earth Asteroid Rendezvous (NEAR), Mars Pathfinder, Lunar Prospector, Stardust, Genesis, Comet Nucleus Tour (CONTOUR), MErcury Surface Space ENvironment GEochemistry and Ranging (MESSENGER), and Deep Impact. [21]

3.1 <u>METEOR (1995)</u>

The Multiple Experiment Transporter to Earth Orbit and Return (METEOR) spacecraft was designed to fly microgravity experiments in orbit. It was lost in 1995, however, when the Conestoga launch vehicle, which was carrying it, failed during ascent. The Recovery Module (RM) was to splashdown off the coast of Virginia.

A Monte Carlo analysis [17] was performed to assess the splashdown dispersion footprint. The 6-DOF POST simulation included the 1995 Global Reference Atmospheric Model (GRAM-95) [22] atmosphere model as well as Service Module (SM) separation, and yo-yo release and de-spin models. Fifty-seven potential uncertainties were identified. Over 3500 trajectories were simulated to assure a good distribution. From a One-Variable-At-A-Time (OVAAT) sensitivity analysis, the center-of-gravity offset from the spin axis and initial attitude and attitude rate uncertainties where shown to produce the greatest dispersions [17].

The OVAAT results were also used to quickly assess the size of the landing ellipse. By computing the L_2 norm of the dispersions from the OVAAT results, an upper bound on the resulting 3-sigma range from the Monte Carlo analysis of no more than 50-60 nm was expected. The Monte Carlo analysis, in fact, showed a splashdown footprint with axes of 43.3 nm downrange, 33.5 nm up-range, and 10.0 nm cross-range. The unsymmetrical landing ellipse gave a 58% probability that the RM would overshoot the nominal splashdown site. [17]

The METEOR study illustrates a fallibility of the Monte Carlo approach – the subjectivity of the dispersion assumptions. This study predicted a larger (approximately double) footprint than previous METEOR entry dispersion analyses that were preformed earlier in the program [23, 24]. Desai et al. [17] attributed the difference to (1) changes in the baseline orbital altitude, target landing site, and mass properties of the RM, (2) the inclusion of additional uncertainties not originally considered, and (3) more conservative estimates of Isp, solid-motor temperature, weight, initial attitude, and body rates.

3.2 <u>Pathfinder (1997)</u>

On July 4, 1997, the second Discovery mission, Mars Pathfinder, successfully came to rest on the surface of Mars just 27 km from its target point. [18] The target landing area was a 100-by-200 km ellipse in the Ares Vallis floodplain (centered at 19.24 N latitude, 33.1 W longitude). Pathfinder, with its little Sojourner rover, captured the imagination of the public and was an overwhelming success.

It is because of the great success of this mission that Pathfinder serves as an excellent example of how Monte Carlo simulations are used in the design of spacecraft. The design of the entry trajectory, along with the accurate determination of potential trajectory dispersions, was critical to the success of the Pathfinder mission. [13] In fact, the primary objective of the Mars Pathfinder mission was to demonstrate a unique, low-cost, reliable system for entering the Martian atmosphere and placing a lander (along with its science payload) safely on the surface of Mars. [13, 14] It also follows that because Pathfinder was a technology demonstration mission, a critical legacy of the program is the reconstruction of the entry trajectory [14] from actual flight data and its comparison to pre-flight predictions. Indeed, these comparisons are used as evidence to validate the Monte Carlo approach to uncertainty analysis in the design of other spacecraft. [16, 25]

Monte Carlo simulations, first developed early in the design phase, were instrumental throughout the Pathfinder program from design, to testing, to operations, and finally to post mission reconstruction [18, 13]. Simulations performed at the Jet Propulsion Laboratory (JPL) used both the 3-DOF Atmospheric Entry Program (AEP) [26] and the 6-DOF Automated Dynamic Analysis of Mechanical Systems (ADAMS) [27] programs. Simulations performed at the NASA Langley Research Center (LaRC) were run with the Program to Optimize Simulated Trajectories (POST) [28].

Monte Carlo simulations were used to design the Entry, Descent, and Landing (EDL) system to be more robust by accommodating a wide range of off-nominal entry conditions. Additionally, the results of the Monte Carlo simulations directly influenced the design of the Pathfinder Aeroshell Thermal Protection System [29] and contributed to the validation of the parachute deployment algorithm [13]. The Monte Carlo results were also used in the design of flight software, and to define certain hardware tests. [30] Finally, Monte Carlo simulations predicted the size and orientation of the landing ellipse on the Mars surface. [18]

In the operations phase of the mission, the navigation team used Monte Carlo simulations to update flight software parameters as the spacecraft approached Mars. These parameters, based on improved estimates of the atmosphere and vehicle state vector, modified the EDL sequence to account for the most likely atmospheric flight conditions. Without this update capability, which improved the probability of a successful parachute deployment, the likelihood of a successful landing would have been adversely affected. [18]

3.3 Stardust (1999)

Stardust is the fourth NASA Discovery mission. The primary goal of Stardust is to collect samples from the tail of the Comet Wild 2 (pronounced "Vilt Two"). Stardust is the first U.S. space mission dedicated solely to the exploration of a comet, and the first robotic mission designed to return extraterrestrial material from outside cis-lunar space. In addition to comet material, Stardust will also bring back samples of interstellar dust. Stardust will capture these particles by using a substance called "aerogel".

The Stardust spacecraft was launched on February 7, 1999, and is expected to encounter the comet in January 2004. Not until January 2006, will Stardust return its samples to Earth by parachuting a reentry capsule (the Sample Return Capsule, SRC). [31] Weighing approximately 125 pounds, the SRC will land within the Utah Test and Training Range (UTTR). [15]

To determine the footprint, over 3200 trajectories were simulated with 41 mission uncertainties. [25] The size of the resulting footprint was 83.5 km in down-range by 29.2 km in cross-range (which is well within UTTR boundaries). More importantly however, Monte Carlo analyses of the entry also showed that capsule attitude excursions near peak heating and drogue chute deployment were within specified limits. [25]

The program management was extremely interested in these attitude excursions because of aerodynamic instabilities found in the Stardust SRC. These instabilities, which were revealed late in the design process, could have necessitated very costly corrective measures. If identified earlier, however, these instabilities could have been eliminated by considering alternative capsule configurations. This demonstrates why Monte Carlo analyses should be included early in the conceptual design phase. [25]

Desai concluded that Stardust, due to the resolution of the SRC instabilities, relied more heavily on Monte Carlo analysis than any previous mission. He also noted that this increased dependence on Monte Carlo results for mission success places considerable importance on the selection of appropriate uncertainties. [25]

3.4 Mars Surveyor (2001)

NASA's Mars Surveyor Program (MSP) is a series of missions designed to send two spacecraft to Mars every opportunity (one orbiter and one lander – launched separately). The first opportunity for the program was in 1997 when the highly successful Mars Global Surveyor (MGS) and Pathfinder missions were launched. The next opportunity was not so fortunate for the program. In 1998, MSP launched the ill-fated Mars Polar Lander (MPL) and Mars Climate Orbiter (MCO) missions. Both missions failed.

The Mars Surveyor 2001 Project consisted initially of two missions, the Mars Surveyor 2001 Orbiter and the Mars Surveyor 2001 Lander. The Orbiter was to nominally orbit Mars for three years, with the objective of conducting a detailed mineralogical analysis of the planet's surface and measuring the radiation environment on orbit. The Lander was to study soil and atmospheric chemistry and radiation at the surface. The Orbiter was also to act as a communications relay for the Lander. Following the failures of the 1998 missions, the 2001 mission was rescoped and the lander mission was cancelled. The orbiter, now called 2001 Mars Odyssey, is currently en route to the red planet.

An Atmospheric Flight Team (AFT) was formed by the MSP '01 mission office in 1997 to develop: (1) aerocapture and precision landing strategies, (2) atmospheric flight simulation test-beds, and (3) a broad set of potential atmospheric guidance algorithms. Both 3- and 6-DOF Monte Carlo simulations were developed for testing and evaluating candidate guidance algorithms for the 2001 Orbiter and Lander. [19, 32] Seven candidate guidance algorithms were submitted for the lander, six for the orbiter. [33, 34, 35, 36, 37]

3.4.1 <u>2001 Mars Odyssey</u>

Launched April 7, 2001, Mars Odyssey will arrive at Mars in October 2001. Once there, the spacecraft will begin using aerobraking to shape its orbit. Aerobraking is a fuel-saving technique that involves dipping into the Martian atmosphere many times to slow down and lower the altitude of the apoapsis above the planet.

Odyssey's primary science mission is to map the distribution of chemical elements and minerals that make up the Martian surface. The spacecraft will especially look for hydrogen, most likely in the form of water ice. It will also record the radiation environment in low Mars orbit to determine the radiation-related risk to any future human explorers who may one day go to Mars. Odyssey is also expected to support future missions in the Mars Exploration program by providing a communications relay. [11]

The forecast variables for the orbiter simulation consisted of aerobrake-exit orbital elements: apoapse, periapse, inclination, and longitude of ascending node. These quantities were calculated using both the actual and navigation states. Navigation states (the position and velocity used by the guidance computer) are propagated the same as the actual states, except that they include IMU measurement errors. These separate states are necessary to evaluate the performance of the guidance algorithm in the presence of imperfect navigational knowledge. [19] Powell [20] presents the evaluation of a numerical roll reversal predictor-corrector guidance algorithm.

3.4.2 <u>MSP '01 Lander</u>

The MSP 2001 Lander was designed to deliver a small, advanced technology rover to Mars. However, this mission has been cancelled due to the restructuring of NASA's Mars Exploration Program in the wake of the Mars Climate Orbiter and Mars Polar Lander failures. The rover was to be more capable, and able to travel farther, than

Pathfinder’s Sojourner rover. The Lander was planned to perform key experiments to assess the radiation environment on the surface. The results of such experiments could influence decisions for human missions to Mars. Walberg [38] reported that exposure to radiation may be one of the key discriminators between the many different Mars mission scenarios.

Mars Surveyor 2001 would have been the first lander on another planet to use a guidance algorithm to actively control its entry. [39] This ability to make a precision landing allows the spacecraft to land safely within a 10-km circle (as compared to the 100-by-200 km landing zone for Mars Pathfinder). Precision landing capability is significant because it allows placing the spacecraft (and its science instruments) closer to interesting features while reducing landing risk. It is certainly easier to find a 10-km area without hazardous features (craters or large rocks) than it is to find a 100-by-200 km area with the same characteristics.

The Entry, Descent, and Landing (EDL) sequence for this mission was designed as follows. The MSP ’01 Lander enters Mars directly from its interplanetary transfer orbit (i.e., it does not insert first into a Martian orbit). Five minutes before atmospheric entry, the cruise stage is jettisoned. The guidance algorithm deploys a super-sonic parachute approximately 226 seconds after entry (and at an altitude of 9- to 10-km). Ten seconds after parachute deployment the aeroshell is released. At an altitude of 1.43 km (measured by radar altimeter), the lander legs deploy and the powered descent begins. The parachute and backshell are released two seconds after descent engine ignition. The Lander touches down about 37 seconds later at a soft 2.5 m/s velocity. [40]

Striepe et al. [19] and Powell [20] give results of Monte Carlo analyses performed for the MSP ’01 Lander. These simulations included models of the gravity, planet, atmosphere, aerodynamic data, control system, inertial measurement unit, and spacecraft mass properties. The precision landing objective for these studies was to deploy a parachute within 10 km of the target. The MSP '01 Lander requirements were to meet the parachute deployment conditions (Mach number and dynamic pressure between established limits) while arriving within 10 km of the target point.

3.4.3 Terminal Descent Study

If even greater landing precision is needed, an actively controlled and guided terminal phase with significant lateral maneuvering capability may be employed. Walberg and Birge [41] studied two terminal descent scenarios: (1) a ballistic parachute and powered gravity turn to hover conditions, followed by a powered lateral translation to the target landing site and (2) a guided, lifting parachute descent, followed by a powered gravity turn to the target landing site.

Walberg and Birge [41] showed that both terminal descent approaches could achieve landings within 10 meters of the lading site. However, the guided parachute option was expected to have better performance (due to fuel savings). This research demonstrates that when the footprint, in this case at parachute deployment, is reduced sufficiently, other technologies (such as a guided parachute) may be used to reduce the landing footprint even further.

3.5 Genesis (2001)

NASA's Genesis mission, scheduled to launch in mid-2001, will orbit about the Sun-Earth L1 libration point for two years collecting solar wind particles. Genesis is the fifth Discovery-class mission and will be the first to return samples from trans-lunar space. The Genesis Sample Return Capsule (SRC) Earth return trajectory was analyzed using Monte Carlo techniques [15, 16].

The Genesis mission demonstrates how mission rules dictate landing footprint requirements. Though the nominal EDL sequence baselines the mid-air capture of the SRC by a helicopter, range safety rules require the SRC to land within the Utah Test and Training Range (UTTR) in the event of an air-snatch failure.

In addition to the landing footprint, the SRC attitude near peak heating and parachute deployment were of interest. Desai and Cheatwood [16] used 3000 random, off-nominal POST trajectories in the Monte Carlo analysis, which included 47 uncertainties. Though not used for screening purposes, a OVAAT sensitivity analysis was performed. This sensitivity study found that the initial state vector, atmospheric density, and wind uncertainties produced the greatest dispersions in the downrange axis of the landing ellipse. The 3-sigma landing footprint obtained was 47.8-by-15.2 km in down-range and cross-range, respectively.

3.6 MUSES-C (2002)

MUSES-C is a Japanese Institute for Space and Astronautical Sciences (ISAS) mission to a near-Earth asteroid. [42] NASA is providing technical assistance and a small nano-rover. The solar-electric spacecraft will rendezvous with the asteroid, deploy the nano-rover, and collect samples for return to Earth. The mission will launch in December 2002. MUSES-C will arrive at the target asteroid, 1998 SF36, in September 2005. [43]

Desai et al. [15] give an overview of the MUSES-C mission and the Sample Return Capsule (SRC) flight path. The spacecraft is spin stabilized and relies solely on aerodynamic stability for attitude control during the entry. This is possible because the center-of-gravity is located sufficiently far forward (in contrast to the Stardust and Genesis capsules). The landing footprint for this mission is roughly 65-by-20 km and will be somewhere in the Southern Hemisphere.

This mission demonstrates how Monte Carlo analyses can influence the design of a vehicle. By studying the aerodynamic stability in all the flight regimes (hypersonic-rarefied, hypersonic-transitional, hypersonic-continuum, supersonic, transonic, and subsonic) and observing the angle-of-attack excursions during the flight-path, engineers can make important decisions concerning attitude control systems, spin rates, and drogue parachutes.

3.7 Mars Ascent Vehicle (2005)

Desai et al. [44] describe the flight analysis of the Mars Ascent Vehicle (MAV). This vehicle was planned to be part the Mars Sample Return mission – originally planned as the Mars Surveyor Project 2005 mission. However, this mission was cancelled and the first sample return mission is not likely before 2011.

The challenging sample return mission requires the development and integration of the following components: Mars lander, Mars rover, MAV, Mars orbiter, and Earth Return Capsule (ERC). The MAV is delivered to the surface, along with the rover, by the lander. After gathering samples, the rover places its cargo in the MAV. The MAV launches the samples into Mars orbit, where they rendezvous with the orbiter. The orbiter transfers the sample container from the MAV to the ERC. Finally, the ERC makes the long return trip back to Earth and enters the atmosphere.

The MAV design described by Desai et al. [44] was a two stage, pressure-fed, liquid rocket. The propellants used were Monomethyl Hydrazine (MMH) and a mixture of Nitrogen Textroxide and Nitrogen Monoxide (MON-25). The system configuration had the second stage placed within the first stage to reduce the volumetric envelope.

To satisfy mission constraints, the MAV needed to be able to insert a 30 kg sample canister (containing 300 g of rock and soil) into a 300-km circular orbit with a 30-degree inclination. The allowable mass for the vehicle was only 426 kg. Because of this tight constraint, the engineers were concerned that the MAV would have to be over-designed to accommodate uncertainties.

Following a Monte Carlo analysis, Desai et al. [44] concluded that off-nominal conditions during the ascent (due to uncertainties in vehicle aerodynamics and atmospheric properties) required vehicle mass increases on the order of 10%. They further recommended that the aerodynamic uncertainties could be reduced by improving the fidelity of the aerodynamic model. This would require expensive wind tunnel tests and additional CFD analyses. This design was later abandoned in favor of a higher bulk density, solid propellant system.

CHAPTER 4

OPTIMIZATION IN DESIGN

All optimization problems may be stated mathematically in the standard form of Equation 1. [45] Here, x_i are the design variables, f is the objective function, g_j are inequality constraints, h_k are equality constraints, and x_i^L and x_i^U are lower and upper side constraints respectively. Therefore, design optimization problems may be classified according to how they handle the various elements of the standard form. These elements are: (1) the objective function, (2) the design variables, and (3) the constraint functions. Additional considerations include the inclusion of metamodeling and uncertainty simulation.

$$\begin{aligned} &\text{given : } \bar{x} = \{x_i\} \; i = 1, 2, \ldots, n \\ &\text{minimize : } f(\bar{x}) \\ &\text{subject to : } \bar{g} = \{g_j(\bar{x}) \le 0\} \; j = 1, 2, \ldots, m \\ &\qquad\qquad \bar{\mathrm{h}} = \{h_k(\bar{x}) = 0\} \; k = 1,2,\ldots,\ell \\ &\qquad\qquad x_{\mathrm{i}}^L \le x_i \le x_{\mathrm{i}}^U \quad i = 1,2,\ldots,n \end{aligned} \qquad (1)$$

This chapter describes the state-of-the-art in design optimization methodologies and discusses the following topics: multi-objective function optimization, design variable screening, unconstrained optimization, metamodeling, and probabilistic methods. The purpose of this chapter is to provide background into optimization methodologies and discuss some of the available techniques.

4.1 Multi-objective vs. Single-objective

All optimizers require a single means of determining the “goodness” of a design or ranking a set of alternatives produced in the optimization process. Because designers are often faced with multiple and competing objectives, they must use methods to “collapse” the multiple objectives into a single scalar function. Several Multi-Criteria Decision Making (MCDM) methods exist in the literature. [46, 47, 48, 49, 50, 51, 52, 53, 54]

Most multi-criteria methods can be classified in one of two groups: those that use a weighted summation of the individual criteria (an Overall Evaluation Criteria, OEC) and those that use a lexicographic (or ordering) approach to ranking criteria and alternatives. A third method [55] combines multi-objective optimization with probabilistic methods by using a joint probability function. The objective for this method is not a summation of criteria, but rather the probability of satisfying all of the criteria simultaneously. [46]

By far, the most simple and common method is the use of an OEC. However, this method suffers from the disadvantage that it requires the subjective formulation of the criteria weights. Ignizio [56, 57] defines the lexicographic minimum as follows. The user first ranks each criteria according to its importance. Designs are first compared based on the highest priority criteria (or goal). If they are equal in the first objective, then they are compared in the second objective – and so on. This method may be particularly suited for Genetic Algorithms (GAs) using tournament selection.

In this research, the objective function is arbitrary (the emphasis being on meeting the constraint) so that multiple criteria are not needed. The form of the proposed objective function is a weighted cost-tolerance function, where the weights are determined from the relative expense of reducing the associated uncertainty.

4.2 Analysis vs. Metamodel

Detailed engineering analysis codes are often very expensive to run, a shortcoming that is only aggravated by optimization procedures that require many function evaluations. In these cases, a “model of the model” or metamodel [58] is often used. Metamodels [59, 60] approximate the actual codes and are orders of magnitude cheaper to run, allowing them to be used efficiently in conjunction with optimization routines. Once created, metamodels can be used repeatedly, avoiding the need to re-execute the expensive analysis code. However, the cost of increased speed is accompanied by a loss of accuracy (which as Su and Renaud [61] point out, is especially true for derivative information). Additionally, metamodels are often limited in the number of variables that may be modeled. [60] Simpson et al. [59] give a good overview of the many metamodeling methods that exist in the literature. These methods include, but are not limited to: response surface equations (RSEs), neural networks, and kriging.

The RSE is the most common, well established, and easiest to use metamodeling technique. [59, 60, 62, 63, 64, 65, 66, 67] Myers and Montgomery [68] explain that the term "response surface model" was taken from the statistical literature. Generally, this term refers to any polynomial-based modeling method. Most often, the underlying model is a second-degree (quadratic) polynomial approximation in the form given by Equation 2. This form accounts for individual parameter effects (linear terms, b_i), second-order curvature (square terms, b_{ii}), and two-variable interactions (cross terms, b_{ij}). Giunta and Watson [63] credit the popularity of these methods to a number of reasons, one of which is that they, "provide a compact and explicit functional relationship between the response and the independent variables."

$$\tilde{y} = b_0 + \sum_{i=1}^{n} b_i x_i + \sum_{i=1}^{n} b_{ii} x_i^2 + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} b_{ij} x_i x_j \qquad (2)$$

Many authors have warned of the dangers of applying response surfaces to deterministic computer codes. Because response surfaces were originally developed to model physical experiments, the statistical determination of the polynomial coefficients (least squares regression) assumes that the response data is contaminated with a normally distributed random error, which has some variance and a mean of zero. [63] However, response surface techniques have been widely applied to deterministic (repeatable) computer experiments where there is no random error. Members of the statistical community have, therefore, questioned both the applicability of the response surface, and the standard statistical tests for model and parameter significance, when used with deterministic data. [59, 69, 70, 71]

Due to the random number generation when employing Monte Carlo analyses, regression of the data <u>is</u> appropriate since random error is present. This also leads to results that are not repeatable. This research proposes a simplified second-order polynomial model that retains only the constant and square terms (i.e., no linear terms or cross terms are present). It is important to note that this metamodel is being used to approximate the expensive Monte Carlo analysis and not individual trajectory simulations (POST runs).

4.3 Many Variables vs. Screening

In constructing an RSE, the first step is normally to conduct a screening test. [55] A screening test is done when the number of design variables is too large for construction of a three-level Design of Experiments (DOE) or when the function evaluations are prohibitively expensive. The underlying assumption of the screening test is the "sparsity of effects" principle [72], which assumes the system response is dominated by main effects and low-order interactions. [59] Following this principle, a two-level fractional factorial is used that accounts for main effects only, but which examines many variables with minimal expense. [67] One specific family of designs commonly used for this purpose is the two level Plackett-Burman (P-B) designs. The scaled estimates from this design and their cumulative total are typically displayed in a Pareto plot. This plot allows the designer to quickly visualize the relative influence of each variable on the response. [73] Design variables with the largest estimates are retained.

Another form of screening often used with Monte Carlo simulations is the One-Variable-At-A-Time (OVAAT) analysis. With all other uncertainties set at their nominal values, each uncertainty is varied independently to its maximum and minimum. Changes in the forecast variables from the nominal trajectory (deltas) are recorded for each uncertainty and displayed in a Pareto Plot. The OVAAT method of screening is used in the proof-of-concept example.

A particular issue in response surface generation is the problem of scalability. Simpson et al. [59] note that as the size of a problem increases, the cost of creating a metamodel quickly begins to outweigh the costs of using the analysis code directly. A central composite DOE, for example, requires 2^n+2n+1 function evaluations to evaluate the ½(n^2-n)+2n+1 RSE coefficients. At 30 design variables, a Central Composite Design (CCD) would require over 1 billion (1,073,741,885) function evaluations to evaluate nearly five hundred (496) RSE coefficients! Simpson et al. [59] further note that, "often the reduction in factors resulting from screening is still not sufficient to bring the problem to a manageable size."

This problem of scalability is particularly relevant to this thesis. Monte Carlo simulations typically include 30 to 50 uncertainties, yet the practical limit for generating a response surface is about eight or ten design variables. While a feasible set of 8-10 extrema may be found that minimize their cost, no information is generated for the remaining 20-42 uncertainties, nor can any claim be made that the solution represents the lowest cost alternative. For the methodology proposed by this research to be useful, it must be scalable to large numbers of design variables without screening.

4.4 Constrained vs. Unconstrained Optimization

Whether called "constraints", "requirements", or "rigid goals", constraints are a natural part of engineering problems and they originate from many sources including the following: outside groups, management, government regulations, customers, thresholds on performance, and even nature. [47, 62] Designs that fail to meet constraints are called infeasible. In mathematical programming (classical optimization), constrained optimization is performed either directly or indirectly. Indirect methods involve penalty functions and sequential unconstrained minimization techniques (SUMT). Direct methods include sequential linear programming (SLP), method of feasible directions, and sequential quadratic programming (SQP). [45] In goal programming, constraints are treated as goals with the highest level of priority (rigid goals). [47, 56, 57, 74]

In this research, the constraint is central to the problem formulation, and should be satisfied as precisely as possible. Because indirect methods (using penalty functions) can only approach the constraint sequentially, a direct method is desired that gives more precise control over the constraints. Because both the objective function and the metamodel of the constraint are analytically differentiable, the method chosen is the method of Lagrange multipliers. The unconstrained Lagrange problem is then solved very quickly, efficiently, and precisely using a Newton-Raphson iteration.

In the event that it becomes impractical to generate a metamodel (due to very large numbers of design variables) an alternate method is needed. In this case, a zero-order optimization method, such as GA, could be used. Most applications of GA use a penalty function method in handling constraints and suffer from problems with premature convergence. However, a niched-penalty approach, proposed by Deb and Agrawal [75], avoids the use of penalty functions by exploiting the pair-wise comparison used in tournament selection. In this method, two feasible solutions are always compared by their objective function value. When a feasible solution is compared to an infeasible one, the feasible solution is always chosen. Two infeasible solutions are compared based on the extent of their constraint violation. This method was not pursued in this research, but remains another possible solution method.

4.5 Probabilistic vs. Deterministic

While there is growing emphasis on risk and uncertainty management in design, engineering analysis codes are exclusively deterministic in nature. A "wrapper" is needed to evaluate the effects of uncertainty on these codes. Internet.com [76] defines a wrapper as, "software that accompanies resources or other software for the purposes of improving convenience, compatibility, or security." This wrapper interfaces many times with the underlying analysis code for the purpose of collecting statistics on the deterministic output. Generally, a histogram is used to produce a visual representation of the probability distribution function (pdf). Alternately, an approximation of the cumulative distribution function (cdf) may be generated. In that case, the pdf is determined by differentiation. While histograms provide valuable information to the designer, they are not compatible with optimization. Selected scalar values (e.g., parameters such as the sample mean and variance or discrete function evaluations from either the pdf or cdf) must be returned to the optimizer.

The most common form of uncertainty wrapper, and the most computationally intensive, is the Monte Carlo Simulation (MCS). This technique utilizes a random (or pseudo-random) number generator to randomly sample from given uncertainty distributions and then deterministically evaluate the analysis routine many times. Large sample sizes, typically 1000's of cases, allow very accurate modeling of forecast distribution functions. The primary advantage of MCS is that the input distributions are sampled in parallel, making the process independent of the number of input distributions. Additionally, there are no restrictions on the types of uncertainties or whether or not there are correlations between them. Spencer and Braun [13] note that, "while computationally intensive, the Monte Carlo approach can give insight into the behavior of systems that are too complex to be resolved analytically."

Other methods of uncertainty analysis are known as fast probability integration methods. Fast probability integration methods include the Most Probable Point (MPP) analysis, the Mean Value method, and the Advanced Mean Value (AMV) method, which combines a simple Mean Value method with the MPP analysis. These and other fast probability methods are included in the commercial software package Fast Probability Integration (FPI) [77], which was developed at the Southwest Research Institute (SwRI). Fast probability methods approximate the desired uncertainty results, with a reduced number of function calls. These methods sample specific points in the design space, which have a known probability of occurrence according to a joint probability distribution function. Since each point in the design space generates a certain response, the sampled response values occur with the same probability as their corresponding uncertainties. [55]

DeLaurentis and Mavris [60] define three general methods of uncertainty analysis: (1) applying MCS directly to the analysis code, (2) applying MCS to a metamodel of the analysis code, and (3) applying fast probability methods directly to the analysis code. So, when the designer is confronted with too large a computational expense to utilize the first method, a choice must be made to either precisely analyze the statistics of an approximation to the code or approximately analyze the statistics on the precise code. In all three methods, the forecast variables of the uncertainty analysis are then modeled with a response surface equation – regardless of whether a metamodel of the analysis code was used to generate the statistics.

For this research, MCS is used directly on the trajectory simulation (method #1) to produce an estimate of the size of the landing footprint. This response is then used to generate one or more response surfaces. However, if the problem becomes large enough (large number of design variables) and computational time is limited, a fast probability method may be required to reduce computational expense. In the event that a fast probability method is used, a metamodel of the uncertainty analysis would not be recommended since this would provide an approximation of an approximation.

CHAPTER 5

RANDOM NUMBER GENERATORS

A Monte Carlo method has been generally defined by Knuth [78] as "any algorithm which employs random numbers." Since random numbers are so central to the concept of Monte Carlo methods, a brief discussion on the subject of "randomness" and "random number generators" is in order. What is a random number? How are random numbers generated on a computer? How can one evaluate the randomness of a sequence of pseudorandom numbers? A much more detailed discussion on the topic can be found in Knuth [78]. Anderson [79] also provides an excellent survey on techniques for generating and testing random numbers.

5.1 Random Numbers

The first question to address is, "what constitutes a random number?" Intrinsically, everyone has a concept of what a random number is. Obviously, one can not look at a single number (5 for example) and determine if it was generated in a random process. Therefore, any definition of randomness must be restricted to a series of numbers. Most will agree that in a random sequence of numbers, the next number in the sequence has nothing to do with other numbers of the sequence, but rather is determined by a random process (i.e., each number is obtained merely by chance). We allow, however, that the next number in the sequence has a specified probability of taking on a given value.

Anderson [79] warns, however, against assuming that random numbers are a well-defined concept. Perhaps the best definition is given by Lehmer [80]. He writes that "a random sequence is a vague notion embodying the idea of a sequence in which each term is unpredictable to the uninitiated and whose digits pass a certain number of tests, traditional with statisticians and depending somewhat on the uses to which the sequence is to be put."

This definition emphasizes the fact that the field of statistics is the only tool available for evaluating randomness and alludes to the practice of performing multiple tests on random sequences. Yet, statistics can only assist – it can not make the determination of whether or not a sequence is random. Knuth [78] writes that, "The mathematical theory of probability and statistics carefully avoids answering the question; it refrains from making absolute statements, and instead expresses everything in terms of how much probability is to be attached to statements involving random sequences of independent events."

5.2 <u>Random Number Generators</u>

"Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin" – John Von Neumann (1951)

"It may seem perverse to use a computer, that most precise and deterministic of all machines conceived by the human mind, to produce 'random' numbers. More than perverse, it may seem to be a conceptual impossibility. Any program, after all, will produce output that is entirely predictable, hence not truly random." – Press et al. [81], Numerical Recipes, (1989)

Leaving the question, "what is random?" unanswered, the question now becomes, "how are random numbers generated on a computer?" Random number generators have existed for over fifty years. Von Neumann [82] first suggested a "middle-square" method for generating random numbers in 1946. The problem with random number generators, of course, is that they are not truly random. The two quotes at the beginning of this section attest to this fact. Knuth [78] raises the objection, "how can a sequence generated in such a way be random, since each number is completely determined by its predecessor?" His answer to this conundrum is that, "the sequence isn't random, but it appears to be."

Because of this, computer algorithms for the production of random numbers are said to generate “pseudo-random” or “quasi-random” sequences. The word “random” is often used for the output of truly random physical process, like the elapsed time between clicks of a Geiger counter. Because only sequences generated by computers are discussed here, (and because many of these methods are very good at simulating random behavior) no distinction is made in this research.

Anderson [79] lists five common techniques for generating random numbers on a computer:

1. Linear, Congruential Generators
2. Shift Register Generators
3. Lagged-Fibonacci Generators
4. Randomizing by Shuffling
5. Combination Generators

Of these five methods, only the first (linear, congruential generators) will be discussed here. Discussions on the other types of generators can be found in Anderson [79] and Knuth [78]. All these methods produce uniform deviates (random numbers that are uniformly distributed between 0 and 1). A reliable source of uniform deviates is essential for any Monte Carlo process. This is because other sorts of deviates (e.g., random numbers that have a Gaussian distribution) are usually generated by performing appropriate operations on one or more uniform deviates.

5.3 Linear, Congruential Generators

The Linear, Congruential Generator (LCG) was first proposed in 1948 by Lehmer. LCGs are defined by use the recurrence relationship, Equation 3, to create a series of integers. Dividing these integers by the largest possible integer, *m*, creates a real number between 0 and 1. The appearance of randomness comes from the LCGs use the modulus (remainder) operation.

$$I_{j+1} = \text{mod}(aI_j + c, m) \qquad \textbf{(3)}$$

Because each integer depends on the previous integer, LCGs require an initial "seed" value to start the sequence. This seed must be provided by some external means – often the current system time. Using the same seed on successive runs of the random number generator will result in the same random sequence.

Because the LCG sequence is completely defined by the constants *a* (multiplier), *c* (increment), *m* (modulus), and x_0 (seed), the designation LCG(a, c, m, x_0) is used to identify the sequence. These algorithms are very common in the literature, and there exists an extensive body of knowledge on their behavior (See [78]). Much of this literature is on how to carefully choose the constants so that the LCG has favorable characteristics, such as a long period (the number of integers returned before the inevitable repetition of the sequence). Table 4 is an excerpt from a table in Numerical Recipes [81] that shows "good" choices for LCG constants. These constants have been carefully chosen to maximize the period of the random number generator. Rules for these choices are given in Knuth [78].

Table 4: Constants for Linear, Congruential Generators. [81]

Multiplier (*a*)	Increment (*c*)	Modulus (*m*)
1861	49297	233280
2661	36979	175000
4081	25673	121500
3661	30809	145800
3877	29573	139968
3613	45289	214326
1366	150889	714025
8121	28411	134456
4561	51349	243000
7141	54773	259200
9301	49297	233280
4096	150889	714025

LCGs are very fast (because of their small operations count), but suffer from several problems. Most notably, LCGs are known to show *k*-tuple correlation (i.e., every *k*th number generated is correlated). Points plotted in k-space show that the numbers are restricted to a number of planes. Additionally, LCGs show less randomness in their least significant bits than in their highest significant bits. Because of these weaknesses, LCGs make poor candidates for use in highly dimensional Monte Carlo processes. They are discussed here, however, because they form the basis for the random number generator used in this research, "ran1".

5.4 The "Ran1" Random Number Generator

The random number generator used in this research is "ran1". This function is a combination generator that uses three LCGs. These three LCGs create a random number even more random than the original random numbers. This routine (originally written in Fortran and translated into Perl) was taken from Numerical Recipes [81] and the LCG constants were selected from Table 4. The three LCGs are:

$$\mathrm{LCG}_1 = \mathrm{LCG}(7141, 54773, 259200, x_0)$$

$$\mathrm{LCG}_2 = \mathrm{LCG}(8121, 28411, 134456, x_0)$$

$$\mathrm{LCG}_3 = \mathrm{LCG}(4561, 51349, 243000, x_0)$$

The first two LCGs are combined so that the first forms the most significant part of the output deviate and the second forms the least significant part. The third LCG is used to further randomize the output by shuffling an array of 97 random numbers.

On the first call to "ran1", LCG_1 is seeded. It is then used to seed the other two generators and the shuffling array is populated. The third LCG produces an integer between 1 and 97, which is used to index the pre-filled array. The selected array element is output and then replaced with a new deviate.

According to Knuth [78], this method of shuffling (called two-sequence shuffling) was first introduced by MacLaren and Marsaglia in 1965. [79] Its purpose is to break up the *k*-tuple correlations that might exist. Combined, the period of "ran1" is practically infinite and it ought to have no sequential correlations. However, testing of “ran1” was required to determine its suitability for this research.

5.5 Tests for Random Number Generators

A “good” random number generator (of uniform deviates) should posses several qualities: serially uncorrelated sequences, long period, and uniformity. A serially uncorrelated sequence means that any subsequence of the random numbers (particularly *k*-tuples) should not be correlated with any other subsequence. The period of a random number generator is the number of deviates produced before the sequence (inevitably) repeats. Ideally, the generator should not repeat at all. Practically, the repetition should occur only after a very large number of deviates (at least larger than the number of deviates required for the Monte Carlo process). Uniformity means that the uniform deviates should be (understandably) uniform. This means that equal fractions of the deviates should fall into equal sub-intervals. This presumes, of course, that the sample size is large enough to count as a representative sample.

Many statistical tests have been developed to test for these properties. In general, all of these tests take a sample of deviates and construct a particular statistic. Knowing the theoretical distribution of the statistic, they compute the probability of exceeding that particular value of the statistic. Very high (or low) probabilities are rare and suggest that the sequence might not be random. Probabilities that are neither very high nor very low support (but do not prove) the hypothesis that the sequence is random. Some simple tests discussed by Knuth are: the equidistribution test, the serial test, the gap test, the poker test, the coupon collector’s test, the maximum-of-t test, and the serial correlation test. [78]

To test the randomness of the “ran1” function, it was compared with two other random number generators, the Perl built-in random number generator (named “rand”) and the Matlab built-in random number generator (also named “rand”). A script was used to generate half a million (500,000) random numbers from each of the three random number generators and the results were written to a file. The random deviates in this file were then compared in the following seven tests.

5.5.1 Histogram Test

The simplest test to run is a visual test, rather than a statistical one. Simply plot a histogram of the random sequences. This plot should show complete and uniform coverage of the interval from 0 to 1.

Figure 6 shows a 100-bin histogram of the 500,000 random numbers generated with the "ran1" function. Note that there is little variation in the number of cases in each bin. This produces an extremely uniform (flat) distribution, which is a desired characteristic. However, this distribution may be too uniform to be considered random. Further testing was required.

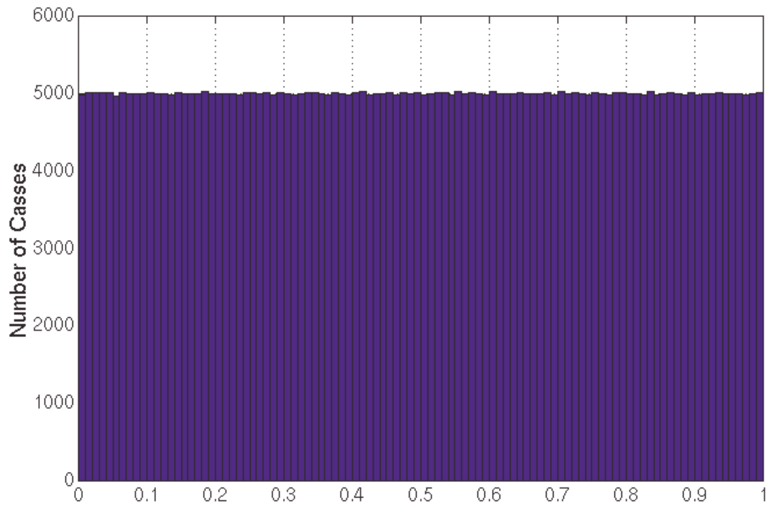


Figure 6: "Ran1" Histogram.

Figure 7 shows a 100-bin histogram of the 500,000 random numbers generated with the Perl built-in "rand" function. Compare this plot with Figure 6. Note that there is more variation in the number of cases in each bin. However, the overall uniformity is good. Results for the Matlab "rand" function were similar.

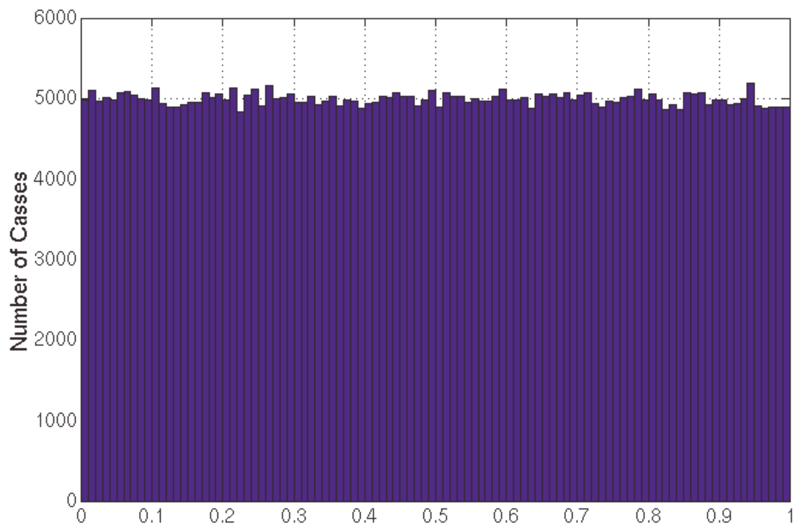


Figure 7: Perl "Rand" Histogram.

5.5.2 <u>Repetition Test</u>

The next test, the repetition test, is a quick and simple way to test for a short period – simply search for matching numbers from within the sequence. Finding a matching number, however, does not guarantee that the sequence repeats, but the converse is true. Not finding a matching number does guarantee that the period is at least as long as the sample.

No matches were found from either the "ran1" or the Matlab "rand" sequences (i.e., each number in these sequences was unique). The Perl "rand" function, however, was found to repeat more than 15 times on average. This suggests that this function may have a short period.

Further investigation showed that the reason for the repetition is that this random number generator produces only 32,768 (2^{15}) unique outputs. If a finer resolution than 2^{15} were required, this random number generator would not be appropriate. However, the sequence of these numbers did not repeat within the sample of 500,000. This was determined by noting that the second number was never found to follow the first number anywhere else in the sequence. This suggests that the Perl random number generator uses probably a shuffling algorithm.

5.5.3 2-D Scatter Plot Test

This test, the 2-D scatter plot, is designed to detect relationships between successive random numbers (i.e., correlations in two dimensional space). The random sequences are used to generate (x, y) pairs, which are plotted. Random number generators that fail this test will show lattice-like patterns (LCGs are particularly bad at this test). These patterns develop when the pairs fall into discrete planes.

Figure 8 shows a plot of adjacent pairs generated with the "ran1" function. This scatter plot shows, as it should, a very uniform coverage of the area. More importantly, there are no discernible patterns (or lattice networks), which would indicate a correlation between adjacent numbers.

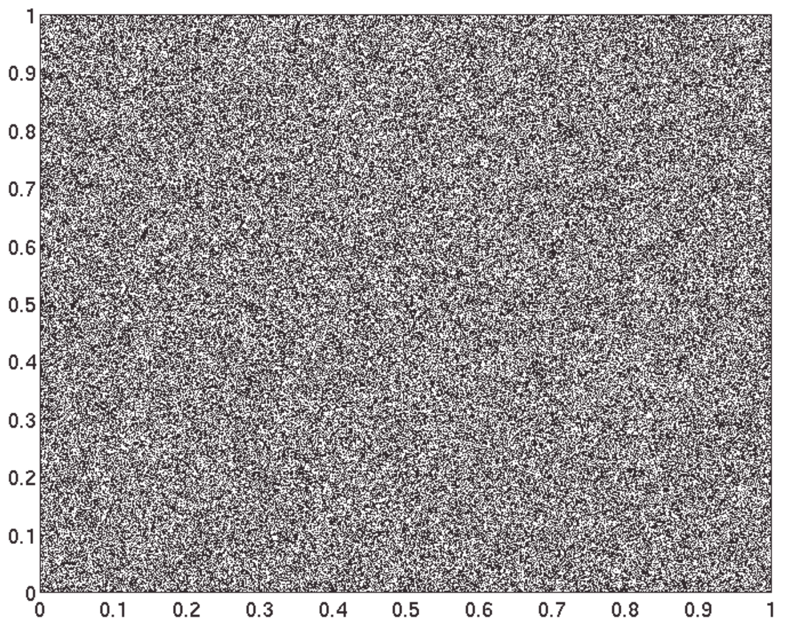


Figure 8: "Ran1" Scatter Plot.

Figure 9 shows a plot of random numbers generated with the Perl built-in "rand" function. Compare this scatter plot with Figure 8. This plot also shows a very uniform coverage of the area and no discernible patterns. Similar results were found for the Matlab "rand" function.

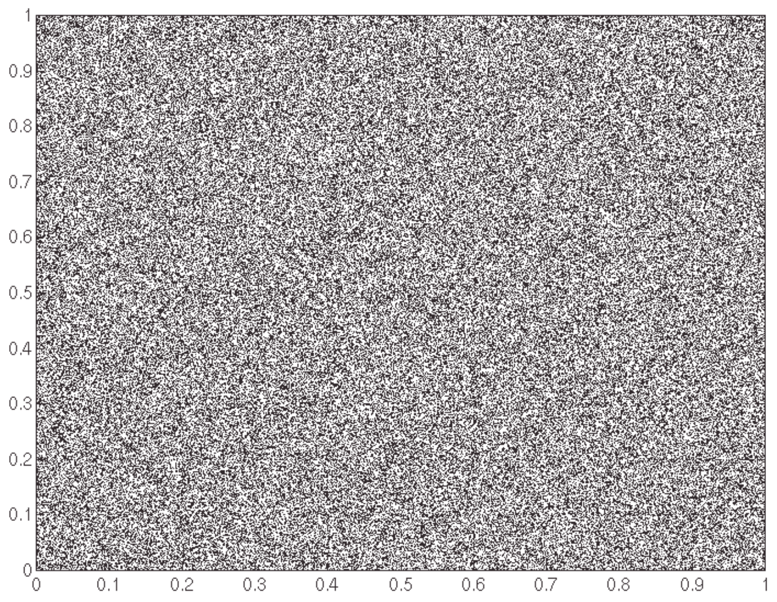


Figure 9: Perl "Rand" Scatter Plot.

All of the random number generators passed this test, indicating that no two dimensional correlations exist. However, there is no guarantee that correlations do not exist in higher dimensions. More complicated test (such as Knuth's spectral test [78]) are needed to evaluate the higher dimensions.

5.5.4 Poker Test

The classic poker test (also known as the partition test) is a statistical test that uses numbers from a random sequence to simulate five-card poker hands. The numbers of specific hands produced (e.g., all different, one pair, two pair, three of a kind, full house, four of a kind, and five of a kind) are compared to the card odds using a chi-square test. [79]

The chi-square is a well known statistical test that applies to situations when observations can fall into a finite number of categories, *k*. This test compares the actual number of each discrete outcome with the theoretical expected number. To perform this test, take a large number, *n*, of independent observations. If p_s is the probability that an observation falls into category *s*, then np_s is the expected number of observations for that category. Count the actual number of observations, *Ys*, which actually do fall into category *s* and form the statistic, *V*, in Equation 4.

$$V = \sum_{s=1}^{k} \frac{\left(Y_s - np_s\right)^2}{np_s} \qquad \textbf{(4)}$$

For large numbers of observations, the distribution of this statistic is approximately chi-square with the number of degrees-of-freedom equal to one less than the number of categories (*k*-1). Finally, determine the probability that a random number from a chi-square distribution would be greater than *V*.

Knuth [78] suggests the following guidelines for interpreting the results. If *V* is less than the 99-percentile value or greater than the 1-percentile value, reject the sequence as "not sufficiently random." If *V* lies between the 99- and 95-percentile or between the 5- and 1-percentile values, the sequence is considered "suspect". If *V* lies between the 95- and 90-percentile, or 10- and 5-percentile, the sequence is considered "almost suspect". These guidelines may apply to any of the statistical tests.

