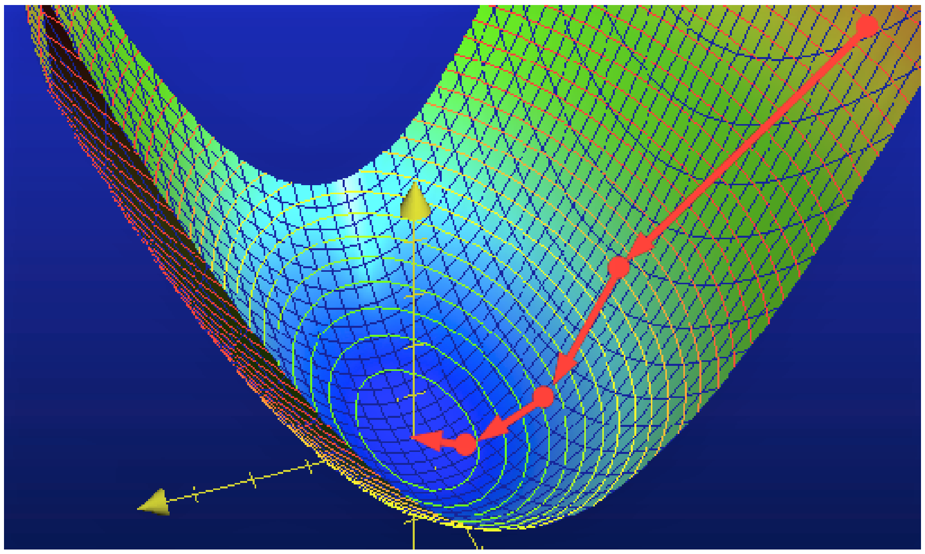

GLOBAL POSITIONING SYSTEM (GPS) ALGORITHM

ECE 174 - Project 2
Internal Technical Memorandum

Professor :
Ken Kreutz-Delgado

Author :
Pierre MOREAU
U07342256

December 1, 2016

Contents

Page

Purpose **1**
Notation 1

Procedure **3**
Step 1 : Linearization 3
Step 2 : Algorithm Development 3
Step 3 : Simulation 3

Results **5**
Step 1 5
Step 2 5
Step 3 7

Conclusion **8**

Appendices **9**

List of Figures

Page

Appendices **10**

Step 2 10

1 Naive Gradient Descent : convergence vs α 10

2 Gaussian Gradient Descent : convergence vs α 10

Step 3 11

3 Convergence of the naive gradient descent algorithm on log scale 11

4 Convergence of the Gaussian gradient descent algorithm on log scale 12

Purpose

GPS positioning is done by receiving positions and time from different satellites in space. The true range between the receiver and the satellite is determined using timing information. *ie* by denoting t_l the time received by the l^{th} satellite, c the speed of light in a vacuum, and t the true time of the signal reception :

$$true\ range := R_l = c(t - t_l) \tag{1}$$

By knowing the true ranges between the receiver and different satellites, one can be located in space. It is assumed that the data received by satellites is accurate, but unfortunately the timing measurements of the receiver are often inaccurate[1] and have to be corrected. The range measured by the receiver is an error-induced pseudo-range.

The motivation of this project is to develop algorithms that gives an accurate localization using the error-induced pseudo-range measured by the receiver.

Notation

All positions are given with respect to a geostationary reference Cartesian coordinate system located at the center of the earth. Distances are given in units of Earth Radii ($1ER = 6.370km$). Since the positioning relies on 3 degrees of freedom for the unknown receiver location, and 1 degree of freedom for the receiver's clock bias b, we need measurements from 4 satellites in order to have enough data to solve the problem.

$S_l = (s_{l,1}, s_{l,2}, s_{l,3})^T$ is the position of the l^{th} satellite, $l = 1, 2, 3, 4; S = (s_1, s_2, s_3)^T$ is the position of the receiver. For a fixed satellite location S_l, the true range R_l from the satellite located at S_l is a nonlinear function of the receiver location S and is given by :

$$R_l(S) = ||S - S_l|| = \sqrt{(S - S_l)^T (S - S_l)},\ l = 1, ..., 4 \tag{2}$$

The errors in the pseudo-range are induced by the receiver's clock bias $b = c(t' - t)$, with t' the reception time measured by the receiver, and some noise $\nu_l \sim N(0, \sigma^2)$ induced by atmospheric inconsistencies. The pseudo-range is then given by :

$$y_l = R_l(S) + b + \nu_l,\ l = 1, ..., 4 \tag{3}$$

[1]Satellites are equipped with expensive atomic clocks, and their orbit is accurately defined, while commercial GPS units are equipped with a cheap clock to keep the prices low.

