Online Estimation of Vehicle Driving Resistance Parameters with Recursive Least Squares and Recursive Total Least Squares*

Stephan Rhode and Frank Gauterin[1]

***Abstract*— We introduce a recursive generalized total least-squares (RGTLS) algorithm with exponential forgetting that is used for estimation of vehicle driving resistance parameters. A vehicle longitudinal dynamics model and available control area network (CAN) signals form appropriate estimator inputs and outputs. In particular, we present parameter estimates for the vehicle mass, two coefficients of rolling resistance, and drag coefficient of one test run on public road. Moreover, we compare the results of the proposed RGTLS estimator with two kinds of recursive least-squares (RLS) estimators. While RGTLS outperforms RLS with simulation data, the recursive least squares with multiple forgetting (RLSMF) estimator provides superior accuracy and sufficient robustness through orthogonal parameter projection with experimental data. Here, RGTLS needs a parameter projection scheme that is equivalent to RLSMF.**

I. INTRODUCTION

Energy-efficient trajectory planning, range prediction, and recuperation strategies are highly reliant on accurate models of the vehicle total driving resistance. Parametric models are commonly used with unknown parameters that vary at different rates such as the vehicle mass, coefficients of rolling resistance, and drag coefficient. This is why one needs fast and accurate estimates of unknown driving resistance parameters.

Previous work mainly focused on the estimation of individual parameters. In particular, the vehicle mass was the subject of numerous studies. Input-output or state-space models of the vehicle longitudinal dynamics in connection with recursive least squares (RLS) or Kalman filters are applied most often to perform online estimation. Available control area network (CAN) signals feed these models with the required driving state information such as the velocity, acceleration or the current gear ratio.

Fathy, Kang, and Stein [1] presented a vehicle mass RLS estimator with a fuzzy supervisor to extract beneficial driving states where the vehicle motion is predominantly longitudinal. Road grade and rolling resistance were separated by a band-pass filter.

Vahidi, Stefanopoulou, and Peng [2] introduced a simultaneous vehicle mass and road grade RLS estimator with multiple forgetting factors. Their estimator considers time-varying and time-constant parameters with multiple forgetting factors.

*This work was accomplished in cooperation with the Energy Management Complete Vehicle Department at Dr. Ing. h.c. F. Porsche AG, Weissach, Germany

[1]S. Rhode and F. Gauterin are with the Institute of Vehicle System Technology, Karlsruhe Institute of Technology, 76131 Karlsruhe, Germany {stephan.rhode, frank.gauterin}@kit.edu

An early attempt to estimate all major driving resistance parameters was made by Bae, Ryu, and Gerdes [3]. They used road grade information from a global positioning system (GPS) and performed simultaneous online estimates of the vehicle mass, one coefficient of rolling resistance, and the drag coefficient with a RLS estimator.

The contribution of this paper is a recursive generalized total least-squares (RGTLS) estimator that offers exponential forgetting and treats data with unequally sized and correlated noise. First, we review briefly the basic total least-squares (TLS), least-squares (LS) and generalized total least-squares (GTLS) methods in Sec. II. Then, we show the transition from TLS into recursive total least squares (RTLS) in Sec. III-A–Sec. III-C, while Sec. III-D introduces RGTLS. Moreover, Sec. III-E provides the well-known RLS and an extension to recursive least squares with multiple forgetting (RLSMF). We compare the results of RGTLS and RLS with simulation data in Sec. III-G. Sec. IV deals with the vehicle longitudinal dynamics model (Sec. IV-A) and methods to exclude adverse driving states Sec. IV-B). The estimates of various driving resistance parameters are presented in Sec. V, where we compare the previously discussed estimators. Finally, Sec. VI gives conclusions.

II. TOTAL LEAST SQUARES

First, we introduce some notation for element-wise matrix operations. The element-wise product is donated by $A \odot B := A_{ij} \cdot B_{ij}$; the element-wise division is donated by $A \oslash B := A_{ij}/B_{ij}$, and the element-wise power by $A \circledcirc k := A_{ij}^k$. Finally, we vectorize the main diagonal elements of matrix into a column vector with $\mathrm{main}(A) := (A_{ii})_{i=1\ldots n}$.

Total least squares is a data fitting method for the unconstrained perturbation problem (1) that is known as errors-in-variables (EIV) model.

$$AX \approx B, \quad A = \overline{A} + \widetilde{A}, \quad B = \overline{B} + \widetilde{B}. \tag{1}$$

The input data $A \in \mathbb{R}^{m \times n}$ and output data $B \in \mathbb{R}^{m \times d}$ are sums of noise-free data $\overline{A}$, $\overline{B}$ and measurement noise $\widetilde{A}$, $\widetilde{B}$, respectively. From now on, we focus on overdetermined $(m > n)$, multivariate $(n > 1)$ and one-dimensional $(d = 1)$ systems, that are also known as multi-input single-output (MISO) systems.

Markovsky and Van Huffel [4] pointed out, that TLS searches for optimal data corrections $[\Delta A \; \Delta B] := [A \; B] - [\widehat{A} \; \widehat{B}]$ (2), to find an approximate solution of the overdetermined system of equations $\widehat{A}\widehat{X} = \widehat{B}$.

$$\min_{[\widehat{A} \; \widehat{B}] \in \mathbb{R}^{m \times q}} \left\| [A \; B] - [\widehat{A} \; \widehat{B}] \right\|_F \tag{2}$$

Copyright ©IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE. S. Rhode and F. Gauterin. "Online estimation of vehicle driving resistance parameters with recursive least squares and recursive total least squares." In: *Intelligent Vehicles Symposium (IV)*, 2013, pp. 269–276, doi: 10.1109/IVS.2013.6629481. *This is the submitted version. Please note, that the results of* RLSMF *in Sec. V differ significantly from the final version of this article.*

If the noise is independently identically distributed (i.i.d.) with zero mean and a covariance matrix

$$\mathrm{cov}([\widetilde{A}\ \widetilde{B}]) = \sigma^2 I, \tag{3}$$

equal to the identity matrix I up to σ^2, TLS is the maximum-likelihood estimator for (1), [4]. Note that σ^2 is an unknown scalar, that does not affect the TLS correction. Another class of EIV identification is instrumental variables (IV), studied in [5].

