GPS position calculation using two nonlinear optimization methods

Dimitri Schreiber

December 3, 2015

Purpose

Temporal and spatial localization is a major challenge. In this white-paper I analyze a naive interpretation of GPS for calculating a receiver's true time and position using two methods for solving the nonlinear least squares problem presented, there is an inconsistent system for our position and time, we are trying to determine a least squares error fit to our data. Satellites transmit their position and time (at time of transmission). The satellites position and time information is assumed to be accurate to an arbitrarily high precision. Given the satellite's position and time for us to receive the transmission we can calculate our distance to each satellite, and triangulate our position from that. Given perfect noiseless data with a perfect system clock this would be a trivial problem, our system would be consistent, we could solve for one intersection of our satellite ranges that would give us our true position, and our internal clock would be correctly synced to satellite's time.

Unfortunately there are multiple sources of error, the speed of light varies depending on the medium it is traveling through, therefore we can not calculate our position perfectly based on the time lag between us receiving the satellite's position and our position, as we do not know the average speed of light for the signal to travel to us. We also have a clock drift, our internal clock is not consistent with the satellite's clock. We have four unknowns that are consistent between our satellite samples, and noise that varies with each sample, due primarily to the atmosphere.

This new system is modeled by:

$$y_\ell = R_\ell(S_{\text{true}}) + b + n$$

Where y_ℓ is our pseudo-range we calculate from our observed information (it is not our actual range from the satellite), $R_\ell(S_{\mathbf{true}}) = \sqrt{(S_{\text{true}} - S_\ell)^T(S_{\text{true}} - S_\ell)}$, $S_{\mathbf{true}}$ is true receiver position we are attempting to determine, S_ℓ is satellite position, b is the clock drift distance, and n is our noise. We are attempting to solve for S_{true} and b, so as to minimize our sum of squared errors across our four satellites of the difference between our pseudo-ranges calculated from the satellite samples and our ranges calculated using the above equation, $R_\ell(S_{\text{true}}) + b + n$, using our learned position and clock drift. The receiver calculates y_ℓ, our pseudo-range from the satellites by multiplying the delta time between satellite transmission and receiver times the speed of light estimate.

For our simplified analysis we are ignoring the noise, n, that differs between samples by setting it to zero. This greatly simplifies our problem, and yields an unrealistically accurate result.

Procedure

We are attempting to minimize the least squared error,

$$\ell(\hat{X}_k) = \|y - h(\hat{X}_k)\|^2$$

where y is our pseudo range calculated from our real world data, $\hat{X}_k$ is our current estimate and $h(\hat{X}_k) = R_\ell(\hat{S}_k) + \hat{b}_k + n$ is our pseudo-range estimate given our current 4D position estimate $\hat{X}_k$ composed of $\hat{S}_k$ and $\hat{b}_k$. We solve for the minimum error through through two iterative algorithms, Steepest Gradient Descent and Gauss Newton Descent. We are attempting to solve for the parameters of our underlying model, X, given our observed data y. y and $h(\hat{X}_k)$ are both length four column vectors, due to our four satellite samples.

Steepest Gradient Descent

Steepest Gradient Descent steps against the gradient, the step that leads to the largest decrease in *loss* at this particular point. The gradient is represented by Jacobian Matrix of our cost (loss) function, $\nabla_X \ell(X)$. This is subtracted from our previous update, times a small term, a_k to prevent significant overshoot and instability as the gradient is only perfectly accurate for an infinitesimally small range. We then solve for our four parameters, our X, Y, and Z location as standard cartesian distances from the center of the earth, and our clock drift b, all in units of Earth Radiuses (6370km). Our results in analysis will be reported in meters, multiplying by $6.37 * 10^6$.

