Adaptive Sampling Algorithms for Minimizing the Variance of Parameter Estimates in the Bidirectional Surface Reflectance Model

Or

Intelligent Instruments on Robotic Helicopters

A Technical Report

Presented to Dr. Kevin Wheeler, NASA/Ames Research Center

Team Members:

Gabriel Cadden
Marian Hofer
Shikha Naik
Kim Ninh
Pinky Thakkar

Faculty Supervisor: Dr. Steven M. Crunk

Center for Applied Mathematics and Computer Science (CAMCOS)

Department of Mathematics
San Jose State University

December 9, 2005
 Adaptive Sampling Algorithms for Minimizing the Variance of Parameter Estimates in the Bidirectional Surface Reflectance Model

Intelligent Instruments on Robotic Helicopters

ABSTRACT

Traditional instruments collect data, which are stored and later analyzed. Often such collected data are not useful, or even worse, measurements that would have been useful were not collected. By the time the data are analyzed, it is too late for a researcher to obtain those useful (and sometimes crucial) measurements. We introduce a paradigm shift where the instrument and analysis systems work in concert. By developing software engines that analyze the data as the measurements are made, we can identify measurements that would improve the analysis results and request them from the instrument. This requires that three systems work symbiotically: the instrument, the inference engine, and the inquiry engine. We will apply this methodology to the problem of mapping the surface of the ground with a spectrometer mounted on an autonomous rotorcraft (robotic helicopter). The inference engine will analyze the data from the spectrometer, and the inquiry engine will use the results of the analysis to direct the helicopter.
Table of Contents

ABSTRACT .......................................................................................................................... 2
I. INTRODUCTION ........................................................................................................... 4
II. THEORY ............................................................................................................................ 6
   A. Linear Regression (i.e. Linear Least-Squares Fit) ...................................................... 6
   B. Nonlinear Regression (i.e. Nonlinear Least-Squares Fit) ......................................... 7
   C. Maximum Likelihood Estimators ........................................................................... 8
   D. Fisher Information Matrix...................................................................................... 11
   E. Bayesian Statistics: ............................................................................................... 12
   F. Information Theory ............................................................................................... 15
   G. Entropy ................................................................................................................. 16
   H. Maximum Entropy Sampling .................................................................................. 16
III. IMPLEMENTATION .................................................................................................. 21
   A. Our Model ............................................................................................................... 21
   B. Linearizing the Model ............................................................................................ 23
   C. Experiment Setup ................................................................................................... 24
IV. RESULTS .................................................................................................................... 25
   A. Estimation of the Parameters ................................................................................ 25
   B. Maximization of Entropy ....................................................................................... 26
V. CONCLUSIONS .......................................................................................................... 31
VI. RECOMMENDATIONS AND FUTURE RESEARCH ................................................. 31
   A. Non-Normal Priors .............................................................................................. 31
   B. Obtaining Posterior Distributions from “Messy” Systems ..................................... 32
   C. Future Research .................................................................................................... 35
APPENDIX A: References ............................................................................................... 35
APPENDIX B: Matlab functions ..................................................................................... 37
I. INTRODUCTION

In order to estimate the amount of carbon sequestered by trees in the United States, the amount of standing woods must be estimated with quantifiable uncertainty (Wheeler, 2005). These estimates come from either satellite images or near ground measurements. The amounts of error in the estimates from these two approaches are currently unknown. To this end, an autonomous helicopter with differential GPS (Global Positioning System), LIDAR (Light Detection and Ranging), stereo imagers, and spectrometers has been developed as a testing platform for conducting further studies (Wheeler, 2005). The controlling software system for this robotic helicopter must be developed such that the ground sampling is optimized.

The simplest sampling method is to send the helicopter to every possible location to gather data of interest. However, this method is highly inefficient as it will require a large amount of time and money. The other possible method employs random sampling. The helicopter could take random samples until the estimates are reasonably correct. Although there is a possible chance that the helicopter will take samples from the locations that offer the most amount of information and therefore minimizing the needed sample set, there is no guarantee that this sample set will be taken every time. Therefore, the most efficient solution is to take only a few samples from “key” locations that offer the most amount of information.

In order to optimize the sampling process, the software must work in tandem with the sampling hardware to control the helicopter’s position (Wheeler, 2005). Once a sample has been taken, the data is fed into the software system, which then calculates the next best position to gather further data. Initially, the software system assumes an
empirical model for the ground being examined. With each addition of data from the instruments, the software system employs the Bayesian Probability Theory to update its model and calculate the helicopter’s next position. The system repeats this process until the uncertainties of its model are within a satisfactory range. This method allows the system to be adaptive during sampling and ensures adequate ground cover.

The model used in this project is based on the BRDF, Bidirectional Reflectance Distribution Function (Wolfgang Lucht, 2000). This function permits the calculation of the amount of reflection based on the reflectance of the object and the positions of the sun and the viewer. With this function, the system can compensate for different positions of the sun during sampling.

This project is based on two research papers, “Bayesian Adaptive Exploration”, written by Thomas J. Loredo, and “Experimental Design to Maximize Information”, written by Paola Sebastiani and Henry Wynn (2001). Both of these papers follow the Bayesian design methods. Therefore, the framework of the implementation presented in this paper is a combination of the mathematics provided by these papers. By using the Bayesian methodology, the system examines current available data with regards to previously collected data in order to predict the future data. This prediction leads to the identification of which new observations will yield the most information.

Applying these methods, this paper outlines the theory and implementation of the sampling software. First, we describe the theories and establish the mathematical background, which include regression, maximum likelihood, Fisher information, Bayesian statistics, information theory, entropy, and maximum entropy sampling. Second, we provide a thorough explanation of the implementation process through the
presentation of various formulae, application of the methods described by Sebastiani and Wynn, prior and posterior methods, regression, likelihood, and solutions. The concluding discussions include comparisons of solutions from different methods, advantages and disadvantages of this particular implementation, avenues for further research, and a discussion of non-normal parameters and their implications.

II. THEORY

In this section of the paper we discuss the mathematical background of our methods from estimating model parameters to establishing an algorithm for maximum entropy sampling.

A. Linear Regression (i.e. Linear Least-Squares Fit)

Linear models represent the relationship between a continuous dependent variable and one or more independent variables (either continuous or categorical) in the form:

\[ y = X\beta + \varepsilon \]  

(Montgomery and Peck, 1992).

where:

- \( y \) is an n-by-1 vector of observations of the dependent variable.
- \( X \) is the n-by-p “design” matrix determined by the independent variables.
- \( \beta \) is a p-by-1 vector of unknown parameters to be estimated.
- \( \varepsilon \) is an n-by-1 vector of random disturbances, independent of each other and usually having a normal distribution.

