Using S-estimators in Parameter Identification

Tiago Milhano
IST/ISR.
Email: tiagomilhano@ist.utl.pt

João Sequeira
IST/ISR
Email: joao.silva.sequeira@ist.utl.pt

Emanuele Di Sotto
GMV
Email: edisotto@gmv.com

Abstract—This paper describes an improved S-estimator using the fast-S formulation and the standard Least Squares identification, aiming at realtime usage in a parameter identification setting. The improvement step uses a recursive version of the Least Squares algorithm. Experiments with simulated and real data are presented. The new algorithm is shown to have the outlier rejection properties of state of the art S-estimators while used in the identification context.

Index Terms—S-estimator, Outlier Rejection, Least Squares, Parameter Identification.

I. INTRODUCTION

This paper describes an hybrid estimation strategy for parameter identification that mixes S-estimation and Least Squares identification.

Parameter identification techniques under perfect information rely on regression techniques, i.e., using optimization techniques to fit a dataset to an a priori defined model. In general, this is a relatively straightforward optimization problem (see for instance the Least Squares, LS, regression for linear models). However, corrupted datasets may generate large deviations in the estimated parameters and naturally raise the issues of (i) how to filter the input dataset, and (ii) how to compute the best parameter estimates from a filtered dataset.

Structured noise can be handled by filtering strategies, e.g., Kalman filtering (the literature on such applications is extensive, see for instance [1]). In realistic conditions, non-Gaussian disturbances, namely unstructured disturbances (also called outliers), are often present and it is well known that their effect on LS estimates can be strong, [2]. Unstructured noise, such as that introduced by outliers, is often better handled by specific techniques. For example, if data covariance is known a priori then a simple threshold rule based on the Mahalanobis distance can be used to detect outliers (see [3]).

S-estimators represent an alternative strategy to detect and remove the effect of outliers, namely by estimating first and second order statistics through outlier robust methods. Robustness to disturbances introduced by outliers is the amount of disturbances a dataset can contain without having the error growing above a certain level, [4]. Formally, robustness has been defined in terms of the breakdown point of the estimator, that is, the fraction of outlying data that corrupts the estimator.

M-estimators and S-estimators have been proposed by multiple authors, in single and multivariate form, as estimators that are robust to outliers (see [5] for an introduction to the formal structure of M and S estimators). M-estimators are claimed to have breakdown points around 30% in linear regression examples (see [2]). S-estimators have been shown to achieve asymptotically the same variances than the M-estimators with higher breakdown points, [5], asymptotically at 50% (see section II ahead).

The key issues in parameter identification are similar in many applications. In some cases, such as in-flight estimation for spacecrafts, the realtime constraint has to be accounted for. The intrinsic computational complexity of the S-estimators has been referred as a major drawback for realtime scenarios. Still, a multivariate S-estimator in standard formulation (see [5]) has been applied to a mapping problem using two solution strategies, namely that proposed in [6], and a solution using a standard optimization formulation, [7]. The differences between the two approaches are small and the results suggest that both can be used in real applications and hence serve as inspiration for the hybrid estimator described in this paper.

The paper is organized as follows. Section II describes the main concepts behind the classical S-estimator in univariate formulation. The realtime variant introduced in the paper is presented in Section III. Section IV discusses the use of the estimator presented in the paper to data from simulation and real applications. Section V concludes the paper with a brief discussion on future evolution for this S-estimator.

II. THE S-ESTIMATOR

A. Breakdown Point

An estimator can be characterized by its breakdown point, or BDP for short. Consider a sample $\mathcal{X}$ of data points $(x_i, y_i), i = 1, \ldots, n$ and any estimator $\mathcal{T}$ of the parameter vector $\theta$. Let $\beta$ be a quantity such that

$$\beta (m, \mathcal{T}, \mathcal{X}) = \sup \| \mathcal{T} (\mathcal{X'}) - \mathcal{T} (\mathcal{X}) \| ,$$

where $\mathcal{X'}$ represents all the samples of which any $m$ of the original points of $\mathcal{X}$ are replaced by arbitrary values representing the outliers. Then the breakdown point of $\mathcal{T}$ is defined as

$$\varepsilon_n^* (\mathcal{T}, \mathcal{X}) = \min \left\{ \frac{m}{n} ; \beta (m, \mathcal{T}, \mathcal{X}) \text{ is infinite} \right\} .$$

Emanuele Di Sotto
GMV
Email: edisotto@gmv.com
The BDP, \( \varepsilon^*_n \), is thus the smallest fraction of outlying or corrupted data that can cause the difference between the estimator values, using pure and contaminated data, to be arbitrarily large, [4]. Considering the least squares estimator, one corrupted observation is enough to cause breakdown, so \( \varepsilon^*_n \) (LS, \( \mathcal{X}_n^* \)) = 1/n which tends to 0 as the sample size \( n \) increases.

