Fast Multiple Fault Exclusion with a Large Number of Measurements

Juan Blanch, Todd Walter, Per Enge. *Stanford University*

In this paper we describe two methods to exclude faulty radio navigation measurements in a situation where there is a large number of measurements (>20) and the probability that any measurement might be faulty is very high (50% and more).

This situation can arise in many applications, as in the case of multiple GNSS measurements in an urban environment, when using terrestrial range measurements (and therefore very prone to be affected by multipath), or when using signals of opportunity (which might have faulty or not calibrated clocks).

Previous work in fault detection for radio navigation has focused in the ability of fault detection methods to detect the outlier measurements [1]. In this work we will first argue that finding the outliers is actually not an issue in the sense that there is a known algorithm that provides an optimal exclusion option (under some assumptions) [3]. This algorithm consists in evaluating all the subsets of measurements for consistency and in choosing the best consistent subset (under some sense). However, this algorithm relies on a combinatorial search which can become intractable as the number of measurements increases. It is therefore useful to develop methods that can achieve the performance of the combinatorial search at a lower computational cost.

In this work, we study two simple and fast exclusion algorithms. The first one is based on a greedy search. In this algorithm the measurements are removed sequentially based on the size of the least squares residual. The second one is based on L_1 norm minimization. For this purpose, we analyze the link between outlier exclusion and sparse signal recovery, for which there is a large body of work (compressed sensing) [5]. In particular, there are rigorous results on the sparsity promoting nature of L_1 norm minimization [6]. Then, we test an implementation of fault detection using both algorithms. We will investigate examples based on two and four core GNSS constellations and attempt to answer the following two questions: how well does each of these algorithms approach the exhaustive algorithm.

INTRODUCTION

In the first part, we will first describe the metric used to measure the consistency of a subset of measurements (the chi-square statistic). Then we will define the optimal algorithm based on this metric, which is the exhaustive search. In the second part, we will describe the exclusion algorithm based on the greedy search. In the third part, we will show how the search for outliers is related to the search for sparse solutions, and its link to L_1 norm minimization, and we will describe an exclusion algorithm based on L_1 norm minimization. Finally, we will test the three algorithms under simulated conditions with a large number of outliers.

EXHAUSTIVE SEARCH ALGORITHM

The exclusion algorithm is used when the set of available measurements appears to be inconsistent. This happens when, depending on which subset of measurements is used, one gets a different enough solution that the probability that it can be explained by nominal errors is low. A measure of the maximum difference between subset position solutions (normalized by the expected standard deviation) is given by the chi-square statistic [7]:

$$\chi^2 = y^T \left(W - WG\left(G^T WG\right)^{-1} G^T W \right) y \qquad (1)$$

In this equation, y is the vector of measurement residuals, W is the weighting matrix (a diagonal matrix whose entries are the inverse of the assumed variances of the measurement error), and G is the geometry matrix. A definition of all these terms in the context of satellite navigation can be found in [7].

The goal of the exclusion algorithm is to remove the measurements that are most likely to be erroneous. A measurement or subset of measurements is likely to be erroneous when the set of remaining measurements appears to be consistent. In fact, the outliers can only be found by finding the consistent subsets.

As we will be characterizing the consistency of the measurements by the chi-square statistic, the exhaustive search algorithm consists on computing the chi-square statistic for all subsets. Once this list is formed, a threshold on the chi-square (for example, determined by a probability of false alert) is chosen. Among this list, we discard all the subsets with a statistic that exceeds the threshold. Of the remaining ones, we choose the one with the largest number of measurements. (If there are several consistent subsets with the same number of measurements we choose the one with the lowest statistic).

As described above the algorithm assumes that we are only considering faults of satellite subsets. It is possible to generalize the above algorithm to all the faults that consist on adding a state to the measurement equation (as in [8]).

Given our chosen metric (the chi-square statistic), the exhaustive algorithm will provide the best performance, in the sense that it will find all the consistent sets of measurements. However, it can be prohibitively expensive in computational load. In this paper we are considering situations were a large number of measurements could be corrupted (up to 50%). For example, if there are 21 measurements and we consider all the possible subsets with 10 or more measurements, then as many as 2^{20}~10^6 subsets may have to be tested. If one considers 30 measurements and all subsets with 15 or more measurements are candidates, then there are on the order of 10^9 subsets to be tested. Although there are techniques to reduce the amount of computation by exploiting the structure of the problem, the exhaustive algorithm becomes impractical with a large number of measurements and a high probability of fault.