The solution of the basic TLS requires the singular value decomposition (SVD) (4) of the data matrix $Z = [A\ B]$, where $Z \in \mathbb{R}^{m\times q}$. The matrices $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{q\times q}$ are orthogonal $U^\top U = I$, $V^\top V = I$ and their columns are called the left and right singular vectors, respectively. The positive diagonal matrix $S \in \mathbb{R}^{m\times q}$ contains the singular values of Z in decreasing order.

$$Z = USV^\top, \quad U^\top ZV = S, \quad S = \mathrm{diag}(S_1, \ldots, S_q). \tag{4}$$

A. Total Least-squares Algorithm

Alg. 1 provides the required computations for the basic TLS solution. First, compute the SVD of Z (Alg. 1 line 3). After that, partition V (Alg. 1 line 4) and finally compute the parameter estimate $\widehat{X}$ according to Alg. 1 line 5, [6, p. 37]. Note that only V is needed from the SVD in Alg. 1 line 3 to compute the parameter estimate $\widehat{X}$ in Alg. 1 line 5. The solution is generic (does exist) if V_{22} is non-singular. In our case with $d = 1$, it is generic if $V_{22} \neq 0$. Furthermore, the solution is unique if $S_n \neq S_q$, [4]. Extensions to the non-generic and non-unique case are categorized in [6, p. 50].

In our opinion, the covariance information $\mathrm{cov}(\widehat{X})$ of the estimate $\widehat{X}$ is as important as the estimate itself. This is a challenging task in TLS and is discussed only insufficiently in the TLS literature. Van Huffel and Vandewalle [6, p. 242] provide an approximate covariance formula that we integrate in Alg. 1 line 6. The data corrections are given with

Alg 1. TLS

input: $A,\ B$

1 *batch*

2 $\quad Z = [A\ B]$

3 $\quad USV^\top = \mathrm{svd}(Z)$

$$\qquad\qquad\quad n \quad d$$

4 $\quad V := \begin{bmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{bmatrix} \begin{matrix} n \\ d \end{matrix}$

5 $\quad \widehat{X} = -V_{12}{V_{22}}^{-1}$

6 $\quad \mathrm{cov}(\widehat{X}) \approx \dfrac{(1+\|\widehat{X}\|_2^2)\sigma^2}{A^\top A - m\cdot\sigma^2\cdot I}$ with $\sigma^2 \approx \dfrac{S_{q,q}}{m}$

output: $\widehat{X},\ \mathrm{cov}(\widehat{X})$

$$[\Delta A\ \Delta B] = S_{q,q} U_{:,q} V_{:,q}^\top, \tag{5}$$

[6, p. 35]. We found that

$$[\Delta A\ \Delta B] \approx [A\ B] \begin{bmatrix} V_{12} \\ V_{22} \end{bmatrix} [V_{12}^\top\ V_{22}^\top] \tag{6}$$

is a more convenient form that is derived according to [7, p. 435]. During our simulations, the error between the exact form (5) and the approximate (6) was in the range of machine precision. Note that (6) has the advantage that the costly matrix U is not required. Finally, the approximate data is $[\widehat{A}\ \widehat{B}] = [A\ B] - [\Delta A\ \Delta B]$.

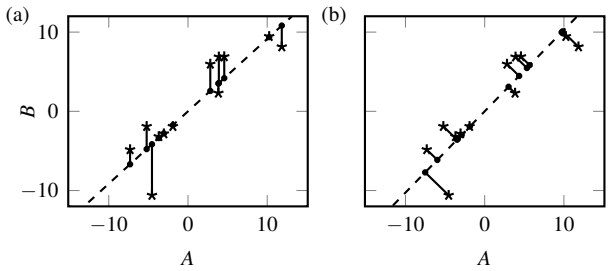

Fig. 1: Data fitting with LS in Fig. 1 (a) and TLS in Fig. 1 (b). $\star$ shows the data $[A\ B]$, $\bullet$ shows approximations $[\widehat{A}\ \widehat{B}]$, - - - shows the estimated model and —— shows the corrections $[\Delta A\ \Delta B]$.

B. How Least Squares Differs from Total Least Squares

We use the term LS as short version of OLS (ordinary least squares). Otherwise, the nomenclature would differ from Sec. III, where RLS is used for recursive versions of LS according to the literature.

Conversely to TLS, LS assumes a constrained perturbation problem with noise-free inputs $\widetilde{A} := 0$. All noise is referred to the output $B = \overline{B} + \widetilde{B}$ and the solution of the overdetermined system of equations is $A\widehat{X} = \widehat{B}$. LS searches for optimal corrections (7) only in B and is the maximum-likelihood estimator for noise-free inputs. The closed-form LS solution is $\widehat{X} = (A^\top A)^{-1} A^\top B$ [8, p. 4], while the covariance is commonly given by $P = \frac{\sigma^2}{A^\top A}$ with $\sigma^2 \approx \frac{\Delta B^\top \Delta B}{m-n}$.

$$\min_{\widehat{B}\in\mathbb{R}^{m\times d}} \|B - \widehat{B}\|_2 \tag{7}$$

Fig. 1 visualizes the difference between LS and TLS. While LS corrects the data vertically and assumes that A is exactly known, TLS performs perpendicular data corrections. That is also the reason why TLS is sometimes called orthogonal regression.

C. Generalized Total Least-squares Algorithm

So far, TLS seems to be the superior method, due to the more realistic unconstrained perturbation model (1). However, as noted, TLS requires quite restrictive conditions for maximum-likelihood characteristics. In practice, it is unlikely that all errors are uncorrelated and equally sized as required by (3).

Generalizations of basic TLS can deal with column-wise or row-wise correlated noise and unequally sized error covariance matrices [9]. Markovsky et al. [10] introduced an element-wise weighted TLS method and accepted the drawback of losing a closed-form solution.

Apart from SVD-based TLS methods, Schaffrin and Wieser [11] introduced an element-wise weighted TLS method based non-linear Lagrange functions and Shen, Li,

and Chen [12] solved this problem with a Newton-Gauss-based scheme.