Examples:

- \( y = \beta_0 + \beta_1x_1 + \beta_2x_2 + \beta_3x_3 \)
• \( y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 \cos(x) \)

→ “linearized” model \( \Rightarrow y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 \), where \( x_2 = x^2 \) and \( x_3 = \cos(x) \)

• \( y = \beta_0 \times \exp(\beta_1 x) \Rightarrow \log(y) = \log(\beta_0) + \beta_1 x \)

→ “linearized” model \( \Rightarrow z = B_0 + \beta_1 x \), where \( B_0 = \log(\beta_0) \)

Solving for the unknown parameters is achieved by solving an over-determined linear system of equations using least-squares technique. Least-squares minimizes
\[ ||r||^2 = ||Ax - y||^2, \]
which is the squared Euclidean norm of the residuals. Matlab uses a numerically stable algorithm that utilizes the QR factorization of A, where Q is orthogonal, and R is upper triangular, to solve this problem.

B. Nonlinear Regression (i.e. Nonlinear Least-Squares Fit)

Like linear models, nonlinear models represent the relationship between a continuous dependent variable and one or more independent variables. However, in nonlinear models, the unknown parameters do NOT enter the model linearly. This model is of the form: \( y = f(X, \beta) + \epsilon \) (Matlab 7.0.1, 2004).

where:

• \( y \) is an n-by-1 vector of observations of the dependent variable.

• \( f \) is any function of \( X \) and \( \beta \).

• \( X \) is the n-by-p “design” matrix determined by the independent variables.

• \( \beta \) is a p-by-1 vector of the unknown parameters.

• \( \epsilon \) is an n-by-1 vector of random disturbances.
Example:

\[ y = \frac{\beta_1 x_2 - \frac{x_3}{\beta_5}}{1 + \beta_2 x_1 + \beta_3 x_2 + \beta_4 x_3} \]  
(Hougen-Watson model for reaction kinetics)

Nonlinear models are more difficult to fit, requiring iterative methods that start with an initial guess of the unknown parameter vector. Each iteration alters the current guess until the algorithm converges. In Matlab, nonlinear least-squares fitting is performed using the Gauss-Newton algorithm with Levenberg-Marquardt modifications for global convergence.

C. Maximum Likelihood Estimators

Maximum likelihood estimation (MLE) is a statistical technique that helps us make inferences about the unknown parameters of a distribution, given a particular observed set of sample data.

Suppose that we have collected a sample data set \( x_1, x_2, x_3, \ldots, x_n \) from a population with a known probability distribution function \( f \) and an unknown distributional parameter \( \beta \). We define the likelihood function to be:

\[
\text{Likelihood} = \text{Probability (that we sample values } x_1, x_2, \ldots, x_n) = f(x_1, x_2, \ldots, x_n| \beta)
\]

The likelihood function tells us how likely the observed sample is as a function of the parameter \( \beta \) (Wackerly et al., 2001). Maximizing the likelihood with respect to \( \beta \) gives the parameter value for which the observed sample is most likely to have been generated—that is, the parameter values that “agrees most closely” with the observed data. The value \( \hat{\beta}_{MLE} \) that maximizes the likelihood function is known as the maximum
likelihood estimator (MLE) of the true parameter $\beta$. This concept can be generalized for distributions that have multiple parameters.

Example (using a discrete distribution)

Consider tossing a coin 50 times and recording each time the value of the outcome; H for heads and T for tail. Suppose the final outcome was 32 heads and 18 tails. Let $P(H) = p \Rightarrow P(T) = 1 - p$. Suppose that this coin came from a box that contained 3 coins: one has $p(H) = 0.35$, the second has $p(H) = 0.50$, and the third has $p(H) = 0.65$. Which coin did we use in this experiment? Using maximum likelihood estimation, we can calculate which coin it most likely was, given the outcome that we observed.

Using our three possible values of parameter $p$, the likelihood function, based on our definition earlier, takes these three values:

$L(32\ \text{heads}|\ p = 0.35) = \binom{50}{32}(0.35)^{32}(1-0.35)^{18} \approx 1.991 \times 10^{-5}$

$L(32\ \text{heads}|\ p = 0.50) = \binom{50}{32}(0.50)^{32}(1-0.50)^{18} \approx 0.01603$

$L(32\ \text{heads}|\ p = 0.65) = \binom{50}{32}(0.65)^{32}(1-0.65)^{18} \approx 0.1156$

Note that the likelihood function is maximized when $p = 0.65$. Thus our coin is most likely the one with $p(H) = 0.65$, which is the maximum likelihood estimator $\hat{p}_{\text{MLE}}$ of the true parameter $p$.

Example (using a continuous distribution)

In the continuous distribution case, maximizing the likelihood is equivalent to maximizing the logarithm of the likelihood, since log is a continuous strictly-increasing
function over the range of the likelihood. Algebraically, this boils down to taking the
first derivative of $\log(\text{Likelihood})$ with respect to the parameter $\beta$, setting this derivative
equal to zero, and solving for $\hat{\beta}_{\text{MLE}}$. If the distribution has multiple parameters, we must
take the partial derivatives of $\log(\text{Likelihood})$ with respect to each parameter, equate them
to zero, and solve the resulting system of equations.

Consider the normal distribution function:

$$f(x \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x - \mu)^2}{2\sigma^2}\right), \text{ where } \mu \text{ and } \sigma^2 \text{ are the unknown mean and variance (respectively) of this distribution.}$$

The corresponding likelihood function is:

$$\text{Likelihood} = f(x_1, x_2, \ldots, x_n \mid \mu, \sigma^2)$$

$$= \prod_{i=1}^{n} \left\{ \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x_i - \mu)^2}{2\sigma^2}\right) \right\} = \left(\frac{1}{2\pi\sigma^2}\right)^\frac{n}{2} \exp\left(\sum_{i=1}^{n} -\frac{(x_i - \mu)^2}{2\sigma^2}\right)$$

Since maximizing the likelihood is equivalent to maximizing its log $\Rightarrow$

Maximize the natural log of the equation above.

$$\ell = \log(\text{Likelihood}) = \left(\frac{n}{2}\right) \log\left(\frac{1}{2\pi\sigma^2}\right) - \sum_{i=1}^{n} \frac{(x_i - \mu)^2}{2\sigma^2}$$

In order to maximize $\ell$, we must take its partial derivatives $\frac{\partial \ell}{\partial \mu}$ and $\frac{\partial \ell}{\partial \sigma^2}$, set

$$\frac{\partial \ell}{\partial \mu} = 0 \text{ and } \frac{\partial \ell}{\partial \sigma^2} = 0,$$

and solve this system of two equations with two unknowns for $\hat{\mu}_{\text{MLE}}$ and $\hat{\sigma}^2_{\text{MLE}}$. The followings are the results produces by this process:
\[
\hat{\mu}_{\text{MLE}} = \frac{\sum_{i=1}^{n} x_i}{n} = \bar{x} \quad \text{and} \quad \hat{\sigma}^2_{\text{MLE}} = \frac{\sum_{i=1}^{n} (x_i - \hat{\mu}_{\text{MLE}})^2}{n} = \frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{n}, \text{where} \ \bar{x} \ \text{is the sample mean.} \quad \text{Thus, the vector of estimated parameters is} \quad \hat{\beta}_{\text{MLE}} = \left[ \begin{array}{c} \hat{\mu}_{\text{MLE}} \\ \hat{\sigma}^2_{\text{MLE}} \end{array} \right] = \left[ \begin{array}{c} \bar{x} \\ \frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{n} \end{array} \right].
\]

