Algorithms for Global Positioning

Thomas Kok
A07121303

December 1, 2010

Introduction

The Global Positioning System is a network of satellites designed to accurately determine the location of a GPS receiver on the Earth's surface. To this end, the GPS satellites constantly transmit their own locations for any GPS receiver to pick up. Using this data in conjunction with timing data, a GPS reciever in contact with four or more satellites can determine its own location. However, the clocks in consumer GPS receivers are not accurate enough to provide sufficient timing data for a good estimate of location. This is complicated by inconsistent conditions of the medium between a satellite and the reciever, which effectively adds random, mean zero noise to the transmitted positions. Because of this imperfect information, it is necessary for the receiver to implement an optimization algorithm which will compensate for these discrepancies in order to accurately determine its own location. We will derive, implement, and test two algorithms: the naive gradient descent algorithm and the Gaussian gradient descent algorithm.

Both of these algorithms will be formulated to minimize the loss function, which is given by

$$l = \frac{1}{2}||y - h(x)||^2$$

where y is the vector of pseudoranges (the true range plus bias and noise) to the satellites and $h(x)$ is the vector of true ranges calculated from the position vector x, which will be an estimate for our purposes. The pseudoranges can be calculated from the nonlinear function

$$y = R(S) + b + v$$

where b is the clock error, called the bias, and v is the noise. We proceed with a multivariate Taylor Series expansion:

$$R(S) \approx R(S_0) + \frac{\partial}{\partial S} R(S_0) \Delta S$$

where $\Delta S = S - S_0$ and we have discarded higher order terms. Therefore, the linear term of $R(S)$ is

$$\frac{\partial}{\partial S} R(S) = \frac{\Delta S^T}{R(S)}$$

which is the unit row vector pointing in the direction of ΔS. We can then express the pseudorange equation as

$$y = R(S_0) + \frac{\Delta S^T}{R(S)}\Delta S + b + v$$

We see that this estimation is more accurate for large values of $R(S)$ because it drives the difference term to zero.

Now we will derive the naive gradient descent algorithm. We wish to minimze the loss function, given by

$$l = \frac{1}{2}||y - h(x)||^2$$

where y is the vector of ranges transmitted by the satellites and $h(x)$ is the calculated true ranges based on the position/bias estimate x. The gradient of a function points in the direction of greatest increase, so the negative gradient will point out the steepest descent. By making small steps, each in the direction of greatest descent, we can reach a minimum. The gradient of l is

$$\nabla l = -\frac{\partial h(x)}{\partial x}(y - h(x)) = -H^T(y - h(x))$$

so we write our algorithm as

$$x_{k+1} = x_k - \alpha\Delta x = x_k + \alpha H^T(x_k)(y - h(x))$$

where α is a scalar used to control the step size. To avoid excessive overshoot and possible instability, we will choose a value $\alpha < 1$. Smaller values of alpha require more iterations of the algorithm to achieve the same results, so $\alpha = .25$ provides a compromise between efficiency and stability. $H(x_k)$ is the linearization of the pseudorange at x_k with a column of ones appended to incorporate the bias calculation. Since we are minimizing the loss function, our termination criteria will be some maximum value for the loss. We will say that the loss should be less than 10^{-15} ER^2 so that there is no appreciable inaccuracy remaining (if the loss function dictates the total error, which is not necessarily true).

For the Gaussian gradient descent algorithm, we again consider the problem of minimizing the loss function. It is apparent that the ideal solution is $y = h(x)$, which corresponds to $l = 0$. If we linearize $y = h(x)$ about some point x_o the resulting approximation is

