

The Linear Algebra Behind GPS

Steven Dindl

December 7, 2025

Background

This paper implements Kalman's (2002) theoretical introduction to the linear algebra behind Global Positioning Systems (GPS). The algebraic computation is reproduced in MATLAB. Then we go over how to manually choose a valid candidate solution from MATLAB's output and demonstrate how real-world noise affects GPS readings using the `gpsSensor` module.

Solutions

This section explains the satellite data we worked with, the MATLAB implementation of Kalman's derivation, and the process we used to interpret the resulting candidate solutions.

Satellite Data

We use the following satellite positions and transmission times from Kalman (2002):

| Satellite | a_i | b_i | c_i | t_i |
|-----------|-------|-------|-------|-------|
| 1 | 1 | 2 | 0 | 11.99 |
| 2 | 2 | 0 | 2 | 8.23 |
| 3 | 1 | 1 | 1 | 33.30 |
| 4 | 2 | 1 | 0 | 10.47 |

Table 1: Satellite positions (a_i, b_i, c_i) and transmission times t_i used in Kalman's model.

These values are substituted into the range equation:

$$(x - a_i)^2 + (y - b_i)^2 + (z - c_i)^2 = (0.047(t - t_i))^2$$

where (x, y, z) is the unknown receiver position and t is the unknown reception time.

MATLAB Implementation

In this subsection I go over the source code for a Kalman (2002) GPS algebraic modeling.

Building the Linear System

```
1 L(i,1) = 2*(a - a1);
2 L(i,2) = 2*(b - b1);
3 L(i,3) = 2*(cp - c1);
4 L(i,4) = -2*(c_light^2)*(ti - t1);
```

This subtracts the satellite 1 equation from satellites 2–4, eliminating quadratic terms and producing a linear 3×4 system.

Parametrizing the Solution

```
1 P = Axyz \ rhs;
2 Q = - (Axyz \ Atcol);
3 xyz = P + Q * t;
```

Solving the 3×3 subsystem lets us express (x, y, z) as a linear function of t .

Enforcing the Unit Sphere

```
1 Acoef = dot(Q,Q);
2 Bcoef = 2 * dot(P,Q);
3 Ccoef = dot(P,P) - 1;
4 tCand = roots([Acoef, Bcoef, Ccoef]);
```

The receiver must lie on $x^2 + y^2 + z^2 = 1$, giving a quadratic in t whose roots yield two algebraic candidates.

Evaluating Residuals

```
1 rvals = ((xyz(1)-satPos(:,1)).^2 ...
2         + (xyz(2)-satPos(:,2)).^2 ...
3         + (xyz(3)-satPos(:,3)).^2) ...
4         - (0.047*(t - tVec)).^2;
```

Each candidate is checked against the original four range equations.

Candidate Selection

The algebraic method produces two candidate solutions. To choose the correct one, we apply this rule:

1. Prefer any candidate where the reception time occurs after all transmission times:
 $t \geq \max_i t_i$.
2. If neither satisfies this (often due to rounding in the given times), select the candidate that best satisfies the original four range equations by comparing residuals.

For our data, neither candidate satisfied the time constraint, so we selected the candidate with smaller residuals.

Results

Running the MATLAB implementation on the satellite data produced two algebraic candidate solutions. The condition number of the 3×3 subsystem, $\text{cond}(A_{xyz}) \approx 12.40$, indicates a well-conditioned system.

| Candidate | t | x | y | z |
|-----------|---------|---------|---------|--------|
| 1 | 19.4041 | -0.7040 | -0.6011 | 0.3783 |
| 2 | 9.6739 | 0.2929 | 0.3631 | 0.8845 |

Table 2: Two candidate solutions from the algebraic solver.

Candidate 1 has a maximum residual of approximately 10.03, while Candidate 2 has a maximum residual of approximately 4.29. Neither candidate satisfies $t \geq \max t_i = 33.30$, which is expected given the rounding in the provided times. Following our selection rule, we choose Candidate 2 based on its smaller residuals:

$$(x, y, z, t) \approx (0.2929, 0.3631, 0.8845, 9.6739)$$

This solution lies on the unit sphere (to numerical precision) and provides the best agreement with all four satellite range equations.

MATLAB 'gpsSensor' Module

While Kalman's algebraic approach assumes perfect range measurements, real GPS receivers must handle random noise in the satellite signals. MATLAB's `gpsSensor` addresses this by modeling position noise as a first-order Gauss-Markov process, where horizontal and vertical position accuracies can be specified independently, and velocity noise is modeled as Gaussian noise.

This layered noise reflects the additional calculations required in practice beyond Kalman's idealized geometric solution.

| Statistic | Altitude (units) |
|-----------|------------------|
| Samples | 20 |
| Mean | 0.232 |
| Std. dev. | 0.277 |
| Min | -0.400 |
| Max | 0.622 |

Table 3: Summary of simulated altitude readings from MATLAB module

Conclusion

Overall, the MATLAB results confirm Kalman's model under the example dataset and the `gpsSensor` showcase explains how real word GPS data can be simulated for more advanced modeling.

References

- Dindl, S. (2025). *MATLAB Algebraic GPS*. GitHub repository. <https://github.com/stevendindl/MATLAB-Algebraic-GPS>
- Kalman, D. (2002). *An Underdetermined Linear System for GPS*. The College Mathematics Journal, 33(5), 384–390. <https://doi.org/10.1080/07468342.2002.11921968>
- MathWorks. (2019). `gpsSensor` (System object) - MATLAB & Simulink. <https://www.mathworks.com/help/nav/ref/gpsensor-system-object.html>