Schuermans et al. [13] provide the simplest kind of GTLS scheme through rescaling the data in a way that the noise covariance matrix meets the form required by TLS (3). This data scaling is performed from line 2–line 4 in Alg. 2. The Cholesky decomposition of the weighting matrix W (line 2) is used to transform the data into a new space in line 4. The basic TLS algorithm is used as nested function in Alg. 2 line 6 to compute the parameter estimates $\widehat{X}'$ in the transformed space. Finally, Alg. 2 line 7 converts $\widehat{X}'$ back in the original space.

Alg 2. GTLS

input: $A,\ B,\ W$

1 *batch*

2 $\quad C = \mathrm{chol}(W)$

3 $\quad C^{-1} := \begin{bmatrix} C_{11} & C_{12} \\ 0 & C_{22} \end{bmatrix} \begin{matrix} n \\ d \end{matrix} \quad \begin{matrix} n & d \end{matrix}$

4 $\quad [A'\ B'] = [A\ B] \cdot C^{-1}$

5 $\quad$ *function call*

6 $\quad\quad$ *input*: $A',\ B'$; Alg. 1; *output*: $\widehat{X}',\ \mathrm{cov}(\widehat{X}')$

7 $\quad \widehat{X} = \dfrac{C_{11} \cdot \widehat{X}' - C_{12}}{C_{22}}$

output: $\widehat{X},\ \mathrm{cov}(\widehat{X}')$

Alg. 2 can treat three different TLS problems:

1) If $W = I$, Alg. 2 works like Alg. 1 in the TLS sense (errors are equally sized and uncorrelated);
2) If $W = \mathrm{cov}([\widetilde{A}\ \widetilde{B}])$, Alg. 2 acts as GTLS (errors are unequally sized and correlated);
3) If W is a diagonal matrix $W = I \odot \mathrm{cov}([\widetilde{A}\ \widetilde{B}])$, Alg. 2 computes the weighted total least-squares (WTLS) solution (errors are unequally sized and uncorrelated).

III. ONLINE ESTIMATORS

A. Steps Towards Recursive Total Least Squares

If one thinks of online applications, the major drawback of Alg. 1 and Alg. 2 is the computational effort. The size of Z, U, and S depends on m. Hence, the SVD computation becomes slower with increasing number of data. A buffer is required to store previous data and computationally expensive re-calculation of old data occurs in each iteration. By reason of this, the batch computation of the SVD is inapplicable on control units that are limited in memory and processing power. However, we take advantage of three important facts to obtain a recursive form of TLS (Alg. 1) and GTLS (Alg. 2):

1) A full SVD is not required to solve TLS. The costliest matrix U is not involved in Alg. 1;
2) The size of V is independent of m, that means it can be easily stored;
3) Powerful SVD update and downdate schemes were discovered in [14–17], that can be used interchangeably.

To the best of our knowledge, Kubus, Kroger, and Wahl [18] proposed the first RTLS algorithm based on the SVD update scheme of Brand [16]. Apart from SVD-based RTLS that we discuss in Sec. III-C, other online-capable TLS methods were discovered. Rayleigh quotient-based RTLS algorithms are shown in [8, 19] while Lim, Choi, and Sung [20] proposed a square root-free Cholesky decomposition (UDU)-based RTLS method. Additionally a rank-revealing ULV decomposition-based RTLS method was introduced in [21, p.117-126].

B. Updating the Singular Value Decomposition

The recursive singular value decomposition (RSVD) Alg. 3 is based on the SVD update algorithm of Gu and Eisenstat [14], but skips the update of U entirely. The interested reader is referred to [14] for the full SVD update procedure. The basic idea is to take advantage of the previous SVD matrices $S(t-1)$ and $V(t-1)$ when new data arrives in Alg. 3 line 2. We focus here on appending a single data row, but the algorithm works as well when a batch of new samples arrives. Herein, we extend our previous scheme [22] with an exponential forgetting factor λ in line 4. Accordingly, Alg. 3 has a fading memory. In line 5 and line 6, we implement the truncation approach of [16] to size the following SVD to an efficient rank. Note that J is not needed further on. Brand [16] defines v as volume of $z^\top$ that is orthogonal to $V(t-1)$. If $v < \mathfrak{v}$ (line 6), where $\mathfrak{v}$ is a chosen threshold near the machine precision, we downsize $S(t)$ to $S(t) \in \mathbb{R}^{q \times q}$ in line 9. This means, that the number of singular values in $S(t)$ remains at q and the computational effort is notably reduced. Otherwise, Matlab's economy-sized SVD command `svd(M,'econ')` ensures that $S(t) \in \mathbb{R}^{q \times q}$ (line 11). Hence, S and V remain in size and Alg. 3 is suitable for online applications.

Alg 3. RSVD

input: $A(t),\ B(t),\ S(t-1),\ V(t-1),\ \mathfrak{v},\ \lambda$

1 *batch*

2 $\quad z = [A(t)\ B(t)]^\top$

3 $\quad a = V(t-1)^\top \cdot z$

4 $\quad M = \begin{bmatrix} \lambda \cdot S(t-1) \\ a^\top \end{bmatrix}$

5 $\quad JK = \mathrm{qr}(z - V(t-1) \cdot a)$

6 $\quad v = \sqrt{\det(K^\top K)}$

7 $\quad$ *if* $v < \mathfrak{v}$ *then*

8 $\quad\quad PNQ^\top = \mathrm{svd}(M)$

9 $\quad\quad S(t) = N_{1:q,1:q}$

10 $\quad$ *else*

11 $\quad\quad PS(t)Q^\top = \mathrm{svd}(M,\ \text{'econ'})$

12 $\quad V(t) = V(t-1) \cdot Q$

output: $S(t),\ V(t)$

C. Recursive Total Least-squares Algorithm

RTLS in Alg. 4 has basically the same structure as TLS (Alg. 1). However, Alg. 4 is iteratively executed throughout the data set from time step $t = 1$ to $t = m$, while Alg. 1 computes the result for the entire data set in one step. The

nested RSVD in Alg. 4 line 3 replaces the batch SVD in Alg. 1 line 3 and requires the additional inputs $S(t-1)$, $V(t-1)$, $\mathfrak{v}$, λ. The next two computations in Alg. 4 line 4 and line 5 are similar to Alg. 1 line 4 and line 5. The parameter covariance matrix in Alg. 4 line 7 requires the update of $(A^\top A)$ that is performed in Alg. 4 line 6.