GREEDY SEARCH ALGORITHM

The greedy search algorithm consists in removing measurements one by one to lower the chi-square statistic until it meets the consistency threshold. At each step, we exclude the measurement that has the largest impact on the chi-square. As shown in [7], this measurement is the one that has the largest normalized residual. At each step we compute (with the remaining measurements):

$$\tilde{x} = \left(G^T W G\right)^{-1} G^T W y \tag{2}$$

For each measurement i, we compute:

$$r_i = \frac{w_i\left(y_i - g_i^T \tilde{x}\right)^2}{1 - g_i^T w_i \left(G^T W G\right)^{-1} g_i} \tag{3}$$

Then, we remove the measurement with the largest r_i. If the chi-square statistic is below the threshold, the algorithm stops, because a self-consistent set has been found. Otherwise, a new residual is removed by repeating the same process. We stress that a new estimate and a new set of normalized residuals must be computed at each step. The algorithm stops when the remaining set of measurements is self-consistent.

L_1 NORM MINIMIZATION ALGORITHM

In this paragraph we show the link between the search for a sparse solution and the search for a consistent set of measurements (or outlier detection). To show this link, we will first assume that there is no nominal noise. We start by writing the observation equation:

$$y = Gx + \varepsilon \tag{4}$$

In Equation (4), y are the measurements, x is the vector of position and clock states (of length p), and ε is noise on the measurement.

Now we want to find the largest set of consistent measurements. This means that we want to find the position solution $\tilde{x}$ and the largest set of measurements J such that for i in J we have:

$$y_i - G_{i,.}\tilde{x} \simeq 0 \tag{5}$$

This is equivalent to finding the position solution that requires the fewer outliers to explain the measurements. That is, we are trying to solve the optimization problem:

$$\min_x \left|y - Gx\right|_{L_0} \tag{6}$$

The notation $\left|\bullet\right|_{L_0}$ denotes the number of non-zero coordinates. We can rewrite this problem as:

$$\min_{x,\varepsilon\in R^n} \left|\varepsilon\right|_{L_0} \text{ subject to } y = Gx + \varepsilon \tag{7}$$

In the lines below we draw from reference [5]. The idea of Compressed Sensing is to solve the problem:

$$\min_{X\in R^n} \left|X\right|_{L_0} \text{ subject to } y = AX \tag{8}$$

In this equation, the norm L_0 is simply the amount of non-zero coordinates of x. That is, the idea of this approach is to find the simplest explanation of the data. It is very important to see that it is not necessary that the system is overdetermined. The solution corresponds to the sparsest x explaining the data.

Now, because the above problem is in general very hard to solve (because it is combinatorial in nature), it can be replaced by the convex problem:

$$\min_{X\in R^n} |X|_{L_1} \text{ subject to } y = AX \tag{9}$$

The only change between Problems (8) and (9) is the objective function, which is now the L_1 norm. The L_1 norm is defined as:

$$|x|_{L_1} = \sum |x_i| \tag{10}$$

Problem (9) can be cast as a linear program (see Appendix) that can be solved efficiently with widely available software tools. It turns out that solving this relaxation often solves the original problem, and recent work provides rigorous proof that these problems have a high probability of being equivalent [6]. This is very useful, as it replaces an unfeasible problem with one that can be treated in a guaranteed time (because of its convexity).

Similarly, we replace Problem (7) by its L_1 relaxation:

$$\min_{x,\varepsilon\in R^n} |\varepsilon|_{L_1} \text{ subject to } y = Gx + \varepsilon \tag{11}$$

Now, to account for the fact that there is nominal noise in the measurements, we can weigh these residuals and write the problem as:

$$\min_x \sum_{i=1}^{n} \frac{|y_i - g_i^T x|}{\sigma_i} \tag{12}$$

In this equation σ_i is the standard deviation of the nominal noise expected on measurement i. It can be useful here to compare this problem to the problem that least squares solves, which is given by:

$$\min_x \sum_{i=1}^{n} \left(\frac{|y_i - g_i^T x|}{\sigma_i} \right)^2 \tag{13}$$

The solution of the Problem (12) does not yield directly the outliers. The idea is now to order the following residuals in descending order:

$$\frac{|y_i - g_i^T X|}{\sigma_i} \tag{14}$$

We now have an order in which to exclude the measurements if the chi-square test does not pass. Instead of having to test billions of subsets, we only have to test n-p+1.

