PASSIVE EMITTER LOCATION USING DIGITAL TERRAIN DATA

BY

TINA L CHOW

BS, Electrical and Computer Engineering
Carnegie Mellon University, 1998

THESIS

Submitted in partial fulfillment of the requirements for the degree of
Master of Science in Electrical Engineering in the
Thomas J Watson School of Engineering of
Binghamton University
State University of New York
2001

Accepted in partial fulfillment of the requirements for the degree of
Master of Science in Electrical Engineering in the
Thomas J Watson School of Engineering of
Binghamton University
State University of New York
2001

Mark L. Fowler______________________________________ October ??, 2001
Advisor, Electrical Engineering Department

Eva Wu____ October ??, 2001
Committee Member, Electrical Engineering Department

Edward Li____ October ??, 2001
Committee Member, Electrical Engineering Department

ABSTRACT

Passive location of a stationary emitter using a single moving platform has always been an important task for many applications. A majority of emitter estimation problems are currently performed using bearing (or angle of arrival) measurements, but a particularly efficient method uses the measured time-history of Doppler-shifted frequency. However, obtaining accurate z-resolution of the emitter has been a nagging problem. In this paper, it is shown that by integrating digital terrain data with frequency measurements, we can obtain extremely accurate z-estimates (since we have assume a stationary emitter and have the elevation information from the map) as well as improved x and y accuracy in almost all cases. Accuracy is measured in terms of standard deviation compared to the actual emitter position, and these runs were simulated in MATLAB over 200 Monte Carlo simulations.

Table of Contents

Table of Figures

Table of Tables

Chapter 1 – Introduction

1.1 Background Information on Emitter Location

Passive location of an emitter has always been an important task for many applications. For military applications, both stealth and the precise location of the target or threat must be determined in order to establish appropriate evasive or counter measures. The emitter's coordinates can be estimated by a single moving observation system, which receives signals (measured signal parameters) from the emitter or by using multiple platforms. Signal parameters that are used to locate emitters include amplitude, phase shift, time delay, and frequency shift. These signal parameters correspond to geometric quantities: angle/direction of arrival (obtained from amplitude or phase shifts), range difference from the emitter to two measurement points (obtainable from time difference of arrival (TDOA) or frequency difference of arrival (FDOA) measurements), and derivative of range difference. The actual location technique depends upon the kind of measured signal parameters, the measurement techniques, and the data collection procedures. In turn, these location techniques determine types of location algorithms, which are defined by the assumed observation model, estimation method, and the procedures of numerical computations. The location algorithm most commonly used is based on the maximum likelihood (ML) or least squares estimators [6]. If the probability density function (pdf) of the measurement errors is known, then the ML estimator can be used; otherwise the least squares estimator is used.

Figure 1 shows a single-platform emitter location scenario. The platform, flying at a certain altitude, velocity, scale factor of acceleration due to gravity (g), and certain time interval of data collection, may obtain different signal measurements (frequency, bearing, azimuth, elevation, etc). The emitter is located at some distance (Range) away from the platform, and the map data (to be described further in Chapter 2) is defined as elevations at certain x-y intervals (resolution).

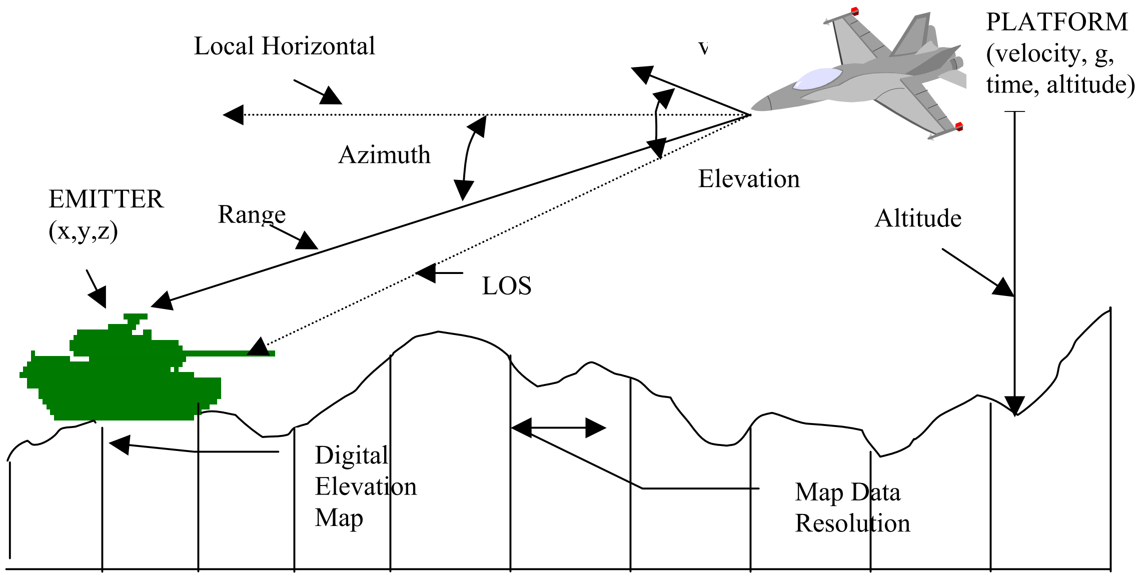

Figure 1 - Passive Emitter Location Problem with one platform

In practice, emitter location techniques have leaned towards using angular measurements (triangulation or azimuth/elevation) or range-difference. Most often used is the angle of arrival (AOA, or equivalently bearing) measurement. AOA involves the use of amplitude or phase shift as the measurement parameters. Data collection procedures classified under AOA include the azimuth/elevation, circulation, and triangulation location techniques [6]. AOA location techniques can be used in both stationary and moving measurement systems, and do not require simultaneous operation of multiple measurement platforms (one is sufficient). However, AOA location techniques have their own benefits and drawbacks. Although the azimuth/elevation technique is very good for locating stationary emitters from the air and needs only a single aircraft platform, it requires high flight altitudes to obtain accurate results; triangulation is good in locating both stationary and moving targets in the presence of true random measurement errors but is susceptible to systematic measurement errors [6]. Circulation provides an almost exact opposite of the benefits and drawbacks of triangulation: it eliminates errors caused by systematic measurement errors but is susceptible to random measurement errors and also introduces an ambiguity caused by the intersection of circles [6].

Plausible techniques used to determine the AOA include amplitude maximum, amplitude comparison, and phase comparison (interferometry). The former two methods are sensitive to the multipath effect, as well as susceptible to either amplitude fluctuations (amplitude maximum) or the necessity of increasing hardware complexity (amplitude comparison requires two or more receiving channels). Phase comparison, on the other hand, provides very high AOA accuracy and is quite insensitive to multipath effects. However, it is a very "expensive" method, requiring high-quality, sensitive receivers and sophisticated signal processing methods at microwave frequencies [4].

Two other emitter location techniques, range difference and differential Doppler, are based on TDOA and FDOA measurements, respectively. For TDOA, the AOA calculation is independent of the signal frequency, phase ambiguities are not present, and multipath reflections are not present because of short time interval measurements. It does, however, require very accurate time delay measurements. On the other hand, FDOA or differential Doppler (DD) requires receiver motion to extract information. Higher receiver speeds and longer pulse widths of the received signal provide more accurate results [4], hence it is useful in locating continuous wave (CW) signals of constant frequency but requires highly synchronized and stable measuring receivers.

Locating a stationary emitter can be accomplished using frequency measurements recorded from a single moving platform. A 2D analysis analyzing emitter location performance between frequency measurements alone and frequency with bearing measurements [1] indicates that the combined set of bearing and frequency measurements provides improved accuracy over the cases of either bearing or frequency measurements alone. Becker points out that because the directions of the principal axes of the error ellipses for the bearing measurement and frequency measurement analyses do not coincide, one can expect significant accuracy improvements with the simultaneous processing of the two measurements. He shows this by performing a Cramer-Rao analysis of single measurement cases and shows that the size and orientation of these ellipsoids can be described in terms of the eigenvectors and eigenvalues of the Fisher information matrix.

The efficiency of using Doppler-shifted frequency measurements makes it a good method for passive emitter location. Instead of using an array of sensors to perform line of sight (LOS), bearing, or angle measurements, an individual sensor can extract information on emitter position, which reduces equipment and calibration costs and enables independent missions. In this analysis, we plan to integrate stored digital terrain data with these frequency measurements in a recursive least-square error algorithm and show that this provides a more accurate estimate of the target location than the case of frequency measurements with some level of *a priori* knowledge of the emitter altitude. The least squares method with terrain data method does not require the extra *a priori* knowledge necessary for the Kalman filter described by Paradowski [6] and Spingarn [7]. There are also issues of an estimation bias that decreases estimation effectiveness. The use of a Kalman filter in the location algorithm and using stored digital terrain data for passive location is described in the study performed by Collins and Baird [3]. Their location algorithm integrates measured data from sensors (azimuth and elevation) and stored terrain data to calculate a least square error estimate of the emitter position. Their LOS intersection algorithm searches along the LOS vector until the closest terrain intersection is found. When this coarse point of intersection is obtained, their algorithm searches through the data elevation map points and uses interpolation between these points to obtain a finer resolution intersection. The three-state Kalman filter is used to calculate the general solution to the iterative minimum mean-square error estimation problem. Different types of passive sensors (helmet mounted, IR sensor, RF sensors), flight altitudes, and terrain types were studied to analyze performance of the terrain-aided estimation. The mean ranging error was used to measure performance: in most cases, the ranging error was quite small, but the performance depended upon the type of sensor that was used for that particular scenario. Another performance characteristic was that non-flat terrain provided decent accuracy over short collection intervals.

1.2 Motivation for This Analysis

This analysis came about to compare the performance of emitter location using an error ellipsoid analysis under different levels of *a priori* information on the emitter's altitude. When using a platform maneuver of a concave or convex circular path with respect to the emitter [4] and having no knowledge of the emitter's z-coordinate, the performance appeared to be very similar for the two paths. However, when one has some knowledge about the emitter's altitude, whether it is complete knowledge of the emitter's altitude or having the terrain data elevations, the performance differed for both paths and was dependent on the platform's altitude. This was somewhat unexpected; as concave and convex paths with respect to the emitter give frequency measurements that are approximately time flipped versions of the other. The down-range slope also influenced accuracy, as positive slopes tended to have an effect similar to an increase in platform altitude. Therefore by having information about the terrain slopes or a priori information of the emitter's z-coordinate, it seems possible that improved x, y, and z accuracy for emitter location can be obtained. We are motivated by this possibility and thus want to exploit map data to obtain a more accurate estimate of an emitter's position.

In practice, it is quite difficult to obtain an accurate estimate of the emitter's z coordinate. By integrating map data into the location algorithm, we automatically have the map altitudes, and hence good estimates of the emitter's z location if we have decent estimates of x and y. In addition, this also provides improved x and y accuracy, indicating that the frequency measurements plus map data method should be studied more in-depth.

In our analysis, we assume that there is just one target to be located, and that the input measurement noise is Gaussian, uncorrelated from measurement to measurement, and has constant variance.

Chapter 2 – Mathematical Concepts

2.1 Digital Terrain Data

2.1.1 Actual Available Map Data

Actual digital terrain elevation data (DTED) obtainable (public domain) from the United States Geological Services is of low resolution: currently, map data of the United States can found at a 90-meter resolution [10]. A 30-meter resolution will be released sometime in the future, and only 50 percent of the US is available at higher resolutions. The US Defense Mapping Agency 's "high resolution" (90 meters) map of the entire world is not available to the general public. The only public domain digital elevation data for the entire world can only be found at a very coarse resolution (1 km).

2.1.2 The Map Used

The map data used in the simulations done here is simply a flat, sloped surface, where actual elevations are defined at intervals of 100 meters in x and y. The slopes of the maps are defined as either dz/dx or dz/dy slopes (see experiment setup). Delineating between down-range and cross-range slopes was done to determine whether particular values or orientations of slopes affected the location accuracy. No interpolation was used to estimate points that fell between values that had defined elevations: an average of the 9 closest map data points was taken (see Figure 2). The 9 grid points are obtained by using the MATLAB round() command (which rounds to the nearest integer) and then obtaining the map grid values by taking the difference between the estimated emitter position and the lowest x/y value range and dividing this result by the map resolution for that dimension (x or y). These grid points are obtained by adding [-1 0 1] to the "rounded" x and y grid points: for example, "5" is the rounded value of "E," the actual emitter position obtained from simulation (see Figure 2). This results in the 9 data points marked "1" through "9," one of which is the original rounded value ("5"). Taking the average of these nine points may introduce some error, especially if the map resolution is low and there are drastic changes in the terrain, so implementing some type of interpolation would probably be nicer in terms of getting an accurate elevation. This would involve the addition of more circuitry or code to implement. However, if the map resolution were high enough, and the terrain elevations do not change drastically, then a lack of interpolation will not introduce severe error penalties.

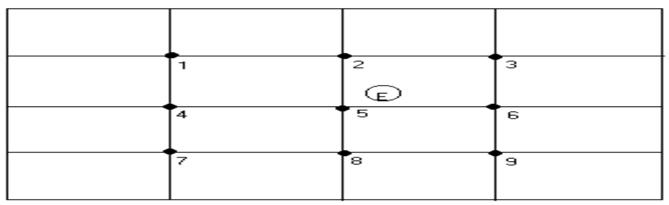

Figure 2 - Selecting Map Grid Points

2.2 Least Squares

Data is generally subject to measurement errors (noise), so we need to determine which fitting model is appropriate: we want to fit these data points to a certain model that predicts a relationship between measured independent and dependent variables. Least squares fitting is a maximum likelihood (ML) estimation of fitted parameters if measurement errors are independent and normally distributed with constant standard deviation [9]. No assumptions are made about the model's (non)-linearity in its parameters.

For the standard linear least squares problem, we want to fit a set of data points (x_i, y_i) i=1..n, to a model that can be a linear combination of any M specified functions $X_k(x)$ of some variable x, e.g.

$$y(x)=a_1+a_2x+a_3x^2+\ldots+a_Mx^{M-1},$$

where in this case the M specified functions are $X_k(x)=x^{k-1}$ for k=1,2..,M. By choosing the a_k values, we want this model to be fitted about the given set of data points so that the sum of the squares of the difference between the measured data and the model is minimized. The sum of squares is given by S, where

$$S = \sum_{i=1}^{N} \left[\frac{y_i - \sum_{k=1}^{M} a_k X_k(x_i)}{\sigma_i} \right]^2$$

and σ_i is the measurement standard deviation of the i^{th} data point y_i. If the σ_i are unknown, they can be set to 1. To minimize S, we need to take its derivative with respect to each a_k, set the derivatives to zero and then solve. The resulting equations,

$$\sum_{i=1}^{N} \frac{1}{\sigma_i^2} \left[y_i - \sum_{j=1}^{M} a_j X_j(x_i) \right] X_k(x_i) = 0, \text{for } k = 1,..., M$$

are called the normal equations. The solution for this normal equation can be obtained by LU decomposition and then using back-substitution, by Cholesky decomposition, or by Gauss-Jordan elimination. Since the solution of a least squares problem using normal equations is susceptible to round-off error, singular value decomposition (SVD) should be used for all but very "easy" least squares problems [9].