$$y \approx h(x_o) + \frac{\partial h(x_o)}{\partial x}\Delta x$$

Inserting this back into the loss function gives

$$l(x) = l(\Delta x) = \frac{1}{2}||\Delta y - H(x_o)\Delta x||^2$$

which leads directly to a solution for Δx

$$\Delta x = H^+(x_o)\Delta y = H^+(x_o)(y - h(x_o))$$

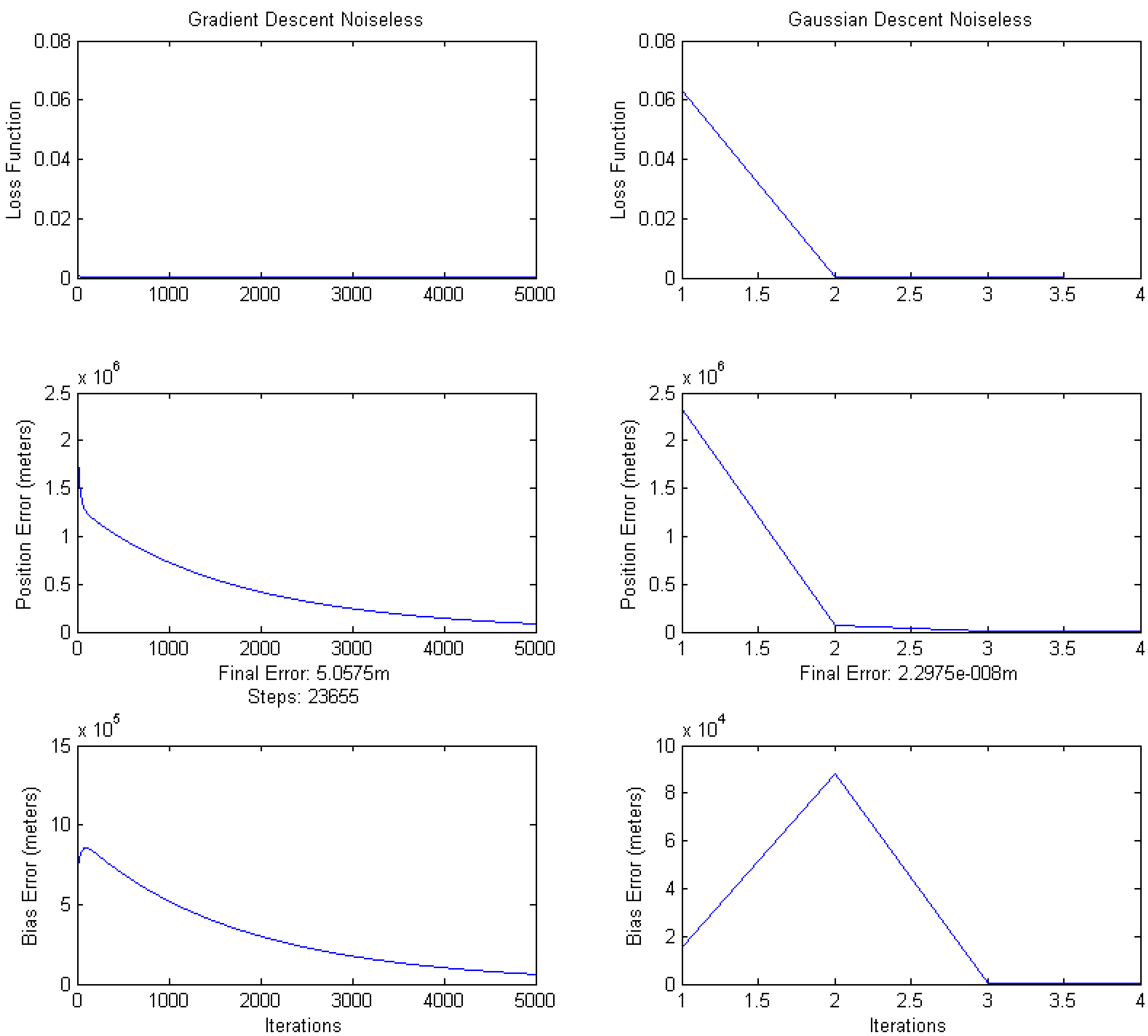
We can now write the iterative algorithm as

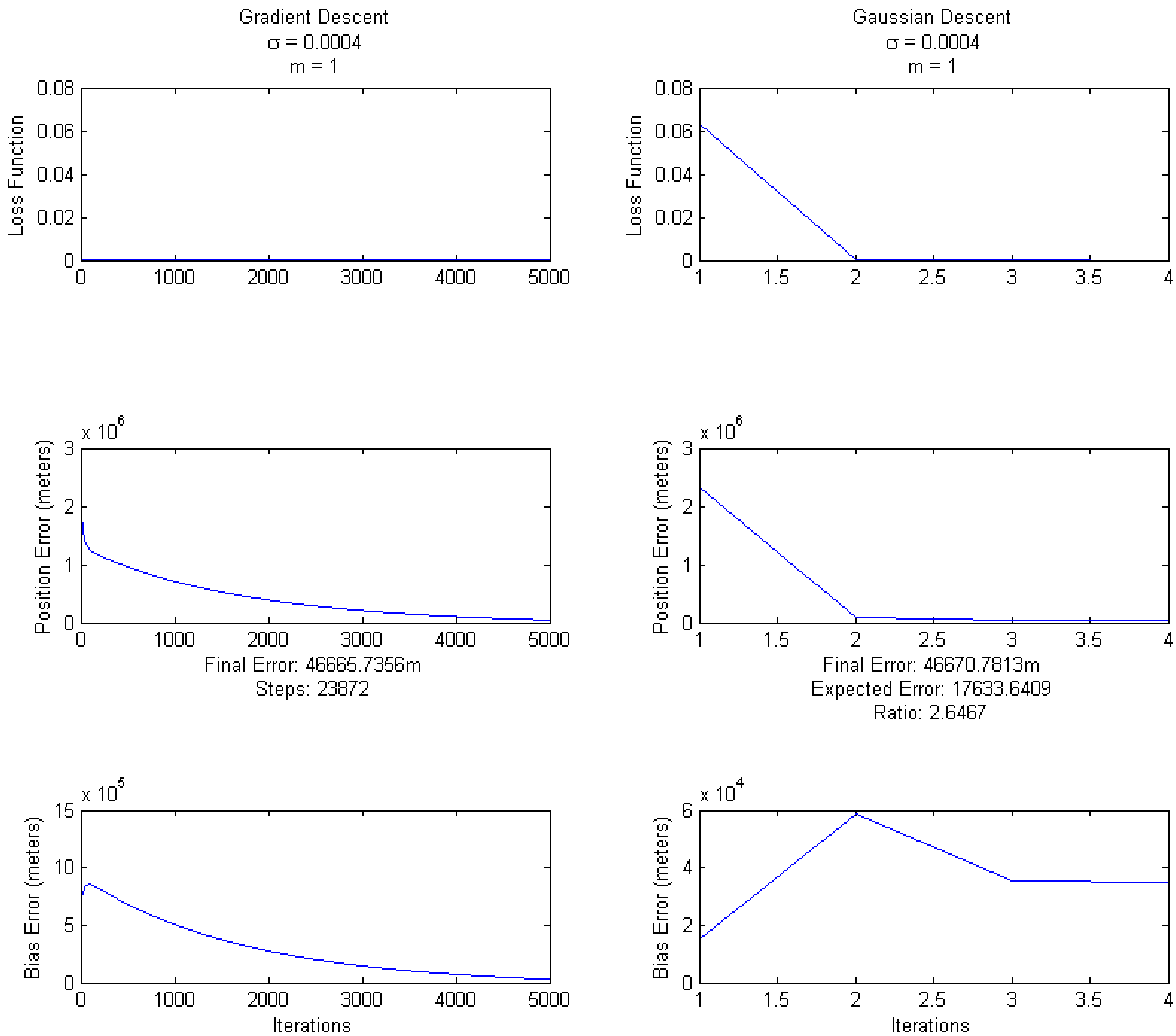
$$x_{k+1} = x_k + \alpha H^+(x_o)(y - h(x_o))$$