Our position is solved for using an iterative algorithm where our general update rule is:

$$\hat{X}_{k+1} = X_k - \alpha_k \nabla_X \ell(\hat{X}_k)$$

Where $\nabla_X \ell(X) = \left(\frac{\partial}{\partial X}\ell(X)\right)^T = -H^T(\hat{X}_k)\left(y - h(\hat{X}_k)\right)$. Yielding $\hat{X}_{k+1} = X_k + \alpha_k H^T(\hat{X}_k)\left(y - h(\hat{X}_k)\right)$ as our specific update rule.

$$H(X) = \frac{\partial}{\partial X} h(X) = \begin{pmatrix} r_1^T(S) & 1 \\ r_2^T(S) & 1 \\ r_3^T(S) & 1 \\ r_4^T(S) & 1 \end{pmatrix} \in \mathbb{R}^{4\times 4},$$

and $r_\ell(S) = \frac{S - S_\ell}{\|S - S_\ell\|}$ for $l = 1, 2, 3, 4$. $r_\ell(S)$ is a unit vector pointing from the satellite to our receiver position estimate, therefore a movement along it is the maximum increase in distance from the satellite, as any other angle decreases with *cosine* of the angle off of this vector

Gauss Newton Method

In the Gauss Newton Method our real position is solved for using the same general method as above, however we are far more greedy with our movements. We linearize our loss function, $l(X)$ around our current position estimate (four dimensional) $\hat{X}_k$ by taking the Taylor expansion of our loss function and only keeping the first order terms. Then we solve this one to one least squares problem, resulting in the update rule

$$\hat{X}_{k+1} = \hat{X}_k + \alpha_k H^{-1}(\hat{X}_k)(y - h(\hat{X}_k))$$

Where $H^{-1}(X) = \left(H^T(\hat{X}_k)H(\hat{X}_k)\right)^{-1}$

This is derived by linearizing the difference inside the loss function, $y - h(\hat{X}_k)$, to the first order taylor expansion. Resulting in $y - h(X) - H(X)(X_{update} - X)$ where X is the value the function in linearized around, our current receiver position guess. This can be rewritten as $\Delta y - H(X)\Delta X$ This is now a standard linear one to one least squares problem of the form $\Delta y = H(X)\Delta X$: $\Delta X = (H(X)^T H(X))^{-1} H(X)^T \Delta y$

This happens to be the same process as Newton's Method where the second order derivative terms are removed. Newton's Method is an iterative method to find the roots of an equation. In higher order systems it is of the form $X_{new} = X_{current} - \nabla f(X_{current})^{-1} f(X_{current})$. In our case we want to solve for the zero of the gradient, as that corresponds to an extrema of the function, in our case the minimum. This above method can be similarly be

applied to a system where $f(X_{current}) = \nabla l(X_{current})$. This yields the following system, where $Hf(X_{current})$ = Hessian of $f(X_{current}) = \nabla\nabla l(X_{current})$

$$X_{new} = X_{current} - Hf(X_{current})^{-1}\nabla f(X_{current})$$

The Gauss Newton method follows the above system, when the second order terms of the Hessian are ignored. This is a fair assumption when those second order terms are small.

For both methods the update method is looped through iteratively until our error has become sufficiently small. We can not directly compute our error, therefore our break criterion is instead our position no longer moving significantly over many iterations, implying we have reached a true minimum of our system, rather than a temporary slow down.

Artificial data is created by choosing and initial receiver position, $S = (x, y, z)$, along with the clock drift of the receiver, b. These are the true values of our system. An initial position and clock drift value guess is given for the receiver:

$$\hat{X}_1 = (0.93310, 0.25000, 0.258819) = (S, b)$$

The true satellite positions, number S_1 through S_4 are:

$$S_1 = (3.585200000, 2.070000000, 0.000000000)^T ER$$
$$S_2 = (2.927400000, 2.927400000, 0.000000000)^T ER$$
$$S_3 = (2.661200000, 0.000000000, 3.171200000)^T ER$$
$$S_4 = (1.415900000, 0.000000000, 3.890400000)^T ER$$

The true receiver position and clock bias are:

$$S = (1.000000000, 0.000000000, 0.000000000)^T$$

$$b = 2.35478806810^{-3} ER$$

Our pseudoranges, Y_1 through Y_4 are calculated using the system formula

$$y_\ell = R_\ell(S_{\text{true}}) + b + n$$

We test the effectiveness of our solution methods by giving initial position and clock bias guesses:

$$\hat{S}_0 = (0.93310, 0.25000, 0.258819)^T ER$$
$$\hat{b}_0 = 0$$

Analysis and Results

Important variables:

`S_genie` = true receiver position

`S_sat` = satellite position

`b_genie` = true receiver clock bias

`y` = pseudo range

`S` = current receiver position estimate

`b` = current receiver clock bias estimate

`a` = learning rate

`grad_descent_error_iter_position` = vector of receiver position errors, each iteration, steepest gradient descent update, in meters

`gauss_newton_error_iter_position` = vector of receiver position errors, each iteration, gauss newton update, in meters

`gauss_newton_loss` = vector of losses, each iteration, gauss newton update

`gradient_descent_loss` = vector of losses, each iteration, steepest gradient descent update

`grad_descent_error_iter_time` = vector of receiver time errors, each iteration, steepest gradient descent update, in meters

`gauss_newton_error_iter_time` = vector of receiver time errors, each iteration, gauss newton update, in meters

Graphs are displayed at the end of the program run. All figures are included at the end of this document.

As discussed above, two methods of nonlinear optimization were tested and will now be compared. The step size, a, equals 0.1 for Steepest Gradient Descent and a equals 1 for Gauss Newton Descent. For Steepest Gradient Descent two other step sizes were tested, $a = 0.2$ and $a = 0.05$, they were unstable and exceedingly slow, respectively. The stopping criteria was selected to be when the position change was less than $10^{-}10ER$, which would correspond $0.67mm$, after between 50 iterations for Steepest Gradient Descent and 2 iterations for Gauss Newton Descent. This would hopefully imply that we are at a resting point, rather than a local slow down.

In tests this stopping method resulted in a position error of $5cm$ after 81351 iterations for Steepest Gradient Descent and a position error of less than a nanometer for Gauss Newton Descent after 6 iterations. As shown in *Figure 6* it appears that Gauss Newton method has reached a minimum, while Steepest Gradient Descent is still decreasing. This is also shown by the significant error left in Steepest Gradient Descent. Given the slow rate at which it converges a smaller per iteration change should have been tolerated for exiting. An increased number of iterations between the two compared

values would solve this issue, perhaps a value of $5,000$ rather than the value of 50 currently used.

As illustrated in *Figures 4, 5, and 6* Gauss Newton method fully converges within 5 iterations, while Steepest Gradient Descent still has significant error after 80 thousand iterations. This shows the tremendous power of Gauss Newton method for problems with small second order derivatives, maintaining much of the power of Newton's Method without the complexity of computing the full hessian matrix. It appears that in this particular instance Gauss Newton method has converged at a quadratic rate, as Newton's Method which does converge at up to a quadratic rate will fully converge for double precision floating point within six steps (every iteration double the number of correct digits: $2, 4, 8, 16, 32, 64$

As the true range increases the linear approximation of the error becomes better, this is due to the surface of the loss function being flatter. This can be analyticaly shown by calculating the magnitude of the second direvative, or in our case the determinant of the second order terms of the Hessian matrix. $y - h(X)$ is the error, $h(X)$ is linearized. $h(X) = R(S) + b$

$$H(X) = \frac{\partial}{\partial X} h(X) = \begin{pmatrix} r_1^T(S) & 1 \\ r_2^T(S) & 1 \\ r_3^T(S) & 1 \\ r_4^T(S) & 1 \end{pmatrix} \in \mathbb{R}^{4\times 4},$$

and $r_\ell(S) = \frac{S - S_\ell}{\|S - S_\ell\|}$ for $l = 1, 2, 3, 4$

The second order terms of the Hessian are $\frac{\partial}{\partial S} r_\ell(S) = \frac{\|S-S_\ell\| - (\frac{\partial}{\partial S}\|S-S_\ell\|)(S-S_\ell)}{\|S-S_\ell\|^2} \leq \frac{2}{\|S-S_\ell\|}$. Given that $\|S - S_\ell\|$ is the true range, as the true range increases the second order terms decrease, making our first order linear approximation better, the loss function surface becomes flatter with larger true ranges.

