Linear Least Squares

Leo Jaeeun Kim
University of Michigan
General Motors

ABSTRACT

Linear Least Squares (LLS) is a fundamental optimization technique widely employed in autonomous vehicles, robotics, and machine learning. This method minimizes the sum of squared errors to estimate system parameters, making it crucial for sensor fusion, trajectory estimation, and model fitting. In autonomous driving, LLS aids in state estimation, mapping, and perception by processing noisy sensor data. In robotics, it supports kinematic calibration, motion planning, and localization. Furthermore, machine learning leverages LLS for regression models and dimensionality reduction techniques like Principal Component Analysis (PCA). This paper explores the theoretical foundation of LLS, its computational efficiency, and its real-world applications, highlighting advancements and limitations in these domains.

INTRODUCTION

Autonomous systems and intelligent machines rely on precise mathematical models to interpret their surroundings and make informed decisions. One of the key techniques for parameter estimation in these systems is Linear Least Squares (LLS), a well-established optimization approach that finds applications across multiple engineering and artificial intelligence disciplines.

LLS is particularly valuable in autonomous vehicles, where accurate perception and localization depend on processing large volumes of sensor data from LiDAR, cameras, and radar. For example, sensor fusion techniques often employ LLS to minimize measurement noise and enhance vehicle state estimation. In robotics, the method plays a pivotal role in kinematic calibration, motion trajectory optimization, and SLAM (Simultaneous Localization and Mapping). In machine learning, LLS serves as the foundation for linear regression and forms the basis for dimensionality reduction techniques such as Principal Component Analysis (PCA) and Ridge Regression.

This paper provides a structured analysis of Linear Least Squares, beginning with its mathematical formulation, followed by its applications in autonomous systems, and concluding with its computational challenges and potential improvements. By exploring the role of LLS in modern technological advancements, we aim to highlight its significance in shaping the future of intelligent systems.

METHOD

1) Linear Least Squares Formulation

1.1) Mathematical Foundation

The Linear Least Squares (LLS) method is a fundamental optimization technique used to estimate unknown parameters in an overdetermined system. It is widely applied in engineering.

A typical LLS problem arises when attempting to solve a system of linear equations: $Ax = b$

where $A \in R^{m*n}$ is the design matrix containing known coefficients, $x \in R^n$ is the vector of unknown parameters, and
$b \in R^m$ is the observation vector. Here, m represents the number of equations or observations, while n denotes the number of unknown parameters to be estimated. When the system is overdetermined (m > n), there are more equations than unknowns, meaning an exact solution may not exist. Having more observation is generally beneficial because it provides more data to estimate the unknown parameters more accurately. However, the issue arises because an overdetermined system often has no exact solution - meaning there may be no single xxx that satisfies Ax = b exactly. Instead, the Linear Least Squares (LLS) method determines an approximate solution by minimizing the residual error: $r = b - Ax$

where r represents the difference between observed(b) and estimated values(Ax). Since an exact solution is not possible, LLS chooses xxx in a way that minimizes the sum of squared residuals:$min\left|\left|b - Ax\right|\right|_2^2$ this means with the squared Euclidean norm(Sum of the squares of all residuals), LLS finds the best-fitting x that makes Ax as close as possible to b, reducing the overall discrepancy. For state estimation (e.g., determining a vehicle's position from sensor data), multiple sensors provide measurements with noise. LLS helps by finding the best estimate of the vehicle's position that minimizes the error across all sensors, rather than forcing an exact but inconsistent solution. This approach ensures that even when perfect accuracy isn't possible, the solution is **optimal** given the available data.

1.2) Derivation of the Normal Equation

To obtain the best-fit solution, we differentiate the squared error function with respect to x and set it to zero: $\frac{d}{dx}\left(||b - Ax||_2^2\right) = 0$. Expanding the norm and solving for x, normal equation is derived: $A^T Ax = A^T b$. The solution is then given by $x = (A^T A)^{-1} A^T b$.