When the model depends nonlinearly on the set of unknown parameters, we need to perform an iterative procedure for minimization. A χ^2 merit function is defined, and given initial values for the parameters, a procedure is developed so that the trial solution improves. This is repeated until χ^2 stops decreasing: this is usually accomplished by taking the gradient of the χ^2 function. The second derivative of the χ^2 merit function, the Hessian matrix, is also necessary to compute the parameters that will minimize the chi-squared merit function. Since the computational complexity of any nonlinear algorithm depends on the number of parameters that needs to be optimized, it is beneficial to try to reduce the number of parameters. For emitter location using a bearing model, a Taylor series expansion about the initial estimate of the state is done to linearize an otherwise nonlinear function of the emitter position [7]. Only the first order terms are retained. This is effective when there are very small perturbation errors (bearing errors must be small) [8].

Nonlinear least squares estimation is an iterative process where all estimates are recalculated every time a revised estimate is obtained. It requires an *a priori* estimate on the emitter position, but does not require an *a priori* estimate of the state vector covariance matrix like the Kalman filter or extended Kalman filter. Though the least squares filter is a batch process, as opposed to a sequential one for the Kalman filters, and therefore implies more processing time, it is not influenced by the covariance matrix, which has implications on estimate accuracy when the number of observations is small. It is, however, more computationally complex than the simple intersection of lines of position (LOP), and does not necessarily assure convergence of the solution. However, a failure to converge is easy to detect [4], the statistical spread of the solution is easy to determine, and even poor initial estimates will not prevent the convergence of a solution.

Since the frequency measurements do not depend linearly on the location parameters, an iterative algorithm, based upon the gradient, must be used. The gradient using the data and measurement model is calculated for an estimate on the least squares inverse cost surface. This will indicate which direction to 'move' the estimate, and this procedure is repeated until the update becomes reasonably small.

2.3 Doppler Location Algorithm

An emitter's geolocation can be estimated by using the frequencies measured over a time period and the platform's navigational data (its path) over the same time period. The noise-free frequency measurements can be modeled by the following equation [11]:

$$f(t,\mathbf{x}) = f_o - \frac{f_o}{c}\left[\frac{V_x(X_p(t)-X)+V_y(Y_p(t)-Y)+V_z(Z_p(t)-Z)}{\sqrt{(X_p(t)-X)^2+(Y_p(t)-Y)^2+(Z_p(t)-Z)^2}}\right] \tag{1}$$

where

$f(t,\mathbf{x})$ is the noise-free measured frequency at time t
$\mathbf{x}$ = [X Y Z f_o], is a vector that describes the emitter's geolocation
f_o = transmitted frequency
c = speed of light, approximately $3\text{x}10^8$ m/s
(X_p, Y_p, Z_p) = platform's antenna position
(V_x, V_y, V_z) = platform's antenna velocity

The actual frequency measurements $\tilde{f}(t,\mathbf{x})$ are noisy versions of equation (1):

$$\tilde{f}(t,\mathbf{x}) = f(t,\mathbf{x}) + v(t),$$

where $v(t)$ is the measurement noise process that is assumed to be Gaussian, is uncorrelated from measurement to measurement, and has constant variance. Since this model is nonlinear in $\mathbf{x}$, it has no closed-form solution. We can linearize this model by expanding $f(t,\mathbf{x})$ in a Taylor series around a current estimate $\hat{\mathbf{x}}_n$ and then discarding the terms that are of second degree and higher. First we define the time vector $\mathbf{t}$=[t_1 t_2 … t_N]. We can re-express the measurement noise as a vector $\mathbf{v}$ by including the time vector elements ($\mathbf{v}$=$v(\mathbf{t})$). The noiseless frequency measurements collected over time are

$$\mathbf{f}(\mathbf{x}) = f(\mathbf{t},\mathbf{x}) \tag{2a}$$

Adding the noisy frequencies

$$\tilde{\mathbf{f}}(\mathbf{x}) = \mathbf{f}(\mathbf{x}) + \mathbf{v} \tag{2b}$$

We then linearize using the Taylor series expansion.

$$\tilde{\mathbf{f}}(\mathbf{x}) = \hat{\mathbf{f}}(\hat{\mathbf{x}}_n) + [\mathbf{x} - \hat{\mathbf{x}}_n]\frac{\partial}{\partial x}\mathbf{f}(\mathbf{x})\Big|_{x=\hat{\mathbf{x}}_n} + \mathbf{v} + \textit{discards} \tag{2c}$$

This is equivalent to

$$\tilde{\mathbf{f}}(\mathbf{x}) \approx \hat{\mathbf{f}}(\hat{\mathbf{x}}_n) + \mathbf{H}[\mathbf{x} - \hat{\mathbf{x}}_n] + \mathbf{v} \tag{2d}$$

where the collection of the measured frequencies can be defined as

$$\tilde{\mathbf{f}}(\mathbf{x}) = \left[\tilde{f}(t_1,\mathbf{x}) \quad \tilde{f}(t_2,\mathbf{x}) \quad \ldots \quad \tilde{f}(t_N,\mathbf{x})\right]^T \tag{2e}$$

and the collection of all the predicted frequencies can be defined as

$$\hat{\mathbf{f}}(\hat{\mathbf{x}}) = \left[\hat{f}(t_1,\hat{\mathbf{x}}) \quad \hat{f}(t_2,\hat{\mathbf{x}}) \quad \ldots \quad \hat{f}(t_N,\hat{\mathbf{x}})\right]^T \tag{2f}$$

and **v** is a vector of noise samples and **H** is an Nx4 matrix

$$\mathbf{H}=\frac{\partial}{\partial \mathbf{x}} f\,(\mathbf{t},\mathbf{x})\big|_{x=\hat{x}_n} =[\mathbf{h}_1|\mathbf{h}_2|\mathbf{h}_3|\mathbf{h}_4]. \tag{3}$$

Each column $\mathbf{h}_i$ of **H** is an Nx1 vector of partial derivatives with respect to one of the parameters at times **t**, respectively, evaluated at the current estimate of $\hat{\mathbf{x}}_n$. To determine each column and element of **H**, we first define

$$\begin{aligned}
\Delta\hat{X}_n(t_j) &= X_p(t_j) - \hat{X}_n \\
\Delta\hat{Y}_n(t_j) &= Y_p(t_j) - \hat{Y}_n \\
\Delta\hat{Z}_n(t_j) &= Z_p(t_j) - \hat{Z}_n \\
\hat{R}_n(t_j) &= \sqrt{\Delta\hat{X}_n^{\,2}(t_j) + \Delta\hat{Y}_n^{\,2}(t_j) + \Delta\hat{Z}_n^{\,2}(t_j)}
\end{aligned} \tag{4}$$

where $\hat{R}_n(t_j)$ is the distance between the platform and the current estimated emitter location at time t_j.. So, for the each column, the j[th] elements are as follows:

$$\begin{aligned}
\mathbf{h}_1(j) &= \frac{\partial}{\partial X} f(t_j,\mathbf{x}) \quad \big|_{x=\hat{x}_n} = -\frac{\hat{f}_o}{c}\left[\frac{-V_x(t_j)}{\hat{R}_j} + \frac{\Delta\hat{X}_n(t_j)[V_x(t_j)\Delta\hat{X}_n(t_j)+V_y(t_j)\Delta\hat{Y}_n(t_j)+V_z(t_j)\Delta\hat{Z}_n(t_j)]}{R_j^{\,3}}\right] \\
\mathbf{h}_2(j) &= \frac{\partial}{\partial Y} f(t_j,\mathbf{x}) \quad \big|_{x=\hat{x}_n} = -\frac{\hat{f}_o}{c}\left[\frac{-V_y(t_j)}{\hat{R}_j} + \frac{\Delta\hat{Y}_n(t_j)[V_x(t_j)\Delta\hat{X}_n(t_j)+V_y(t_j)\Delta\hat{Y}_n(t_j)+V_z(t_j)\Delta\hat{Z}_n(t_j)]}{R_j^{\,3}}\right] \\
\mathbf{h}_3(j) &= \frac{\partial}{\partial Z} f(t_j,\mathbf{x}) \quad \big|_{x=\hat{x}_n} = -\frac{\hat{f}_o}{c}\left[\frac{-V_z(t_j)}{\hat{R}_j} + \frac{\Delta\hat{Z}_n(t_j)[V_x(t_j)\Delta\hat{X}_n(t_j)+V_y(t_j)\Delta\hat{Y}_n(t_j)+V_z(t_j)\Delta\hat{Z}_n(t_j)]}{R_j^{\,3}}\right] \\
\mathbf{h}_4(j) &= \frac{\partial}{\partial f_o} f(t_j,\mathbf{x}) \quad \big|_{x=\hat{x}_n} = 1-\frac{1}{c}\left[\frac{[V_x(t_j)\Delta\hat{X}_n(t_j)+V_y(t_j)\Delta\hat{Y}_n(t_j)+V_z(t_j)\Delta\hat{Z}_n(t_j)]}{R_j^{\,3}}\right] \approx 1
\end{aligned} \tag{5}$$

Since the first term of the Taylor series expansion in (2d) is the predicted frequency vector, we can subtract it from both sides of the equation to obtain an expression in a difference quantity $\Delta\mathbf{x}$.

$$\Delta\mathbf{f}(\hat{\mathbf{x}}_n) \approx \mathbf{H}\Delta\mathbf{x}+\mathbf{v} \tag{6}$$

where $\Delta\mathbf{x} \equiv \mathbf{x} - \hat{\mathbf{x}}_n$. This results in a linear model in terms of a known difference quantity, $\Delta\mathbf{f}(\hat{\mathbf{x}}_n)$ and an unknown difference quantity, $\Delta\mathbf{x}$. The least squares estimate of $\Delta\mathbf{x}$ can then be calculated from

$$\Delta\hat{\mathbf{x}} = (\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H})^{-1}\mathbf{H}^T\mathbf{R}^{-1}\,\Delta\mathbf{f}(\hat{\mathbf{x}}_n) \tag{7}$$

where $\mathbf{R}$ is a diagonal matrix of the frequency measurement variances $\sigma_1^2, \sigma_2^2, \ldots \sigma_N^2$.

We can update the estimate by

$$\hat{\mathbf{x}}_{n+1} = \hat{\mathbf{x}}_n + \Delta\hat{\mathbf{x}} \tag{8}$$

Given an initial estimate $\hat{\mathbf{x}}_1$ it can be iteratively improved using the recursion in (7) and (8); convergence can be assessed by monitoring the size of the update $\Delta\mathbf{x}$ – then the recursion can be terminated either when the size of $\Delta\mathbf{x}$ drops below a specified level or after a specified number of iterations.

In our analysis, we only have assumed having a single initial estimate. However, in some cases you may have several alternative initial estimates; then each estimate may be individually iterated until convergence and then the best solution can be selected by comparing the least squares error (cost). This is defined as

$$C(\hat{\mathbf{x}}) = \Delta\mathbf{f}^T(\hat{\mathbf{x}})\mathbf{R}^{-1}\Delta\mathbf{f}(\hat{\mathbf{x}}) \tag{9}$$

Substituting the actual elements of the $\mathbf{R}$ matrix, the cost can be rewritten as follows:

$$C(\hat{\mathbf{x}}) = \sum_{n=1}^{N} \frac{(\tilde{f}(t_n,\mathbf{x}) - \hat{f}(t_n,\hat{\mathbf{x}}))^2}{\sigma_n^2} \tag{10}$$

After convergence occurs, the cost function can be evaluated for the solutions and lowest cost value is selected. The corresponding emitter location for this cost value is selected.

Chapter 3 – Experimental Setup

3.1 Emitter Location Algorithm without Using Map Data

Our MATLAB simulation involves 20 location iterations and a 200 run Monte Carlo simulation. We first obtain an initial estimate for our emitter position as follows: the true values of the emitter position and operating frequency are randomly perturbed to obtain our initial estimates - the frequency f_o is perturbed by ±10MHz, x and y by ±10km, and z by ±500m. Then for our specified time instances, we need to compute the frequencies corresponding to our estimated emitter position (and using our platform navigation data). We then calculate the residuals by taking the difference between the calculated frequencies (from our initial estimate) and the actual measured frequencies. Next, we compute the Jacobian matrix that corresponds to (1). This results in an Nx4 matrix, whose columns are defined by (5). The best estimate is calculated using (7), with **R** being set to the identity matrix (or a multiple of, since we've assumed that our noise has constant variance). This estimate is used for the next iteration, which repeats until the maximum number (20) of iterations has been reached. Figure 3 shows a flow diagram of this process, with the shaded box as the extra step that is added when using map data.

The MATLAB code used is listed in the Appendix.

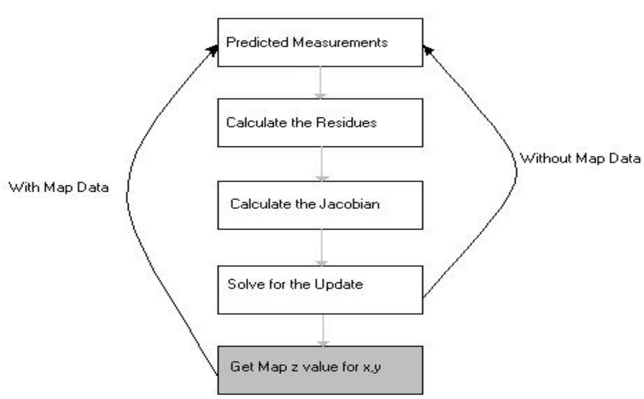

Figure 3 - Flow Chart of Emitter Position Estimation

3.2 Emitter Location Algorithm Using Map Data

We can use the map data because we assume that we are locating a stationary emitter. Since the emitter doesn't move, we assume it is sitting on the ground and by using the map elevations, we automatically will have obtained an estimate of the z position. Once an estimate of (x, y) is found, either from the initial guess or from the Doppler location processing (after having gone through one iteration loop), the calculated z is discarded regardless of what is obtained. Using the generated x and y coordinate estimates, the corresponding map value for z is extracted. This new z value, along with the generated x and y estimates, are used as our new update values in our location loop. For the map-augmented simulation, we drop the column of the Jacobian matrix **H** that corresponds to the partial derivative with respect to z, as our z coordinates are obtained from the map data.

3.3 Experiment Configuration

Figure 4 shows a sketch of the experiment setup. The platform is flying at some altitude towards the emitter, which is stationary and on the ground. The map data range is set such that the emitter is always located within the map, even for situations where the simulations run over different emitter positions along a set range. The map is set to a constant slope, either positive or negative value (sloped towards/away from the emitter/platform). The emitter frequency, f_o, can be anywhere between 1 and 10 GHz.

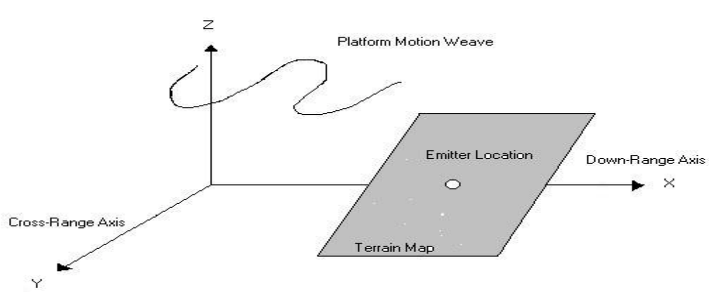

Figure 4 - Orientation of Axes, Map, and Platform

3.4 Description of Platform Motion

The platform motion can be described as a sinusoidal weave movement, where we specify the variables g, T, and alt_kft. The altitude, alt_kft, may range anywhere up to 20kft (though for some of the simulations the values are even higher to see at what altitude the performance degrades). The variable g is a scale factor of acceleration due to gravity, and ranges anywhere from 1 to 3. T is the duration of time that the platform flies and collects data, and ranges anywhere from 20 to 100 seconds.

The velocity of the platform's motion was set to a value of 200 m/s (though it can be changed, ranging from 100 to 300 m/s). It uses a constant acceleration to maneuver the turns of the weave and has no vertical acceleration.