where $H^+(x_o)$ is the pseudoinverse. If $H(x_o)$ is square and full rank, the pseudoinverse is identical to the normal inverse, which may be used instead for efficiency. We will use four satellites to test these algorithms, so we may use the normal inverse. For this algorithm we will use $\alpha = 1$, which is the standard least squares solution applied iteratively, and the same termination condition.

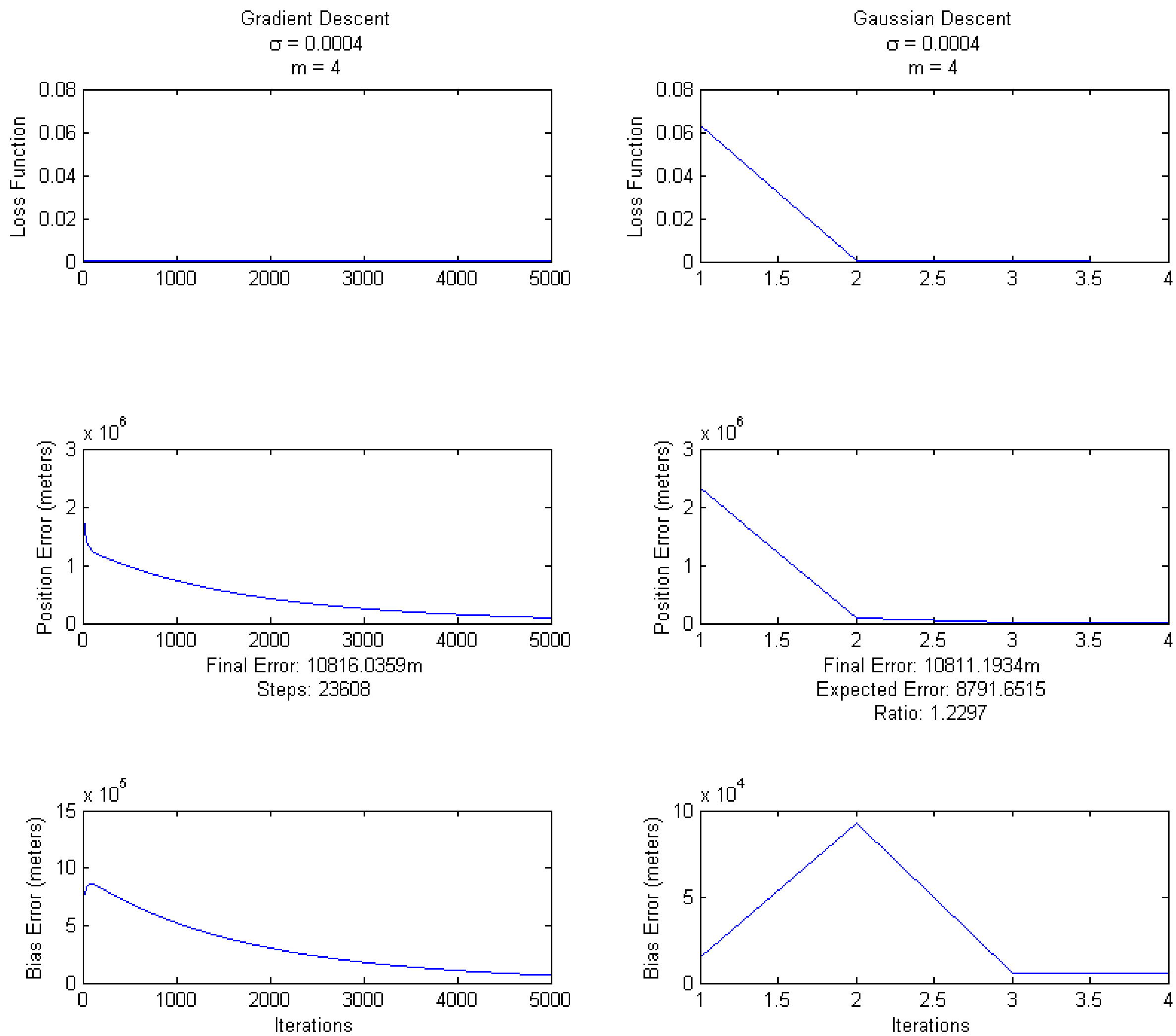
Simulation

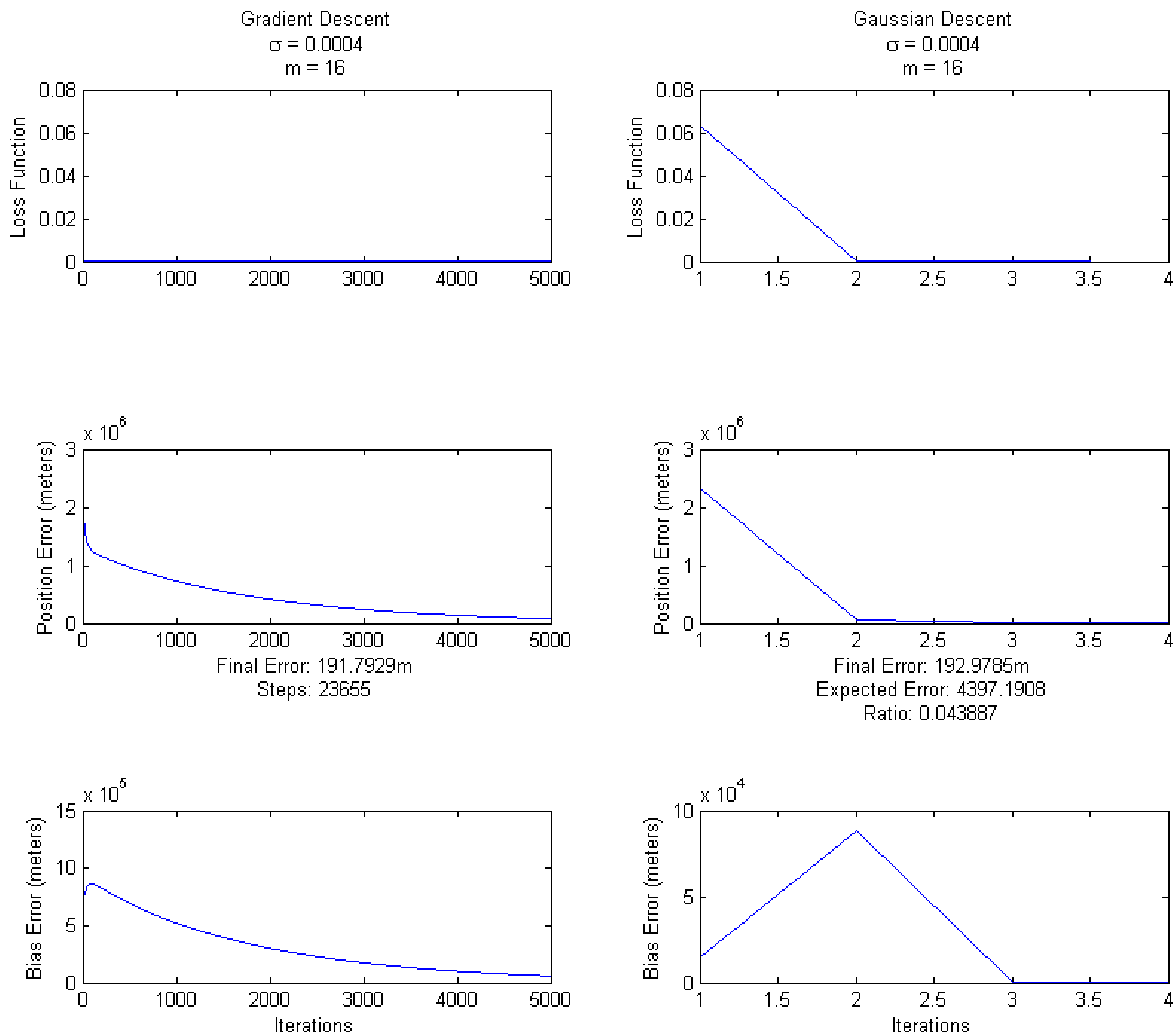
We will test these algorithms with reasonable arbitrary data. We will use four satellites, the minimum number necessary to find a solution. We will run tests with noiseless data, then with noise approximated by adding the mean of m mean zero, variance σ^2, normally distributed numbers to each pseudorange, where $m = \{1, 4, 16, 256\}$ is the number of measurements taken to each satellite and $\sigma$is the standard deviation of the noise. We expect that averaging noise will cause it to tend toward zero, affecting our calculations less, so more measurements implies greater accuracy. The algorithms will be given an initial position guess that is on the Earth but not very close to the true location, and an initial bias guess of zero earth radii.