For problem notation, we define $x = \begin{pmatrix} S \\ b \end{pmatrix} \in R^4$, and $h(x) = \begin{pmatrix} R_1(S)+b \\ R_2(S)+b \\ R_3(S)+b \\ R_4(S)+b \end{pmatrix}$. So the pseudo-range equation (3) becomes :

$$y_l = h_l(x) + \nu_l,\ l = 1, ..., 4 \tag{4}$$

For simplification purposes, we ignore the noise, *ie* $\nu_l = 0$.

Procedure

Step 1 : Linearization

Since we ignore the noise, the pseudo-range equation becomes $y_l = h_l(x),\ l = 1, ..., 4$. This is a nonlinear inverse problem. The first step would be to linearize the pseudo-range equation, *ie* linearize h.

In this first step, we assume that we know a nominal receiver location $\hat{S}$ and clock bias $\hat{b}$, so we linearize the pseudo-range equation about the nominal values. Let $\Delta x := x - \hat{x}$. By doing a Taylor series expansion of h around $\hat{x} = (\hat{S}^T, \hat{b})^T$ we have :

$$y(x) = h(x) = h(\hat{x} + \Delta x) = h(\hat{x}) + \frac{\partial h(\hat{x})}{\partial x}\Delta x + higher\ order\ terms, \tag{5}$$

Let $\Delta y := h(x) - h(\hat{x}) = y - h(\hat{x})$, and $H(\hat{x}) := \frac{\partial h(\hat{x})}{\partial x}$ the jacobian matrix of $h(x)$. Then by getting rid of higher order terms, the pseudo-range equation (3) becomes :

$$\Delta y \approx H(\hat{x})\Delta x \tag{6}$$

which is a linear inverse problem to solve.

Step 2 : Algorithm Development

To compute an accurate GPS position, it is necessary to implement optimization algorithms in the receiver. We will derive two algorithms : the gradient descent (*aka. steepest descent, aka naive*) and the Gauss-Newton (*aka. Gaussian*) gradient descent algorithms. These algorithms are formulated to minimize the loss function :

$$l = \frac{1}{2}||y - h(x)||^2 \tag{7}$$

Step 3: Simulation

In order to test our algorithms, we implement them into Matlab, for it is a powerful development interface to compute, test, and compare our algorithms.

For our initial estimate of the vehicle location, we pick $S0$ to be far from the current position (*ie* about 2330 km off of the actual location).

Since for testing purposes, we only have access to satellite positions, but no timing data, we need to generate the "measured" pseudo-range from the exact location, for later use.

We can then apply our algorithm into Matlab to generate the best estimate of the receiver's location.

Results

Step 1

Recall from equation (5) that the linearization of $y(x)$ depends on the linearization of $h(x)$:

$$H(x) := \frac{\partial h(x)}{\partial x} = \begin{pmatrix} \frac{\partial h_1(x)}{\partial x} \\ | \\ \frac{\partial h_4(x)}{\partial x} \end{pmatrix}, \ \frac{\partial h_l(x)}{\partial x} = (\frac{\partial R_l(S)}{\partial S}, \ \frac{\partial b}{\partial b}) = (\frac{(S - S_l)^T}{R_l(S)}, \ 1)$$

This linearization depends on the linearization of the true range $R(S)$: $\frac{\partial R_l(S)}{\partial S} = \frac{(S-S_l)^T}{R_l}$. This is the unit vector pointing from the l^{th} satellite to the receiver. The higher order terms are all divided by $R_l(S)$. So as the true range increases, the higher order terms tend to 0. Therefore the linearization of the pseudo-range equation (6) becomes more accurate.

Step 2

To minimize the loss function (7) from a current estimation x_k of the optimal solution x_∞, we need to iteratively follow the direction of the negative gradient of l. This gives the direction of the steepest descent.