3.5 Error Ellipsoids and Confidence Limits

Estimation errors are generally described by the error probability density function (pdf), but it can be more convenient to describe a confidence region, a multidimensional generalization of the confidence interval for the estimates. A confidence region (or confidence interval) is a region of M dimensional space (hopefully small) that contains a certain (hopefully large) percentage of the total probability distribution. In our case, the region is in the shape of an ellipsoid, which is exact for a Gaussian pdf. The size of the ellipsoid indicates the relative magnitude of the error, and the ellipsoid can be found through the eigenvalues and eigenvectors of **J**, the Fisher information matrix [1,4,9], given by

$$\mathbf{J}=\mathbf{H}^T\mathbf{\Sigma}^{-1}\mathbf{H}, \qquad (11)$$

where $\mathbf{\Sigma}$ is simply the error covariance matrix. Since we've assumed constant variance, this matrix is just a multiple of the identity matrix, or simply just the identity matrix. The eigenvectors of **J** define the orientation of the ellipsoid axes and the reciprocal of the square root of the eigenvalues determine the ellipsoid axes length.

3.6 Scatter Plots

A scatter plot is a plot of the x-y estimation error for the Monte Carlo runs. The use of scatter plots enables us to view the error in the estimates of xe and ye on each run, and get a good idea of the performance of the Doppler location algorithm with and without the map data. In Figures 5 and 6, the plots show scatter plots for different emitter angles relative to the x-axis (see Figure 7) for the case of 200 Monte Carlo runs. The angles used are printed to the left of each plot.

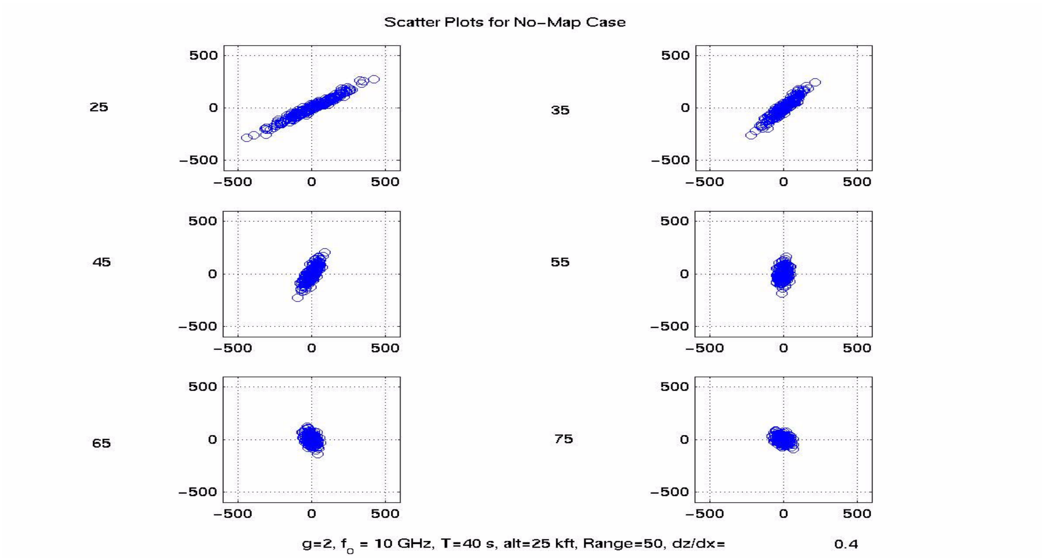

Figure 5 - 200 Estimates of the x,y without using map data for varying angles

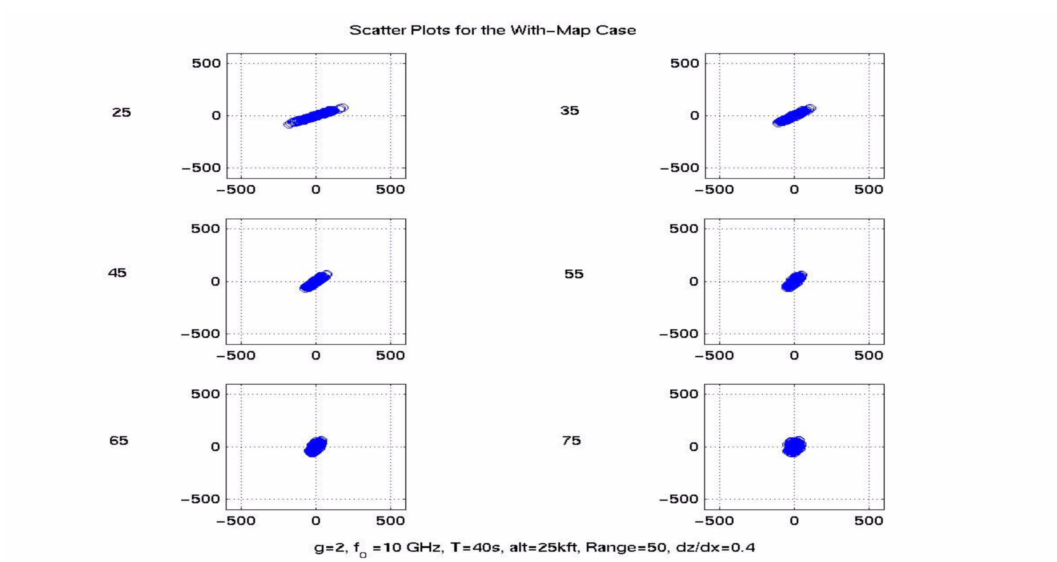

Figure 6 - 200 Estimates of x,y using map data for varying angles

The scatter plots show that the orientation of the emitter estimates follows the expected theory: that for locations nearly directly in front of the platform, the x scatter is large and the y scatter is small, and as the emitter position nears a location to either side of the platform, the x and y errors should be roughly equal. It is expected that for emitter locations near the zero degree mark (y-component=0), the scatter of estimates will be the highest, and for locations roughly to either side of the platform (near ninety degrees; x-component=0), the scatter will be lowest. We can also see that using the map data definitely makes the scatter of the estimates smaller.

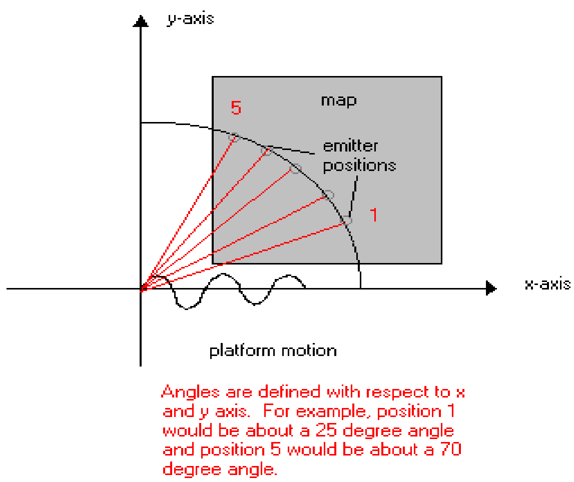

Figure 7 - XY Plane View and Angle Definitions

From the scatter plots, we can see that as the angle variable changes from near the 0 degree mark to the 90 degree mark, the error ellipse bound gets smaller: it changes from a long cigar-shaped ellipse with most of the scatter error along the x-axis to one that has roughly the same error scatter in both the x and y axes at the 45 degree mark. As the emitter position moves to the 90 degree mark, the x and y scatter becomes both circular in shape and very small. The orientation of the error ellipsoid/scatter roughly follows the angular position of the emitter.

Chapter 4 – Simulation Results

4.1 Results

In order to classify the performance on the test cases, a calculation of the standard deviation of the 200 Monte Carlo estimates from the emitter's actual position was calculated using the MATLAB command. These standard deviations from the actual (x, y, z) values are then plotted over a set of variables (angle/range, slope, altitude, etc.) and the behavior analyzed. In the percent improvement plots, we compare to the standard deviations obtained for the no-map data conditions, which simply assume that the z emitter coordinate is 0 for the initial estimate. The percent improvement is calculated by the following:

$$100 * \frac{\sigma_{without} - \sigma_{with}}{\sigma_{without}}$$

where "with" and "without" indicate whether using the map data or not, respectively.

The variables that could be altered in terms of platform dynamics were acceleration (g, a multiple of the acceleration due to gravity, 9.8 meters/second^2), velocity (meters/second), frequency (f_o, GHz), and time duration (T, seconds, measurements taken at 1 second intervals). The platform maneuvered in a sinusoidal weave, and its positions (Xp, Yp, Zp) and velocities (Vx, Vy, Vz) over time were recorded.

Of most concern in emitter location is being able to locate an emitter that is situated near the zero-degree mark, because that is the most common tactical scenario. Figures 8-10 show varying emitter positions between 25 and 75 degrees (run at 10-degree intervals). As we expect, for increasing angle, the standard deviation decreases for both the x and y coordinates. When we use the map data, the percent improvement for the x coordinate decreases to a minimum at around 55 degrees and then increases again to ninety degrees. The percent improvement for y is greatest at low angles and decreases as the emitter position is moved to the ninety-degree mark. Also as expected, because we have the map elevation data, the percent improvement in z is always around the one hundred percent mark. Between Figures 9 and 10, where we have changed the sign of the slope, we really don't see much drastic differences for these small slope values. In Figures 11 and 12, we simulate over varying values of slope, between -0.3 and 0.3 at intervals of 0.1. We don't see any indication here that slope improvements depend on the value of the slope itself. However the orientation of the slope, whether dz/dx or dz/dy, affects the percent improvement: there is small improvement in performance for the x parameter, but the y parameter benefits the most from having the slope data. The change in standard deviations for the map cases is very little for both x and y, and is greatest for z, whose standard deviations do happen to depend upon the slope values.

Figures 13-17 simply reiterate some of the statements made earlier, where increasing altitude provides greater percent improvements in x for smaller angle values for emitter positions. We also can see in Figures 13 and 14 that for a reasonably high altitude (55kft) the y standard deviation increases, but with bound, for the with-map case. The difference in slope has little to no effect in performance. Figures 15-17 vary the platform altitude with cases at 5kft, 30kft and 70kft. It becomes evident that the improvement gets markedly better for low angles once the platform altitude increases to 10kft, and these low angle values correspond to the emitter locations where there is the most difficulty in obtaining accurate estimates.

The platform flight parameters (g, T, altitude) are varied in the next few figures. In Figure 18, varying the value of g from 1 to 3 with steps of 0.5 seems to indicate that there is the smallest standard deviation for small g value. The greatest percent improvements are shown for the y-axis, so big erratic flight patterns aren't necessary for good map-augmented performance. Figure 19 shows a run over varying altitude from 10 to 60kft with steps of 10kft shows that by using the map data, there is always a decrease in the error standard deviation as the altitude increases, unlike the case for where there is no map data used. The percentage improvements are fairly constant for both x and y, though there is a slight increase in improvement as the altitude gets large. Increasing the duration for how long data is collected (25 to 60 seconds in steps of 5 seconds) shows in Figure 20 behavior as we expect; that increasing the number of data samples, we can obtain a better estimate for the emitter position. Using the map data confirms what we expect, with the standard deviations decreasing for increasing time in a very consistent manner. Interestingly, there is a greater improvement in the y performance when using map data. Figure 21 shows plots varying range and Figure 22 plots versus operating frequency. Increasing range (varying between 36 to 50km with increments of 2km) increases the standard deviation for both with and without map cases. However there is greater improvement in the y dimension than the x, and this trend is also evidenced in varying the operating frequency from 1 to 10 GHz. We expect and see increased precision for greater operating frequencies, and for the map case, a larger improvement for the y dimension accuracy. Table 1 shows a summary of results using map data.

	X Improvement	Y improvement	Z Improvement
Angle	Altitude Dependant: low altitudes give greater improvement for higher angle values and higher altitudes give better improvement for low angle values	Large improvements for lower angles, decreases for higher angle values	Nearly 100 percent improvement
Range	Fairly constant improvement over no-map case over increasing range values	Fairly constant improvement over no-map case over increasing range values	Same as above
Frequency	Constant improvement over 1-10 GHz range	Greater improvement than for x improvement	Same as above
G	Fairly constant	Greater improvement than for x	Same as above
Collection Time	Varies slightly	Varies, but greater than for x	Same as above
Altitude	Fairly constant	Fairly constant	Same as above
Slope	Fairly constant	Greater improvement than for x, usually around 50 % over no map case	Same as above

Table 1 - Summary of Results with Map Data

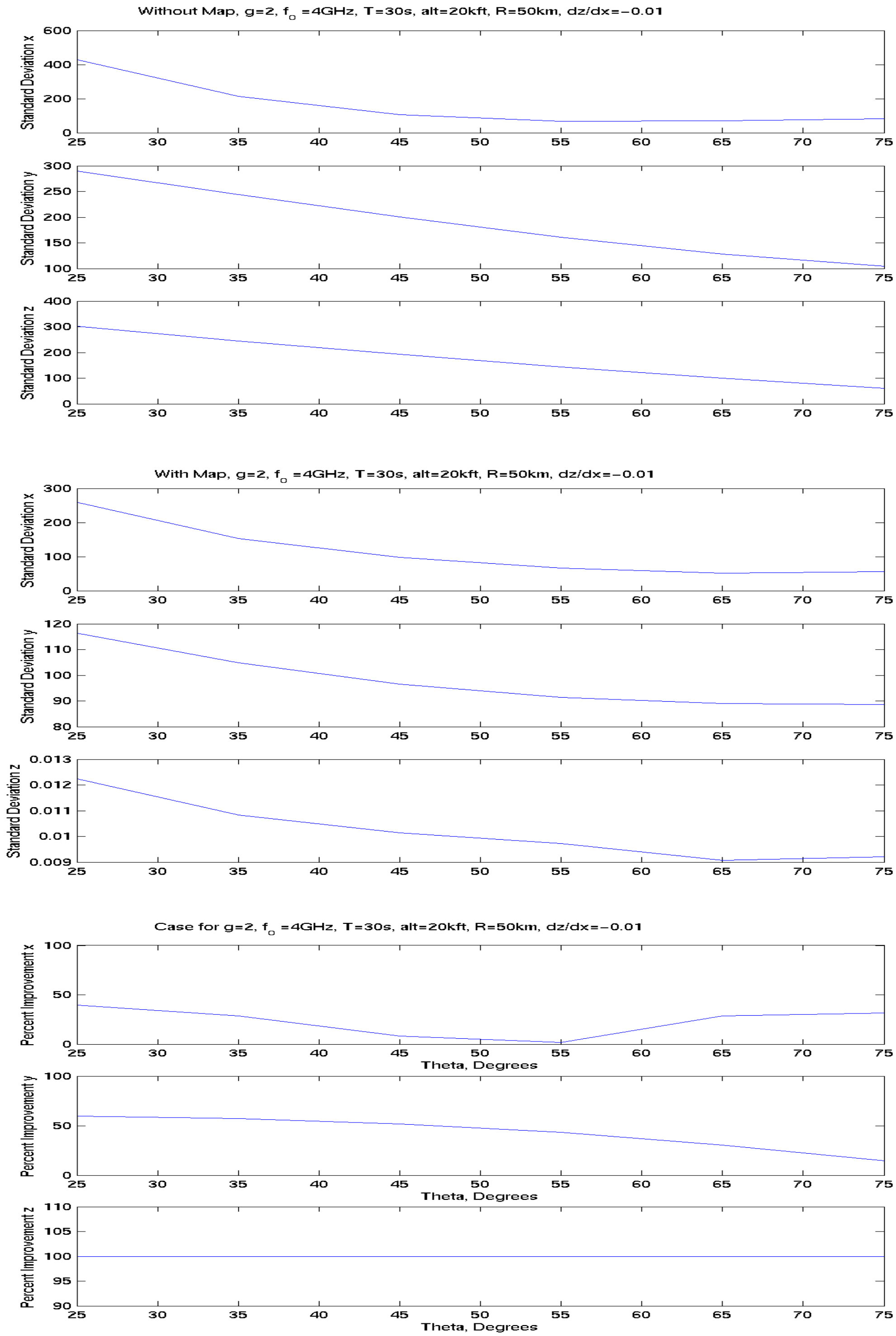

Figure 8 - Standard Deviations and Percent Improvement for moderate altitude, negative slope