Alg 4. RTLS

1 *for* $t \leftarrow 1$ *to* m *do*
  input: $A(t)$, $B(t)$, $S(t-1)$, $V(t-1)$, $(A^\top A)(t-1)$, $\mathfrak{v}$, λ
2 *function call*
    input: $A(t)$, $B(t)$, $S(t-1)$, $V(t-1)$, $\mathfrak{v}$, λ
3   Alg. 3
    output: $S(t)$, $V(t)$
4 $V(t) := \begin{array}{c} \begin{matrix} n & d \end{matrix} \\ \begin{bmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{bmatrix} \end{array} \begin{matrix} n \\ d \end{matrix}$
5 $\widehat{X}(t) = -V_{12}{V_{22}}^{-1}$
6 $(A^\top A)(t) = ((A^\top A)(t-1) \cdot \lambda^2) + A(t)^\top A(t)$
7 $\mathrm{cov}(\widehat{X}(t)) \approx \dfrac{(1+\|\widehat{X}(t)\|_2^2)\sigma^2}{(A^\top A)(t) - t \cdot \sigma^2 \cdot I}$ with $\sigma^2 \approx \dfrac{S(t)_{q,q}}{t}$
  output: $\widehat{X}(t)$, $\mathrm{cov}(\widehat{X}(t))$, $S(t)$, $V(t)$, $(A^\top A)(t)$

D. Recursive Generalized Total Least-squares Algorithm

The transition from RTLS to RGTLS is similar as TLS to GTLS and shown in Alg. 5. The data transformation in Alg. 5 line 2–line 4 is equal as in Alg. 2 line 2–line 4. The same holds for the conversion of $\widehat{X}'$ in Alg. 5 line 7 compared with Alg. 2 line 7. Note that we use now two nested functions in Alg. 5 line 6. Similar to the statements in Sec. II-C, Alg. 5 can treat various RTLS extensions through an appropriate setting of W. Therefore, we can use solely RGTLS from now on. On the other hand, this comfortable procedure is costly in the case of $W = I$ due to the unessential data transformation (Alg. 5 line 2–line 4) and back conversion (Alg. 5 line 7). The pure RTLS scheme (Alg. 4) is computationally cheaper if $W = I$.

Alg 5. RGTLS

1 *for* $t \leftarrow 1$ *to* m *do*
  input: $A(t)$, $B(t)$, $S'(t-1)$, $V'(t-1)$, $(A^\top A)'(t-1)$, $\mathfrak{v}$, λ, W
2 $C = \mathrm{chol}(W)$
3 $C^{-1} := \begin{array}{c} \begin{matrix} n & d \end{matrix} \\ \begin{bmatrix} C_{11} & C_{12} \\ 0 & C_{22} \end{bmatrix} \end{array} \begin{matrix} n \\ d \end{matrix}$
4 $[A'(t)\ B'(t)] = [A(t)\ B(t)] \cdot C^{-1}$
5 *function call*
    input: $A'(t)$, $B'(t)$, $S'(t-1)$, $V'(t-1)$, $(A^\top A)'(t-1)$, $\mathfrak{v}$, λ
6   Alg. 4
    output: $\widehat{X}'(t)$, $\mathrm{cov}(\widehat{X}'(t))$, $S'(t)$, $V'(t)$, $(A^\top A)'(t)$
7 $\widehat{X}(t) = \dfrac{C_{11} \cdot \widehat{X}'(t) - C_{12}}{C_{22}}$
  output: $\widehat{X}(t)$, $\mathrm{cov}(\widehat{X}'(t))$, $S'(t)$, $V'(t)$, $(A^\top A)'(t)$

E. Recursive Least Squares

Without detailed explanation, we present the well-known RLS scheme in Alg. 6, [23, p. 365]. RLS offers simple implementation and is computationally cheap.

Alg 6. RLS

1 *for* $t \leftarrow 1$ *to* m *do*
  input: $\widehat{X}(t-1)$, $P(t-1)$, $A(t)$, $B(t)$, λ
2 $L(t) = \dfrac{P(t-1) \cdot A(t)^\top}{\lambda + A(t) \cdot P(t-1) \cdot A(t)^\top}$
3 $\widehat{X}(t) = \widehat{X}(t-1) + L(t)(B(t) - A(t) \cdot \widehat{X}(t-1))$
4 $P(t) = (I - L(t) \cdot A(t))P(t-1)\dfrac{1}{\lambda}$
  output: $\widehat{X}(t)$, $P(t)$

Vahidi, Stefanopoulou, and Peng [2] developed the vector-type forgetting RLS estimator of [24] further to prevent the wind-up effect. The wind-up effect is the exponential growing of the covariance matrix P during low system excitation or when multiple parameters vary at different rates. The extension of [2] outperforms the basic RLS estimator (Alg. 6) in their experiments. Note that there is no proof of convergence as for the most RLS extensions. Vahidi, Stefanopoulou, and Peng [2] present their recursive least-squares with multiple forgetting (RLSMF) scheme for two parameters ($n = 2$).

We can rewrite the equations of [2] and present an n-independent form in Alg. 7 that is related to the basic RLS scheme in Alg. 6. Note that, in contrast to RLS, RLSMF has a diagonal covariance matrix P and requires a forgetting factor for each parameter $\Lambda = [\Lambda_1\ \ldots\ \Lambda_n]^\top$.

Alg 7. RLSMF

1 *for* $t \leftarrow 1$ *to* m *do*
  input: $\widehat{X}(t-1)$, $P(t-1)$, $A(t)$, $B(t)$, Λ
2 $L(t) = \dfrac{(P(t-1) \cdot A(t)^\top) \oslash \Lambda}{1 + \sum(P(t-1) \cdot (A(t)^\top \circledcirc 2) \oslash \Lambda)}$
3 $\widehat{X}(t) = \widehat{X}(t-1) + L(t)(B(t) - A(t) \cdot \widehat{X}(t-1))$
4 $P(t) = \mathrm{diag}(\mathrm{main}((I - L(t) \cdot A(t))P(t-1)) \oslash \Lambda)$
  output: $\widehat{X}(t)$, $P(t)$

F. Parameter Projection

Parameter projection improves robustness if prior knowledge about the reasonable range for some or all parameters is available. In many cases, a non-zero or larger-than-zero condition makes sense and often, there is specific prior knowledge to set lower and upper bounds for each parameter. Timmons et al. [25] showed that orthogonal parameter projection reduces to a simple saturator when constraints are given for each parameter. These user-defined constraints are considered by $\mathfrak{X}$ in Alg. 8. More sophisticated parameter projection in closed-loop applications is discussed in [25, 26].