For this research, we add a slight twist to the classic poker test. Instead of poker hands, six-ball lottery picks were generated. The lottery game simulated was the seven-state "big game" lottery. In this game, five numbered white balls are drawn from a pool of 50. A sixth numbered ball, the "money ball", is drawn from a separate pool of 36 yellow balls. The probability of matching *k* of the five white balls, p_k, is given by Equation 5. The variable *q* in Equation 5 accounts for whether or not the money ball is also matched, where *q*=1 if the money ball is matched and *q*=35 otherwise. The binomial coefficients (also known as permutations), given by Equation 6, express the number of ways *k* objects can be selected from *n* objects, without replacement.

$$p_k = \frac{\binom{5}{k}\binom{45}{5-k}}{\binom{50}{5}} \frac{q}{36} \quad q = \begin{cases} 1 & \text{money ball matches} \\ 35 & \text{does not match} \end{cases} \tag{5}$$

$$\binom{n}{k} = \frac{n!}{k!(k-1)!} \tag{6}$$

Table 5 lists all possible outcomes for the "big game" lottery along with published odds of winning and prizes. The jackpot varies from drawing to drawing and is the estimated annuitized earnings for a single winner (it was worth a particularly large $90 million on May 4, 2001). [83] These odds from Table 5 are used to determine the expected values for the chi-square test. The "winning" numbers were arbitrarily taken from the May 4, 2001, drawing. The white ball numbers were: 11, 24, 27, 35, and 47. The "big money ball" was 22.

Table 5: The Big Game Lottery.

Match	Odds	Prize
0,1, or 2 White Balls	30:31	$0
0 White Balls & "The Big Money Ball"	1:62	$1
1 White Ball & "The Big Money Ball"	1:102	$2
2 White Balls & "The Big Money Ball"	1:538	$5
3 White Balls	1:220	$5
3 White Balls & "The Big Money Ball"	1:7705	$100
4 White Balls	1:9686	$150
4 White Balls & "The Big Money Ball"	1:339002	$5,000
5 White Balls	1:2179296	$150,000
5 White Balls & "The Big Money Ball"	1:76275360	Jackpot

Choosing lottery numbers over poker hands to simulate has a hidden advantage. Because the results from the lottery drawings are published, a fourth set of (truly) random numbers may be included for comparison with the other three random number generators – actual winning numbers for 408 lottery drawings.

Because the *V* statistic is only approximately chi-square, Knuth [78] suggests selecting n large enough that the expected values (np_s) for each outcome are larger than or equal to five. Therefore, the first 408 simulated lottery tickets from each random number generator were compared in a chi-square statistic that included only the first three outcomes. Even though the expected value for matching one white ball and the money ball is exactly 4, it is included (violating Knuth's thumb-rule).

The observed results and associated χ^2 probabilities are shown in Table 6. This table shows results of simulated lottery tickets compared against the May 4, 2001 winning numbers. The first 408 tickets are compared along with 408 actual lottery drawings. The results for the actual lottery drawings demonstrate that a truly random process will not show a probability that is either too high or too low. Only the first three outcomes are included in the chi-square statistic.

Table 6: Lottery Test (n=408).

Match	Expected Value	Lottery	Ran1	Perl Rand	Matlab Rand
0,1, or 2 White Balls	394.8	394	397	398	398
0 White Balls & "Big Money Ball"	6.6	6	3	7	4
1 White Ball & "Big Money Ball"	4.0	7	2	2	4
2 White Balls & "Big Money Ball"	0.8	0	2	0	0
3 White Balls	1.9	1	4	1	2
3 White Balls & "Big Money Ball"	0.1	0	0	0	0
4 White Balls	0.0	0	0	0	0
4 White Balls & "Big Money Ball"	0.0	0	0	0	0
5 White Balls	0.0	0	0	0	0
5 White Balls & "Big Money Ball"	0.0	0	0	0	0
χ^2 Statistic (V)	0	2.303	2.9601	1.052	1.0373
Probability (p)	100%	31.62%	22.76%	59.10%	59.53%

A maximum of 83333 lottery tickets can be simulated with 500000 random numbers. Again, because of Knuth's thumb-rule, only the first seven outcomes are compared (those with expected values greater than 5). Only the three random number generators are compared. The observed results and the associated χ^2 probabilities are shown in Table 7. This table shows results of simulated lottery tickets for all 83333 simulations. Again, these tickets are compared against the May 4, 2001 winning numbers. The first seven outcomes are included in the chi-square statistic.

Table 7: Lottery Test (n=83333).

Match	Expected Value	Ran1	Perl Rand	Matlab Rand
0,1, or 2 White Balls	80644.8	80624	80575	80583
0 White Balls & "The Big Money Ball"	1344.1	1365	1358	1364
1 White Ball & "The Big Money Ball"	817.0	774	823	844
2 White Balls & "The Big Money Ball"	154.9	161	156	158
3 White Balls	378.8	387	404	367
3 White Balls & "The Big Money Ball"	10.8	10	10	9
4 White Balls	8.6	11	7	7
4 White Balls & "The Big Money Ball"	0.2	1	0	0
5 White Balls	0.0	0	0	1
5 White Balls & "The Big Money Ball"	0.0	0	0	0
χ^2 Statistic (*V*)	0	3.741	2.2954	2.2682
Probability (*p*)	100%	71.17%	89.06%	89.35%

Following Knuth's guidelines, none of the results from either Table 6 or Table 7 would be classified as "suspect" or "almost suspect". However, the results in Table 7 for the Perl and Matlab random number generators are very close to "almost suspect".

5.5.5 <u>Frequency Test</u>

The frequency test (also known as the equidistribution test) is another statistical test based on the idea that a bad random number generator will not approximate the correct cumulative distribution function sufficiently well. For uniform deviates, this means that the sequence must indeed be uniformly distributed. Empirical distributions created with each sequence of random numbers are compared to the theoretical distribution using a Kolmogorov-Smirnov (K-S) test. [78]

Whereas the chi-square test, used in the poker test, is applicable only for discrete outcomes, the Kolmogorov-Smirnov test applies to continuous random quantities, which may assume infinitely many values. Given n independent observations $[x_1, x_2, \ldots, x_n]$ taken from some continuous distribution function $F(x)$, construct the empirical distribution function, $F_n(x)$. This distribution is constructed by counting the number of observations that are less than x, and then dividing that number by the total number of observations. The statistic used in this test is K_n, given by Equation 7, which measures the maximum difference between the actual and theoretical distributions.

$$K_n = \sqrt{n} \max_{-\infty < x < +\infty} \left| F_n(x) - F(x) \right| \tag{7}$$

The distribution of the K-S statistic, Equation 8, is gives the probability that the value of the statistic is less than some value, which is a function of the number of observations, n. Unlike the chi-square statistic, Equation 8 is exact for all values of n.

$$P\left(K_n \le \frac{t}{\sqrt{n}} \right) = \frac{t}{n^n} \sum_{0 \le k \le t} \binom{n}{k} (k-t)^k (t+n-k)^{n-k-1} \tag{8}$$

Again, the binomial coefficients in this equation are given by Equation 6. Unfortunately, Equation 8 is unwieldy to calculate for large values of n. Fortunately, Knuth [78] explains that this distribution tends toward the approximation given by Equation 9 when n becomes large. The subscript, ∞, indicates an infinite (or at least very large) sample size.

$$F_\infty(x) = 1 - e^{-2x^2} \tag{9}$$

The K-S test was applied to the random sequences produced by each of the random number generators. Since they are uniform deviates, the theoretical distribution, $F(x)$, is just $F(x)=x$. This test was applied first to subsets of the data (n=100000) and then to the entire set (n=500000). Dividing the data into subsets like this can often indicate local non-randomness in addition to global non-randomness. The results of this test are presented in Table 8. The "ran1" statistic (0.0721) indicates a nearly perfect match of the theoretical cumulative distribution function.

Table 8: Kolmogorov-Smirnov Test.

	Ran1		Matlab Rand		Perl Rand	
Range	K_{100000}	p	K_{100000}	p	K_{100000}	p
1:100000	0.2581	87.52%	1.0689	10.18%	1.2662	4.05%
100001:200000	0.3291	80.53%	1.2221	5.04%	1.0789	9.75%
200001:300000	0.3779	75.15%	0.7729	30.28%	0.5781	51.25%
300001:400000	0.2508	88.18%	0.8033	27.51%	0.8078	27.12%
400001:500000	0.2202	90.76%	0.8340	24.88%	1.4079	1.90%
1:500000	0.0721	98.96%	0.8893	20.56%	1.0167	12.65%

The probability associated with "ran1" data (98.96%) is very high, which indicates that the data matches the theoretical distribution extremely well. In-fact, the data may match too well, since a statistic lower than 0.0721 is expected only about 1% of the time. Using Knuth's guidelines, "ran1" would be classified as "suspect". However, only one subset of this data, 400000-500000, would be classified as even "almost suspect". Therefore, "ran1" shows better local randomness than global randomness.

5.5.6 Serial Correlation Test

The serial correlation test is a very quick statistical test to perform. This test looks for correlations between successive random numbers. The correlation coefficient, Equation 10, is calculated for the sequences $U(1{:}n)$ and $U(2{:}n,1)$, where U is the sequence of random uniform deviates.

$$C = \frac{\left(\sum_{i=1}^{n-1} U_i U_{i+1} + U_n U_1\right) - \left(\sum_{i=1}^{n} U_i\right)^2}{\sum_{i=1}^{n} U_i^2 - \left(\sum_{i=1}^{n} U_i\right)^2} \qquad \textbf{(10)}$$

Correlation coefficients, which appear frequently in statistics, range from –1 to +1. A value of zero (or near zero) is desired since it indicates that the sequences are uncorrelated. A value near one indicates that the sequences are linearly correlated. Knuth [78] suggests that the serial correlation coefficient, C, should fall between the values $\mu_n \pm 2\sigma_n$, given by Equation 11 and Equation 12, 95% of the time.

$$\mu_n = \frac{-1}{(n-1)} \qquad \textbf{(11)}$$

$$\sigma_n = \frac{1}{(n-1)}\sqrt{\frac{n(n-3)}{(n+1)}} \qquad \textbf{(12)}$$

The results for the serial correlation test are presented in Table 9. This table shows satisfactory results from the serial correlation test. The expected range, given by

Equation 11 and Equation 12 is from –0.00283 to +0.00283. All of the results are within this range.

Table 9: Serial Correlation Test.

Random Number Generator	Correlation Coefficient
Ran1	7.8581e-04
Perl Rand	4.1554e-04
Matlab Rand	4.6485e-04

It is not surprising that all three random number generators passed this test. This is because the 2-D scatter plot (6.5.3) already showed visually that there was no correlation. The serial correlation test supports this result quantitatively.

5.5.7 <u>Comparison of Optimized Results</u>

The statistical tests attempted to assign probabilities to the hypothesis that the sequences of random numbers produced by the three random number generators were random. There are two obvious problems with this approach. First, it is already known that the sequences are not random since they were generated by a computer following an explicit function. Second, the results are subject to interpretation as to whether a given probability (20% for example) is or is not sufficiently "random" for the proposed application.

Knuth [78] writes that, "a truly random sequence will exhibit local nonrandomness; local nonrandomness is necessary in some applications, but it is disastrous in others. We are forced to conclude that no sequence of "random" numbers can be adequate for every application." Press et al. [84] further explain that, "A pragmatic point of view, then, is that randomness is in the eye of the beholder (or programmer). What is random enough for one application may not be random enough for another."

By far, the best test of a random number generator is in the actual application that it would be used. For this purpose, the MSP '01 optimization problem (described in Chapter 10) was completed twice – first with the "ran1" random number generator and second with the Perl “rand” random number generator. Table 10 compares optimized solutions using the “ran1” and Perl “rand” random number generators.

Table 10: Optimization Comparison, MSP '01.

	Ran1	Perl Rand	Difference
x_1	0.0479888	0.04720095	1.66%
x_2	5.0000000	5.00001856	0.00%
x_3	0.92537808	0.92664521	0.14%
x_4	0.17286401	0.17297628	0.06%
x_5	0.12213146	0.12235445	0.18%
x_6	0.09108389	0.09118628	0.11%
x_7	0.09995603	0.0996624	0.29%
x_8	0.00248455	0.00248678	0.09%
x_9	0.04247787	0.04266928	0.45%
x_{10}	0.04389342	0.04403462	0.32%
x_{11}	0.01705527	0.01699975	0.33%
x_{12}	0.0244741	0.02437944	0.39%
x_{13}	15.5818713	15.55111435	0.20%
x_{14}	0.13331093	0.13171737	1.20%
x_{15}	0.01124481	0.01121914	0.23%
x_{16}	0.00846536	0.00846064	0.06%
x_{17}	0.00825664	0.00827603	0.23%
x_{18}	3216.05921	3212.452625	0.11%
x_{19}	885.919342	885.9009899	0.00%
x_{20}	1916.18485	1921.492353	0.28%
x_{21}	2.04263505	2.04517699	0.12%
x_{22}	1.47950077	1.47889136	0.04%
x_{23}	2.10088651	2.09903687	0.09%
x_{24}	0.00041319	0.00041051	0.65%
x_{25}	1.95192831	1.95191665	0.00%
x_{26}	0.12015699	0.12023507	0.06%
x_{27}	0.06969548	0.06995173	0.37%

The close agreement between these two results (less than 2% difference) indicates that either random number generator is appropriate for use in this application. The similar solutions also bolster confidence in the results. Note that some differences are expected just due to the randomness of the Monte Carlo analysis.

Based on the results of these seven tests for randomness, we conclude that “ran1” is sufficiently random. While some suspicion was introduced in the frequency test, the results of the other tests support this conclusion. Particular notice is given to the optimization results given in this section since they apply directly to this research.

CHAPTER 6

LANDING FOOTPRINTS

Because an objective of this methodology is to control the size of a landing footprint, it is necessary to first clearly state how to define the footprint. This last statement may at first seem trivial, but consider that there are many ways to make such a definition. In fact, the numbers of ways are limited only by the imagination of the designer. So which way is best? As far as the methodology is concerned, any method is sufficient (so long as the designer clearly states the definition). A clear statement is needed so that others may reproduce and properly interpret the results. After all, the purpose of the footprint in the optimization is only to provide one or more numbers that indicate the degree to which our goals are met.

Physically, of course, the footprint is a boundary – drawn in the sand (so to speak). The footprint is drawn such that a certain probability is associated with landing within the boundary. This means that any proper (and complete) definition for a footprint must contain two statements. The first statement must explain precisely how the boundary is to be drawn. In order that others may reproduce the work, this statement must include the precise method with which the boundary is determined from a set of data. The second statement must provide the probability associated with the footprint. Strictly speaking, only the method of drawing the boundary is required to employ an optimization scheme (since this provides the number we seek). However, the statement of probability is necessary for others to properly interpret the results. For example, a 3-km footprint with a 1% probability is very different from a 3-km footprint with a 99% probability.

Often in the literature, only one statement or the other is made (e.g., an author may refer to a “3-sigma” ellipse or a “99th-percentile” ellipse). However, these practices are incomplete. If an ellipse is 3-sigma, then what is the probability of landing within the ellipse? The answer depends on the shape of the actual distribution. If there is a 99% probability of landing within the ellipse, then how is the boundary constructed (or drawn) from the data?

In general, a footprint can take on any shape. The customary shape, however, is an ellipse. This research therefore, will only discuss ellipses and their degenerate form, the circle. Five methods of drawing a footprint are described along with their advantages and disadvantages. The first three produce circles. The final two produce ellipses.

6.1 <u>3-Sigma Radius</u>

The first method of drawing a footprint is to create a circle with a 3-sigma radius. For each simulation, record the latitude and longitude of the landing site and calculate the straight-line distance (or range) to the target landing site. This range is called the miss-distance. Next, the sample mean, μ, and variance, σ^2, of the miss-distance are determined. Finally, a circle is constructed, which is centered on the target with a radius equal to the $\mu+3\sigma$ miss-distance.

The sample mean and variance may be calculated in a running fashion, using Equation 13 and Equation 14. This reduces storage requirements, since each x_i need not be stored. The sequence is initialized by setting $\sigma_1^2=0$ and $\mu_1=x_1$. The variance should be updated first, since it depends on the previous mean.

$$\sigma_i^2 = \sigma_{i-1}^2 \frac{(i-2)}{(i-1)} + \frac{(\mu_{i-1} - x_i)^2}{i} \tag{13}$$

$$\mu_i = \mu_{i-1} \frac{(i-1)}{i} + \frac{x_i}{i} \tag{14}$$

The 3-sigma method is the simplest and requires only a minimal amount of calculation. Because only one parameter (radius) is used, only one constraint is needed in the optimization. This simplifies the optimization and ensures the existence of a solution. An additional advantage of this method is that it makes no assumption on the shape of the underlying distribution. This makes the 3-sigma method applicable to many situations. This method was chosen for the proof-of-concept problem.

The primary disadvantage of this method, however, is that the probability associated with the footprint is unknown. The reason for this is that, while the definition of the radius requires no assumption on the distribution of the miss-distance, the probability depends heavily upon its shape. Additionally, since the shape of the distribution may change as the uncertainties are reduced, the probability (even if it were known) would not be consistent. Because of this, two footprints may not compare well. Even though they were both 3-sigma, they might have different probabilities.

The lack of a theoretical probability seems disastrous, but it is not (as will be seen in Section 7.6). The choice in the number of standard deviations (three) in the definition was arbitrary, but not without some basis. It was chosen to be "far out on the tail" of the distribution. If the distribution is normal, then $\mu+3\sigma$ will include 99.87% of all cases. Since the distribution is not normal, the probability will not be .9987, but it should be fairly close (certainly greater than 50%, probably greater than 90%, and potentially greater than 99%). For example, if the distribution is Rayleigh with scale parameter equal to 1.2, then the $\mu+3\sigma$ probability is .9944.

Additionally, while the probability can not be predicted, it can be determined empirically. After the radius is calculated, count the number of simulations that fall within it and divide by the total number of cases. This simple procedure can be applied to any of the methods as a check on the theoretical probabilities.

6.2 Rayleigh

The second method constructs a *p*-confidence circle by assuming that the miss-distance follows a Rayleigh distribution, Equation 15. The probability of landing within this circle is *p* (which is a user input). The Rayleigh distribution is a good choice because it has the following special property: if X and Y are independent normal distributions with zero means and equal variances, then $Z=\sqrt{X^2+Y^2}$ has a Rayleigh distribution. So the Rayleigh distribution is a natural choice for range calculations where the underlying distributions are normal.

$$f_{\text{Rayleigh}}(t) = 2\frac{t}{b^2}\exp\left[-\left(\frac{t}{b}\right)^2\right] \tag{15}$$

The single parameter, *b*, of the Rayleigh distribution is the scale parameter. The cumulative distribution function (cdf) of the Rayleigh is given by Equation 16. The inverse of this function has the closed-form expression, Equation 17. Using Equation 17 then, it is a simple matter to determine the radius that gives a probability, *p*.

$$F_{\text{Rayleigh}}(t) = 1 - \exp\left[-\left(\frac{t}{b}\right)^2\right] \tag{16}$$

$$F^{-1}_{\text{Rayleigh}}(p) = b\left[-\ln(1-p)\right]^{\frac{1}{2}} \tag{17}$$

The method of constructing the *p*-confidence circle is described as follows. For each simulation, record the latitude and longitude of the landing site and calculate the miss-distance. The sample mean and variance of the miss-distance are calculated in a running fashion, using Equation 13 and Equation 14. The scale parameter, *b*, of the Rayleigh distribution is found from the sample mean using Equation 18. This parameter could equally be determined from Equation 19 and the sample variance. Finally, a circle is constructed, which is centered on the target landing site, with radius $R=F^{-1}(p)$ from Equation 17.

$$\mu_{\text{Rayleigh}} = b\frac{\sqrt{\pi}}{2} \tag{18}$$

$$\sigma^2_{\text{Rayleigh}} = b^2\left[1 - \frac{\pi}{4}\right] \tag{19}$$

The advantage of this method is the ease with which a footprint can be constructed to any probability, *p*. Like the 3-sigma method, only one parameter (radius) is used; and therefore, only one constraint is needed in the optimization.

The primary limitation of this method stems directly from the fundamental assumption that the miss-distance is Rayleigh distributed. This assumption presumes that the down-range and cross-range distributions (from which the miss-distance is calculated) are normal and have equal variances. It is the implied assumption of equal variances that it the problem. The variances are only equal when the footprint is circular. Therefore, this method is only appropriate for near circular footprints (i.e., ellipses with eccentricities near zero).

6.3 Weibull

The third method, an extension of the Rayleigh method, constructs a p-confidence circle by assuming that the miss-distance follows a Weibull distribution, Equation 20. The probability of landing within this circle is *p* (which is a user input). This extension is natural because the Weibull distribution is a generalization of the Rayleigh distribution. In fact, the Rayleigh distribution is a special case of a Weibull distribution where the shape parameter, *k*, is exactly two.

$$f_{\text{Weibull}}(t) = k\frac{t^{k-1}}{b^k}\exp\left[-\left(\frac{t}{b}\right)^k\right] \tag{20}$$

The Weibull distribution is commonly used to describe failure time in reliability studies, and the breaking strengths of materials in reliability and quality control tests. Early references on the Weibull distribution include Fisher and Tippet [85] and Weibull [86, 87]. The two parameters, *b* and *k*, of the Weibull distribution are the scale and shape parameters, respectively.

The shape parameter makes the Weibull distribution extremely versatile. When *k*=1, the Weibull takes the shape of the exponential distribution; when *k*=2, the Weibull takes the shape of the Rayleigh distribution; and when *k*=3.25, the Weibull closely approximates a normal distribution. The cumulative distribution function (cdf) of the Weibull is given by Equation 21. The inverse of this function has the closed-form expression, Equation 22. Using Equation 22 then, it is a simple matter to determine the radius that gives a probability, *p*. The mean and variance of the Weibull are given by

Equation 23 and Equation 24. The symbol, Γ, denotes the special math function, Gamma (the continuous generalization of the factorial).

$$F_{\text{Weibull}}(t) = 1 - \exp\left[-\left(\frac{t}{b}\right)^{k}\right] \tag{21}$$

$$F^{-1}_{\text{Weibull}}(p) = b\left[-\ln(1-p)\right]^{\frac{1}{k}} \tag{22}$$

$$\mu_{\text{Weibull}} = b\Gamma\left(1+\frac{1}{k}\right) \tag{23}$$

$$\sigma^2_{\text{Weibull}} = b^2\left[\Gamma\left(1+\frac{2}{k}\right) - \Gamma^2\left(1+\frac{1}{k}\right)\right] \tag{24}$$

Unfortunately, there is no closed form solution for estimating the parameters of a Weibull. Instead, simultaneous non-linear equations must be solved for the maximum likelihood parameter estimates. Fortunately, one of these equations can be solved for *b* in terms of *k*. [88] Substituting this expression, Equation 25, into the other equation results in a single, more complicated, expression with *k* as the only unknown.

$$b = \left(\frac{n}{\sum_{i=1}^{n} x_i^k}\right)^{\frac{-1}{k}} \tag{25}$$

This more complicated expression must be solved iteratively (typically using a Newton-Raphson technique). The update for *k* is Equation 26, where *g* and g' are given by Equation 27 and Equation 28. Once the shape parameter, *k*, has been found, it is substituted back into Equation 25 to determine the scale parameter, *b*.

$$k_i = k_{i-1} - \frac{g}{g'} \tag{26}$$

$$g = \frac{n}{k} + \sum_{i=1}^{n} \ln(x_i) - n \frac{\sum_{i=1}^{n} x_i^k \ln(x_i)}{\sum_{i=1}^{n} x_i^k} \tag{27}$$

$$g' = \frac{-n}{k^2} - \frac{n}{\left(\sum_{i=1}^{n} x_i^k\right)^2} \left\{ \sum_{i=1}^{n} x_i^k \sum_{i=1}^{n} [\ln(x_i)]^2 x_i^k - \left[\sum_{i=1}^{n} x_i^k \ln(x_i)\right]^2 \right\} \tag{28}$$

An initial closed-form estimate for *k* is given by Law and Kelton [89] and Menon [90]. Leemis [88] notes that this initial estimator, Equation 29, should be reasonably close to the maximum likelihood estimator, *k*, and allows the algorithm to converge to the desired accuracy in just a few iterations.

$$k_0 = \frac{6}{(n-1)\pi^2} \left\{ \sum_{i=1}^{n} [\ln(x_i)]^2 - \frac{1}{n}\left[\sum_{i=1}^{n} \ln(x_i)\right]^2 \right\}^{-\frac{1}{2}} \tag{29}$$

Finally, the designer may even check the fit of a Weibull distribution by plotting the data on a Weibull plot. If the Weibull distribution is an appropriate model for the data, then a plot of ln(*t*) versus ln[-ln(1-*F*(*t*))] should result in a straight line. [88]

The method of constructing the *p*-confidence circle is summarized as follows. For each simulation, record the latitude and longitude of the landing site and calculate the miss-distance. Then estimate the parameters *b* and *k* from the data using the procedure described above. Finally, a circle is constructed, which is centered on the target landing site, with radius $R=F^{-1}(p)$ from Equation 22.

Similar to the Rayleigh method, the primary advantage of this method is the ability to construct a footprint to any probability, *p*. Like the previous two methods, only one parameter (radius) is used; and therefore, only one constraint is needed in the optimization. The extra shape parameter allows better fits in most cases than the Rayleigh. An additional benefit of the Weibull method is the ability to assess the validity of the assumption using a Weibull plot.

The primary disadvantage of this method is the iterative calculations involved in the parameter estimation. These calculations require notable time and effort and must be performed after all of the data has been obtained. Also, while the Weibull distribution avoids the requirement of equal variances, the designer still must take care to ensure that the Weibull assumption is acceptable.

6.4 3-Sigma Down-range and Cross-range

The next method of drawing a footprint is the two parameter analogy to the 3-sigma range. This method constructs an ellipse, which has a 3-sigma semi-major axis in the down-range direction and a 3-sigma semi-minor axis in the cross-range direction. For each simulation, we record the down-range and cross-range distances to the target landing site with respect to the nominal azimuth angle. Next, determine the sample means and variances of the ranges. Finally, construct an ellipse that is centered on the mean landing site with a semi-major axis, a, equal to the 3σ down-range and a semi-minor axis, b, equal to the 3σ cross-range. The sample mean and variance may be calculated in a running fashion, using Equation 13 and Equation 14.

Like the 3-sigma range, this method is simple and requires only a minimal amount of calculation. Because two parameters (a and b) are used, the designer is allowed more control of the shape of the footprint. An additional advantage of this method is that it makes no assumption on the shape of the underlying distribution. This makes the 3-sigma down-range and cross-range method applicable to many situations. This method was used during the experimentation process of the MSP '01 problem.

The primary disadvantage of this method is that the probability associated with the footprint is unknown. The reason for this is that, while the definitions of a and b require no assumptions on the distributions of the down-range and cross-range distances, the probability depends heavily upon their shapes. Also, because the cross-range and down-range distributions are two-sided, the ellipse is centered on the means. This has the undesirable side-effect that the ellipse may not be centered on the target. Finally, while the extra parameter improves control over the shape of the ellipse, it complicates the optimization procedure by adding an additional constraint.

6.5 Bivariate Normal

The final and most elegant method is the bivariate normal. In this method, we construct a *p*-confidence ellipse by assuming that the down-range and cross-range values follow a bivariate normal (BVN) distribution, Equation 30. The parameter, ρ, in this equation is the correlation coefficient. The BVN distribution has the properties that the conditional distribution of one variable, given the other, is normal and the contours are ellipses. The probability of landing within the constructed ellipse is *p* (which is a user input).

$$f(x,y)=\frac{1}{2\pi\sigma_x\sigma_y\sqrt{(1-\rho^2)}}\exp\left\{\frac{-1}{2(1-\rho^2)}\left[\left(\frac{x-\mu_x}{\sigma_x}\right)^2-2\rho\left(\frac{x-\mu_x}{\sigma_x}\right)\left(\frac{y-\mu_y}{\sigma_y}\right)+\left(\frac{y-\mu_y}{\sigma_y}\right)^2\right]\right\}$$

(30)

6.5.1 Constructing the Ellipse

Assuming for the moment that a BVN is constructed from two independent standard normal distributions (*X* and *Y*), then the miss-distance, given by Equation 31, has a Rayleigh distribution with scale parameter $b=\sqrt{2}$. Using the procedure for the Rayleigh method, a *p*-confidence circle could be constructed.

$$R=\sqrt{X^2+Y^2}$$ **(31)**

If a transformation existed between this space of standard normal distributions and a space of non-standard and correlated normal distributions (defined by a covariance matrix), then we could project the p-confidence circle into any given non-standard normal space. The circle would take on the shape of an ellipse under the transformation, which would be the desired *p*-confidence ellipse. Such a transformation does exist and it is the Cholesky decomposition of the covariance matrix.

The proof of this, follows beginning with the desired covariance matrix, Σ, given by Equation 32. The matrix shown is two dimensional, but the proof applies generally to *k* dimensions. The function denoted by cov(*x*,*y*) is this covariance of *x* and *y* and is defined by Equation 33, where *E*(*x*) is the expectation (or mean) of *x*. Since the covariance matrix is symmetric and positive definite, it has a Cholesky decomposition given by Equation 34.

$$\Sigma = \begin{bmatrix} \sigma_x^2 & \operatorname{cov}(x,y) \\ \operatorname{cov}(x,y) & \sigma_y^2 \end{bmatrix} \qquad \textbf{(32)}$$

$$\operatorname{cov}(x,y) = E(xy) - E(x)E(y) \qquad \textbf{(33)}$$

$$\Sigma = LL^T \qquad \textbf{(34)}$$

Next, take a *k*-dimensional space of independent standard normal distributions, denoted $N_k(0,I)$. Construct a *k*-vector of deviates from the *k* independent distributions, W. Because W is $N_k(0,I)$, the mean is *E*(W)=0 and the variance is Var(W)=$E(WW^T)$=I.

Finally, construct a new vector Z=*L*W by applying the transform, *L* (the Cholesky decomposition of Σ), to W. Because a linear combination of normal distributions results in a normal distribution, the vector Z is also normal. The mean of Z is *E*(Z) = *E*(*L*W). Because the expectation is a linear operator, *E*(*L*W)=*L*(*E*(W)) = O. Similarly, the variance is Var(Z) = Var(*L*W) = $E(LWW^TL^T) = L(E(WW^T))L^T = LL^T = \Sigma$. The vector Z is then $N_k(0,\Sigma)$. Therefore, Z belongs to a *k*-dimensional space of normal distributions with the covariance matrix, Σ. This proves that *L* is the desired transformation.

Figure 10 shows a standard $N_2(0,I)$ bivariate normal space. Any vertical slice of this function is a standard normal function, N(0,1). Any horizontal slice is a circle. The distribution of the miss-distance (range to the origin) is a Rayleigh function with scale parameter, *b*, equal to the square root of two. Figure 11 is an example of a general $N_2(0,\Sigma)$ bivariate normal space. The data is correlated with a covariance matrix, Σ. Any horizontal slice of this function is an ellipse.

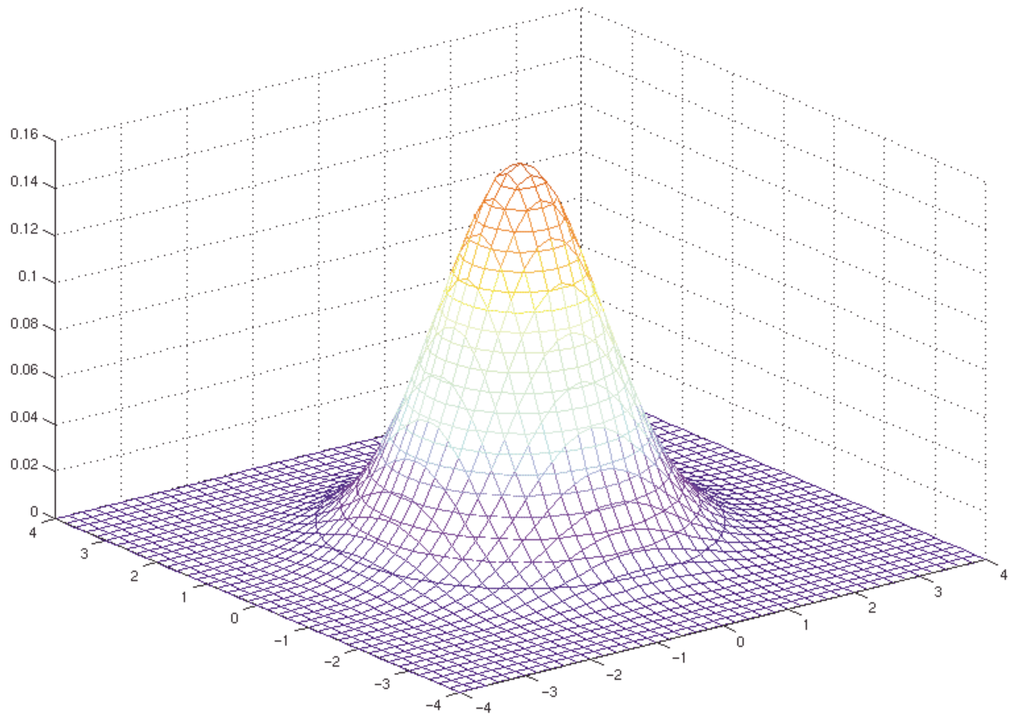


Figure 10: Standard Bivariate Normal Space.

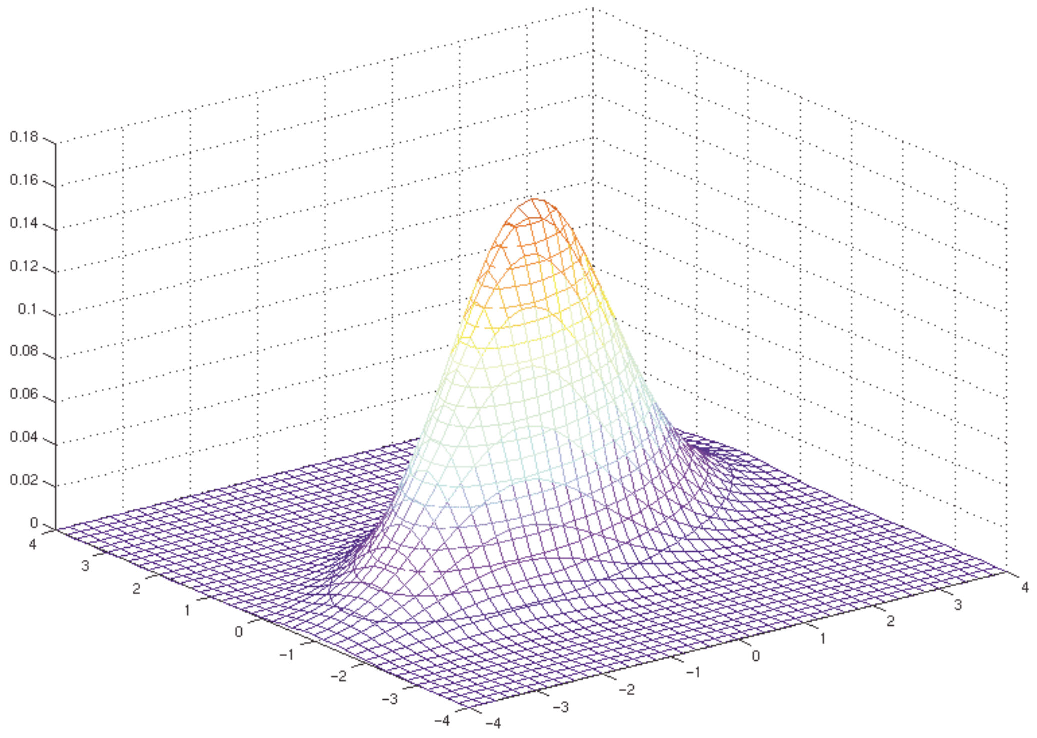


Figure 11: Correlated Bivariate Normal Space.

Figure 12 is a top view of Figure 10. This figure shows the circular contour lines, which result from equal variances and uncorrelated bivariate normal data. Figure 13 is a top view of Figure 11. This figure highlights the elliptical contour lines of the $N_2(0,\Sigma)$ correlated bivariate normal. Note that the semi-major axis is rotated counter clockwise approximately 20 degrees in this example.

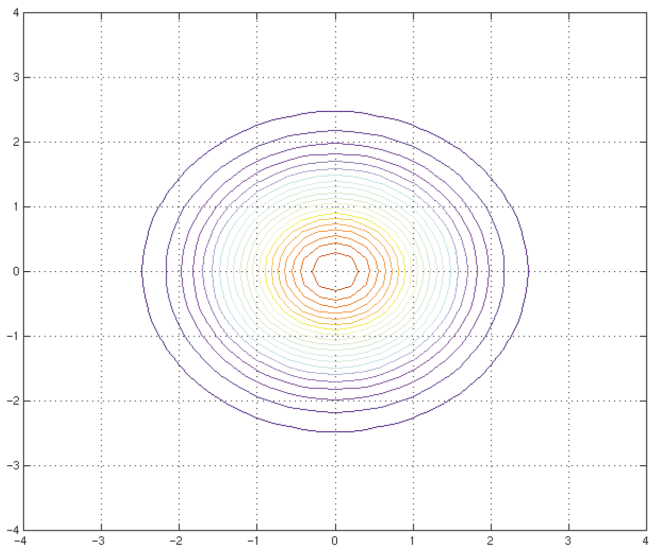


Figure 12: Standard Bivariate Normal Contours.

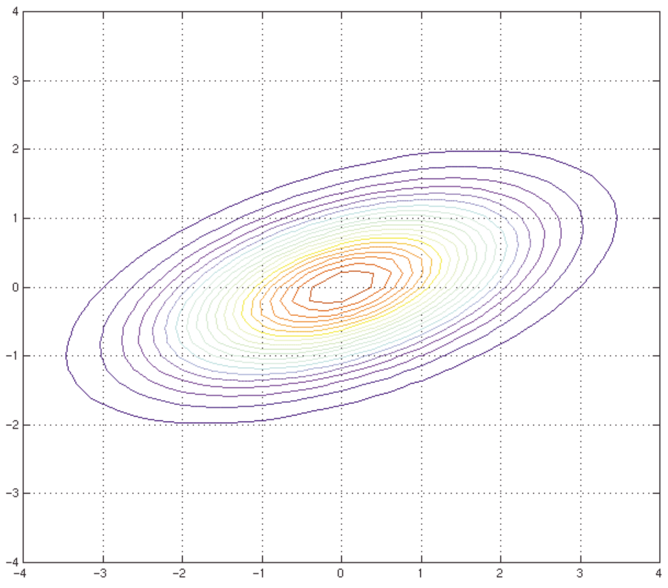


Figure 13: Correlated Bivariate Normal Contours.

The procedure is as follows. For each simulation, record the down-range and cross-range distances to the target with respect to the nominal azimuth angle. Next, determine the sample means and variances of the ranges. Then construct the transformation, *L*, between the correlated down-range and cross-range data (assumed to be from a BVN distribution) and a standard $N_2(0,I)$ space based on the covariance matrix of the data. This covariance is determined from the variances and covariance of the down-range and cross-range data. The covariance is calculated using Equation 33. The transform *L* is the Cholesky decomposition of the covariance matrix (which is symmetric, positive definite). Performing the decomposition results in Equation 35.

$$L = \begin{bmatrix} \sigma_x & 0 \\ \dfrac{\operatorname{cov}(x,y)}{\sigma_x} & \sqrt{\sigma_y^2 - \dfrac{\operatorname{cov}^2(x,y)}{\sigma_x^2}} \end{bmatrix} \tag{35}$$

Since the range in standard normal space has a Rayleigh distribution, construct a circle with a probability of *p* from the inverse of the Rayleigh cumulative distribution function. This cdf for a scale parameter equal to the square root of two is shown in Equation 36. This circle is mapped to an ellipse in real space by applying the transform, *L*. Either Equation 37 or Equation 38 can be used for this purpose, where *x* and *y* are coordinates in real space and x' and y' are coordinates in standard normal space. The transformed circle (now an ellipse – centered on the mean down-range and cross-range and potentially rotated by an angle theta) encloses (100*p*)% of all landings.

$$F^{-1}_{\text{Rayleigh}(\sqrt{2})}(p) = \sqrt{-2\ln(1-p)} \tag{36}$$

$$\begin{bmatrix} x \\ y \end{bmatrix} = L \begin{bmatrix} x' \\ y' \end{bmatrix} + \begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix} \tag{37}$$

$$\begin{bmatrix} x & y \end{bmatrix} = \begin{bmatrix} x' & y' \end{bmatrix} L^T + \begin{bmatrix} \mu_x & \mu_y \end{bmatrix} \tag{38}$$

The dashed line in Figure 14 is an example of a 99.5% BVN footprint ellipse for 2000 simulations. This ellipse is approximately 3-by-2 km. Note that nine simulations lie outside the BVN footprint (10 misses would be expected).

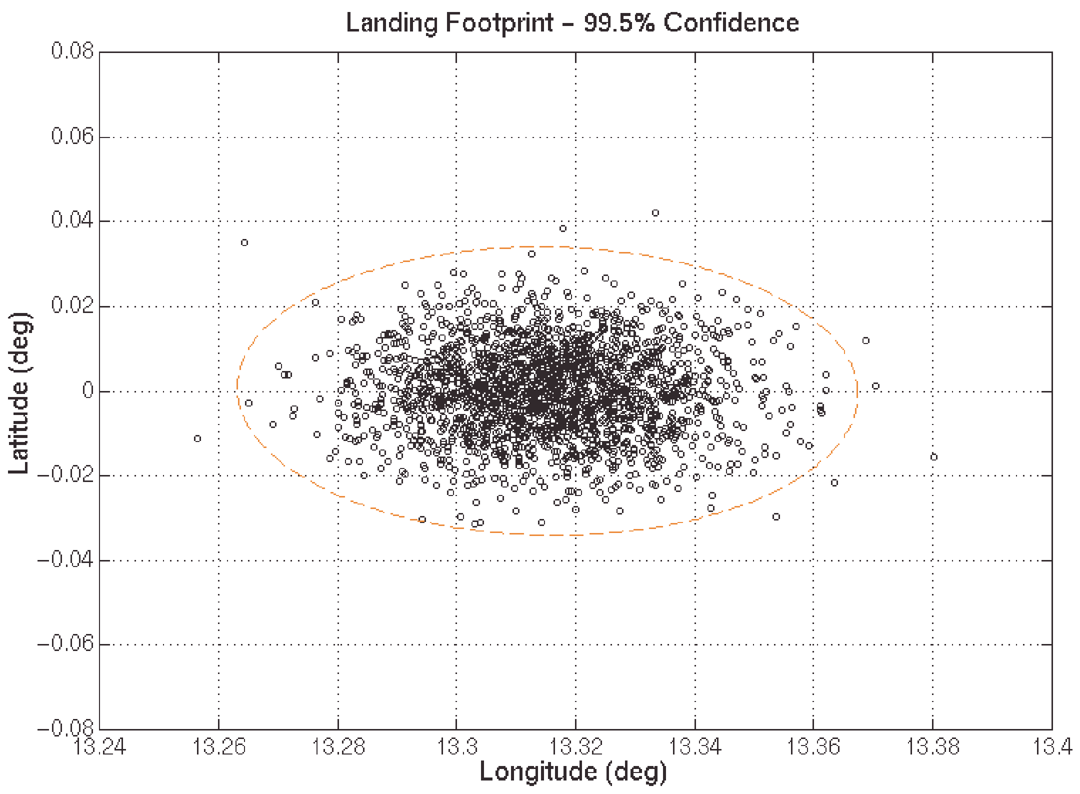


Figure 14: Example BVN Footprint.

6.5.2 Measuring the Size of the Axes

Once the *p*-confidence ellipse has been constructed, the size of the semi-major axis, *a*, and semi-minor axis, *b*, need to be determined. This determination is complicated by the fact that, under the transformation, the ellipse may be rotated by an angle, θ. Fortunately however, the semi-axes may be easily identified since they are the only vectors that retain the property of orthogonality under the transform, *L*.

To find θ, Take the orthogonal vectors [*r*cosθ *r*sinθ] and [-*r*sinθ *r*cosθ] and apply the transform *L* (or alternately post-multiply by L^T, which is equivalent). Next, take the dot product of the transformed vectors *u* and *v*, given by Equation 39 and Equation 40, and set the result, Equation 41, to zero. Substituting the trigonometric identities, Equation 42 and Equation 43, Equation 41 is solved for the unknown angle, θ. The result is shown in Equation 44.

$$u = \begin{bmatrix} r\cos\theta & r\sin\theta \end{bmatrix} \begin{bmatrix} L_{11} & L_{12} \\ L_{21} & L_{22} \end{bmatrix}^T \qquad \textbf{(39)}$$

$$v = \begin{bmatrix} -r\sin\theta & r\cos\theta \end{bmatrix} \begin{bmatrix} L_{11} & L_{12} \\ L_{21} & L_{22} \end{bmatrix}^T \qquad \textbf{(40)}$$

$$\left(L_{12}^2 - L_{11}^2 + L_{22}^2 - L_{21}^2\right)\cos\theta\sin\theta + \left(L_{11}L_{12} + L_{22}L_{21}\right)\left(\cos^2\theta - \sin^2\theta\right) = 0 \qquad \textbf{(41)}$$

$$\cos\theta\sin\theta = \frac{1}{2}\sin(2\theta) \qquad \textbf{(42)}$$

$$\cos^2\theta - \sin\theta = \cos(2\theta) \qquad \textbf{(43)}$$

$$\tan(2\theta) = \frac{-2\left(L_{11}L_{12} + L_{22}L_{21}\right)}{\left(L_{12}^2 - L_{11}^2 + L_{22}^2 - L_{21}^2\right)} \qquad \textbf{(44)}$$

The procedure for determining the size of the ellipse is as follows. Determine the transform, *L* (Equation 35), using the procedure from the previous section. Solve for θ using Equation 44. Equation 39 and Equation 40 then give the vectors for the semi-axes, where *r* is the radius of the desired *p*-confidence Rayleigh circle (from Equation 36). The size of the semi-major axis, *a*, is the norm of the larger vector, Equation 45. The size of the semi-minor axis, *b*, is the norm of the smaller vector, Equation 46.

$$a = \max(\|u\|_2, \|v\|_2) \qquad \textbf{(45)}$$

$$b = \min(\|u\|_2, \|v\|_2) \qquad \textbf{(46)}$$

6.5.3 <u>Confidence Interval for the Axes</u>

An added and very important benefit of the BVN assumption is that it also allows the calculation of confidence intervals on the size of the ellipse axes. These confidence intervals depend on the number of simulations and therefore answer the question, "how many Monte Carlo simulations are necessary?" Of course, this answer will depend on the confidence level and the allowable error.

Assuming the angle θ is zero, the transformation, *L*, (given by Equation 35) reduces to multiplying by the respective down-range or cross-range standard deviation. This is not a bad assumption since a well-chosen reference azimuth angle will result in a zero rotation angle. In this case, the semi-major axis (of the *p*-confidence ellipse) is given by Equation 47, where *x* is either the down-range or cross-range, such that $\sigma_x > \sigma_y$.

$$a = F^{-1}_{Rayleigh(\sqrt{2})}(p)\sigma_x \tag{47}$$

Because the inverse of the Rayleigh cdf is exact, the only error in determining the size of the semi-major axis comes from the error in estimating the standard deviation. From Equation 47, note that the confidence in the standard deviation (or variance) is projected directly to the semi-major ellipse. Because the distributions are assumed normal, the confidence interval on the variances may be calculated using well-known statistical results.

Equation 48 shows the confidence interval for a sample variance, where *A* and *B* are given by Equation 49 and Equation 50. [91] Here n is the number of simulations, s^2 is the sample variance, and α is tail-end probability (1-*p*). Applying Equation 47, the resulting confidence interval on the semi-major axis is Equation 51, where $\hat{a}$ denotes the size of the axis determined using the sample variance, s^2. Similar equations can be written for the semi-minor ellipse. Note that the confidence interval on the semi-major axis is independent of the probability, *p*, of the ellipse.

$$\frac{(n-1)s^2}{B} \le \sigma^2 \le \frac{(n-1)s^2}{A} \tag{48}$$

$$A = F^{-1}_{\chi^2_{n-1}}(\frac{\alpha}{2}) \tag{49}$$

$$B = F^{-1}_{\chi^2_{n-1}}(1-\frac{\alpha}{2}) \tag{50}$$

$$\sqrt{\frac{(n-1)}{B}}\hat{a} \le a \le \sqrt{\frac{(n-1)}{A}}\hat{a} \tag{51}$$

Values for the divisors *A* and *B* were determined for values of n ranging from 2000 to 10000 for the confidence levels 90%, 95%, and 99%. These values are shown in Table 11 and Table 12. Table 11 provides calculated values of the divisor, *A*, for various combinations of confidence level and numbers of simulations. *A* is defined by Equation 49.