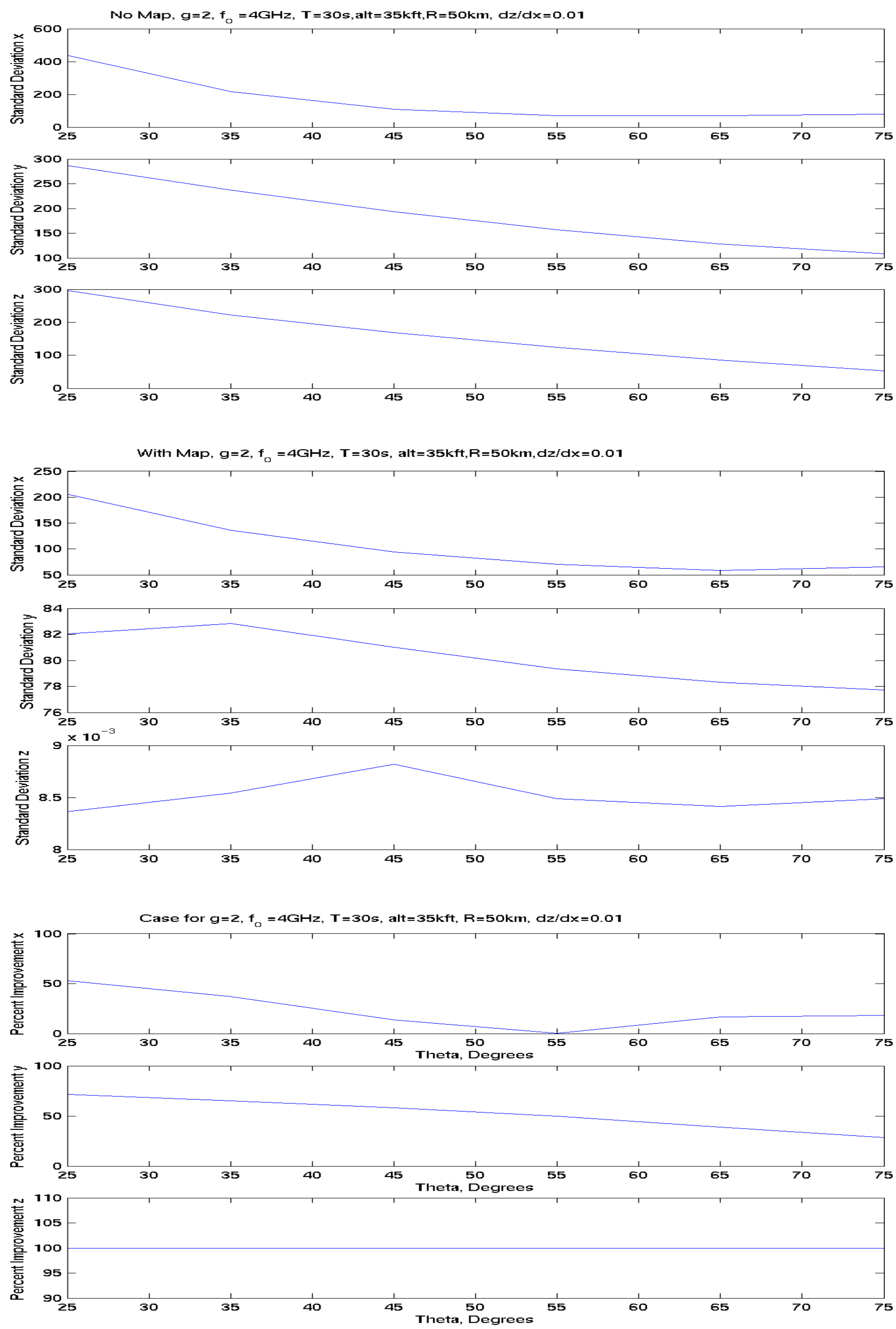

Figure 9 - Standard Deviation, Percent Improvement for Increased Platform Altitude, Different Slope

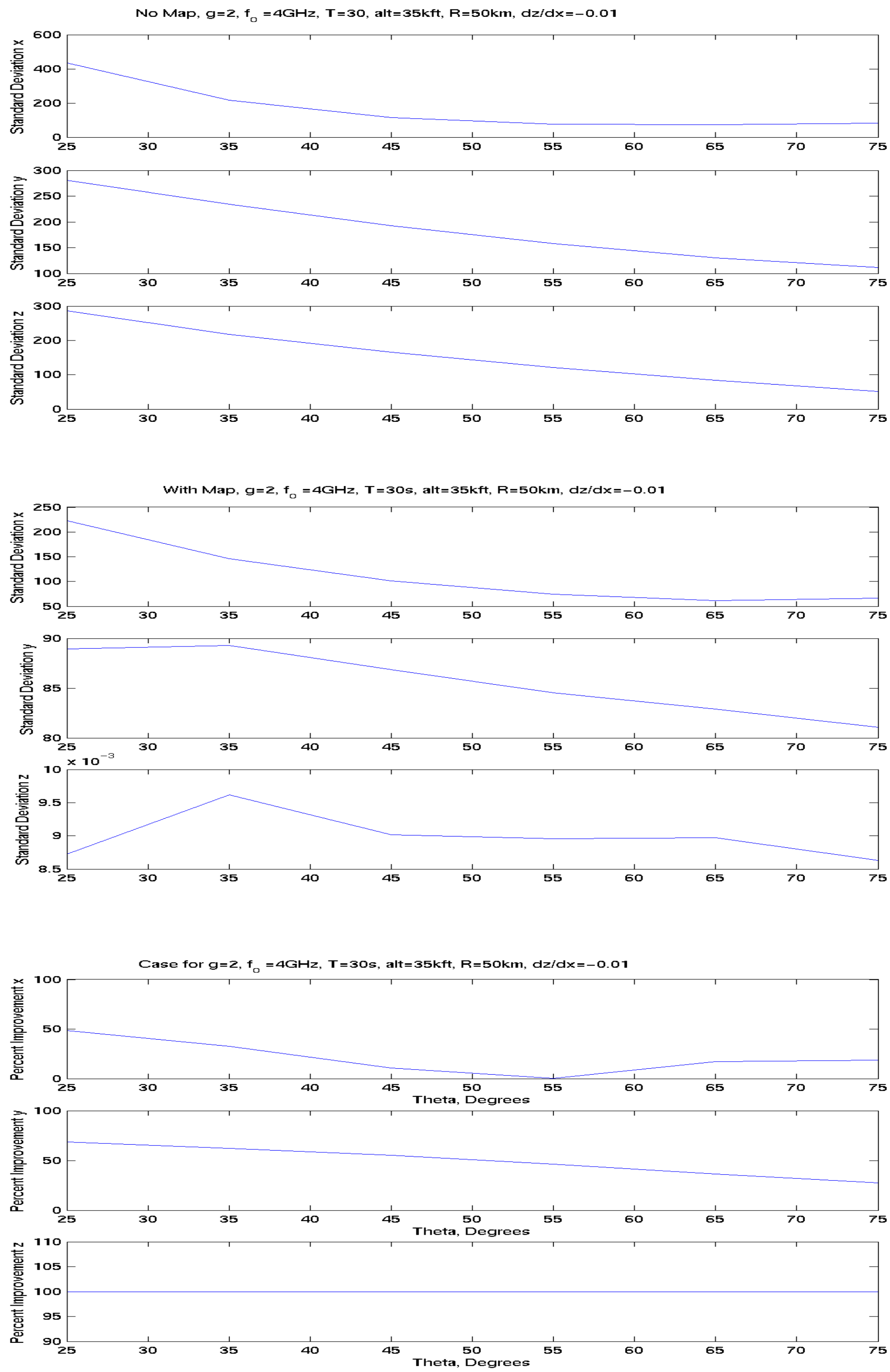

Figure 10 - Same as previous but with negative slope

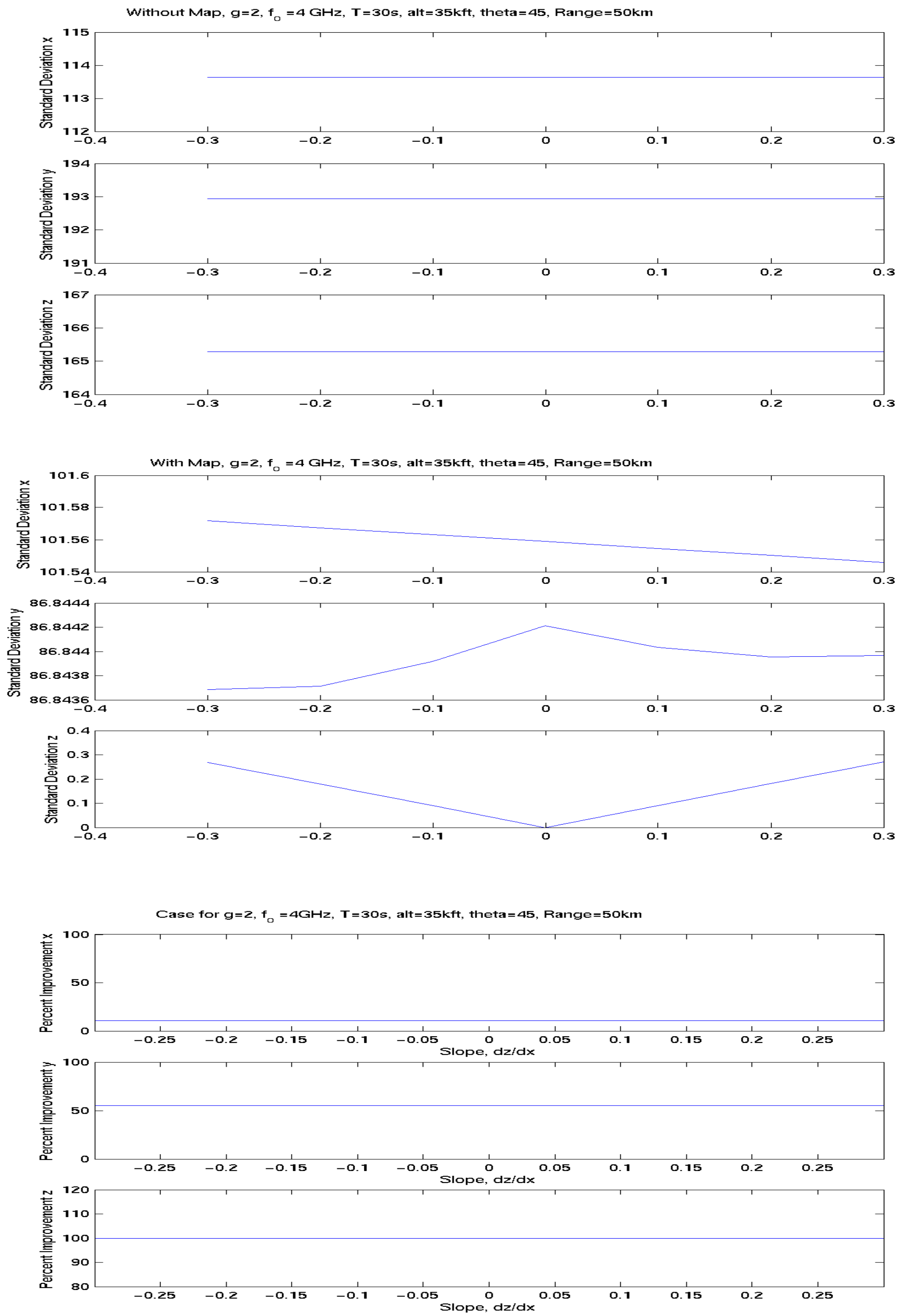

Figure 11 - Standard Deviations, Percent Improvement for varying slopes (dz/dx)

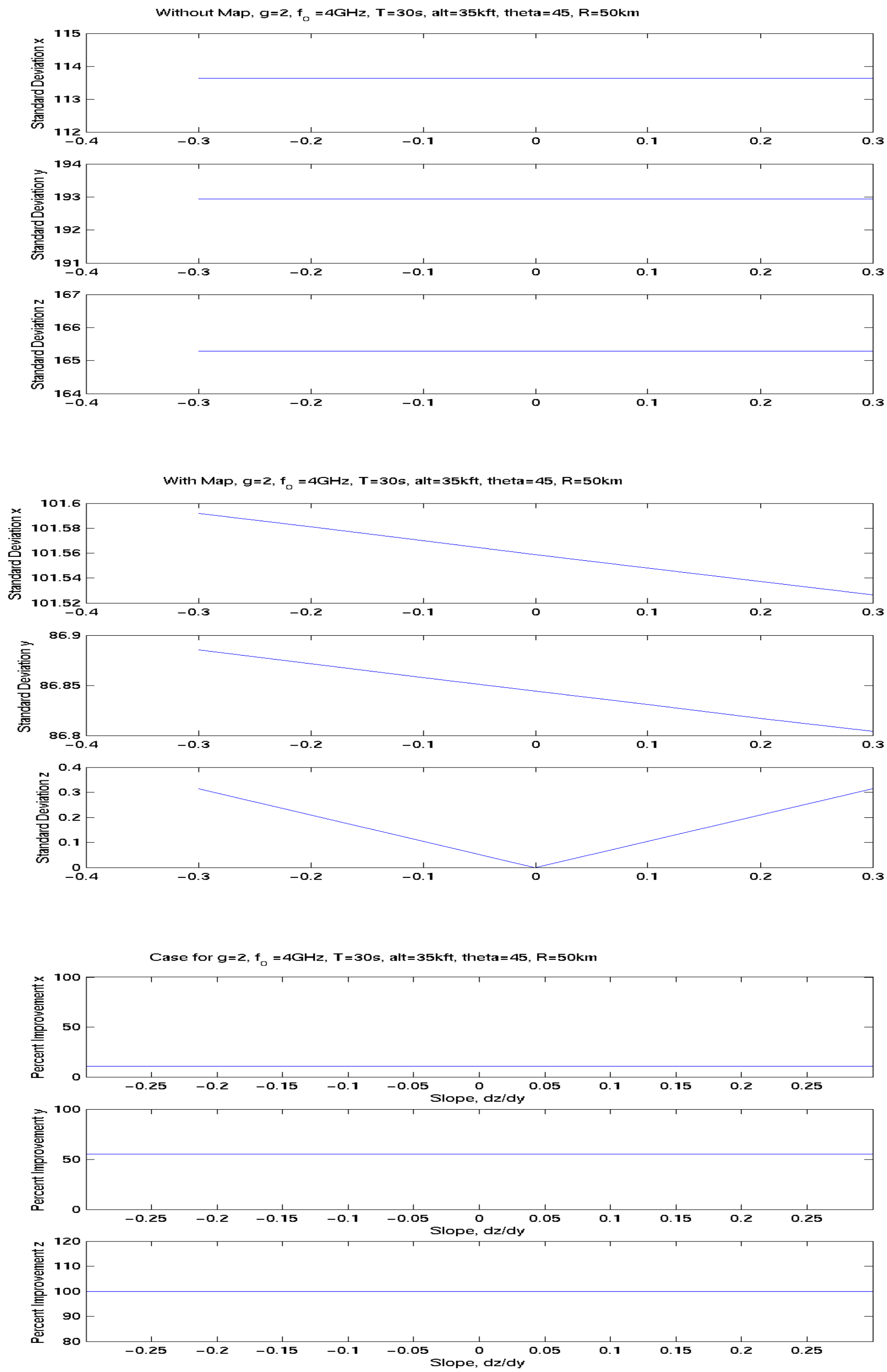

Figure 12 - Standard Deviations and Percent Improvement for varying slopes (dz/dy)