Conclusion

In this white-paper we analyzed two methods of nonlinear optimization and applied them to a nonlinear problem, position determination from satellite ranges, a GPS(GLONASS) problem. We used a model of the system to determine our underlying position and clock bias given inconsistent distances to each satellite. We had a significantly simplified project as we did not include sampling noise, we assumed they were perfect noiseless samples. This resulted in our system converging to a perfectly accurate result which is unrealistic. It also allowed us to take a far more naive approach to solving the problem, we did not have to trust different samples and satellites differently,

or calculate the precision of each sample (or satellite given the path between us and the satellite).

This allowed us to solve the problem with significantly less effort, however it still adequately showed the comparative speed advantage of Gauss Newton method over Steepest Gradient Descent.

Figures

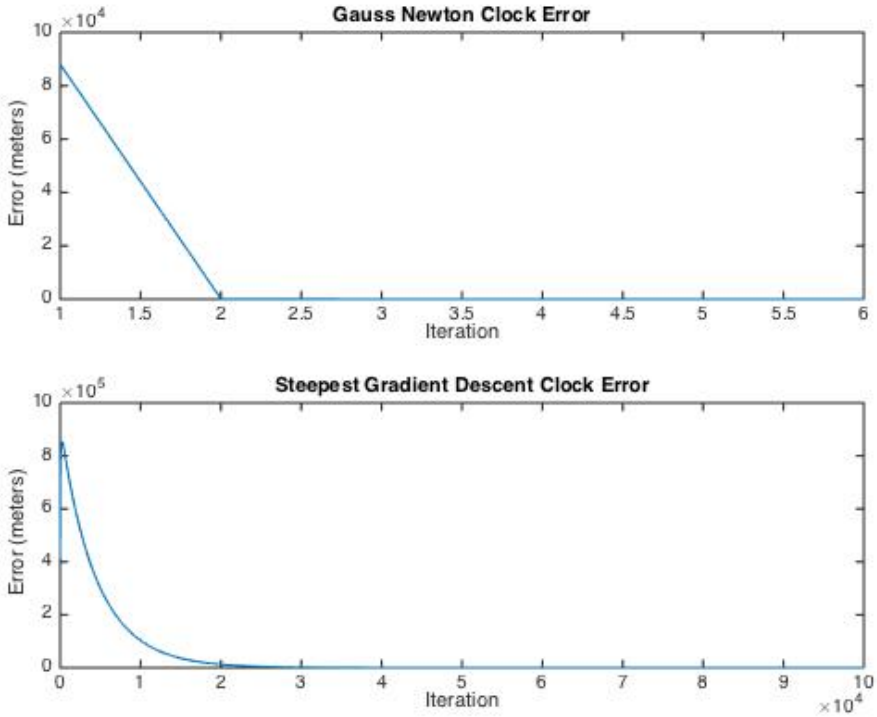

Figure 1: Comparison of clock bias error

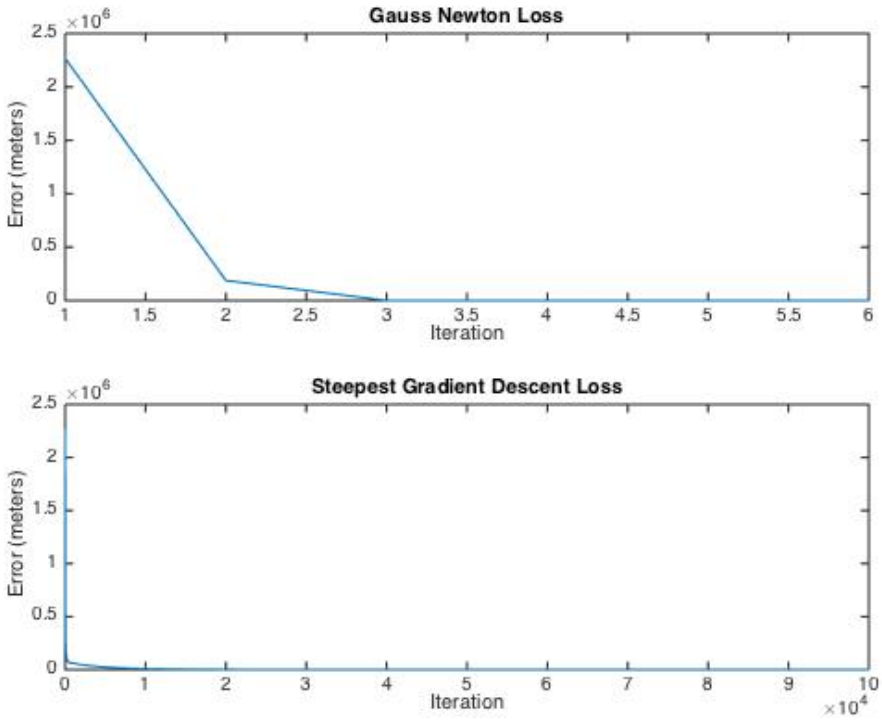

Figure 2: Comparison of loss values

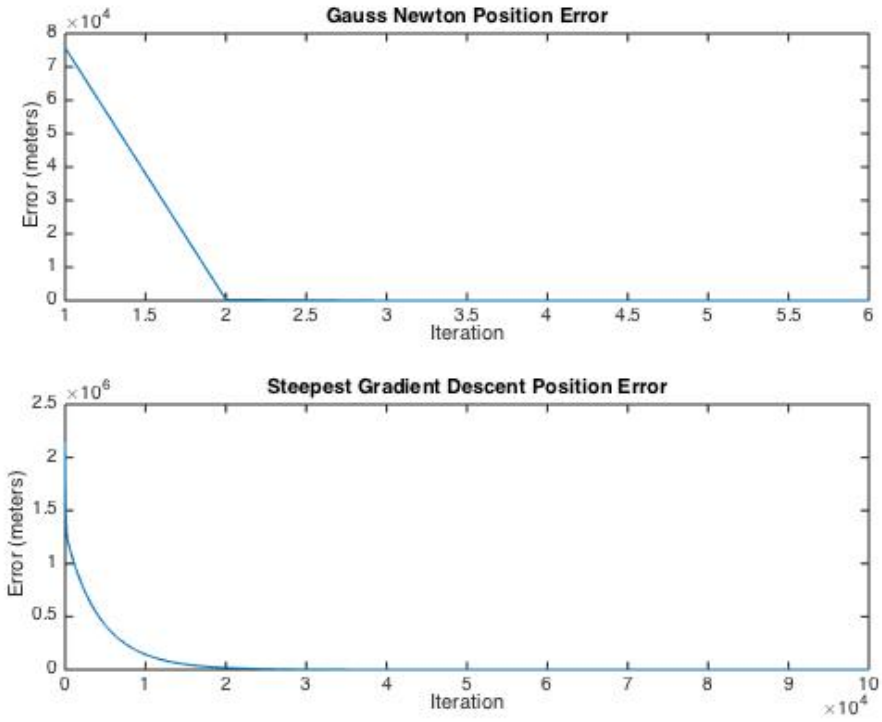

Figure 3: Comparison of position error

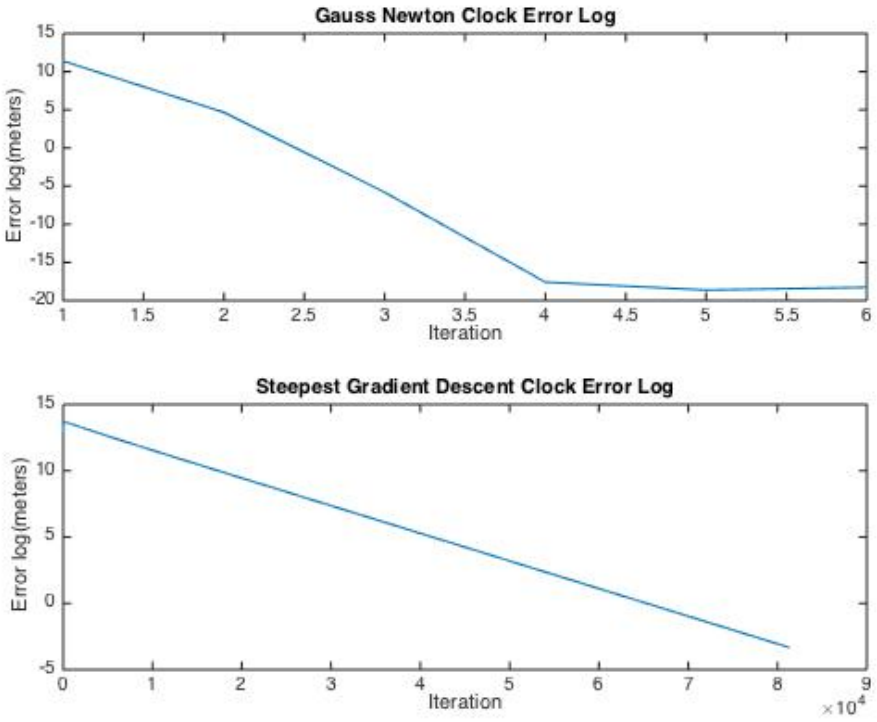

Figure 4: Comparison of clock bias error with logarithmic scale