Table 11: Divisor for Confidence Interval for σ^2 (*A*).

Confidence	n=2000	n=4000	n=6000	n=8000	n=10000
90%	1896.1	3853.0	5820.0	7792.1	9767.5
95%	1877.0	3825.6	5786.2	7753.0	9723.7
99%	1839.9	3772.4	5720.6	7677.0	9638.5

Table 12 provides calculated values of the divisor, *B*, for various combinations of confidence level and numbers of simulations. *B* is defined by Equation 50.

Table 12: Divisor for Confidence Interval for σ^2 (*B*).

Confidence	n=2000	n=4000	n=6000	n=8000	n=10000
90%	2104.1	4147.2	6180.3	8208.2	10233
95%	2124.8	4176.2	6215.6	8248.8	10278
99%	2165.6	4233.1	6284.9	8328.6	10367

Table 13 shows the BVN footprint confidence interval for the ratio of actual over calculated semi-major axis. Various combinations of confidence level and numbers of simulations are shown. Note that this table is independent of the probability, *p*, associated with the footprint. Note also that the error in the size of the ellipse is reduced by either accepting a lower confidence or increasing the number of simulations.

Table 13: Confidence Interval Ratios for Ellipse Semi-axes.

Conf.	*n*=2000	*n*=4000	*n*=6000	*n*=8000	*n*=10000
90%	0.975 - 1.027	0.982 - 1.019	0.985 - 1.015	0.987 - 1.013	0.989 - 1.012
95%	0.970 - 1.032	0.979 - 1.022	0.982 - 1.018	0.985 - 1.016	0.986 - 1.014
99%	0.961 - 1.042	0.972 - 1.030	0.977 - 1.024	0.980 - 1.021	0.982 - 1.019

Figure 15 is constructed from the data in Table 13. It shows the footprint error as a function of number of simulations for four common confidence levels: 90%, 95%, 98%, and 99%. This figure may be used *a priori* to determine the number of simulations required to achieve acceptable errors with the prescribed confidence level.

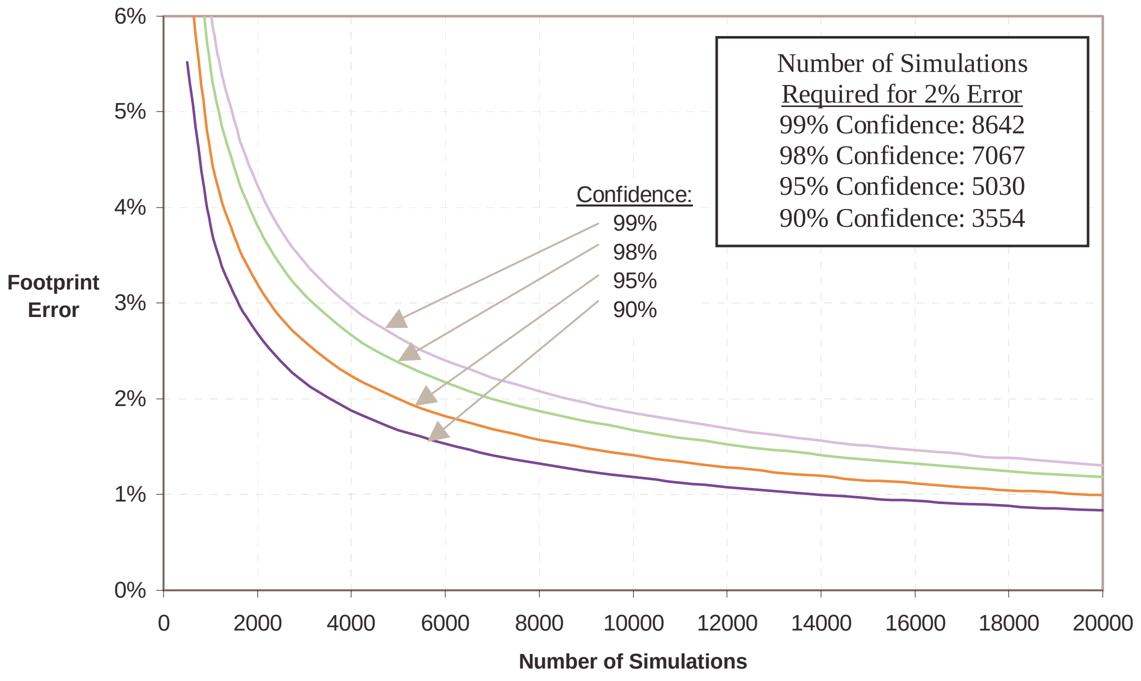


Figure 15: BVN Footprint Error vs. Number of Simulations.

6.5.4 Determining the Empirical Probability

To determine the experimental probability of landing within an ellipse, the number of cases landing outside the ellipse must be counted. A similar approach to the BVN transform is used to easily count these cases. Equation 52 shows the transformation from a unit circle to a general ellipse with semi-axes *a* and *b* rotated by angle θ. The inverse transform Equation 53, therefore, transforms the general ellipse back to a unit circle. Each down-range and cross-range pair may be transformed under Equation 53. Once transformed, the distance is measured to the origin of the unit circle space. Any point with a miss-distance greater than unity lies outside the ellipse. The number of cases outside the ellipse are counted and divided by the number of simulations to produce the probability.

$$T = \begin{bmatrix} a\cos\theta & a\sin\theta \\ -b\sin\theta & b\cos\theta \end{bmatrix} \quad (52)$$

$$T^{-1} = \begin{bmatrix} \frac{1}{a}\cos\theta & -\frac{1}{b}\sin\theta \\ \frac{1}{a}\sin\theta & \frac{1}{b}\cos\theta \end{bmatrix} \quad (53)$$

This is a simple procedure to determine the empirical probability associated with any general ellipse. The procedure is not restricted to BVN ellipses. It may be applied to any ellipse determined by any procedure.

6.5.5 Summary

A footprint is constructed which assumes the landing data (down-range and cross-range) can be represented by a bivariate normal distribution function, given by Equation 30. This footprint has an arbitrary probability, p (i.e., the confidence associated with the ellipse is a user defined value). Typical values of p are higher than 0.99 (0.995 was used in this research). The procedure for creating, measuring, and determining the error of a BVN footprint is summarized as follows:

1) Generate down-range and cross-range data based on target landing site and nominal azimuth angle. Alternately, latitude and longitude may be used directly. In this case, however, the size of the ellipse will be expressed in degrees (which is not a very useful unit of measure for distance).

2) Calculate sample statistics (mean and variance) for down-range, cross-range, and the product of down-range and cross-range. The mean of this product is used in calculating the covariance between down-range and cross-range. The covariance is given by Equation 33.

3) Form the transformation matrix, L, by the Cholesky decomposition of the covariance matrix. The lower-triangular matrix L is a function of the sample statistics, given by Equation 35.

4) Determine the footprint ellipse parameters. Find the rotation angle, θ, from the transformation matrix using Equation 44. This angle is needed for measuring the size of the semi-major axis, *a*, and the semi-minor axis, *b*. These parameters are given by Equation 45 and Equation 46, where the vectors *u* and *v* are defined in Equation 39 and Equation 40. The radius, *r*, in these equations is a function of the desired probability, *p*. This radius is determined from the inverse of the Rayleigh cumulative distribution function Equation 36.

5) Construct a p-confidence circle in standard bivariate normal space by generating points along a circle of radius, *r*. This is the same radius determined in the previous step. The number of points generated depends on the desired resolution of the ellipse. Generally, 360 points are sufficient to produce a smooth curve.

6) Transform the *p*-confidence circle points to the correlated bivariate normal space. This transformation is accomplished through multiplication by the transformation matrix, *L*, and then translating the means. Either Equation 37 or Equation 70 is used, depending on whether the points were generated by columns or rows.

7) Plot the footprint. The ellipse is superimposed over a scatter plot of the down-range and cross-range pairs. This ellipse should encompass approximately (100*p*)% of the simulations.

8) To measure the empirical probability, first translate the down-range cross-range pairs by subtracting the mean down-range and cross-range. Then construct the inverse transform matrix, T^{-1}, from Equation 53. Using this transformation, transform all of the translated down-range and cross-range pairs to a space where the ellipse boundary becomes a unit circle. Measure the range from the origin to each point. Any point with a range greater than unity lies outside the ellipse.

9) Check the error bounds on the ellipse by determining the confidence intervals given by Equation 51. This interval depends on the number of simulations, n, and the confidence interval divisors, *A* and *B*. The divisors can be calculated from Equation 49 and Equation 50 for a given number of simulations and desired tail-end probability, α. Alternately, *A* and *B* can be taken from Table 11 and Table 12 for common values of *n* and α.

While this procedure seems complicated, it is easily implemented by following the given equations. One advantage of this method is that the probability at every point on the ellipse is constant (i.e., the ellipse is a horizontal slice of the distribution surface). Practically, this means that a miss outside the ellipse in the down-range axis is no more likely than one in the cross-range axis. Additionally, correlated data may be used without difficulty and the confidence of the ellipse is easily specified. Finally, the assumed distribution makes it possible to predict the error in the size of the ellipse to any desired confidence. This allows the user to determine the number of simulations necessary.

The primary disadvantage of this method is the assumption of normally distributed down-range and cross-range. However, this assumption is very good and the method has been used very successfully. Additionally, It is important to remember that the confidence interval is also dependent on this assumption. Therefore, the error estimates are only as appropriate as the assumption of normality.

6.6 Comparisons

This section compares the different footprint methods on two sets of data; First, the proof-of-concept baseline data and second, the proof-of-concept solution. The proof-of-concept problem is described in Chapter 9. The desired confidence for all of the methods is 99.5%. Of course, the 3-sigma method is not able to target a specific confidence level. However, the combined $\pm 3\sigma$ probability of two independent normal distributions is $.9974^2$ (or 99.48%). Therefore, the 3-sigma circle is expected to have a similar probability. For each of the methods (3-sigma, Rayleigh, Weibull, and BVN) a footprint ellipse was constructed and an empirical probability was determined by counting the simulations that landed outside the ellipse.

Figure 16 is a histogram showing the Weibull approximation of miss-distance from the proof-of-concept baseline Monte Carlo analysis. The super-imposed Weibull curve fits very well with the data. This shows the usefulness the very versatile distribution function for modeling range data.

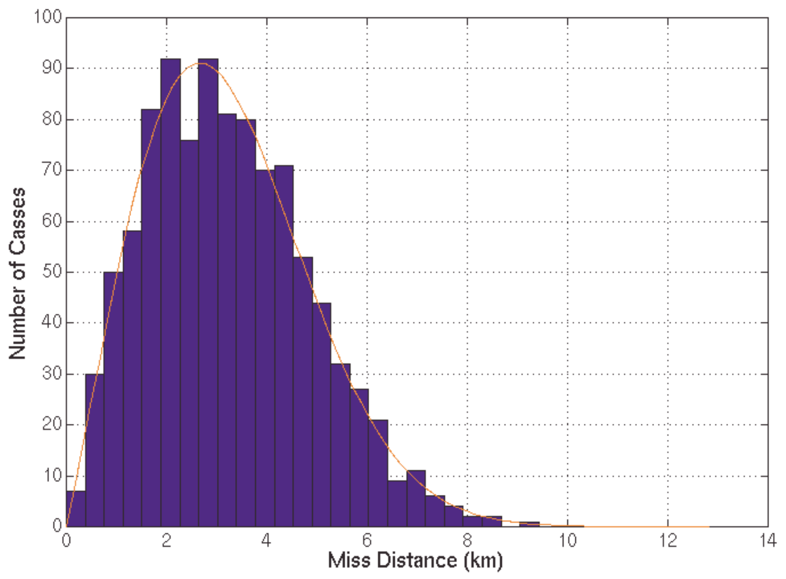


Figure 16: Weibull Approximation, Proof-of-Concept Baseline.

Figure 17 is a Weibull plot of the miss-distance data shown in Figure 16. A plot of data from a Weibull distribution will fall in a straight line with a slope determined by the shape factor and an intercept determined by the scale factor. This plot shows that a Weibull distribution is a good approximation to this data.

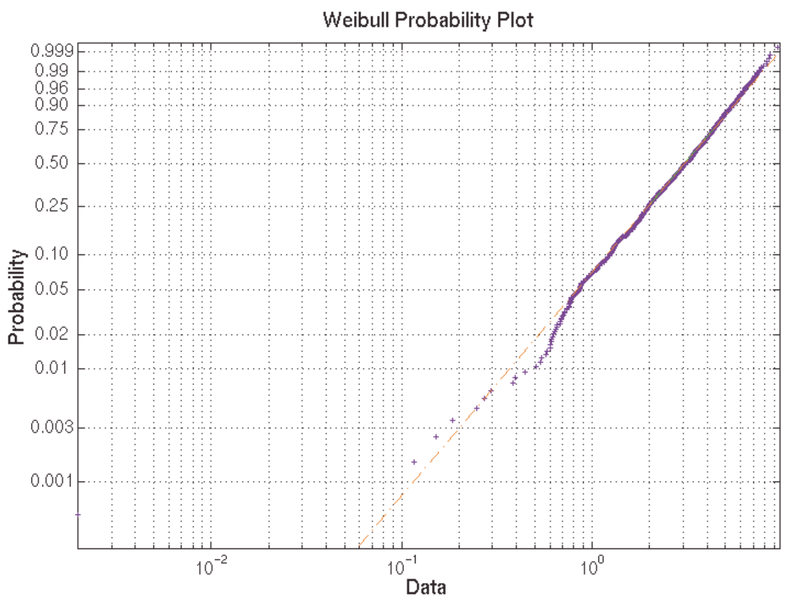


Figure 17: Weibull Plot, Proof-of-Concept Baseline.

Table 14 shows statistics (mean, variance, and probability) calculated from the proof-of-concept baseline Monte Carlo (prior to optimization). Compare these statistics with those predicted by the best-fitting Rayleigh and Weibull functions. Note that the scale parameter, *b*, for the Rayleigh distribution is calculated from the sample mean and variance. Therefore, the mean is exact for the Rayleigh. Note also that the shape parameter, *k*, equals two for the Rayleigh, by definition. The Rayleigh and Weibull approximations were also used to predict the probability of the 3-sigma range. These predictions are given by the parameter $F(R_{3\sigma})$ in Table 14.

Table 14: Statistics, Proof-of-Concept (Baseline).

Parameter	Sample	Rayleigh	Weibull
mean	3.2394	3.2394	3.2361
variance	2.6449	2.8673	2.6596
$F(R_{3\sigma})$	0.9950	0.9928	0.9949
b	-	3.6553	3.6536
k	-	2.0000	2.0831

Table 15 lists footprint sizes and probabilities for the proof-of-concept baseline Monte Carlo (prior to optimization). Note that *a*=*b*=*r* (i.e., the semi-axes are equal) for the methods that produce circular footprints. The data in this table is plotted in Figure 18.

Table 15: Footprint, Proof-of-Concept (Baseline).

Parameter	3σ	Rayleigh	Weibull	BVN
a	8.1183	8.4137	8.1348	8.5835
b	8.1183	8.4137	8.1348	8.0644
eccentricity	0.0000	0.0000	0.0000	0.3425
Area	207.05	222.39	207.89	217.46
p-predicted	?	0.9950	0.9950	0.9950
p-sample	0.9950	0.9970	0.9950	0.9980

Figure 18 compares several of the different methods. Four methods (3-sigma radius, Rayleigh, Weibull, and BVN) were used to generate footprints for the proof-of-concept baseline. This figure shows excellent agreement between all four footprints.

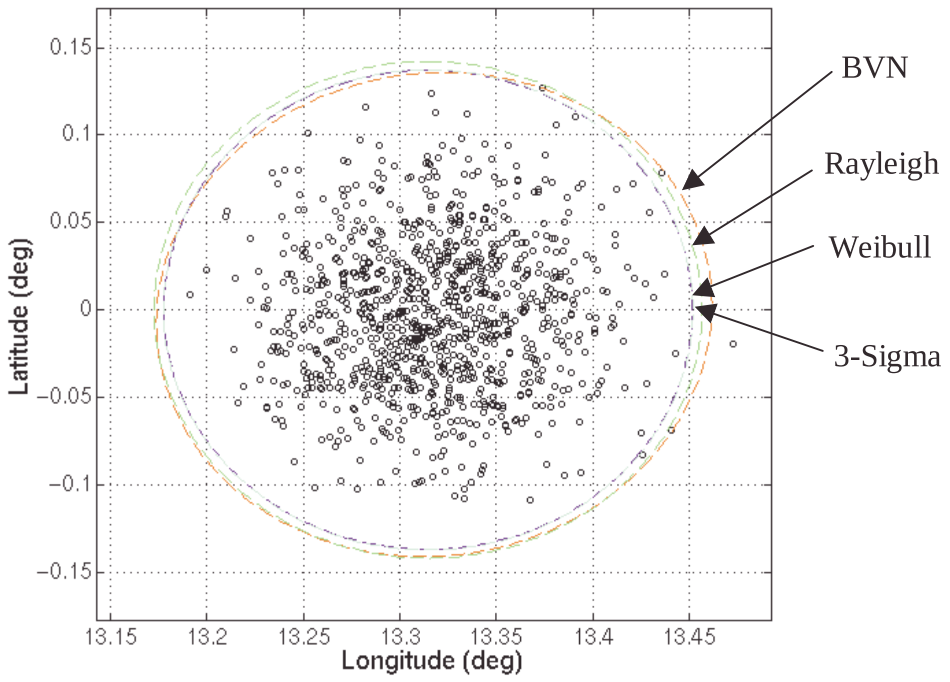


Figure 18: Footprint Comparisons, Proof-of-Concept Baseline.

Figure 19 is a histogram showing the Weibull approximation of miss-distance from the solution to the proof-of-concept problem. Compare this plot with Figure 16. The super-imposed Weibull curve again fits very well with the data. Note that most of the miss-distances are now less than the target 3 km.

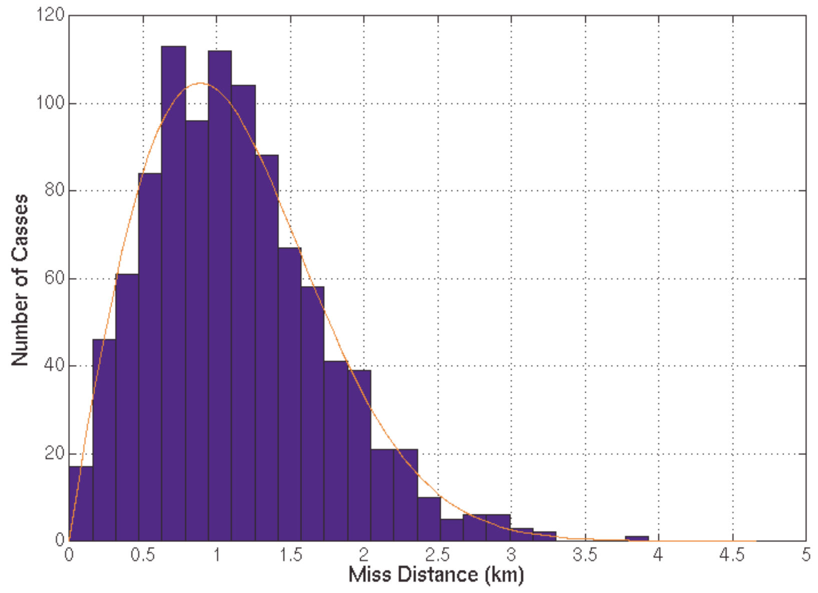


Figure 19: Weibull Approximation, Proof-of-Concept Solution.

Figure 20 is a Weibull plot of the miss-distance data shown in Figure 19. This plot shows that a Weibull distribution is a good approximation to this data.

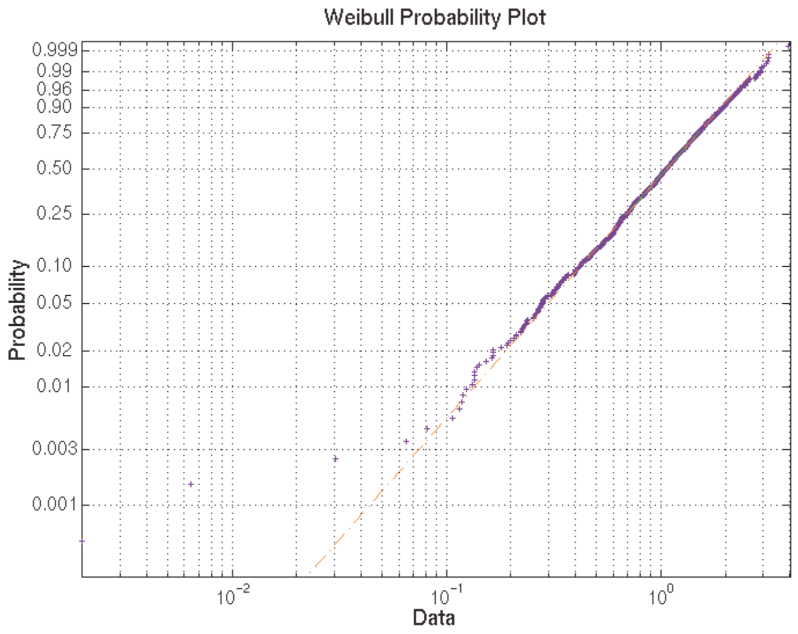


Figure 20: Weibull Plot, Proof-of-Concept Solution.

Table 16 shows statistics from the proof-of-concept optimized solution and compares these with the Rayleigh and Weibull assumptions. Compare this table with Table 14.

Table 16: Statistics, Proof-of-Concept (Solution).

Parameter	Sample	Rayleigh	Weibull
mean	1.1296	1.1296	1.1299
variance	0.3682	0.3487	0.3674
$F(R_{3\sigma})$	0.9944	0.9953	0.9940
b	-	1.2747	1.2741
k	-	2.0000	1.9431

Table 17 lists footprint sizes and probabilities for the proof-of-concept optimized solution. To determine empirical probabilities more accurately, 5000 simulations were performed (rather than the 1000 for the baseline). Compare this table with Table 15.

Table 17: Footprint, Proof-of-Concept (Solution).

Parameter	3σ	Rayleigh	Weibull	BVN
a	2.9500	2.9341	3.0053	3.3402
b	2.9500	2.9341	3.0053	2.5034
eccentricity	0.0000	0.0000	0.0000	0.6620
Area	27.34	27.05	28.37	26.27
p-predicted	?	0.9950	0.9950	0.9950
p-sample	0.9944	0.9942	0.9946	0.9958

Figure 21 is a footprint ellipse for the solution to the proof-of-concept problem. The footprint shown assumes a bivariate normal (BVN) distribution of latitude and longitude. Also shown for reference is the 3-km target radius. This plot shows that the footprint has a non-zero eccentricity, but covers nearly the same area as the circle. This means that while the two ellipses appear very different, they both contain approximately the same percentage of landings.

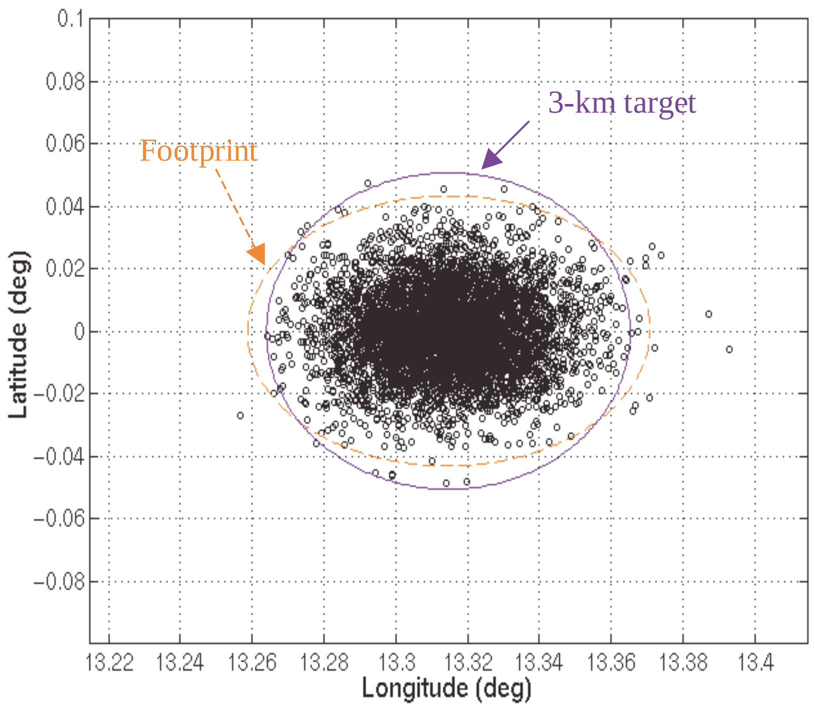


Figure 21: BVN Footprint, Proof-of-Concept Solution.

In summary, four of the methods were demonstrated on two sets of data. The first data set was 1000 latitude-longitude pairs from the proof-of-concept baseline. The second data set was 5000 latitude-longitude pairs from the proof-of-concept solution. All four methods compared very well. The BVN method is recommended, however, because of the ability to calculate a confidence interval.

CHAPTER 7

TRAJECTORY SIMULATION

Chapter 3 described the current state-of-practice in Monte Carlo analyses and Chapter 6 discussed landing footprints. The purpose of these chapters was to provide a brief background to Monte Carlo trajectory simulation and to examine how footprints are constructed, respectively. This chapter explains the mechanics of how trajectory analyses are performed and automated to perform a large number of Monte Carlo simulations. Three computer codes are described: the Program to Optimize Simulated Trajectories (POST), Monte Carlo POST (mcp), and a simple automation script (or “wrapper”).

7.1 POST

The basic trajectory analysis module used in this research is POST [28]. Martin Marietta (now Lockheed Martin) and the NASA Langley Research Center developed POST in 1970 as a Space Shuttle trajectory optimization program. This code is continually updated and has been widely used in industry to solve many ascent, entry, and orbital transfer problems. POST is available in two versions, a 3-Degree of Freedom (3-DOF) version that integrates the translational equations of motion, and a 6-DOF version that simultaneously integrates the translational and rotational equations of motion. The numerical integration within the atmosphere is typically performed using a fourth order Runge-Kutta method, though other options are available. The 3-DOF version of POST is used exclusively in this research. The smaller time-step requirement of the 6-DOF simulation requires significantly more computer time, making 3-DOF a better candidate for optimization.

To run POST, an input deck is first created that specifies the vehicle and central attracting body characteristics as well as a sequence of events (or phases) that will occur along the vehicle's flight path. Aerodynamic and atmospheric properties are typically supplied by interpolating external tables. Guidance algorithms may be supplied by the user, compiled, and linked with the source code.

There were no such specialized modules used in the proof-of-concept. However, a specialized POST executable was used in the Mars Surveyor Program (MSP) 2001 example. This executable included custom routines for the aerodynamics, gravity model, atmospheric properties, and guidance. These models are discussed in Chapter 10.

In the POST input deck, the user also defines independent variables (design variables), dependent variables (constraints), and an objective function for numerical optimization. Numerical optimization is typically used when designing the nominal trajectory to find the independent variables that result in the lowest propellant requirements subject to any mission constraints. Two numerical optimizers are available, a projected gradient optimizer and a Non-linear Programming SOLver (NPSOL). For the proof-of-concept problem, the nominal trajectory was optimized using NPSOL to determine the controls necessary to ensure a soft touchdown. When simulating off-nominal trajectories, POST is run in a non-optimizing mode by specifying a maximum number of iterations equal to negative one (maxitr = -1).

POST generates several output files that provide the user with information concerning the vehicle's trajectory and optimization process. One of these output files, 'profilb', is a binary listing of user specified vehicle and flight path characteristics at a user specified frequency (typically every second). For example, this output file may be used to extract conditions at the time of landing such as geodetic latitude and longitude.

7.2 Monte Carlo POST

Figure 22 sketches the general Monte Carlo trajectory analysis and defines some of the terms used. Mavris and Bandte [55] explain that, "A Monte Carlo Simulation is effectively a random number generator that selects values for each random variable with a frequency proportional to the shape of the corresponding probability distribution." Uncertainty dispersions are generated from random deviates of the appropriate distribution. The range of these dispersions is defined by the extrema (minimum and maximum). A contingency (or off-nominal) trajectory is simulated deterministically with these random dispersions and the values of forecast variables are recorded. This is repeated many times (typically 1000 to 10000 times). Each such simulation requires the generation of a POST input deck, the integration of the equation of motions (by running POST), and the collection of all the forecast results. The results are generally presented graphically as histograms of the forecast variables. Another common representation of the results is a scatter plot of longitude and latitude, which is used to define the landing footprint.

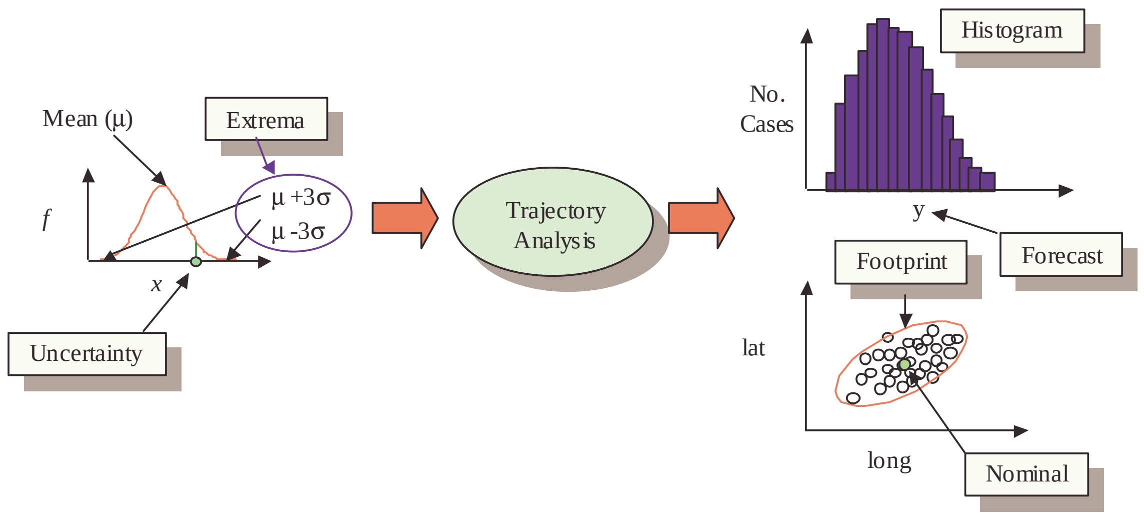


Figure 22: Monte Carlo Process.

Because POST is deterministic, a "wrapper" is needed to evaluate the effects of uncertainty on the nominal trajectory. The wrapper interfaces with POST many times for the purpose of collecting statistics on forecast variables. A Monte Carlo wrapper was written for this research in the Practical Extraction and Reporting Language (Perl) [92]. Perl was chosen because the computer language's powerful regular expressions make it particularly suited for parsing and modifying text files (a handy trait for a wrapper). Perl is an interpreted language and therefore, does not require compilation.

Called “Monte Carlo POST” (or just “mcp”) the wrapper automates the Monte Carlo process by performing a variety of specific tasks. Figure 23 illustrates the automated tasks performed by the mcp program. In this figure, “sim” is a wildcard for the simulation name. The complete source code for mcp is included in Appendix B.

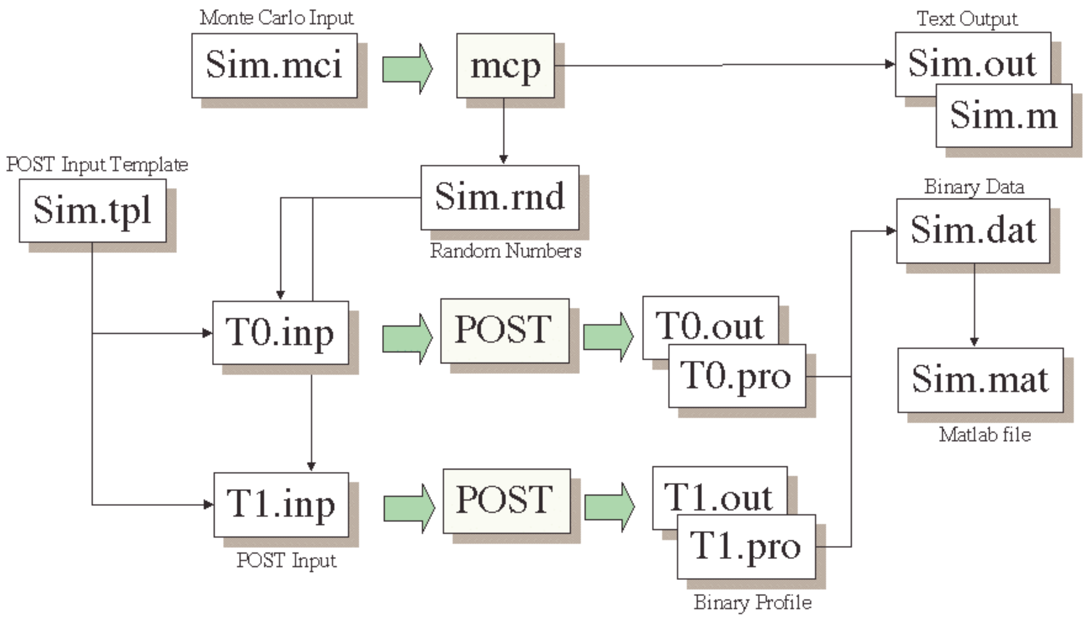


Figure 23: Information Flow, “mcp” Program.

Mcp is a general program that applies to any Monte Carlo problem (i.e., no problem-specific information is contained within the mcp program). All of the problem-specific information is defined in two files: a template POST deck (sim.tpl) and a Monte Carlo input file (sim.mci). This characteristic of the code is important because it allows for the automation of the Monte Carlo process. Altering an input deck is a simple matter compared to altering the code each time.

First, mcp parses uncertainty and forecast variable definitions from the Monte Carlo input file. The “sim.mci” input file contains all of the unique information needed to completely define the Monte Carlo analysis – aside from the nominal trajectory, which is contained within “sim.tpl”. Also defined within this input file are any calculated variables (input or output variables that are calculated from other variables) and any special instructions (e.g., special Matlab instructions that the user desires to include in the automatic Matlab file).

The uncertainties are defined by name, shape (distribution), nominal value, and extrema (minimum and maximum). Four distributions are possible: uniform, normal (or Gaussian), triangular, and discrete. In the case of a normal distribution, the nominal value is the mean and the extrema are the 3-sigma values ($\mu \pm 3\sigma$) by convention. This is done because the normal distribution has no finite minimum or maximum. For the triangular distribution, the nominal value is the mode (most likely value) and the extrema are the actual minimum and maximum values, which need not be symmetric about the mode. For the uniform and discrete distributions, the nominal value can be anywhere within the interval formed by the extrema.

Forecast variables are defined by name, POST output variable name, and the event (or phase) number at which it is desired. The value for the forecast variable is taken at the beginning of the specified event. Any number of forecast variables may be defined.

Next, mcp generates random numbers (dispersions) for each of the uncertainties. All of the random numbers needed for the Monte Carlo process are actually generated before any trajectories are simulated and stored in a file. In this way, this file can be re-used in a subsequent analysis to compare or validate results in a repeatable manner. If an existing file is specified, mcp will use the random numbers in that file rather than creating a new one.

Figure 24 shows a histogram of a uniform deviate. Only a random number generator for a uniform deviate is needed. All other deviates are generated as a function of one or more uniform deviates.

Figure 25 shows a histogram of a normal (or Gaussian) deviate. These numbers were generated using the “K-R” routine. The interval 0 to 1 contains the mean plus and minus 3 times the standard deviation. Of course, 0.26% of the random numbers generated should be outside of this interval.

The basis for generating deviates of different distributions can be done very simply if the inverse of the cumulative distribution function is known. In the case of the normal distribution, the inverse of the cdf is not known analytically. Therefore, normal random numbers are generated using the “K-R” [93] routine, which is based on probability mixing. [94, 95, 96] The Kinderman-Ramage (K-R) algorithm mixes a triangular distribution (from the sum of two uniform distributions) with the appropriate distribution needed to make the resulting mixture standard normal. The triangle acceptance-rejection technique as described by Marsaglia [97, 98] is used for this purpose. The tail algorithm of Marsaglia [99] as modified by Ahrens and Dieter [94] is used to generate the tails of the distribution (values greater than ±2.21603σ). [93]

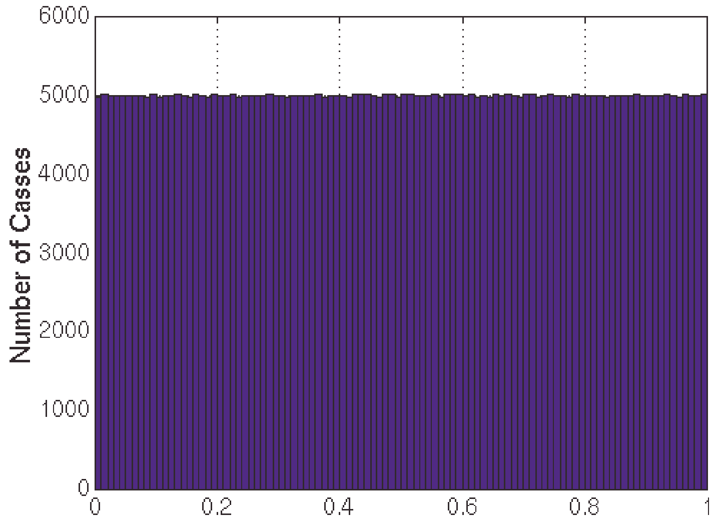


Figure 24: Uniform Deviate.

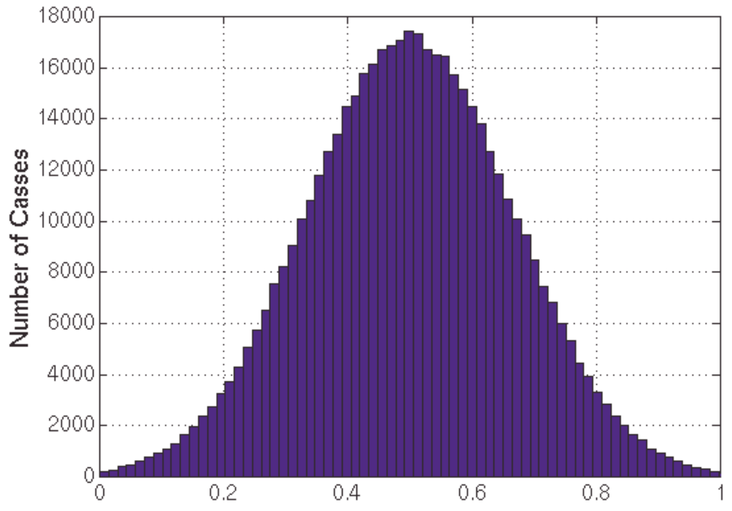


Figure 25: Normal (Gaussian) Deviate.

Figure 26 shows a histogram of a triangular deviate. The interval is from 0 to 1 and the mode can be specified anywhere within the range. Unlike the normal distribution, no numbers are generated outside the interval. Triangular distributions are simulated by inverse transformation. This method first generates a random number from a uniform distribution and then uses the inverse transformation of the cumulative distribution function to arrive at the desired random number.

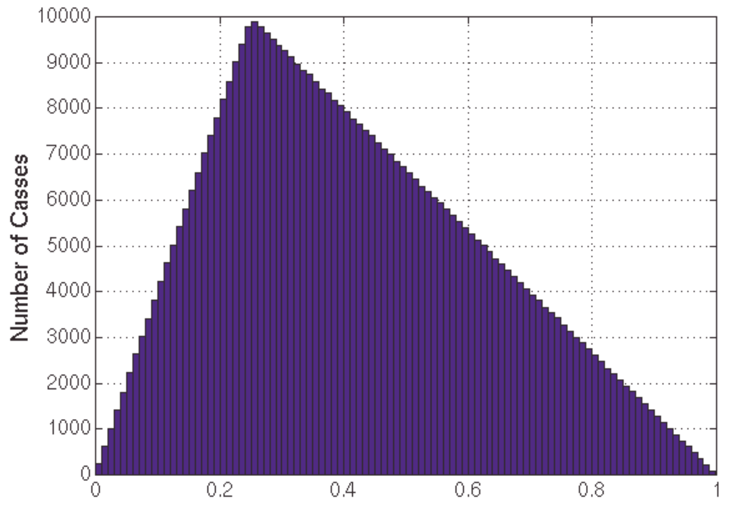


Figure 26: Triangular Deviate.

Figure 27 shows a histogram of a discrete deviate. This deviate divides the uniform deviate into an equal number of sub-intervals (which can be specified). A function discretizes a uniform deviate such that any number falling within a sub-interval is returned as the upper bound of that sub-interval.

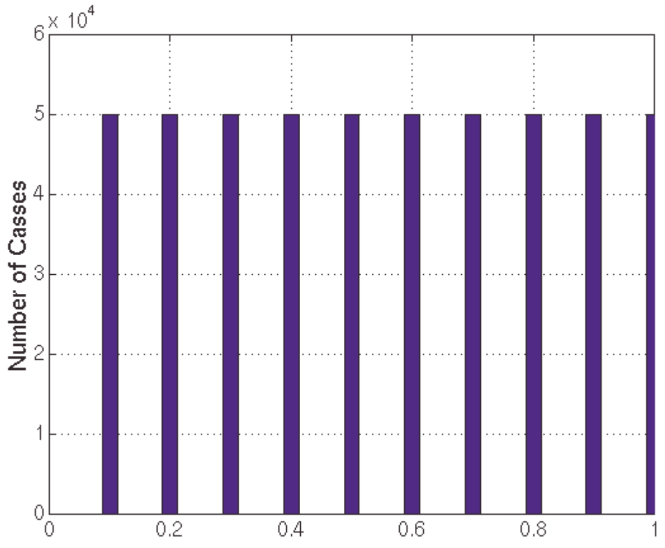


Figure 27: Discrete Uniform Deviate.

Either of two random number generators may be used. For the proof-of-concept problem, the Perl built-in “rand” number generator was used. For the MSP ’01 example, the function “ran1” was used. Both functions are initialized by a seed value, which is taken from the system clock each time the code executes. These random number generators are tested and compared in Chapter 5.

After the random dispersions are generated, unique POST input decks (t0.inp and t1.inp) are created from a template (“sim.tpl”). Placeholders (or markers) – such as ***name*** – are placed in the template to indicate where each of the uncertainties is to be inserted. This template is easily created from the input deck used to design the nominal trajectory. As a check, a nominal trajectory is constructed from the template using the nominal uncertainty values. Run at the beginning of each Monte Carlo process, this trajectory can be compared to the actual nominal to ensure that all markers are being replaced properly.

The next step is to run the trajectory simulations. In order to speed up the process, the Perl "fork" command is used. This command allows the parent mcp process to launch a child process. By doing this, mcp is able to replicate itself (a user specified number of times), creating and running multiple POST decks simultaneously. As each child process finishes, a new one is launched in its place. This strategy takes maximum advantage of multiprocessor parallel machines to significantly reduce the time required to complete the necessary runs. Each processor can be executing a simulation independent of the other processors.

Next, mcp extracts the forecast variables from the POST binary profilb files (t0.pro and t1.pro) and records them in a binary data file (sim.dat). Binary files are used so that information can be transferred without loss of precision. As each value is extracted from the ".pro" files, its influence is added to a running calculation of the sample statistics (minimum, maximum, mean, and variance). These statistics are recorded in a text output file ("sim.out").

When all of the Monte Carlo processes are complete, the results stored in "sim.dat" are written to another binary file in Matlab ".mat" format, which is used to save and retrieve Matlab workspaces. This ".mat" file allows the forecast data to be imported directly into Matlab without any loss of precision.

Finally, a Matlab "m-file" ("sim.m") is automatically generated that loads the "sim.mat" file and displays the results of each forecast variable in a frequency histogram. The latitude and longitude of the touchdown point from each simulated trajectory may also be displayed in a scatter plot, creating a visualization of the landing footprint.

7.3 Automation

Because many experiments must be performed (where each experiment is a complete Monte Carlo analyses), an additional wrapper is needed. This wrapper is a very simple Perl script that automates the process of generating mcp input files, running the Monte Carlo simulations, and creating footprints.

In the same fashion that mcp creates a POST deck from a template (“sim.tpl”); the new wrapper searches for and replaces markers – such as ***X1*** – in a template mcp input file (“sim.gst”). Where mcp replaced uncertainty values with random dispersions, this wrapper replaces the defined uncertainty extrema with desired values. Forecast variable statistics from each Monte Carlo are extracted from the mcp output file (“sim.out”) and used to calculate the size of the footprint. The calculated size of the footprint is written to yet another text output file (“gs.out”).

This wrapper, originally written for one purpose, was modified to perform two other tasks. The original purpose was to automatically conduct the gridsearch for the proof-of-concept. The other two tasks were to conduct Design of Experiments (DOE) runs specified in an input file and to automatically perform the experiments needed for determining the ellipse surface coefficients.

Because it could take days to compete all the required cases, the wrapper is executed from the Unix prompt using the “nohup” command. This command allows the user to logout without terminating the process. When all the Monte Carlo processes are complete, the script sends an email to inform the user. This email not only informs the user that the data is available in the output file, it also includes the start and stop times of the process.

CHAPTER 8

MATHEMATICAL PROBLEM FORMULATION

The largest possible uncertainty extrema are sought that would produce a landing footprint of a given size – while minimizing the cost. The mathematical formulation of this problem begins by expressing the optimization in standard form:

> *Given a set of Monte Carlo mission uncertainties (specified by mean or mode, maximum, minimum, and distribution type), find the extrema ($\pm 3\sigma$ limits) that minimize the cost (that is the cost to the program associated with reducing the uncertainties), subject to the two constraints: (1) that the length of the semi-major axis of the footprint ellipse is equal to a specified value; and (2) that the length of the semi-minor axis of the footprint ellipse is equal to a specified value.*

The problem in this form is a constrained optimization. In this case, the optimization is a minimization and there are two equality constraints. Note that the design variables are the uncertainty extrema ($\pm 3\sigma$ limits); the objective function is cost; and the constraints have to do with the size of the landing ellipse (footprint).

Constrained optimization problems are difficult to solve directly, so a series of mathematical steps is used to reduce the problem to one that can be solved easily. In each step, the problem is made easier, but also more complicated in some way. Figure 28 outlines the steps taken in solving the constrained optimization.