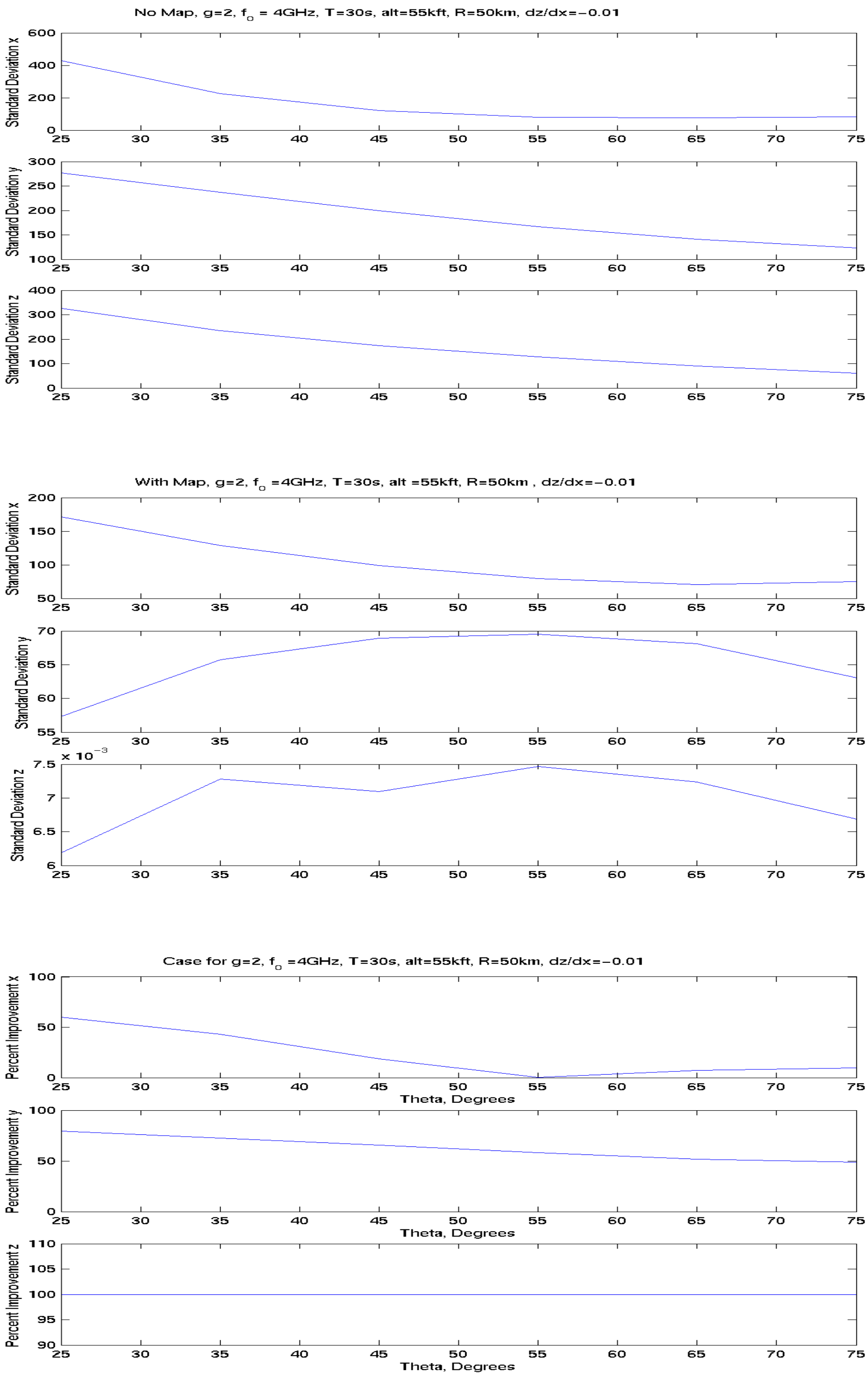

Figure 13 - Standard Deviations, Percent Improvements for high platform altitude (55kft)