Gradient Descent

The gradient of l estimated at $\hat{x}$ is given by :

$$\nabla_x l(\hat{x}) = -\frac{\partial h(\hat{x})}{\partial x}(y - h(\hat{x})) = -H^T(\hat{x})(y - h(\hat{x})) \tag{8}$$

An improved estimation x_{k+1} of the current estimate x_k can be computed by taking small step sizes $\alpha_k > 0$ into the direction of $-\nabla_x l(x_k)$. This gives the naive gradient descent algorithm :

$$x_{k+1} = x_k - \alpha_k \nabla_x l(x_k) = x_k + \alpha_k H^T(x_k)(y - h(x_k)) \tag{9}$$

Gauss-Newton Gradient Descent

The Gauss-Newton method also tries to minimize the loss function, but re-linearizes $h(x)$ iteratively about the current estimates x_k. This gives the loss function :

$$l_{gauss}(x_k) = \frac{1}{2}||y - h_{linearized}(x_k)||^2 = l(\Delta x) = \frac{1}{2}||\Delta y - H(x_k)\Delta x||^2 \tag{10}$$

where $\Delta y = y - h(x_k)$, $\Delta x = x - x_k$. The minimum of this loss function provides a least square solution to the linearized inverse problem (6). Finding the correction of the current estimate x_k is equivalent to finding the solution Δx, using the pseudo-inverse H^+ of H:

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

This yields the algorithm :

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

This provides an improved estimate x_{k+1} of the optimal solution x_∞ from our current estimate x_k. We add a constant $\alpha_k > 0$ to take small steps, so to control convergence. Furthermore, in our problem, H is a 4 by 4 matrix of full rank. Therefore H^{-1} exists. This gives the Gauss-Newton gradient descent algorithm :

$$x_{k+1} = x_k + \alpha_k \Delta x = x_k + \alpha_k H^{-1}(x_k)(y - h(x_k)) \tag{11}$$

Step-Size Determination

The choice of step-size is critical for a fast convergence. A constant value $0 < \alpha_k < 1$ proves to be reliable for both algorithms. To determine the best value α, we plotted the convergence speed versus α for both algorithms in figures page 10.

We want to pick the step sizes that optimize the convergence rate. Note that the naive algorithm returns NaN if $\alpha > 0.41$, and the Gaussian algorithm returns NaN if $\alpha < 0.5$. Therefore, it is obvious from the figures to select $\alpha = 0.3$ for the naive algorithm, and $\alpha = 1$ for the Gaussian algorithm.

Termination Criteria

Theoretically, one should iterate an infinite number of times to converge to the exact solution :

$$x_{true} := x_\infty = \lim_{k \to \infty} x_k$$

But for obvious computational reasons, the algorithm should be stopped either once a close enough approximation is found, or whether it takes too many iterations. The algorithms are implemented in ER-units, so by choosing an error margin of $1m$, this corresponds to an error margin of $1.57e - 7ER$.

For such precision, the Gauss-Newton algorithm converges almost instantly, while the naive gradient descent takes thousands of steps. We define the maximum number of iterations to be 300,000 for the naive algorithm, and 1,000 for the Gaussian.

Step 3

In a first phase, we pretend to know the true receiver's position to compute the missing pseudo-range data. We will later use our knowledge of the true position to compare our results.

In a second phase, we pick an estimate of the solution $x0$, and apply the optimization algorithms to converge to the receiver's true position, using the "measured" pseudo-range data.

We plotted in page 11 and page 12 the position prediction error and the bias prediction error, on a log scale. The position prediction error is close to the evaluation of the loss function. The table 1 compares the outputs and runtimes of both algorithms. The steepest descent algorithm takes tens of thousands of iterations to converge within 1m of the true position, while the Gaussian algorithm converges in only 3 steps. Moreover, the naive algorithm has an error in mm, while the Gaussian algorithm has an error in μm. So the Gaussian algorithm has converged closer to the true solution, despite the same termination margin parameter. This is due to a greater α value.