Alg. 8 can be used in conjunction with RLS and RLSMF. So far, we have no solution to include any parameter projection in the presented SVD-based TLS schemes.

Alg 8. Parameter projection

input: $\widehat{X}(t),\ \mathfrak{X}$

for $i \leftarrow 1$ *to* n *do*

  if $\widehat{X}_i(t) < \mathfrak{X}_{i,1}$ *then*

    $\widehat{X}_i(t) = \mathfrak{X}_{i,1}$

  else if $\widehat{X}_i(t) > \mathfrak{X}_{i,2}$ *then*

    $\widehat{X}_i(t) = \mathfrak{X}_{i,2}$

output: $\widehat{X}(t)$

G. Simulation Example

A three-input one-output channel model was simulated to verify RGTLS (Alg. 5) with TLS (Alg. 1). We created uniformly distributed input data A with Matlab's `unifrd` command in the range of $(-20 \ldots 20)$ for $n = 3,\ m = 3000$ and use $\overline{X} = [3\ 12\ 8]^\top$ for the first half of our data $t = 1 \ldots m/2$. For $t = m/2+1 \ldots m$ we set $\overline{X} = [6\ 12\ 8]^\top$ to perform a step in parameter X_1. Then, we compute the output with $\overline{A} \cdot \overline{X} = \overline{B}$. White Gaussian noise with different noise power was added to $[\overline{A}\ \overline{B}]$ with `wgn` to get noisy data $[A\ B]$. The noise covariance matrix is shown in (8).

$$\operatorname{cov}([\widetilde{A}\ \widetilde{B}]) = \begin{bmatrix} 6.4210 & 0.0286 & -0.2366 & 0.0718 \\ 0.0286 & 2.5874 & -0.0367 & -0.0124 \\ -0.2366 & -0.0367 & 16.1123 & -0.1608 \\ 0.0718 & -0.0124 & -0.1608 & 10.6422 \end{bmatrix} \quad (8)$$

RGTLS (Alg. 5) was used with $W = I$ to meet the TLS case and λ was fixed at $\lambda = 0.997$. TLS (Alg. 1) was executed in a loop from $t = 1 \ldots m$. The data was scaled row-wise with $\Lambda = [\lambda^{t-1}\ \lambda^{t-2}\ \ldots\ \lambda^{t-t}]^\top$ inside each loop. The result is presented in Fig. 2 showing the high accuracy of RGTLS. After roughly 500 samples, there is virtually no difference between RGTLS and TLS to see in Fig. 2 (a) and this is exactly what we want. The parameter variance differs only in the first few samples in Fig. 2 (b). The speed-up of Alg. 5 compared with Alg. 1 is remarkable.

Now, we compare RGTLS with RLS in Fig. 3 in terms of the relative estimation error (zero is desired). In RGTLS, W was fixed to $W = \operatorname{cov}([\widetilde{A}\ \widetilde{B}])$ taken from (8). Both estimators use $\lambda = 0.997$. The step in $\overline{X}_1$ at $t = 1500$ affects both estimators. However, RLS seems more sensitive than RGTLS. In addition to that, RLS gives biased estimates in $\widehat{X}_{3\mathrm{RLS}}$. That is explainable with the large noise variance of $\operatorname{cov}([\widetilde{A}\ \widetilde{B}])_{3,3}$ in (8). The estimates of RGTLS have slightly larger variances. There is no big difference in computational time for this amount of data.

Finally, we use RGTLS in a RLS sense by setting $W = \operatorname{diag}([10^{-5}\ 10^{-5}\ 10^{-5}\ 10^{5}])$ in Fig. 4. That means we assume that all inputs have negligible noise variance compared with the output. Apart from $\widehat{X}_1$ around the parameter step ($t = 1500$), RGTLS shows very similar results to RLS. By this, we can set W in an intuitive way and take advantage of RGTLS if prior knowledge of the noise covariance is available. At the moment, an extension of RGTLS to multiple forgetting like in RLSMF is not available.

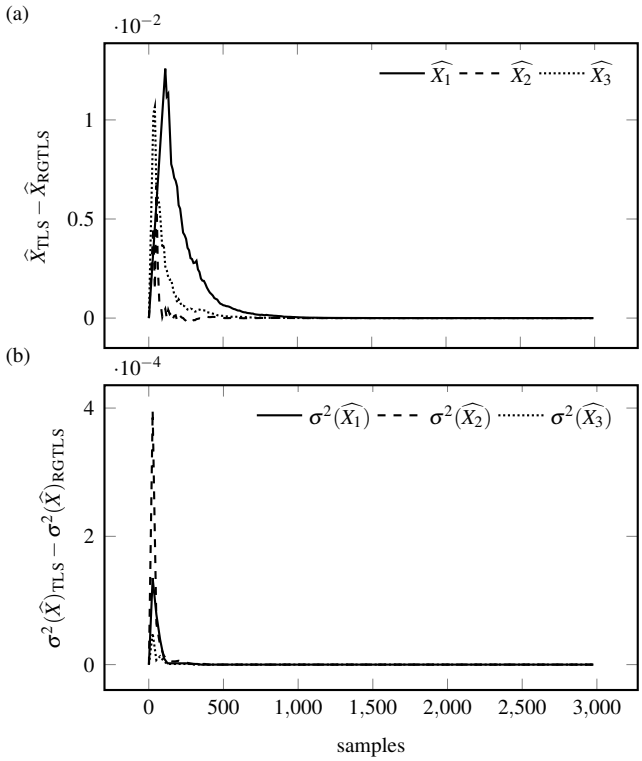

Fig. 2: Validation of RGTLS with $W = I$ (Alg. 5) with TLS (Alg. 1). Alg. 5 took 1.35 seconds, while Alg. 1 required 259 seconds on an Intel P8600 CPU.