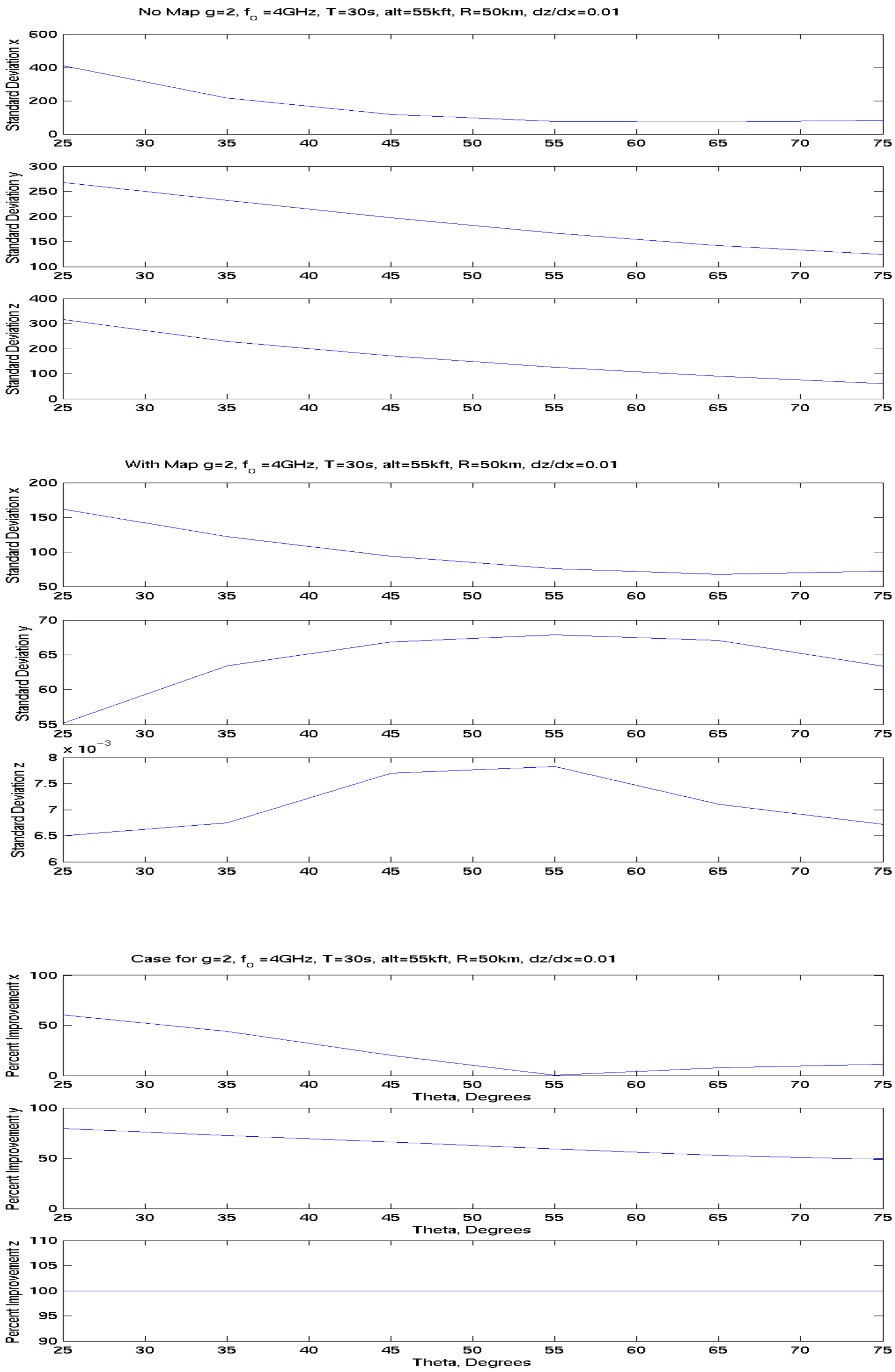

Figure 14- Same Parameters as previous figure, but with positive slope

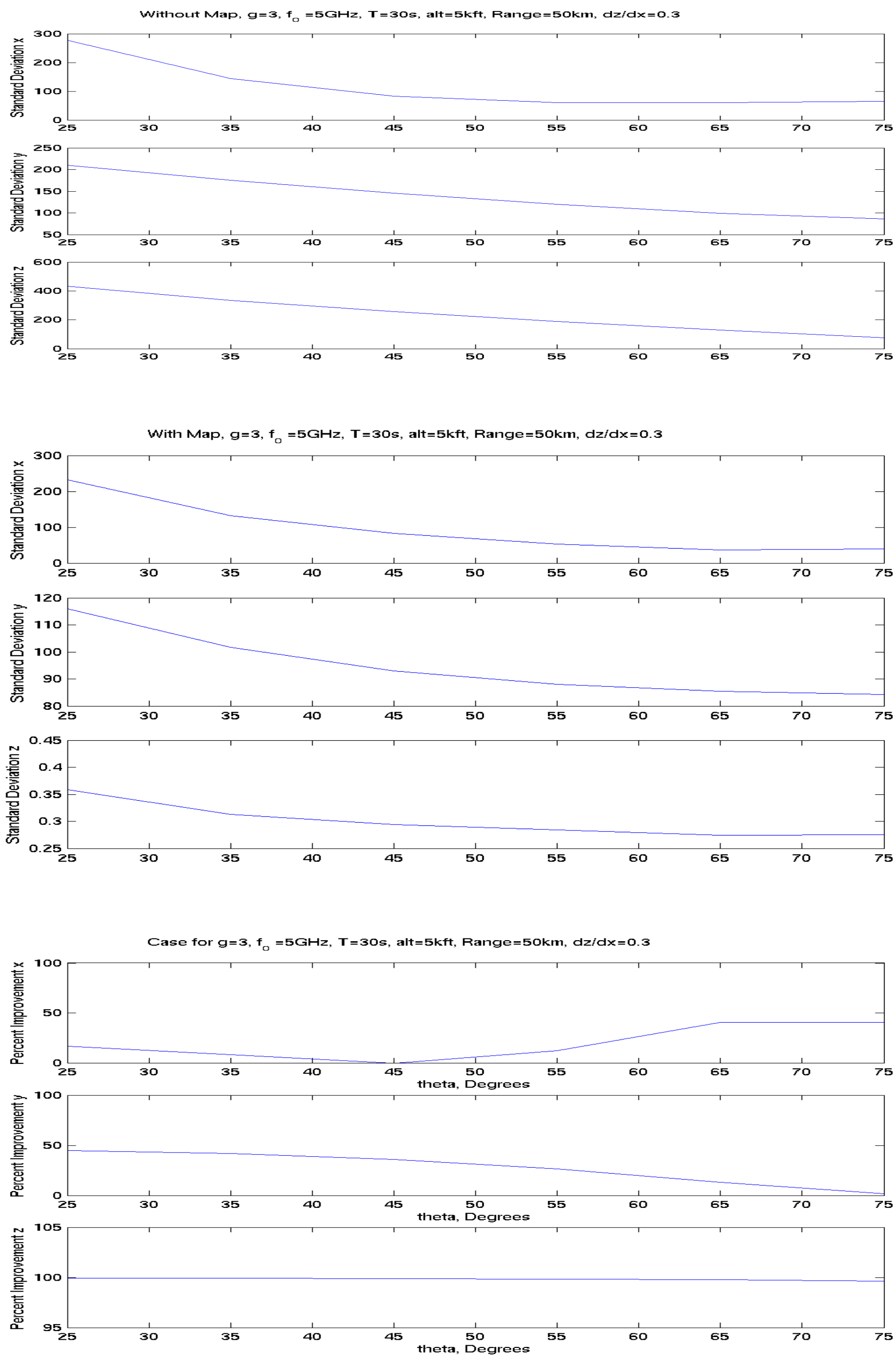

Figure 15 - Varying Emitter position, platform altitude

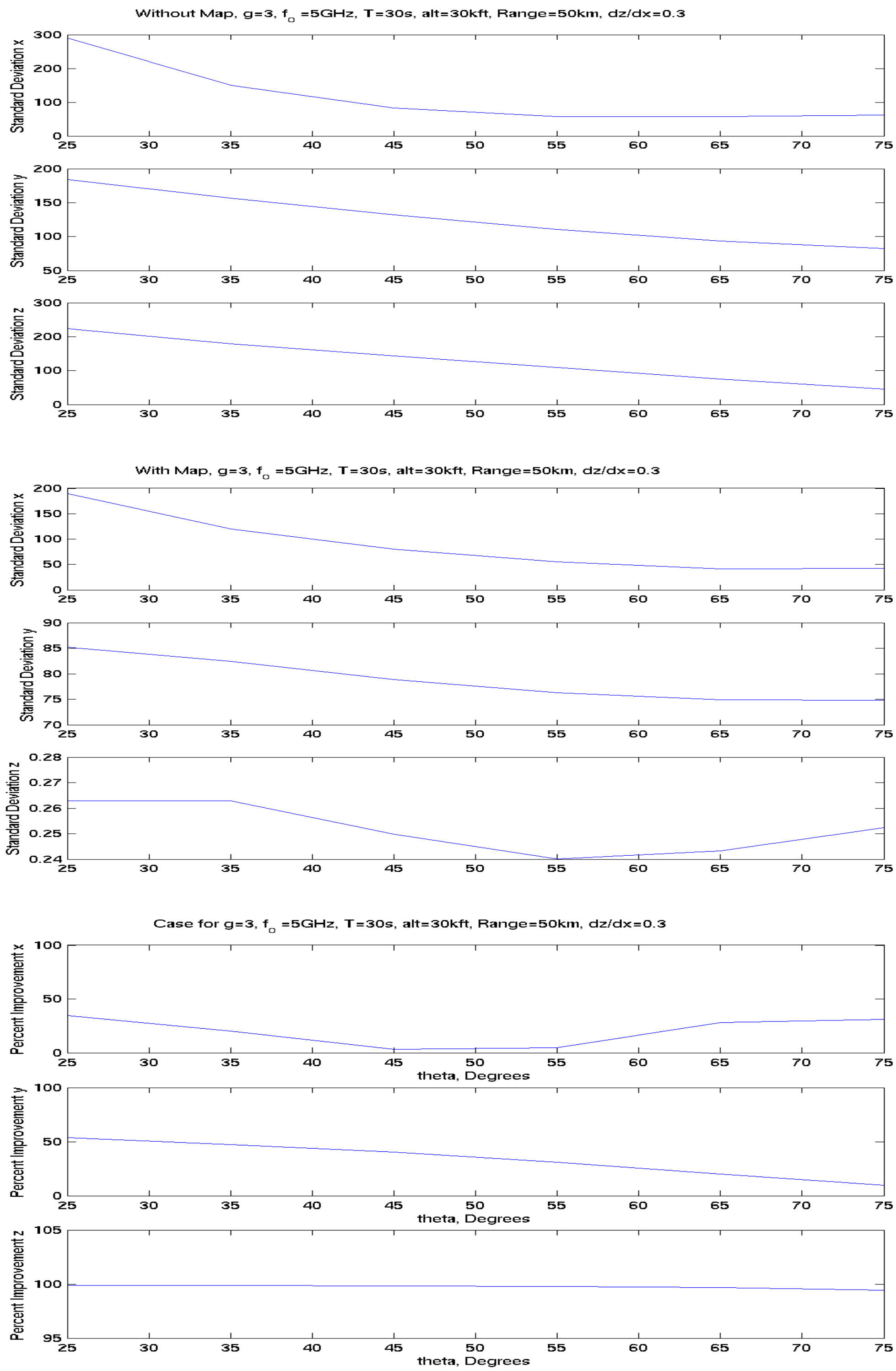

Figure 16 - Increasing Altitude

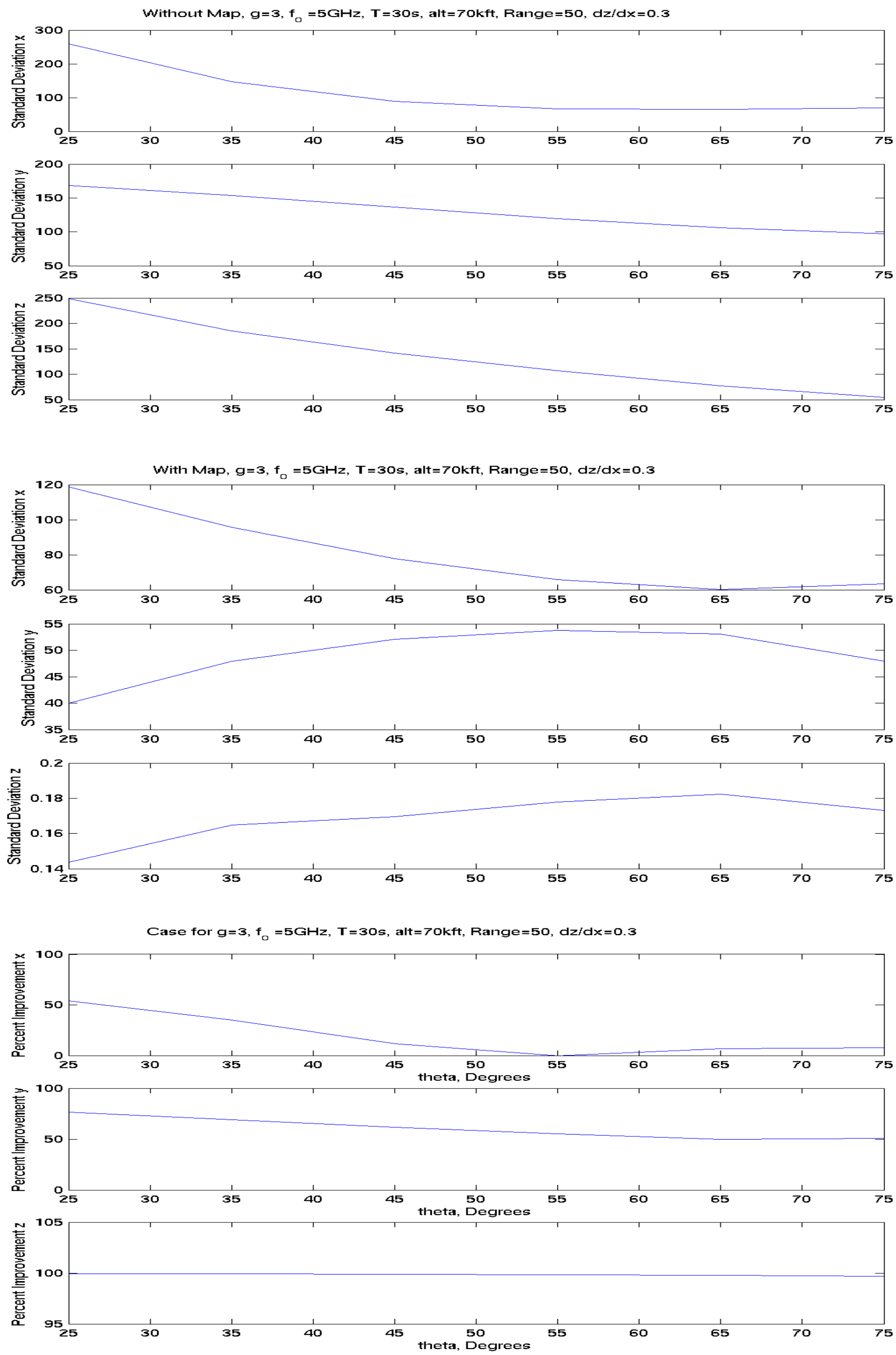

Figure 17 - Increasing Altitude

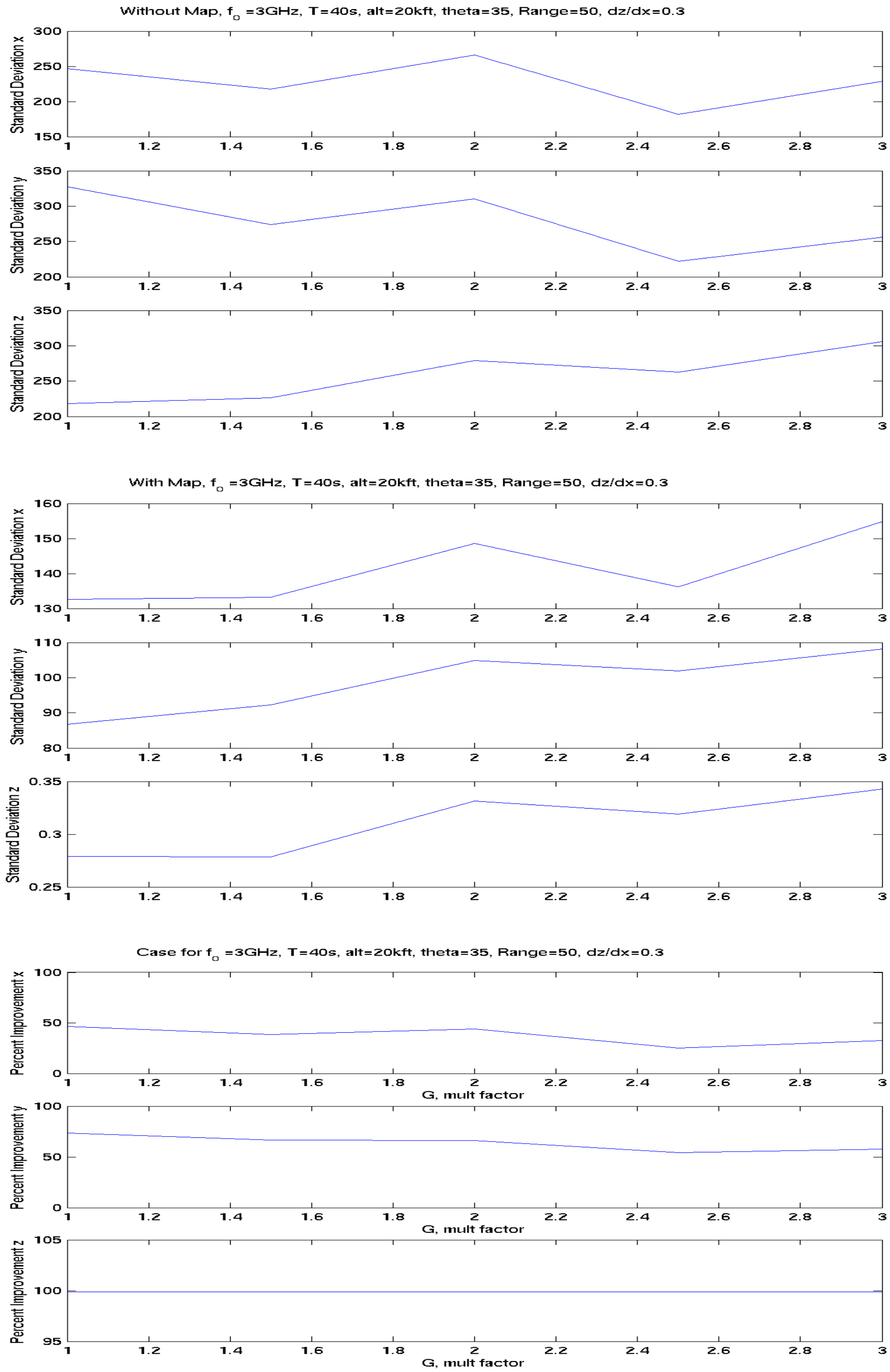

Figure 18 - Changing g value

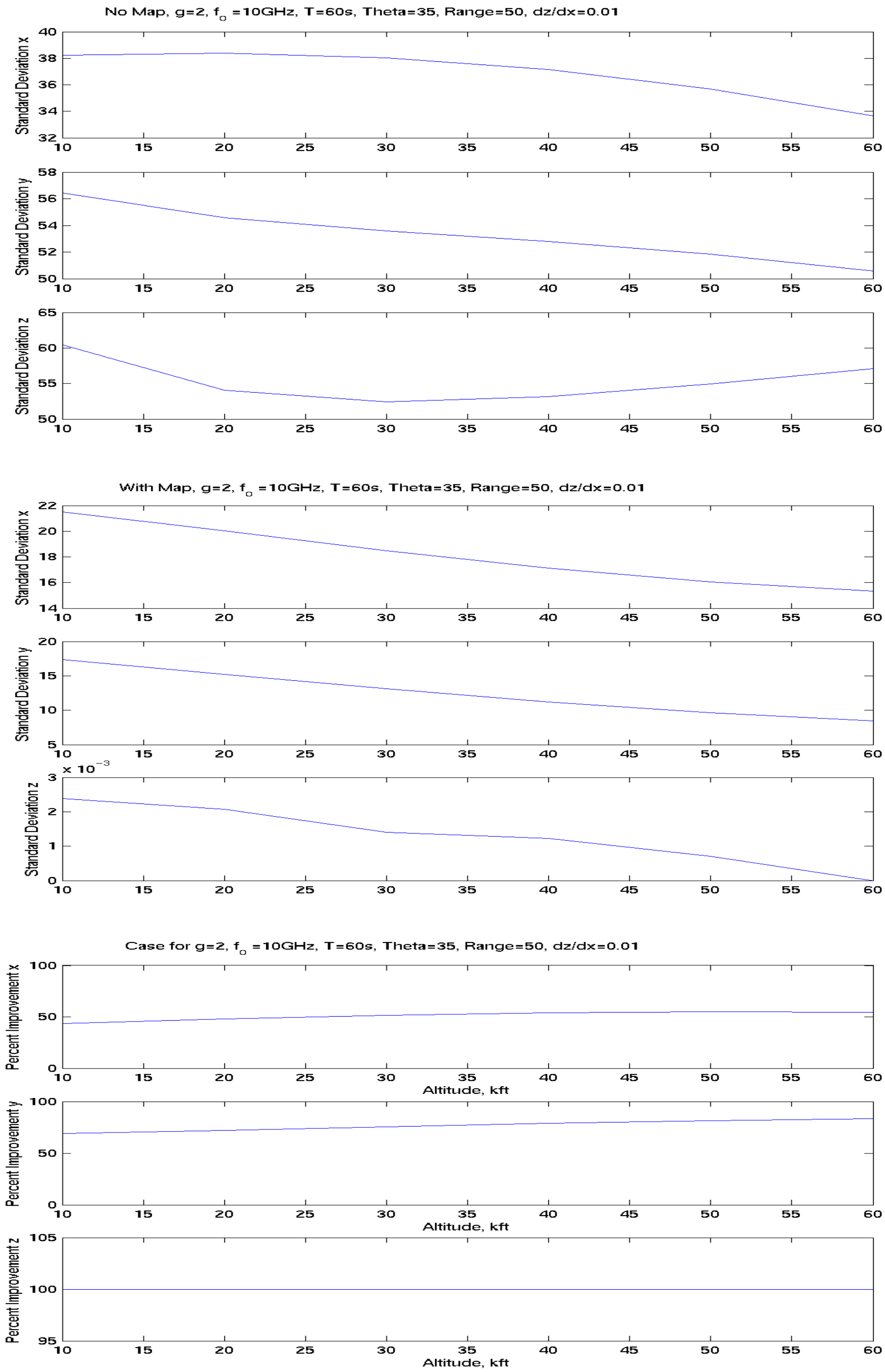

Figure 19 - Changing Platform Altitude

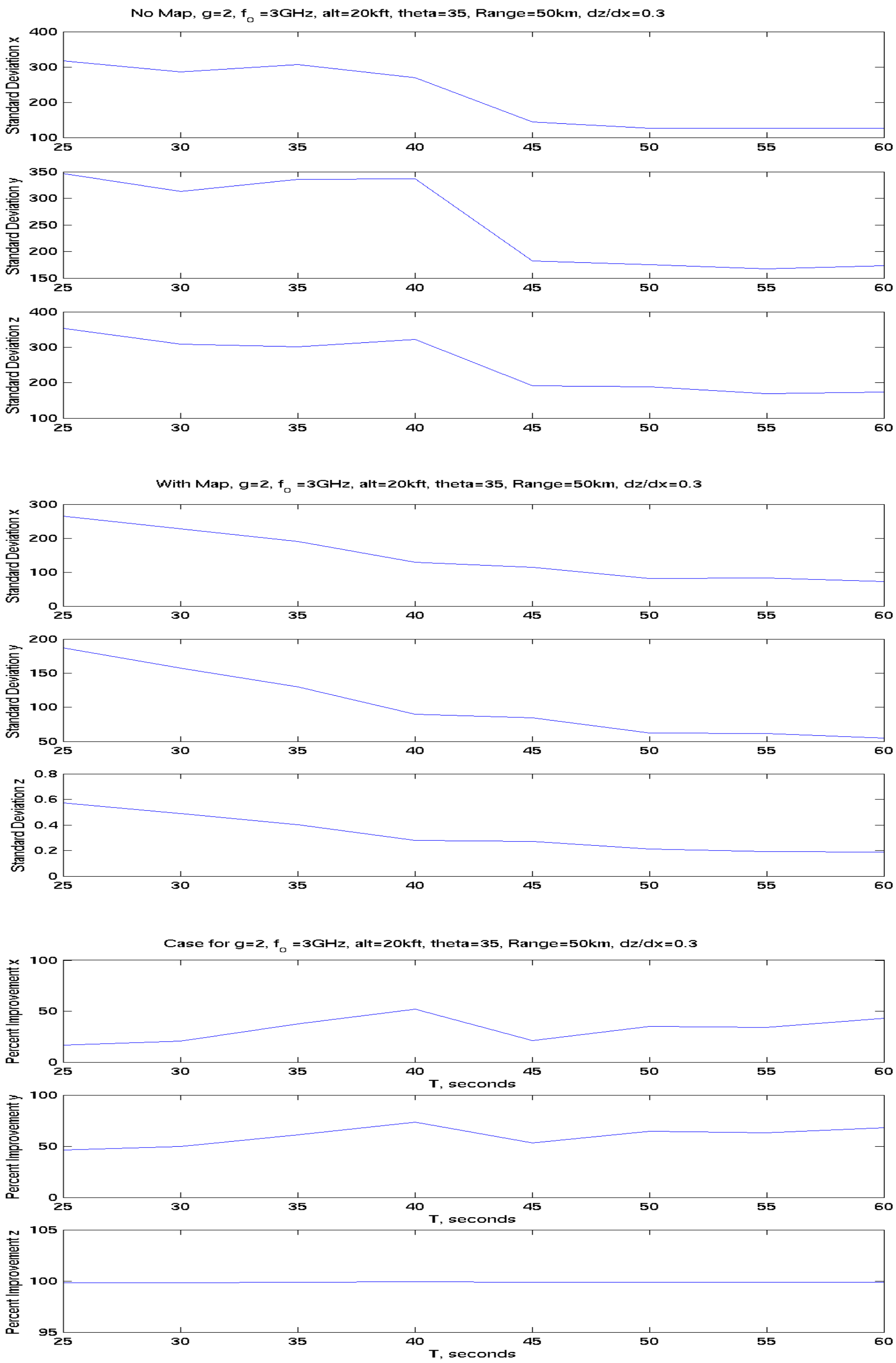

Figure 20 - Varying Time Duration (25:5:60) seconds

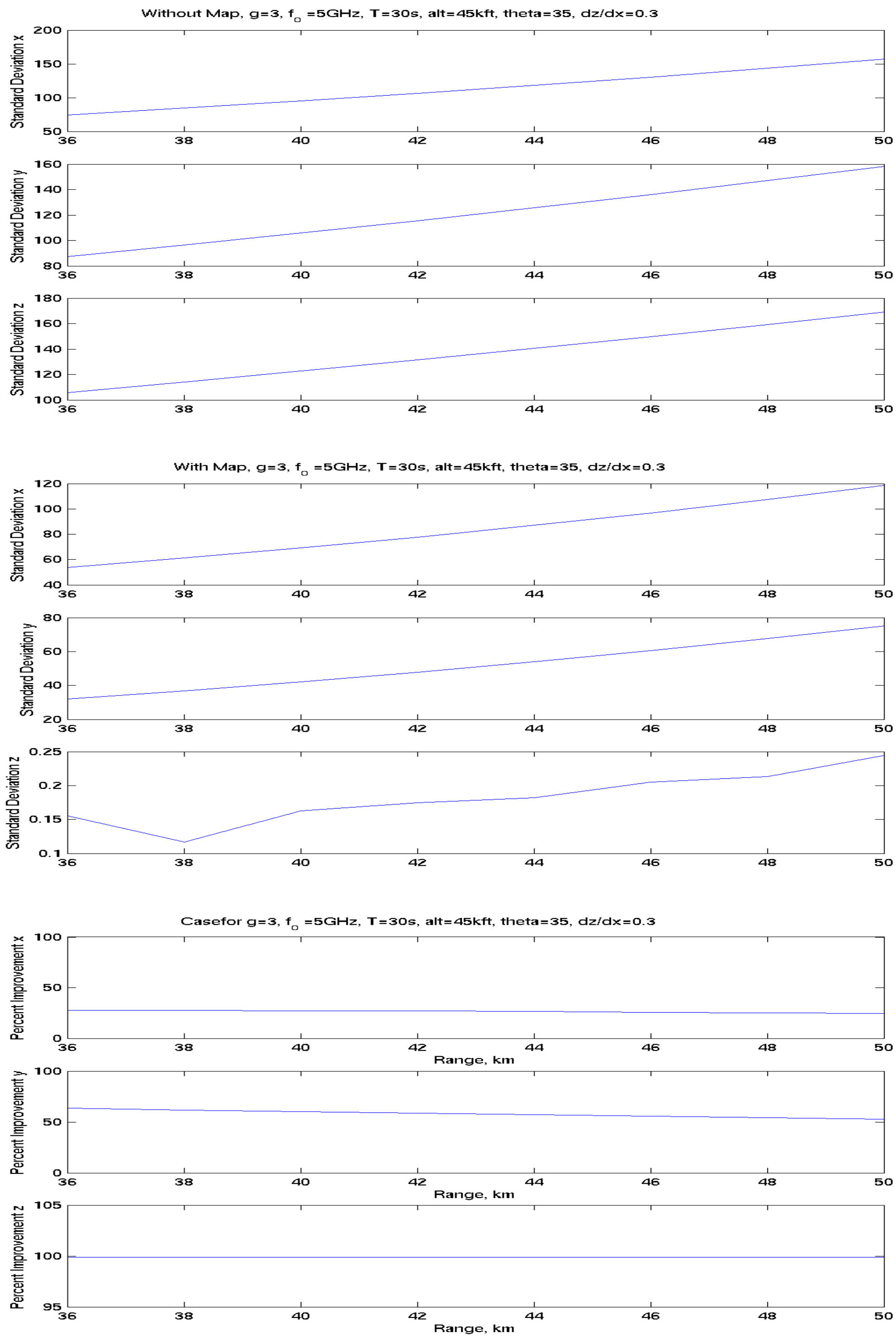

Figure 21 - Varying Range (36:2:50)km

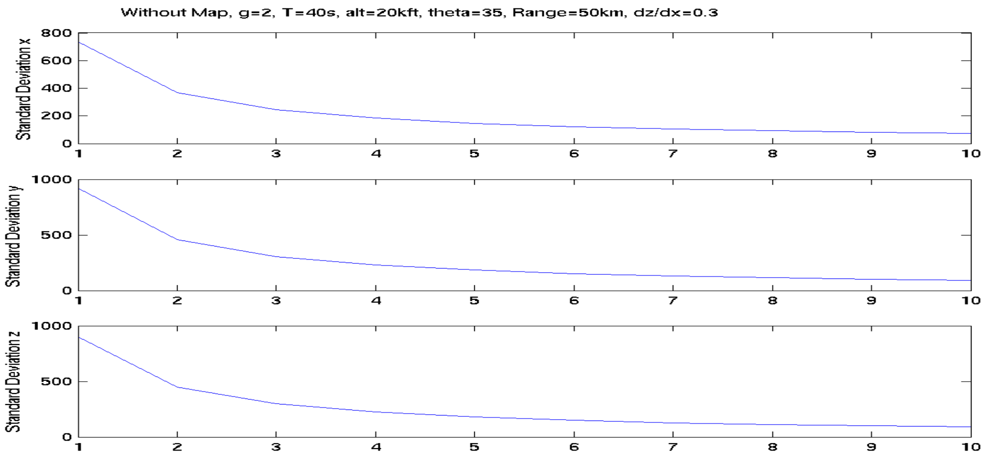

Figure 22 - Varying fo (1:1:10)GHz

4.2 Conclusions

For all combinations of altitudes, operation frequencies, and platform parameters using a weave motion, we see that there is a marked improvement using map data over the case where map data is not used. Although for some configurations, the improvement behavior is slight (higher altitudes), there are no situations where the improvement is lower for either the x or y coordinates. The worst performance occurs on the x-axis, when there is little improvement over the no-map case when the emitter is located roughly at a 45 to 55 degree angle out from the platform. There is no set altitude to fly the platform, where one altitude will get remarkably worse or better results than another: the platform can be flown at commercial airline altitudes or even higher, a necessity for locating anti-aircraft missiles located on the ground during reconnaissance missions. However, higher platform altitudes do provide greater percent improvements for emitter positions that are located at angle positions around the 0 degree mark. Performance also does not depend on the value of the terrain slope itself. Improvement tends to be constant both for the x and y coordinates though there is greater improvement in y.

Areas to explore for further study include a "true" interpolation for the map data. Available map resolution is not much better than the 100 m spacing in x and y that was used in this analysis, but terrain may have sharp discontinuities in certain regions, and it may be beneficial to obtain a better guess for a map elevation than to simply select the closest value available on the map. Different platform trajectories can also be looked at, depending on what types of “real” flight patterns are performed.

References

1. Becker, K., “An Efficient Method of Passive Emitter Location,” *IEEE Transactions on Aerospace and Electronic Systems*, AES-28 (Oct. 1992), 1091-1104.

2. Cadzow, J. A., “Least Squares, Modeling, and Signal Processing,” *Digital Signal Processing* **4**, (1994) 2-20.

3. Collins, N. and Baird, C., “Terrain Aided Passive Estimation,” *IEEE Proceedings of the National Aerospace and Electronics Conference*, v2. May 22-26 1989, 909-916.

4. Fowler, M., “Analysis of Passive Emitter Location with Terrain Data,” *IEEE Transactions on Aerospace and Electronic Systems*, AES-37 (April 2001), 495-507.

5. Foy, W. H., “Position-Location Solutions by Taylor-Series Estimation,” *IEEE Transactions on Aerospace and Electronic Systems*, AES-12 (March 1976), 187-193.

6. Paradowski, L. R., “Microwave Emitter Position Location: Present and Future,” *12th International Conference on Microwaves and Radar*, MIKON-98, vol.4. 20-22 May 1998; Krakow, Poland.

7. Spingarn, K., “Passive Position Location Estimation Using the Extended Kalman Filter,” *IEEE Transactions on Aerospace and Electronic Systems*, AES-23 (July 1987), 558-567.

8. Rao, K.D, and Reddy, D.C., “A New Method for Finding Electromagnetic Emitter Location,” *IEEE Transactions on Aerospace and Electronics Systems*, AES-30 (October 1994), 1081-1085.

9. Press, W., Teukolsky, S., Vetterling, W., and Flannery, B., *Numerical Recipes in C*, NY, NY: Cambridge University Press, 1992.

10. http://www.npac.syr.edu/projects/terrain/data/overview.html

11. Fowler, M., "Doppler Location Via Nonlinear Least Squares,” unpublished notes.

Appendix: MATLAB Code Used

File: run_dopp_sims.m

```
function [S,NAV,xe,ye,ze]=run_dopp_sims(g,T,alt_kft,motion,fo,xe,ye,ze)
 %
 % USAGE:
[S,NAV,xe,ye,ze]=run_dopp_sims(g,T,alt_kft,motion,fo,xe,ye,ze);
 %
 % Inputs: g = g-level of horizontal acceleration
 %         T = number of time instants (note: 1 sample/second)
 %         alt_kft = platform altitude in thousands of feet
 %         motion = string defining motion model: 'weave' or 'turn'
 %         fo = emitter frequency in GHz
 %         xe = emitter x-location
 %         ye = emitter y-location
 %         ze = emitter z-location
 %
 % Output: S = matrix of Monte Carlo Run Results
 %         Each column is an estimate of the emitter;
 %           The 1st row gives the fo estimates
 %           The 2nd row gives the xe estimates
 %           The 3rd row gives the ye estimates
 %           The 4th row gives the ze estimates
 %         NAV = platform info (see dopp_sim for details)
 %         xe, ye, ze = emitter location
 global current_stdx current_stdy current_stdz

  N_Loc_iter=20; % Set the # of times to run the iteration loop for each
estimate
 N_runs=200;  % Set the # of Monte Carlo runs

 % Allocate space for some matrices
 S_est_dopp=zeros(4,N_runs);
 S_est_lbi =zeros(4,N_runs);
 for k=1:N_runs   % loop over the Monte Carlo runs

   % Compute inital estimates by randomly perturbing true values