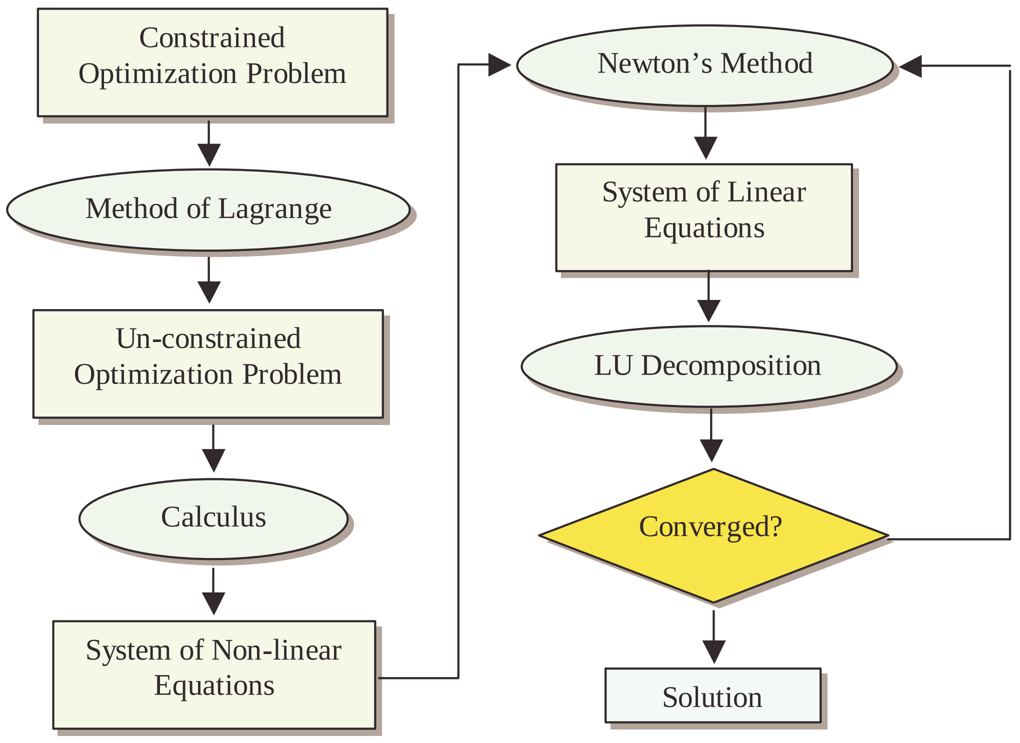

Constrained Optimization Problem
Method of Lagrange
Un-constrained Optimization Problem
Calculus
System of Non-linear Equations
Newton's Method
System of Linear Equations
LU Decomposition
Converged?
Solution

Figure 28: Optimization Methodology

First, the method of Lagrange multipliers is used to convert the constrained optimization problem to an unconstrained optimization. The penalty for this simplification is the addition of more unknown variables (the Lagrange multipliers).

Next, calculus is used to find the optimum solution to the unconstrained problem. For this step, the method of Lagrange is solved with substitution by setting the partial derivative of the objective function equal to a linear combination of the partial derivatives of the constraint functions. This step converts the unconstrained optimization problem into a system of non-linear equations. The complication being that now there is a system of equations instead of just one.

Next, Newton's method (also referred to as a Newton-Raphson iteration) is used to approximate the non-linear system of equations with a system of linear equations. Because this is only an approximation, we are required to repeat this step iteratively until the solution is found. Additionally, initial guesses are required to jump-start the sequence.

Finally, the system of linear equations, is solved using LU decomposition. The solution of the linear algebra problem is used to update the guess from the previous iteration of Newton's method. Newton's method is repeated with the new guess until a stopping a criterion is reached. When this iteration converges, the desired solution to the original constrained optimization problem has been found.

8.1 Constraints

8.1.1 Constraint Function

Because the bulk of the effort is invested in determining the value of the constraint, rather than the objective function, the constraint is central to the problem. In general, the constraints could be any mission requirements that may be deduced from forecast distributions of the Monte Carlo process. However, this research will focus on a very common mission requirement – the size of the landing footprint.

The greatest difficulty with the constraints is that they involve the size of the footprint, which can only be found by running a Monte Carlo analysis. These Monte Carlo analyses require hours of computer time and are, therefore, too expensive to run within a numerical optimization method (particularly one that uses gradients). As a result, a metamodel is created by running a set of experiments and fitting the results to a polynomial model. Two metamodel approaches are described, the response surface and the ellipse surface (so named because its contour lines are elliptical).

8.1.2 Determining Range

A very common constraint in entry trajectories is one on the size of the semi-axes (downrange and cross-range) of the footprint. Depending on the problem, either two axes (ellipse) or one axis (circle) may be used to construct the footprint. When only one axis is used, the miss-distance is calculated from each simulated landing site to the target. When two axes are used, this range must be further broken-down into down-range and cross-range components by taking the dot product with a unit vector in the direction of the nominal azimuth angle.

To determine the miss-distance, the geodetic latitude (L) and longitude (θ) of the touchdown point are compared with the nominal landing point and a range between the two is calculated. Equation 54 shows the calculations involved. The latitude and longitude of the landing point are first converted to x, y, z coordinates in the Geocentric Equatorial (GCE) reference frame assuming an oblate ellipsoid with a polar radius of r_p and an equatorial radius of r_e. [100] The Euclidean (straight-line) distance, R, is then calculated to the nominal landing point.

$$
\begin{aligned}
\varepsilon &= \sqrt{1-\left(\frac{r_p}{r_e}\right)^2} \\
\hat{x} &= \left|\frac{r_e}{\sqrt{1-\varepsilon^2 \sin^2(L)}}\right| \cos(L)\cos(\theta) \\
\hat{y} &= \left|\frac{r_e}{\sqrt{1-\varepsilon^2 \sin^2(L)}}\right| \cos(L)\sin(\theta) \\
\hat{z} &= \left|\frac{r_e(1-\varepsilon^2)}{\sqrt{1-\varepsilon^2 \sin^2(L)}}\right| \sin(L) \\
R &= \sqrt{(\hat{x}-\hat{x}_0)^2+(\hat{y}-\hat{y}_0)^2+(\hat{z}-\hat{z}_0)^2}
\end{aligned}
\qquad \textbf{(54)}
$$

Because the range calculation involves taking the square root, this choice of metric causes some difficulty in fitting a response surface, as will be shown later. Because the range is also non-negative (≥0), forecast distributions of range are single sided and non-symmetrical, as shown in Figure 29. This means that a normal distribution (which is two sided and symmetrical) may not be a good approximation.

Figure 29 shows a histogram plot of miss-distance from the nominal proof-of-concept Monte Carlo analysis (i.e., the analysis performed using the initial guesses for the uncertainty extrema). The superimposed line is a Rayleigh distribution with the same mean as the sample data. This plot shows a significant number of landings outside the 3-km target. This serves as the baseline for the proof-of-concept.

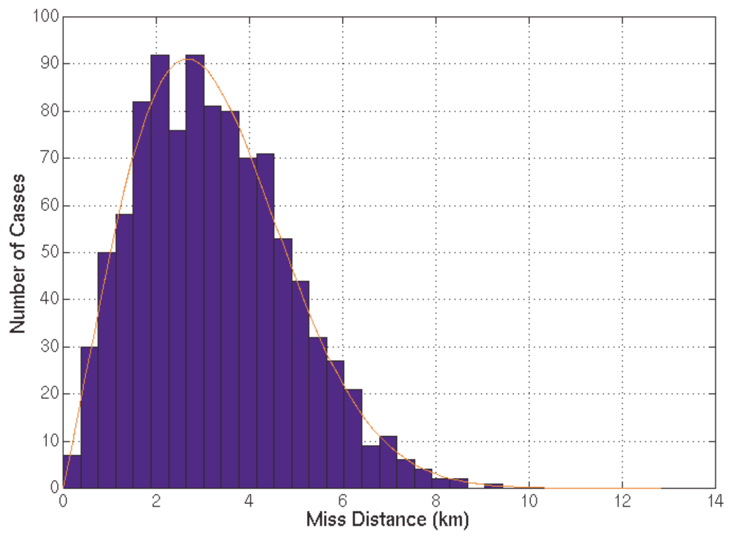


Figure 29: Miss-distance, Proof-of-Concept Baseline.

8.1.3 <u>Response Surface</u>

The most common metamodeling approach in aerospace applications is Response Surface Methodology (RSM) [55, 59, 60, 62, 64, 65]. Simpson et al. [59] suggest that optimization is the principle driver for this, noting that aerospace design applications typically involve computationally intensive analysis and optimization routines.

RSM is based on a statistical approach to rapidly construct empirical metamodels that allow the modeling of a complex system with a simplified equation. [62, 66, 67] RSM typically employs CCD designs from the experimental design literature. [67] These designs efficiently sample the design space. The specific DOE chosen will dictate the number of simulation runs required. The number of runs depends on the number of design variables, the number of levels considered, and the number of interactions modeled. Experiments are then performed at each of the sample points specified in the DOE matrix. The resulting system response data is used to construct a polynomial approximation of the analysis code through a least-squares regression of the data. [65]

The CCD is a five-level fractional factorial design that combines a two-level full factorial with a "star" pattern. [67, 101] Fractional factorial designs reduce the required number of cases by neglecting higher order effects (typically third order or higher). [62]

The star pattern is comprised of center points and axial points. For center points, all the factors are at the midrange value. For axial points, one factor is set at its outer value ($\pm\alpha$) while all other values are set at their midrange. A face-centered design evaluates each variable at only three levels by setting alpha equal to exactly one. Figure 30 is a three-dimensional representation of the sample points from a face-centered central composite design. We see that no fewer than 15 samples must be taken even when the dimensionality of the problem is limited to only three design variables. In general, the CCD requires a minimum of 2^n+2n+1 experiments. As the number of design variables increases, the required number of sample points very quickly increases beyond reasonable limits. Multiple center points are necessary to measure random error in the Monte Carlo process. The use of a random number generator in the analysis leads to non-repeatable results for simulations using the same design variable settings.

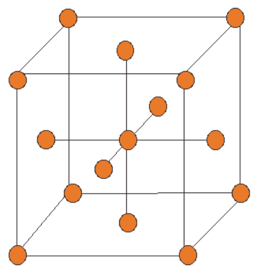


Figure 30: Central Composite Design of Experiments

After running the prescribed cases and collecting the response data, a multivariate regression is performed to estimate the coefficients in the RSE. Generally, the exact relationships between the design variables and the observed responses are either too complex or unknown. [62] Therefore, a simplified empirical model is used to approximate these relationships. Equation 2 shows the empirical model widely used in RSM.

This equation is second order with respect to the design variables (quadratic) and includes two variable interactions. For the two-variable case, six coefficients are determined. In general, $\frac{1}{2}(n^2-n)+2n+1$ coefficients are needed. For example, a problem with 27 design variables would require 406 coefficients. If a CCD design of experiments is used to determine these coefficients, 134217783 experiments (Monte Carlo analyses) would be required. At 2000 simulated trajectories per experiment, that would amount to over 2.68x10^{11} simulations (POST runs). Even at four seconds per simulation, that would take over 34,000 years!

8.1.4 Ellipse Surface

Because response surface metamodeling is not feasible for large design problems, a different approach is needed. The new approach, referred to as an "ellipse surface", is a simplification of the more general response surface. This approach is specific to the problem proposed in this research. In this problem, the contour lines of the miss-distance are expected to be elliptical. If we assume that the contour lines are elliptical then we need only determine the relationship between the semi-major axes and the constant value of the response along the contour line (range). The derivation of this relationship reveals much about the general problem and so it is included here.

Figure 31 provides a visual representation of the metamodel surface for a two-dimensional problem. Similar surfaces in higher dimensions are not able to represented graphically. For this reason, the proof-of-concept problem was restricted to two dimensions.

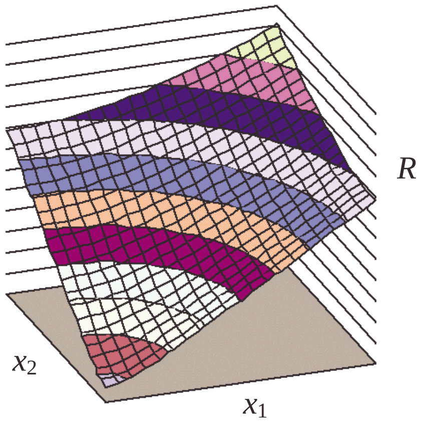


Figure 31: Metamodel Surface Plot.

To determine the relationship between the miss-distance (range) and the size of the elliptical contour lines, results from the best response surface fit of the gridsearch were used. Points from several contour lines were fit with the equation of the ellipse. From this equation, the semi-major axes (*a* and *b*) were determined. These values were then plotted against the range. The plot showed a nearly linear trend.

Assuming that the semi-major axes are linear with the range, a particular polynomial form is derived. Beginning with the equation for an ellipse, Equation 55, substitute linear expressions for the semi-major axes to arrive at Equation 56. The linear assumptions are $a = c_1R$ and $b = c_2R$.

$$\frac{x_1^2}{a^2} + \frac{x_2^2}{b^2} = 1 \tag{55}$$

$$\frac{x_1^2}{(c_1R)^2} + \frac{x_2^2}{(c_2R)^2} = 1 \tag{56}$$

Multiplying both sides by the square of the range (R), results in Equation 57.

$$\frac{x_1^2}{c_1^2} + \frac{x_2^2}{c_2^2} = R^2 \tag{57}$$

Finally, generalize for *n*-variables and add a constant term to account for uncertainty sources not included in the model. This produces Equation 58. The standard RSE, Equation 2, is repeated below for convenience.

$$\tilde{y}^2 = b_0 + \sum_{i=1}^{n} \frac{x_i^2}{c_i^2} \tag{58}$$

$$\tilde{y} = b_0 + \sum_{i=1}^{n} b_i x_i + \sum_{i=1}^{n} b_{ii} x_i^2 + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} b_{ij} x_i x_j \tag{2}$$

Comparing these two equations, note that they are identical in form if the linear terms (b_i) and the two-variable interaction terms (b_{ij}) are neglected from the response surface. Note also that the square of the range appears as the response instead of the range itself. We postulate that the square of the range is a more natural response.

Because Equation 58 includes no two-variable interactions, each coefficient may be determined uniquely by setting all other design variables to zero. Therefore, a "minimum-point" design of experiments may be constructed by evaluating the design space at the origin and each point where all design variables are set to zero with the exception of one, which is set at its baseline value. The metamodel coefficients are determined by Equation 59 and Equation 60.

$$b_0 = R^2, \quad x_i = 0, \quad i = 1 \ldots n \tag{59}$$

$$b_i = \frac{\left(R^2 - b_0\right)}{\left(x_i^0\right)^2}, \; x_j = 0, \quad j = 1 \ldots n, \;\; j \neq i \tag{60}$$

A minimum-point design is one in which the number of design points is exactly equal to the number of coefficients to be fit in the model. [102, 103, 104] These designs require the absolute minimum number of experiments to estimate the model coefficients. [64] This property makes the ellipse surface suitable for large problems (40-50 design variables) since only *n*+1 Monte Carlo analyses must be performed.

Therefore, the ellipse surface is chosen for the metamodel. The equality constraint, h_k, is written as Equation 61, where R_k may be the target size of either the radius (for a circular footprint) or one of the semi-axes (for an elliptical footprint).

$$h_k = b_{0,k} + \sum_{i=1}^{n} b_{i,k} x_i^2 - R_k^2 = 0 \tag{61}$$

8.2 Objective

8.2.1 Objective function

The objective function in this analysis is in many respects arbitrary. This is because its purpose is only to determine the "best" set of design variables from those that are feasible (since many combinations could meet the footprint requirement). The only requirement for this function is that it must somehow measure the "size" or "cost" of changing an uncertainty from its initial value. From the point of view of a decision maker, the "best" alternative (among feasible candidates) is typically the lowest cost. Therefore, a "cost-tolerance" function is proposed. Three models are discussed: the reciprocal model, the minimum-distance model, and the cost-plus-quadratic model.

8.2.2 Reciprocal Model

Discussing the use of quantitative manufacturing knowledge in design and the abstraction of empirical data, Dong [105] states that, "one important abstraction is the relation between design accuracy and production cost." The cost-tolerance models, shown in Table 18, are examples of such relations that have been used by Dong and others in the design and automated manufacturing of mechanical parts. These relations express the production cost, c, as a function of the tolerance, δ, and are based on empirical data from common production processes such as drilling, milling, grinding, and casting. Any of these models may be used as the objective function in the optimization process.

Table 18: Cost Tolerance Models. [105]

Model	Reference	Form
Exponential	Speckhart [106]	$c(\delta)= a_0e^{-a1\delta}$
Reciprocal squared	Spotts [107]	$c(\delta)= a_0\delta^{-2}$
Reciprocal powers (RP)	Sutherland and Roth [108]	$c(\delta)= a_0\delta^{-a1}$
RP and exponential hybrid	Michael and Siddall [109]	$c(\delta)= a_0\delta^{-a1}e^{-a2\delta}$
Reciprocal model	Chase and Greenwood [110]	$c(\delta)= a_0\delta^{-1}$
Modified exponential	Dong and Soom [111]	$c(\delta)= a_0e^{-a1(\delta-a2)}+a_3$
Discrete	Lee and Woo [112]	$c(\delta_k)= \delta_k$ (k=1,2,…,n)
Combined RP and exponential	Dong and Hu [113]	$c(\delta)= a_0+a_1\delta^{-a2}+a_3e^{-a4\delta}$
Linear and exponential	Dong and Hu [113]	$c(\delta)= a_0+a_1\delta+a_2e^{-a3\delta}$
Cubic polynomial	Dong and Hu [113]	$c(\delta)= \sum a_i\delta^i$ (i=1,2,3)
4th order polynomial	Dong and Hu [113]	$c(\delta)= \sum a_i\delta^i$ (i=1,2,…,4)
5th order polynomial	Dong and Hu [113]	$c(\delta)= \sum a_i\delta^i$ (i=1,2,…,5)

When a more detailed definition exists in the later stages of design, actual cost-tolerance data may exist. When less is known in the earlier stages of design, a simple expression is needed. Any of the models in Table 18 may be used - with the exception of the discrete model, which is not differentiable. A summation of reciprocals is proposed, Equation 62.

$$f = \sum_{i=1}^{n} w_i \frac{x_i^0}{x_i} \qquad (62)$$

Figure 32 shows contour lines of the reciprocal cost-tolerance model for a two dimensional problem with equally weighted design variables. This model was used as the objective function in the proof-of-concept problem. As either of the uncertainties is reduced to zero, the cost increases quickly to infinity. The cost is also scaled so that it has a value of unity at the initial guesses (when $x_i = x_i^0$).

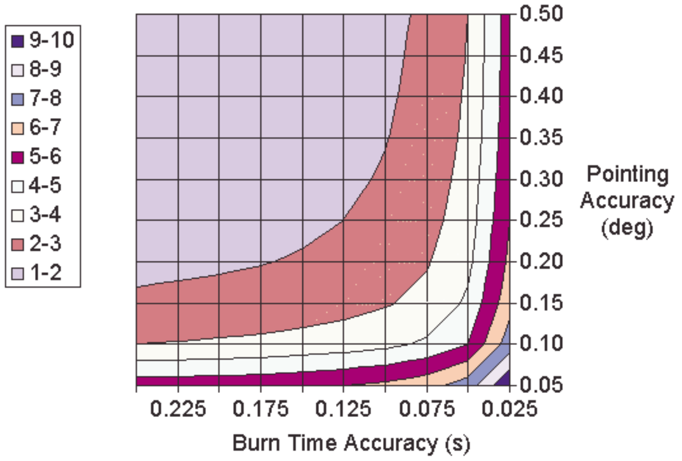


Figure 32: Reciprocal Model Contours.

Advantages of this function include both its simplicity and the stern penalty it levies to reducing any uncertainty to exactly zero. As is the case in nature, the cost goes to infinity as the uncertainty is reduced to zero. Likewise, if all tolerances are allowed to go to infinity, the cost approaches zero. Mathematically, these characteristics prevent non-physical solutions outside the design space (e.g., negative variances). Each term in the summation is weighted and normalized by its initial value. These weights allow the favoring of an inexpensive tolerance over an expensive one and need not necessarily sum to unity.

The primary disadvantage of the reciprocal model is that the minimum of the function (unconstrained) is infinity. Therefore, if some of the uncertainties have a negligible influence on the constraints (i.e., they have small metamodel coefficients – *b* values), then the optimizer will drive their values to infinity. This is because the cost function is essentially unconstrained with respect to those uncertainties. The result is that these uncertainties can be set unreasonably high (to the point that the trajectory is significantly different from the nominal and crashes).

Figure 33 shows the reciprocal cost-tolerance function. Note that the function has been scaled to a value of unity at the initial guess (tolerance=1). Note also that in the absence of an active constraint, the minimum solution is positive infinity. Because of this, care must be taken to ensure that each design variable has a significant effect on the footprint size (either down-range or cross-range).

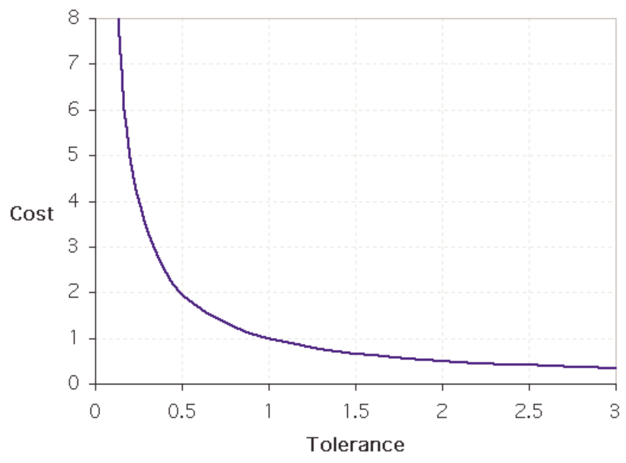


Figure 33: Reciprocal Cost-Tolerance Function.

8.2.3 Minimum-Distance Model

An alternative to the cost-tolerance model is the minimum-distance model. This model minimizes the size of the change from the initial values (i.e., the Euclidean distance is minimized). This model might be particularly appropriate in the later stages of design when any change to the design could be expensive. This function is the sum of the squares of the difference between the uncertainty and its initial value, as shown in Equation 63. Again, The weights allow the favoring of an inexpensive uncertainty over an expensive one and need not necessarily sum to unity.

$$f = \sum_{i=1}^{n} w_i \left[\left(x_i - x_i^0 \right)^2 + 1 \right] \qquad \textbf{(63)}$$

The primary advantage of this method is that the unconstrained minimum of the function occurs at the initial values. This means that as the metamodel coefficients (*b*'s) go to zero, the uncertainties go to their initial values. This ensures that the solution remains close to initial guesses and only small changes in the trajectory are made.

The disadvantage of this method is that the function does not go to infinity as the uncertainties are reduced to zero. This means that physically impossible negative solutions are mathematically possible. Additionally, unlike the reciprocal model, there is no "incentive" to relax any tolerance. When using this method, the initial guesses should be very good and conservative (allowing for some tightening of uncertainties).

Figure 34 shows the minimum-distance function. This function has also been scaled to unity at the initial guesses. This is also the function minimum, so that in the absence of an active constraint, the parameter remains unchanged. Note, however, that the function continues below zero. This means that negative (non-valid) solutions are possible. This function is most applicable in the later stages of design when only minor changes are needed (or desired).

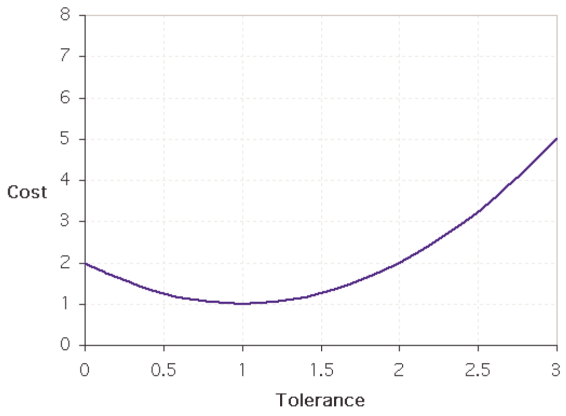


Figure 34: Minimum-Distance Cost-Tolerance Function.

8.2.4 Cost-Plus-Quadratic Model

The final method seeks to combine the advantages of the reciprocal and minimum-distance models, while avoiding their disadvantages. This function, called the "cost-plus-quadratic" function, is a linear combination of the reciprocal function and a quadratic function, shown in Equation 64. Both terms of the function are scaled by the initial values such that they are unity when equal to the initial values.

$$f_{cpq} = \sum_{i=1}^{n} w_i \left\{ \beta \frac{x_i^0}{x_i} + (1-\beta)\left[\frac{x_i}{x_i^0}\right]^2 \right\} \tag{64}$$

The parameter β allows the function to take on any shape between the reciprocal and the quadratic. We choose β such that the minimum of the function occurs at the initial values. To do this, solve for the minimum of Equation 64 by taking the partial derivative, Equation 65, and setting it equal to zero. Then substitute the initial values, $x_i = x_i^0$, shown in Equation 66. Finally, solving for β, the result is shown in Equation 67.

$$\frac{\partial f_{cpq}}{\partial x_i} = w_i \left\{ -\beta x_i^0 x_i^{-2} + 2(1-\beta)\left[\frac{1}{x_i^0}\right]^2 x_i \right\} \tag{65}$$

$$-\beta\left[\frac{1}{x_i^0}\right] + 2(1-\beta)\left[\frac{1}{x_i^0}\right] = 0 \tag{66}$$

$$\beta = \frac{2}{3} \tag{67}$$

The cost-plus-quadratic function, Equation 68, incorporates the best of both worlds since it increases to infinity as the uncertainty is reduced to zero and the minimum occurs at the initial value. Using this function, negative solutions are not possible, and if any of the uncertainties have small or zero metamodel coefficients then there will be little or no change from the initial value.

$$f_{cpq} = \sum_{i=1}^{n} w_i \left\{ \frac{2}{3}\frac{x_i^0}{x_i} + \frac{1}{3}\left[\frac{x_i}{x_i^0}\right]^2 \right\} \qquad \textbf{(68)}$$

This last statement is significant because when using the reciprocal function, the designer must know which uncertainties are important and which are not to avoid divide by zero errors. When using the minimum-distance model or the cost-plus-quadratic model, however, any uncertainty can be included without worry of a divide-by-zero (even if that uncertainty has no effect on the size of the footprint). All of the design variables may be retained in the optimization. Those variables with zero coefficients (*b*'s) simply remain un-changed from their initial values.

Figure 35 shows the cost-plus-quadratic (cpq) function. This function combines the desirable characteristics of the reciprocal model and the minimum-distance function. Once again, the function is scaled such that its value is unity at the initial guess. Like the minimum-distance function, the minimum of the cpq function occurs at the initial guesses. This prevents the solution from degenerating in the absence of active constraints. Like the reciprocal model, the function goes to infinity as the uncertainty is reduced to zero. This prevents the possibility of negative solutions. The only disadvantage of using this function is that, like the minimum-distance model, there is no "incentive" to relax any tolerance. Therefore, the cost-plus-quadratic model is chosen for the objective function.

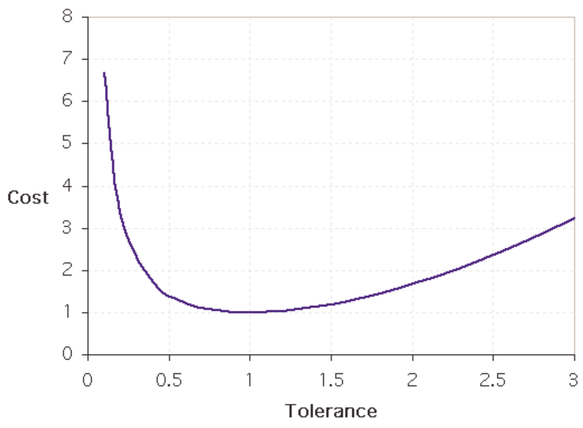


Figure 35: Cost-Plus-Quadratic Cost-Tolerance Function.

8.3 Optimization Problem

Now that the constraint and objective have been chosen, write the optimization problem in standard form, Equation 69. The design variables, x_i, are the uncertainty extrema (±3σ values). The objective function is the cost-plus-quadratic function, which has been normalized by the initial values, x_i^0. Each uncertainty is also weighted with a relative cost value, w_i. The constraints are the ellipse surface metamodels of the semi-major and semi-minor footprint axes. The metamodel coefficients, b_i, are determined by experimentation.

$$\begin{aligned}
&\text{given :} && \bar{x} = \{x_i = 3\sigma_i\} && i = 1, 2, \ldots, n \\
&\text{minimize :} && f(\bar{x}) = \sum_{i=1}^{n} w_i \left\{ \frac{2}{3}\frac{x_i^0}{x_i} + \frac{1}{3}\left[\frac{x_i}{x_i^0}\right]^2 \right\} && \\
&\text{subject to :} && h_k = b_{0,k} + \sum_{i=1}^{n} b_{i,k} x_i^2 - R_k^2 = 0 && k = 1, 2
\end{aligned} \qquad \textbf{(69)}$$

Because both the constraint (in the form of the metamodel) and the objective function are simple polynomials, the derivatives may be determined analytically. In fact, as many derivatives as needed are available. This derivative information can be exploited in the optimization process by using the method of Lagrange multipliers and a Newton-Raphson iteration to solve the constrained optimization. This technique is very efficient, easily programmed, and precisely satisfies the constraints.

8.4 Method of Lagrange Multipliers

The method of Lagrange multipliers is a well-known technique for solving optimization problems with equality constraints. This technique allows classic methods of solving unconstrained optimization problems to be used in constrained optimization problems. The trick is to augment the original objective function with the product of the constraints and unknown constants, λ, known as Lagrange multipliers. The augmented objective function (Lagrangian) is written as Equation 70.

$$\tilde{f}(\bar{x},\bar{\lambda}) = \sum_{i=1}^{n} w_i \left\{ \frac{2}{3}\frac{x_i^0}{x_i} + \frac{1}{3}\left[\frac{x_i}{x_i^0}\right]^2 \right\} + \sum_{k=1}^{\ell} \lambda_k \left\{ b_{0,k} + \sum_{i=1}^{n} b_{i,k} x_i^2 - R_k^2 \right\} \qquad \textbf{(70)}$$

Next, use the method of Lagrange multipliers with substitution by writing Equation 71. This equation enforces tangency of the objective function and constraints at the solution point. This equation equates the partial derivative of the objective function (with respect to the n design variables only) with the partial derivatives of the second term in the Lagrangian.

$$\frac{\partial f_{cpq}}{\partial x_i} = \lambda_0 \frac{\partial h_0}{\partial x_i} + \lambda_1 \frac{\partial h_1}{\partial x_i} \quad i = 1,2,\ldots,n \qquad \textbf{(71)}$$

Then solve Equation 71 for the design variables, x_i, as functions of the Lagrange multipliers, λ_k. Substituting the partial derivatives of the terms in Equation 70 into Equation 71, produces Equation 72. Multiplying both sides of Equation 72 by x_i^2, rearranging, and dividing by two, gives Equation 73. Equation 73 is solved for x_i, resulting in Equation 74.

$$-\frac{2}{3} w_i x_i^0 x_i^{-2} + \frac{2}{3} w_i \left[\frac{1}{x_i^0}\right]^2 x_i = 2 x_i \left(\lambda_0 b_{i,o} + \lambda_1 b_{i,1}\right) \qquad \textbf{(72)}$$

$$-\frac{1}{3} w_i x_i^0 = x_i^{3} \left(\lambda_0 b_{i,o} + \lambda_1 b_{i,1} - \frac{1}{3} w_i \left[\frac{1}{x_i^0}\right]^2 \right) \qquad \textbf{(73)}$$

$$x_i = \left\{ \frac{-\frac{1}{3} w_i x_i^0}{\lambda_0 b_{i,o} + \lambda_1 b_{i,1} - \frac{1}{3} w_i \left[\frac{1}{x_i^0}\right]^2} \right\}^{\frac{1}{3}} \qquad \textbf{(74)}$$

Equation 74 was developed specifically for the cost-plus-quadratic objective function. Similar expressions may be derived for the reciprocal model (Equation 75) and the minimum cost model (Equation 76). These expressions are included here because the choice to use a particular objective function is dependent upon the specific problem (although the cost-plus-quadratic method is recommended for most cases).

$$x_i = \left\{ \frac{-\frac{1}{2} w_i x_i^0}{\lambda_0 b_{i,o} + \lambda_1 b_{i,1}} \right\}^{\frac{1}{3}} \qquad \textbf{(75)}$$

$$x_i = \left\{ \frac{-w_i x_i^0}{\lambda_0 b_{i,o} + \lambda_1 b_{i,1} - w_i} \right\} \qquad \textbf{(76)}$$

Finally, we substitute Equation 74 into the constraint functions, Equation 61, to yield Equation 77. Equation 77 is a 2-by-2 system of non-linear equations, which is a function of only the two Lagrange multipliers λ_0 and λ_1. Once the Lagrange multipliers are known, they are substituted back into Equation 74 to find the optimal design variables.

$$h_k = b_{0,k} + \sum_{i=1}^{n} b_{i,k} \left\{ \frac{-\frac{1}{3} w_i x_i^0}{\lambda_0 b_{i,o} + \lambda_1 b_{i,1} - \frac{1}{3} w_i \left[\frac{1}{x_i^0} \right]^2} \right\}^{\frac{2}{3}} - R_k^2 = 0 \qquad k = 1,2 \qquad \textbf{(77)}$$

8.5 <u>Newton-Raphson Iteration</u>

Newton-Raphson iteration is a numerical technique that reduces a system of simultaneous non-linear equations to a system of simultaneous linear equations. This method expands each of the non-linear equations in a Taylor's series. The higher order terms (those with higher than first derivatives) are neglected. In the case of the solution to the method of Lagrange multipliers with substitution, this approximation leads to the linear system of equations (written in matrix form), Equation 78.

$$\begin{bmatrix} \frac{\partial h_0}{\partial \lambda_0} & \frac{\partial h_0}{\partial \lambda_1} \\ \frac{\partial h_1}{\partial \lambda_0} & \frac{\partial h_1}{\partial \lambda_1} \end{bmatrix} \begin{bmatrix} \Delta\lambda_0 \\ \Delta\lambda_1 \end{bmatrix} = \begin{bmatrix} -h_0 \\ -h_1 \end{bmatrix} \tag{78}$$

Taking the partial derivative of the Equation 77 with respect to the Lagrange multipliers results in Equation 79. This equation is used to construct the entries of the 2-by-2 Jacobian matrix in Equation 78.

$$\frac{\partial h_k}{\partial \lambda_j} = \sum_{i=1}^{n} -2\left(-w_i x_i^0\right)^{\frac{2}{3}} \left\{3\left(\lambda_0 b_{i,o} + \lambda_1 b_{i,1} - \frac{1}{3} w_i \left[\frac{1}{x_i^0}\right]^2\right)\right\}^{-\frac{5}{3}} b_{i,j} b_{i,k} \tag{79}$$

Equation 79 was developed specifically for the cost-plus-quadratic objective function. Similar expressions may be derived for the reciprocal model (Equation 80) and the minimum cost model (Equation 81).

$$\frac{\partial h_k}{\partial \lambda_j} = \sum_{i=1}^{n} -\frac{4}{3}\left(-w_i x_i^0\right)^{\frac{2}{3}} \left\{2\left(\lambda_0 b_{i,o} + \lambda_1 b_{i,1}\right)\right\}^{-\frac{5}{3}} b_{i,j} b_{i,k} \tag{80}$$

$$\frac{\partial h_k}{\partial \lambda_j} = \sum_{i=1}^{n} -2\left(-w_i x_i^0\right)^{2} \left\{\left(\lambda_0 b_{i,o} + \lambda_1 b_{i,1} - w_i\right)\right\}^{-3} b_{i,j} b_{i,k} \tag{81}$$

This system of linear equations, Equation 78, is solved using the efficient PLU decomposition method. The Lagrange multipliers are updated and the process is repeated iteratively until the method converges to a solution. Because Newton's method has quadratic convergence, typically only a few iterations are required.

CHAPTER 9

PROOF-OF-CONCEPT (2D PROBLEM)

9.1 Problem Statement

A proof-of-concept is performed for a simple ballistic entry terminating in by a soft (powered) landing on Mars. The simple test problem was chosen so that it could be exhaustively examined with minimal computational expense. This entry includes no lift, no winds, no parachutes, and no guidance. Because the attracting body in this problem is Mars, the following inputs to POST must be modified: 1) size of the oblate reference ellipsoid, 2) planet rotational rate, 3) gravitational constant, 4) gravity harmonics, 5) atmospheric model, 6) and wind model. Tables of density, pressure, and temperature are used to simulate Mars' atmosphere. Table 19 shows the remaining Mars specific constants used in this example. Table 19 provides POST input parameters specific to Mars. These parameters are used in the attracting body (planet) and gravitational models. A spherical gravity well is assumed (i.e., all gravity harmonics of higher order than J2 are assumed zero). System International (SI) units are used throughout.

Table 19: Mars Constants, POST Input.

Input Name	Value	Units
Gravitational constant	4.2828e+13	m^3/s^2
Equatorial radius	3393.9	km
Polar radius	3376.8	km
Mean radius	3397.2	km
Planet rotation rate	7.0882e-05	rad/sec

This example is designed to represent (very loosely) the entry, descent, and landing (EDL) of a Mars expedition crew using a reusable Mars Transfer Vehicle (MTV). The crew separates from an interplanetary transfer vehicle in a 300-km circular orbit and descends for a landing at the planet’s equator, where a habitat, supplies, and an In-Situ Resource Utilization (ISRU) propellant factory await their arrival. It is assumed that mission requirements dictate a landing within 3 km of the surface assets. Therefore, the primary objective is to successfully locate the minimum cost extrema that satisfy a 3-km landing footprint with 99.87% confidence.

Figure 36 illustrates the simple EDL sequence of the proof-of-concept. Following a de-orbit burn, the vehicle descends ballistically through the atmosphere until reaching a nominal altitude of 38.3 km. From there, the vehicle performs a powered descent to the surface.

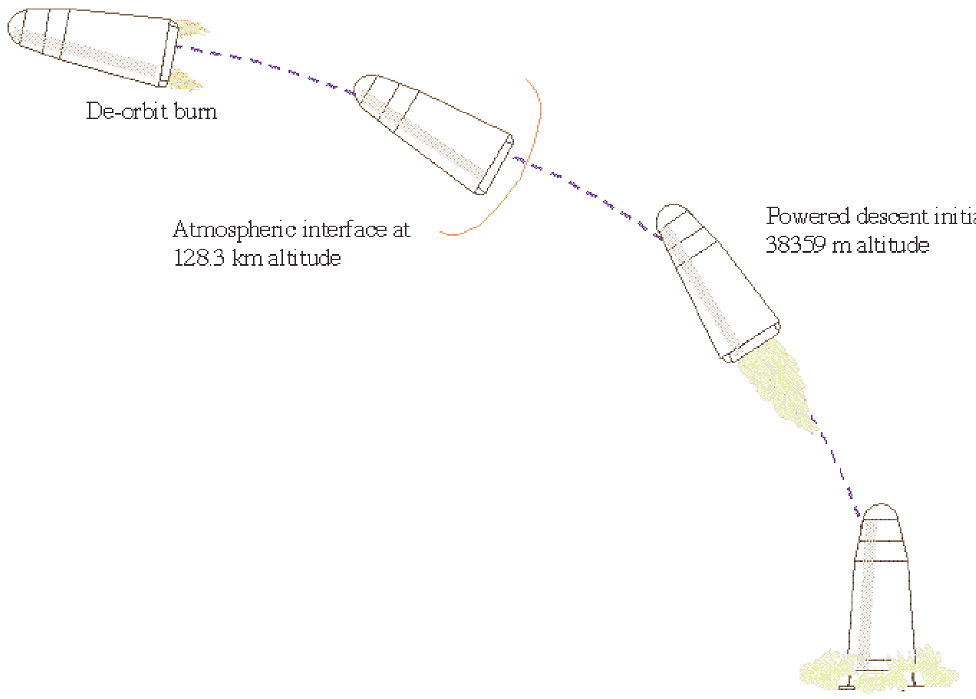

De-orbit burn
Atmospheric interface at 128.3 km altitude
Powered descent initiated at 38359 m altitude

Figure 36: EDL Sequence, Proof-of-Concept.

A previously performed trajectory optimization has determined the nominal controls that minimize the propellant requirements. The trajectory begins in a circular 300-km orbit about Mars at zero degree inclination. The EDL begins with a 108.9 second de-orbit burn, which imparts 1823.8 m/s of ideal velocity change (ΔV). The vehicle descends through atmospheric interface (defined at an altitude of 128.3 km) and at 38.3 km above the reference ellipsoid the descent engines are fired at 100% throttle. The terminal descent burn nominally occurs 505.9 seconds after de-orbit initiation. The MTV continues to decelerate under thrust until a soft vertical landing is accomplished at T+573.1 seconds. The total ideal ΔV for the descent is 3999.6 m/s.

It is supposed that experts have been consulted to arrive at baseline uncertainty definitions. Twelve uncertainties are identified. Table 20 lists these uncertainties along with their mode and dispersion extrema values. All of the uncertainties are assumed to follow normal or Gaussian probability distributions.

Table 20 describes the mission uncertainties used in the proof-of-concept problem. Only three uncertainties (yaw, pitch, and burn-time extrema) were optimized. All other uncertainties were present in the Monte Carlo analysis, but considered uncontrollable. This table is in the form required for "mcp" input.

Table 20: Mission Uncertainties, Proof-of-Concept.

Uncertainty	Mode	3σ Extrema	Distribution	Description
burn_time	108.888099	± 0.25 sec	normal	de-orbit burn time (sec)
power_alt	38359.1014	+5/-0 m	normal	power-on altitude (m)
gross_wt	1.10E+06	± 100 kg	normal	gross weight (kg)
Isp	453.6	± 0.5 sec	normal	specific impulse (sec)
density	1	± 0.5 %	normal	density multiplier
pressure	1	± 0.5 %	normal	pressure multiplier
temperature	1	± 0.5 %	normal	temperature multiplier
thrust_vac	6.40E+05	± 10 N	normal	vacuum thrust (N)
yaw	270	± 0.5 deg	normal	initial yaw (deg)
pitch	0	± 0.5 deg	normal	initial pitch (deg)
drag	1.2	± 0.5 %	normal	drag coefficient
shutdown	0.25	± 0.25 m	normal	engine shutdown alt. (m)

9.2 Results

9.2.1 Screening

The proof-of-concept is limited to only two design variables to facilitate visualization of the design space. Because of this, a One-Variable-At-A-Time (OVAAT) screening is performed to determine the two variables that have the largest impact on the landing footprint. Each of the twelve uncertainty variables is set independently at its respective maximum (+3σ) and minimum (–3σ) value. Deltas between the nominal and contingency trajectories are collected and presented in a Pareto chart. The results of the OVAAT analysis show that the three uncertainty variables that generate the largest deltas in range are pitch, yaw, and burn-time. Because pitch and yaw may be treated as a single uncertainty (pointing accuracy), the two design variables chosen are the dispersion extrema for pointing accuracy and burn-time.

9.2.2 Sample Size Determination

The next step was to determine the sample size for use in the Monte Carlo analysis. This determination involves a trade-off between accuracy and the time required to complete a large gridsearch. Obviously, the larger the sample size the better the accuracy of parameter estimates and the larger the computational expense. It is common practice to run 2000 to 5000 cases per Monte Carlo analysis.

Since many Monte Carlo analyses (221) would be needed for a reference grid search, a low sample size was desired. The arbitrary size of the grid search was chosen as a compromise between the conflicting goals of maximizing the resolution and minimizing the computational time. A baseline sample size of 1000 function evaluations was decided upon. Based upon this number and an average speed of 1500 to 1800 POST runs per hour, the expected time to complete a 221-point gridsearch was approximately 5-6 days. This average speed is applicable to a dual-processor SGI Octane Workstation and depends on system usage. An effort was made to conduct the majority of the runs at night and on weekends to minimize the impact on other users.

9.2.3 Gridsearch

Once a sample size was determined, a gridsearch was initiated to provide a visualization of the design space. The results from the gridsearch are also used in assessing the fit of the metamodels. For practical problems, a gridsearch is not conducted because of the computational expense involved.

The gridsearch was performed in two stages consisting of a 121-point gridsearch and a 100-point gridsearch. Combined, these two searches involved 221 Monte Carlo analyses and 221,000 contingency (off-nominal) trajectory simulations and required 123.1 hours (or 5.1 days) to complete. The average speed achieved on a dual-processor R12000 SGI Octane workstation with 640 MB of RAM (running five simulations at a time in parallel) was 1795 POST simulations per hour.

The contour plot of the gridsearch data, Figure 37, shows the concentric elliptical contour lines expected. Figure 37 shows constant radius contour lines constructed from the gridsearch data. Note that the contours resemble one quadrant of concentric ellipses. This observation forms the motivation for the elliptical approximation of the design space. Note also the nearly equal spacing between contours along the axes. This observation leads to the assumption that the size of the ellipses is related linearly to the value of the contour line. Finally, note the irregularity (or roughness) of the lines caused by the random behavior (non-repeatability) of the Monte Carlo process. Quite a bit of variation or irregularity can be seen due to the non-repeatability of the Monte Carlo analysis. The inner edges of the surface plot (where one dispersion is zero) are nearly linear. The outer edges (where one dispersion is at its maximum) show a parabolic curvature. These observations will become important in choosing the form of the metamodel to use.

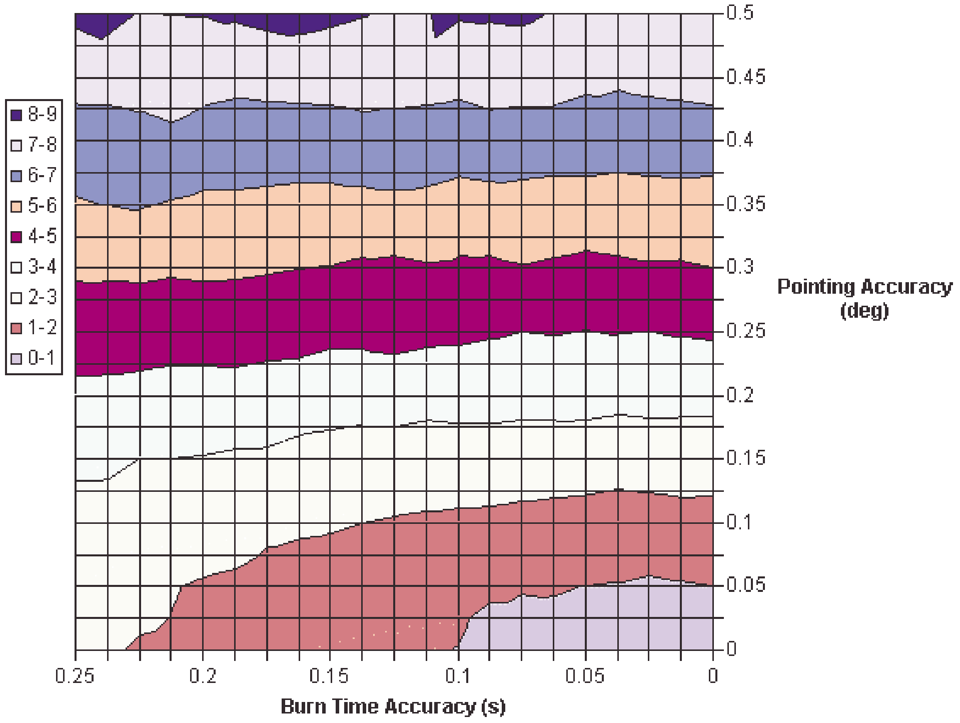


Figure 37: Contour Plot, Gridsearch.

9.2.4 <u>Design of Experiments</u>

A 14-point face-centered Central Composite Design (CCD) for two variables with six center points is used for this problem. For the two-variable case, there are four corner points and four face-centered star points. Because the design is face centered, the design variables are evaluated at only three levels. Most computer analyses are deterministic and therefore do not require the evaluation of multiple center points. When running Monte Carlo analyses, however, multiple center points are necessary to evaluate the experimental error. This Design of Experiments (DOE) is used for generating a Response Surface Equation (RSE). Experiments were determined using a commercial software package (JMP). Table 21 lists the CCD design with the resultant range values. Table 21 is the face-centered central composite design of experiments for the proof-of-concept. Each experiment consists of running a 1000 simulation Monte Carlo analysis with the uncertainty extrema indicated. The $\mu+3\sigma$ range (R) is recorded for each experiment. Note that six multiple center points result in six different ranges from 3.99

km to 4.32 km. This is due to the randomness inherent in the Monte Carlo process. Where possible range values were taken directly from the gridsearch results so that additional Monte Carlo analyses were not necessary. Only five additional center points were evaluated. Had all fourteen Monte Carlo analyses been performed, nearly eight hours would have been expected for completion.

