Estimating Drone Position: A Polynomial Regression Approach

Jessica Monahan
Student ID: 4994809
j.a.monahan@student.tudelft.nl

Lars Boertjes
Student ID: 4704541
l.c.boertjes@student.tudelft.nl

March 4, 2024

1 Introduction

During this assignment we formulated a linear regression problem, solved it using both least squares and gradient descent methods, with our own implementations of both methods.

2 Question 1

1. Before we can formulate a linear regression problem, we need to determine a model that represents our assumed relation between the input variables and the output variables, involving a set of parameters. The assumed relationship is based on what best fits the measured values and can for example be simple, multivariate or polynomial.

2. The optimal solution would be finding the parameters for the model such that the difference between the predicted values and the actual values is minimized. The difference between these values is expressed with the objective function. The optimal solution can be estimated with the ordinary least squares method or with an optimization algorithm as gradient descent.

3 Question 2

3.1 Trajectory of Drone Position

Figure 1 shows the trajectory of the drone plotted with matplotlib.

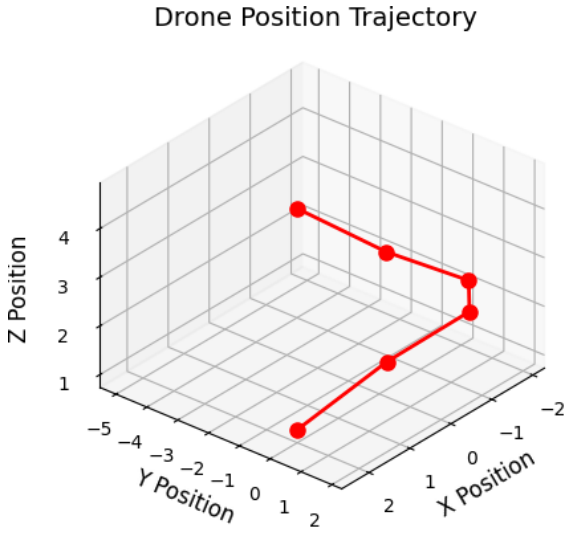

Figure 1: Drone Position Trajectory

3.2 Methodology: Gradient Descent and Ordinary Least Squares

To solve the regression problems the two estimation methods were tested. Models were made for the x, y and z position over time separately.

Ordinary least squares In the code provided, the method of ordinary least squares (OLS) finds the optimal parameters for a polynomial regression model:

$$y = \alpha_0 + \alpha_1 x + \alpha_2 x^2 + ... + \alpha_k x^k$$

The objective of OLS is to minimize the sum of squared errors between the observed and predicted values.

$$E = \sum_{i=1}^{n} \Big(y_i - (\alpha_0 + \alpha_1 x_i + \alpha_2 x_i^2 + \cdots + \alpha_k x_i^k) \Big)^2$$

Using the `least_squares` function, observed input data are processed to determine the parameters of the best-fitting polynomial. The amount of parameters k and therefore degree k-1 in the function can be altered. The function constructs a design matrix $\mathbf{D}$ and then uses the least squares formula

$$\boldsymbol{\alpha} = (\mathbf{D}^\mathsf{T}\mathbf{D})^{-1}\mathbf{D}^\mathsf{T}\mathbf{y}$$

to compute the optimal parameters which will form the polynomial regression function The residuals are found by taking the square root of the sum of squared differences between the predicted and actual values.

Gradient Descent Solver The polynomial regression model and the objective are the same as those of OLS. However, instead of finding a closed-form solution for the regression, optimization is used to find the local minimum. The *gradient_solver* first calculates the predicted y-value using a polynomial regression function based on the polynomial's degree (k - 1). Then, for each parameter α we must calculate:

$$\vec{p}_{n+1} = \vec{p}_n - \eta \nabla f(\vec{p}_n)$$

To start, for each α_j parameter, the partial derivative $\nabla f(\vec{p}_n)$ by following the chain rule is defined as:

$$\frac{\partial f}{\partial \alpha_j} = -2 * \sum_{i=1}^{n} \Big(t^j * (y_i - (\alpha_0 + \alpha_1 x_i + \alpha_2 x_i^2 + \cdots + \alpha_k x_i^k)) \Big)$$

We then determine $\eta \nabla f(\vec{p}_n)$ by multiplying each derivative by the learning rate; this is called the step size. The square of each step size is taken, added together, and then the square root is taken to check if the tolerance is reached. If not, the step size is subtracted from the current value of the parameters until an optimal solution is reached.