   fo_o = fo + 10e-3*(2*(rand(1)-0.5));  % uniform perturbation in [-10
MHz, +10 MHz]
   xe_o = xe + 10e3*(2*(rand(1)-0.5));  % uniform perturbation in [-10
km, +10 km]
   ye_o = ye + 10e3*(2*(rand(1)-0.5));  % uniform perturbation in [-10
km, +10 km]
   ze_o = ze + 500*(2*(rand(1)-0.5));  % uniform perturbation in [-500
m, +500 m

   % Run Doppler Location Algorithm

[S_dopp,J,DEL_X,xe,ye,NAV]=dopp_sim(g,T,alt_kft,motion,xe,ye,ze,fo,xe_o,
ye_o,ze_o,fo_o,N_Loc_iter);
   S(:,k)=S_dopp(:,end);
 end   % end loop over Monte Carlo Runs

%figure
```

```
%subplot(2,1,1)
%plot(NAV(2,:),NAV(3,:),'b',S(2,:),S(3,:),'bo',xe,ye,'rx');
title('platform, emitter, estimates'); axis equal
%subplot(2,1,2);plot(S(2,:)-xe,S(3,:)-ye,'bo');title('200 estimation
errors');axis equal
%figure;plot3(S(2,:)-xe, S(3,:)-ye,S(4,:)-ze,'o');grid;title('3d
scatter');axis equal

current_stdx=sqrt(var(S(2,:)-xe));
current_stdy=sqrt(var(S(3,:)-ye));
current_stdz=sqrt(var(S(4,:)-ze));

sprintf('Standard Deviation of xe is %0.5g\n', current_stdx)
sprintf('Standard Deviation of ye is %0.5g\n', current_stdy)
sprintf('Standard Deviation of ze is %0.5g\n', current_stdz)
```

File: dopp_sim.m

```
function
[S,J,DEL_X,xe,ye,NAV]=dopp_sim(g,T,alt_kft,motion,xe,ye,ze,fo,xe_o,ye_o,
ze_o,fo_o,N)

global x_left y_bottom map_yes_no x_res y_res xcell ycell y2 tt RadC Z t


 % Usage:
[S,J,DEL_X,xe,ye,NAV]=dopp_sim(g,T,alt_kft,motion,xe,ye,ze,fo,xe_o,ye_o,
ze_o,fo_o,N);
 %
 %
 % Inputs: g = g-level of horizontal acceleration
 %         T = number of time instants (note: 1 sample/second)
 %         alt_kft = platform altitude in thousands of feet
 %         Range = range to emitter from origin in km
 %         fo = emitter frequency in GHz
 %         motion = string defining motion model: 'weave' or 'turn'
 %         theta = angle (in degrees) of emitter location
 %         xe_o = initial guess of emitter's x location (scalar)
 %         ye_o = initial guess of emitter's y location (scalar)
 %         ze_o = initial guess of emitter's z location (scalar)
 %         fo_o = initial guess of emitter's frequency in GHz (scalar)
 %         t  =  vector of time instants  (row vector)
 %         N  =  maximum number of iterations to perform
 %
 %
 % Outputs: S = state vector matrix; each column is the iteratively
computed
 %             state vector, where state vector = [fo xe ye ze]
 %
 %          J = computed mean square error at each iteration
 %
 %          DEL_X = a matrix whose ith column is the state update vector
at ith iteration
 %
 %          xe, ye = true emitter location
 %
 %          NAV = platform navigation data in rows of matrix NAV
```

```
 %                  NAV(1,:) = time instants
 %                  NAV(2,:) = Xp: platform X positions
 %                  NAV(3,:) = Yp: platform Y positions
 %                  NAV(4,:) = Zp: platform Z positions
 %                  NAV(5,:) = Vx: platform X velocity
 %                  NAV(6,:) = Vy: platform Y velocity
 %                  NAV(7,:) = Vz: platform Z velocity


 var_1 = 1.^2;  %% 1 Hz RMS; converted to variance in Hz^2
 fo=fo*1e9;
 fo_o=fo_o*1e9;
%% Decide if platform motion should be "weave" or "turn"
 if strcmp(motion,'weave')
     [Xp,Yp,Zp,Vx,Vy,Vz,long_vect,trans_vect]=motionw(g,T,alt_kft);
 elseif strcmp(motion,'turn')
     [Xp,Yp,Zp,Vx,Vy,Vz,long_vect,trans_vect]=motionca(g,T,alt_kft);
 elseif strcmp(motion, 'circle')
     [Xp,Yp,Zp,Vx,Vy,Vz] = Circle_Path(tt,RadC,Z);
 else
     error('Specified motion string is invalid')
 end

%t=0:T;

 NAV=[t;Xp;Yp;Zp;Vx;Vy;Vz];

   % state vector = [fo xe ye ze]
 S=zeros(4,N+1);
 S(:,1)=[fo_o xe_o ye_o ze_o].';

 c=2.998e8;  % speed of light

 % Generate Doppler Measurements

 R=sqrt( (Xp-xe).^2  +  (Yp-ye).^2  +  (Zp-ze).^2);
 z_nf=fo - (fo/c)*(Vx.*(Xp-xe) + Vy.*(Yp-ye) + Vz.*(Zp-ze))./R;
 z=z_nf+sqrt(var_1)*randn(size(z_nf));

 %%%%%%%%%%%%%%%%%%%%%%%%


 fo_tilde=fo_o;
 xe_tilde=xe_o;
 ye_tilde=ye_o;
 ze_tilde=ze_o;


 J=zeros(1,N);

 for n=1:N
 % Use Current states to compute corresponding "measurements" and
differential

 R_tilde=sqrt( (Xp-xe_tilde).^2  +  (Yp-ye_tilde).^2 +  (Zp-
ze_tilde).^2);
 z_tilde=fo_tilde - (fo_tilde/c)*(Vx.*(Xp-xe_tilde) + Vy.*(Yp-ye_tilde)
+ Vz.*(Zp-ze_tilde))./R_tilde;

 J(n)=sum( (z-z_tilde).^2)/length(t);
```

```
 del_z= z - z_tilde;


 % Compute the H entries:

H=jaco_dopp_sim(Xp,Yp,Zp,Vx,Vy,Vz,xe_tilde,ye_tilde,ze_tilde,fo_tilde/1e
9);


 % Define Covariance Matrix
 R_mat=diag(var_1*ones(1,length(t)));
 R_inv=inv(R_mat);


 % Solve for state update:

    % form matrix and vector needed for normal equations:
 A=(H.')*R_inv*H;
 b=(H.')*R_inv*(del_z.');

    % solve normal equations:
 del_x=inv(A)*b;       DEL_X(:,n)=del_x;

if map_yes_no ==0

 S(:,n+1) = S(:,n) + del_x;

else
S(1,n+1)=S(1,n)+del_x(1);
S(2,n+1)=S(2,n)+del_x(2);
S(3,n+1)=S(3,n)+del_x(3);
S(4,n+1)=S(4,n);          % no updating happening
end

 % "Re-Initialize" parameters for the next loop

 fo_tilde = S(1,n+1);
 xe_tilde = S(2,n+1);
 ye_tilde = S(3,n+1);
 ze_tilde = S(4,n+1);

%%%% Simple "pick the closest grid point" with ze available for
%%%% map data

if map_yes_no == 1            %%%% 1 = use map data
  %%%% Begin Section for MAP data script
  grid_x=round((xe_tilde-x_left)/x_res);
  grid_y=round((ye_tilde-y_bottom)/y_res);
  ze_tilde=y2(grid_x,grid_y);


else ze_tilde = S(4,n+1);     %%%% 0 = just perform the dopp estimates
end

S(4,n+1)=ze_tilde;  % either way (map or no map) store the current
estimate of ze
%%%% End Section for MAP

end   %%% for n = 1:N
```