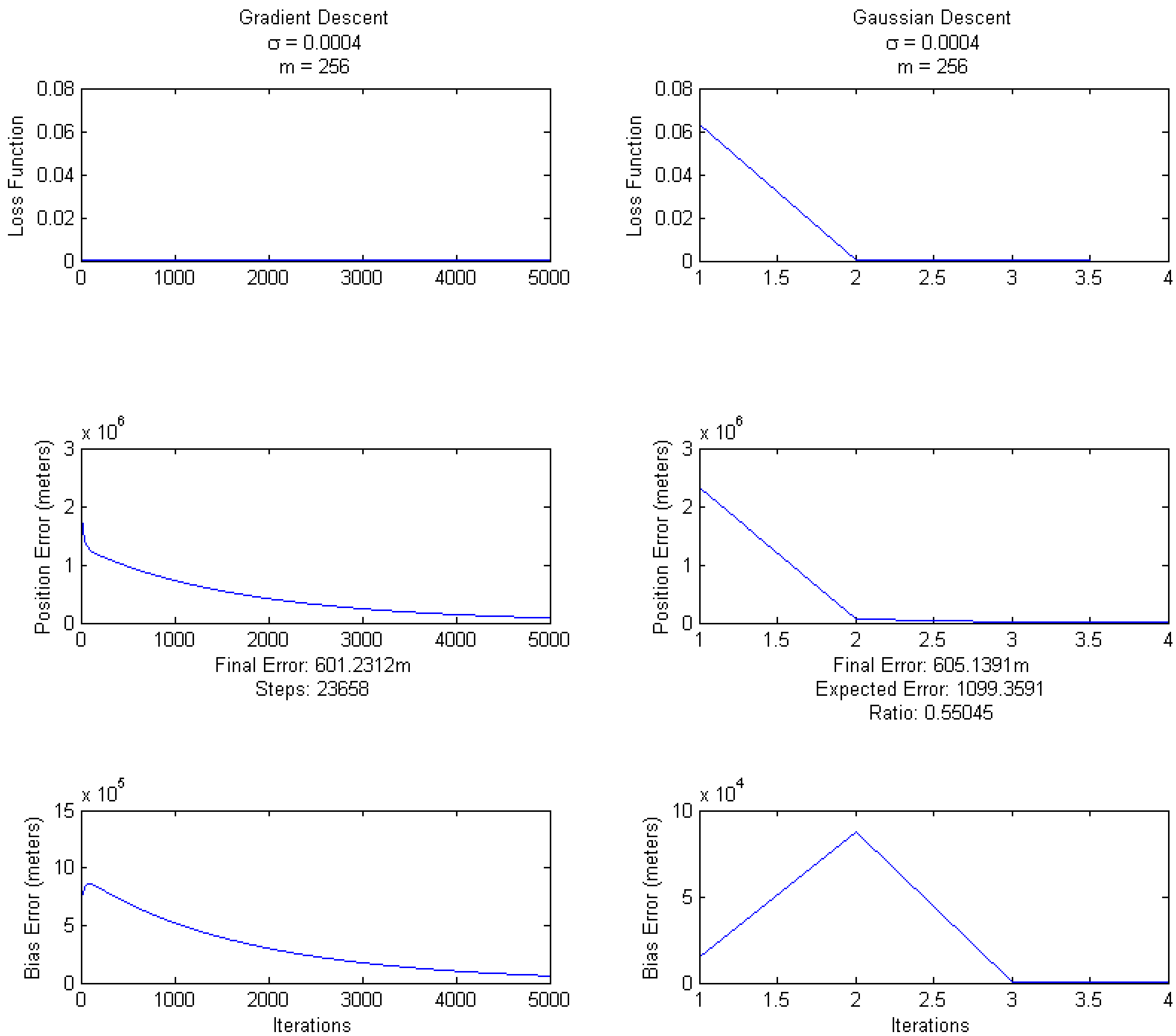




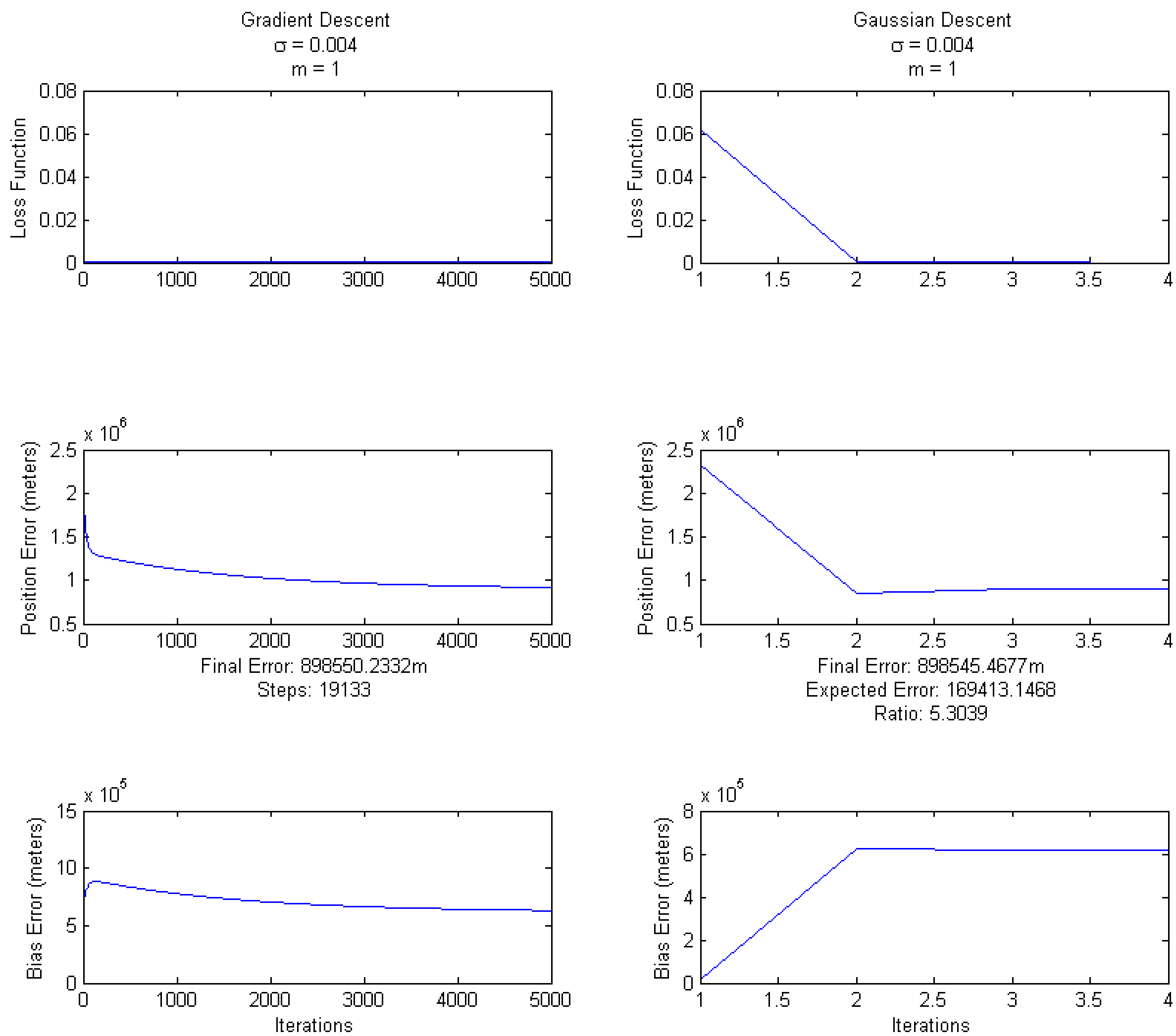
Gradient Descent
σ = 0.0004
m = 1
Gaussian Descent
σ = 0.0004
m = 1
Loss Function
Position Error (meters)
x 10^6
Final Error: 46665.7356m
Steps: 23872
Final Error: 46670.7813m
Expected Error: 17633.6409
Ratio: 2.6467
Bias Error (meters)
x 10^5
x 10^4
Iterations