### D. Fisher Information Matrix

\[
F = \begin{bmatrix}
\frac{\partial^2 \ell}{\partial \beta_1^2} & \frac{\partial^2 \ell}{\partial \beta_1 \partial \beta_2} & \ldots & \frac{\partial^2 \ell}{\partial \beta_1 \partial \beta_p} \\
\frac{\partial^2 \ell}{\partial \beta_2 \partial \beta_1} & \frac{\partial^2 \ell}{\partial \beta_2^2} & \ldots & \frac{\partial^2 \ell}{\partial \beta_2 \partial \beta_p} \\
\vdots & \vdots & \ddots & \vdots \\
\frac{\partial^2 \ell}{\partial \beta_p \partial \beta_1} & \frac{\partial^2 \ell}{\partial \beta_p \partial \beta_2} & \ldots & \frac{\partial^2 \ell}{\partial \beta_p^2}
\end{bmatrix}
\]

Fisher Information Matrix = \( F = -\left[ \frac{\partial^2}{\partial \beta^2} \log f(X, \vec{\beta}) \right] \), where \( \vec{\beta} \) is the vector of the unknown parameters of distribution \( f \) (Myung and Navarro, 2004). The Fisher information matrix helps measure the accuracy of the estimated parameters.

The Cramér-Rao inequality states that the reciprocal of the Fisher information is a lower bound on the variance estimator of \( \vec{\beta} \) (Spall, 2004).

\[
F^{-1} \approx \begin{bmatrix}
V(\hat{\beta}_1) & \text{Cov}(\hat{\beta}_1, \hat{\beta}_2) & \ldots & \text{Cov}(\hat{\beta}_1, \hat{\beta}_p) \\
\text{Cov}(\hat{\beta}_2, \hat{\beta}_1) & V(\hat{\beta}_2) & \ldots & \frac{\partial^2 \ell}{\partial \beta_2 \partial \beta_p} \\
\vdots & \vdots & \ddots & \vdots \\
\text{Cov}(\hat{\beta}_p, \hat{\beta}_1) & \text{Cov}(\hat{\beta}_p, \hat{\beta}_2) & \ldots & V(\hat{\beta}_p)
\end{bmatrix}
\]

The standard errors of our approximations are the square roots of the diagonal of \( F^{-1} \).
E. Bayesian Statistics:

The most common way to view statistics is in “Frequentist” terms, where probability is derived from observed (or defined) frequency distributions or proportions of populations. Thus, frequentist statisticians do not look at questions in which things might be personally believed, only at what is known. In contrast, Bayesians would use some degree of belief to weigh the known frequency, and later update their beliefs in the light of new information —such updating is known as Bayesian inference. Whereas both a Bayesian and a Frequentist would view the probability of pulling an ace out of a deck of cards as 1/13, because that is known to be true, and known not to change, only a Bayesian would assign a probability to something like a tree making a sound when nobody is around to hear it. Subjectivism is an element that belongs only in the Bayesian world. This movement of using subjective beliefs was first brought to light by Reverend Thomas Bayes. His paper “An Essay Toward Solving a Problem in the Doctrine of Chances,” published in 1764, introduced a theorem which has become the backbone behind the Bayesian world, this theorem is commonly known as Bayes’ Theorem.

In the two random events case, Bayes’ Theorem states that the probability of the random event A occurring, given that we know that a random event B has occurred, is given by the relation of the product between the probability of event A occurring on its own with the probability of event B occurring, given that we know event A has occurred. All of that divided by the probability of B occurring on its own.

\[
P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}
\]

Bayesian Statistics is essentially an extension of Bayes’ Theorem, wherein we combine known information, often times referred to as a prior distribution, with new
information, which is also known as a likelihood, to obtain new expectations, which we
call a posterior distribution. Mathematically, the posterior distribution is equal to the
product of a prior distribution and a likelihood function divided by a normalizing
constant.

In this project, Bayes’ Theorem is implemented as follows:

\[
P(\text{model} \mid \text{observed data}) = \frac{P(\text{observed data} \mid \text{model})P(\text{model})}{P(\text{observed data})}
\]

Thus, the probability of what the model parameters should be, given the observed
data, is equal to the likelihood of the observed data occurring given our model
parameters, multiplied by the probability of our model parameters being what they are,
all divided by the total probability of the data samples being observed regardless of
anything else. Bayes’ Theorem is often viewed as a way of revising beliefs in the light of
newly obtained information. The applications of Bayes’ Theorem are just about endless.

To illustrate the simple use of Bayes’ Theorem, we have created a small example.
Keep in mind, this example is not intended to solve anything that is not obvious; it is just
here to show the use of the theorem.

**Example:**

Consider a horse race featuring two horses: Stallion vs. Gumshoe. Previously, both
horses had participated in 20 races, where Stallion won 14 of them and Gumshoe won
the other 6. However, upon further examination of previous races’ track conditions, it
turns out that 5 out of the 6 winnings of Gumshoe occurred when it had rained the
night before the race, causing the track to be soft the next day. Suppose that you know
that it rained last night. What’s the chance of Gumshoe winning the race today?
We want to estimate the probability of Gumshoe winning given that it has rained the night before—this is our posterior. It is equal to the probability that it has rained the night before, given that gumshoe has won, which we said was 5 out of 6 races; this is the likelihood. We then multiply that by the probability of Gumshoe winning the race with no knowledge of anything else, which out of 20 races he has won 6 of them; this is also known as our prior. We then need to divide all that by the probability that it has rained, which we know out of the 20 races it has rained 7 times, so we evaluate this equation and find that our posterior is 71.4%. Thus, the odds are pretty decent in favor of Gumshoe winning this race; not so obvious when you just look at their overall match-ups.

\[
P(\text{Gumshoe W | Rain}) = \frac{P(\text{Rain | Gumshoe W}) \times P(\text{Gumshoe W})}{P(\text{Rain})}
\]

\[
P(\text{Gumshoe W | Rain}) = \frac{(5/6)(6/20)}{(7/20)}
\]

\[
= 71.4\%
\]

What this goes to show is that in Bayesian Statistics, the more information you know, the more accurate your posterior is going to be.

Bayesian inference, which relies heavily on conditional probabilities, is basically performed by evaluating the probability of a hypothesized model, given observed data. In doing so, one creates what could be viewed as an artificial learning system, where belief revision is used. Thus, a prior subjective probability is replaced by a posterior probability that incorporates newly acquired information. This system has two stages: first some of the probabilities are directly altered by some non-inferential process, such as personal experience or intuition. The second stage involves updating the rest of the prior
beliefs to make them more in line with the newly acquired information. This learning systems leads to later discussion which involve priors that are “messy”.

**F. Information Theory**

Claude E. Shannon is widely credited as the founder of Information Theory. He looked at the general problem in which a source of information is to be encoded, sent over a channel, and then in turn is decoded by the recipient. The information revolution is one of the most important developments in the twentieth century. An essential ingredient to this revolution has been a branch of mathematics called information theory which is concerned with encoding and transmitting data from one place to another.