Table 21: Design of Experiments Table, Proof-of-Concept.

Experiment	Pointing Accuracy (x_1)	Burn Time Accuracy (x_2)	3-sigma Range (R)
1	0	0	0.59518
2	0	0.25	2.24288
3	0.5	0	7.86088
4	0.5	0.25	8.22863
5	0	0.125	1.13872
6	0.5	0.125	7.89053
7	0.25	0	4.10832
8	0.25	0.25	4.58441
9	0.25	0.125	4.31660
10	0.25	0.125	4.18025
11	0.25	0.125	4.24036
12	0.25	0.125	4.02583
13	0.25	0.125	3.98749
14	0.25	0.125	4.17729

9.2.5 Metamodel Generation

Once the DOE table is completed, the next step is to generate the response surface. A least-squares regression of the 14 responses in Table 21 is used to determine the six coefficients in Equation 2. A commercial statistical software package, JMP, was used for this purpose. Once the RSE is calculated, the validity of the model must be assessed. Several indicators are used to assess the fit of the response surface: 1) visual comparison with the gridsearch results; 2) numerical computation of the Root Mean Squared (RMS) residual and the R-square parameter; 3) plotting the gridsearch results against the predicted values; and 4) observing the residual plot. Equation 2, is repeated below for convenience.

$$\tilde{y} = b_0 + \sum_{i=1}^{n} b_i x_i + \sum_{i=1}^{n} b_{ii} x_i^2 + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} b_{ij} x_i x_j \tag{2}$$

Three response surface methods are generated for comparison. The first is a standard quadratic RSE (designated by "response surface"). The second is a simplified polynomial designed to produce elliptical contour lines (the "ellipse surface"). Three points uniquely determine the ellipse surface and so no regression is required. The first three experiments in Table 21 are used for this purpose. The third method, called the "squared surface", is another standard RSE, but fit to the square of the range (R^2) instead of the range (R). This method was inspired by the occurrence of R^2 in the derivation of the ellipse surface, Equation 57. All three methods are also compared with a "best" curve-fit, which is a regression of all 226 data points resulting from the gridsearch and the five additional center point evaluations. These "best" fits bring the total number of different response surface methods up to six. The coefficients determined for each of the six metamodel methods are shown in Table 22. The linear and cross term coefficients for the ellipse methods are zero by design.

Table 22: Curve Fit Coefficients, Proof-of-Concept.

Method	b_0	b_1	b_2	b_{11}	b_{22}	b_{12}
Response Surface	0.6583	11.639	2.6220	5.9521	13.040	-10.240
Best Response Surface	0.3080	12.875	3.6961	6.2729	8.6015	-12.424
Ellipse Surface	0.5952	0.0000	0.0000	244.79	70.965	0.0000
Best Ellipse Surface	0.1852	0.0000	0.0000	255.71	71.289	0.0000
Squared Surface	0.6822	5.0958	-11.647	234.10	115.23	9.9243
Best Squared Surface	0.1914	0.8331	-1.3821	257.16	81.968	-12.616

The Root Mean Square (RMS) residual is calculated for each of the six methods based on the entire 226-point data set. The RMS measures how closely the model fits the data points, the lower the RMS the better the fit. The sum of the squares of the residuals and the sum of the mean error are used to calculate R-square (also based on all 226 points). R-square measures the percentage of the variation from the mean assignable to the model. The remaining percentage is attributed to a random error with zero mean. An R-square of unity indicates a perfect model fit. Table 23 compares the RMS and R-square scores of the six response surfaces. Table 23 compares two goodness-of-fit indicators (RMS and R-square) for the six metamodel methods. The "best" methods, which use all 226 gridsearch experiments in the regression, are provided for illustration only. The response surface method shows the poorest results.

Table 23: Goodness of Fit Parameters, Proof-of-Concept.

Method	Experiments	RMS	R-square
Response Surface	14	0.23837	0.98645
Best Response Surface	226	0.15104	0.99534
Ellipse Surface	3	0.14209	0.99551
Best Ellipse Surface	226	0.09819	0.99804
Squared Surface	14	0.13638	0.99597
Best Squared Surface	226	0.09592	0.99813

Residual plots show the error (residual) as a function of the response. Ideally under the assumption that the only error is from random error, there should be no discernable pattern to the residuals. However, residual plots from all six response surfaces show distinctive patterns. Figure 38 and Figure 39 show a prediction profile for the response surface and ellipse surface respectively. Figure 38 plots the actual range determined from the gridsearch against the predicted values from the response surface metamodel. The response surface is a very general metamodel which has been applied to many situations. Ideally, these points would all fall along the x=y line. The degree to which these points differ from the x=y line indicates the fit (and appropriateness) of the model. While the results of this plot appear optimistic, much better results will be shown for other metamodels.

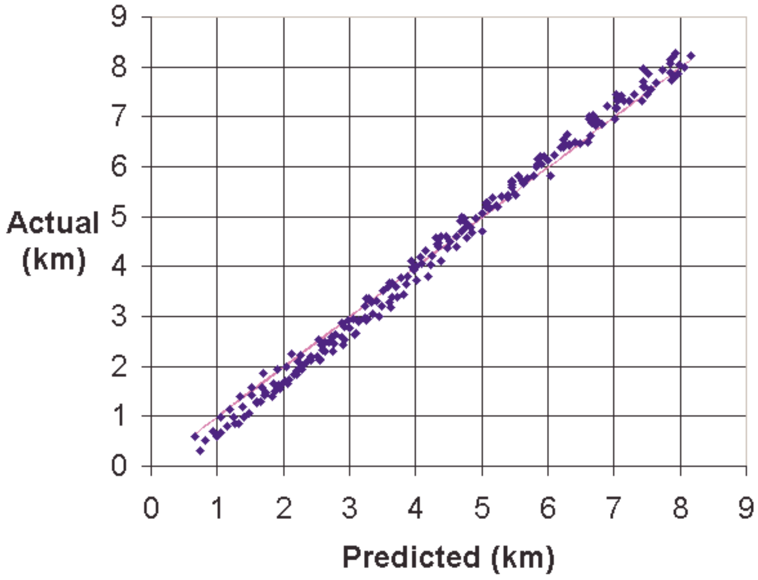


Figure 38: Prediction Profile, Response Surface.

Figure 39 plots the actual range determined from the gridsearch against the predicted values from the ellipse surface metamodel. This figure is compared with Figure 38 for the response surface. While this model is much less general than the response surface, and requires restrictive assumptions about the shape of the design space, the fit is shown to be much improved.

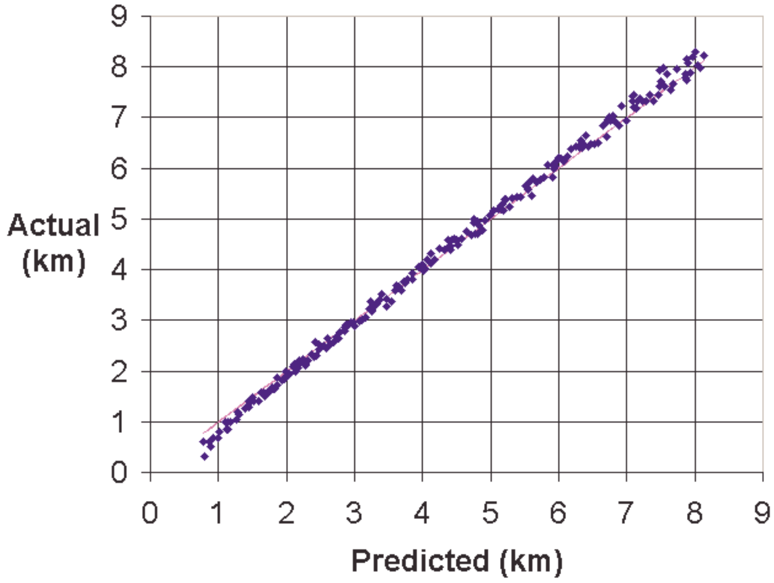


Figure 39: Prediction Profile, Ellipse Surface.

9.2.6 <u>Numerical Optimization</u>

The next step is to solve the constrained optimization problem. A Newton-Raphson iteration is used to solve the optimization problem given by Equation 82 and Equation 83. This iterative method is very efficient and the solution is quickly found to very small tolerances within only a few iterations (less than 50). Equal weights are used in the reciprocal cost function for the two design variables (w_1=w_2=0.5).

$$\begin{aligned}
&\text{given :} && \bar{\mu} = \{\mu_i\}, \\
& && \bar{x} = \{x_i = 3\sigma_i\},\ i = 1, 2, \ldots, n \\
&\text{minimize :} && f(\bar{x}) = \sum_{i=1}^{n} w_i \left(\frac{x_i^0}{x_i} \right) \\
&\text{subject to :} && h = b_0 + \sum_{i=1}^{n} b_i x_i + \sum_{i=1}^{n} b_{ii} x_i^2 + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} b_{ij} x_i x_j - R^2 = 0 \\
&\quad\text{(or)} && h = b_0 + \sum_{i=1}^{n} \frac{x_i^2}{c_i^2} - R^2 = 0
\end{aligned} \qquad (82)$$

$$\tilde{f}(\bar{x}, \lambda) = \sum_{i=1}^{n} w_i \left(\frac{x_i^0}{x_i} \right) + \lambda \left[b_0 + \sum_{i=1}^{n} b_i x_i + \sum_{i=1}^{n} b_{ii} x_i^2 + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} b_{ij} x_i x_j - R^2 \right] \qquad (83)$$

As an independent check on the Newton-Raphson solver, an expected optimum is determined graphically. Since the response surfaces are algebraic equations, we can solve for the value of one design variable along the desired (3-km) contour line as a function of the other. We then plot the cost function along this contour line to show the region of the minimum cost solution. This plot, Figure 40, shows that the expected optimum for the best squared surface is near [0.157, 0.19]. This method (best squared surface) was chosen because it had the highest R-square value and lowest RMS. Figure 40 is a busy plot, which illustrates the effect of altering the uncertainty weightings in the cost-tolerance model. The left hand vertical axis plots the target 3-km contour line. This line plots the set of feasible solutions (i.e., combinations of x_1 and x_2 which meet the 3-km constraint). The other vertical axis plots the value of the objective function (cost) for two settings of the weights. The first of these curves is for an equal 50/50 weighting of the two design variables. The second is for an unequal 80/20 weighting where the first variable, pointing accuracy, is determined to be more expensive. Matching the minimums in these two curves with the contour line constraint, shows that the constrained optimum has indeed changed to favor the reduction of the less expensive variable, burn time accuracy, and the relaxation of the more expensive variable, pointing accuracy. Note also that the new solution results in a higher minimum cost. The best squared surface was used for this example because it has the best RMS and R-square values. Figure 40 also demonstrates how the solution changes with different weighting.

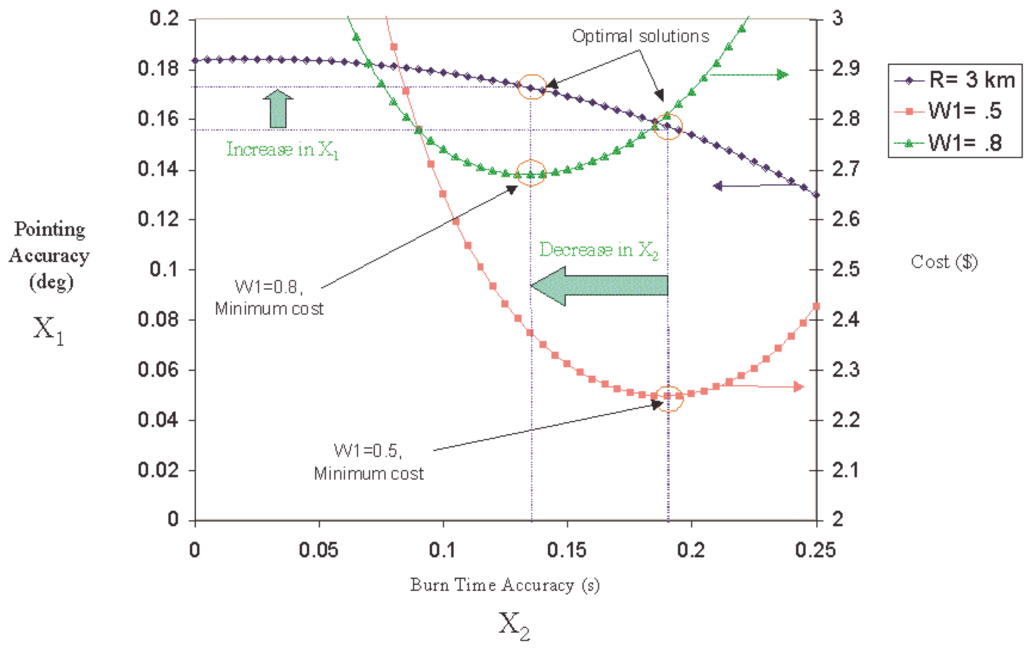


Figure 40: Effect of Weighting Values.

The actual solution points are listed in Table 24 along with the value of the relative cost function for each method. Table 24 compares the optimized solutions for the six metamodel methods along with the cost value at the solution. The optimum for the best squared surface is indeed very close to [0.157, 0.19], which provides confidence in the solution technique.

Table 24: Optimized Solutions, Proof-of-Concept.

Method	x_1	x_2	Cost
Response Surface	0.14748	0.15694	2.49160
Best Response Surface	0.15281	0.17503	2.35019
Ellipse Surface	0.15567	0.18668	2.27561
Best Ellipse Surface	0.15627	0.18987	2.25810
Squared Surface	0.15586	0.17745	2.30837
Best Squared Surface	0.15758	0.18903	2.24780

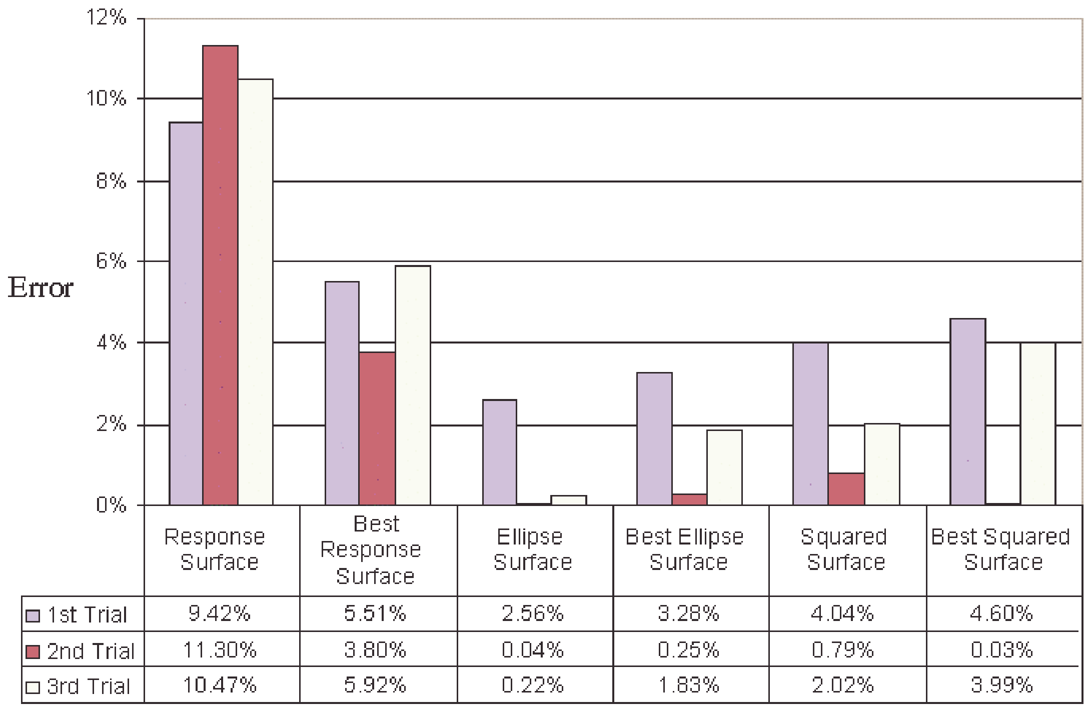
9.2.7 Verification and Comparison of Solutions

Finally, the solutions are verified by running Monte Carlo simulations with the optimum extrema determined by each of the methods and measuring the size of the 3-sigma footprint. The actual footprint is compared to the 3-km footprint predicted by each of the metamodels. Because of the non-repeatability of Monte Carlo analyses, the solutions are compared based on multiple verification runs. Each solution was verified three times and the average error calculated in terms of deviations from the 3-km constraint. Table 25 lists the three trials for each of the six response surface methods and their average errors. Table 25 shows the average relative error from three validation runs at the optimized solution points. Results differ between trials because of the randomness in the Monte Carlo analysis. The response surface methods show unacceptably high errors. This data is also shown graphically in Figure 41.

Table 25: Solution Validation Runs, Proof-of-Concept.

Method	1st R	2nd R	3rd R	Avg. error
Response Surface	2.71744	2.66109	2.68581	10.40 %
Best Response Surface	2.83469	2.88586	2.82236	5.08 %
Ellipse Surface	3.07690	2.99880	3.00663	0.94 %
Best Ellipse Surface	2.90168	2.99235	2.94513	1.79 %
Squared Surface	2.87892	2.97629	2.93933	2.28 %
Best Squared Surface	2.86201	3.00088	2.88020	2.87 %

Figure 41 compares the solutions generated from the different metamodels. Three verification Monte Carlo analyses were conducted at each of the predicted solution points. Since each model predicted a footprint of exactly 3 km, a relative error was generated using the actual footprint range from the verifications. The response surface showed unacceptably large errors (nearly 10%) while the ellipse surface consistently showed the lowest error in each trial. Additionally, the ellipse surface error was less than 3% in all cases. This is fortunate since the ellipse surface is the easiest to generate (requires the fewest number of sample points).



	Response Surface	Best Response Surface	Ellipse Surface	Best Ellipse Surface	Squared Surface	Best Squared Surface
▨ 1st Trial	9.42%	5.51%	2.56%	3.28%	4.04%	4.60%
🟥 2nd Trial	11.30%	3.80%	0.04%	0.25%	0.79%	0.03%
□ 3rd Trial	10.47%	5.92%	0.22%	1.83%	2.02%	3.99%

Figure 41: Metamodel Prediction Error, Proof-of-Concept.

Figure 42 compares the proof-of-concept solutions found using the different metamodels. The solution points are plotted against the contour lines from the gridsearch. Note that three of the solutions are closely grouped, two more of the solutions are closely grouped, and one solution is far removed from the others.

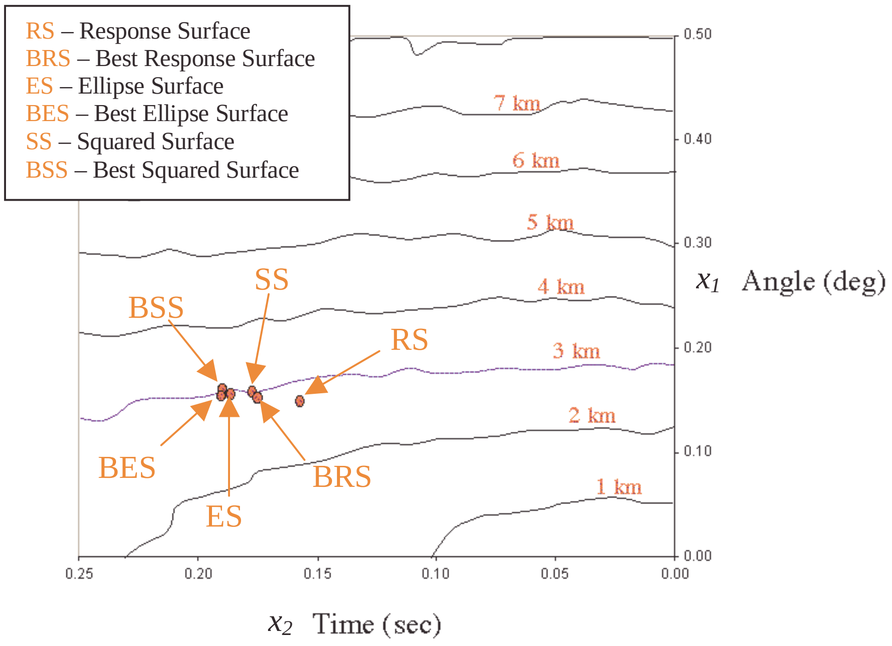


Figure 42: Metamodel Solutions, Proof-of-Concept.

The two response surface methods, which were the only methods to not fit R^2, show unacceptable errors. Comparison of this method with the squared surface indicate a factor of four decrease in average error associated with fitting R^2. It is interesting to note that the 3-point ellipse method showed the lowest average error of the six methods for this problem. A plot of all six solution points shows that three of the methods (ones with the best fits in terms of R-squared values) are grouped closely with an average error of about 2%. Two other methods (the squared response surface and the best response surface) are grouped closely with a larger average error of around 4%. The standard response surface solution stands alone from the other five with an error of over 10%.

The proof-of-concept has shown generally that an acceptable metamodel of the design space can be constructed. This problem further showed that the ellipse surface metamodel, specifically, produces a sufficient model of the design space. In fact, the ellipse surface provided a better approximation than the more complicated response surface, in this example. With the applicability of the ellipse surface assumptions justified by the excellent results, this method may now be applied with confidence to larger problems with more design variables. This is significant since, of the metamodel methods compared, only the ellipse surface requires few enough sample points to feasibly solve problems with more than just a few uncertainties. This point will be demonstrated in Chapter 10 on a “real world” problem involving 27 design variables. This problem (because of its size) can not be solved in a reasonable time using a standard response surface.

CHAPTER 10

"REAL WORLD" EXAMPLE

The validity of the proposed methodology was demonstrated by the proof-of-concept example. However, an additional “real world” example was necessary to demonstrate the implementation of the method on a more complex problem. For this example, it was important to use a problem for which Monte Carlo analyses had been performed and documented. Preferably, this problem would include a lifting entry and a guidance algorithm. The reason for this was to demonstrate the method on a difficult problem that might contain non-linear relationships and correlations between uncertainties. Additionally, a problem with more than twenty uncertainties was sought so that the method could be demonstrated on a problem that was too large to be practically solved by other means (e.g., gridsearch or response surface).

The problem chosen for the example was the Mars Surveyor Program (MSP) 2001 Lander. An Atmospheric Flight Team (AFT) was formed by the MSP ’01 program office with the task of developing aerocapture and precision landing test-bed simulations and candidate guidance algorithms. The AFT was composed primarily of personnel from the Jet Propulsion Laboratory (JPL), Johnson Space Center (JSC), Langley Research Center (LaRC), and Lockheed-Martin Astronautics (LMA). Both 3- and 6-DOF flight simulations were developed. Several papers were published on the results of this study. [19, 20]

10.1 MSP 2001 Lander

Had the program not been cancelled, Mars Surveyor 2001 would have been the first lander on another planet to use a guidance algorithm to actively control its entry. [39]

This ability to make a precision landing would have allowed the spacecraft to land safely within a 10-km circle (an order of magnitude improvement over the 100-by-200 km landing zone for Mars Pathfinder).

Autonomous control of the vehicle's atmospheric flight path is required for precision landings. Trajectory corrections are performed by orienting the vehicle's lift vector in the appropriate directions. An atmospheric guidance algorithm receives navigational information from on-board sensors, determines the required trajectory correction, and commands the control system to orient the vehicle to the proper attitude.

Calspace [40] describes the Entry, Descent, and Landing (EDL) sequence for this mission as follows. The MSP '01 Lander enters Mars directly from its interplanetary transfer orbit (i.e., it does not insert first into a Martian orbit). Five minutes before atmospheric entry, the cruise stage is jettisoned.

At entry into the Martian atmosphere, the lander is housed within a 2.65 meter diameter 70 degree sphere-cone aeroshell similar to the Viking configuration. The entry vehicle uses aeromaneuvering to deliver the lander to an acceptable parachute deployment condition. The guidance algorithm deploys a super-sonic, 13 meter diameter, ballistic, disk-gap-band parachute approximately 226 seconds after entry (and at an altitude of 9- to 10-km). Ten seconds after parachute deployment the aeroshell is released. The parachute decelerates the lander to approximately 80 m/s.

At a radar altimeter altitude of 1.43 km, the lander legs deploy and the powered descent begins. The parachute and backshell are released two seconds after descent engine ignition. The Lander touches down about 37 seconds later at a soft 2.5 m/s velocity. Figure 43 shows an artist's rendering of the lander and rover on the surface of Mars.

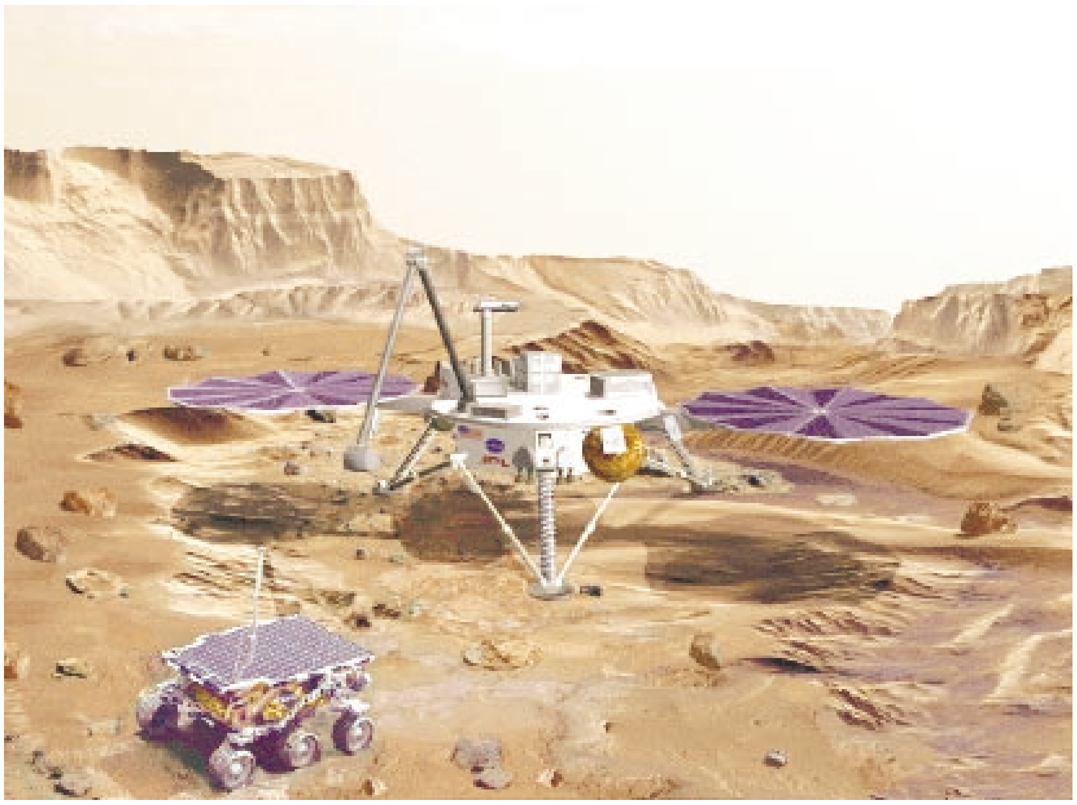


Figure 43: MSP 2001 Lander.

10.2 Monte Carlo Simulation

Several detailed computer models are needed to construct the simulation. These models include: gravitational attraction, central attracting body (planet), atmospheric properties, vehicle aerodynamics, inertial measurement unit (IMU), guidance, and mass properties. Several of these models are discussed here.

10.2.1 Guidance/IMU

The guidance algorithm used in this problem was based on the Apollo Earth entry algorithm. Carman et al. [33] describe the modifications made for Mars entry and present Monte Carlo results for the MSP ’01 Lander. This algorithm was compiled and linked with the POST source code.

POST also included an IMU mode. The most important input for this model is the initial position and velocity knowledge error. Actual and navigation state estimates for the entry were provided by the project office. This entry state (at a reference 3522.2-km radius) had a mean inertial flight path angle of –14.5 deg (with a maximum variation of ±0.23 deg), an inertial velocity of 6973 m/s (±29 m/s), and an azimuth angle of 101.56 deg (±0.09 deg).

10.2.2 Gravity/planet model

The Mars gravitational model is based on one used in the Artificial Satellite Analysis Program (ASAP) [114]. The model is a 50-by-50 gravity field that uses zonal, sectorial, and tesseral harmonic terms to determine the acceleration due to gravity. Zonal harmonics depend only on the mass distribution that is symmetric about the north-south axis of the planet (they do not depend on longitude). In contrast, sectorial harmonics depend only on longitude. However, tesseral harmonics are a function of both latitude and longitude. Even numbered harmonics are symmetric about the planet's equatorial plane, whereas odd numbered harmonics are not symmetric. [100] AFT members from LaRC added this gravity model to POST as a FORTRAN subroutine.

The planet model used was an oblate spheroid given by the constants in Table 19, which is repeated here for convenience. The mean planet radius is used to calculate orbital altitudes (apoapsis and periapsis).

Table 19: Mars Constants, POST Input.

Input Name	Value	Units
J2	0.0019586	
Gravitational constant	4.2828e+13	m^3/s^2
Equatorial radius	3393.9	km
Polar radius	3376.8	km
Mean radius	3397.2	km
Planet rotation rate	7.0882e-05	rad/sec

10.2.3 <u>Atmospheric model</u>

The Mars atmospheric model was provided by the Mars Global Reference Atmosphere Model (Mars-GRAM) version 3.7, which is included as FORTRAN subroutines in POST. Mars-GRAM [115] provides temperature, density, pressure, and wind velocity as well as random perturbations to density. This data is a function of the spacecraft location (latitude, longitude, and altitude) and the Julian date.

Curve fits of "climate factors" were used to interpolate the effects of dust in the atmosphere. These curve fits are a function of the parameter Tau and better emulate the expected atmospheric conditions based on Global Circulation Models (GCM) data. Table 26 shows the climate factor curve fits (taken from Table 5 in [19]). Additional controls of the Mars-GRAM atmosphere include the update distance between calls to the program, the f10.7-cm solar flux value, and the seed value for the density perturbations.

Table 26: Climate Factor Curve Fits. [19]

Climate Factor Constant	Equation
Surface (CF0)	= 1.01290 – 0.0077011 * Tau
5 km (CF5)	= 0.95753 + 0.0314560 * Tau
15 km (CF15)	= 0.94510 + 0.0638120 * Tau
30 km (CF30)	= 0.90674 + 0.1024100 * Tau
50 km (CF50)	= 0.79403 + 0.0795230 * Tau
75 km (CF75)	= 0.85103 + 0.0074796 * Tau
Surface Pressure (CFp)	= 0.56121 + 0.2065100 * Tau

10.2.4 <u>Aerodynamics</u>

The vehicle aerodynamic properties are supplied by a FORTRAN subroutine in POST. This routine smoothly interpolates between discrete solutions in a database. This database contains both free-molecular solutions (based on a velocity of 6975 m/s), for the rarefied region of the atmosphere, and Computational Fluid Dynamic (CFD) solutions, for the continuum regime. A bridging function [116] is used in the transitional region between rarefied and continuum regimes.

The Langley Aerothermodynamic Upwind Relaxation Algorithm (LAURA) [117, 118, 119] generated fifty-two CFD solutions at thirteen points along the trajectory. Extensive validations of LAURA solutions have been made against Viking data, wind tunnel tests, and Mars Pathfinder mission results. [120] LAURA was also used for the Mars Pathfinder [121], Mars Microprobe [116], and Stardust [122] missions.

10.2.5 <u>Mission Uncertainties</u>

Where possible, the mission uncertainties were chosen to conform to those used by Striepe et al. [19] (Table 4 in the reference). Two additional uncertainties were added, the time of entry (Julian date) and the maximum bank rate. The addition of these variables intentionally stressed the problem such that the baseline footprint would be much larger than the 10-km target. This was necessary to provide an academic example that required improvement over the baseline. Table 27 describes the mission uncertainties used in the Mars Surveyor Program 2001 lander problem. Normal, uniform, discrete, and triangular distributions were all used. This table is in the form required for “mcp” input.

Table 27: Mission Uncertainties, MSP '01.

Uncertainty	Mode	3σ Extrema	Distr.	Description
mg_seed	0	1 to 29999	discrete	Mars GRAM pert. seed
atm_flag	0	1 to 1	discrete	0=unperturbed 1=perturbed
Jdate	2452308.5	+/- 0.5 days	uniform	Julian date of entry
update_dist	0.5	+4.5/-0 km	uniform	MG dist. between calls (km)
Bank	0	+/- 5 deg	normal	initial bank angle (deg)
max_rate	20	+/- 1 %	normal	max bank rate (deg/s)
max_accel	1.78	+/- 0.178 deg/s^2	normal	max bank accel (deg/s^2)
Trim	0	+/- 2 deg	normal	trim alpha incr. (deg)
ca_kn	0	+/- 0.10	normal	axial force incr. Kn>.1 (%)
cn_kn	0	+/- 0.10	normal	normal force incr. Kn>.1 (%)
ca_M10	0	+/- 0.03	normal	axial force incr. M>10 (%)
cn_M10	0	+/- 0.05	normal	normal force incr. M>10 (%)
ca_M5	0	+/- 0.10	normal	axial force incr. M<5 (%)
cn_M5	0	+/- 0.08	normal	normal force incr. M<5 (%)
flt_path	-14.5	+/- 0.23 deg	normal	flight path angle (deg)
velocity	6973	+/- 29 m/s	normal	entry velocity (m/s)
azimuth	101.56	+/- 0.09 deg	normal	entry azimuth (deg)
latitude	18.1505349	+/- 0.01 deg	normal	initial latitude (deg)
longitude	250.338677	+/- 0.01 deg	normal	initial longitude (deg)
radius	3522200	+/- 0 m	normal	initial radius (m)
Xcg	0.7155	+/- 0.01 m	normal	axial c.g. location (m)
x_error	0	+/- 5266.7 m	normal	IMU knowledge error (m)
y_error	0	+/- 5266.7 m	normal	IMU knowledge error (m)
z_error	0	+/- 5266.7 m	normal	IMU knowledge error (m)
u_error	0	+/- 2.033 m/s	normal	IMU knowledge error (m/s)
v_error	0	+/- 2.033 m/s	normal	IMU knowledge error (m/s)
w_error	0	+/- 2.033 m/s	normal	IMU knowledge error (m/s)
imu_seed	0	1 to 29999	discrete	IMU seed
Tau	1	+/- 0.7	triangle	climate factor
Angle	0	+/- 1.5708 rad	uniform	c.g. clocking angle (rad)
cg_offset	0.017	+/- 0.005 m	normal	normal c.g. location (m)
Mass	523	+/- 2 kg	normal	initial mass (kg)

Twenty-seven design variables were optimized. Five variables from Table 27 were not optimized: the Mars-GRAM seed, the atmospheric perturbation flag, the Mars-GRAM update distance, the initial radius, and the IMU seed. The initial radius was not varied because the initial state was defined at that altitude as the atmospheric interface.

The seed values, update distance, and perturbation flag are necessary to control the internal random perturbations provided by the atmosphere and IMU models. It is not possible to "optimize" these uncertainties in the same sense as the other uncertainties. Their effect is accounted for in the constant term of the metamodel.

10.3 Baseline Monte Carlo Results

Figure 44 is a plot of the parachute deployment conditions for the nominal MSP '01 Monte Carlo results. The overlaid box indicates representative Mach number and dynamic pressure limits on parachute deployment. Note that only a small percentage of the simulations result in parachute deployment beyond the target limits. It is important for the optimization of the landing footprint to not result in a larger percentage of simulations deploying the parachute outside of the limits.

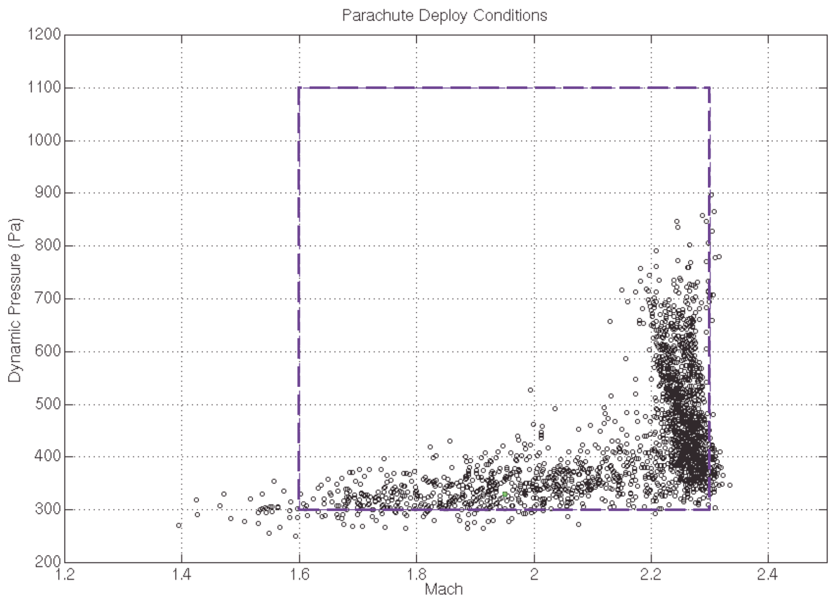


Figure 44: Parachute Deployment, MSP '01 Baseline.

Figure 45 plots the baseline landing footprint for the MSP '01 problem. All of the footprints in this chapter were produced using the bivariate normal (BVN) footprint method for a 99.5% probability. The small overlaid circle in Figure 45 is the target 10-km miss-distance. The footprint ellipse (approximately 118-by-4 km) is much larger than that reported by Striepe et al. [19]. This is due to the added uncertainties, particularly the arrival Julian date, which creates large variations in atmospheric density.

A 10-by-1.41 km target ellipse was used in the optimization process to reduce this footprint such that greater than 99.5% of the simulated landings will miss the nominal target by fewer than 10 km. Note that for the baseline the down-range dispersions are significantly greater than those in the cross-range direction, resulting in a very highly eccentric ellipse. This is the reason for the 1.41-km semi-minor axis target. Solutions at target values greater than this were not possible because of interrelations between the two constraints. While it is always possible to meet a single constraint, this is a disadvantage of using multiple constraints. Note also that the footprint ellipse is not centered on the nominal landing site.

Figure 46 shows the same footprint as Figure 45 only in the down-range and cross-range coordinate system. This system is more convenient to use than latitude-longitude since the ellipse is generated in the desired units, km. Both down-range and cross-range are measured from the nominal landing site. The nominal azimuth angle is needed to properly orient the system axes such that the down-range and cross-range have the usual meanings. This is also necessary to ensure that the 3-sigma down-range and cross-range footprint closely approximates the BVN ellipse axes. If the azimuth angle is chosen properly, the footprint will not be rotated by any angle in this coordinate system. A close examination of both figures show that the BVN footprint method produces the same ellipse in both coordinate systems. This shows that the BVN method is appropriate for use regardless of whether the data is correlated.

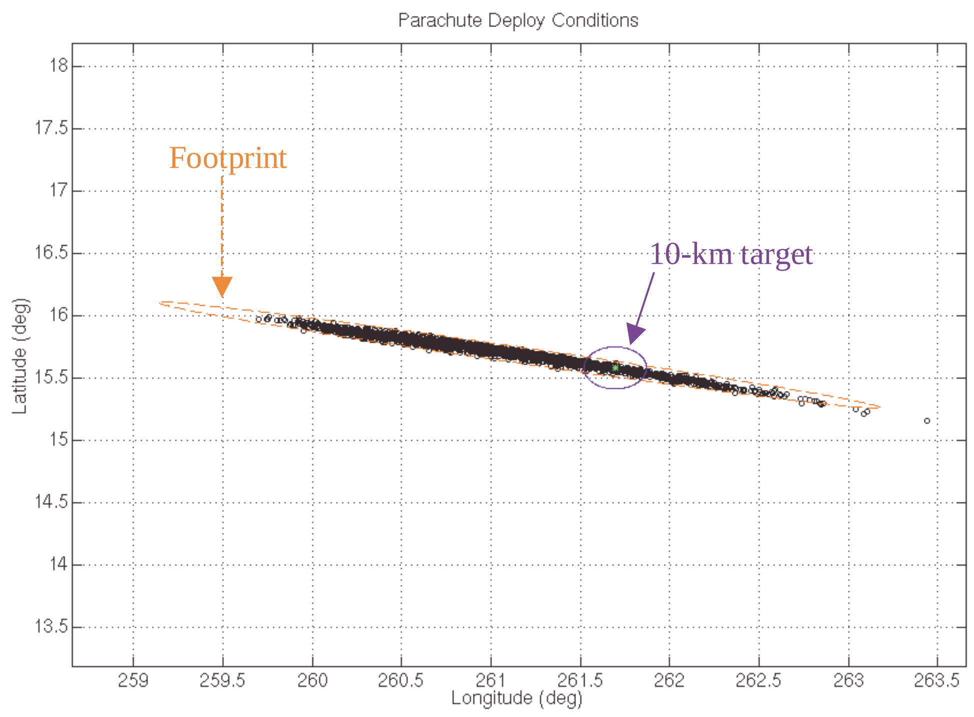


Figure 45: BVN footprint, MSP ’01 Baseline (lat,long).

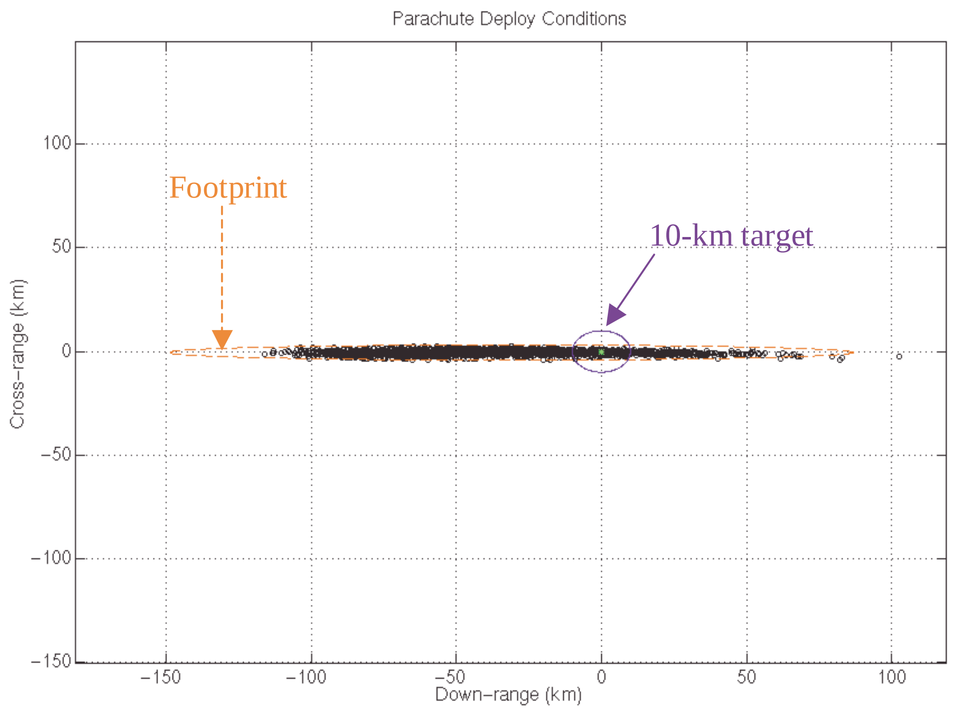


Figure 46: BVN footprint, MSP ’01 Baseline (dr,cr).

Figure 47 shows the empirical miss-distance cumulative distribution function for the baseline MSP ‘01 problem. Note that approximately 80% of the simulations missed the target landing site by more than the allowable 10 km. This formed the motivation for the MSP ‘01 optimization problem. It was desired to find the minimum-cost solution that results in fewer than 0.5% of the simulations missing the target. The cost-plus-quadratic objective function, with equal weightings for all uncertainties, was used.

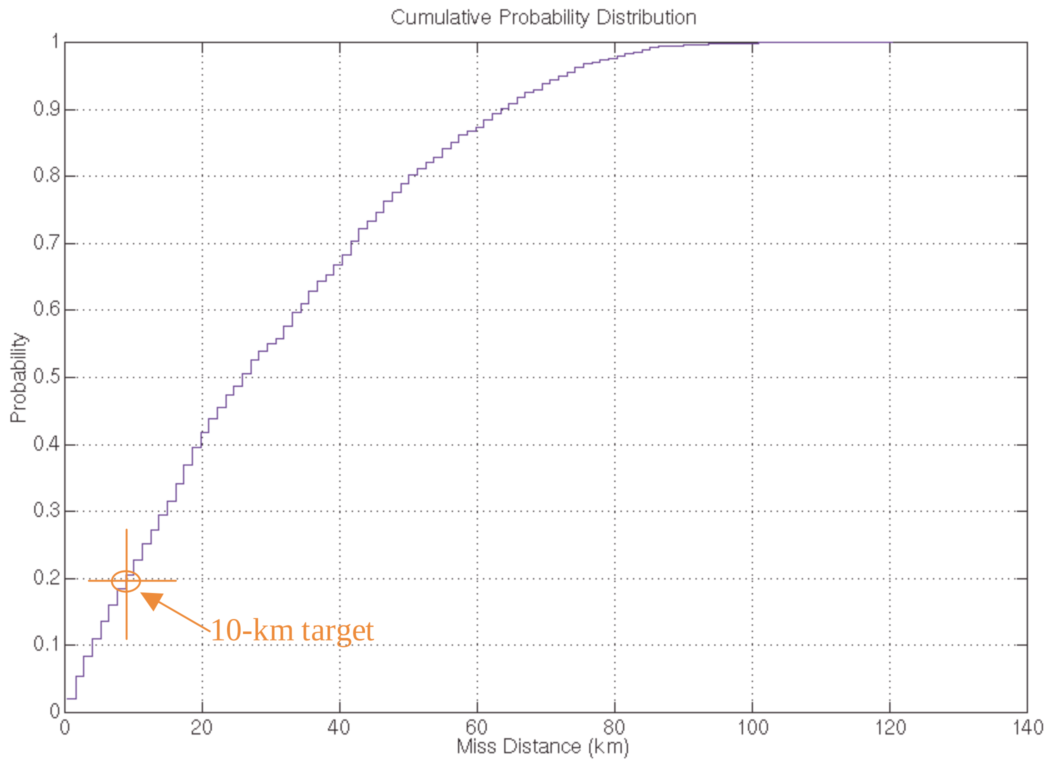


Figure 47: Miss-distance cdf, MSP ’01 Baseline.

10.4 Optimized Solution

Table 28 shows the progression of the design variables through two iterations of the optimization process. These iterations were required to bring the footprint sufficiently close to the target. The fact that it required two iterations demonstrates that (1) multiple iterations may be required to account for non-linearities and uncertainty correlations, and (2) that the iterations converge quickly.

Table 28: Uncertainty Extrema, MSP '01.