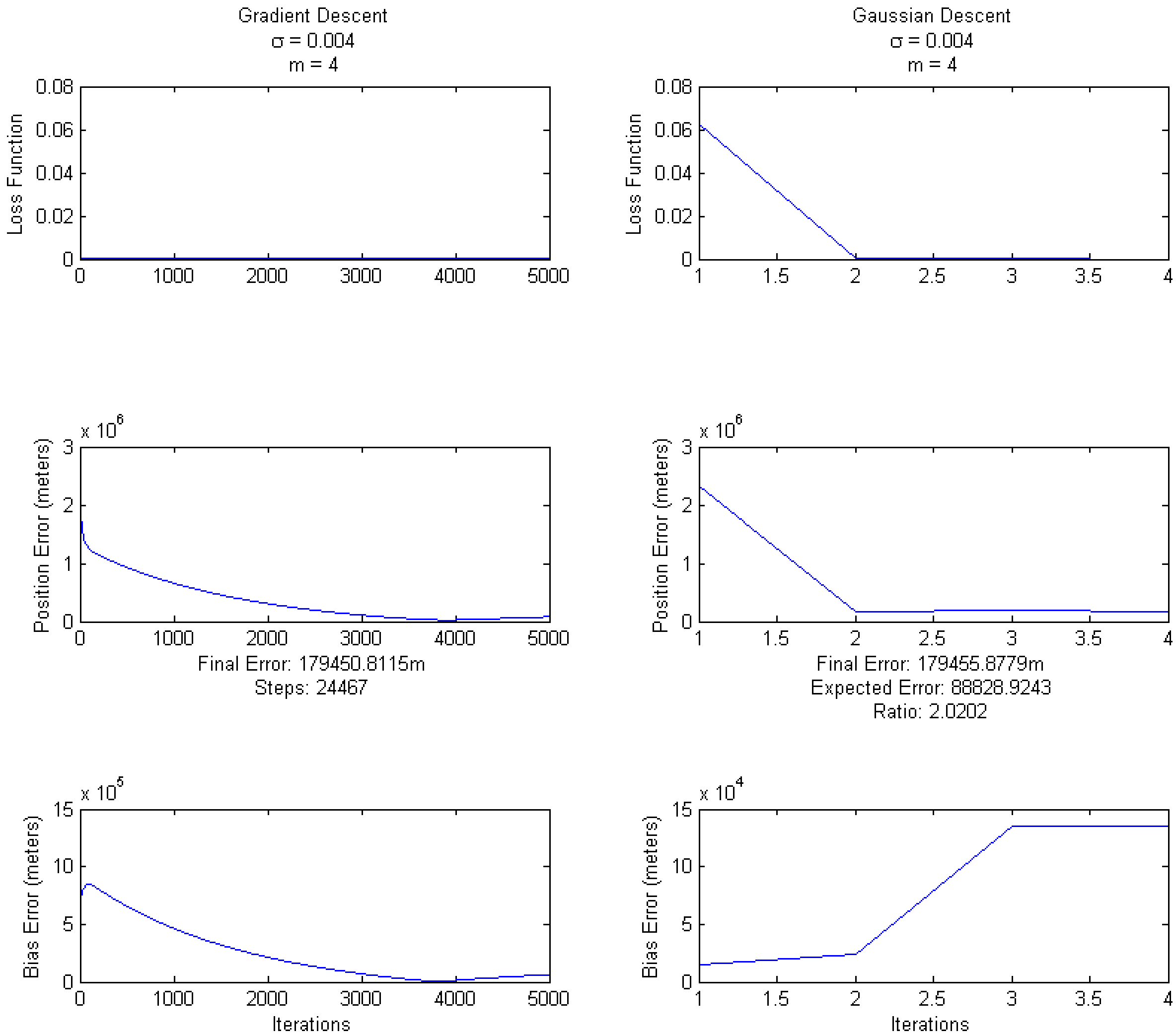

Gradient Descent
σ = 0.0004
m = 4
Gaussian Descent
σ = 0.0004
m = 4
Loss Function
Position Error (meters)
Final Error: 10816.0359m
Steps: 23608
Final Error: 10811.1934m
Expected Error: 8791.6515
Ratio: 1.2297
Bias Error (meters)
Iterations