3.3 Constant Velocity Model

We first assume the drone moves with constant velocity. Velocity is defined as the rate of change of position with respect to time. Therefore, the first derivative of the drones position with respect to time should be constant. This leads to the following model:

$$\text{Position} = f(t) = a_0 + a_1 * t$$

$$\text{Speed} = f'(t) = a_1$$

The linear regression models, using gradient descent, for the drone's x, y, and z coordinates over time t are as follows:

$x(t) = 1.027 - 0.442t$ with a sum-of-squares error of 8.327 and speed -0.442 m/s

$y(t) = 1.535 - 0.59t$ with a sum-of-squares error of 6.022 and speed -0.590 m/s

$z(t) = 1.282 + 0.483t$ with a sum-of-squares error of 1.631 and speed 0.483 m/s

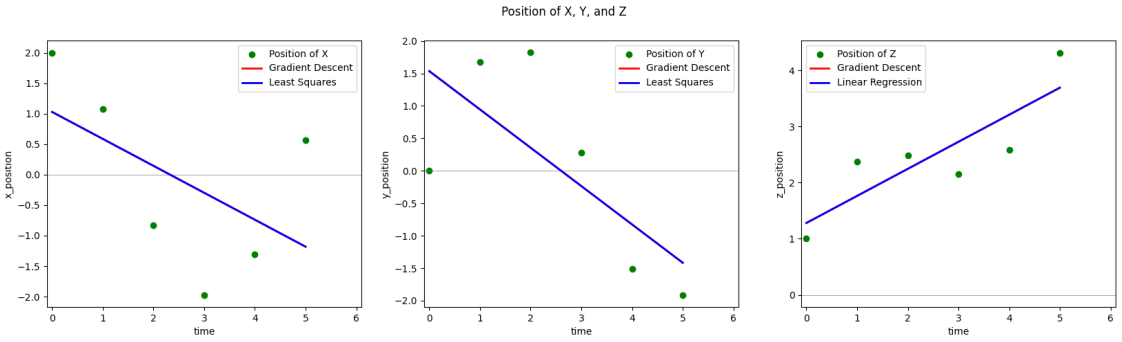

Figure 2: Constant Velocity Regression (lines overlap)

Figure 2 shows the resulting predicted position values for both Gradient Descent and OLS. A regression line was computed using 10,000 iterations and a learning rate of 0.001. The similarity between the Gradient Descent and Linear Regression lines is evident from the overlap in the plot. The differences in sum-of-squares errors seem valid. The data points of x and y are more dispersed compared to the regression line then z.

3.4 Constant Acceleration Model

We next consider the drone moving with constant acceleration. Acceleration is defined as the rate of change of velocity with respect to time. Therefore, the second derivative of the drone position with respect to time should be constant. Thus position over time shall be modeled as following:

$$\text{Position} = f(t) = a_0 + a_1 * t + a_2 * t^2$$

$$\textit{Speed} = f'(t) = a_1 + 2a_2 * t$$

$$\text{Acceleration} = f''(t) = 2a_2$$

The final polynomial regression models, using gradient descent, for each axis are:

$x(t) = 2.473 - 2.61t + 0.434t^2$ residual of 1.309 and acceleration $0.867m/s^2$

$y(t) = 0.457 + 1.028t - 0.324t^2$ residual of 2.113 and acceleration $-0.647m/s^2$

$z(t) = 1.465 + 0.208t + 0.055t^2$ residual of 1.519 and acceleration $0.11m/s^2$

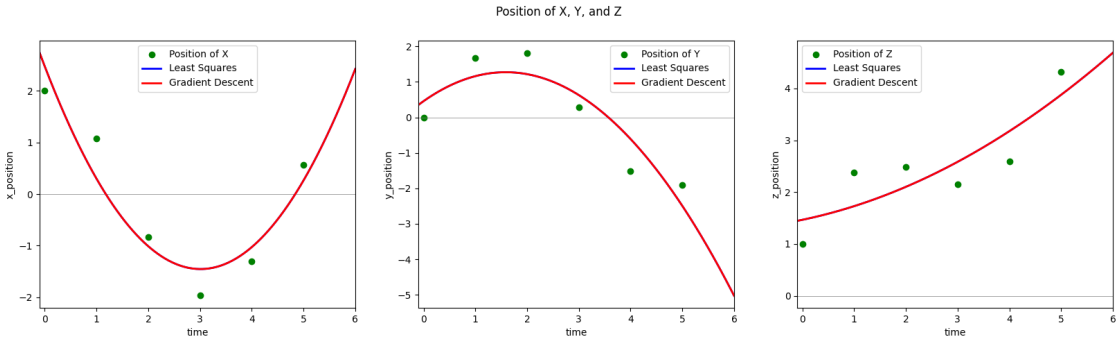

Figure 3: Constant Acceleration Regression max 18,000 iterations (lines overlap)

In Figure 3 both Gradient Descent and Least Squares regression lines are plotted. Due to their similarity they are nearly indistinguishable in the plot, highlighting the effectiveness of gradient descent in closely approximating the least squares solution with sufficient iterations. For reference, Figure 4 shows the regression line using Gradient Descent with only 200 iterations, which is less fitting to the data points.

We have experimented with the parameters within the gradient descent. The final chosen ones are a learning rate of 0.0005 and a maximum of 18,000 iterations. As seen in Figure 5, x converged at 17250 iterations, y at 14205, and z at 15374. We tested higher tolerance values but these produced a slightly less accurate regression line compared to the OLS line.

Based on the constant acceleration model, the drone's predicted position at the next second is $(2.424, -5.026, 4.689)$. The drone's predicted position is plotted in Figure 6.