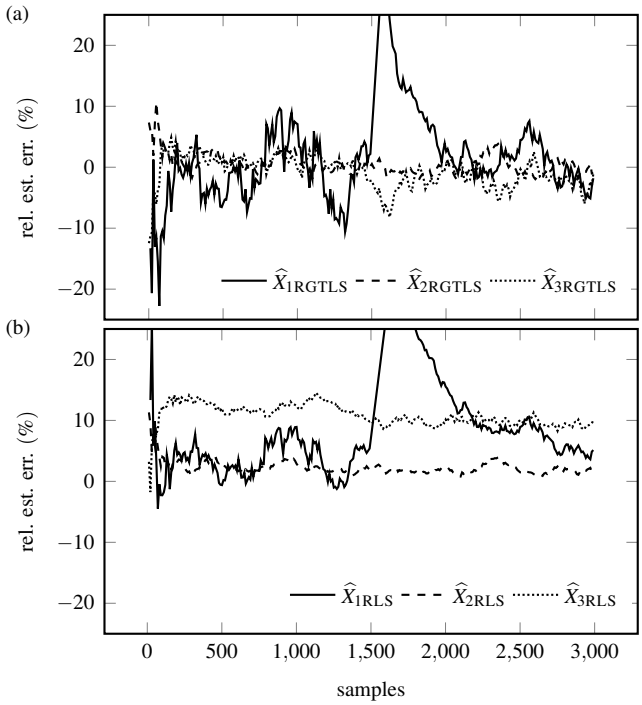

Fig. 3: Relative estimation error of RGTLS with $W = \operatorname{cov}([\widetilde{A}\ \widetilde{B}])$ (Alg. 5) in Fig. 3 (a) and RLS (Alg. 6) in Fig. 3 (b). Alg. 5 took 1.24 seconds, while Alg. 1 required 0.98 seconds on an Intel P8600 CPU.

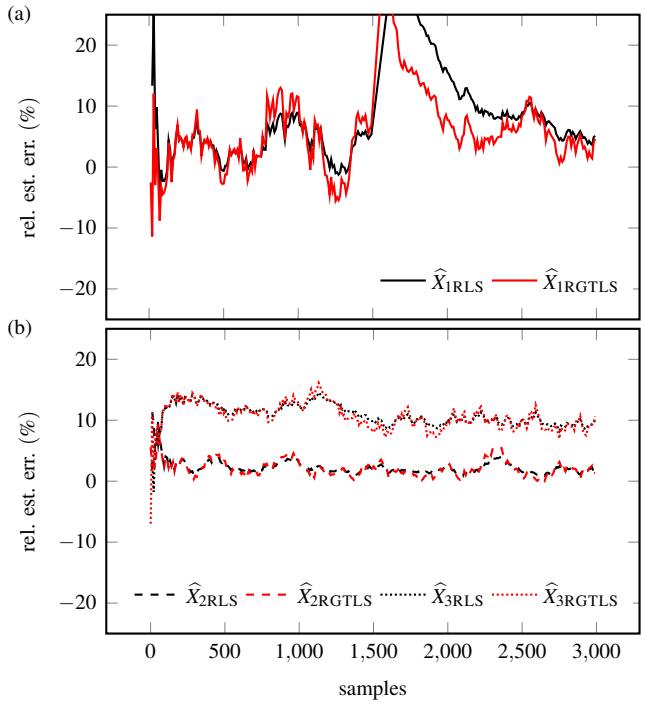

Fig. 4: Relative estimation error of RGTLS with $W = \mathrm{diag}([10^{-5}\ 10^{-5}\ 10^{-5}\ 10^{5}])$ (Alg. 5) and RLS (Alg. 6).

IV. DRIVING RESISTANCE ESTIMATION

For convenience of the reader, Sec. IV-A briefly reviews the transition from a vehicle longitudinal dynamics model into a MISO system. More detailed information can be found in [22].

A. Vehicle Longitudinal Dynamics Model

We take advantage of the force equilibrium (9) between the tractive force F on the left-hand side and the sum of rolling resistance, climbing resistance, aerodynamic drag, and acceleration resistance, that is known as total driving resistance.

$$\begin{aligned} F =& m \cdot g \cdot \cos\alpha \cdot v(f_{\mathrm{r0}}/v + f_{\mathrm{r1}}) + m \cdot g \cdot \sin\alpha \\ &+ c_{\mathrm{x}} \frac{\rho}{2} A \cdot v^2 + (m + m_{\mathrm{rot}}) a_{\mathrm{x}}, \end{aligned} \tag{9}$$

In (9), α, v, ρ, g, and a_{x} are the road-grade, velocity, air density, gravity acceleration, and longitudinal acceleration respectively.

One can decompose from (9) a vector of unknown time-varying parameters

$$X = \begin{bmatrix} m \cdot f_{\mathrm{r0}} & m & m \cdot f_{\mathrm{r1}} & A \cdot c_{\mathrm{x}} \end{bmatrix}^\top, \tag{10}$$

with the vehicle mass m, the coefficients of rolling resistance $f_{\mathrm{r0,1}}$, the drag coefficient c_{x} and the vehicle cross-sectional A. Note that m_{rot} in (9) is known by the use of gear, dynamic rolling radius and reduced moment of inertia.

CAN signals of F, α, v, ρ, and a_{x} form adequate input and output data to feed the MISO model

$$\underbrace{\begin{bmatrix} A_{11} & \dots & A_{1n} \\ \vdots & \vdots & \vdots \\ A_{m1} & \dots & A_{mn} \end{bmatrix}}_{\text{inputs}} \underbrace{\begin{bmatrix} X_1 & \dots & X_n \end{bmatrix}^\top}_{\text{parameters}} \approx \underbrace{\begin{bmatrix} B_1 \\ \vdots \\ B_m \end{bmatrix}}_{\text{output}}. \tag{11}$$

All CAN signals are read at 100 hertz frequency and run through a third-order Butterworth filter with 1 hertz pass frequency and 10 hertz stop frequency. A simple Boolean logic excludes harmful driving states that are not considered in (9), such as braking. Moreover, we use an outlier detection scheme that excludes additional adverse driving states automatically.

B. Outlier Detection

Knorr and Ng [27] introduced several distance-based outlier detection schemes that are optimized for fast batch computation of large data sets. Their method is purely data-driven, without regression or any kind of model. Furthermore, their method can handle various assumptions of the data distribution (Gaussian, Student-t, ...) by appropriate setting of only two parameters.