B. The univariate S-estimator

Consider a sample of regression data, \((\phi_i, y_i)\), and \(\theta\), a \(p\)-dimensional vector of parameters to estimate. A regression model can be defined as

\[
y_i = \phi_i^T \theta + e_i , \quad i = 1, ..., n ,
\]

where \(e_i\) is the observation error. Given an estimate of \(\theta\), denominated \(\hat{\theta}\), the residues of the estimate are defined as \(r_i = y_i - \phi_i^T \hat{\theta}\). When considering the Least Squares (LS) estimator, \(\hat{\theta}\) is the solution that minimizes the functional

\[
\mathcal{J} = \sum_{i=1}^{n} r_i^2 ,
\]

or some weighted variation. All the observations have influence on the cost. If the observation error, \(e_i\), is simply Gaussian white noise, the residues will also follow a Gaussian distribution and the LS estimate will be optimal.

However, in some cases, there are some observations that do not correspond to reality. They are called outliers and can be of two types: the measurement \(y_i\) does correspond to reality, denoted as vertical outlier; and the values in \(\phi_i\) do not correspond to reality these being called leverage points

(see Section 6 of [8] on outlier categories).

To begin detailing the S-estimator, consider a function, \(\rho\), with the following characteristics

C1: \( \rho \) is symmetric, \( C^\infty \), and \( \rho(0) = 0 \); 
C2: there exists \( c > 0 \) such that \( \rho \) is strictly increasing on \([0, c]\) and constant on \([c, \infty[\).

A common \(\rho\) function is (see [4])

\[
\rho(u) = \begin{cases} 
\frac{u^2}{2} - \frac{u^4}{2c^2} + \frac{u^6}{6c^4} & \text{for } |u| \leq c \\
\frac{c^2}{6} & \text{for } |u| > c
\end{cases},
\]

the derivative of which is the Tukey’s biweight function

\[
\psi(u) = \begin{cases} 
u \left(1 - \left(\frac{|u|}{c}\right)^2\right)^2 & \text{for } |u| \leq c \\
0 & \text{for } |u| > c
\end{cases} .
\]

Now consider a random variable \(G\), following a normal distribution (\(G \sim N(0, 1)\)), and another, \(H\), as \(H = \rho(G)\) with expected value

\[
K = E[H],
\]

By considering a large enough number of observations of \(H\), denominated \(h_i\), the average of these observations should approximate the expected value \(K\) (strong law of the large numbers)

\[
1 \frac{1}{n} \sum_{i=1}^{n} h_i = \frac{1}{n} \sum_{i=1}^{n} \rho(g_i) \approx K ,
\]

where \(g_i\) are observations of the random variable \(G\). Expression (3) can be transformed to use the residues of a given estimate \(\hat{\theta}\) for a set of observations as

\[
1 \frac{1}{n} \sum_{i=1}^{n} \rho \left( \frac{r_i}{s} \right) = K .
\]

The parameter \(s \in \mathbb{R}\) scales the residues of the observations. Intuitively, the residues must be scaled in order to fit the standard normal distribution, such that (4) holds. Smaller \(s\) indicates that the residues, for the given observations and estimate \(\hat{\theta}\), are small and have small variance, and vice-versa. In the presence of outliers, some residues will have a larger absolute value, leading by (4), to a larger scaling. Depending on the choice of the function \(\rho\), equation (4) can have multiple scaling solutions or none. In the first case, one must consider \(s\) equal to the supremum of the set of solutions; with respect to the second case, one consider \(s = 0\) (from [4]). However, when considering the S-estimator problem with the \(\rho\) function has in (1), there are only two solutions and they are symmetric of each other. Therefore, it is only necessary to find the positive solution.