When solving a Linear Least Squares (LLS) problem, computational efficiency and numerical stability are important factors, especially when dealing with large datasets or real-time applications. However, directly computing the $A^T A$ can be numerically unstable, particularly when A is ill-conditioned (when a small change in input causes large changes in the output). An ill-conditioned matrix amplifies small numerical errors, leading to unreliable solutions. Instead of computing the inverse explicitly, more stable techniques are used, such as QR decomposition and Singular Value Decomposition (SVD). These methods avoid direct matrix inversion and provide more accurate results by breaking the problem into mathematically well-behaved steps. QR decomposition factorizes A into an orthogonal matrix Q and an upper triangular matrix R, making it easier to solve for x. Similarly, SVD decomposes A into three matrices that allow for better handling of numerical errors, making it ideal for highly ill-conditioned problems.

The standard method for solving LLS using the normal equation involves inverting $A^T A$, which has a computational complexity of $O(n^3)$ (where n is the number of unknown parameters). This becomes inefficient for large systems, such as real-time sensor fusion in autonomous vehicles, where thousands of data points from cameras, LiDAR, and radar must be processed quickly. QR decomposition offers a more stable alternative at a similar computational cost, while SVD is even more robust but requires more computations.

Because of these computational trade-offs, different approaches are used depending on the application. In autonomous driving, where real-time performance is critical, QR decomposition is often preferred for state estimation and sensor fusion. In machine learning, where training datasets are large but speed is not always the main constraint, SVD-based methods are commonly used for regression and dimensionality reduction. In robotics, both QR and SVD methods are applied in tasks like motion planning and kinematic calibration, ensuring numerical precision in control algorithms.

Method	Pros	Cons
Normal Equation	Simple, fast for small data	Unstable for large/ill-conditioned problems
QR Decomposition	More stable than normal equation	Slightly slower
SVD	Most stable handles ill-conditioned cases	Computationally expensive

The example code for QR and SVD can be found in https://github.com/woa0425/Leo_ADAS_script/blob/main/LLS/LLS_python/Linear_Regression.py

2) Application to Linear Regression

Linear regression predicts the value of one variable based on another. The predicted variable is the dependent variable, while the predictor is the independent variable. As a core statistical modeling technique, linear regression applies Linear Least Squares (LLS) to estimate relationships between variables. It provides a simple, interpretable mathematical model that generates predictions by minimizing the sum of squared residuals between observed and predicted values. With a closed-form solution via the normal equation and efficient computational techniques like QR decomposition and Singular Value Decomposition (SVD), linear regression is widely used in machine learning, robotics, and autonomous systems.

Linear regression is a fundamental statistical technique used to model the relationship between a dependent variable y and one or more independent variables x. The general form of a simple linear regression equation is:

$$y = \beta_0 + \beta_1 + \epsilon$$

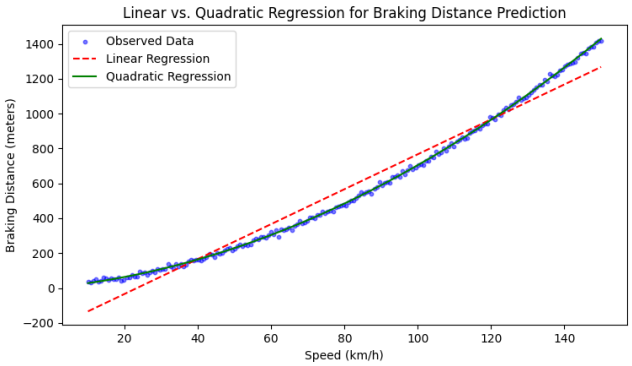
Where β_0 is the intercept, β_1 is the slope of the regression line, and ϵ represents the error term accounting for deviations from the modeled relationship. In multiple linear regression, the equation extends to incorporate multiple independent variables:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n + \epsilon$$

The estimation of the regression coefficients β is typically performed using the Linear Least Squares (LLS) method, which minimizes the sum of squared residuals between the observed and predicted values. This process is expressed mathematically as:

$$min \sum \left(y_i - (\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n)\right)^2$$

The optimal solution can be obtained analytically through the normal equation: $\beta = (X^T X)^{-1} X^T y$, where X is the matrix of input features, y is the vector of observed outputs, and β is the vector of regression coefficients.