	Name	Initial Values	Units	1st Iteration	2nd Iteration
x_1	jdate	0.5	days	0.07241208	0.0479888
x_2	bank	5	deg	5	5
x_3	max_rate	1	%	0.92793034	0.92537808
x_4	max_accel	0.178	g's	0.17297628	0.17286401
x_5	trim	2	deg	0.13692035	0.12213146
x_6	ca_kn	0.1		0.09118567	0.09108389
x_7	cn_kn	0.1		0.1	0.09995603
x_8	ca_M10	0.03		0.00271375	0.00248455
x_9	cn_M10	0.05		0.04274252	0.04247787
x_{10}	ca_M5	0.1		0.0447418	0.04389342
x_{11}	cn_M5	0.08		0.01766993	0.01705527
x_{12}	flt_path	0.23	deg	0.02613341	0.0244741
x_{13}	velocity	29	m/s	15.7618974	15.5818713
x_{14}	azimuth	0.09	deg	0.13061347	0.13331093
x_{15}	latitude	0.01	deg	0.01119022	0.01124481
x_{16}	longitude	0.01	deg	0.00848979	0.00846536
x_{17}	xcg	0.01	m	0.00828815	0.00825664
x_{18}	x_error	5266.7	m	3251.20079	3216.05921
x_{19}	y_error	5266.7	m	933.951039	885.919342
x_{20}	z_error	5266.7	m	1964.79807	1916.18485
x_{21}	u_error	2.033	m/s	2.04675043	2.04263505
x_{22}	v_error	2.033	m/s	1.48836239	1.47950077
x_{23}	w_error	2.033	m/s	2.09662743	2.10088651
x_{24}	cg_offset	0.005	m	0.00045295	0.00041319
x_{25}	mass	2	kg	1.95191636	1.95192831
x_{26}	tau	0.7		0.12637274	0.12015699
x_{27}	angle	1.570795	rad	0.06995173	0.06969548
f	cost	1.00000		4.20878	4.54064

Table 29 compares the BVN footprint ellipses in each iteration. The desired footprint size was a 10-km semi-major axis, *a*, by 1.41-km semi-minor axis, *b*. The small angles, θ, indicate that the nominal azimuth angle was appropriately chosen to define the down-range and cross-range directions. The empirical probability, p_e, shows that the BVN assumption very closely predicted the desired 0.995 probability. The target probability, p_t, is the empirical percentage of simulated landings that fell within the target circle.

Table 29: BVN Footprint Comparisons, MSP '01.

Parameter	Baseline	1st Iteration	2nd Iteration
a (km)	117.86	12.47	10.28
b (km)	3.67	1.36	1.36
θ (deg)	0.0025	0.0037	0.0011
p_e	.9945	.9955	.9960
p_t	.1724	.9925	.9985

Table 30 shows the computational time required to run the experiments for generating the metamodel coefficients. Each set of experiments included 56,000 (28*2000) simulated trajectories. The 1st and 2nd iteration experiments where run on an SGI Origin 2000 Deskside server with eight R10000 processors (VAB17). Ten simulations at a time were run in parallel for approximately 50 hours.

The experiments used for the Perl "rand" random number generator comparison (presented in Chapter 5) where run on an SGI Origin 2000 Rackmount server with sixteen R12000 processors (VAB03). Sixteen simulations at a time where run in parallel, requiring only 5½ hours for completion.

Table 30: Computational Times, MSP '01.

	Machine	Processors	RAM	Time (hrs:min:sec)	Speed (sims/hr)
1st Iteration	VAB17	8 (180 MHz)	2048M	49:56:10	1121.43
2nd Iteration	VAB17	8 (180 MHz)	2048M	52:51:53	1059.31
Perl "rand"	VAB03	16 (400 MHz)	6016M	5:34:54	10032.85

10.4.1 1st Iteration Results

First, ellipse surface metamodel coefficients were determined using the initial values of the uncertainties. These coefficients are shown in Table 31, where b_1 is the coefficient associated with the semi-major axis of the footprint ellipse and b_2 is the coefficient associated with the semi-minor axis of the ellipse. The constant coefficient for the semi-major axis was b_1^0=5.46078. The constant coefficient for the semi-minor axis was b_2^0=0.0107546.

Table 31: Metamodel Coefficients, MSP '01 1st Iteration.

	Name	Initial Values	Units	b_1	b_2
x_1	jdate	0.5	days	1.22836E+03	1.11526E-01
x_2	bank	5	deg	0.00000E+00	0.00000E+00
x_3	max_rate	1	%	2.35353E-01	0.00000E+00
x_4	max_accel	0.178	g's	2.66302E+00	1.37638E-02
x_5	trim	2	deg	7.29601E+02	8.25109E-01
x_6	ca_kn	0.1		2.98396E+01	2.33682E-03
x_7	cn_kn	0.1		0.00000E+00	0.00000E+00
x_8	ca_M10	0.03		1.40361E+06	2.61342E+02
x_9	cn_M10	0.05		2.29073E+02	3.93089E+00
x_{10}	ca_M5	0.1		9.52389E+02	1.29390E+00
x_{11}	cn_M5	0.08		1.34883E+04	6.32736E+01
x_{12}	flt_path	0.23	deg	1.20401E+04	1.56307E+00
x_{13}	velocity	29	m/s	5.82058E-03	4.15855E-06
x_{14}	azimuth	0.09	deg	5.45351E-01	7.23506E+01
x_{15}	latitude	0.01	deg	0.00000E+00	2.47673E+03
x_{16}	longitude	0.01	deg	6.09220E+03	1.46814E+02
x_{17}	xcg	0.01	m	7.09383E+03	1.58738E+01
x_{18}	x_error	5266.7	m	1.12905E-07	3.01172E-09
x_{19}	y_error	5266.7	m	6.02554E-06	1.01381E-08
x_{20}	z_error	5266.7	m	6.42669E-07	2.47658E-08
x_{21}	u_error	2.033	m/s	0.00000E+00	4.18945E-03
x_{22}	v_error	2.033	m/s	3.55481E-01	4.59155E-03
x_{23}	w_error	2.033	m/s	1.41527E-02	3.15644E-02
x_{24}	cg_offset	0.005	m	5.03599E+07	5.44338E+04
x_{25}	mass	2	kg	1.77141E-02	0.00000E+00
x_{26}	tau	0.7		3.22735E+02	1.40048E-01
x_{27}	angle	1.570795	rad	4.30772E+03	1.36806E+01

Table 32 shows a useful normalization of the metamodel coefficients, $x_i^0 b_i^{0.5}$. The units of this coefficient are range, and it represents the contribution to the semi-major or semi-minor axis due to the uncertainty being set at its full initial value. This coefficient can be used to rank the "importance" of each uncertainty – though care should be exercised in interpreting the results since they are dependant on the assumed initial values. This table shows that the top five contributors to the semi-major axis are: c.g. clocking angle (x_{27}), trim angle-of-attack (x_5), axial aerodynamic coefficient at Mach 10 (x_8), c.g. lateral offset (x_{24}), and initial flight path angle (x_{12}).

Table 32: Uncertainty Ranking, MSP '01 1st Iteration.

	Name	x_i^0	Units	$x_i^0 b_1^{0.5}$	$x_i^0 b_2^{0.5}$
x_{27}	angle	1.570795	rad	103.09639	5.80995
x_5	trim	2	deg	54.02224	1.81671
x_8	ca_M10	0.03		35.54221	0.48498
x_{24}	cg_offset	0.005	m	35.48237	1.16655
x_{12}	flt_path	0.23	deg	25.23728	0.28755
x_1	jdate	0.5	days	17.52400	0.16698
x_{19}	y_error	5266.7	m	12.92815	0.53029
x_{26}	tau	0.7		12.57538	0.26196
x_{11}	cn_M5	0.08		9.29113	0.63636
x_{20}	z_error	5266.7	m	4.22214	0.82883
x_{10}	ca_M5	0.1		3.08608	0.11375
x_{13}	velocity	29	m/s	2.21249	0.05914
x_{18}	x_error	5266.7	m	1.76969	0.28903
x_{22}	v_error	2.033	m/s	1.21212	0.13776
x_{17}	xcg	0.01	m	0.84225	0.03984
x_{16}	longitude	0.01	deg	0.78053	0.12117
x_9	cn_M10	0.05		0.75676	0.09913
x_6	ca_kn	0.1		0.54626	0.00483
x_3	max_rate	1	%	0.48513	0.00000
x_4	max_accel	0.178	g's	0.29047	0.02088
x_{25}	mass	2	kg	0.26619	0.00000
x_{23}	w_error	2.033	m/s	0.24186	0.36119
x_{14}	azimuth	0.09	deg	0.06646	0.76553
x_2	bank	5	deg	0.00000	0.00000
x_7	cn_kn	0.1		0.00000	0.00000
x_{15}	latitude	0.01	deg	0.00000	0.49767
x_{21}	u_error	2.033	m/s	0.00000	0.13159

Figure 48 plots the parachute deployment conditions for the optimized MSP ‘01 solution Monte Carlo results. This plot is compared with Figure 44 for the nominal case. Note that a similarly few number of simulations resulted in deployments outside the Mach number and dynamic pressure limits. The optimized solution would not be valid (meaningful) if this were not so. It is interesting to note that in contrast to the nominal case, none of the solution simulations showed parachute deployments near the upper Mach number limit.

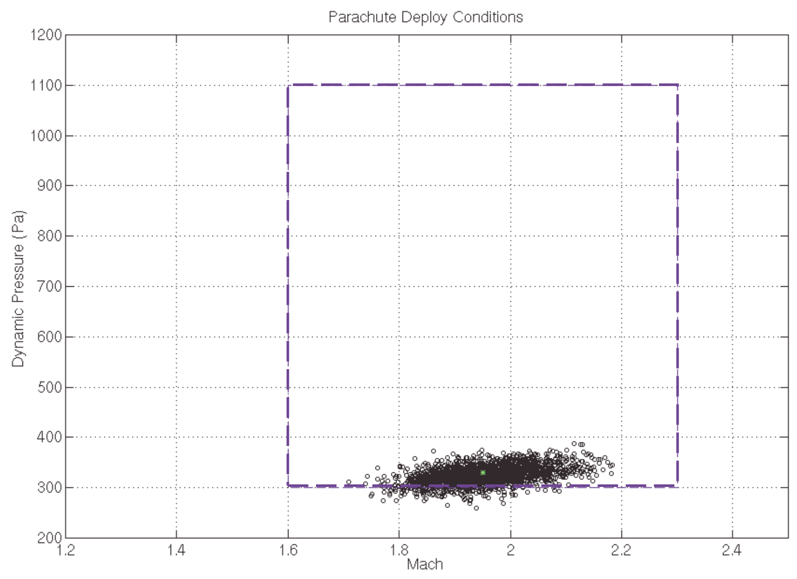


Figure 48: Parachute Deployment, MSP ’01 Solution.

Figure 49 plots the solution landing footprint for the MSP ‘01 problem. Figure 50 shows the same footprint as Figure 49, only in the down-range and cross-range coordinate system. Again, the overlaid circle is the 10-km target miss-distance. The footprint has been significantly reduced, but too large a percentage of simulations remain outside the target miss-distance. Another iteration of the procedure must be performed to correct this.

Note that the down-range dispersions have been reduced by a greater percentage than those in cross-range, resulting in a less eccentric ellipse. Ideally, the ellipse would perfectly match the target circle (i.e., an ellipse with zero eccentricity). Again, the 1.41-km semi-minor axis was chosen to minimize the eccentricity of the ellipse as much as possible.

Note also that the footprint ellipse is still not centered on the nominal landing site. This is due to a difference between the nominal and off-nominal trajectories. Specifically, the nominal trajectory set the Mars-GRAM atmospheric perturbation flag to zero (indicating no perturbations in the atmospheric properties), whereas all the other simulations did not. The difference in the mean trajectory, compared to the nominal trajectory, is therefore the atmospheric perturbations.

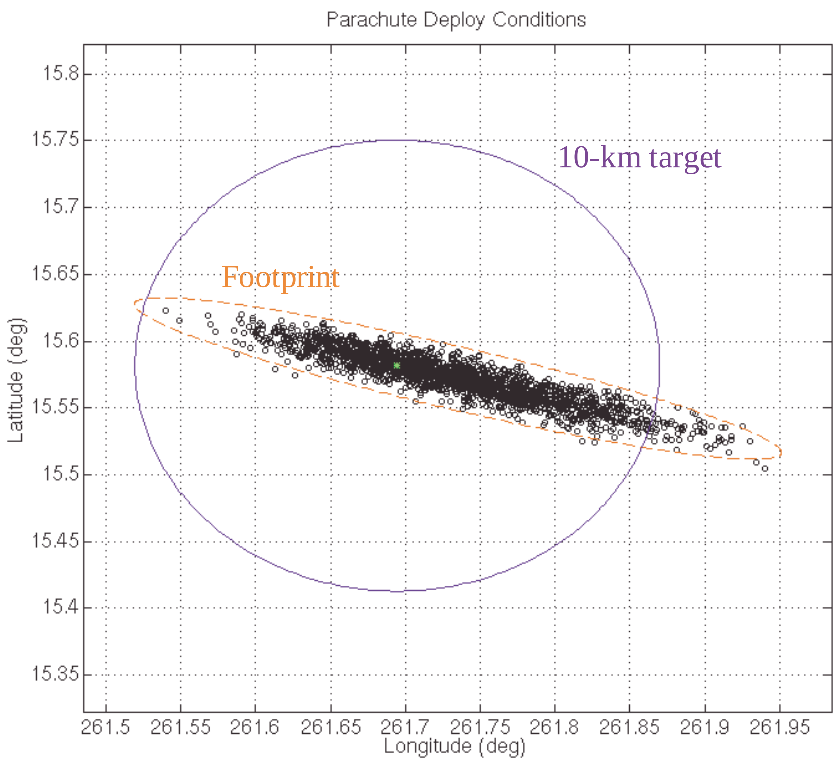


Figure 49: BVN footprint, MSP '01 1st Iteration (lat,long).

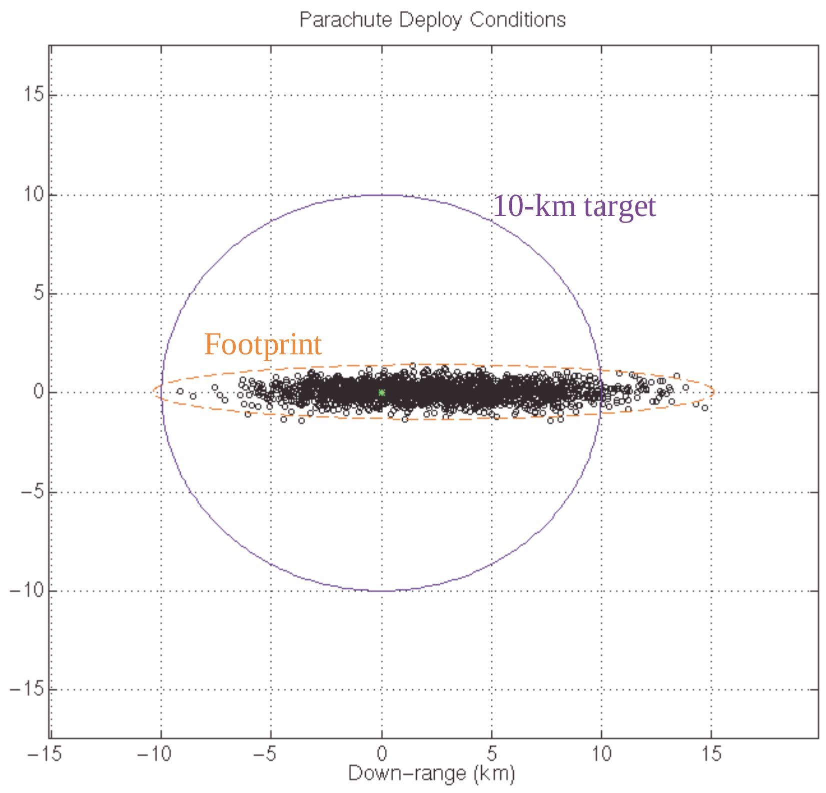


Figure 50: BVN footprint, MSP '01 1st Iteration (dr,cr)

To correct the discrepancy between the nominal landing site and the mean landing site, the Monte Carlo inputs were changed such that the Mars-GRAM atmospheric perturbation flag was set to one (instead of zero) for the nominal trajectory. The atmospheric perturbation seed used for the nominal was still zero. In this way, the random perturbations for the nominal cases were consistent and repeatable. This had the effect of moving the nominal landing site much closer to the mean. Figure 51 shows the footprint for this case along with the target 10-km circle centered on the nominal landing. The effect has been to center the ellipse in the target circle, but the ellipse is still too big. Figure 52 shows the same footprint as Figure 51, only in the down-range and cross-range coordinate system.

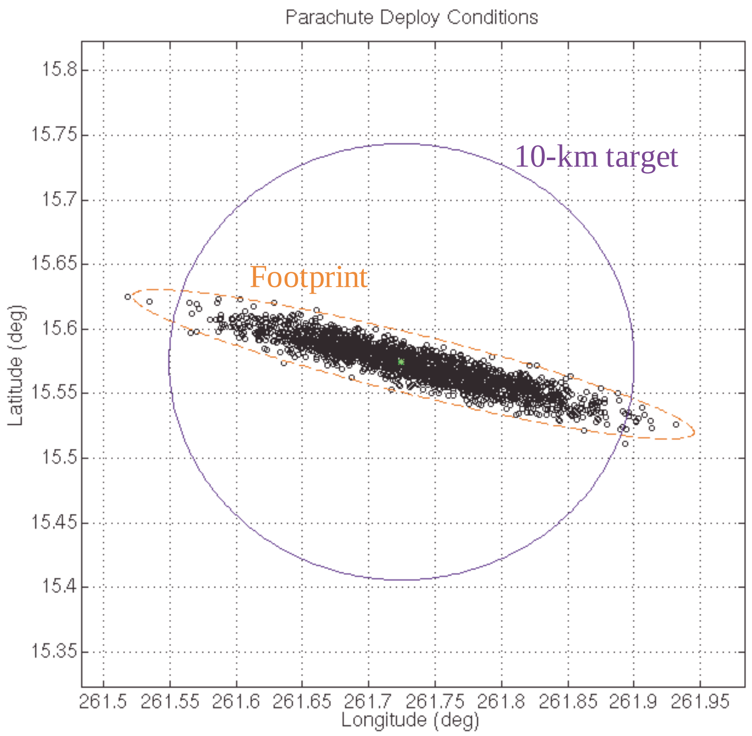


Figure 51: BVN footprint, MSP '01 Re-centered (lat,long).

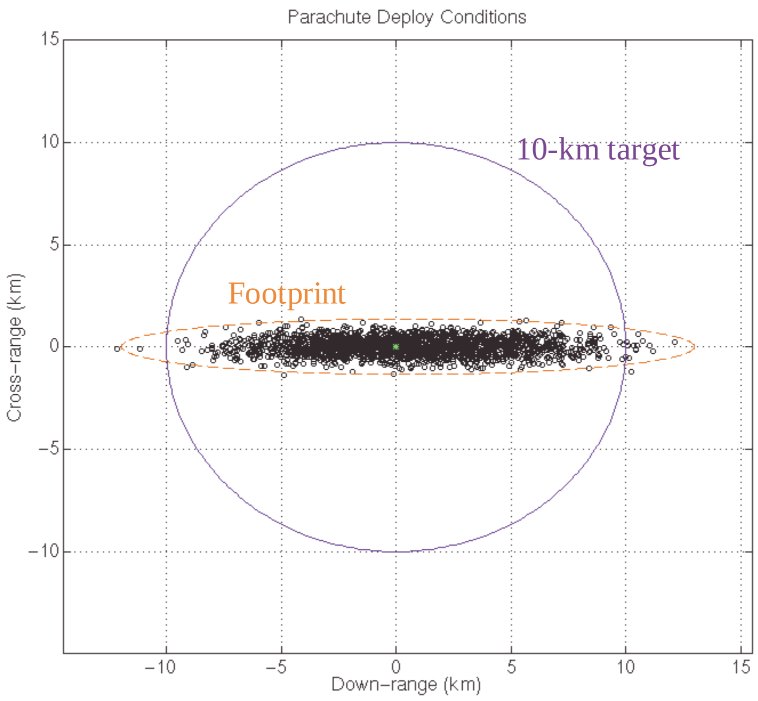


Figure 52: BVN footprint, MSP '01 Re-centered (dr,cr).

10.4.2 2nd Iteration Results

Next, a second iteration of the procedure was performed. The solution from the 1st iteration was taken as the new initial values and new metamodel coefficients were determined. These coefficients are shown in Table 33, where b_1 is the coefficient associated with the semi-major axis of the footprint ellipse and b_2 is the coefficient associated with the semi-minor axis of the ellipse. The constant coefficient for the semi-major axis was b_1^0= 5.45200. The constant coefficient for the semi-minor axis was b_2^0= 0.0102429.

Table 33: Metamodel Coefficients, MSP '01 2nd Iteration.

	Name	Initial Values	Units	b_1	b_2
x_1	jdate	0.072412	days	1.56308E+04	8.04970E-02
x_2	bank	5	deg	0.00000E+00	0.00000E+00
x_3	max_rate	0.92793	%	3.24334E-01	5.58387E-05
x_4	max_accel	0.172976	g's	2.21636E+00	1.46765E-02
x_5	trim	0.13692	deg	7.35684E+02	9.30055E-01
x_6	ca_kn	0.091186		1.36626E+01	5.02281E-02
x_7	cn_kn	0.1		4.48291E+00	2.48580E-02
x_8	ca_M10	0.002714		1.38524E+06	2.85912E+02
x_9	cn_M10	0.042743		3.51012E+02	2.84401E+00
x_{10}	ca_M5	0.044742		9.96201E+02	1.57644E+00
x_{11}	cn_M5	0.01767		1.22760E+04	1.22829E+02
x_{12}	flt_path	0.026133	deg	1.07202E+04	2.48020E+00
x_{13}	velocity	15.7619	m/s	4.75793E-03	5.48438E-06
x_{14}	azimuth	0.130613	deg	0.00000E+00	7.24657E+01
x_{15}	latitude	0.01119	deg	0.00000E+00	2.40550E+03
x_{16}	longitude	0.00849	deg	4.32994E+03	1.70791E+02
x_{17}	xcg	0.008288	m	5.64354E+03	9.09556E+00
x_{18}	x_error	3251.201	m	1.10658E-07	3.18669E-09
x_{19}	y_error	933.951	m	6.63671E-06	9.74834E-09
x_{20}	z_error	1964.798	m	7.21126E-07	2.51656E-08
x_{21}	u_error	2.04675	m/s	5.52909E-02	4.10298E-03
x_{22}	v_error	1.488362	m/s	2.81780E-01	4.44303E-03
x_{23}	w_error	2.096627	m/s	7.78217E-03	3.35029E-02
x_{24}	cg_offset	0.000453	m	5.21438E+07	5.77545E+04
x_{25}	mass	1.951916	kg	0.00000E+00	1.00209E-04
x_{26}	tau	0.126373		3.44542E+02	2.11515E-01
x_{27}	angle	0.069952	rad	7.61865E+01	3.51608E-02

Table 34 ranks the “importance” of each uncertainty at the second iteration – though again care should be exercised in interpreting the results since they are dependant on the assumed initial values. When comparing these values with those in Table 32, note that the most important uncertainties have changed slightly. This table shows that now the top five contributors to the semi-major axis are: Julian date (x_1), trim angle-of-attack (x_5), c.g. lateral offset (x_{24}), axial aerodynamic coefficient at Mach 10 (x_8), and initial flight path angle (x_{12}).

Table 34: Uncertainty Ranking, MSP ’01 2nd Iteration.

	Name	x_i^0	Units	$x_i^0 b_1^{0.5}$	$x_i^0 b_2^{0.5}$
x_1	jdate	0.072412	days	9.05319	0.02054
x_5	trim	0.13692	deg	3.71376	0.13205
x_{24}	cg_offset	0.000453	m	3.27078	0.10885
x_8	ca_M10	0.002714		3.19398	0.04589
x_{12}	flt_path	0.026133	deg	2.70581	0.04116
x_{19}	y_error	933.951	m	2.40603	0.09221
x_{26}	tau	0.126373		2.34571	0.05812
x_{11}	cn_M5	0.01767		1.95777	0.19583
x_{20}	z_error	1964.798	m	1.66849	0.31169
x_{10}	ca_M5	0.044742		1.41217	0.05618
x_{13}	velocity	15.7619	m/s	1.08722	0.03691
x_{18}	x_error	3251.201	m	1.08152	0.18353
x_9	cn_M10	0.042743		0.80079	0.07208
x_{22}	v_error	1.488362	m/s	0.79007	0.09921
x_{17}	xcg	0.008288	m	0.62263	0.02500
x_{27}	angle	0.069952	rad	0.61057	0.01312
x_{16}	longitude	0.00849	deg	0.55865	0.11095
x_3	max_rate	0.92793	%	0.52846	0.00693
x_{21}	u_error	2.04675	m/s	0.48127	0.13110
x_6	ca_kn	0.091186		0.33705	0.02044
x_4	max_accel	0.172976	g's	0.25752	0.02096
x_7	cn_kn	0.1		0.21173	0.01577
x_{23}	w_error	2.096627	m/s	0.18496	0.38376
x_2	bank	5	deg	0.00000	0.00000
x_{14}	azimuth	0.130613	deg	0.00000	1.11187
x_{15}	latitude	0.01119	deg	0.00000	0.54883
x_{25}	mass	1.951916	kg	0.00000	0.01954

Figure 53 plots the parachute deployment conditions for the second iteration results. This plot is compared with Figure 44 for the nominal case and Figure 48 for the first iteration results. Note that again a similarly few number of simulations resulted in deployments outside the Mach number and dynamic pressure limits. This is important because it shows that the optimization did not adversely affect the guidance system's ability to execute a successful parachute deployment. In fact, fewer parachute deployments are initiated at the upper Mach number limit.

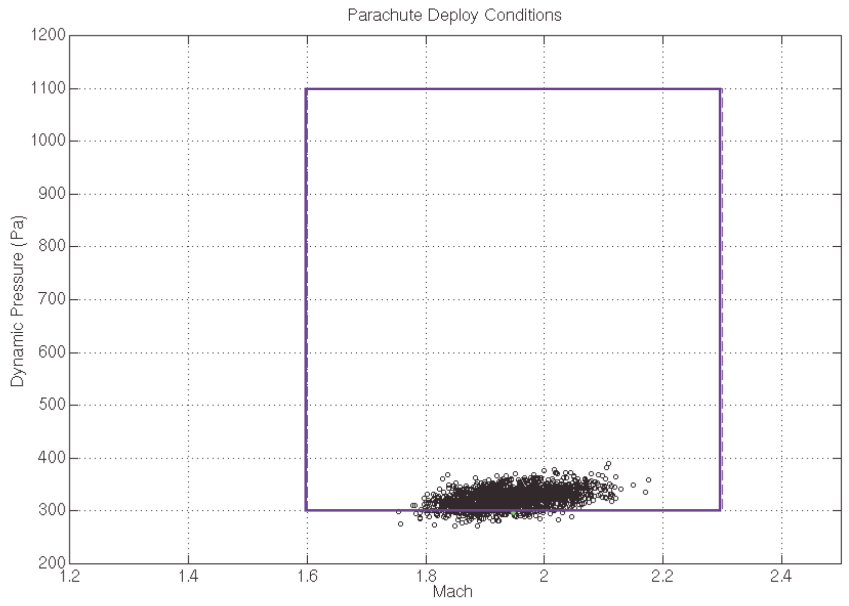


Figure 53: Parachute Deployment, MSP '01 2nd Iteration.

Figure 54 plots the solution landing footprint for the second iteration results. Compare this plot with Figure 45 and Figure 51. Figure 55 shows the same footprint as Figure 54, only in the down-range and cross-range coordinate system. Again, the overlaid circle is the 10-km target miss-distance. The footprint now has been reduced to an approximately 10-km semi-major axis. No further iterations are required. Note that only three of the 2000 simulated landings fall outside the target circle.

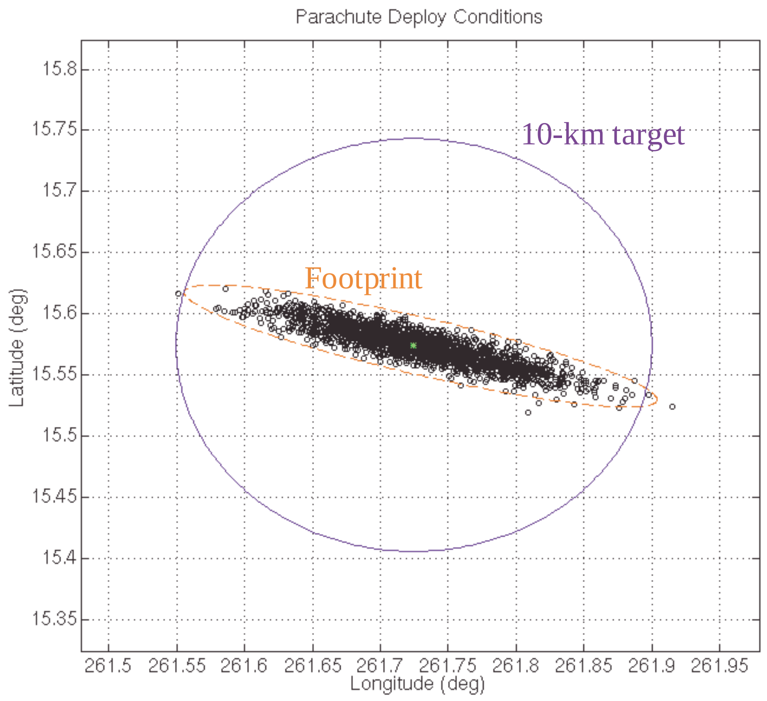


Figure 54: BVN footprint, MSP '01 2nd Iteration (lat,long).

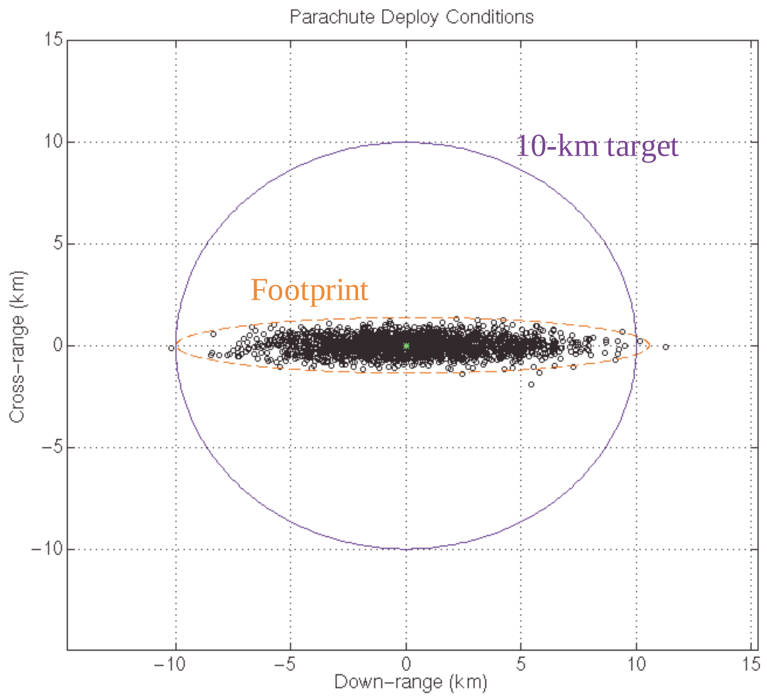


Figure 55: BVN footprint, MSP '01 2 nd Iteration (dr,cr).

Figure 56 shows the empirical miss-distance cumulative distribution function for the second iteration results. This plot is compared to Figure 47 for the nominal case. Note that nearly all of the simulations missed the target landing site by less than the allowable 10 km. This is precisely the desired result.

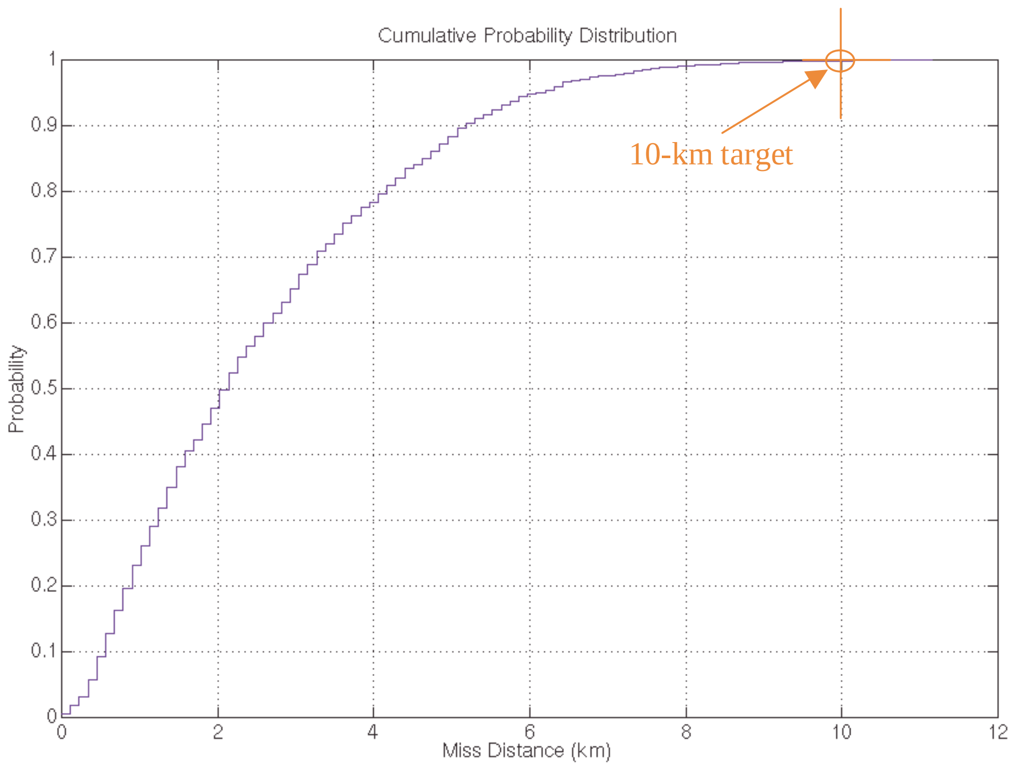


Figure 56: Miss-distance cdf, MSP '01 2nd Iteration.

10.4.3 Cost Trade

The final step of the methodology is to evaluate the optimum answer. In this example, Table 28 shows that several uncertainties have been reduced by an order-of-magnitude. These uncertainties are: Julian date (x_1), trim angle-of-attack (x_5), axial aerodynamic coefficient at Mach 10 (x_8), initial flight path angle (x_{12}), c.g. lateral offset (x_{24}), and c.g. clocking angle (x_{27}). Comparison with Table 32 and Table 34 show that these variables were the ones identified as the most important. Again, the reason for such tight tolerances was that the problem was intentionally stressed by adding the Julian date uncertainty. Therefore these results are academic and are not representative of the MSP ’01 spacecraft or the guidance algorithm.

A trade study was conducted to demonstrate how uncertainty weightings can be used. For this study, it was assumed that the initial flight path optimum value of ±.0245 deg was too restrictive (i.e., not feasible with available technology). It was also assumed that error in arrival Julian date was overestimated and therefore, could be reduced further at little or no expense. To correct these observations, the weighting values for these uncertainties were altered and the optimization was performed again to find a new set of optimum extrema. The new optimization utilized the same metamodel coefficients as the 1st iteration, given in Table 31.

Table 35 shows the results of this trade study. The w_1=1, w_{12}=1 column shows the results for equal weightings, where every uncertainty is given a weighting of unity, and then normalized by the sum of weightings. These are the 1st iteration results from Table 28. The next column shows the results when the Julian date (x_1) is weighted at w_1=0.001 and the initial flight path angle (x_{12}) is weighted at w_{12}=1000 (all other variables are weighted at w_i=1). Likewise, the final column shows w_1=0.0001 and w_{12}=100000. In all cases, the weighting values were then normalized by the sum of the weightings.

Table 35: Cost Trade, MSP '01.

	Name	Init. Val.	Units	w_1=1 w_{12}=1	w_1=0.001 w_{12}=1000	w_1=0.00001 w_{12}=100000
x_1	jdate	0.5	days	0.07241208	.00237708	0.00011599
x_2	bank	5	deg	5	5	5
x_3	max_rate	1	%	0.92793034	0.49727147	0.11759256
x_4	max_accel	0.178	g's	0.17297628	0.11661445	0.02943623
x_5	trim	2	deg	0.13692035	0.04488996	0.01016648
x_6	ca_kn	0.1		0.09118567	0.04634969	0.01086605
x_7	cn_kn	0.1		0.1	0.1	0.1
x_8	ca_M10	0.03		0.00271375	0.00089012	0.0002016
x_9	cn_M10	0.05		0.04274252	0.01896985	0.00437281
x_{10}	ca_M5	0.1		0.0447418	0.01511467	0.00342698
x_{11}	cn_M5	0.08		0.01766993	0.00580565	0.00131492
x_{12}	flt_path	0.23	deg	0.02613341	0.08431065	0.08839497
x_{13}	velocity	29	m/s	15.7618974	5.46601654	1.24066563
x_{14}	azimuth	0.09	deg	0.13061347	0.14482724	0.04139751
x_{15}	latitude	0.01	deg	0.01119022	0.01171344	0.02717202
x_{16}	longitude	0.01	deg	0.00848979	0.00371906	0.00085673
x_{17}	xcg	0.01	m	0.00828815	0.00354243	0.0081436
x_{18}	x_error	5266.7	m	3251.20079	1151.09535	261.48801
x_{19}	y_error	5266.7	m	933.951039	306.661309	69.454904
x_{20}	z_error	5266.7	m	1964.79807	646.68233	146.45724
x_{21}	u_error	2.033	m/s	2.04675043	2.05121972	2.08011816
x_{22}	v_error	2.033	m/s	1.48836239	0.56947963	0.12989547
x_{23}	w_error	2.033	m/s	2.09662743	1.48349171	0.38002743
x_{24}	cg_offset	0.005	m	0.00045295	0.00014852	0.00003364
x_{25}	mass	2	kg	1.95191636	1.36474	0.35046497
x_{26}	tau	0.7		0.12637274	0.04151624	0.00940313
x_{27}	angle	1.570795	rad	0.06995173	0.02291612	0.0518975
f	cost	1.00000		4.20878	18.17441	204.49926

This table shows how the uncertainty in Julian date is reduced by several orders-of-magnitude, due to the smaller weighting, while the uncertainty in flight path angle is increased more than a factor of four, due to the larger weighting. Most all of the other variables have been reduced somewhat to correct for the higher flight path angle uncertainty.

The maximum feasible value for any uncertainty can be found by substituting zero for each of the other uncertainties in the metamodel. Equation 84 shows this calculation, where R^2 is the square of the desired semi-axis of the footprint. The final flight path value of 0.08839, shown in Table 35, is very close to the maximum possible value, 0.08861.

$$x_i^{\max} = \sqrt{\frac{\left(R^2 - b_0\right)}{b_i}} \tag{84}$$

For comparative purposes, the cost value shown is for a reciprocal model with equal weighting. That is to say that the cost value shown is not the same as the objective function which was minimized. This cost shows a dramatic increase in cost necessary to relax the uncertainty in initial flight path angle.

In summary, this chapter presented the Mars Surveyor 2001 Lander example, describing the mission, the simulations, and the results. The purpose of this chapter, to demonstrate the methodology on a “real world” problem, was completed successfully. Conclusions, lessons learned, and accomplishments of this research are presented in the next chapter, Chapter 11.

CHAPTER 11

COMMENTS AND CONCLUSIONS

11.1 Methodology Summary

An approach to optimize the uncertainties in the Monte Carlo analysis of spacecraft landing footprints has been developed. This approach is based on the following assumptions: (1) the engineer can control dispersions of uncertainty variables by altering their probability density ($\pm3\sigma$ extrema) and (2) there exists a real cost to changing any extremum from the baseline. It follows then, that if the uncertainty dispersions are controllable, then so are the forecast variables. Therefore, a non-unique set of uncertainty extrema exists that results in any desired forecast performance. Additionally, any number of feasible sets of extrema may be ranked according to their associated costs.

A metamodel is used to first write expressions for the semi-major and semi-minor axes of the landing footprint as functions of the independent uncertainty extrema. In general, any forecast variable may be used in a constraint. The metamodel, called an "ellipse surface" because it has elliptical contour lines, is a simplified response surface that has no linear or cross terms. The coefficients of the metamodel are determined by performing an (n+1) minimum-point design of experiments, where each experiment consists of performing a Monte Carlo analysis and constructing a footprint.

A cost-tolerance objective function is written for the cost as a function of the uncertainty extrema. The recommended function, the "cost-plus-quadratic" function, is a linear combination of a reciprocal cost model and a quadratic function. Other possible functions are the reciprocal cost-tolerance model and the minimum Euclidean distance model.

Next, an optimization is performed that minimizes the cost subject to the constraint that the landing footprint is a specified size. Finally, one or more validation runs are performed using the optimum extrema to ensure that the desired outcome is obtained.

11.1.1 Optimization Procedure

1) Prepare the nominal trajectory simulation. Much of the manual time involved in Monte Carlo analyses is spent preparing the nominal trajectory. This is because each event in the Entry, Descent, and Landing (EDL) sequence must be modeled, aerodynamic and mass properties must be obtained, and the trajectory must be optimized. Typically, an optimization is performed to maximize performance (e.g., minimize weight) subject to a variety of mission constraints.

2) Construct the Monte Carlo simulation. Complete a table similar to Table 27. This table provides most of the required mcp input (*sim*.mci). First, identify the uncertainties by name, nominal value, distribution, and extrema. The nominal values for the uncertainties are taken from the nominal trajectory. The distribution and extrema for each uncertainty must be specified by some means (typically by experts). Second, identify where each uncertainty will go in the POST input deck. This is done by placing markers (e.g., ****name****) in place of values in the nominal input deck. This modified POST deck is the template, (*sim*.tpl), which is used to construct the random trajectories. Care must be taken to match the uncertainty and marker names.

3) Determine the number of simulations to run. Select the number of simulations, s, from Table 13 or Figure 15 that gives the desired confidence interval at the desired confidence level. For example, if an error of ±2% is desired (with a confidence of 90%), 3554 simulations should be run. Likewise, if the same error were required, but with a 95% confidence, 5030 simulations would be appropriate. For a 99% confidence, 8642 simulations are required. These confidence intervals are only applicable to bivariate normal (BVN) footprints.

4) Run the baseline Monte Carlo analysis using the initial guesses for each uncertainty extrema. Plot and evaluate the footprint and compare it to the target. The footprint may be constructed using any of the methods described in Chapter 6. However, the BVN method is recommended. If the mean or nominal landing sites are not near the target, the nominal inputs (nominal POST deck) must be adjusted.

5) Decide upon the cost model. If actual cost information is not known, then the cost-plus-quadratic model is recommended because of its better numerical characteristics. The weightings of the individual uncertainties must also be determined. More expensive uncertainties should be weighted greater than less expensive ones.

6) Run experiments to determine the ellipse surface coefficients. If there are uncertainties present that are not design variables (i.e., will not be optimized), then the first experiment is run with each design variable set at zero. In other words, a Monte Carlo analysis is run where only those uncertainties that are <u>not</u> design variables are varied. The results from this experiment determine the constant terms, $b_{0,k}$, in the metamodel as shown in Equation 59. Experiments are also run for each design variable. For these experiments, every design variable is set to zero except for one, which is set to its initial guess value. The results from these experiments determine the quadratic terms, $b_{i,k}$, in the model as shown in Equation 60.

$$b_0 = R^2, \quad x_i = 0, \quad i = 1 \ldots n \tag{59}$$

$$b_i = \frac{\left(R^2 - b_0\right)}{\left(x_i^0\right)^2}, \; x_j = 0, \; j = 1 \ldots n, \; j \neq i \tag{60}$$

This is the most computationally time-consuming step in the process. Since there are n design variables (uncertainties) and s simulations per experiment (runs per Monte Carlo), then n+1 experiments (Monte Carlo analyses) and $s(n+1)$ simulations are required. For example, 27 design variables require 28 experiments. At 2000 runs each, this amounts to 56,000 randomly generated trajectories. Even at 4 seconds per simulation, that requires 62 hours of computer time.

7) Solve the 2-by-2 system of equations, Equation 78, for the Lagrange multipliers. Initial guesses are required for the Lagrange multipliers. These initial guesses need to be somewhat close to the solution, so some experimenting may be required. Once the solution is found, substitute the Lagrange multipliers into Equation 74. The resulting design variables are the optimum extrema.

$$\begin{bmatrix} \frac{\partial h_0}{\partial \lambda_0} & \frac{\partial h_0}{\partial \lambda_1} \\ \frac{\partial h_1}{\partial \lambda_0} & \frac{\partial h_1}{\partial \lambda_1} \end{bmatrix} \begin{bmatrix} \Delta\lambda_0 \\ \Delta\lambda_1 \end{bmatrix} = \begin{bmatrix} -h_0 \\ -h_1 \end{bmatrix} \qquad \textbf{(78)}$$

$$x_i = \left\{ \frac{-\frac{1}{3} w_i x_i^0}{\lambda_0 b_{i,o} + \lambda_1 b_{i,1} - \frac{1}{3} w_i \left[\frac{1}{x_i^0} \right]^2} \right\}^{\frac{1}{3}} \qquad \textbf{(74)}$$

8) Validate the solution. Use the optimum extrema to run a single Monte Carlo analysis at the solution point. If desired, several trials may be made at the same point and the results averaged. Plot and evaluate the footprint and compare it to the target. The footprint may be constructed using any of the methods described in Chapter 6. However, the BVN method is recommended.

9) Evaluate whether the new footprint is sufficiently close to the desired ellipse. If it is sufficiently close, continue. If not, set the initial design variable guesses to the optimized design variables from step (7). Repeat steps (6) through (9). If the problem does not converge within a few of these iterations, stop and evaluate the reason why.

10) Evaluate whether the solution is feasible. In other words, determine if the optimized uncertainties are physically and economically achievable. If they are

achievable, end. If not, reevaluate the uncertainty weightings. Increase the weight of uncertainties that were determined to be too restrictive. Repeat steps (7) through (10). These changes may be evaluated very quickly so long as step (6) does not need to be repeated.

11.2 Objectives

All the objectives established in Chapter 2 were completed successfully. The purpose of the proof-of-concept was to show that the methodology would work. The purpose of the "real world" problem was to demonstrate the method at a realistic scale. Both of these tasks were accomplished successfully. The following specific observations were made pertaining to each of these two problems.

11.2.1 Proof-of-Concept

1) The proof-of-concept successfully located the minimum cost extrema that satisfied the 3-km landing footprint. The appropriateness of the metamodeling technique was evaluated by comparing three forms of the model based on the quality of their fits. The standard response surface was deemed unacceptable due to an R-square value less than .99 and an average error greater than 3%. The remaining models (the ellipse surface and the squared response surface) were determined appropriate. Plots of the design space support the solutions found using the Newton-Raphson iteration.

2) A gridsearch was used to provide a visualization of the design space. The resultant surface was very closely approximated with the simplified ellipse surface method. This method is very important because it provides a means for efficiently scaling the methodology to problems with larger numbers of design variables due to the minimal number of function evaluations required.

3) It appears more appropriate to fit the square of the range, rather than the range itself. The two response surface methods, which did not fit R^2, were shown to have

unacceptable errors. Comparisons between the methods indicate a factor of four decrease in average error associated with fitting R^2.

11.2.2 MSP '01 Example

1) The Mars Surveyor 2001 Lander example demonstrated that the developed solution procedure could be applied to "real world" problems. The solution was found in only two iterations of the methodology. It is very important to note that with 27 design variables, this problem could not have been practically solved using a response surface metamodel.

2) The cost-plus-quadratic function, in conjunction with the method of Lagrange multipliers with substitution, showed better numerical stability than the other two objective functions. When attempted, the direct solution of the method of Lagrange multipliers diverged for a problem with two constraints. The reciprocal cost-tolerance model led to diverging solutions also when some of the uncertainty coefficients were found to be small. The minimum-distance model led to negative solutions.

3) The BVN footprint method was shown to lead to empirical probabilities very close to the theoretical probabilities. This supports the hypothesis that normal distributions are appropriate approximations to the down-range and cross-range distributions (even in case of guided, lifting trajectories). This is important because the BVN method is the only method that allows the calculation of a confidence interval, and therefore, a determination of the required number of simulations.