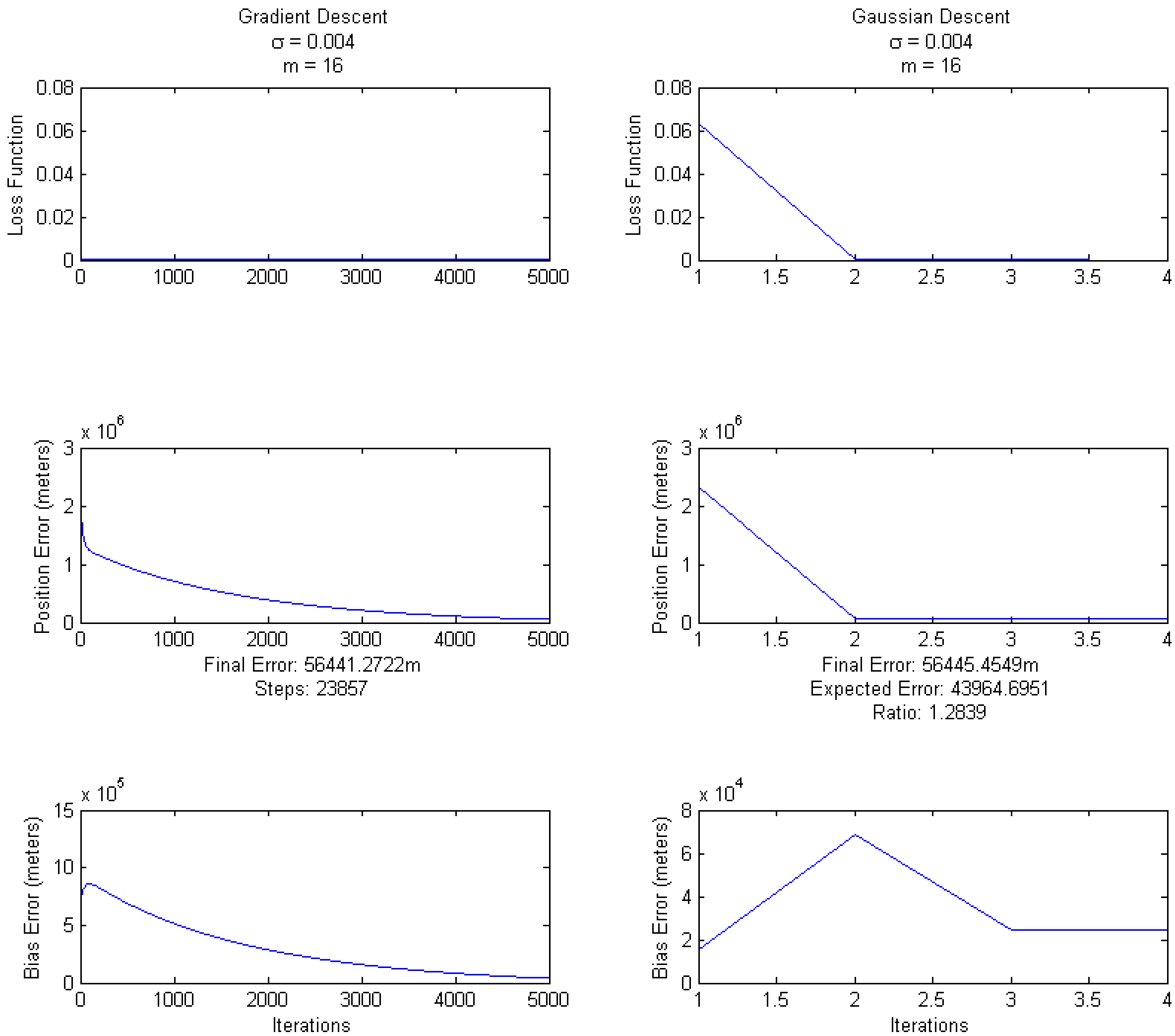

Gradient Descent
σ = 0.0004
m = 16
Gaussian Descent
σ = 0.0004
m = 16
Loss Function
Position Error (meters)
Bias Error (meters)
x 10^6
x 10^5
x 10^4
Final Error: 191.7929m
Steps: 23655
Final Error: 192.9785m
Expected Error: 4397.1908
Ratio: 0.043887
Iterations
Iterations