Those results are greatly in favor of the Gauss-Newton algorithm. Indeed, this algorithm converges in over 7,000x less iterations. It is important to note that, however, it does not converge 7,000x faster. The Gauss-Newton iteration steps require more computation that the naive gradient algorithm, because the computation of H^{-1} is very expensive. This makes the convergence of the Gaussian algorithm only 18x faster.

algorithm	α	convergence rate	runtime	position error	bias error
naive gradient descent	0.3	22 328 iterations	2.7 s	1 e-3 m	7.214 e-4 m
Gaussian gradient descent	1	3 iterations	0.15 s	2.873 e-6 m	2.991 e-6 m

Table 1: Optimization algorithms results comparison

Conclusion

The goal was to compute optimization algorithms in order to give an accurate estimate the current position of a common GPS receiver.

When implementing the steepest descent algorithm, or the Gauss-Newton gradient descent algorithm in the absence of noise, we are able to find a very accurate location of the receiver. Even though the Gaussian steps involve more complicated computations, the greatly reduced number of steps for convergence makes it 18x more efficient than the naive gradient descent algorithm.

One should note that the unreasonably good accuracy of the results is only due to the noiseless computation of the synthetic data. In the real world, modeling errors, represented in the noise, greatly affects the results. A way to improve those results is to take multiple measurements from the satellites and find the Maximum Likelihood Estimate (MLE) for x.

Appendices

Step 2

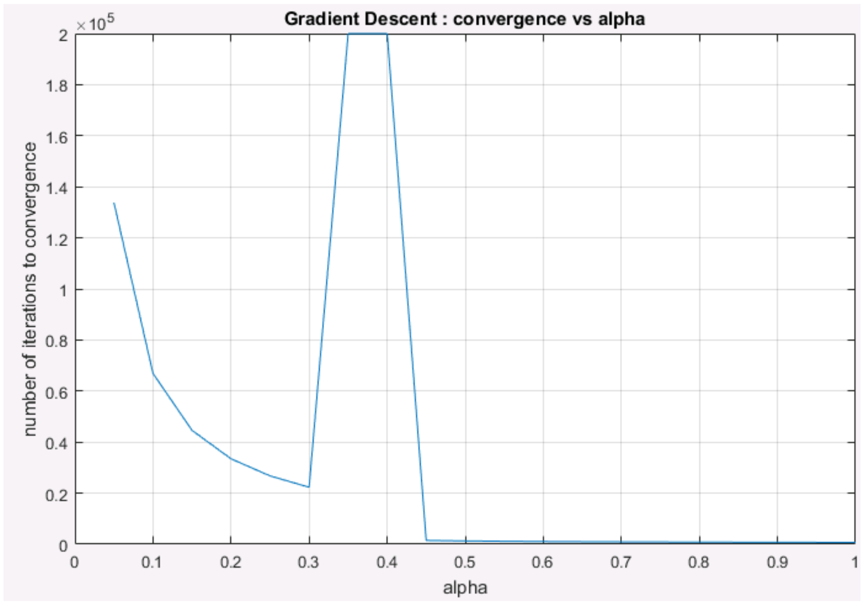

Figure 1: Naive Gradient Descent : convergence vs α
The results of the algorithm are NaN if $\alpha > 0.41$. The best choice is $\alpha = 0.3$.

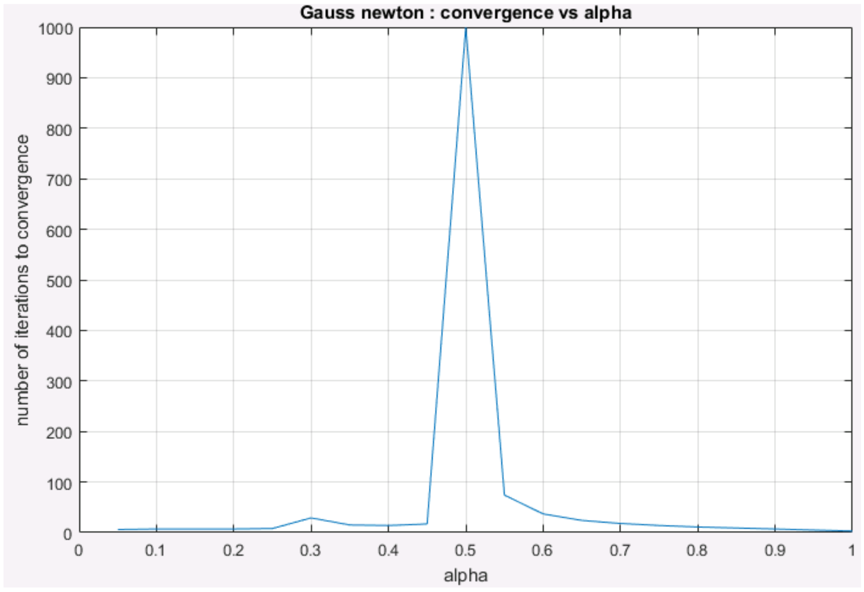

Figure 2: Gaussian Gradient Descent : convergence vs α
The results of the algorithm are NaN if $\alpha < 0.5$. The best choice is $\alpha = 1$.