The example code for L2 regularization can be found in https://github.com/woa0425/Leo_ADAS_script/blob/main/LLS/LLS_python/QR_SVD_implementation.py

IMPLEMENTATION

Sensor fusion is a critical technique in autonomous systems, robotics, and intelligent transportation, where data from multiple sensors (e.g., LiDAR and Radar) is combined to obtain a more reliable estimate of an observed variable. This implementation utilizes Linear Least Squares (LLS) regression to process fused sensor data and generate a real-time estimated distance.

This approach leverages an expanding window regression model, allowing the algorithm to continuously improve as more data points become available. The method is particularly useful in applications where a reliable distance estimation is required despite sensor noise or uncertainty.

This implementation follows a structured approach to sensor fusion using Linear Least Squares (LLS) regression. The first step involves collecting data from multiple sensors, specifically simulated LiDAR and Radar measurements over 20 time steps. LiDAR provides lower noise (σ = 1), making it more reliable, while Radar introduces higher noise (σ = 2). To enhance accuracy, the sensor readings are fused using a weighted average, where LiDAR contributes 70% and Radar contributes 30%, reflecting their relative reliability in distance estimation.

Once the data is fused, it is processed using an Expanding Window LLS Filter, ensuring that each new observation is added without discarding previous data. This approach improves the accuracy of trend estimation while maintaining regression stability over time. The LLS regression operates by maintaining a growing dataset of sensor readings and their corresponding timestamps. At each step, it computes the best-fit line using the normal equations of least squares, solving for both the slope and offset. The predicted distance for the next time step is then extrapolated using the equation:

$$y_{pred} = offset + (slope * t_{next})$$

This allows the model to forecast the expected distance based on previous observations, effectively smoothing out fluctuations in sensor data. The final step involves prediction and visualization, where the estimated distance is plotted alongside the raw LiDAR, Radar, and Fused Data. The LLS-predicted values (red line) provide a smoothed output, effectively reducing sensor noise and improving the stability of the fusion process—making it particularly useful for applications that require precise distance estimation and robust sensor fusion.

The Expanding Window LLS Algorithm is designed to continuously improve distance estimation by incorporating new sensor data while retaining historical observations. The process begins with model initialization, where an expanding buffer is set up to store sensor readings and corresponding timestamps. A minimum window size is defined, representing the number of samples required before the LLS computation begins. In cases where data is missing, default output values are assigned to maintain continuity in predictions.

The update mechanism ensures that with each new time step, a fused sensor reading is added to the dataset, and a corresponding timestamp is stored for tracking. Once a sufficient number of samples (≥ initial window size) is available, the algorithm proceeds to compute the linear regression coefficients for estimating trends in the data.

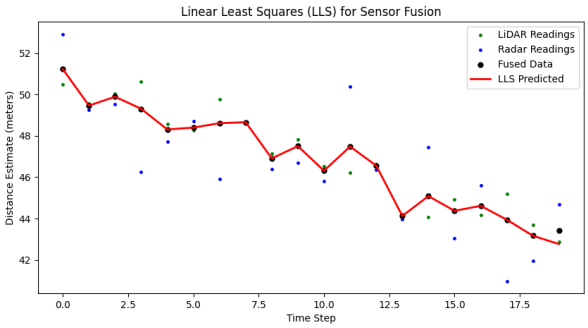
During the LLS regression calculation, the model processes N time samples and computes the necessary statistical sums, including the sum of time values ($\sum t$), the sum of squared time values ($\sum t^2$), the sum of sensor values($\sum u$), and the sum of time-sensor products($\sum t * u$). Using these values, the best-fit line parameters are determined by solving the normal equations:

$$Determinant = (N * \sum t^2) - (\sum t)^2$$
$$Offset = \frac{(\sum t^2 * \sum u) - (\sum t * \sum t * u)}{determinant}$$
$$Slope = \frac{(-\sum t * \sum u) - (N * \sum t * u)}{determinant}$$