11.3 Comments

This research has been an academic exercise aimed at demonstrating a methodology for finding the minimum-cost set of extrema that satisfy the footprint requirements. It is obvious that when using this methodology in actual problems, the optimum set of extrema is only as good as the assumed cost function (though the solution will always be a feasible set, if one exists). Accurately predicting the cost, therefore, is a key factor in obtaining meaningful results. Unfortunately, cost is typically very difficult to predict early in the design process. As a result, the engineer may be faced with certain situations that require slight modifications to the procedure described above. Some of these situations are discussed here.

11.3.1 Determining Metamodel Coefficients

When determining the metamodel coefficients, a minimum-point design of experiments was used, where each coefficient was determined uniquely from a specific experiment. The value of the design variable for these experiments was chosen arbitrarily to be the initial values of the uncertainties. However, any non-zero value, x_i, may be used, as shown in Equation 85. The initial value was chosen because it is the current best estimate of the solution.

$$b_i = \frac{\left(R^2 - b_0\right)}{x_i^2}, \; x_j = 0, \; j = 1 \ldots n, \; j \neq i \qquad \textbf{(85)}$$

If computer resources are not limited, two or more experiments may be used to approximate the slope of the design space (using forward or central differencing). If more experiments are run, the designer may also add the linear response surface terms to the ellipse surface metamodel.

11.3.2 Uncertainty Correlations

The ellipse surface metamodel is necessarily a very simple model, which assumes that there are no correlations between design variables. This simplification is necessary to limit the number of coefficients needed and reduce the computational time. If there are only small correlations between variables, then this surface is a good approximation to the actual design space. Multiple iterations of the method will account for small differences between the actual and approximate surfaces (this was demonstrated in the “real world” example).

If the correlations are large, but known, then the design variables may be replaced with a set of uncorrelated variables through a transformation (similar to the BVN transformation in Chapter 6). If the correlations are large, but not known, then a standard response surface may be used in place of the simple ellipse surface. In this case, the number of design variables may need to be reduced through a screening process.

11.3.3 Variable Screening

Determining the importance of each uncertainty is useful both in understanding the problem, and in screening design variables when necessary. The "importance" of different uncertainties (in terms of variability of the range) can be determined by comparing their metamodel coefficients. In general, variables with smaller coefficients are less important than those with larger ones. Mathematically, the value of b_i is related to the slope of the linear edge of the metamodel surface and represents the change in the response (range squared) per unit change in the square of the design variable. The units of b_i are range squared divided by the units of the uncertainty squared. Because the units are different for each design variable, they are not easily compared.

It is more useful to compare the coefficient, $x_i^0\sqrt{b_i}$, which has units of just range. However, this coefficient is influenced by the presumed size of the uncertainty extrema, since it represents the change in range due to a full-scale change in the uncertainty. For example, entry flight path angle may be very important in determining the size of the footprint, but if it is allowed to vary only ±0.01 degrees, it may have less effect than another uncertainty, which is allowed to vary ±100%. Therefore, b_i and $x_i^0\sqrt{b_i}$ only provide an indication of "importance", and not an absolute measure.

11.3.4 Fixed Budget Solution

This research was conducted around the premise that what was desired was the minimum-cost uncertainties that would meet a prescribed footprint size. An equally valid problem would be a search for the smallest footprint available for a fixed budget. In this case, the objective and constraints are switched as shown in Equation 86. The designer may still use the same metamodel and cost-tolerance relations. This would be a small departure from the presented methodology.

$$\begin{aligned} &\text{given :} && \bar{x} = \{x_i = 3\sigma_i\} && i = 1, 2, \dots, n \\ &\text{minimize :} && f(\bar{x}) = b_0 + \sum_{i=1}^{n} b_i x_i^2 && \\ &\text{subject to :} && h = \sum_{i=1}^{n} w_i \left\{ \frac{2}{3}\frac{x_i^0}{x_i} + \frac{1}{3}\left[\frac{x_i}{x_i^0}\right]^2 \right\} - Budget = 0 && \end{aligned} \qquad \textbf{(86)}$$

11.3.5 <u>Discrete Cost Function</u>

If actual cost data is known, it is likely to be discontinuous. A discrete cost function, however, requires a different optimization method. This is because the method of Lagrange multipliers presented in this thesis requires continuous functions. A zero-order method, such as genetic algorithm or simulated annealing, is recommended. The same metamodel may be used for the constraints.

11.3.6 <u>Side Constraints</u>

If the engineer knows that an uncertainty should be between certain limits, then placing side constraints on each design variable is an effective method for preventing unreasonably small or large solutions. However, if any inequality constraints are desired (e.g., side constraints on each design variable) another optimization method must be used. This is because the method of Lagrange multipliers applies only to equality constraints. The same metamodel and objective function may be used in conjunction with any optimization method without changing the general methodology.

11.3.7 <u>Shifting the Cost Axis</u>

Another method for limiting the decrease of a particular uncertainty, is to use a cost-tolerance model that goes to infinity at some positive limit, x^{lim}.   This simulates a tolerance for which there is a non-zero practical lower limit. This can also be accomplished simply by using a change of variables (given by Equation 87) and solving for x_i'.

$$x_i' = x_i - x_i^{\lim} \tag{87}$$

The designer must remember, when using this option, that the design variable must be set to x_i^{lim} (rather than zero) when determining the other metamodel coefficients. Likewise, the range must be divided by $\left(x_i^0 - x_i^{\lim}\right)^2$ when determining the associated metamodel coefficient, as shown in Equation 88.

$$b_i = \frac{\left(R^2 - b_0\right)}{\left(x_i^0 - x_i^{\lim}\right)^2},\ x_j = 0,\ \ j = 1 \ldots n,\ \ j \neq i \tag{88}$$

11.3.8 Fixing a Variable

If the designer wishes to fix an uncertainty at a particular value, then the best approach is to generate a new set of metamodel coefficients, fixing the particular uncertainty at the desired value. This will have the effect of increasing the constant term, b_0, and reducing the number of design variables by one. If, however, the square root of the new b_0 exceeds the desired range squared, then the problem is infeasible at that uncertainty limit. This is because the range can not be reduced below the square root of b_0, as shown in Equation 59.

$$b_0 = R^2,\ \ \ x_i = 0,\ \ \ i = 1 \ldots n \tag{59}$$

Generating new coefficients, however, requires additional computer time, which may be expensive. A "quick-and-dirty" solution is to mathematically add the contribution of the desired uncertainty to the constant term, b_0, and keep the remaining coefficients. This is shown in Equation 89, where b_0' is the new constant term.

$$b_0' = b_0 + b_i x_i^2 \tag{89}$$

The validity of this approach is dependant upon the validity of the original assumption of independent variables. The engineer must understand in this case that the coefficients are being used somewhat outside their original intended purpose. Remember that when the coefficients are determined, only the design variables are set to zero (i.e., the uncertainties which are not optimized are allowed to vary).

11.3.9 Over-shoot, Under-shoot

It is often the case that the variables that result in over-shoot (positive) down-range miss-distances are different from those that result in undershoot (negative) miss-distances. This can lead to asymmetric footprints. In this case, the down-range data may be divided into positive and negative data sets. These data sets may be used in a problem with three constraints: positive semi-major axis, negative semi-major axis, and semi-minor axis. The means of constructing the footprint must likewise account for different sizes in positive and negative down-range.

11.4 Research Accomplishments

This research achieved the following ten specific accomplishments.

1) This research accomplished its primary goal – to go beyond the capabilities of current Monte Carlo analysis of planetary entry trajectories. By employing an optimization methodology, this research has shown that is possible to control the size of the landing footprint and establish tolerances for mission uncertainties. In a sense, the Monte Carlo analysis has been performed backwards: beginning with a desired output and proceeding to the required inputs. Although the emphasis in this research was landing footprints, any Monte Carlo forecast variable might be controlled through a constraint in the same way.

2) A simplified metamodel was developed, the “ellipse surface”, that scales well. This metamodel is equivalent to a response surface without linear or cross terms (i.e., only the constant and squared terms are present). This simplified model enables solutions of typical “real world” problems. This is because the computational expense associated with a standard response surface is prohibitive, even for problems with more than a just a few uncertainties.

3) An objective function was formulated that had better properties than both the reciprocal cost-tolerance model and the minimum Euclidean distance model. The “cost-plus-quadratic” model provides a more stable numerical problem because of

two desirable characteristics: (1) the cost becomes infinite as the uncertainty is reduced to zero, and (2) the unconstrained minimum occurs when the uncertainties are set to their initial values.

4) Equations were presented for three possible objective functions: the reciprocal model, the minimum-distance model, and the cost-plus-quadratic model. These different models allow the user to tailor the method to a particular problem. Following the procedure outlined in Chapter 8, many more functions are possible. For instance, any of the twelve cost-tolerance models listed in Chapter 8 may be used.

5) A technique for solving a constrained optimization with many design variables was explained. This technique is outlined in detail in Chapter 8 with supporting information in Appendix A. The classical methods of Lagrange multipliers, Newton-Raphson iteration, and LU decomposition were all applied to solve this problem. A numerical solver was written in C^{++} to perform this optimization.

6) Five methods of constructing footprints were described in detail. These descriptions included construction techniques, assumptions, and probabilities. Three methods created circular footprints: the 3-sigma range, Rayleigh, and Weibull methods. Two methods generated elliptical footprints: the 3-sigma down-range and cross-range, and the bivariate normal (BVN) methods. Any probability may be specified for the Rayleigh, Weibull, and BVN methods. Advantages and disadvantages of each were discussed.

7) The BVN method of constructing a footprint ellipse was recommended. This method is the most general, elegant, and statistically sound method, requiring only that the down-range and cross-range distributions be normally distributed. A transform, based on the covariance matrix of the data, is used to project a circle of a given probability from a standard, uncorrelated, bivariate normal space to the actual, correlated, BVN space. A procedure for constructing the ellipse and measuring the semi-major and semi-minor axes was presented. The use of this

method also allows the calculation of confidence intervals on the predicted size of the ellipse.

8) A confidence interval on the size of the BVN footprint was derived. This confidence interval for the semi-major axis and semi-minor axes of the ellipse is based on the confidence interval for the sample variance of a normal distribution and is independent of the ellipse probability. Because the confidence interval is a function of the number of sample points, it may be used to determine, *a priori*, how many simulations must be performed to achieve a given accuracy in the size of the footprint.

9) The "ran1" and the Perl built-in "rand" random number generators were evaluated for their appropriateness in modeling random processes. These functions were compared against each other and the Matlab built-in random number generator in seven tests. Some of these tests were graphical and some were statistical. Possible weaknesses were found in both functions. However, a head-to-head comparison in the actual optimization process (the most demanding and telling of all the tests) showed that both functions produced similar results.

10) The Monte Carlo process was automated with the "mcp" (Monte Carlo POST) program. This program, written in Perl, orchestrates the flow of information needed to perform Monte Carlo simulations. All the problem specific information is contained within two files: an input file, which defines the uncertainties and forecast variables; and a template POST input deck, which defines how the uncertainties are inserted into the simulation. Mcp also has the capability to run multiple simulations in parallel on multi-processor computers.

APPENDIX A

MATHEMATICS REVIEW

A.1. Method of Lagrange Multipliers

The method of Lagrange multipliers is a well-known technique, devised by the eighteenth-century French mathematician Joseph Louis Lagrange, for solving optimization problems with equality constraints. [123] This technique allows classic methods of solving unconstrained optimization problems to be used in constrained optimization problems. The trick is to augment the original objective function with the product of the constraints and unknown constants, λ, known as Lagrange multipliers. The minimum of this augmented objective function, Equation 90, satisfies the constraints exactly while minimizing the original objective function. Equation 90 is often referred to as the Lagrangian function in texts. [45] This technique eliminates the ℓ constraints, but at the price of adding ℓ new unknowns.

$$\tilde{f}\left(\bar{x},\bar{\lambda}\right)= f\left(\bar{x}\right)+\sum_{k=1}^{\ell}\lambda_k h_k\left(\bar{x}\right) \qquad \textbf{(90)}$$

The Lagrange multipliers are not just arbitrary constants, however, they have some physical meaning. Rao [124] shows that the value of the multipliers (at the solution point) is the marginal change in the objective function with respect to the constraint. In other words, λ is the change in objective function that would accompany a relaxation of the constraint. Therefore, λ indicates how tightly the constraint is binding at the optimum point.

A.1.1 <u>Direct Solution</u>

Rao [124] shows that a necessary condition for a function, *f(x)*, subject to the constraints $h_k(x)=0$, $k=1..\ \ell$, to have a relative minimum at a point x^* is that the first partial derivatives of the Lagrangian with respect to each of its arguments must be zero. This means that we can solve the original optimization problem by minimizing the Lagrangian in the normal way (we solve for a zero gradient). An $n+\ell$ system of equations is formed by taking the partial derivative of the augmented objective function with respect to each of the n design variables and ℓ Lagrange multipliers. We set each of these partial derivatives to zero and solve them as a system of simultaneous non-linear equations. Note that the partial derivative of the Lagrangian with respect to any of the Lagrange multipliers is the value of the corresponding constraint itself. This ensures that each of the constraints will be satisfied when the solution is found.

This method does not guarantee, however, that a <u>minimum</u> will be found. In fact, the requirement for a zero gradient is satisfied by any critical point. For the solution to be a minimum, the Hessian (matrix of second partial derivatives) must be positive definite. [45] Even this only guarantees a relative minimum and not a global one. A Newton-Raphson iteration is used to solve this system of simultaneous equations.

The advantage of this solution method is that it can be applied generally to any cost function and set of equality constraints for which the derivatives are known. The disadvantage is that instabilities can occur in the numerical solution of these equations (by Newton’s method), which lead to divergence. Often, very good initial guesses are required.

A.1.2 <u>Solution with Substitution</u>

A slightly different formulation of the problem can be obtained using substitution. Enforcing the tangency of the objective and constraint functions at the solution point, we set the partial derivative (with respect to the n design variables only) of the objective function equal to the partial derivative of the second term in the Lagrangian. This is shown in Equation 91.

$$\frac{\partial f}{\partial x_i} = \sum_{k=1}^{\ell} \lambda_k \frac{\partial h_k}{\partial x_i} \quad i = 1,2,\ldots,n \tag{91}$$

Each of these n equations is then solved for expressions that relate the design variables to the Lagrange multipliers. This is easily done when the equations are a function of only one design variable (independent of the other design variables). For instance, if the Lagrangian is given by Equation 92, then equating the partial derivatives yields Equation 93. This equation is easily solved for x_i to give Equation 94.

$$\tilde{f}(\bar{x}, \bar{\lambda}) = \sum_{i=1}^{n} w_i \frac{x_i^0}{x_i} + \sum_{k=1}^{\ell} \lambda_k \left\{ b_k^0 + \sum_{i=1}^{n} b_{ik} x_i^2 - R_k^2 \right\} \tag{92}$$

$$-w_i x_i^0 x_i^{-2} = 2 x_i \left(\lambda_0 b_{io} + \lambda_1 b_{i1} \right) \tag{93}$$

$$x_i = \left\{ \frac{-\frac{1}{2} w_i x_i^0}{\lambda_0 b_{io} + \lambda_1 b_{i1}} \right\}^{\frac{1}{3}} \tag{94}$$

Now that equations have been written for each of the design variables as a function of the Lagrange multipliers, substitute these expressions into the *k* constraint functions. Since the constraint functions must all equal zero, this now forms a system of ℓ simultaneous, non-linear equations. As with the direct solution described above, this system of equations is solved numerically using a Newton-Raphson iteration.

The advantage of this formulation, over the direct solution, is a smaller and more stable system of equations. Amazingly, the size of the system is limited only by the number of constraints (no matter how many design variables are present). The disadvantage is that the objective function must be chosen carefully so that the equations can be solved easily for x_i. Three possible objective functions that meet this restriction are the reciprocal model, the minimum-distance model, and the cost-plus-quadratic model.

In summary, the method of Lagrange multipliers reduces an optimization problem with constraints to one that is unconstrained (and therefore much easier to solve). This method is only applicable, however, to equality constraints. The solution to this problem may be made either directly or after a substitution that reduces the size of the system to the number of constraints.

A.2. Newton-Raphson Method

The Newton-Raphson method (often called just Newton's method) is a well-known iterative method for solving root-finding problems in one dimension. This method is easily extended, however, to multiple dimensions and is therefore applicable to the solution of simultaneous non-linear equations. This method solves a system of non-linear equations by expanding each of the equations using a Taylor's series. The higher order terms (those with higher than first derivatives) are neglected, which introduces an error term.

The neglected error term is of order $(\Delta X)^2$. Its neglect leads to a system of linear equations. Its existence, however, demands that the solution be repeated until ΔX is very small. Geometrically, this method is equivalent to approximating the intersection of multiple surfaces with the intersection of their tangent planes.

Beginning with an initial guess, X^0, the solution proceeds iteratively by updating the guesses, $X^0+\Delta X$. The Taylor's series expansion (about the initial guess) is shown in Equation 95.

$$f\left(\overline{X}^0 + \Delta\overline{X}\right) = f\left(\overline{X}^0\right) + \sum_{i=1}^{n}\left[\frac{\partial f}{\partial X_i}\left(\overline{X}^0\right)\right]\Delta X_i + order\left\{(\Delta X)^2\right\} \qquad \textbf{(95)}$$

Because the current guess, X^0, is known at the beginning of each iteration, the only unknown in Equation 95 is the update ΔX. Written in matrix form, the partial derivative terms from the n equations become an n-by-n matrix, J. This matrix of partial derivatives is known as the Jacobian, Equation 96. The system of equations given by Equation 95 becomes Equation 97. The X^0 subscripts indicate that the Jacobian and the right-hand side of Equation 97 are functions of the current guess.

$$J_{\bar{X}} = \begin{bmatrix} \frac{\partial f_1}{\partial X_1} & \frac{\partial f_1}{\partial X_2} & \cdots & \frac{\partial f_1}{\partial X_n} \\ \frac{\partial f_2}{\partial X_1} & \frac{\partial f_2}{\partial X_2} & \cdots & \frac{\partial f_2}{\partial X_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial X_1} & \frac{\partial f_n}{\partial X_2} & \cdots & \frac{\partial f_n}{\partial X_n} \end{bmatrix} \tag{96}$$

$$J_{\bar{X}^0} \Delta \bar{X} = -\bar{f}_{\bar{X}^0} \tag{97}$$

A.2.1 <u>Direct Solution</u>

In the case of the direct solution to the method of Lagrange multipliers (described in 5.1.1), a non-linear system of equations is derived from the partial derivatives of the Lagrangian with respect to its arguments. The vector of these partial derivatives is the gradient of the Lagrangian, Equation 98, which is given the symbol, $\nabla\tilde{f}$.

$$\nabla\tilde{f} = \begin{bmatrix} \frac{\partial\tilde{f}}{\partial X_1} \\ \frac{\partial\tilde{f}}{\partial X_2} \\ \vdots \\ \frac{\partial\tilde{f}}{\partial X_n} \end{bmatrix} \tag{98}$$

The partial derivatives of these equations (needed in the Taylor's series expansion) are the second partial derivatives of the Lagrangian with respect to its arguments. This creates an ($n+\ell$) by ($n+\ell$) matrix of second partial derivatives known as the Hessian, Equation 99, which takes the place of the Jacobian in Equation 97. Remember that the vector X, in this case, includes both the n design variables and ℓ Lagrange multipliers. Substituting, Equation 97 becomes Equation 100.

$$H_{\overline{X}} = \begin{bmatrix} \dfrac{\partial^2 \tilde{f}}{\partial X_1^2} & \dfrac{\partial^2 \tilde{f}}{\partial X_1 \partial X_2} & \cdots & \dfrac{\partial^2 \tilde{f}}{\partial X_1 \partial X_n} \\ \dfrac{\partial^2 \tilde{f}}{\partial X_2 \partial X_1} & \dfrac{\partial^2 \tilde{f}}{\partial X_2^2} & \cdots & \dfrac{\partial^2 \tilde{f}}{\partial X_2 \partial X_n} \\ \vdots & \vdots & \ddots & \vdots \\ \dfrac{\partial^2 \tilde{f}}{\partial X_n \partial X_1} & \dfrac{\partial^2 \tilde{f}}{\partial X_n \partial X_2} & \cdots & \dfrac{\partial^2 \tilde{f}}{\partial X_n^2} \end{bmatrix} \qquad \textbf{(99)}$$

$$H_{\overline{X}^0} \Delta \overline{X} = -\nabla \tilde{f}_{\overline{X}^0} \qquad \textbf{(100)}$$

This system of linear equations is solved using the efficient PLU decomposition method. The variables are updated and the process is repeated iteratively until the method converges to the solution. Because Newton's method has quadratic convergence, typically only a few iterations are required.

A.2.2 Solution with Substitution

In the case of the solution to the method of Lagrange multipliers with substitution (described in 5.1.2), a system of non-linear equations is derived from the constraint functions. The partial derivatives of these equations (needed in the Taylor's series expansion) are just the partial derivatives of the constraints with respect to the Lagrange multipliers. Remember that the vector *X*, in this case, includes only the ℓ Lagrange multipliers. This creates an ℓ by ℓ Jacobian matrix. Substituting, Equation 97 becomes Equation 101.

$$J_{\overline{\lambda}} \Delta \overline{\lambda} = -\overline{h}_{\overline{\lambda}} \qquad \textbf{(101)}$$

This system of linear equations is solved using the efficient PLU decomposition method. The variables are updated and the process is repeated iteratively until the method converges to the solution. Because Newton's method has quadratic convergence, typically only a few iterations are required.

In summary, Newton's method is a numerical technique that reduces a system of simultaneous non-linear equations to a system of simultaneous linear equations (which are much easier to solve). This simplification, however, requires an iterative solution procedure.

A.3. PLU Matrix Decomposition

PLU decomposition (or just LU decomposition) is a numerical technique for solving the system of simultaneous linear equations given by Equation 102. Here *A* is an *n*-by-*n* matrix and *x* and *b* are *n*-vectors. This technique is similar to the well-known Gaussian elimination, but with an important distinction. The data vector, *b*, does not need to be known while performing operations on the matrix *A*.

$$Ax = b \tag{102}$$

Ralston and Rabinowitz [125] describe the LU technique as equivalent to dividing the Gaussian elimination algorithm into two distinct processes. The first process is the triangular decomposition of *A* (splitting the matrix into two triangular matrices), which is independent of *b*. The second process is a combination of forward- and back-substitution to get the solution *x*. This means that once the first process is complete, we can solve Equation 102 for any right-hand side *b*. The procedure is as follows.

Write the matrix *A* as a product of two matrices *LU*, where *L* is lower triangular (zero elements above the diagonal) and *U* is upper triangular (zero elements below the diagonal). This is done because the solution of a triangular set of equations is quite trivial. Once the decomposition of A is known, solve the linear set Equation 103 in two steps. First, solve for the vector *y* such that *Ly=b*, Equation 104. Then use the result to solve *Ux=y*, Equation 105 for *x*. Substituting Equation 103, Equation 104, and Equation 105 into Equation 102, Equation 106 shows that this process is indeed the same as solving Equation 102. The first step, *Ly=b*, is solved by forward substitution and the second step, *Ux=y*, is solved by back-substitution.

$$A = LU \tag{103}$$

$$Ly = b \tag{104}$$

$$Ux = y \tag{105}$$

$$(A)x = (LU)x = L(Ux) = L(y) = b \tag{106}$$

LU decomposition is performed “in-place” by Crout’s Algorithm, which is described in Numerical Recipes [81]. “In place” means that the corresponding L_{ij} or U_{ij} can be stored in the location that A_{ij} used to occupy. This reduces memory requirements and is possible because each A_{ij} is used only once and never again. The diagonal elements of L are neither calculated nor stored (because they are always unity by design). The elements of the L and U matrix are calculated using Equation 107 and Equation 108.

$$U_{ij} = A_{ij} - \sum_{k=1}^{i-1} L_{ik} U_{kj} \tag{107}$$

$$L_{ij} = \frac{1}{U_{jj}} \left(A_{ij} - \sum_{k=1}^{j-1} L_{ik} U_{kj} \right) \tag{108}$$

The order in which these equations are solved, however, is important. Crout’s method fills in the matrix by columns from left to right, and within each column from top to bottom. Partial pivoting (the selection of the largest element in a column to reduce round-off error) is essential for the stability of Crout’s method. The P in PLU is a permutation matrix. This is an identity matrix which has had columns interchanged. Pre-multiplying by P has the same effect as performing the row interchanges necessary for partial pivoting.

In summary, the LU decomposition is an efficient algorithm for solving the system given by Equation 102. The efficiency comes from the forward- and back-substitution used in the solution. The advantage over Gaussian elimination is that the data vector b does not need to be known during the decomposition of A. This means that the decomposition can be performed once and then used many times with different data vectors.

A.4. Cholesky Matrix Decomposition

Cholesky decomposition is closely related to LU decomposition. If the matrix *A* happens to be both symmetric and positive-definite, then it has a special decomposition, Equation 109, called the Cholesky decomposition (sometimes referred to as "taking the square root" of a matrix). In this decomposition, the upper triangular matrix, *U*, is actually the transpose of the lower triangular matrix, *L*. Symmetric means that $A_{ij}=A_{ji}$ for all *i* and *j*. Positive-definite means that *A* has all positive eigenvalues.

$$A = LL^T \qquad \textbf{(109)}$$

A Fortran routine for performing the Cholesky decomposition is given by Press et al. [81]. The updating formulas (analogous to Equation 107 and Equation 108) are given by Equation 110 and Equation 111. These of these formulas guarantees that the matrix remains positive definite and nonsingular during the decomposition, even in the presence of finite round-off.

$$L_{ii} = \left(A_{ii} - \sum_{k=1}^{i-1} L_{ik}^2 \right)^{\frac{1}{2}} \qquad \textbf{(110)}$$

$$L_{ji} = \frac{1}{L_{ii}} \left(A_{ij} - \sum_{k=1}^{i-1} L_{ik} L_{jk} \right) \qquad \textbf{(111)}$$

When you can use it, Cholesky decomposition is about a factor of two faster than LU decomposition for solving the linear system, Equation 102. This is in part due to the symmetric nature of the problem, which requires calculating only one of the triangular matrices. Another advantage of this method is that Cholesky decomposition is extremely stable and does not require pivoting. In fact, the success or failure of the Cholesky decomposition routine is an efficient way to test whether or not a symmetric matrix is positive-definite.

APPENDIX B

SOURCE CODE

B.1. “mcp” Perl Module

```
package MCPost;

------------------------------------
  Get the Outputs from the PRO File
------------------------------------
sub read_pro{

        my($i, $j, $n, $m, names);
        my %mco=        %main::mco;
        my phases=     main::phases;
        my normals=    main::normals;
        my specials=   main::specials;
        my $pro_file=   _[0];

        --------------------------------
          Open the .pro
        --------------------------------
        open PRO,"<$pro_file" or die "Can't open $pro_file ->  $!\n";

        --------------------------
          Read header and title
        --------------------------
        for($i=0;$i<12;++$i){
                get_char(PRO,8);                 gets 8 characters at a time
        }                                        because of Fortran's char*8

        ----------------------------------
          Read number of print variables
        ----------------------------------
        m= get_int(PRO,1);                      must put integers in an array
        $m= m[0];                               even though there is only one
        return if $m==0;

        -------------------------------
          Discard the next two char*8
        -------------------------------
        get_char(PRO,8);
        get_char(PRO,8);

        --------------------------------
          Read names of print variables
        --------------------------------
        for($i=0;$i<$m;++$i){
                push names, get_char(PRO,8);            these are char*8 also
                $itime= $i if $names[$i]=~/^time {4}$/;  index of "time"
        }

        --------------------------------
          Get the numbers n at a time
        --------------------------------
        while (1){
                n= get_int(PRO,1);              have to put in array
                $n= n[0];                       why is this different than m?
                last if $n<$m;                   ran out of numbers
                $num= $n/$m;                     POST prints this many times
                d= get_double(PRO,$n);          here are the numbers we want
                for($j=0;$j<$num;++$j){
                     get all the numbers
                    foreach $outvar (names){
                        $outvar=~ tr/ //d;
                        $pro{$outvar}= shift d;
                    }
                    check for phase change/ get normals
                    if($pro{tdurp}==0){
                        $phase= shift phases;
                        foreach $name (normals){
                                $rec= $mco{$name};
                                if($rec->{critr} == $phase){
                                        $rec->{record}= $pro{$rec->{outvar}};
                                }
                        }
                    }
                    check for specials
                    foreach $name (specials){
                        $rec= $mco{$name};
                        if( $mco{time}== $pro{ $rec->{critr} } ){
                                $rec->{record}= $pro{$rec->{$outvar}};
                        }
                    }

                }
        }

        --------------------------------------------
          If last phase not found take final values
        --------------------------------------------
        foreach $phase (phases){
                foreach $name (normals){
                        $rec= $mco{$name};
                        if($rec->{critr} == $phase){
                                $rec->{record}= $pro{$rec->{outvar}};
                        }
                }
        }
}

------------------------------------
  Get the Outputs from the PRO File
------------------------------------
sub write_dat{

        my %mco=        %main::mco;
        my ocalcs=     main::ocalcs;
        my outputs=    main::outputs;
        my $seed=       _[0];
        my $dat=        _[1];


        open(DAT,">>$dat") or die "Can't open DAT, stopped";
        $data= pack('i',$seed);

        $calc_output= join ' ', ocalcs;
        foreach $output (outputs){
                if( $calc_output=~ m/(^| )$output( |$)/ ){
                        $expr= $mco{$output}->{expr};
                        $value= eval $expr;
                }
                else{
                        $value= $mco{$output}->{record};
                }
                if ($value eq ""){
                        warn qq[WARNING: "$output" not found... deleting from list\n];
                }
                else{
                        $data.= pack('d',$value);
                        push founds, $output;
                }
        }
        syswrite DAT,$data, 4+64*founds;
        main::outputs= founds;
        close(DAT);
}


----------------------------
  Reads n characters
  characters are 1 byte long
----------------------------
sub get_char{
        my $chars;                       my characters
        my $trash;                       unwanted garbage
        my $in= _[0];                   input file handle
        my $n= _[1];
        sysread $in, $trash, 4;          4 garbage bytes
        sysread $in, $chars, $n;         here are the good bytes
        sysread $in, $trash, 4;          4 more garbage bytes
        return $chars;
}

----------------------------
  Reads n integers
  integers are 4 bytes long
----------------------------
sub get_int{
        my integers;                    my integers (an array)
        my $trash;                       unwanted garbage
        my $buffer;                      temporary storage
        my $in= _[0];                   input file handle
        my $n= _[1];
        sysread $in, $trash, 4;          4 garbage bytes
        sysread $in, $buffer, 4*$n;      here are the good bytes
        sysread $in, $trash, 4;          4 more garbage bytes
        integers= unpack "i$n",$buffer;
        return integers;
}

----------------------------
  Reads n doubles (real*8's)
  doubles are 8 bytes long
----------------------------
sub get_double{
        my doubles;                     my doubles (an array)
        my $trash;                       unwanted garbage
        my $buffer;                      temporary storage
        my $in= _[0];                   input flie handle
        my $n= _[1];
        sysread $in, $trash, 4;          4 garbage bytes
        sysread $in, $buffer, 8*$n;      here are the good bytes
        sysread $in, $trash, 4;          4 more garbage bytes
        doubles= unpack "d$n",$buffer;
        return doubles;
}

 --------------------------------------------
  Definitions for use in regular expressions
 --------------------------------------------
sub definitions{
         -------------------
          Defines a number
         -------------------
        $a_number= q{
                (?x)                     allow whitespace & comments
                (?:\d+                    one or more digits
                \.?                       might have a dot (.)
                \d*                       zero or more digits
                (?:e[+-]?)?               could be exponential
                \d*                       zero or more digits
                |                             (or)
                \.                        a leading dot
                \d+                       one or more digits
                (?:e[+-])?                could be exponential
                \d*)                      zero or more digits
        };

         ---------------------------------------------------
          Defines Dispersion Sourcess (Input variables +/-x )
         ---------------------------------------------------
        $main::a_dispersion= q{
                (?xi)                     ws & comments, case insensitive

                \b(\w+)\b                 a word                ($1= name)
                \s*(?:=>|:|,|\s+)\s*      => or , or : or ws
                (}.$a_number.q{)          a valid number        ($2= nominal)
                \s*\+\s*/\s*-\s*          +/-
                (}.$a_number.q{)          a valid number        ($3= 3-sigma)
                \s*
                \(?(%\.?|\w+\.?)\)?       % or word             ($4= units)
                \s*(?:=>|:|,|\s+)\s*      => or , or : or ws
                (uniform|                 uniform or
                 triangle|                triangle or
                 triangular|              triangular or
                 normal|                  normal or
                 gaussian)                gaussian              ($5= distribution)
                \s*\;?\s*                 optional semi-colon
        };
         ------------------------------------------
          Defines Dispersions (special case: +x/-y )
         ------------------------------------------
        $main::a_special_dispersion= q{
                (?xi)                     ws & comments, case insensitive

                \b(\w+)\b                 a word                ($1= name)
                \s*(?:=>|:|,|\s+)\s*      => or , or : or ws
                (}.$a_number.q{)          a valid number        ($2= nominal)
                \s*\+\s*                  +
                (}.$a_number.q{)          a valid number        ($3= +3-sigma)
                \s*/\s*                   /
                \s*-\s*                   -
                (}.$a_number.q{)          a valid number        ($4= -3-sigma)
                \s*
                \(?(%\.?|\w+\.?)\)?       % or word             ($5= units)
                \s*(?:=>|:|,|\s+)\s*      => or , or : or ws
                (uniform|                 uniform or
                 triangle|                triangle or
                 triangular|              triangular or
                 normal|                  normal or
                 gaussian)                gaussian              ($6= distribution)
                \s*\;?\s*                 optional semi-colon
        };
         ------------------------------------------------
          Defines Output Variables ( begining of phase x)
         ------------------------------------------------
        $main::an_outvar= q{
                (?xi)                     ws & comments, case insensitive

                \b(\w+)\b                 a word                ($1= name)
                \s*(?:=>|:|,|\s+)\s*      => or , or : or ws
                \b(\w+)\b                 a word                ($2= POST variable)
                \s*(?:|at)\s*             or at
                \s*phase\s*               phase
                (\d+)                     a number              ($3= phase number)
                \s*(?:=>|:|,|\s+)\s*      => or , or : or ws
                \(?(%\.?|\w+\.?)?\)?      % or word (optional)  ($4= units)
                \s*\;?\s*                 optional semi-colon
                (?:[/]/?\s*(\S.*\S))?   optional comment      ($5= description)
        };
         ---------------------------------------------
          Defines Special Output Variables ( time x)
         ---------------------------------------------
        $main::a_special_outvar= q{
                (?xi)                     ws & comments, case insensitive

                \b(\w+)\b                 a word                ($1= name)
                \s*(?:=>|:|,|\s+)\s*      => or , or : or ws
                \b(\w+)\b                 a word                ($2= POST variable)
                \s*(?:|at)\s*             or at
                \s*time\s*                time
                \b(\w+)\b                 a word                ($3= special variable)
                \s*(?:=>|:|,|\s+)\s*      => or , or : or ws
                \(?(%\.?|\w+\.?)?\)?      % or word (optional)  ($4= units)
                \s*\;?\s*                 optional semi-colon
                (?:[/]/?\s*(\S.*\S))?   optional comment      ($5= description)
        };
         --------------------------------------------
          Defines Calculated Variables (Eval)
         --------------------------------------------
        $main::a_calculated_var= q{
                (?x)                      allow ws & comments

                \b(\w+)\b                 a word                ($1= name)
                \s*=\s*                   =
                (?!>)                     (not a =>)
                (.+)                      stuff                 ($2= commands)
                \s*\;\s*                  mandatory semi-colon!
                \(?(%\.?|\w+\.?)?\)?      % or word (optional)  ($3= units)
                \s*
                (?:[/]/?\s*(\S.*\S))?   optional comment      ($4= description)
        };
         --------------------------------------------
          Defines an Include File Input
         --------------------------------------------
        $main::an_include_input= q{
                (?xi)                     ws & comments, case insensitive

                \*                         *
                \s*include\s*             include
                \'?                       optional '
                ([\w\/]+)                 file stuff            ($1= name)
                (.\w+)?                   optional extension    ($2= ext)
                \'?                       optional '
                \s*(?:=>|:|,|\s+)\s*      => or , or : or ws
                \s*1\s*                   1
                (?:..|:|-|to)             .. , : , - , or to
                \s*(\d+)\s*               a number              ($3= max)
                \s*\;?\s*                 optional semi-colon
                (?:[/]/?\s*(\S.*\S))?   optional comment      ($4= description)
        };
         --------------------------------------------
          Defines an Input Marker (*** whatever ***)
         --------------------------------------------
        $main::a_marker= q{
                (?x)                      allow ws & comments

                \*{2,3}                    **, ***
                \s*\b(\w+)\b\s*           a word                ($1= name)
                \*{2,3}                    **, ***
        };
         --------------------------------------------
          Defines an Include File Marker
         --------------------------------------------
        $main::an_include= q{
                (?xi)                     ws & comments, case insensitive

                \*                         *
```

```
                \s*include\s*             include
                \*{2,3}\s*                 **, ***
                \'?                       optional '
                ([\w\/.]+)                file stuff              ($1= name)
                \'?                       optional '
                \s*\*{2,3}                 **, ***
        };
         --------------------------------------------
          Defines an Event
         --------------------------------------------
        $main::an_event= q{
                (?xi)                     ws & comments, case insensitive

                \n(?!c)                    not a comment
                \s*event\s*=\s*            event =
                (\d+)                      a number               ($1= phase)
                \s*,                       ,
        };
}

 ------------------------------------------------------
  Determine distance from Reference Lat, Long
 ------------------------------------------------------
sub distance{
        my($lat,$long,$latref,$longref,$ae,$be)= _;
          ae= equatorial radius
          be= polar radius
        my($e,$i1,$j1,$k1,$i2,$j2,$k2,$d);
        my $rpd= 3.141592654/180.0;
        $lat*= $rpd;
        $long*= $rpd;
        $latref*= $rpd;
        $longref*= $rpd;
        $e= sqrt(1-($be/$ae)**2);
        $i1= abs($ae/sqrt(1-$e**2*sin($latref)**2))*cos($latref)*cos($longref);
        $j1= abs($ae/sqrt(1-$e**2*sin($latref)**2))*cos($latref)*sin($longref);
        $k1= abs($ae*(1-$e**2)/sqrt(1-$e**2*sin($latref)**2))*sin($latref);
        $i2= abs($ae/sqrt(1-$e**2*sin($lat)**2))*cos($lat)*cos($long);
        $j2= abs($ae/sqrt(1-$e**2*sin($lat)**2))*cos($lat)*sin($long);
        $k2= abs($ae*(1-$e**2)/sqrt(1-$e**2*sin($lat)**2))*sin($lat);
        $d= sqrt(($i2-$i1)**2+($j2-$j1)**2+($k2-$k1)**2);
        return $d;
}

1;
```