The scaling \(s\) is the baseline for the class of the S-estimators, hence the name. The S-estimator, for a scalar \(s\), can then be defined (as in [4]) as the solution to the optimization problem

\[
\min_{\hat{\theta}} s \left( r_1 \left( \hat{\theta} \right) , ..., r_n \left( \hat{\theta} \right) \right) \\
\text{s.t.} \quad \frac{1}{n} \sum_{i=1}^{n} \rho \left( \frac{r_i}{s} \right) = K .
\]

C. Breakdown Point of S-estimators

Consider a function \(\rho\) for which conditions C1 and C2 hold, and also

C3: \( \frac{K}{\rho(c)} = \frac{1}{2} . \)

With \(\rho\) as in (1), condition C3 is achieved by taking \(c \approx 1.547\) and resulting in \(K \approx 0.1995\). Now, consider Lemma 1 from [4].

Lemma 1. For each \(\rho\)-function satisfying conditions C1 to C3 and for each \(n\) number of samples, there exists positives constants, \(\alpha\) and \(\beta\), such that the scaling \(s\) given by (4) satisfies

\[
\alpha \text{ median} \left( |r_i| \right) \leq s (r_i) \leq \beta \text{ median} \left( |r_i| \right) , \quad i = 1, ..., n\]

Assuming that all the samples with \(\phi_i = 0_{p \times 1}\) have been deleted from the sample data set (they provide no information on \(\theta\)) and that no more than half the points
\((\phi_1, y_i)\) of the sample should lie on a vertical subspace, one has Lemma 2.

**Lemma 2.** For any \(\rho\)-function satisfying conditions \(C1\) to \(C3\), there always exists a solution to (5).

The proof for both Lemmas can be found in [4].

Consider that, in general, any \(p\) observations determine a single \(\theta\). For example, for a simple regression \((p = 2)\), this means that no two points may coincide or determine a vertical plane.

The BDP of the S-estimators is then given by Theorem 1 of [4]

**Theorem 1.** An S-estimator constructed from a function \(\rho\) satisfying conditions \(C1\) to \(C3\) has a breakdown point

\[
\varepsilon^*_n = \frac{(n/2) - p + 2}{n}
\]

at any sample \(\{(\phi_1, y_1), \ldots, (\phi_n, y_n)\}\) in general position.

The BDP depends weakly on \(n\), and \(\varepsilon^* \to 50\%\) as \(n \to \infty\). For an arbitrary data set with more than 50% of outliers it becomes impossible to discriminate good observations from polluted data.

The above lemmas and theorem guarantee that the S-estimator problem of (5), given the conditions \(C1\) to \(C3\), has a solution. Furthermore, if there is a priori knowledge of the data set, it is possible to tune the S-estimator to be more efficient at the cost of BDP. For example, replacing condition \(C3\) by \(K/\rho(c) = \gamma\) where \(0 < \gamma < 1/2\), then the corresponding S-estimator has a BDP tending to \(\varepsilon^* = \gamma\) for \(n \to \infty\). In the case of a \(\rho\) as in (1), values of \(c\) larger than 1.547 yield S-estimators with better asymptotic efficiencies at a Gaussian central model, but smaller BDP.

**D. An Example**

<table>
<thead>
<tr>
<th>(x)</th>
<th>(y)</th>
<th>(x)</th>
<th>(y)</th>
<th>(x)</th>
<th>(y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.44</td>
<td>9</td>
<td>6.53</td>
<td>17</td>
<td>6.63(*)</td>
</tr>
<tr>
<td>2</td>
<td>2.96</td>
<td>10</td>
<td>7.04</td>
<td>18</td>
<td>5.93(*)</td>
</tr>
<tr>
<td>3</td>
<td>3.42</td>
<td>11</td>
<td>7.47</td>
<td>19</td>
<td>5.36(*)</td>
</tr>
<tr>
<td>4</td>
<td>4.08</td>
<td>12</td>
<td>8.15</td>
<td>20</td>
<td>5.08(*)</td>
</tr>
<tr>
<td>5</td>
<td>4.51</td>
<td>13</td>
<td>8.53</td>
<td>21</td>
<td>4.94(*)</td>
</tr>
<tr>
<td>6</td>
<td>4.96</td>
<td>14</td>
<td>9.93(*)</td>
<td>22</td>
<td>13.1</td>
</tr>
<tr>
<td>7</td>
<td>5.51</td>
<td>15</td>
<td>8.44(*)</td>
<td>23</td>
<td>13.3</td>
</tr>
<tr>
<td>8</td>
<td>5.92</td>
<td>16</td>
<td>7.50(*)</td>
<td>24</td>
<td>14.0</td>
</tr>
</tbody>
</table>

This section presents a simple example illustrating the difference in robustness to outliers of the Least Squares and S-estimators. Consider the points in Table I. The goal is to find the linear relation between \(x\) and \(y\), such as

\[
y = \beta_1 x + \beta_2 .
\]

and the goal is to estimate \(\theta = [\beta_1, \beta_2]^T\). However, from the Table I and also Fig. 1, one can see that, in the interval of \(x = [14, 21]\), the values of \(y\) do not follow the obvious linear trend. The data points in this interval are vertical outliers (marked in Table I with (*)).