We should note that the conditions under which the search of the sparsest solution coincides with the L_1 minimization do not hold in our application in at least two points. First, there is nominal noise; second, the matrix A has a very specific structure.

PERFORMANCE EVALUATION

The goal of the algorithm described above is to find a self-consistent set of measurements among a set of measurements which appears to be inconsistent. It is important to stress that we are not testing the ability of the algorithm to find the actual outliers. Instead, we are evaluating how it compares to the exhaustive search algorithm. In addition, we are not testing the algorithm under worst case outliers, that is, outliers which have a non-negligible probability of remaining undetected. Outliers are modeled as random errors with a significantly larger standard deviation.

We now examine some simulation results corresponding to a multi-constellation GNSS geometry corresponding to the G matrix included in the Appendix. A nominal noise of 1 m is assumed in each measurement. The threshold for self-consistency was given by setting false alert probability P_{fa} to 10^{-4}, that is, for n measurements and p states, we set:

$$T_n = \chi^2_{n-p}\left(1 - P_{fa}\right) \tag{15}$$

We simulated the outliers as Gaussian errors with a standard deviation of 10 m and 20 m instead of 1 m.

Each simulation consisted on:

- simulating the nominal noise
- choosing N_{outliers} randomly
- simulating the bias in each outlier

For each number of outliers, 1000 simulations were performed. The three algorithms were applied to each simulation.

Two core constellation results

In the first set of results we only considered the two core constellations in G, that is, the first 19 measurements. Figure 1 shows the standard deviation of the position error for each number of outliers for each of the algorithms. It also shows the standard deviation of the position error when no exclusion is applied.

We can see that, in this simulation, the L1 minimization algorithm matches the performance of the exhaustive algorithm for all numbers of outliers. This suggests that it is likely that the exhaustive algorithm can be replaced by the L_1 minimization algorithm without loss of performance (at least when considering the standard deviation of the position as a metric).

Figure 1 also shows that the greedy search matches the performance of the exhaustive algorithm up to a maximum number of outliers of four, and the degradation beyond four is slow (up to 30% for eight outliers). This is useful, because of the three algorithms, the greedy search is by far the simplest one.

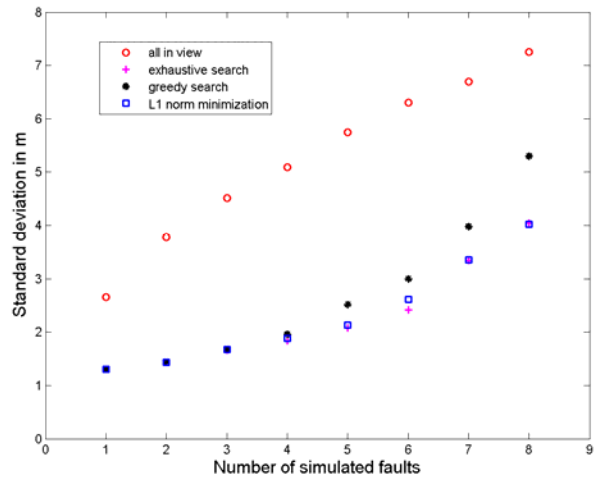

Figure 1. *Standard deviation of the position error for each algorithm in the two core constellation example.*

Four core constellation results

In the four core constellation case (Figure 2 and 3), it was not possible to run the exhaustive algorithm for more than six simulated faults (as it became too time consuming). The trends are similar to the previous example: the L_1 minimization algorithm matches the exhaustive algorithm (at least for the values where the comparison was possible), the greedy search algorithm performs as well up to a large number of faults, and the difference in performance between the L_1 minimization and greedy search grows slowly beyond that point.