The basic idea in distance-based outlier detection is to compute the Euclidean distance between each sample ($Z_{i,:}$) and the rest of the data set. The number of times when this distance is smaller than a threshold corresponds to the number of neighbors for this individual sample. Finally, the observed sample ($Z_{i,:}$) is no outlier if the number of neighbors exceed a second threshold. If the second condition fails, $Z_{i,:}$ is marked as outlier.

We use the nested-loop scheme among the algorithms presented in [27] and refer the reader to the corresponding pseudo code section in this remarkable reference. We normalize the data with the standard score $Z' = (Z - \mu(Z))/\sigma(Z)$ before applying nested loop. The recursive sample mean is given by (12), [28] and the recursive sample covariance by (13), [29, p. 19].

$$\mu(Z(t)) = \frac{(t-1)}{t} \cdot \mu(Z(t-1)) + \frac{1}{t} \cdot Z(t) \tag{12}$$

$$\mathrm{cov}(Z(t)) = \mathrm{cov}(Z(t-1)) + \frac{1}{t} \cdot (Z(t)^\top Z(t) - \mathrm{cov}(Z(t-1)) \tag{13}$$

Nested loop is performed in batch mode on a small number of samples (buffer size m) to detect outliers and remove them before we apply one of the estimators from Sec. III. We accept the delay of m samples in the parameter estimates due to the partition of the data in small batches. This procedure fails presumably if the number of outliers is too large in respect to m. Further work is needed to find a recursive distance-based outlier detector.

V. RESULTS WITH EXPERIMENTAL DATA

We performed several test runs with a grand touring sports car on a 22.9 kilometer long public road. The track offers rich variation in road grade and curvature, while the maximum allowed velocity is 100 kilometers per hour. We used a CAN

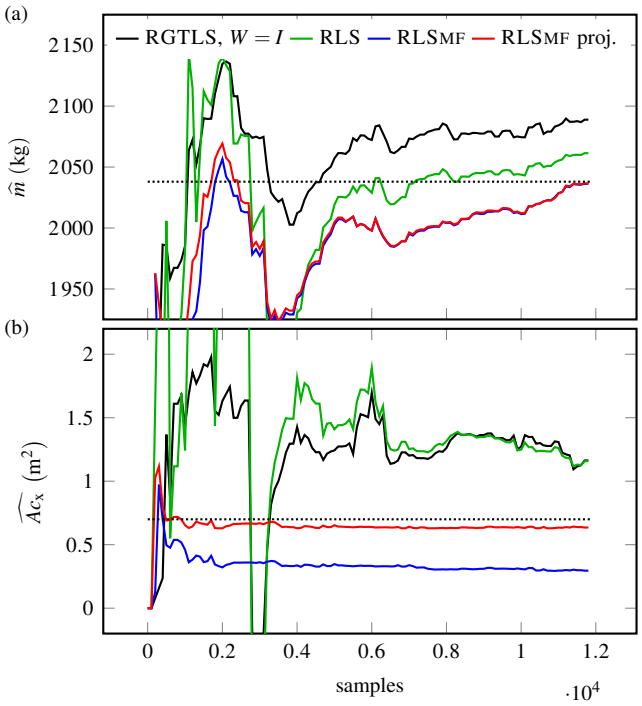

Fig. 5: Estimated mass $\hat{m}$ in Fig. 5 (a) and $\widehat{Ac_x}$ in Fig. 5 (b). ····· shows the true parameters m and Ac_x, respectively.

logger to record full sets of CAN signals in order to execute the presented algorithms afterwards. The following results were obtained by Matlab computations of one test run. In addition, we successfully implemented the presented online-capable algorithms on hardware.

A. Mass and Drag Coefficient Estimates

As noted, we assume that m, A, c_x have negligible variations in time. Hence, we desire smooth and constant estimates for these parameters. Furthermore, we know the true parameters through balancing the vehicle prior to the test run and from wind tunnel measurements. Fig. 5 compares the results for $\hat{m}$ in Fig. 5 (a) and $\widehat{Ac_x}$ in Fig. 5 (b) of four different estimators. In particular, we use:

1) RGTLS with $W = I$ in the TLS sense (Alg. 5);
2) RLS (Alg. 6);
3) RLSMF (Alg. 7);
4) RLSMF with parameter projection; (Alg. 7 with Alg. 8).

First, we notice that RLS outperforms RGTLS and RLSMF with or without parameter projection in the mass estimates in Fig. 5 (a). However, RLS tends to overestimate the mass from sample 10^4 on. Notice that the parameter projection has practically no influence on RLSMF after approximately 3000 samples. RGTLS computes the smoothest results with the smallest break down around sample 3000. Apart from this, the curves of RLS and RGTLS show a similar trend. RLSMF gives high accurate mass estimates at the end of the data set.

The ranking changes in Fig. 5 (b). Now, RLSMF with parameter projection is superior and we clearly observe the benefit of the projection. Note that we use a lower bound for $\widehat{Ac_x}$ near the true value. Nevertheless, RLSMF gives smooth results as well. RLS and RGTLS produce roughly similar results from sample 6000 on, where both estimators show a large bias. The poor results of RLS, RGTLS, and RLSMF correspond to the results of [3]. Bae, Ryu, and Gerdes [3] used RLS with a similar model and explained the poor drag coefficient results with the narrow speed range that does not permit a precise separation between velocity-dependent and velocity-independent parameters. We observed a heavily one-sided velocity distribution in our test runs and suppose that this non-uniform distribution introduces an undesired weighting towards small velocities. Further work is required to compensate this negative weighting effect.

B. Estimates for Coefficients of Rolling Resistance

As mentioned above, we suppose some variation in time for the coefficients of rolling resistance. Time-dependent exact values for f_{r0} and f_{r1} are not available due to several environmental influences. However, we know the value of f_{r0} from standard rolling resistance measurements and expect $\widehat{f_{r0}}$ to be in the same range.