LS estimation yields \(y = 0.316x + 2.95\), which is much attracted to the vertical outliers, as seen in Fig. 1. The LS estimate does fit neither the good nor the bad data points and, in the end, some outliers have smaller residues than some regular points.

The S-estimator produces a linear relation that neglects the outliers and gives a good approximation to the other points. The S-estimator, \(\hat{\theta}_S = [0.503, 1.97]^T\) yields a scaling \(s_S = 0.132\), much smaller than the scaling of the LS estimator, \(s_{LS} = 1.71\).

**III. IMPROVED S-ESTIMATOR**

The proposed algorithm is a variation of the fast-S algorithm (see [9]). The idea behind the fast-S is summarized as follows. First, multiple random samples of size \(p\) are taken from the dataset and, for each sample, a candidate for the parameter vector \(\theta\) is computed. If a given sample does not supply a unique solution to \(\theta\), more observations are added to the sample in question until it does. Next, the candidates are improved using the method described in subsection III-A.

With an appropriate number of initial candidates and by applying consecutive improvement steps to them, the candidate with smaller scaling \(s\) is chosen as the approximation to the solution of the problem in (5), [9].

The goal of the algorithm proposed here is to combine robustness to outliers, i.e. a high BDP, with the realtime estimation of the LS.

The fast-S algorithm deals with large static data sets. The improved S-estimator is capable of evolving and adapting to new observations by improving RLS estimates and comparing them to previous ones. The comparison is made by means of the scalings, \(s\), of the improved estimates.
A. Improvement Step

For each \( \hat{\theta} \), let \( \mathbf{r}(\hat{\theta}) = [r_1(\hat{\theta}) \cdots r_n(\hat{\theta})]^T \) be the vector of residues and \( s_f(\mathbf{r}(\hat{\theta})) \) represent the solution of (4).

Given an estimate \( \hat{\theta}^{(0)} \) of the regression coefficients, the I-step is defined (see [9]) as follows.

**I-step(a):** Compute \( s_0 = s_f(\mathbf{r}(\hat{\theta}^{(0)})) \);

**I-step(b):** Compute the weights \( w_i = w \left( r_i(\hat{\theta}^{(0)}) / s_0 \right) \), where \( w(u) = \rho'(u) / u \) and \( \rho' = \psi \) as in (2);

**I-step(c):** Define \( \hat{\theta}^{(1)} \) as the Weighted Least Squares (WLS) estimator (see [10]) where the \( i \)-th observation receives weight \( w_i \), for \( i = 1, \ldots, n \).

Having defined the I-step, consider now the following Lemmas, for which the proofs are given in [9]. For the Lemmas to hold an additional condition on the \( \rho \) is necessary.

**C4:** The function \( \psi(u)/u \) is decreasing in \( u > 0 \).

Considering a \( \rho \) as in (1), one has

\[
\frac{\psi(u)}{u} = \begin{cases} 
\left( 1 - \frac{u}{c} \right)^2 & \text{for } |u| \leq c \\
0 & \text{for } |u| \geq c
\end{cases}
\]

which satisfies condition **C4**.

**Lemma 3.** Suppose that \( \rho \) satisfies conditions **C1**, **C2** and **C4** and that \( \hat{\theta}^{(1)} \) was obtained by applying the I-step above to the estimate \( \hat{\theta}^{(0)} \). Then \( s_f(\mathbf{r}(\hat{\theta}^{(1)})) \leq s_f(\mathbf{r}(\hat{\theta}^{(0)})) \).

**Lemma 4.** Assume that \( \rho \) satisfies conditions **C1**, **C2** and **C4**. Then, for any starting point \( \theta^{(0)} \), any accumulation point of the sequence \( \theta^{(n)} \) obtained by applying the I-step consecutively is a local minimum of \( s_f(\mathbf{r}(\theta)) \).

The Lemmas describe two important characteristics of the I-step: applying the I-step to an estimate will result in a new estimate with a smaller scaling \( s \); and, by applying consecutive I-steps, the estimate will converge to one that yields a local minimum to the problem of (5).