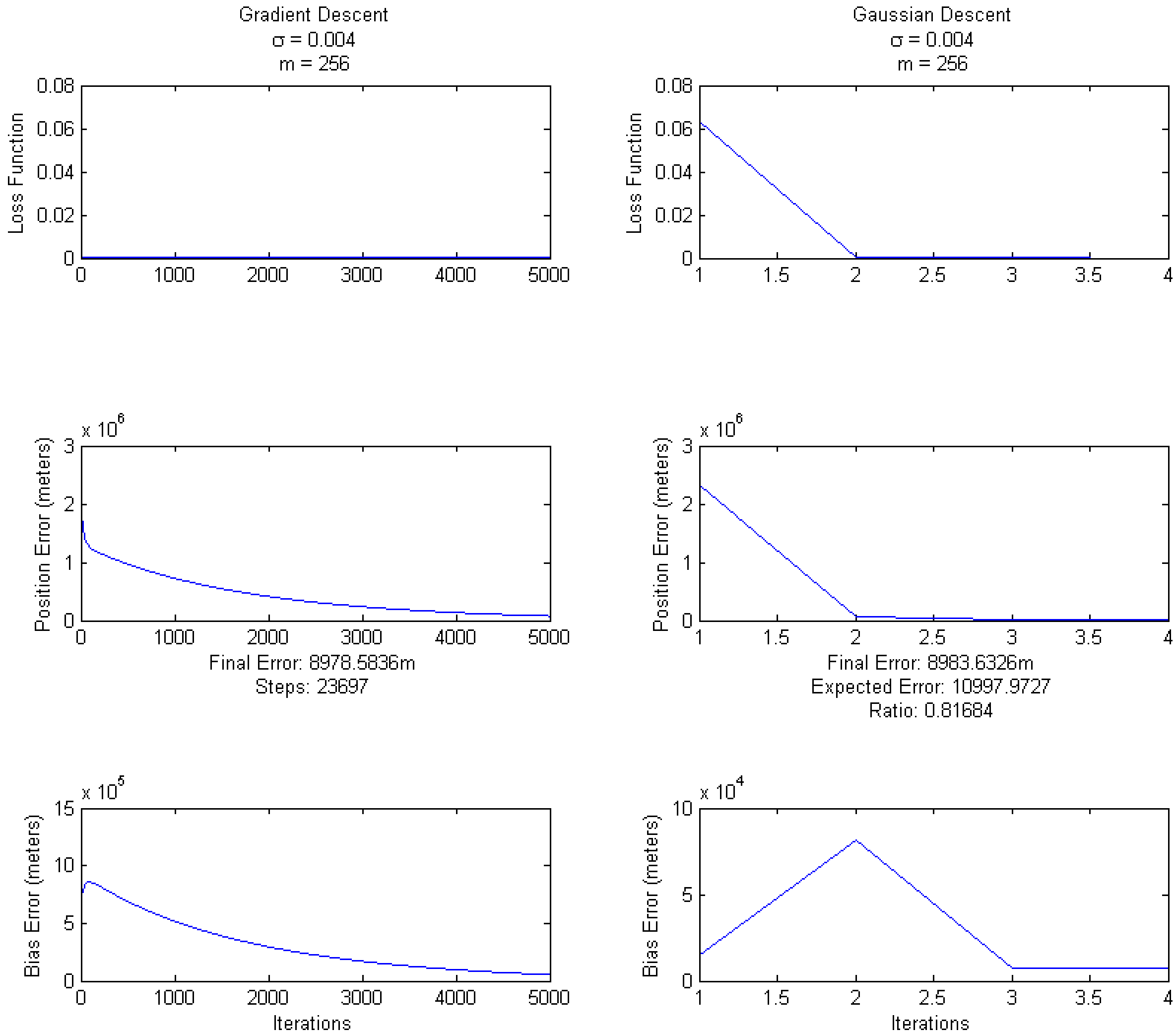


Gradient Descent
σ = 0.0004
m = 256
Loss Function
Position Error (meters)
Final Error: 601.2312m
Steps: 23658
Bias Error (meters)
Iterations
Gaussian Descent
σ = 0.0004
m = 256
Loss Function
Position Error (meters)
Final Error: 605.1391m
Expected Error: 1099.3591
Ratio: 0.55045
Bias Error (meters)
Iterations


Gradient Descent
σ = 0.004
m = 1
Gaussian Descent
σ = 0.004
m = 1
Loss Function
Position Error (meters)
x 10^6
Final Error: 898550.2332m
Steps: 19133
Final Error: 898545.4677m
Expected Error: 169413.1468
Ratio: 5.3039
Bias Error (meters)
x 10^5
Iterations


Gradient Descent
σ = 0.004
m = 4
Gaussian Descent
σ = 0.004
m = 4
Loss Function
Position Error (meters)
x 10^6
Final Error: 179450.8115m
Steps: 24467
Final Error: 179455.8779m
Expected Error: 88828.9243
Ratio: 2.0202
Bias Error (meters)
x 10^5
x 10^4
Iterations


Gradient Descent
σ = 0.004
m = 16
Loss Function
Position Error (meters)
Final Error: 56441.2722m
Steps: 23857
Bias Error (meters)
Iterations
Gaussian Descent
σ = 0.004
m = 16
Loss Function
Position Error (meters)
Final Error: 56445.4549m
Expected Error: 43964.6951
Ratio: 1.2839
Bias Error (meters)
Iterations



We see that, given enough iterations, the naive gradient descent algorithm returns very similar results to the Gaussian descent algorithm. Neither method was able to consistently calculate accurate locations in the presence of noise. While not desireable, this is not unexpected, as the algorithms minimize the loss function rather than the position error. Clearly they are finding minima to the loss function that do not always correspond to the true location. In every case the optimization did decrease the position error, but except in the noiseless case, not by enough to determine the true location to useful accuracy. The Gaussian algorithm is generally not too far off its predicted error.

Conclusion

It is apparent that the Gaussian algorithm is far superior to the naive algorithm. Although each step of the Gaussian iteration does involve more complicated calculations, the vastly reduced number of steps for convergence makes it hundreds or thousands of times more efficient. As we anticipated, having more measurements available from each satellite genereally reduces the error in the calculations.