In terms of what we were looking at, Shannon defined self-information of some system as the *negative logarithm* of that system, which is a measure of the freedom of choice with which a message is selected from the set of all possible messages. The value obtained is often called bits, this value is a way of measuring the capacity of a communication channel. This is based upon the *log* that is used in computation and quite frequently in programming, questions are presented in a Bernoulli fashion, thus we use log base 2, and furthermore describe information as bits. Shannon’s Self-Information, which is simply a measure of the amount of information content, is defined by the following equation:

$$\text{Info}[p(a)] = -\log[p(a)]$$

Information theory provides a great example of how different branches of mathematics have been applied to problems in the real world. The applications as mentioned before are numerous in fields such as coding theory, data compression theory, and error correction theory. As the information revolution progresses, information
theorists will continue to come up with codes necessary for accurate and rapid communication.

**G. Entropy**

Entropy is a concept not only in thermodynamics, but also in Information Theory. Entropy is the measure of randomness in an event, or the number of ways a system might be arranged and still gives the same appearance. This definition can be applied from both thermodynamics and Information Theory. According to Shannon, we can quantify entropy in a statistical manner by saying that Entropy is equal to the expected value of information. Since we know the expected value of something is the summation of that something multiplied by the function of that something, we can look at entropy in the following manner, the entropy of a system is equal the negative summation of the that system multiplied by the function of that system:

$$E[Info[p(a)]] = -\sum p(a)\log[p(a)]$$

**H. Maximum Entropy Sampling**

Information about the precision of an estimate is conveyed by the width of the confidence interval. If the confidence level is high and the resulting interval width is sufficiently narrow, then the value of the parameter is reasonably precise. A very wide confidence interval indicates a lot of uncertainty concerning the value of the parameter that we are estimating.

For example, let $\bar{x}$ and $s$ be the sample mean and sample standard deviation computed from a random sample obtained from a normal population with mean $\mu$. Then
a 100(1-α)% confidence interval for μ is \((\bar{x} - t_{α/2, n-1} \frac{s}{\sqrt{n}}, \bar{x} + t_{α/2, n-1} \frac{s}{\sqrt{n}})\), and so the interval width is \((2t_{α/2, n-1} \frac{s}{\sqrt{n}})\). In order to make the interval width half of the old width, we need to have \((2t_{α/2, n-1} \frac{s}{\sqrt{4n}})\) as the new width. In other words, in ordinary statistics, we need to collect four times the data to make the interval width half as wide.

We use the principle of Maximum Entropy for analyzing the available information in order to determine a unique posterior probability distribution. Let \(Θ\) be a multivariate discrete variable taking values on the domain \(Ω\). Then we can use the Maximum Entropy Sampling (MES) principle to derive a simple design criterion. The principle of maximum entropy is a method for analyzing the available information in order to determine a unique epistemic probability distribution. The principle of maximum entropy states that the entropy concept can also be used to infer a model for a probability distribution. It states that the least biased model that encodes the given information is that which maximizes the uncertainty measure \(H(p)\) while remaining consistent with this information. The maximum entropy principle is like other Bayesian methods in that it makes explicit use of prior information. This is an alternative to the methods of inference of classical statistics.

**Theorem 1:**

Let \(Y_i \mid (x_i, θ) = μ(x_i, θ) + ε_i\) where \(ε_i\) are independent and are identically distributed \((iid) N(0, σ^2)\) and \(σ^2\) is known; suppose \(Θ\) has a two point prior with mass \(1 - w\) and \(w\), \(0 < w < 1\), at \(θ_1\) and \(θ_2\) respectively. Suppose that each \(x_i\) varies in a compact subset \(X\)
of $\mathbb{R}^d$ and that $\mu(x, \theta)$ is a bounded function over $X$. Then an optimal design can be
found supported at a single point $x^*$ in $X$, with:

$$x^* = \arg \min_x \min \left\{ \mu(x, \theta_1) - \mu(x, \theta_2) \right\}^2$$

The following example illustrates the use of this criterion.

Example:

Consider the simple regression problem $Y_i \mid (x_i, \theta) = \mu(x_i, \theta) + \epsilon_i$ or $Y_i = \beta_0 + \beta_1 x_i + \epsilon_i$

$$\begin{bmatrix}
\beta_0 \\
\beta_1
\end{bmatrix} = \begin{bmatrix}
5 \\
7
\end{bmatrix} \text{ with probability } = w$$

$$\begin{bmatrix}
\beta_0 \\
\beta_1
\end{bmatrix} = \begin{bmatrix}
17 \\
3
\end{bmatrix} \text{ with probability } = 1 - w$$

In figure (1), we show a graph of $y = 5 + 7x$ and $y = 17 + 3x$

![Figure (1)](image)

We want to find the next best $x$ value at which to take a sample in order to make the best possible decision about which set of parameters is true. After choosing this new
$x$ value, we measure an observation (i.e. the $y$-value). We wish this $y$-value to give us the most information about the correct model. Therefore, by the given design criterion

$$x^* = \arg x \min \{\mu(x, \theta_1) - \mu(x, \theta_2)\}^2,$$

we get:

$$x^* = \arg x \min \{(5 + 7x) - (17 + 3x)\}^2 \Rightarrow x^* = \arg x \min \{144 - 96x + 16x^2\},$$

where $144 - 96x + 16x^2$ is a parabola, as illustrated in figure (2).

We can find the minimum of this parabola at $x = 3$. There is no maximum of this parabola. There we find a bounded space to find the maximum. For example if we wish to find the maximum in the bounded interval $[1,4]$ then the parabola has the maximum at $x = 1$, as illustrated in figure (3).
In addition, in figure (4) the circled area shows that the two lines from figure (1) have great separation at $x = 1$. Thus we find the maximum information by using the MES principle. That is, by choosing to take a new observation (y-value) at $x = 1$, we gain the most information about which of the two lines is correct.
III. IMPLEMENTATION

A. Our Model

Our model is based on the semi-empirical MISR (multi-angle imaging spectrometer) BRDF (bi-directional reflectance distribution function) Modified Rahman model. It has 3 parameters $r_0$, $k$, and $b$, and 4 input variables $\theta_s$, $\phi_s$, $\phi_v$, $\theta_v$, where $\theta$ is the zenith angle, and $\phi$ is the azimuth angle. Each angle is measured with respect to the point of view of the observer or instrument taking the observation (as indicated by the subscript $v$), or the sun (as indicated by the subscript $s$), as can be seen in figure 5 below. The model is:

$$R(\theta_s, \phi_s, \theta_v, \phi_v) = r_0 \left[ \cos(\theta_s) \cos(\theta_v) \left( \cos(\theta_s) + \cos(\theta_v) \right) \right]^{k-1} \exp(-b \cdot p(\Omega)) h(\theta_s, \theta_v, \phi_s, \phi_v)$$

where $R(\theta_s, \theta_v, \phi_s, \phi_v)$ is the reflectance, $h(\theta_s, \theta_v, \phi_s, \phi_v) = 1 + \frac{1 - r_0}{1 + G(\theta_s, \theta_v, \phi_s, \phi_v)}$, $G(\theta_s, \theta_v, \phi_s, \phi_v) = \sqrt{\tan^2(\theta_s) + \tan^2(\theta_v) - 2 \tan(\theta_s) \tan(\theta_v) \cos(\phi_s - \phi_v)}$, and $p(\Omega) = \cos(\theta_s) \cos(\theta_v) + \sin(\theta_s) \sin(\theta_v) \cos(\phi_s - \phi_v)$.
After obtaining initial satellite data, we will divide our algorithm into two parts:

1. **Regression Analysis:**

Regression is performed, and the following quantities are calculated:

   a. Estimates of the parameters $r_0$, $k$, and $b$.  
   
   b. $R^{-1}$: The covariance matrix of the estimated parameters  
   
   c. $\sigma$: the estimated standard deviation of the errors, which are assumed to be normal

2. **Maximization of Entropy:**

In this step, we find the best location (point) to which to send the helicopter. This is done by using the techniques suggested by Sebastiani and Wynn:

To maximize the amount of information about the posterior parameters, we should maximize the entropy of the distribution function. Mathematically, maximizing the entropy is achieved by maximizing the quantity

$$\log[\text{det}(X\Sigma^{-1}X + R)]$$

where $X$ is the design matrix of new observations to be taken, $X'$ is the transpose of the $X$ matrix, $\Sigma$ is the variance matrix of the error terms in the regression, and $R$ is the inverse of the prior covariance matrix of the parameters (Sebastiani and Wynn, 2000).

In our case, the $X$ matrix is a function of the input data ($\theta_0$, $\theta$, $\phi_0$, $\phi$), $R$ is taken to be the inverse of the posterior variance matrix of the parameters taken from the regression in step 1, and as the error terms are assumed to be independent and identically distributed, their covariance matrix $\Sigma$ is an identity matrix multiplied by the variance of the errors, which again is taken to be the estimated value from step 1, thus we take $\Sigma = \sigma^2 I$.  

22
With these values, and noting that finding the set of values that maximize
the log of a function is equivalent to finding the parameters that maximize the
function itself, we are left with maximizing:
\[ \det(X\Sigma^{-1}X + R) = \det\left(\frac{1}{\sigma^2}X'X + R\right) = \det(R) \det\left(1 + X'XR^{-1}\frac{1}{\sigma^2}\right). \]
Since \( R \) is a constant with respect to the parameters to be estimated, this leads us to
maximizing \( \det\left(1 + \frac{1}{\sigma^2}X'XR^{-1}\right) \). Hence, the determinant above is maximized
by finding the values of the input data \((\theta_v, \phi_v)\) for a given position of the sun at the
current time \((\theta_s, \phi_s)\) that maximizes this last quantity.

The robotic helicopter is programmed to go to the coordinates defined at the end
of step 2 in order to collect new observations. These observations are used to supplement
the original data set, and we return to step one. This loop is repeated until the parameter
estimates achieve the desired level of accuracy in their measurements, as measured by the
posterior covariance matrix of the parameters, \( R^{-1} \).

B. Linearizing the Model

A near linear fit of this model is accomplished by taking the natural logarithm of
\( R(\theta_z, \theta_v, \phi_z, \phi_v) \), which results in the following equation:
\[
\ln R(\theta_z, \theta_v, \phi_z, \phi_v) = \ln r_0 + (k-1) \ln \left[ \cos(\theta_z) \cos(\theta_v) \{ \cos(\theta_z) + \cos(\theta_v) \} \right] - b.p(\Omega)
+ \ln h(\theta_z, \theta_v, \phi_z, \phi_v)
\]
Note that aside from term \( \ln h(\theta_z, \theta_v, \phi_z, \phi_v) \), the function is linear in all three
parameters \( r_0, k, \) and \( b \). “Linearization” of \( \ln h \) (which contains a nonlinear \( r_0 \)) is
accomplished by using the value of $r_0$ from the previous iteration, where at iteration $n$ in the linear least-squares fit:

$$h^{(n)}(\theta_v, \phi, \phi_v) = 1 + \frac{1 - r_0^{(n-1)}}{1 + G^n(\theta_v, \phi, \phi_v)}$$

where $r_0^{(0)}$ is set equal to zero.

In Matlab, functions `regress` and `regstats` return the estimated linear fit of the parameter along with their confidence intervals (i.e. error bars), covariance matrix, and other statistics about the fit such as estimated function value and mean squared error. The function `nlinfit` estimates the coefficients of a nonlinear regression function using least squares. In order to get confidence intervals, the outputs of `nlinfit()` are used with `nlparci()`.

C. Experiment Setup

We assume that the sun is moving slightly between observations. For simplicity, we assume the sun move through $2\pi$ radians in a day, so it moves through

$$\frac{2\pi}{24 \times 60} \approx 0.004363323 \text{ radians per minute.}$$

Thus, we allow for the position of the sun move 0.01 radians between measurements (i.e., assume about 2 minutes before the helicopter can move between locations to take new measurements).

Assuming 10 random observation locations for $\phi$ and $\theta_v$ (zenith and azimuth of the viewpoint), and that $\phi_s$ and $\theta_s$ (zenith and azimuth of the sun) take on the values:

$$\left(\frac{\pi}{4} + 0.01 \times k \right) \text{ for } k = 1, 2, \ldots, 10.$$ 

These 10 random observations can then be examined in and of themselves, or be taken to be the initial random observations with which a first
regression analysis can be performed in order to begin the maximum entropy process outlined above.

Our simulation produces the following table of values:

<table>
<thead>
<tr>
<th>Observation</th>
<th>R</th>
<th>$\phi_s$</th>
<th>$\phi_v$</th>
<th>$\theta_s$</th>
<th>$\theta_v$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.1423</td>
<td>0.7954</td>
<td>-1.3666</td>
<td>0.7954</td>
<td>0.46996</td>
</tr>
<tr>
<td>2</td>
<td>0.17198</td>
<td>0.8054</td>
<td>0.81864</td>
<td>0.8054</td>
<td>1.1367</td>
</tr>
<tr>
<td>3</td>
<td>0.13773</td>
<td>0.8154</td>
<td>-1.1001</td>
<td>0.8154</td>
<td>1.3021</td>
</tr>
<tr>
<td>4</td>
<td>0.13922</td>
<td>0.8254</td>
<td>-1.5731</td>
<td>0.8254</td>
<td>1.4017</td>
</tr>
<tr>
<td>5</td>
<td>0.1381</td>
<td>0.8354</td>
<td>-1.1798</td>
<td>0.8354</td>
<td>0.87445</td>
</tr>
<tr>
<td>6</td>
<td>0.16613</td>
<td>0.8454</td>
<td>1.6239</td>
<td>0.8454</td>
<td>0.7061</td>
</tr>
<tr>
<td>7</td>
<td>0.15521</td>
<td>0.8554</td>
<td>1.0219</td>
<td>0.8554</td>
<td>0.12954</td>
</tr>
<tr>
<td>8</td>
<td>0.14739</td>
<td>0.8654</td>
<td>-1.1965</td>
<td>0.8654</td>
<td>0.065186</td>
</tr>
<tr>
<td>9</td>
<td>0.14211</td>
<td>0.8754</td>
<td>2.9088</td>
<td>0.8754</td>
<td>1.4279</td>
</tr>
<tr>
<td>10</td>
<td>0.14066</td>
<td>0.8854</td>
<td>-1.548</td>
<td>0.8854</td>
<td>1.4571</td>
</tr>
</tbody>
</table>

**IV. RESULTS**

**A. Estimation of the Parameters**

We perform the three statistical methods we mentioned in Chapter II using the model parameters $r_0 = 0.1$, $k = 0.9$, and $b = -0.1$ for this simulation.