Using the data and results of the example of Section II-D, the LS estimator was

\[
\hat{\theta}_{LS} = [\hat{\beta}_1, \hat{\beta}_2]^T = [0.316, 2.95]^T.
\]

Taking \( \hat{\theta}_{LS} \) as \( \hat{\theta}^{(0)} \), the I-step was performed 5 times (see Table II). Clearly Lemmas 3 and 4 hold.

Note that by the fourth iteration the result is approximately the same as in the example of Section II-D.

B. Algorithm

The main idea behind the fast-S algorithm was to improve candidates, computed as the hyperplanes determined by samples of size \( p \), randomly taken from the whole data set. The same idea was used for the realtime algorithm. However, instead of improving candidates from random samples, the routine improves estimates computed with a standard Recursive Least Squares (RLS) estimator (see [10]). Therefore, one can say that this new algorithm is a refinement (a filtering stage) of the RLS estimator. The structure of the algorithm can be seen in Fig. 2.

To apply the I-step and to compare the quality of the estimates it is necessary to compute the scaling of the current estimate. In order for this scaling, \( s \), to be statistically significant a large enough sample of observations must be used as it should be representative of the process being identified. For example, the outlier density inside the sample should be similar to the one of the whole data set or, at least, less or equal to the BDP for which the estimator was designed. Often the amount of observations received between update instants will be insufficient to provide a rich sample. Therefore, it is necessary to create a sample pool of \( N \) observations.

In the case of the identification of constant quantities, one can select for the pool the \( N \) observations that yields the smallest residues, up until the current update instant. However, usually when identifying parameters in realtime, the goal is to watch the evolution of this parameters through time. For these cases, a sliding window, formed by the \( N \) past observations, is a more adequate method of constructing the sample pool.

The algorithm for one update instant \( t \) is as follows.

**Step 1:** Compute \( \phi(t) \) and \( y(t) \) and perform an update step of the RLS algorithm as in [10]. This yields the estimate \( \hat{\theta}_{RLS}(t) \) and the covariance matrix \( P_{RLS}(t) \).

**Step 2:** Add \( \phi(t) \) and \( y(t) \) to the observations sample

---

**Table II:** Example: 5 consecutive iterations of I-step.

<table>
<thead>
<tr>
<th>Iter.</th>
<th>( \beta_1^{(k)} )</th>
<th>( \beta_2^{(k)} )</th>
<th>( s^{(k)} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.316</td>
<td>2.95</td>
<td>1.71</td>
</tr>
<tr>
<td>1</td>
<td>0.381</td>
<td>2.69</td>
<td>1.28</td>
</tr>
<tr>
<td>2</td>
<td>0.465</td>
<td>2.17</td>
<td>0.576</td>
</tr>
<tr>
<td>3</td>
<td>0.494</td>
<td>2.00</td>
<td>0.191</td>
</tr>
<tr>
<td>4</td>
<td>0.500</td>
<td>2.00</td>
<td>0.136</td>
</tr>
<tr>
<td>5</td>
<td>0.500</td>
<td>2.00</td>
<td>0.136</td>
</tr>
</tbody>
</table>

**Figure 2:** Structure of the improved S-estimator.
The problem does not provide a unique solution. This is due to the fact that the observations in the sample window variables \( \Phi(t) = \begin{bmatrix} \vdots & \vdots & \vdots \\ \phi(t-2) & y(t-2) \\ \phi(t-1) & y(t-1) \\ \phi(t) & y(t) \end{bmatrix} \) and \( Y(t) = \begin{bmatrix} \vdots & \vdots & \vdots \\ y(t-2) & \vdots & y(t) \end{bmatrix} \). If the number of observations on the sample is:
- less than \( N \), make \( \theta(t) = \hat{\theta}_{RLS}(t) \) and set the scaling \( s(t) \) as a very large number. Go back to Step 1.
- equal to \( N \), advance to Step 3.
- greater than \( N \), make the size of the sample equal to \( N \) by discarding the oldest observations. Then, go on to Step 3.

Step 3: Perform \( k \) improvement steps, as described in subsection III-A, using the sample window to compute the residuals and to improve the estimate via WLS. After \( k \) I-steps one has an improved estimate \( \hat{\theta}_{imp}(t) \).

Step 4a) Define
\[
\dot{s}_{min} = (1 + \eta)^{t_0 - 1} \cdot s_{min};
\]
where the scalar \( \eta \) is the scaling forgetting factor.