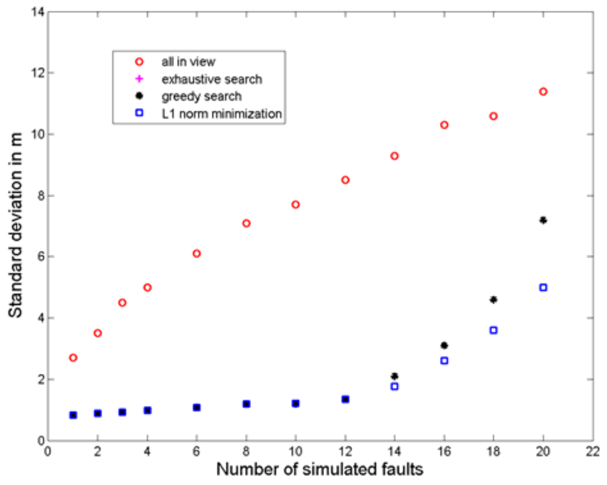

Figure 2. *Standard deviation of the position error for each algorithm in the four core constellation example (and outliers 10 times worse than nominal).*

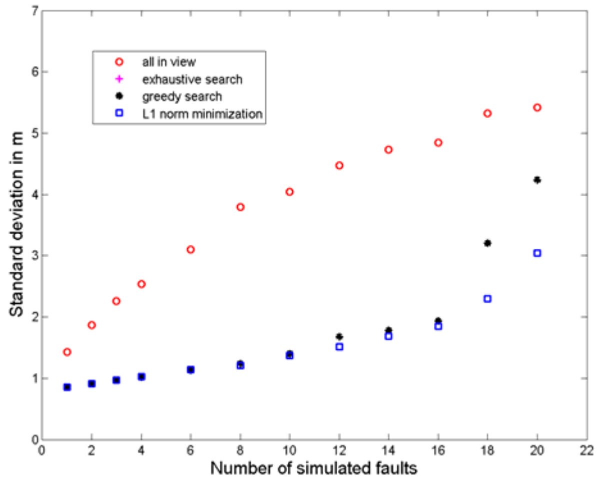

Figure 3. *Standard deviation of the position error for each algorithm in the four core constellation example (and outliers 20 times worse than nominal).*

In Table 1, we have compared the speed of each algorithm for one position fix in the four core constellation example. As expected, the runtime of the exhaustive algorithm increases exponentially with the number of faults. Both the L_1 minimization algorithm and the greedy search are much faster than the exhaustive algorithm (even for only two faults), and the runtime is not very sensitive to the number of faults. It is interesting, that the L_1 minimization, which requires a linear program solution, has a runtime comparable to the greedy search.

Number of faults	Exhaustive	L1 minimization	Greedy search
2	0.03 s	0.010 s	0.007 s
4	2.8 s	0.013 s	0.008 s
6	121 s	0.012 s	0.008 s
8	-	0.017 s	0.011 s
10	-	0.019 s	0.011 s
12	-	0.019 s	0.008 s
14	-	0.013 s	0.007 s
16	-	0.015 s	0.009 s

Table 1. *Speed comparison in the four core constellation case.*

SUMMARY

Fault exclusion consists in finding subsets of measurements that appear to be self-consistent. If one assumes that all faults have a similar probability of occurring and are not correlated, an adequate strategy (and probably optimal one) is to find the largest subset that appears to be consistent. An exhaustive search algorithm will therefore always find the largest self-consistent subset. However, the exhaustive search algorithm becomes impractical when the number of faults is large. We have defined and studied two simple and fast algorithms and compared their performance to the exhaustive search algorithm in two example geometries. The algorithm based on L_1 norm minimization matches closely the performance of the exhaustive algorithm for all the situations where it could be tested. The greedy search algorithm matched the performance of the exhaustive algorithm for a small number of faults, but its performance degraded slightly when more faults were present.

REFERENCES

[1] Nathan L. Knight , Jinling Wang. "A Comparison of Outlier Detection Procedures and Robust Estimation Methods in GPS Positioning," *Journal of Navigation*, Volume 62, Issue 04, October 2009, pp 699-709.

[2] M. Yetkin, C. Inal. "L 1 Norm Minimization in GPS Networks." *Survey Review*, Volume 43, Issue 323, 01 October 2011, pp. 523-532.

[3] Carlone, A. Censi, F. Dellaert, "Selecting good measurements via ℓ_1 relaxation: a convex approach for robust estimation over graphs", *Int. Conf. on Intelligent Robots and Systems* (IROS), accepted, 2014.

[4] Peter J. Huber, "The place of the L1-norm in robust estimation," *Computational Statistics & Data Analysis*, 1987, vol. 5, issue 4, pages 255-262.

[5] Candès, E.J., & Wakin, M.B., *An Introduction To Compressive Sampling*, IEEE Signal Processing Magazine, V.21, March 2008

[6] David L. Donoho. "For most large underdetermined systems of linear equations, the minimal l_1 solution is also the sparsest solution." Comm. Pure Appl. Math., 59(7):907–934, July 2006.