With each parameter, the standard errors for linear regression are directly calculated from the regression process. However, with nonlinear regression and MLE, we use the Fisher Information Matrix as an approximation tool (lower bound).
<table>
<thead>
<tr>
<th>Method</th>
<th>Parameter</th>
<th>Point Estimate</th>
<th>Standard Error</th>
<th>95% CI (Confidence Interval)</th>
<th>CI Width</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear Regression</td>
<td>$r_0$</td>
<td>0.099706</td>
<td>0.012090</td>
<td>0.071113, 0.128299</td>
<td>0.0571857</td>
</tr>
<tr>
<td>Non L. Regression</td>
<td>$r_0$</td>
<td>0.099880</td>
<td>0.014693</td>
<td>0.065131, 0.134629</td>
<td>0.0694979</td>
</tr>
<tr>
<td>MLE</td>
<td>$r_0$</td>
<td>0.099706</td>
<td>0.014692</td>
<td>0.064959, 0.134453</td>
<td>0.0694932</td>
</tr>
<tr>
<td>Linear Regression</td>
<td>$k$</td>
<td>0.896183</td>
<td>0.006883</td>
<td>0.879905, 0.912461</td>
<td>0.0325566</td>
</tr>
<tr>
<td>Non L. Regression</td>
<td>$k$</td>
<td>0.897198</td>
<td>0.008207</td>
<td>0.877788, 0.916608</td>
<td>0.0388191</td>
</tr>
<tr>
<td>MLE</td>
<td>$k$</td>
<td>0.896182</td>
<td>0.008206</td>
<td>0.876775, 0.915589</td>
<td>0.0388144</td>
</tr>
<tr>
<td>Linear Regression</td>
<td>$b$</td>
<td>−0.082983</td>
<td>0.014922</td>
<td>−0.118274, −0.047692</td>
<td>0.0705811</td>
</tr>
<tr>
<td>Non L. Regression</td>
<td>$b$</td>
<td>−0.081389</td>
<td>0.017515</td>
<td>−0.122812, −0.039966</td>
<td>0.0828460</td>
</tr>
<tr>
<td>MLE</td>
<td>$b$</td>
<td>−0.082997</td>
<td>0.017515</td>
<td>−0.124420, −0.041574</td>
<td>0.0828460</td>
</tr>
</tbody>
</table>

According to the table of results, Nonlinear regression provides the best point estimate for parameters $r_0$ and $k$, while MLE and Linear Regression provide the best point estimate for parameter $b$. However, Linear Regression provides the narrowest 95% CI for all three parameters. Although linear regression did not provide the best point estimate for $r_0$ and $k$, its estimates were very close to those of Nonlinear Regression and MLE. Thus, we conclude there is no practical reason why we should not choose Linear Regression for parameter estimation.

Nonlinear regression and MLE have the same standard errors (SE) since both utilize the Fisher information matrix to obtain estimates of the standard errors. In theory, those estimates of the standard errors are lower bounds on the true errors of the model. With every parameter, we find: SE(using Fisher) > SE(linear regression).

**B. Maximization of Entropy**

We implemented Bayesian adaptive algorithm with following assumptions:
1. We have no initial data

2. The sun moves slightly between observations (0.01 radians between measurements)

3. The helicopter moves between locations to take the new measurements with the time interval of two minutes

After obtaining 10 random (as described in the section above) observation locations for $\phi_v$ and $\theta_v$ (zenith and azimuth of the viewpoint), and $\phi_s$ and $\theta_s$ (zenith and azimuth of the sun) values changing slightly between observation, we obtain the following table of values of parameter estimates and their standard errors.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Estimates</th>
<th>Standard errors choosing random observations</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_0$</td>
<td>0.09394008</td>
<td>0.01521365</td>
</tr>
<tr>
<td>$k$</td>
<td>0.8597921</td>
<td>0.00866166</td>
</tr>
<tr>
<td>$b$</td>
<td>-0.1868726</td>
<td>0.01877663</td>
</tr>
</tbody>
</table>

Alternatively, using our Bayesian adaptive algorithm we can choose the initial ten unique observation locations with the same positions of sun (moving slightly between observation). We take the identity matrix as an uninformative prior matrix $R^{-1} = \sigma^2 I$ and assume that error variance $\sigma^2$ is about $10^{-6}$, based on previous knowledge of the measurement instrument. The flowing table gives the locations for the viewpoint using location decider algorithm.
The robotic helicopter is sent to these locations, and collects reflectance data. Through the iterative regression process described earlier, the following parameter estimates are obtained.

<table>
<thead>
<tr>
<th>Observation</th>
<th>$\theta_i$</th>
<th>$\phi_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.5607963</td>
<td>0.7953982</td>
</tr>
<tr>
<td>2</td>
<td>1.1254487</td>
<td>-2.3361945</td>
</tr>
<tr>
<td>3</td>
<td>0.4902369</td>
<td>0.8153981</td>
</tr>
<tr>
<td>4</td>
<td>1.5607963</td>
<td>-2.3161945</td>
</tr>
<tr>
<td>5</td>
<td>1.5607963</td>
<td>-2.3061945</td>
</tr>
<tr>
<td>6</td>
<td>0.5193236</td>
<td>0.8453982</td>
</tr>
<tr>
<td>7</td>
<td>0.5295889</td>
<td>0.8553983</td>
</tr>
<tr>
<td>8</td>
<td>1.5607963</td>
<td>0.8653982</td>
</tr>
<tr>
<td>9</td>
<td>1.1441487</td>
<td>-2.2661945</td>
</tr>
<tr>
<td>10</td>
<td>1.5607963</td>
<td>-2.2561945</td>
</tr>
</tbody>
</table>

We can compare the standard error for the estimation of the parameters by choosing random locations at which to take observations or by choosing locations according to the Bayesian adaptive algorithm to determine where to collect the reflectance data. Our results show that the standard errors using Bayesian adaptive algorithm are much smaller than the standard errors for the estimates when the locations

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Estimates</th>
<th>Standard errors choosing random observations</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_0$</td>
<td>0.1089894</td>
<td>0.014492107</td>
</tr>
<tr>
<td>$K$</td>
<td>0.914301</td>
<td>0.003327716</td>
</tr>
<tr>
<td>$B$</td>
<td>-0.09810308</td>
<td>0.010880038</td>
</tr>
</tbody>
</table>
of the observations were random, so the parameter estimates are more accurate in using the Bayesian adaptive algorithm if a fixed number of observations are to be taken.

Alternatively, assume that we have some initial data from a satellite. If we have ten initial data points then we have two options.

1. We can take more observations at random or

2. We can pick only a few well chosen locations using the Bayesian adaptive algorithm at which to take more observations that would give us approximately the same parameter estimates.