(b) Compute the scaling
\[
s_{imp}(t) = \alpha \left( \hat{\theta}_{imp}(t) \right)
\]
and
\[
\hat{\theta}(t) = \begin{cases} 
\theta(t-1), & s_{imp}(t) \geq \dot{s}_{min} \\
\hat{\theta}_{imp}(t), & s_{imp}(t) < \dot{s}_{min}. 
\end{cases}
\]
(c) If \( s_{imp}(t) < \dot{s}_{min} \) also make \( s_{min} = s_{imp}(t) \) and \( t_0 = t \), else ignore this sub-step.

At some moment, not only the vector of parameters being estimated \( \theta \) changes, but the general quality of the measurements also decreases. This new regime inserts more noise and/or outliers into the identification process and the scalings corresponding to the estimates computed before the change in \( \theta \) are smaller than the ones computed after. Therefore, the algorithm will always ignore the estimates on the new value of \( \theta \) on account of their scalings, becoming deadlocked on the old estimates.

This is a similar problem to the case of the dormant RLS algorithm, solved using a forgetting factor \( \lambda \). Similarly, the proposed solution to avoid a deadlock on the improved S-estimator is the scaling forgetting factor \( \eta \).

Furthermore, in order to use the improved S-estimator for realtime estimation, a small modification of the I-step is necessary. The improved estimates may be computed using the batch formula of the weighted least squares. However, it may occur that the observations in the sample window do not form persistently exciting data (see [10]) and the WLS problem does not provide a unique solution. This is not normally a problem for the fast-S algorithm since the samples or subsamples are obtained randomly from a larger set and, even if there is no need to take samples, the data sets considered are much larger. To tackle this problem, instead of solving the batch formula for the WLS, the problem is solved recursively. Then, the modified I-step at iteration \( i = 1, \ldots, k \) is implemented as follows:

**I-step(a):** Compute the weights, \( w_n(i) \), \( n = 1, \ldots, N \), for each observation on the sample as presented in subsection III-A.

**I-step(b):** Define a weighting matrix as
\[
W(i) = \begin{bmatrix}
\sqrt{w_1(i)} & 0 & \cdots & 0 \\
0 & \sqrt{w_2(i)} & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & \cdots & \cdots & \sqrt{w_N(i)}
\end{bmatrix},
\]
and new regression matrices as
\[
\Phi_w(i) = W(i)\Phi(t) \quad \text{and} \quad Y_w(i) = W(i)Y(t).
\]

**I-step(c):** Compute an update step of the RLS algorithm, obtaining the improved estimate \( \hat{\theta}_{imp}(t) \) and the covariance matrix \( P_{imp}(i) \), with the matrices \( \Phi_w(i) \) and \( Y_w(i) \).

The internal RLS algorithm in step (c) is initialized with the covariance matrix computed by the external RLS routine at that update time, \( P_{RLS}(t) \), on the first iteration, \( i = 1 \). Also, since weighting of the observations already has a filtering effect on noise and/or outliers, the forgetting factor of the internal RLS, \( \lambda_{Istep} \), can be set at a smaller value than the one used for the external routine. This increases the rate of convergence of the improved estimates and their scalings.

The BDP of the improved S-estimator can be computed, for each update step, as in Section II-C, factoring the number of parameters being estimated, \( p \), size of the sample pool, \( N \), and the S-estimator design parameters, \( c \) and \( K \).

### IV. Case Studies

The new algorithm is compared with the ordinary RLS using two case studies.

In the first case the performance of the improved S-estimator is assessed by estimating the inertia tensor of the orbital stage of a launcher vehicle. The data was obtained using a simulator with realistic models for the environment and spacecraft systems.

For the second case, the algorithm is used to determine the radius of curvature of the trajectory of a car in a roundabout. Real measurements performed with an IMU sensor mounted on a car are used.

#### A. Spacecraft Inertia Estimation

The inertia estimation of a spacecraft, using an LS approach, can be done using the equality provided by the
rotational equations of motion [11], which for a spacecraft with constant inertia, \( J \), are
\[
J \ddot{\omega} + \omega \times (J \omega) = \tau,
\]
where \( \dot{\omega} \) and \( \omega \) are the angular acceleration and velocity, respectively, and \( \tau \) is the sum of all torques on the body. The regression form \( y = \phi^T \theta \) can be seen in [12], where \( \theta \) is a \( 6 \times 1 \) vector of deviations of the components of the inertia tensor with respect to a nominal value.