[7] Blanch, J., Walter, T., Enge, P., Lee, Y., Pervan, B., Rippl, M., Spletter, A., "Advanced RAIM user Algorithm Description: Integrity Support Message Processing, Fault Detection, Exclusion, and Protection Level Calculation," *Proceedings of the 25th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS 2012)*, Nashville, TN, September 2012.

[8] Blanch, J., Walter, T., and Enge, P.," Optimal Positioning for Advanced RAIM," *NAVIGATION*, Vol. 60, No. 4, Winter 2013, pp. 279-290.

APPENDIX

Example geometry

$$G = \left[\begin{array}{ccccccc}
-0.690 & -0.062 & -0.720 & 1 & 0 & 0 & 0 \\
0.197 & 0.932 & -0.302 & 1 & 0 & 0 & 0 \\
-0.866 & -0.450 & -0.212 & 1 & 0 & 0 & 0 \\
-0.325 & 0.780 & -0.534 & 1 & 0 & 0 & 0 \\
0.668 & 0.290 & -0.684 & 1 & 0 & 0 & 0 \\
-0.090 & -0.821 & -0.563 & 1 & 0 & 0 & 0 \\
-0.508 & -0.849 & -0.143 & 1 & 0 & 0 & 0 \\
0.387 & -0.645 & -0.657 & 1 & 0 & 0 & 0 \\
-0.987 & -0.074 & -0.137 & 0 & 1 & 0 & 0 \\
-0.680 & 0.446 & -0.581 & 0 & 1 & 0 & 0 \\
0.045 & 0.774 & -0.631 & 0 & 1 & 0 & 0 \\
0.696 & 0.675 & -0.241 & 0 & 1 & 0 & 0 \\
0.507 & 0.827 & -0.241 & 0 & 1 & 0 & 0 \\
0.683 & 0.230 & -0.692 & 0 & 1 & 0 & 0 \\
0.486 & -0.555 & -0.674 & 0 & 1 & 0 & 0 \\
0.057 & -0.977 & -0.204 & 0 & 1 & 0 & 0 \\
0.243 & -0.787 & -0.566 & 0 & 1 & 0 & 0 \\
-0.444 & -0.423 & -0.789 & 0 & 1 & 0 & 0 \\
-0.871 & 0.165 & -0.461 & 0 & 1 & 0 & 0 \\
-0.786 & 0.366 & -0.497 & 0 & 0 & 1 & 0 \\
-0.832 & 0.266 & -0.486 & 0 & 0 & 1 & 0 \\
-0.017 & 0.637 & -0.770 & 0 & 0 & 1 & 0 \\
-0.299 & -0.355 & -0.885 & 0 & 0 & 1 & 0 \\
0.804 & 0.525 & -0.278 & 0 & 0 & 1 & 0
\end{array} \right.$$

```
  0.378   -0.466   -0.799        0   0   1   0
 -0.105   -0.973   -0.201        0   0   1   0
  0.396   -0.809   -0.432        0   0   1   0
  0.601    0.404   -0.688        0   0   1   0
  0.951   -0.232   -0.200        0   0   0   1
  0.452   -0.548   -0.703        0   0   0   1
 -0.376   -0.579   -0.722        0   0   0   1
 -0.921   -0.302   -0.244        0   0   0   1
 -0.207    0.858   -0.468        0   0   0   1
 -0.781   -0.483   -0.393        0   0   0   1
 -0.687    0.252   -0.680        0   0   0   1
 -0.000    0.946   -0.322        0   0   0   1
  0.550    0.502   -0.667        0   0   0   1
  0.814   -0.247   -0.525        0   0   0   1];
```

Casting L1 minimization as a linear program

The L_1 minimization problem is given by:

$$\min_x |y - Gx| \tag{16}$$

This is not a linear program, but it can be turned into one by introducing the vectors t^+ and t^- :

$$\begin{aligned} &\text{min} \quad \begin{bmatrix}1 & \cdots & 1\end{bmatrix} t^+ + \begin{bmatrix}1 & \cdots & 1\end{bmatrix} t^- \\ &\text{such that } y - Gx = t^+ - t^- \\ &\text{and} \qquad t^+ \geq 0, \;\; t^- \geq 0 \end{aligned} \tag{17}$$