We take the next observation based on Bayesian adaptive algorithm that uses the prior distribution for the variance of the parameter, matrix $R^{-1}$ and the estimated variance of the parameters from the regression on the first ten data points. It also calculates the variance of errors using the above input arguments.

Let us first take 30 more observations at random. So we have 40 data points including the 10 initial data points. The table below gives the numeric values of the parameter estimates and the standard errors.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Estimates</th>
<th>Standard errors choosing random observations</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_0$</td>
<td>0.09865134</td>
<td>0.008383804</td>
</tr>
<tr>
<td>$k$</td>
<td>0.8934277</td>
<td>0.003941684</td>
</tr>
<tr>
<td>$b$</td>
<td>-0.1467683</td>
<td>0.010459822</td>
</tr>
</tbody>
</table>

Compared to our previous experiment with 10 random locations, the standard errors for 30 random locations are about half as large, which is expected. Recall that we had noted earlier that in random sampling, approximately four times as much data is
required in order to obtain estimates that are twice as accurate when using random sampling. To investigate the Bayesian adaptive algorithm we pick 10 key locations at which to take observations. We proceed as follows:

1. Start with the first ten initial data points
2. Use the regression output as the prior distribution $R^{-1}$ and using the MSE for $\sigma^2$
3. Pick 10 new locations using the location decider algorithm (still assuming the sun is moving slightly between observations)

After deciding upon the locations we collect reflectance data. We get the following parameter estimates with the collected data.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Estimates</th>
<th>Standard errors choosing random observations</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_0$</td>
<td>0.1008255</td>
<td>0.010640548</td>
</tr>
<tr>
<td>$k$</td>
<td>0.8999695</td>
<td>0.002497330</td>
</tr>
<tr>
<td>$b$</td>
<td>-0.1071326</td>
<td>0.009354542</td>
</tr>
</tbody>
</table>

The table above shows that 10 carefully chosen locations at which to collect the reflectance observations provide approximately the same parameter estimates and standard errors, but the Bayesian adaptive algorithm is a lot more efficient to choose locations, as only 10 additional locations were required, rather than 30 additional random locations.
V. CONCLUSIONS

One of the main advantages of this particular implementation is its higher efficiency when compared to random sampling. The reduction rates of the standard errors, or the error bars, are much faster than those offered by mere random sampling. With these algorithms, future positions that will yield the most amount of information are found much more quickly. This efficiency amounts to savings of time and money during actual data collection and analysis.

The linear regression methods utilized in this implementation offered a big advantage. Linear regression yields systems of linear equations. These systems are in turn much simpler to solve. However, this implementation is not a true linear regression. It is in fact a near linear regression. The initial guesses are needed in order to complete the calculations, and then iterative procedures are employed.

VI. RECOMMENDATIONS AND FUTURE RESEARCH

A. Non-Normal Priors

A big part of the future work that should be done is to investigate the issue of working with data parameters that are not normal. In our model we make the assumption that the parameters, \( r_0 \), \( k \), and \( b \), are to be normally distributed. This assumption should not be interpreted as far fetched because of the central limit theorem, which states that with the collecting of more information on a system, that the estimates of the parameters of the system will become normally distributed, hence the name normal. In making assumptions though, questions often arise as to what to do if your assumptions are incorrect.
In terms of what we were doing, an interesting question arises as far as what to do if the distribution is not normal, or if the distribution is completely unknown. As mentioned in the previous section on Bayesian Statistics, we want to use our learning process on our system. In that section it was discussed as to how to use this system when our priors are known, but now they are not known and that is where the problem lies. We need a way to garner the best possible estimates of those priors. To solve problems like this, we can use sampling, that is, sampling in terms of Bayesian Estimation Methods. Using Bayesian Estimation Methods basically runs a recursive Bayesian system. It is a method for estimating unknown probability distributions over time using new information.

**B. Obtaining Posterior Distributions from “Messy” Systems**

A problem that one might run into is obtaining a posterior distribution from a system in which the distribution of its parameters is unknown because high-dimensional systems are often difficult to normalize. They can result in integrals that cannot be computed directly. In this case one would want to use Monte Carlo Markov Chain methods to evaluate the posterior. These are the sampling Bayesian Estimation Methods that were discussed earlier. There are several options in doing so; the most prevalent would be the Metropolis-Hastings Algorithms which generates a random walk using a density proportion and a rejection method for next moves. In special cases of this, Gibbs Sampling is usually preferred. It is preferable because unlike Metropolis-Hastings Algorithms, Gibbs does not have the random walk properties. Gibbs Sampling relies on conditional probabilities. Before getting into these, one should first have an understanding of Markov Chains and the Markov process.
A Markov Chain is essentially just any kind of sequence of random variables; with the range of the probability of any random variable occurring is the state space. Markov Chains are often times built in matrix form because computation is easier and more understandable this way. In matrix terms, our state space becomes a state matrix, where each probability is the probability of moving from one state to the next. In Bayesian terms, the original state matrix can be viewed as the prior distribution. In a Markov process we use Bayesian properties to change our probabilities when we are given new information about the probabilities. As was mentioned in the Bayesian section, this new information is also know as the likelihood, which in the Markov process is referred to as the transitional probabilities. These transitional probabilities, like the state probabilities can be formed into a transition matrix. One would then take the original state matrix and the transition matrix and use matrix multiplication to garner a new matrix. This new matrix, in Bayesian terms would be your posterior, in the Markov process, is your new state matrix. Markov Chain theory states that by recursively doing this process, i.e. taking the new state matrix and applying the transitional matrix to it, we will eventually reach a convergence, also known as a steady-state matrix.

Our Monte Carlo Markov Chain solution to the problems earlier stated essentially does the following: you first build a distribution from a Markov Chain, and then choose transition probabilities to use that distribution to create a new distribution of interest. We then have a system where each random variable is chosen based on the previous variable in the chain, and all that is needed is to run the system until convergence, also known as “walking” the Markov chain. In using this type of method, tough integration is not required and neither is normalization. That’s how MCMC methods work, but more
specifically how does Metropolis-Hastings Algorithms work? We start by taking an initial value from our system. It can be anything, as long as it lies inside the system. Then we take this initial value and sample a candidate point from a “jumping distribution”. The candidate point is equal to the probability of obtaining the new point, given we have observed the initial point. Once we have the point, we now need to know whether or not to accept or reject it, so we develop an acceptance ratio. The acceptance ratio is equal to the probability of the candidate point divided by the initial point. If our acceptance ratio is less than 1, we accept the candidate point with a probability of the acceptance ratio. If the acceptance ratio is greater than 1, then we accept the candidate point and set that point as the new point to use instead of the initial point. We then continue on by repeating this process, which does the following: When we accept a candidate point with a probability of the acceptance ratio obtained, this probability becomes the new probability of the move in the Markov Chain, and by repeating this using these transition probabilities we will reach the convergence as earlier discussed.