In the following scenarios, the acceleration and the velocity are obtained from simulated sensor data and the torque is estimated from the actuator commands given by the onboard software.

In both scenarios, the initial guess for the inertia matrix has an error of 10% in each component, the initial covariance matrix is \( P_0 = 2000 I_6 \), \( kg^2 m^4 \) and the exponential forgetting factor of the RLS estimator is \( \lambda_{RLS} = 0.9992 \). The parameters related to the S-estimator are shown in Table III.

### Table III: S-estimator Parameter Values

<table>
<thead>
<tr>
<th></th>
<th>Case Study A</th>
<th>Case Study B</th>
</tr>
</thead>
<tbody>
<tr>
<td>( c/1K )</td>
<td>1.547/0.1995</td>
<td>1.547/0.1995</td>
</tr>
<tr>
<td>( k )</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>( N )</td>
<td>60</td>
<td>20</td>
</tr>
<tr>
<td>( \lambda_{Jstep} )</td>
<td>0.9</td>
<td>0.8</td>
</tr>
<tr>
<td>( \eta )</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 3: Estimates error for spacecraft baseline test

1) **Baseline test:** This test is a baseline and to attest the behaviour of the new algorithm in nominal conditions, i.e. with realistic sensor/actuator noise but no outliers.

Fig. 3 shows the results, in terms of percentage of error made in estimating the moments of inertia, of both the standard RLS algorithm and the S-estimator. The estimates of the two algorithms converge to the real value. The estimates of the RLS algorithm are faster to converge but more erratic, while the S-estimator produces estimates that converge slightly slower but show a filtering effect.

2) **Misfire test:** In this test, a misfire scenario was included where, in the time period between 2000 and 2500 seconds of the simulation, some of the actuators of the spacecraft will not fire, even if commanded to. From the estimation algorithm point-of-view, this will produce a torque that will not generate motion (results shown in Fig. 4).

The estimates of the RLS algorithm, during the misfire time window, deviate from the real values of the moments of inertia, producing errors that can be higher than 1000% (out of scale in Fig. 4). The same is not true for the the S-estimator as it ignores the estimates made during this period.

Figure 4: Estimates error for spacecraft misfire test

### B. Radius of Curvature Estimation

The objective of this experiment is to determine the radius of curvature of the trajectory of a car in a roundabout. In the following tests, the data was collected using a real IMU sensor mounted on an actual car performing an approximate circular trajectory. The radius of curvature, \( r \), is related to the acceleration in the radial direction, \( a_r \), and with the angular velocity, \( \omega \), at which the car travels in its circular motion by
\[
a_r = -\omega^2 r,
\]
which is already in regression form. Using the notation of Section II: \( y = a_r, \phi = -\omega^2 \) and \( \theta = r \).

The RLS algorithm was initialized with \( r_0 = 0 \) \( m \) and \( P_0 = 100 m^2 \) and \( \lambda_{RLS} = 0.99 \). Unless expressly mentioned otherwise, the values of the parameters related to the S-estimator are the ones shown in Table III.

1) **Baseline test:** Once again, the performance of the S-estimator is compared to the RLS in nominal conditions. The GPS data indicates that the radius of the trajectory in this test varies between 10 and 10.5 meters.

The results are shown in Fig. 5, where both algorithms show similar performances, thus validating the implementation of the S-estimator. The RLS estimates present a cyclic variation, which is realistic since the trajectory is not perfectly circular, while the S-estimates converge to a constant value.
2) Scaling forgetting factor: In this test, the car started by performing the trajectory described in the previous test however, at a given point, the radius of the trajectory increases. While following the widest trajectory, the noise intensity present on the IMU measured data is larger.

The goal of this test is to illustrate the deadlock characteristic of the developed algorithm, and how tuning the scaling forgetting factor, $\eta$, can prevent it.

Fig. 6 shows the RLS estimates and two versions of the S-estimator, one with a null scaling forgetting factor (denominated S0) and another with $\eta = 0.01$ (denominated S1). As one can see, the S0 estimator is deadlocked on the smaller radius for more than 2500 updates while, by tuning $\eta$, the S1 estimator follows the RLS estimates after approximately 100 updates. Furthermore, the S1 estimator is able to present results that show the smaller variations in the curvature radius than the RLS estimates. This strengthens the validity of the new algorithm since, in the absence of the outliers and with a proper tuning of the scaling forgetting factor, the results of the improved S-estimator approximate those of the RLS algorithm.