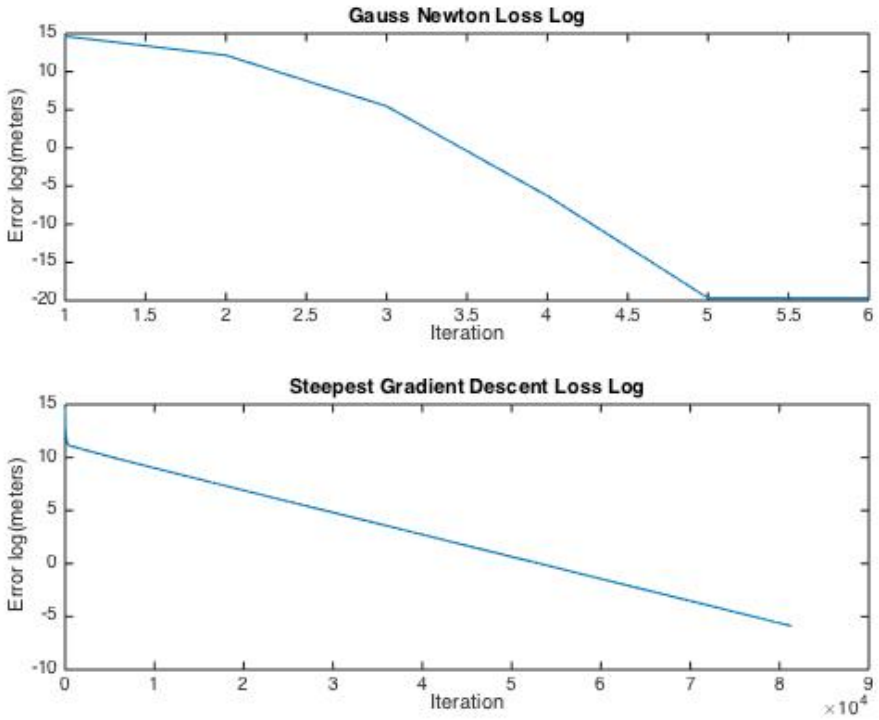

Figure 5: Comparison of loss value with logarithmic scale

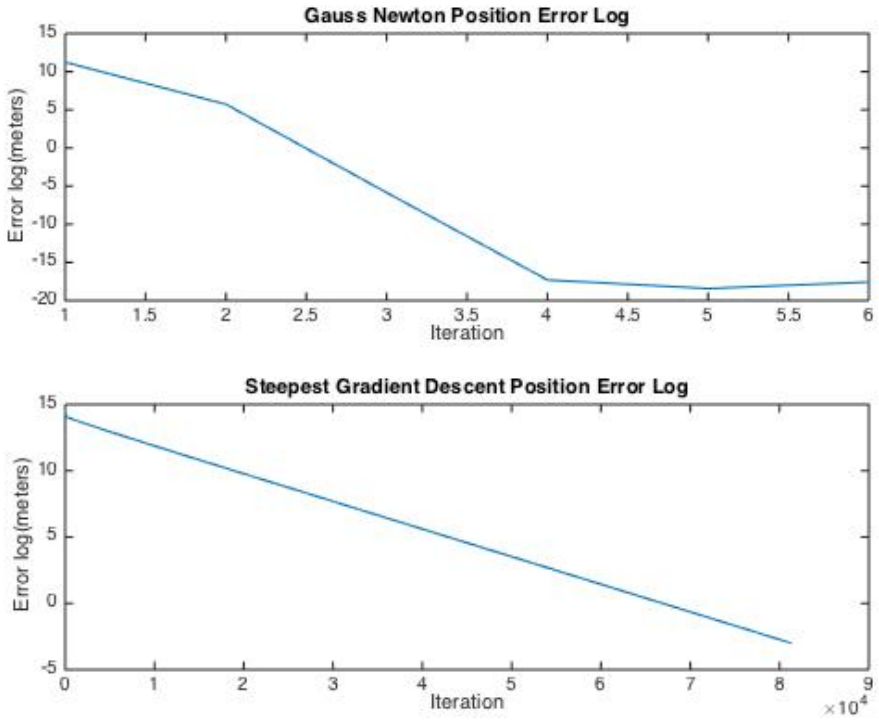

Figure 6: Comparison of position error with logarithmic scale

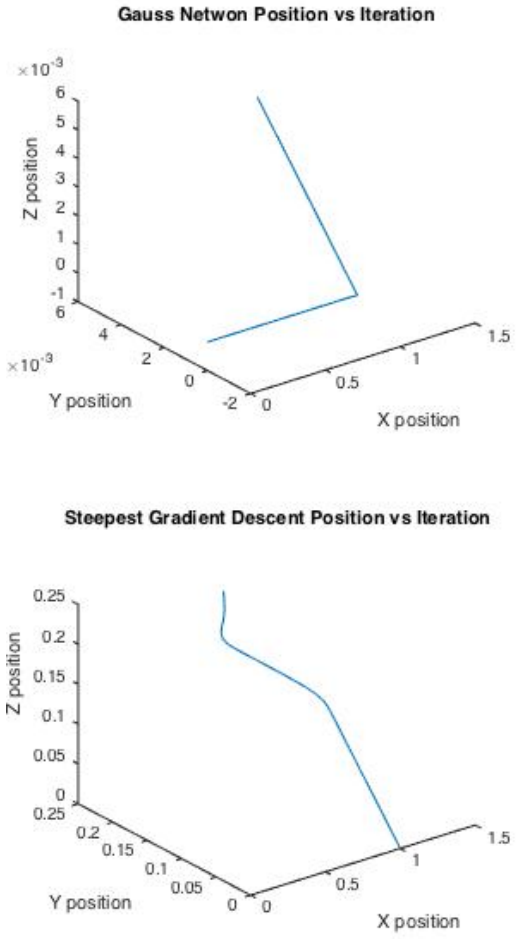

Figure 7: Comparison of 3D position between descent methods

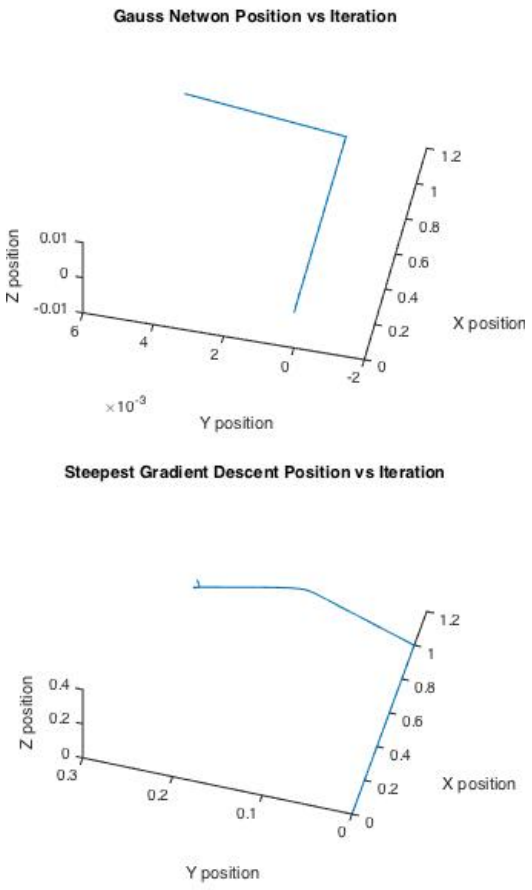

Figure 8: Comparison of 3D position between descent methods