A special case of Metropolis-Hastings Algorithms is the Gibbs sampler, wherein the random value is always accepted during our rejection process. What remains is how to form a Markov Chain when the values converge to the target distribution. One of the main points of the Gibbs sampler is that it only considers univariate conditional distributions. Gibbs uses these because they are far easier to compute than complex joint distributions and often times have simpler forms, and thus it is easier to consider a sequence of conditional distributions than it is to obtain the marginal by integration of the joint distributions. The sampler starts out with an initial value, say \( y_0 \) for \( y \) and obtains \( x_0 \) by generating a random variable from the conditional distribution of \( x \) given that \( y \) is
equal to $y_0$. The sampler then uses that value obtained to generate a new value of $y_1$. This value comes from the conditional distribution of $y$ given $x$ is equal to $x_0$. The sampler proceeds like this, seemingly playing tennis where it volleys back and forth from one conditional distribution to the other. Repeating this process will eventually, like in the other MCMC methods bring us to a steady-state.

**C. Future Research**

A future research project could perhaps examine the situation where rather than sending the helicopter to the next best location, we should send it to the next best efficient location in terms of time, power, etc.

In addition to the application discussed in this paper, this research has potential applications in other fields, such as, military, medical and robotics for object recognition and tracking.

**APPENDIX A: References**


APPENDIX B: Matlab functions

Matlab functions are provided on the attached CD. An overview of the functions, their parameters, and their return values are provided below.

\[ \text{[betas cov mse] = linrahman(R,A)} \]
Input :
\[ A = [\text{phis phiv thetas thetav}] \text{ (angles MUST be in this particular order)} \]
R is the BRDF dependent vector
Output:
- This function calculates a "linearized" least squares fit of the Modified Rahman model parameters \( r_0, k, \text{ and } b \).
- It outputs the vector \( \text{betas} = [r_0\_\hat{\text{hat}} k\_\hat{\text{hat}} b\_\hat{\text{hat}}] \) (in that order), the covariance matrix, and \( \text{mse} \) (mean squared error).
- Included is a code for printing out 95% intervals and the standard errors. To activate the code for 95% CI and SE's, simply remove comments from the code at the end of this program.

\[ \text{betas = nlinrahman(R,A,beta0)} \]
Input:
\[ A = [\text{phis phiv thetas thetav}] \text{ (angles MUST be in this particular order)} \]
R is the BRDF dependent vector
beta0 = \([r_0 \text{ k b}] \) (in this order) is the initial guess of parameters
Output:
- This function calculates nonlinear least squares fit of the Modified Rahman model parameters \( r_0, k, \text{ and } b \).
- It outputs the vector \( \text{betas} = [r_0\_\hat{\text{hat}} k\_\hat{\text{hat}} b\_\hat{\text{hat}}] \) after calling \text{modrahman}.
- Included is a code for printing out 95% intervals for the estimated betas. To activate the code for 95% CI, simply remove the comment from the code at the end of this program.
\textbf{R\_hat = modrahman(beta0,A)}

\textbf{Input:}
beta0 = [r0 k b] (in this order) is the initial guess of parameters
\[A = [\text{phis phiv thetas thetav}]\] (angles MUST be in this particular order)

- This function accepts a matrix \(A = [\text{phis phiv thetas thetav}]\) and an initial guess 1x3 vector \(\text{beta0} = [r0 \, k \, b]\)
- Angles in matrix \(A\) MUST be in that particular order, and parameters in vector \(\text{beta0}\) MUST be in that particular order.

\textbf{Output:}
- It outputs the corresponding BRDF value of the Modified Rahman model of reflectance (\(R\_hat\))
- This function is intended to be used with nlinrahman (it will be called by nlinrahman)

\textbf{F = loglik(R,A,x0)}

\textbf{Input:}
\(R\) is the BRDF dependent vector
\[A = [\text{phis phiv thetas thetav}]\] (angles MUST be in this order)
\(x0\) is the vector of initial guesses of \(r0 \, k \, b\) \(s\) (MUST be in this order)

\textbf{Output:}
- This function calculates \(-\log(\text{Likelihood})\) of the error~\(N(0,s^2)\) given the vector of the parameters \(x0=[r0 \, k \, b \, s]\) (in that order) where \(s\) is the standard deviation of the error of the model, Error~\(N(0,s^2)\)
- \text{loglik} is intended to be used w/fminsearch to obtain MLE’s of \(r0 \, k \, b\) using \(\text{betas} = \text{fminsearch}(\text{@}(x) \, \text{loglik}(R,A,x),x0)\)

\[\text{[betas, s]} = \text{maxlik}(R,A,\text{beta0},t)\]

\textbf{Input:}
\(R\) is the BRDF dependent vector
\[A = [\text{phis phiv thetas thetav}]\] (angles MUST be in this order)
\(\text{beta0}\) is the vector of initial guesses = [\(r0 \, k \, b\)] (parameters MUST be in this order)
t is an initial guess of the standard deviation of the error.

Output:
- This function calculates the maximum likelihood estimators of \( r_0, k \) & \( b \)
- It also calculates the MLE of \( s \), the standard deviation of the error by calling the function loglik then finding it's maximum using fminsearch. The error is assumed to be Normal\((0,s^2)\)

\[ \text{newXMat} = \text{xMatFunction(newThetas, newThetav, newPhis, newPhiv)} \]

Input:
newThetas and newPhis are new positions of the Sun and newPhis and newPhiv are the new positions of the viewpoint.

Output:
- This function calculates the new X matrix where newXMat=[\(1 \ X_1 \ X_2\)]. Here X1 and X2 are the vectors of observations.
- This function is intended to be used in maxFunc().

\( \text{maxDet} = \text{maxFunc (newXMat, priorCovMat, varErr)} \)

Input:
newXMat contains three columns having the following format:
newXMat=[\(1 \ X_1 \ X_2\)]
PriorCovMat is the prior covariance matrix
varErr is the variance of standard errors

Output:
- The functions calculates the max \( \{\text{det}[I + (1/\sigma^2)X'X R^{-1}]\}\) and stores it in the variable maxDet.
- This function is called in minfunction( ) that calculates the minimum of the determinant corresponding the new view values

\( \text{minvalue} = \text{minFunction(newViewVals,ThetaS,PhiS,priorCovMat, varErr)} \)

Input:
newViewVals contains the position for the view in the same vector.
ThetaS and PhiS are the positions of the Sun.
priorCovMat is prior covariance matrix
varErr is the variance of standard errors.
Output:
- This function calculates the minimum i.e. $\max \{ \det \left[ I + \left( \frac{1}{\sigma^2} \right)X'X R^{-1} \right] \}$
corresponding the new view values.
- This function is being called in ReadInputMat( )

[newPhiV, newThetaV] = newPositions(newSunTheta, newSunPhi, priorCovMat, varErr)

Input:
newSunTheta and newSunPhi are the new positions of the Sun.
priorCovMat is the prior covariance matrix.
varErr is variance of standard errors
Output:
- This function calculates the new values of theta and phi for the viewpoint i.e. new location for the view at which to take observations.
- This function uses a in built matlab function fmincon( ) to find a minimum of a constrained nonlinear multivariable function.
- fmincon( ) needs an initial guess for the locations of the view and after a maximum number of iterations returns the new locations for viewpoint.