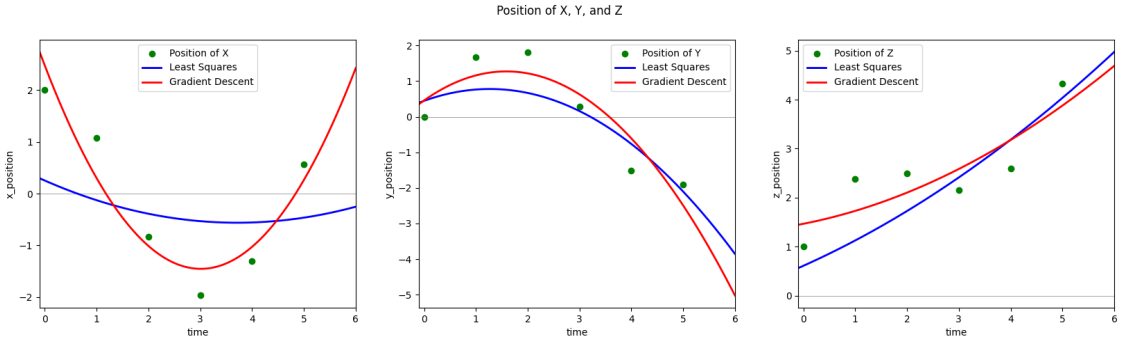

Figure 4: Constant Acceleration Regression 200 iterations

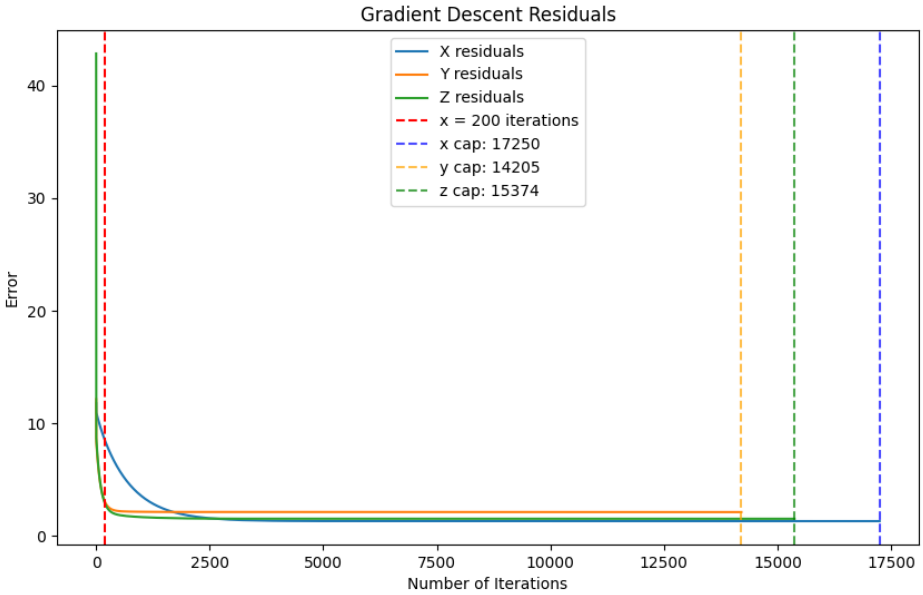

Figure 5: Residuals vs Iterations

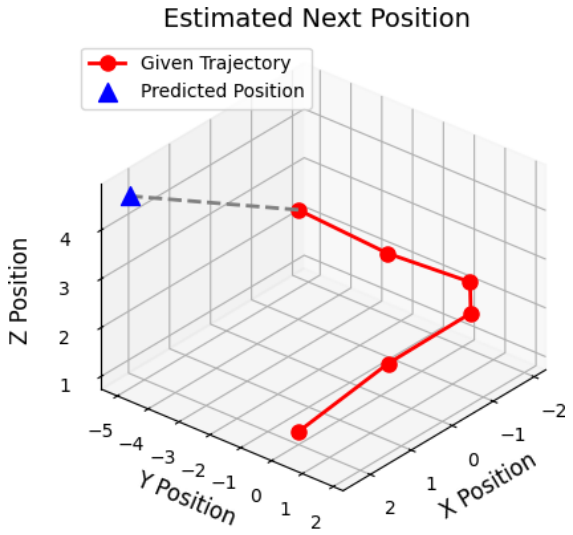

Figure 6: Drone Position Trajectory

4 Conclusion

The total sum-of-squares error for the constant velocity model is *15.98*. For the constant acceleration model this is *4.94*. Therefore we can conclude that the acceleration model produces a more accurate fit for position prediction.

This assignment learned us the importance of choosing the correct linear model, as a great difference in errors can be found. Thereby, creating an own implementation helped us understand linear regression with least squares and gradient descent better. Finally, the influence the values of number of iterations, learning rate and step size on the gradient descent solver has been experimented with.

5 Contribution

The workload was divided evenly. We both tried to solve all the questions, discussed our solutions and then chose the best ones. The report was also written together.

- Lars (50%)
- Jessica (50%)