Step 3

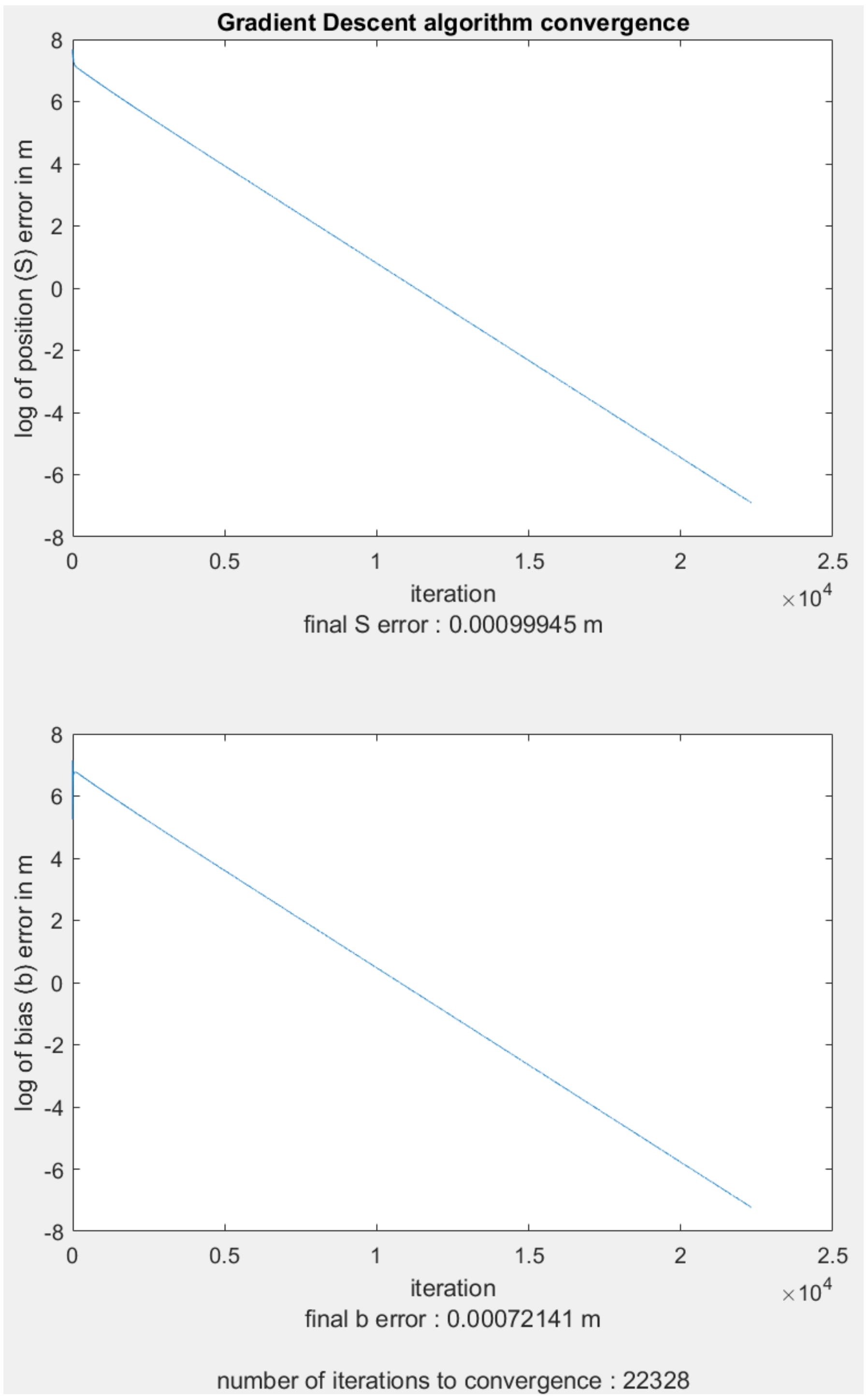

Figure 3: Convergence of the naive gradient descent algorithm on log scale