3) Outliers: For this test, the data used on the baseline test was contaminated with simulated outliers. The probability of the outlying value being $y$, $\phi$ or the combination of both is the same. Their magnitude, $m_*$, is computed as

$$ m_y(t) = y(t) + (m_y^{MAX} - y(t))u_y(t) $$
$$ m_\phi(t) = \phi(t) + (m_\phi^{MAX} - \phi(t))u_\phi(t) $$

where $m_*^{MAX}$ are the maximum values the outliers can take and $u_*$ are uniformly distributed random variables in the $[0, 1]$ interval. The maximum values for the outliers are 0.4 and 4 for $\phi$ and $y$, respectively. The percentage of outliers used is approximately 15% and a realization of a contaminated data set can be seen in Fig. 7.

A Monte Carlo study consisting of 500 test runs ($N_r = 500$) was performed, with $\eta = 0.01$.

The performance criteria to compare the RLS and S-estimators will be the root mean square error (RMSE) of their estimates with respect to the estimates computed with the RLS algorithm, using outlier free data set. In order to accept that the S-estimator produces results closer to the estimates computed with a clean data set than the RLS algorithm, one must test the hypothesis

$$ H_1 : \Delta \triangleq \text{RMSE}^{(RLS)} - \text{RMSE}^{(S)} > 0. $$

Computing the sample mean, $\bar{\Delta}$, and its standard error, $\sigma_{\bar{\Delta}}$, yields

$$ \bar{\Delta} = \frac{1}{N_r} \sum_{i=1}^{N_r} \Delta_i = 0.9387 \text{ m}, $$
$$ \sigma_{\bar{\Delta}} = \sqrt{\frac{1}{N_r^2} \sum_{i=1}^{N_r} \left( \Delta_i - \bar{\Delta} \right)^2} = 0.0312 \text{ m}. $$
The hypothesis $H_1$ is accepted, with a significance of 95%, if the condition
\[
\mu \triangleq \frac{\bar{\Delta}}{\sigma_{\Delta}} > G(95\%) = 1.64,
\]
where $G(1-\alpha)$ is represents the point on the standard Gaussian distribution corresponding to upper tail probability of $\alpha$, is true.

Since $\mu = 30.04$, one can say that the S-estimator outperforms the RLS in the presence of outliers. Fig. 8 shows the average of the estimates for both estimators in the Monte Carlo test runs. Note that, in the presence of outliers, the mean performance of the S-estimator approximates the LS optimal solution.

![Figure 8: The average of the estimates from RLS and S-estimators of the Monte Carlo runs compared with the RLS estimates with clean data.](image)

V. CONCLUSIONS

An improved S-estimator to be used together with a standard RLS algorithm was developed. The algorithm is based on the fast-S [9] and its local improvement step. The basic idea is to improve the current RLS estimate at each update step and compare it to the previously improved estimates using the scaling $s$.

The algorithm was tested with simulated and real data and it allies the recursive features of the RLS estimator with the outlier filtering capabilities of the S-estimator.

Additional tuning parameters make the new algorithm more versatile. These are: (i) the BDP of the S-estimator, which sets the values of the constants $c$ and $K$; (ii) the $k$ number of I-steps performed at each update; (iii) the size of the sample window, $N$; (iv) the forgetting factor of the I-step internal RLS estimator, $\lambda_{I_{\text{step}}}$; (v) the scale forgetting factor, $\eta$.

While tracking changes in $\theta$, the fact that the algorithm depends on a observation pool creates a delay in the identification. This delay depends on the BDP of the S-estimator being used and on the size of the sample pool, $N$. The smallest delay one can expect is calculated as
\[
\min \{ \text{delay} \} = (1 - \varepsilon^*_N) N \Delta t,
\]
where $\varepsilon^*_N$ is the BDP for which the S-estimator was designed and $\Delta t$ is the time between update instants. However, this is rather simplistic, since there are other parameters that greatly influence the delay, such as the scale forgetting factor and the outlier density in the past samples, i.e. their quality.

The new algorithm was tested in scenarios simulating real time. It has a larger computational complexity than the ordinary RLS, requiring a time two orders of magnitude higher than the RLS to compute an estimate. However, the average time to compute an estimate is still low enough for realtime implementation.

Future work on this subject should be focused on testing the algorithm in realtime and on studying how exactly each of the mentioned tuning parameters influences the performance and behavior of the algorithm.

ACKNOWLEDGEMENTS

This work was partially supported by FCT project PEst-OE/EEI/LA0009/2011.

REFERENCES