B.2. <u>"mcp" Executive</u>

```
! /usr/sbin/perl

use MCPost;
use RandDist;
srand();


 --------------------------------------
  Default Files
 --------------------------------------
$sim= $ARGV[0];
$post= "../post";
$tpl= "$sim.tpl";         template file (POST .inp w/ markers)
$rnd= "$sim.rnd";         random seed file
$oat= "$sim.oat";         one-at-a-time seed file
$rrn= "$sim.rrn";         rerun seed file
$out= "$sim.out";         output file
$dat= "$sim.dat";         temporary output data file
$mat= "$sim.mat";         matlab .mat file (binary data)
$mmf= "$sim.m";           matlab .m file (plot definitions)
```

```
 -----------
  Defaults
 -----------
$nseg= 10;                number of parallel processes
$nruns= 800;              number of montecarlo runs
$nominal= 0;
$monte= 1;
$ovat=0;
$rerun=0;


MCPost::definitions();

 -------------------------------------
  Process Monte-Carlo Input File (.mci)
 -------------------------------------
$matlab= 0;
$perl= 0;
open(MCI,"<$sim.mci") or die "Can't open MCI, stopped";          monte carlo input
while(<MCI>){
        next if m/^\s*(\|\/)(?!!)\/?/;                          next if a comment

        ($post =$1, print "$post\n")  if m/(?:\!|set\s+post)\s*(\S*post\S*)/i;
        $tpl  =$1  if m/set\s+template\s*(\S*\.tpl)/i;
        $rnd  =$1  if m/set\s+seeds\s*(\S*\.rnd)/i;
        $oat  =$1  if m/set\s+ovat\s*(\S*\.oat)/i;
        $rrn  =$1  if m/set\s+rerun\s*(\S*\.rrn)/i;
        $out  =$1  if m/set\s+output\s*(\S*\.out)/i;
        $dat  =$1  if m/set\s+data\s*(\S*\.dat)/i;
        $mat  =$1  if m/set\s+mat\s*(\S*\.mat)/i;
        $mmf  =$1  if m/set\s+mfile\s*(\S*\.m)/i;

        $nseg  =$1  if m/set\s+nseg\s*(\S*)/i;
        $nruns  =$1  if m/set\s+nruns\s*(\S*)/i;

        $matlab=1, next if  m/<matlab>/i;                        matlab start
        $matlab=0, next if ( $matlab and m/<\\matlab>/i );       matlab stop
        $mattext.= $_, next if $matlab;                          save matlab text

        $perl=1, next if  m/<perl>/i;                            perl start
        $perl=0, next if ( $perl and m/<\\perl>/i );             perl stop
        $perltext.= $_, next if $perl;                           save perl text

        if(m/$an_outvar/){
                $name= {};
                $mco{$1}= $name;
                $name->{outvar}=  $2;
                $name->{critr}=   $3;
                $name->{units}=   $4;
                $name->{comment}= $5;
                push normals, $1;
                push outputs, $1;
        }
        if(m/$a_special_outvar/){
                $name= {};
                $mco{$1}= $name;
                $name->{outvar}=  $2;
                $name->{critr}=   $3;
                $name->{units}=   $4;
                $name->{comment}= $5;
                push specials, $1;
                push outputs, $1;
        }
        if(m/$a_calculated_var/){
                $name= {};
                $mco{$1}= $name;
                $name->{expr}=    $2;
                $name->{units}=   $3;
                $name->{comment}= $4;
                push calcs, $1;
        }
        if(m/$a_dispersion/){
                $name= {};
                $mci{$1}= $name;
                $name->{nominal}= $2;
                $name->{actual}=  $2;
                $name->{units}=   $4;
                $name->{distr}=   $5;
                if($name->{units} eq '%'){
                        $name->{high}= $2*(1+$3/100);
                        $name->{low}=  $2*(1-$3/100);
                }
                else{
                        $name->{high}= $2+$3;
                        $name->{low}=  $2-$3;
                }
        }
        if(m/$a_special_dispersion/){
                $name= {};
                $mci{$1}= $name;
                $name->{nominal}= $2;
                $name->{actual}=  $2;
                $name->{units}=   $5;
                $name->{distr}=   $6;
                if($name->{units} eq '%'){
                        $name->{high}= $2*(1+$3/100);
                        $name->{low}=  $2*(1-$4/100);
                }
                else{
                        $name->{high}= $2+$3;
                        $name->{low}=  $2-$4;
                }
        }
        if(m/$an_include_input/){
                $name= {};
                $mcf{$1.$2}=  $name;
                $name->{file}=    $1;
                $name->{ext}=     $2;
                $name->{max}=     $3;
                $name->{comment}= $4;
                push includes,   $1.$2;
        }
}
close(MCI);


 ----------------------------------------------------
  Determine if Calculated variable is input or output
 ----------------------------------------------------
CALC: foreach $calcvar (calcs){
        $expr= $mco{$calcvar}->{expr};
        foreach $outvar (normals, specials){
                if($expr =~ m/$outvar/){
                        push ocalcs, $calcvar;
                        push outputs, $calcvar;
                        foreach $name (keys %mco){
                                $replacement= '$mco{'.$name.'}{record}';
                                $expr =~ s/$name/$1$replacement$2/g;
                        }
                        $mco{$calcvar}->{expr}=$expr;
                        next CALC;
                }
        }
        push icalcs, $calcvar;
}

 ---------------------------------------------
  Redirect STDOUT & STDERR
 ---------------------------------------------
`cat $sim.mci $tpl > $out`;
open(STDERR,">>$out") or die "Can't open OUT, stopped";  open output
select STDERR;                                          make it STDOUT too
```

```
 --------------------------------------------
  Slurp the template POST deck
 --------------------------------------------
open(TPL,"<$tpl") or die "Can't open TPL, stopped";      open template
undef $/;                                                change line seperator
$post_deck= <TPL>;                                       slurp the whole thing!
close(TPL);                                              close template
$/= "\n";                                                reset line seperator

 --------------------------------------------
  Check to see if any markers aren't found
  or any inputs aren't used
 --------------------------------------------
markers= $post_deck =~ m/$a_marker|$an_include/g;       get all markers
$inputs= join ' ', keys %mci, keys %mcf, icalcs;
foreach $marker (markers){
        next if $marker eq "";
        warn qq[DANGER!: "$marker" not found in input\n]
                if $inputs!~ m/(^| )$marker( |$)/;
}
$markers= join ' ', markers;
foreach $input (keys %mci, keys %mcf, icalcs){
        warn qq[WARNING: "$input" input not used\n]
                if $markers!~ m/(^| )$input( |$)/;
}

--------------------------------
  Get the Phases
--------------------------------
phases= $post_deck =~ m/$an_event/g;    get all phases

--------------------------------
  Create Seed file
-------------------------------
($nominal= 1, $monte=0) if $ARGV[1] eq "nominal";
($ovat= 1, $monte=0) if $ARGV[1] eq "ovat";
($rerun= 1, $monte=0) if $ARGV[1] eq "rerun";

print "$nominal\t$monte\t$ovat\t$rerun\n";

$nruns= 0 if $nominal;
(monte_seeds(), $seed_file= $rnd) if ($monte or $nominal);
(ovat_seeds(), $seed_file= $oat) if $ovat;
(rerun_seeds(), $seed_file= $rrn) if $rerun ;

 --------------------------------
  Open SEED file (.rnd/.rrn/.oat)
 --------------------------------
open(SEEDS,"<$seed_file") or die "Can't open SEEDS, stopped";
sysread SEEDS, $buffer, 4*3;
i= unpack('i3',$buffer);
$length= i[0];
$n=      i[1];
$m=      i[2];
sysread SEEDS, $stamp, $length;

 ---------------------------------------------------
  Process control:  launches & maintains nseg processes
 ---------------------------------------------------
$i= 1;                   intialize counters
$count= 1;               intialize counters
open(DAT,">$dat") or die "Can't open DAT, stopped";

 -----------------------------------
 Run nominal case
 -----------------------------------
sysread(SEEDS, $buffer, 4+8*$m) or die;                  read numbers from SEEDS
numbers= unpack("id$m",$buffer);                        put the numbers in this array
$seed= shift numbers;                                   the seed number is first
```

```
foreach $name (keys %mci){
        $mci{$name}{actual}= shift numbers;             store the rest
}
run_post();

PROCESS: while(1){

         ---------------------------------
         Get actual values from SEEDS file
         ---------------------------------
        sysread(SEEDS, $buffer, 4+8*$m) or next PROCESS;         read numbers from SEEDS
        numbers= unpack("id$m",$buffer);                        put the numbers in this
array
        $seed= shift numbers;                                   the seed number is first
        foreach $name (keys %mci){
                $mci{$name}{actual}= shift numbers;             store the rest
        }

        FORK: {
                if($pid= fork){
                         ------------------------------
                         parent
                         child pid is available in $pid
                         ------------------------------
                        $iproc{$pid}= $i;
                        (++$i,++$count, redo PROCESS) if $count<$nseg;
                }
                elsif(defined $pid){
                         ------------
                         child
                         ------------
                        run_post();
                        exit 0;
                }
                elsif($! =~ /No more process/){
                         -----------------------
                         recoverable fork error
                         -----------------------
                        sleep 1;                        sleep one second
                        redo FORK;                      try the fork again
                }
                else{
                         ----------------------
                         non-recoverable error
                         ----------------------
                        die "Can't fork: $!";
                }
        }
}
continue{
        $pid= wait;
        last PROCESS if $pid==-1;
        $i= $iproc{$pid} if $iproc{$pid};
        delete $iproc{$pid};
}

close(SEEDS);

------------------------------------
  Read Outputs from the DAT File
------------------------------------
print "reading DAT file...\n";
open(DAT,"<$dat") or die "Can't open DAT, stopped";
$array_size= 0;
RUN: while(1){
        sysread(DAT, $buffer, 4) or last RUN;
        seeds= unpack "i", $buffer;
        ++$array_size;
        $n=$array_size;
        push runs, seeds[0];
        foreach $output (outputs){
                sysread(DAT, $buffer, 8) or last RUN;
                doubles= unpack "d", $buffer;
                $value= doubles[0];
                push $output, $value;
                if ($n==1) {
                        $mean{$output}= $value;
                        $variance{$output}= 0;
                        $max{$output}= $value;
                        $min{$output}= $value;
                        $nom{$output}= $value;
                }
                elsif($n>1){
                        $variance{$output}*= ($n-2)/($n-1);
                        $variance{$output}+=($mean{$output}-$value)*($mean{$output}-
$value)/$n;
                        $mean{$output}*= ($n-1.0)/$n;
                        $mean{$output}+= (1.0/$n)* $value;
                        if ($value > $max{$output} ){ $max{$output}= $value; }
                        if ($value < $min{$output} ){ $min{$output}= $value; }
                }
        }
}
close(DAT);

--------------------------------
  Write out statistics
--------------------------------
if($monte or $nominal){
        print  "Name\tNom\tMin\tMax\tMean\tVariance\n";
        print STDERR "Name\tNom\tMin\tMax\tMean\tVariance\n";
        foreach $output (outputs){

                print  "$output\t";
                print  "$nom{$output}\t";
                print  "$min{$output}\t";
                print  "$max{$output}\t";
                print  "$mean{$output}\t";
                print  "$variance{$output}\n";

                print STDERR "$output\t";
                print STDERR "$nom{$output}\t";
                print STDERR "$min{$output}\t";
                print STDERR "$max{$output}\t";
                print STDERR "$mean{$output}\t";
                print STDERR "$variance{$output}\n";
        }
}

--------------------------------
  Write the data to .mat
--------------------------------
print "writing MATLAB file...\n";
open(MAT,">$mat") or die "Can't open MAT, stopped";

chop($header= "Created by pmat in MATLAB 5.0 MAT-File format: ".`date`);
$how_many= 124 - length($header);
$pad= ' ' x $how_many;
$header .= $pad;                                pad with space to 124
$cc= 256;                                       copy write symbol?
syswrite( MAT, $header, 124) or die;            padded header
syswrite( MAT, pack( 's', $cc), 2) or die;      cc (must have)
syswrite( MAT, "MI", 2) or die;                 Mathworks Inc.

push outputs, "runs";
foreach $name (outputs){

        --------------------------------
          Determine name length
        --------------------------------
        $name=~ tr/ //d;                        chop all spaces first
        $name_length= length $name;             get name length
        if($name_length>31){                      max name length is 31
                $name= substr($name,0,31);
                $name_length= 31;
        }

        ------------------------------------
          Name padded to fit into x8 bytes
        ------------------------------------
        SWITCH: {
                $name_bytes= 32, last SWITCH if $name_length>16;
                $name_bytes= 16, last SWITCH if $name_length>8;
                $name_bytes=  8, last SWITCH if $name_length>4;
                $name_bytes=  4;
        }

        ---------------------
          Matrix dimensions
        ---------------------
        $rows= $array_size;
        $columns= 1;                                      only one column
        $data_bytes= $rows*$columns*8;                    data is 8 byte doubles
        $record_size= $data_bytes + $name_bytes;          this many + ...
        $record_size+= 48 if $name_bytes>4;               12 ints * 4 bytes= 48
        $record_size+= 44 if $name_bytes==4;              11 ints * 4 bytes= 44

        -------------------------------------------------------
          Apparently random sequence of 10 numbers (must have)
        -------------------------------------------------------
        numbers= split " ", "14 $record_size 6 8 6 0 5 8 $rows $columns";
        foreach $i (numbers){
            syswrite( MAT, pack('i',$i), 32) or die;              first 10 integers
        }

        -------------------------------------------------------------------
          Write name length to OUT file (special case if name length <= 4)
        -------------------------------------------------------------------
        if($name_bytes==4){
            syswrite( MAT, pack('s',$name_length), 16) or die;   name length
            syswrite( MAT, pack('s',1), 16) or die;              flag
        }

        else{
            syswrite( MAT, pack('i',1), 32) or die;              flag
            syswrite( MAT, pack('i',$name_length), 32) or die;   name length
        }

        ----------------------------------------------------
          Write name to OUT file (must pad with zero bytes)
        ----------------------------------------------------
        $how_many= $name_bytes - $name_length;
        $pad= "\0" x ($how_many);
         syswrite( MAT, $name, $name_length) or die;             the name
        (syswrite( MAT, $pad, $how_many) or die) if $how_many;   the padding

        -------------------------------------------
          Write OUT 9 and length of data in bytes
        -------------------------------------------
        syswrite( MAT, pack('i',9), 32) or die;                  9 means doubles
        syswrite( MAT, pack('i',$data_bytes), 32) or die;        the data bytes

        ----------------------------------------------------------------
          Finally write the matrix by columns (assumed only one column)
        ----------------------------------------------------------------
        for($i=0;$i<$rows*$columns;++$i){
            syswrite( MAT, pack('d',$name[$i]), 64) or die;
        }

}
{
        ---------------------------------------
          Save Input variable names in "ivars"
```

```
---------------------------------------
$data= '';
$maxlen= 0;
strings= keys %mci;
foreach $string (strings){
        $maxlen= length $string if length $string > $maxlen;
}
for($i=0;$i<$maxlen;++$i){
        foreach $string (strings){
                $data.= "\0";
                $data.= substr $string, $i, 1 if $i<length $string;
                $data.= ' ' if $i>=length $string;
        }
}

--------------------------------
  Determine name length
--------------------------------
$name= "ivars";
$name_length= length $name;              get name length
if($name_length>31){                     max name length is 31
        $name= substr($name,0,31);
        $name_length= 31;
}

------------------------------------
  Name padded to fit into x8 bytes
------------------------------------
SWITCH: {
        $name_bytes= 32, last SWITCH if $name_length>16;
        $name_bytes= 16, last SWITCH if $name_length>8;
        $name_bytes=  8, last SWITCH if $name_length>4;
        $name_bytes=  4;
}

------------------------------------
  Matrix dimensions
------------------------------------
$rows= strings;
$columns= $maxlen;
$data_bytes= $rows*$columns*2;
$record_size= $data_bytes;
if($data_bytes%8){
        $data_pad= (8-$data_bytes%8);
        print "padding $data_pad\n";
        $record_size+= $data_pad;
}
$record_size+= $name_bytes + 48 if $name_bytes>4;        12 ints * 4 bytes = 48
$record_size+= $name_bytes + 44 if $name_bytes==4;       11 ints * 4 bytes = 44

----------------------------------------------------
  Apparently random sequence of numbers (must have)
----------------------------------------------------
numbers= split " ", "14 $record_size 6 8 4 0 5 8 $rows $columns";
foreach $i (numbers){
        syswrite( MAT, pack('i',$i), 32) or die;
}

-------------------------------------------------------------------
  Write name length to MAT file (special case if name length <= 4)
-------------------------------------------------------------------
if($name_bytes==4){
    syswrite( MAT, pack('s',$name_length), 16) or die;   name length
    syswrite( MAT, pack('s',1), 16) or die;              flag
}
else{
    syswrite( MAT, pack('i',1), 32) or die;              flag
    syswrite( MAT, pack('i',$name_length), 32) or die;   name length
}

----------------------------------------------------
```

```
          Write name to MAT file (must pad with zero bytes)
        ----------------------------------------------------
        $how_many= $name_bytes - $name_length;
        $pad= "\0" x ($how_many);
         syswrite( MAT, $name, $name_length) or die;              the name
        (syswrite( MAT, $pad, $how_many) or die) if $how_many;   the padding

        -------------------------------------------
          Write MAT 4 and length of data in bytes
        -------------------------------------------
        syswrite( MAT, pack('i',4), 32) or die;
        syswrite( MAT, pack('i',$data_bytes), 32) or die;

        ----------------------------------------------------------------
          Finally write the matrix by columns (assumed only one column)
        ----------------------------------------------------------------
        syswrite( MAT, $data, $data_bytes) or die;
        if (defined $data_pad ){syswrite( MAT, "\0" x $data_pad, $data_pad ) or die};
}

--------------------------------
  Write the .m file
--------------------------------
print "writing .m file...\n";
open(MMF,">$mmf") or die "Can't open MMF, stopped";
print MMF "load $sim\n";
$i=0;
foreach $output (outputs){
        ++$i;
        print MMF "\nfigure($i)\n";
        print MMF "hist($output,20)\n";
        if($mco{$output}{comment} eq ""){
                print MMF "title(\'Histogram\')\;\n";
        }
        else{
                print MMF "title(\'Histogram - $mco{$output}{comment}\')\;\n";
        }
        if($mco{$output}{units} eq ""){
                print MMF "xlabel(\'$output\')\;\n";
        }

        else{
                print MMF "xlabel(\'$output ($mco{$output}{units})\')\;\n";
        }
        print MMF "ylabel('number of cases')\;\n";
        print MMF "grid\n";
}
print MMF "$mattext\n";
close(MMF);

 ------------------ ***** subroutines ***** -------------------------------

 --------------------
 Run a POST deck
 --------------------
sub run_post{
        $new_deck= $post_deck;
       $tempdir= 'temp'.$seed;                    unique name for temp dir
       `mkdir $tempdir` unless -e $tempdir;       make a temporary directory
       chdir $tempdir;                            change to that directory
       `rm *` if -e "profilb";                    remove .pro if it exists
       $inp= "T$seed.inp";                        temp INP file name
       $out= "T$seed.out";                        temp OUT file name
       $pro= "profilb";                           temp PRO file name
       open(INP,">$inp")                          open INP
               or die "Can't open INP, stopped";  or die
       $new_deck =~                               in the template,
               s/$a_marker/$mci{$1}{actual}/g;    replace all markers
       print INP "$new_deck";                     write whole thing at once
       close(INP);                                close it
       `$post < $inp > $out`;                     run POST
       MCPost::read_pro($pro);                    read PRO file
       `rm *` unless $seed==0;                    clean-up
       chdir "..";                                get out of temp dir
       MCPost::write_dat($seed,$dat);             append outputs to DAT file
       `rm -r $tempdir` unless $seed==0;          clean-up
       print "run $seed\n";
}

 -------------------------------------------
 Create Monte Carlo Random Seed File (.rnd)
 -------------------------------------------
sub monte_seeds{
        chop($stamp= `date`);
        $length= length $stamp;
        $n= $nruns;
        $m= keys %mci;
        open(RND,">$rnd") or die "Can't open RND, stopped";
        syswrite RND, pack('i',$length), 4;
        syswrite RND, pack('i',$n),      4;
        syswrite RND, pack('i',$m),      4;
        syswrite RND, $stamp, $length;
        for($i=0;$i<=$n;++$i){
                undef dp_values;
                foreach $name (keys %mci){
                        $input= $mci{$name};
                        if($i==0){
                                push dp_values, $input->{nominal};
                        }
                        elsif($input->{distr}=~ m/(normal|gaussian)/i){
                                $value= $input->{low}+($input->{high}-$input->{low})
                                        *RandDist::normal();
                                push dp_values, $value;
                        }
                        elsif($input->{distr}=~ m/(triangle|triangular)/i){
                                $x_bar=($input->{nominal}-$input->{low})
                                        /($input->{high}-$input->{low});
                                $value= $input->{low}+($input->{high}-$input->{low})
                                        *RandDist::triangular($x_bar);
                                push dp_values, $value;
                        }

                        else{
                                $value= $input->{low}+($input->{high}-$input->{low})
                                        *rand();
                                push dp_values, $value;
                        }
                }
                syswrite RND, pack("id$m",$i, dp_values), 4+ 8*$m;
        }
        close(RND);
}

 -------------------------------------------
 Create OVAT Seed File (.oat)
 -------------------------------------------
sub ovat_seeds{
        chop($stamp= `date`);
        $length= length $stamp;
        $m= keys %mci;
        $n= 2*$m;
        open(RND,">$oat") or die "Can't open RND, stopped";
        syswrite RND, pack('i',$length), 4;
        syswrite RND, pack('i',$n),      4;
        syswrite RND, pack('i',$m),      4;
        syswrite RND, $stamp, $length;
         --------------
         nominal case
         --------------
        $i=0;
        undef dp_values;
        foreach $name (keys %mci){
                $input= $mci{$name};
                push dp_values, $input->{nominal};
        }
        syswrite RND, pack("id$m",$i, dp_values), 4+ 8*$m;
         --------------
         high cases
         --------------
        foreach $invar (keys %mci){
                ++$i;
                print "$i\t$invar\n";
                undef dp_values;
                foreach $name (keys %mci){
                        $input= $mci{$name};
                        if($name eq $invar){
                                push dp_values, $input->{high};
                        }
                        else{
                                push dp_values, $input->{nominal};
                        }
                }
                syswrite RND, pack("id$m",$i, dp_values), 4+ 8*$m;
        }
         --------------
         low cases
         --------------
        $i=0;
        foreach $invar (keys %mci){
                --$i;
                undef dp_values;
                foreach $name (keys %mci){
                        $input= $mci{$name};
                        if($name eq $invar){
                                push dp_values, $input->{low};
                        }
                        else{
                                push dp_values, $input->{nominal};
                        }
                }
                syswrite RND, pack("id$m",$i, dp_values), 4+ 8*$m;
        }
        close(RND);
}

 ------------------------------------------------------
 Create Rerun Seed File (.rrn) from (.rnd) and rerun
 ------------------------------------------------------
sub rerun_seeds{
        open(IN,"<$rnd") or die "Can't open RND, stopped";
        open(RND,">$rrn") or die "Can't open RND, stopped";
        sysread IN, $buffer, 4*3;
        i= unpack('i3',$buffer);
        $length= i[0];
        $n=      i[1];
        $m=      i[2];
        sysread IN, $stamp, $length;
        sysread IN, $buffer, 4+ 8*$m;
        $size= rerun;
        syswrite RND, pack('i',$length), 4;
        syswrite RND, pack('i',$size),   4;
        syswrite RND, pack('i',$m),      4;
        syswrite RND, $stamp, $length;
        syswrite RND, $buffer, 4+ 8*$m;
        for($i=1;$i<=$n;++$i){
                sysread IN, $buffer, 4+ 8*$m;
                numbers= unpack("id$m",$buffer);
                $seed= shift numbers;
                foreach $run (rerun){
                        syswrite( RND, $buffer, 4+ 8*$m) if $run==$seed;
                }
        }
        close(IN);
        close(RND);
}
```

B.3. “RandDist” Perl Module

```
package RandDist;

$seed=time();

--------------------------------------------------------------------

       Uniform random number generator (0,1)

       For the sake of commonality in calling

--------------------------------------------------------------------
sub uniform{
        return ran1($seed);
}

--------------------------------------------------------------------

       Discrete random number generator (0:1:1/$max)

       Coded by:  David Way  8/26/00

--------------------------------------------------------------------
sub discrete{

        my $max= _[0];
        my $d= uniform()*$max;

        return (($d%$max)/$max or 1);
}


--------------------------------------------------------------------

       Triangular random number generator for interval (0,1)

       Coded by:  David Way  8/25/00

--------------------------------------------------------------------
sub triangle{

        my $x_bar= _[0];
        my $y= uniform();

        return ($y<=$x_bar)?sqrt($y*$x_bar):1-sqrt((1-$y)*(1-$x_bar));
}


--------------------------------------------------------------------

       Normal random number generator for interval (0,1)

       This routine follows the article by kinderman and ramage
       "Computer Generation of Normal Random Variables" in the
       Journal of the American Statisitical Association
       Volume 71, number 356, pp.893-896.

       Coded in Perl by:  David Way  7/22/99

       Includes subroutines:
               triangular_center, accept_reject, and gen_tail

--------------------------------------------------------------------
sub normal{
```

```
my $u= uniform();
my $xi= 2.216035867166471;
my $x;

my const = (
         .884070402298758,       const[0]
         .911312780288703,       const[1]
         .958720824790463,       const[2]
         .973310954173898,       const[3]

         .479727404222441,       const[4]

        -.630834801921960,       const[5]
         .755591531667601,       const[6]
         .034240503750111,       const[7]

        1.105473661022070,       const[8]
         .872834976671790,       const[9]
         .049264496373128,       const[10]

        -.595507138015940,       const[10]
         .805577924423817,       const[11]
         .053377549506886,       const[12]
);

SWITCH:{
        if ($u < const[0]) {
                $x = triangular_center($u);
                last SWITCH;
        }
        if ($u >= const[0] && $u < const[1] ) {
                $x = accept_reject(
                        const[4],
                        const[11],
                        const[12],
                        const[13],
                );
                last SWITCH;
        }
        if ($u >= const[1] && $u < const[2] ) {
                $x = accept_reject(
                        const[4],
                        const[8],
                        const[9],
                        const[10],
                );
                last SWITCH;
        }
        if ($u >= const[2] && $u < const[3] ) {
                $x = accept_reject(
                        $xi,
                        const[5],
                        const[6],
                        const[7],
                );
                last SWITCH;
        }
        if ($u >= const[3]){
                $x = gen_tail($u);
                last SWITCH;
        }
}


return ($x+3)/6;         transforms interval to (0,1)



-------------------------------------------------------------------
```

```
               Generates the triangular center (a multiple of the sum
               of two uniform distributions).

        -------------------------------------------------------------------
        sub triangular_center{
                my $u= _[0];
                my $v= uniform();
                my $const= 1.131131635444180;

                return $xi*($const*$u+$v-1);
        }

        -------------------------------------------------------------------

               Acceptance-Rejection techniques on subintervals of (-xi,xi)
               for the difference between the standard normal density and
               the triangular density.

        -------------------------------------------------------------------
        sub accept_reject{
                my ($A,$B,$C,$D) = _;
                my $pi= 3.141592;
                my $const= .180025191068563;
                my( $v, $w, $z, $min, $max, $t, $f);

                LOOP:{
                        $v= uniform();
                        $w= uniform();
                        $z= $v- $w;
                        $min= ($v<$w)?$v:$w;
                        $max= ($v>$w)?$v:$w;
                        $t= $A + $B* $min;

                        last LOOP if ($max <= $C);
                        $f= exp(-0.5*$t**2)/sqrt(2*$pi)-$const*($xi-abs($t));
                        last LOOP if $D*abs($z)<=$f;
                        redo LOOP;
                }

                return ($z<0)?$t:-1*$t;
        }

        -------------------------------------------------------------------

               Generates the tail (for deviates larger than in absolute
               value than xi) using the tail algorithm of Marsaglia as
               modified by Ahrens and Dieter.

        -------------------------------------------------------------------
        sub gen_tail{
                my $u= shift(_);

                my $const= .986655477086949;
                my ($v, $w, $t);

                LOOP:{
                        $v= uniform();
                        $w= uniform();
                        $t= 0.5*$xi**2-log($w);


                        redo LOOP if ($t*$v**2 > 0.5*$xi**2);
                }

                return ($u<$const)?sqrt(2*$t):-1*sqrt(2*$t);
        }

}

      DOUBLE PRECISION FUNCTION RAN1(idum)
-------------------------------------------------------------------
```

```
  this is a double precision random number generator taken
  from "Numerical Recipes", pp. 196-97; it is based on three linear
  congruential generators; the values of m, a, and c for these
  generators are taken from the table on p. 198; VAX overflow occurs
  at approximately 10e38 or 10e-38; the values for m, a, and c are
  taken from the case where overflow occurs at 2e30 so we are well
  within the range for which these values can be used; this routine
  is called RAN1 and is a portable uniform random number generator


       Coded in Perl by:  David Way

-------------------------------------------------------------------
$ix1= 0;
$ix2= 0;
$ix3= 0;
$iff= 0;
staticr= 0 x 97;

sub ran1{
        my $idum= $_[0];

        my($m1,$m2,$m3,$ia1,$ia2,$ia3,$ic1,$ic2,$ic3);
      ($m1=259200,$ia1=7141,$ic1=54773);
      ($m2=134456,$ia2=8121,$ic2=28411);
      ($m3=243000,$ia3=4561,$ic3=51349);

         my $rm1 = 1.0/$m1;
          my $rm2 = 1.0/$m2;

      if (($idum<0) or ($iff==0)){
          $iff = 1;
          $ix1 = ($ic1-$idum)%$m1;
          $ix1 = ($ia1*$ix1+$ic1)%$m1;
          $ix2 = ($ix1)%$m2;
          $ix1 = ($ia1*$ix1+$ic1)%$m1;
          $ix3 = ($ix1)%$m3;
          for ($j=0; $j<97; ++$j){
              $ix1 = ($ia1*$ix1+$ic1)%$m1;
              $ix2 = ($ia2*$ix2+$ic2)%$m2;
              $staticr[$j] = ($ix1 + $ix2*$rm2)*$rm1;
          }
          $idum = 1;
      }

      $ix1 = ($ia1*$ix1+$ic1)%$m1;
      $ix2 = ($ia2*$ix2+$ic2)%$m2;
      $ix3 = ($ia3*$ix3+$ic3)%$m3;
      $j = (97*$ix3)/$m3;
      my $rand1 = $staticr[$j];
      $staticr[$j] = ($ix1 + $ix2*$rm2)*$rm1;
      return $rand1;
}

1;
```

REFERENCES

1. Raeburn, P., Uncovering the Secrets of the Red Planet: Mars, National Geographic Society, Washington, D.C., 1998.

2. Bulfinch, T., Bulfinch’s Mythology, Crown Publishers, Inc., New York, NY, 1979.

3. Columbia Encyclopedia, "Mars, in Roman Religion and Mythology," http://www.bartleby.com/65/ma/Mars-god.html, 2001.

4. Sheehan, W., The Planet Mars: a history of Observation and Discovery, University of Arizona Press, Tucson, AZ, 1996.

5. NASA, "Exploring Mars: Educational Brief," EB-1999-02-128-HQ, http://www.lpi.usra.edu/expmars/edbrief/edbrief.html, 1999.

6. Carreau, M., "'NASA Pins Hopes on Mars Probe," *Houston Chronicle*, April 3, 2001.

7. Walter, M., The Search for Life on Mars, Perseus, Cambridge, MA, 1999.

8. Beatty, K., Peterson, C., and Chaikin, A., The New Solar System, 4th Ed., Sky Pub., Cambridge, NY, 1999.

9. NASA, "Mars Exploration," http://mars.jpl.nasa.gov/, 2001.

10. Siceloff, S., "NASA Set to Fly Mars Odyssey This Week," *Florida Today*, April 4, 2001.

11. NASA, "2001 Mars Odyssey - Overview," http://mars.jpl.nasa.gov/odyssey/overview/index.html, 2001.

12. Munk, M., and Powell, R., "Aeroassist Technology Planning for Exploration," AAS-00-169, 10th AAS/AIAA Space Flight Mechanics Meeting, Clearwater, Florida, January 24-26, 2000.

13. Spencer, D. and Braun, R., "Mars Pathfinder Atmospheric Entry Trajectory Design," AAS-95-379, Proceedings of the AAS/AIAA Astrodynamics Conference, Halifax, Canada, Feb. 14-17, 1995.

14. Spencer, D., Thurman, S., Peng, C., Kallemeyn, P., Blanchard, R., and Braun, R., "Mars Pathfinder Atmospheric Entry Reconstruction," AAS-98-146, Proceedings of the AAS/AIAA Space Flight Mechanics Meeting, Monterey, CA, Feb. 9-11, 1998.

15. Desai, P., Mitcheltree, R., and Cheatwood, F., "Sample Returns Missions in the Coming Decade," IAF-00-Q.2.04, International Astronautical Congress, Rio de Janeiro, Brazil, 2000.

16. Desai, P. and Cheatwood, F., "Entry Dispersion Analysis for the Genesis Sample Return Capsule," AAS-99-469, AAS/AIAA Astrodynamics Specialist Conference, Girdwood, AK, 1999.

17. Desai, P., Braun, R., Powell, R., Engelund, W., and Tartabini, P., "Six Degree-of-Freedom Entry Dispersion Analysis for the METEOR Recovery Module," AIAA-96-0903, 34th AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV, January 15-18, 1996.

18. Braun, R., Spencer, D., Kallemeyn, P., and Vaughan, R., "Mars Pathfinder Atmospheric Entry Navigation Operations," AIAA Atmospheric Flight Mechanics Conference, New Orleans, LA, Aug. 11-13, 1997.

19. Striepe, S., Queen, E., Powell, R., Braun, R., Cheatwood, F., Aguirre, J., Sachi, L, and Lyons, D., "An Atmospheric Guidance Algorithm Testbed for the Mars Surveyor Program 2001 Orbiter and Lander," AIAA 98-4569, AIAA Atmospheric Flight Mechanics Conference, Boston, MA, Aug.1998.

20. Powell, R., "Numerical Roll Reversal Predictor-Corrector Aerocapture and Precision Landing Guidance Algorithms for the Mars Surveyor Program 2001 Missions," AIAA 98-4574, AIAA Atmospheric Flight Mechanics Conference, Boston, MA, Aug. 1998.

21. NASA, "Discovery Program Description," http://discovery.nasa.gov/overview.html, 1999.

22. Justus, C., Jeffries, W., Yung, S., and Johnson, D., "The NASA/MSFC Global Reference Atmospheric Model-1995 Version (GRAM-95)," NASA TM-4715, Systems Integration and Analysis Laboratory, Science and Engineering Directorate. NASA Marshall Space Flight Center, AL 35812 and Computer Sciences Corporation, Huntsville, AL., August 1995.

23. McCusker, T. and Hill, S., "Landing Dispersions for the Commercial Experiment Transporter Recovery System," AIAA Paper 93-3695, AIAA Atmospheric Flight Mechanics Conference, Monterey, CA, Aug. 9-11, 1993.

24. Summerset, T. et al., "Comet Deorbit and Reentry Error Analysis," The Aerospace Corporation, Contract No. F04701-88-C-0089, Aug., 1992.

25. Desai, P., Mitcheltree, R., and Cheatwood, F., "Entry Dispersion Analysis for the Stardust Comet Sample Return Capsule," AIAA-97-3812, GNC, AFM, and MST Conference and Exhibit, New Orleans, LA, 1997.

26. “Atmospheric Entry Program,” Jet Propulsion Laboratory, 1993.

27. Peng, C., Tsan, S., Smith, K., Sabahi, D., Short, K., and Mauritz, A., "Model Correlation for Mars Pathfinder Entry, Descent, and Landing Simulation," Proceedings of the 1997 IEEE Aerospace Conference, Feb., 1997.

28. Brauer, G., Cornick, D., and Stevenson, R, "Capabilities and Applications of the Program to Optimize Simulated Trajectories (POST)," NASA CR-2770, Feb., 1977.

29. Congdon, W., "Ablation Model Validation and Analytical Sensitivity Study for the Mars Pathfinder Heat Shield," AIAA 95-2129, 30th AIAA Thermoplastics Conference, San Diego, CA, June 19-22, 1995.

30. Muirhead, B., "Mars Pathfinder Flight System Integration and Test," Proceedings of the IEEE Aerospace Conference, Feb., 1997.

31. NASA, "Stardust Mission," http://stardust.jpl.nasa.gov/mission/, 2001.

32. Pastor, P., Bishop, R., and Striepe, S., "Evaluation of Mars Entry Reconstructed Trajectories Based on Hypothetical 'Quick-Look' Entry Navigation Data," 10th AAS/AIAA Space Flight Mechanics Meeting, Clearwater, FL, Jan. 24-26, 2000.

33. Carman, G., Ives, D., and Geller, D, "Apollo-Derived Mars Precision Lander Guidance," AIAA 98-4570, AIAA Atmospheric Flight Mechanics Conference, Boston, MA, Aug. 1998.

34. Ro, T. and Queen, E, "Mars Aerocapture Terminal Point Guidance and Control," AIAA-98-4571, AIAA Atmospheric Flight Mechanics Conference, Boston, MA, Aug. 1998.

35. Bryant, L, Tigges, M., and Iacomini, C., "Analytic Drag Control for Precision Landing and Aerocapture," AIAA-98-4573, AIAA Atmospheric Flight Mechanics Conference, Boston, MA, Aug. 1998.

36. Tu, K., Munir, M., Mease, K., and Bayard, D., "Drag-Based Predictive Tracking Guidance for Mars Precision Landing," AIAA-98-4573, AIAA Atmospheric Flight Mechanics Conference, Boston, MA, Aug. 1998.

37. Powell, R., Willcockson, W., and Johnson, M., "Numerical Predictor Corrector Aerocapture and Precision Landing Guidance Algorithms for the Mars 2001 Missions," AIAA-98-4573, AIAA Atmospheric Flight Mechanics Conference, Boston, MA, Aug. 1998.

38. Walberg, G., "How Shall We Go To Mars? A Review of Mission Scenarios," *Journal of Spacecraft and Rockets*, Vol. 30, No. 2, pp. 129-139, 1993.

39. Braun, R., Powell, R., Cheatwood, F., Spencer, D., and Mase, R., "The Mars Surveyor 2001 Lander: A First Step Towards Precision Landing," IAF-98-Q.3.03, 1998.

40. California Space Institute (Calspace), "Mars Exploration: Summary of Mars Missions: Mars Surveyor 2001 Lander," http://calspace.ucsd.edu/marsnow/library/mars_exploration/robotic_missions/orbiters/older_missions30.html, 2000.

41. Walberg, G. and Birge, B., "Terminal Guidance Techniques for a Mars Precision Lander," AIAA 2000-5342, Space 2000 Conference & Exposition, Long Beach, CA, Sep. 2000.

42. The Institute of Space and Astronautical Science, “MUSES-C Homepage,” http://www.muses-c.isas.ac.jp/English/, 2000.

43. NASA, “MUSES-C,” http://neo.jpl.nasa.gov/missions/musesc.html, 2000.

44. Desai, P., Braun, R., Engelund, W., and Cheatwood, F., "Mars Ascent Vehicle Flight Analysis," AIAA-98-2850, AIAA/ASME Joint Thermophysics and Heat Transfer Conference, Albuquerque, NM, 1998.

45. Vanderplaats, G., Numerical Optimization Techniques for Engineering Design with Applications, McGraw-Hill, Inc., New York, NY, 1984.

46. Bandte, O., Mavris, D., and DeLaurentis, D., "Viable Designs Through a Joint Probabilistic Estimation Technique," 1999-01-5623, AIAA and SAE, 1999 World Aviation Conference, San Francisco, CA, Oct. 19-21, 1999.

47. Mistree, F., Hughes, O., and Bras, B., "The Compromise Decision Support Problem and the Adaptive Linear Programming Algorithm," Structural Optimization: Status and Promise, (Kamat, M. ed.), Washington, DC: AIAA, pp. 247-286, 1993.

48. Hwang, C., and Masud, A., Lecture Notes in Economics and Mathematical Systems: Multiple Objective Decision-Making, Springer Verlang, Berlin, Heidelberg, New York, 1979.

49. Hwang, C., and Yoon, K., Lecture Notes in Economics and Mathematical Systems: Multiple Attribute Decision-Making, Springer Verlang, Berlin, Heidelberg, New York, 1981.

50. Zeleny, M., Multiple Criteria Decision Making, McGraw-Hill Book Company, New York, 1982.

51. Osycka, A., Multicriterion Optimization in Engineering with FORTRAN Programs, Ellis Horwood Limited, Chichester, 1984.

52. Steuer, R., Multiple Criteria Optimization: Theory, Computation, and Application, John Wiley & Sons, New York, 1986.

53. Stadler, W. and Dauer, J., "Multicriteria Optimization in Engineering: A Tutorial and Survey," Structural Optimization: Status and Promise, (Kamat, M. ed.), *Progress in Astronautics and Aeronautics*, Vol. 150, AIAA, Washington, DC, pp. 209-244, 1992.

54. Stadler, W., "Caveats and Boons of Multicriteria Optimization," *Microcomputers in Civil Engineering*, Vol. 10, Blackwell, Cambridge, MA, pp. 291-299, 1995.

55. Mavris, D. and Bandte, O., "A Probabilistic Approach to Multivariate Constrained Robust Design Simulation," AIAA-97-5508, World Aviation Congress and Exposition, Anaheim, CA, Oct. 1997.

56. Ignizio, J., Linear Programming in Single and Multi-Objective Systems, Prentice-Hall, Englewood Cliffs, NJ, 1982.

57. Ignizio, J., "Introduction to Linear Goal Programming," Quantitative Applications in the Social Sciences, (Sullivan, J. and Niemi ed.), Sage University Papers, Beverly Hills, CA, 1985.

58. Kleijnen, J, Statistical Tools for Simulation Practitioners, Marcel Dekker, New York, 1987.

59. Simpson, T., Peplinski, J., Koch, P., and Allen, J., "On the Use of Statistics in Design and the Implications for Deterministic Computer Experiments," DETC97DTM3881, ASME Design Engineering Technical Conferences, Sacramento, CA, 1997.

60. DeLaurentis, D. and Mavris, D., "Uncertainty Modeling and Management in Multidisciplinary Analysis and Synthesis," AIAA 2000-0422 38th AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV, Jan. 2000.

61. Su, J. and Renaud, J., "Automatic Differentiation in Robust Optimization," *AIAA Journal*, Vol. 35, No. 6, pp. 1072-1079, Jun. 1998.

62. Mavris, D., Baker, A., and Schrage, D., "Simultaneous Assessment of Requirements and Technologies in Rotorcraft Design," Proceedings of the 56th Annual Forum of the American Helicopter Society, Virginia Beach, VA, May 2000.

63. Giunta, A. and Watson, L., "A Comparison of Approximation Modeling Techniques: Polynomial Versus Interpolating Models," AIAA-98-4758, AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, 7th, St. Louis, MO, Sept. 1998.

64. Unal, R., Lepsch, R., and McMillin, M., "Response Surface Model Building and Multidisciplinary Optimization Using Overdetermined D-Optimal Designs," AIAA-98-4759, 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, St. Louis, MO, Sept.1998.

65. Unal, R., Braun, R., Moore, A., and Lepsch, R., "Response Surface Model Building Using Orthogonal Arrays for Computer Experiments," 19th Annual International Conference of the International Society of Parametric Analysis, New Orleans, LA, May 1997.

66. Box, G. and Drapper, N., Empirical Model-Building and Response Surfaces, John Wiley & Sons, Inc., New York, NY, 1987.

67. Box, G., Hunter, W., and Hunter, J., Statistics for Experiments, John Wiley & Sons, Inc., New York, NY, 1978.

68. Myers, R. and Montgomery, D., Response Surface Methodology: Process and Product Optimization Using Designed Experiments, pp. 1-141, 279-401, 462-480, John Wiley & Sons, Inc., New York, NY, 1995.

69. Sacks, J., Welch, W., Mitchell, T. and Wynn, H., "Design and Analysis of Computer Experiments," *Statistical Science*, Vol. 4, No. 4, pp.409-435, Nov. 1989.

70. Welch, W., Yu, W., Kang, S., and Sacks, J., "Computer Experiments for Quality Control by Parameter Design," *Journal of Quality Technology*, Vol. 22, No. 1, pp.15-22, 1990.

71. Owen, A., "Orthogonal Array Designs for Computer Experiments, Department of Statistics, Stanford University," http://lib.stat.cmu.edu/designs/owen.small, 1994.

72. Montgomery, D., Design and Analysis of Experiments, 4th Ed., John Wiley & Sons, New York, 1997.

73. SAS Institute Inc., "JMP Statistics and Graphics Guide: Version 3.1 of JMP," Cary, NC, 1995.

74. Ignizio, J., "Generalized Goal Programming: An Overview," *Computers and Operations Research*, Vol. 5, No. 3, pp.179-197, 1983.

75. Deb, K. and Agrawal, S., "A Niched-Penalty Approach for Constraint Handling in Genetic Algorithms," Artificial Neural Nets and Genetic Algorithms: Proceedings of the International Conference in Portoroz, Slovenia, (Dobnikar et al. ed.), Springer Wien, New York, 1999.

76. INT Media Group Inc, “ISP Glossary: wrapper,” http://isp.webopedia.com/TERM/w/wrapper.html, 2001.

77. Southwest Research Institute, FPI User's and Theoretical Manual, San Antonio, TX, 1995.

78. Knuth, D., "Seminumerical Algorithms (Vol. 2)", The Art of Computer Programming, Addison-Wesley, Reading, MA, 1969.

79. Anderson, S., "Random Number Generators on Vector Sumpercomputers and Other Advanced Architectures," *SIAM Review*, Vol. 32, No. 2, pp. 221-251, 1990.

80. Lehmer, D., "Mathematical Methods in Large-Scale Computing Units," Proc. of 2nd Symp. on Large-Scale Digital Calculating Machinery, pp. 141-146, 1951.

81. Press, W., Flannery, B., Teukolsky, S., and Vetterling, W., Numerical Recipes: The Art of Scientific Computing (FORTRAN Version), Cambridge University Press, New York, NY, 1989.

82. Von Neumann, J., "Various Techniques Used in Connection with Random Digits," The Monte Carlo Method, National Bureau of Standards Applied Mathematics Series 12, p. 36, 1951.

83. Georgia Lottery Corporation, "The Big Game," http://www.georgialottery.com/lottery/biggame.html, 2001.

84. Press, W., Flannery, B., Teukolsky, S., and Vetterling, W., Numerical Recipes in Fortran 77: The Art of Scientific Computing, 2nd Ed., Cambridge University Press, New York, NY, 1992.

85. Fisher, R. and Tippett, L, "Limiting Forms of the Frequency Distribution of the Largest or Smallest Member of a Sample," *Proceedings of the Cambridge Philosophical Society*, Vol. 24, pp. 180-190,1928.

86. Weibull, W., "A Statistical Theory of the Strength of Materials," *Ingeniors Vetenskaps Akademien Hadlingar*, No. 153, 1939.

87. Weibull, W., "A Statistical Distribution Function of Wide Applicability," *Journal of Applied Mechanics*, Vol. 18, pp. 293-297, 1951.

88. Leemis, L., Reliability: Probabilistic Models and Statistical Methods, Prentice-Hall Inc., Upper Saddle River, NJ, 1985.

89. Law, A. and Kelton, W., Simulation Modeling and Analysis, 2nd ed., McGraw-Hill, Inc., New York, NY, 1991.

90. Menon, M., "Estimation of the Shape and Scale Parameters of the Weibull Distribution," *Technometrics*, Vol. 5, No. 2, pp. 175-182, 1963.

91. Hogg, R. and Tanis, E., Probability and Statistical Inference, Prentice Hall, Inc., Upper Saddle River, NJ, 1997.

92. Wall, L., Christiansen, T., and Schwartz, R., Programming Perl, 2nd ed., O'Reilly & Associates, Inc., 1996

93. Kinderman, J. and Ramage, J., "Computer Generation of Normal Random Variables," *Journal of the American Statistical Association*, Vol. 71, No. 356, pp. 893-896, 1976.

94. Ahrens, J. and Dieter, U., "Computer Methods for Sampling from the Exponential and Normal Distributions," *Communications of the ACM*, Vol. 15, No. 10, pp. 873-882, 1972.

95. Marsaglia, G. and Bray, T., "A Convenient Method for Generating Normal Variables," *SIAM Review*, Vol. 6, No. 3, pp. 260-264, 1964.

96. Newman, T. and Odell, P., The Generation of Random Variates, New York, Hafner Press, 1971.

97. Marsaglia, G., "Random Variables and Computers," in Transactions of the Third Prague Conference on Information Theory, Statistical Decision Functions, Random Processes, June 1962, Prague: Publishing House of the Czechoslovak Academy of Sciences, pp. 499-512, 1964.

98. Marsaglia, G., MacLaren, M., and Bray, T., "A Fast Procedure for Generating Normal Random Variables," *Communications of the ACM*, Vol. 7, No. 1, pp. 4-10, 1964.

99. Marsaglia, G., "Generating a Variable from the Tail of the Normal Distribution," *Technometrics*, Vol. 6, No. 1, pp.101-102, 1964.

100. Bate, R., Mueller, D., and White, J., Fundamentals of Astrodynamics, Dover Publications, Inc., New York, NY, 1971.

101. Myers, R., Response Surface Methodology, Virginia Commonwealth University, Allyn and Bacon Inc., Boston, Mass., 1971.

102. Craig, J., "D-Optimal Design Method: Final Report and User's Manual," USAF Contract F33615-78-C-3011, FZM-6777, General Dynamics, Fort Worth Div., 1978.

103. Lucas, J., "Optimum Composite Designs," *Technometrics*, Vol. 16, No. 4, Nov. 1974.

104. Mitchell, T., "An Algorithm for the Construction of D-Optimal Experimental Designs," *Technometrics*, Vol. 16, No. 2, May 1974.

105. Dong, Z., "Design for Automated Manufacturing," Concurrent Engineering: Automation, Tools, and Techniques, (Kusiak, A., Ed.), John Wiley & Sons, Inc, New York, 1993, pp. 207-233

106. Speckhart, F., "Calculation of Tolerance Based on a Minimum Cost Approach," *ASME Journal of Engineering for Industry*, Vol. 94, No. 2, pp.447-453, 1972.

107. Spotts, M., "Allocation of Tolerances to Minimize Cost of Assembly," *ASME Journal of Engineering for Industry*, pp. 762-764, Aug.1973.

108. Sutherland, G. and Roth, B., "Mechanism Design: Accounting for Manufacturing Tolerances and Costs in Function Generating Problems," *Journal of Engineering for Industry, Transactions of ASME*, Vol. 98, pp. 283-286, 1975.

109. Michael, W. and Siddall, J., "Optimization Problem with Tolerance Assignment and Full Acceptance," *Journal of Mechanical Design, Transactions of the ASME*, Vol. 103, pp. 842-848, 1981.

110. Chase, K. and Greenwood, W., "Design Issues in Mechanical Tolerance Analysis," *Manufacturing Review*, Vol. 1, No. 1, March, pp. 50-59, 1988.

111. Dong, Z. and Soom, A., "Automatic Optimal Tolerance Design for Related Dimension Chains," *Manufacturing Review*, Vol. 3, No. 4, pp.262-271, 1990.

112. Lee, Y. and Woo, T., "Optimum Selection of Discrete Tolerances," *Journal of Mechanisms, Transmissions, and Automation in Design*, Vol. 111, June, pp. 243-251, 1987.

113. Dong, Z. and Hu, W., "Automated Process Sequence Identification and Optimal Process Tolerance Assignment in Computer-Aided Process Planning," *Computers in Industry*, Vol. 17, pp. 19-32, 1991.

114. Kwok, J., "The Artificial Satellite Analysis Program (ASAP) Version 2.0," JPL EM 312/87-153, 1987.

115. Justus, C., James, B., and Johnson, D., "Recent and Planned Improvements in the Mars Global Reference Atmospheric Model (Mars-GRAM)," Advances in Space Research, Vol. 19, No. 8, pp. 1223-1231, 1997.

116. Mitcheltree, R., Wilmoth R., Cheatwood, F., Brauckmann, G., Greene, F., "Aerodynamics of Stardust Sample Return Capsule," AIAA-97-2304, Applied Aerodynamics Conference, Atlanta, GA, 1997.

117. Gnoffo, P, Gupta, R. and Shinn, J., "Conservation Equations and Physical Models for Hypersonic Air Flows in Thermal and Chemical Nonequilibrium," NASA TP-2867, 1989.

118. Cheatwood, F. and Gnoffo, P., "User's Manual for the Langley Aerothermodynamic Upwind Relaxation Algorithm (LAURA)," NASA TM-4674, 1996.

119. Riley, C. and Cheatwood, F., "Distributed-Memory Computing with the Langley Aerothermodynmic Upwind Relaxation Algorithm (LAURA)," NASA National Symposium on Large-Scale Analysis and Design on High-Performance Computers and Workstations, Williamsburg, VA, 1997.

120. Gnoffo, P., Braun, R., Weilmuenster, K., Powell, R., Mitcheltree, R., and Engelund, W., "Prediction and Validation of the Mars Pathfinder Aerodynamic Database," AIAA-98-2445, June 1998.

121. Gnoffo, P., Weilmuenster, K., Braun, R., and Cruz, C., "Influence of Sonic-Line Location on Mars Pathfinder Probe Aerothermodynamics," *Journal of Spacecraft and Rockets*, Vol. 33, No. 2, Mar.-Apr. 1996.

122. Mitcheltree, R., Moss, J., Cheatwood, F., Greene, F., and Braun, R., "Aerodynamics of the Mars Microprobe Entry Vehicles," AIAA-97-3658, Atmospheric Flight Mechanics Conference, New Orleans, LA, 1997.

123. Taylor, A., Advanced Calculus, Chapter 6, Ginn and Co., Lexington, MA, 1955.

124. Rao, S., Optimization: Theory and Applications, 2nd Ed., Wiley Eastern Limited, Daryaganj, New Delhi, 1984.

125. Ralston, A. and Rabinowitz, P., A First Course in Numerical Analysis, McGraw-Hill, New York, NY, 1978.

VITA

David Wesley Way was born September 11, 1969, in Winston-Salem, North Carolina, but considers Beaufort, North Carolina his home. In 1987, he graduated valedictorian from East Carteret High School and accepted an appointment to the United States Naval Academy in Annapolis, Maryland. David graduated from the Naval Academy, received a Bachelor of Science degree in Aerospace Engineering, and was commissioned an Ensign in the United States Navy in 1991.

Following graduation, he entered the nuclear power training program. Over the next two years, David trained for submarine duty at the Nuclear Power School in Orlando, Florida; the Nuclear Power Training Unit in Idaho Falls, Idaho; and the Submarine Officer Basic Course in Groton, Connecticut. In 1993, he received orders to the unique fast-attack submarine the U.S.S. NARWHAL (SSN-671) in Charleston, South Carolina. In 1994, Narwhal changed home port to Norfolk, Virginia. While aboard Narwhal, David completed qualifications for Engineer Officer. He was awarded the Navy/Marine Corps Achievement Medal, the Navy Excellence Ribbon, the National Defense Medal, the Armed Forces Service Medal, the Sea Service Deployment Ribbon, and the N.A.T.O. Service Medal. In 1996, he resigned his commission to pursue graduate studies at the Georgia Institute of Technology in Atlanta, Georgia.

In October of 1996, David began his graduate studies in Aerospace Engineering under the advisement of Dr. John R. Olds. In the summer of 1997, he participated in the NASA Langley Aerospace Research Summer Scholarship program at the Langley Research Center in Hampton, Virginia, and was subsequently the recipient of a NASA Graduate Student Researcher’s Program fellowship with the Vehicle Analysis Branch of the Space Systems and Concepts Division. In December 1997, David received his Master of Science degree in Aerospace Engineering. His individual research for that degree was in the development of a web-based, liquid rocket engine analysis tool, SCORES (Space Craft Object-oriented Rocket Engine Simulation).

In addition to his individual work, David has contributed as either the team lead or propulsion analyst to many design projects with the Space Systems Design Lab. In accordance with his fellowship, he conducted research during the summers at NASA Langley. There he has performed Monte Carlo uncertainty analysis of Mars Ascent Vehicle trajectories and developed autopilot simulations for a Mars precision lander. The culmination of his studies is this doctoral dissertation entitled, “Uncertainty Optimization Applied to the Monte Carlo Analysis of Planetary Entry Trajectories.”