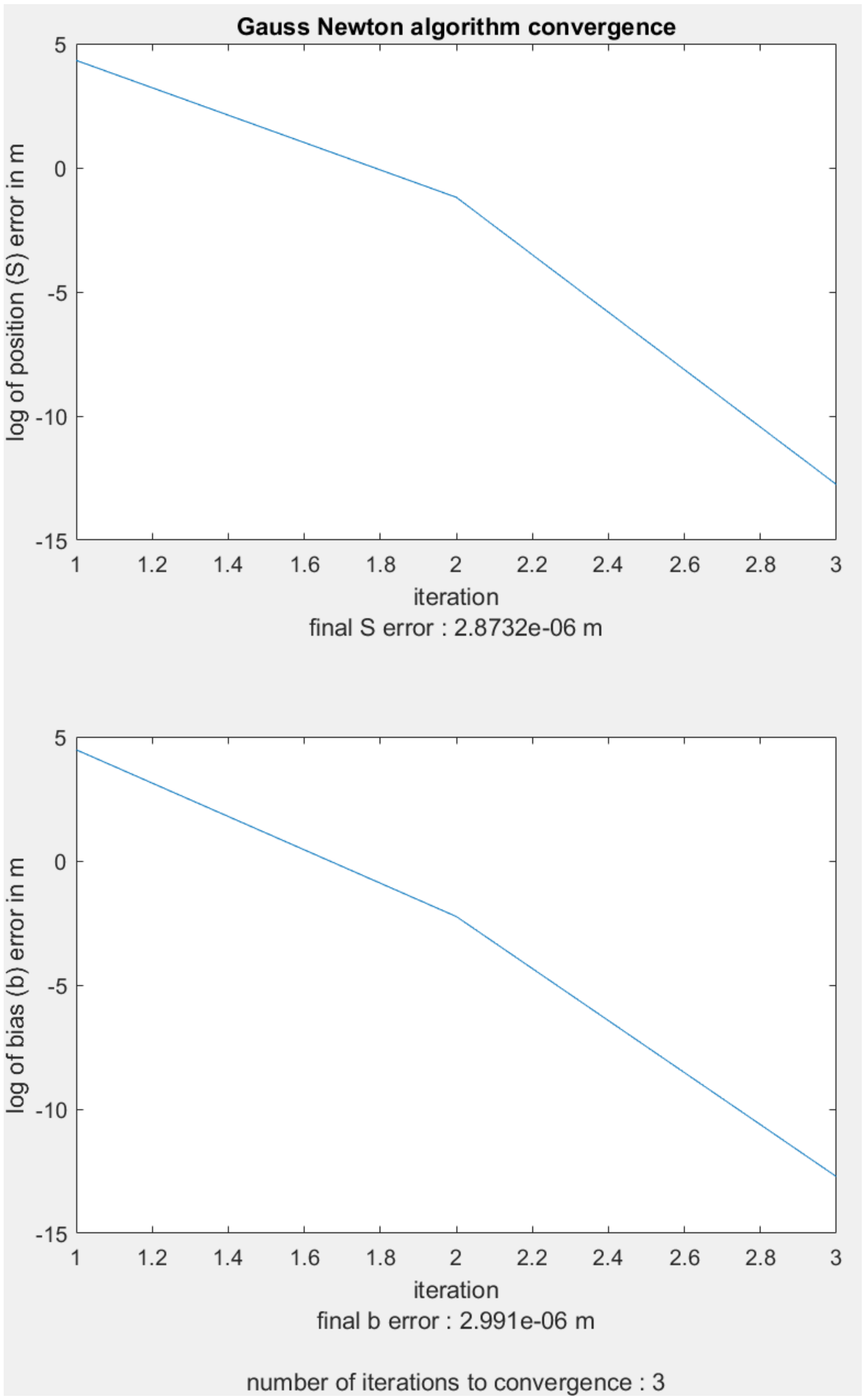

Figure 4: Convergence of the Gaussian gradient descent algorithm on log scale