File: motionw.m

```
function [Xp,Yp,Zp,Vx,Vy,Vz,long_vect,trans_vect]=motionw(g,T,alt_kft)
 %
 % USAGE: [Xp,Yp,Zp,Vx,Vy,Vz,long_vect,trans_vect]=motionw(g,T,alt_kft);
 %
 % Inputs: g = acceleration of platform  (g>0 curves down; g<0 curves
up)
 %         T = Final Time in Seconds (Initial Time = 0)
 %         alt_kft = platform altitude in thousands of feet
 %
 %
 % Outputs: Xp, Yp, Zp = XYZ positions of antenna #1 (in meters)
 %          Vx, Vy, Vz = velocities of platform (in m/s)
 %          long_vector = unit vector pointing along longitudinal
baseline
 %          trans_vector = unit vector pointing along transversal
baseline



 % \loc\motionw.m     Weave trajectory generator
2/24/95
 % Uses constant g turns to +/- angle of maxturn from nominal path
 % Uses constant flight path angle (no vertical acceleration)
 % Left/right turn transitions use linear turn rate change over 0 sec
period

 %%%%%  g = 2

 %%%%% tend = 20;
 tend=T;


 dt = 1; dtt=1/20;
 vel = 200; accelh = g*9.81; accelv=0.5*9.81;  alt0 =
alt_kft*1000*0.3048;

 dtr=pi/180;

 t = 0:dt:tend ;   % vector of times of measurements
 maxturn = 30*dtr ;
 vhoriz = vel ;   % assume cos(gamma) close to 1 & vert axis can be
decoupled
 radturn = (vhoriz^2)/accelh ;   % turn radius
 turnrate = vel/radturn;
 turndur = 2*maxturn/turnrate ;   % turn duration
 turndur = fix(turndur/dtt)*dtt ;
 gamma0 = -asin(min(accelv*turndur/vel,0.5))/2 ; %accelv & gamma0
assumed small
 dtt = 1/20 ;   % time increment for turn generation

 vx =  vel*cos(gamma0)*cos(maxturn) ;   % initialize velocities
 vy = -vel*cos(gamma0)*sin(maxturn) ;
```

```
 vz =  vel*sin(gamma0) ;

 XYZA(1,1) = 0 ;
 px = 0 ;  py = 0 ;  pz = alt0 ;
 itt=0 ;   % count of trajectory integration times
 im=0  ;   % count of measurement times

 n=0;
 for tt = 0 : dtt : tend
 n=n+1;
 trem = rem(tt,2*turndur) ;
 if trem < turndur
   turnangl = maxturn - turnrate*trem ;
   turn_angle(n)=turnangl;
   ahoriz = accelh ;
   az = accelv ;  %Note: vert & horiz decoupled; only good for small az
& gamma
 else
   turnangl = -maxturn + turnrate*(trem-turndur) ;
   turn_angle(n)=turnangl;
   ahoriz = -accelh ;
   az = -accelv ;
 end   % if trem
 vx =  vhoriz*cos(turnangl) ;
 vy = -vhoriz*sin(turnangl) ;
 vz = vz + az*dtt ;
 if itt > 0
   px = px + vx*dtt ;
   py = py + vy*dtt ;
   pz = pz + vz*dtt ;
 end   % if itt

 if rem(1e-7 + itt*dtt,dt) <1e-6   % if remainder=0 with tolerance
   im = im+1 ;
   XYZADOT(1,im) = vx ;
   XYZADOT(2,im) = vy ;
   XYZADOT(3,im) = vz ;
   XYZA(1,im) = px ;
   XYZA(2,im) = py ;
   XYZA(3,im) = pz ;
   ah(im) = ahoriz ;
 end   % if rem
 itt = itt + 1 ;
 end   % for tt
 NMEAS = im;

 hdg = pi/2 - atan2(XYZADOT(2,:),XYZADOT(1,:)) ;   % (1,n)
 roll = atan(-ah/9.81) ;   % approx. for small pitch
 pitch = atan(XYZADOT(3,:)./sqrt(XYZADOT(1,:).^2 + XYZADOT(2,:).^2)) ;
% (AOA=0
 XYZhpr=[XYZA;hdg;pitch;roll];

 Xp=XYZA(1,:);Yp=XYZA(2,:);Zp=XYZA(3,:);
 Vx=XYZADOT(1,:);Vy=XYZADOT(2,:);Vz=XYZADOT(3,:);

 long_vect=[sin(hdg);cos(hdg);sin(pitch)];
 vect_norms=sqrt(sum(long_vect.^2));    %%% Compute norms of vectors
 long_vect=long_vect./vect_norms(ones(1,3),:);  %%% Normalize the
vectors
```

```
 trans_vect=[cos(hdg);-sin(hdg);-sin(roll)];
 vect_norms=sqrt(sum(trans_vect.^2));    %%% Compute norms of vectors
 trans_vect=trans_vect./vect_norms(ones(1,3),:);  %%% Normalize the
vectors
```

File: motionca.m

```
function [Xp,Yp,Zp,Vx,Vy,Vz,long_vect,trans_vect]=motionca(g,T,alt_kft)
 %
 % USAGE:
[Xp,Yp,Zp,Vx,Vy,Vz,long_vect,trans_vect]=motionca(g,T,alt_kft);
 %
 % Inputs: g = acceleration of platform  (g>0 curves down; g<0 curves
up)
 %         T = Final Time in Seconds (Initial Time = 0)
 %         alt_kft = platform altitude in thousands of feet
 %
 % Outputs: Xp, Yp, Zp = XYZ positions of platform (in meters)
 %          Vx, Vy, Vz = velocities of platform (in m/s)
 %          long_vector = unit vector pointing along longitudinal
baseline
 %          trans_vector = unit vector pointing along transversal
baseline



 % \a2a\camotion.m      Constant acceleration motion            7/31/95
 % Fixed arc path in horizontal plane starting at 0,0 with velocity
along x axis % Vertical motion: fixed flight path angle

 %%%%  g=0.1
 %%%% tend = 20;
 tend=T;
 dt =1 ;

 vel = 250 ;  accelh = g*9.81 ;  accelv = 0 ;  gamma0 = 0 ;  alt0 =
alt_kft*1000*0.3048 ;

 t = 0 : dt : tend ;
 NMEAS = length(t) ;
 vhoriz = vel*cos(gamma0) ;   % assume thrust varied to maintain speed
 radturn = (vel^2)/accelh ;  % turn radius
 turnangle = t*vhoriz/radturn ;     % 1 by NMEAS
 XYZA(1,:) = radturn*sin(turnangle) ;    % XYZA is 3 by NMEAS
 XYZA(2,:) = radturn*(-1 + cos(turnangle)) ;
 XYZA(3,:) = alt0 + vel*sin(gamma0)*t + 0.5*accelv*t.^2 ;
 XYZADOT(1,:) =  vhoriz*cos(turnangle) ;
 XYZADOT(2,:) = -vhoriz*sin(turnangle) ;
 XYZADOT(3,:) = vel*sin(gamma0)*ones(1,NMEAS) + accelv*t;

 hdg = pi/2 - atan2(XYZADOT(2,:),XYZADOT(1,:)) ;
 roll = atan(accelh/9.81)*ones(1,NMEAS) ;
 pitch = zeros(1,NMEAS) ;
 XYZhpr=[XYZA;hdg;pitch;roll];

 Xp=XYZA(1,:);Yp=XYZA(2,:);Zp=XYZA(3,:);
```

```
 Vx=XYZADOT(1,:);Vy=XYZADOT(2,:);Vz=XYZADOT(3,:);

 long_vect=[sin(hdg);cos(hdg);sin(pitch)];
 vect_norms=sqrt(sum(long_vect.^2));    %%% Compute norms of vectors
 long_vect=long_vect./vect_norms(ones(1,3),:);  %%% Normalize the
vectors


 trans_vect=[cos(hdg);-sin(hdg);-sin(roll)];
 vect_norms=sqrt(sum(trans_vect.^2));    %%% Compute norms of vectors
 trans_vect=trans_vect./vect_norms(ones(1,3),:);  %%% Normalize the
vectors
```

File: see_scatter.m

```
global x_left y_bottom map_yes_no x_res y_res xcell ycell y2 t
current_stdx current_stdy current_stdz

S2 = randn('state');
randn('state', S2)
T2 = rand('state');
rand('state', T2);

x_left = 10000;               %%%% meters
y_bottom = 10000;

x_res = 100;                  %%%% x-dimension resolution, meters
y_res = 100;                  %%%% y-dimension resolution, meters

%%%% create spacing for xy map

for jj=1:1:601,
  xcell(jj) = x_left + jj*x_res;
  ycell(jj) = y_bottom + jj*y_res;
end

dz_dy=-0.3;
dz_dx=0;

% x = 1:601;
%  y2=zeros([601,601]);
%    for stu = 1:length(x),
%    y2(:,stu) = dz_dx*x(stu);       %%%% slope edits here
%  end

 x = 0:1:501;
  y2=zeros([501,501]);
    for stu = 1:length(x),
    y2(stu,:) = dz_dy*x(stu);       %%%% slope edits here
  end
%if dz_dy<0
%    y2=y2+400;
%end

g = 3;
fo = 5;
T = 30;
```

```
t = 0:T;
alt_kft=45;
theta=35;
motion = 'weave';
%Range=50;
[Xp,Yp,Zp,Vx,Vy,Vz,long_vect,trans_vect]=motionw(g,T,alt_kft);
%xe=Range*1000*cos(theta*pi/180);ye=Range*1000*sin(theta*pi/180);
%testing shift
%Yp=Yp+y_bottom+18000;

for map_yes_no=0:1,          %%  just look at using map data condition
counterp=1;
figure
for Range=36:2:50,

[Xp,Yp,Zp,Vx,Vy,Vz,long_vect,trans_vect]=motionw(g,T,alt_kft);
% x = 1:601;
%  y2=zeros([601,601]);
%    for stu = 1:length(x),
%    y2(:,stu) = dz_dx*x(stu);       %%%% slope edits here
%  end

% x = 0:1:501;
%  y2=zeros([501,501]);
%    for stu = 1:length(x),
%    y2(stu,:) = dz_dy*x(stu);       %%%% slope edits here
%  end


fprintf('*** RANGE RUN %g ***', Range)
fprintf('\r')
  randn('state', S2);
  rand('state', T2);
xe=Range*1000*cos(theta*pi/180);ye=Range*1000*sin(theta*pi/180);

if map_yes_no == 0
  ze=0;
else
  %%%%% Now need to compute ze from map for the given xe and ye
      % Compute indices of map grid points closest to (xe,ye)
      xe_grid_index=round((xe-x_left)/x_res);
      ye_grid_index=round((ye-y_bottom)/y_res);
      % Compute indices of 9 map grid points around (xe,ye)
      x_grid_points=xe_grid_index+[-1 0 1];
      y_grid_points=ye_grid_index+[-1 0 1];
      % Compute z values at 9 map grid points around (xe,ye)
      z_grid_values=y2(x_grid_points,y_grid_points);
      ze=mean(mean(z_grid_values));
end
[S,NAV,xe,ye,ze]=run_dopp_sims(g,T,alt_kft,motion,fo,xe,ye,ze);

subplot(2,4,counterp)
plot(S(2,:)-xe,S(3,:)-ye,'bo');axis equal;axis([-600 600 -600 600])

grid on;
if map_yes_no==0
  stdevx0(counterp) = current_stdx;stdevy0(counterp)=current_stdy;
  stdevz0(counterp) = current_stdz;
else
  stdevx1(counterp) = current_stdx;stdevy1(counterp)=current_stdy;
```

```
  stdevz1(counterp) = current_stdz;
end
  counterp=counterp+1;
end % loop

Range=36:2:50;
figure
title('Solid=STD(X,Y,Z) from Matlab')
subplot(3,1,1)
if map_yes_no==0
 plot(Range,stdevx0,'b-')
 ylabel('Standard Deviation x')
 subplot(3,1,2)
 plot(Range,stdevy0,'b-')
 ylabel('Standard Deviation y')
 subplot(3,1,3)
 plot(Range,stdevz0,'b-')
 ylabel('Standard Deviation z')
else
 plot(Range,stdevx1,'b-')
 ylabel('Standard Deviation x')
 subplot(3,1,2)
 plot(Range,stdevy1,'b-')
 ylabel('Standard Deviation y')
 subplot(3,1,3)
 plot(Range,stdevz1,'b-')
 ylabel('Standard Deviation z')
end % if statement
end % for map loop

figure
%% plotting percentage improvements
title('Percentage Improvements')
subplot(3,1,1)
     plot((Range),(100*(stdevx0-stdevx1)./stdevx0))
axis([min(Range) max(Range) 0 100]);
     ylabel('Percent Improvement x')
     xlabel('Range, km')
subplot(3,1,2)
     plot((Range),(100*(stdevy0-stdevy1)./stdevy0))
axis([min(Range) max(Range) 0 100]);
     ylabel('Percent Improvement y')
     xlabel('Range, km')
subplot(3,1,3)
     plot((Range),(100*(stdevz0-stdevz1)./stdevz0))
axis([min(Range) max(Range) 95 105]);
     ylabel('Percent Improvement z')
     xlabel('Range, km')
```