Finally, the prediction of the next value is made by extrapolating the best-fit line to estimate the sensor value at the next time step. The key reason for using Linear Least Squares (LLS) regression in sensor fusion, despite the fused data already being an estimate, is to extract

trends and improve stability over time. Even after fusion, sensor data may still contain fluctuations due to varying noise levels from different sensors, and LLS helps by finding a best-fit line, effectively filtering out short-term inconsistencies. Fusion provides an instantaneous best estimate, but it does not predict future values; regression identifies patterns over time and extrapolates future estimates, which is useful for predictive control in autonomous systems. If a sensor fusion algorithm outputs slightly biased or noisy estimates due to transient errors, LLS can provide a more stable output by considering historical data rather than reacting to every fluctuation. The expanding window approach allows the model to incorporate past data, improving its ability to recognize long-term trends and make better estimates as more information becomes available.

This enables the model to provide continuous, real-time predictions based on historical trends while effectively smoothing out sensor noise. The expanding window approach ensures that the algorithm remains adaptable and improves its accuracy over time, making it ideal for sensor fusion applications in autonomous systems and robotics.



The example code for Sensor fusion LLS can be found in https://github.com/woa0425/Leo_ADAS_script/blob/main/LLS/LLS_python/SensorFusionLLS_implementation.py

CONCLUSION

The application of Linear Least Squares (LLS) extends beyond sensor fusion, serving as a foundational technique in various domains of advanced driver-assistance systems (ADAS), robotics, and machine learning. In ADAS, LLS is widely used for vehicle state estimation, trajectory prediction, and sensor calibration, ensuring more accurate and robust decision-making for autonomous navigation. The ability of LLS to process noisy sensor data and extract meaningful trends is crucial for safety-critical applications, such as collision avoidance, lane-keeping assistance, and adaptive cruise control.

In robotics, LLS plays a significant role in motion planning, kinematic calibration, and Simultaneous Localization and Mapping (SLAM). By minimizing error in sensor-based measurements, LLS contributes to more precise localization and improved robotic autonomy, particularly in dynamic environments where sensor inaccuracies can impact real-time decision-making. The computational efficiency of LLS allows its integration into robotic control systems, where rapid updates and predictions are required for smooth and stable motion execution.

Beyond autonomous systems, LLS is a fundamental tool in machine learning, particularly in regression analysis, feature engineering, and dimensionality reduction techniques such as Principal Component Analysis (PCA). The method provides a mathematically sound approach to estimating relationships between variables while maintaining computational efficiency. As datasets grow in complexity and size, optimized LLS solutions, such as QR decomposition and Singular Value Decomposition (SVD), continue to enhance numerical stability and performance.

Despite its effectiveness, LLS has inherent limitations, particularly in nonlinear and highly dynamic systems where relationships between variables cannot be accurately captured by a linear model. Alternative techniques, such as Kalman filters, Bayesian estimation, and deep learning-based predictive models, offer enhanced adaptability in complex real-world scenarios. However, LLS remains an essential building block for parameter estimation and model fitting, providing a strong theoretical foundation upon which more advanced techniques are developed.

Future research can explore hybrid approaches that integrate LLS with adaptive filtering techniques, enabling more resilient and flexible estimation models. Additionally, leveraging machine learning algorithms alongside LLS can help refine sensor fusion and control strategies in autonomous vehicles and robotics. As ADAS and robotic systems continue to evolve, improving estimation techniques like LLS will be essential in achieving higher levels of autonomy, precision, and reliability in intelligent systems.

REFERENCES

APA (7th edition):

Physics Hypertextbook. (n.d.). *Linear regression*. Retrieved from https://physics.info/linear-regression/