Fig. 6 gives the results for $\widehat{f_{r0}}$ in Fig. 6 (a) and $\widehat{f_{r1}}$ in Fig. 6 (b) for the same set of estimators as used in Sec. V-A. RLSMF with and without parameter projection gives reasonable results for $\widehat{f_{r0}}$ in Fig. 6 (a). As in Fig. 5 (b), RGTLS and RLS show similar results from sample 6000 on that are unlikely large. RGTLS and RLS fail entirely in the estimation of $\widehat{f_{r1}}$ in Fig. 6 (b) by showing negative values. RLSMF with and without parameter projection gives reasonable estimates above zero that range approximately one decade below $\widehat{f_{r0}}$.

VI. CONCLUSIONS

The presented RGTLS estimator with exponential forgetting can treat basic TLS and the extensions WTLS and GTLS in real-time by appropriate setting of the weighting matrix W. Further, the simulations show that RGTLS performs RLS-similar results by intuitive adjustment of W. We present a flexible algorithmic structure that builds on an interchangeable SVD update scheme and calls for clearly arranged nested functions. If prior knowledge of the noise covariance matrix is present, RGTLS outperforms RLS in our simulations.

On the other hand, RGTLS and RLS fail in the estimation of particular driving resistance parameters such as the coefficients of rolling resistance. This is presumably caused through unobserved outliers and the mixture of time-constant and time-varying parameters. Overall RLSMF with parameter projection shows superior accuracy in the estimation of vehicle mass, drag coefficient, and coefficients of rolling resistance. Sufficient robustness is added due to a basic parameter saturator. An equivalent parameter projection approach for RGTLS is desirable in the future.

ACKNOWLEDGMENT

The authors would like to thank Zoltán Fuszenecker and Markus Knobloch for their support in programming.

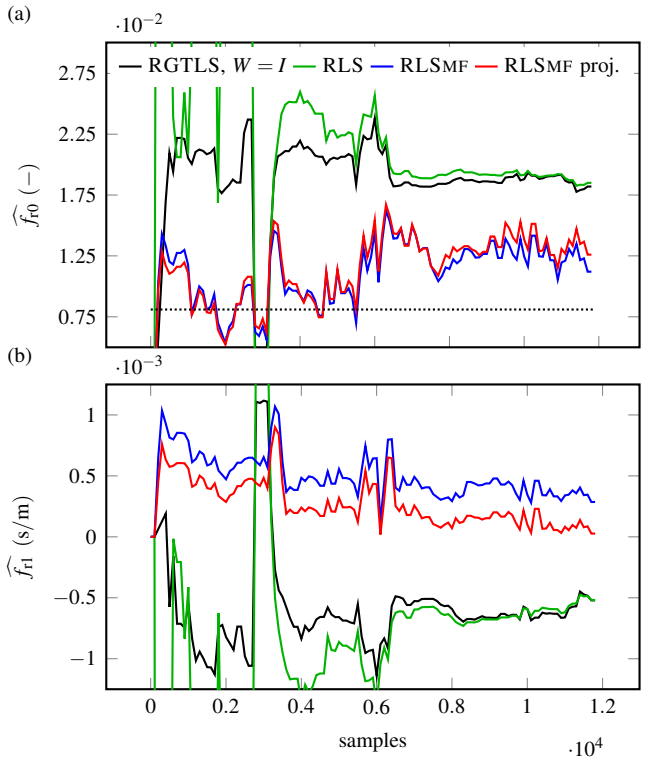

Fig. 6: Estimated coefficient of rolling resistance $\widehat{f_{r0}}$ in Fig. 6 (a) and $\widehat{f_{r1}}$ in Fig. 6 (b). ····· shows the value of f_{r0} from rolling resistance measurements.

References

[1] H.K. Fathy, D. Kang, and J.L. Stein. "Online vehicle mass estimation using recursive least squares and supervisory data extraction." In: *American Control Conference, 2008*. 2008, pp. 1842–1848.

[2] A. Vahidi, A. Stefanopoulou, and H. Peng. "Recursive least squares with forgetting for online estimation of vehicle mass and road grade: theory and experiments." In: *Vehicle System Dynamics* 43.1 (2005), pp. 31–55.

[3] H.S. Bae, J. Ryu, and J.C. Gerdes. "Road grade and vehicle parameter estimation for longitudinal control using GPS." In: *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*. 2001, pp. 166–171.

[4] I. Markovsky and S. Van Huffel. "Overview of total least-squares methods." In: *Signal Processing* 87.10 (2007), pp. 2283–2302.

[5] T. Söderström. "Errors-in-variables methods in system identification." In: *Automatica* 43.6 (2007), pp. 939–958.

[6] S. Van Huffel and J. Vandewalle. *The total least squares problem : computational aspects and analysis*. Frontiers in applied mathematics ; 9. Philadelphia: SIAM, 1991.

[7] S. Van Huffel and J. Vandewalle. "Algebraic connections between the least squares and total least squares problems." In: *Numerische Mathematik* 55.4 (1989), pp. 431–449.

[8] G. Cirrincione and M. Cirrincione. *Neural Based Orthogonal Data Fitting: The EXIN Neural Networks*. Vol. 38. Wiley, 2010.

[9] S. Van Huffel and J. Vandewalle. "Analysis and Properties of the Generalized Total Least Squares Problem $AX \approx B$ When Some or All Columns in A are Subject to Error." In: *SIAM Journal on Matrix Analysis and Applications* 10.3 (1989), pp. 294–315.

[10] I. Markovsky et al. "The element-wise weighted total least-squares problem." In: *Computational Statistics & Data Analysis* 50.1 (2006), pp. 181–209.

[11] B. Schaffrin and A. Wieser. "On weighted total least-squares adjustment for linear regression." In: *Journal of Geodesy* 82 (7 2008), pp. 415–421.

[12] Y. Shen, B. Li, and Y. Chen. "An iterative solution of weighted total least-squares adjustment." In: *Journal of Geodesy* 85 (4 2011), pp. 229–238.

[13] M. Schuermans et al. "On the equivalence between total least squares and maximum likelihood PCA." In: *Analytica Chimica Acta* 544 (2005), pp. 254–267.

[14] M. Gu and S.C. Eisenstat. "A stable and fast algorithm for updating the singular value decomposition." In: *New Haven: Yale University Department of Computer Science. RR-939* (1993).

[15] M. Gu and S.C. Eisenstat. "Downdating the Singular Value Decomposition." In: *SIAM Journal on Matrix Analysis and Applications* 16.3 (1995), pp. 793–810.

[16] M. Brand. "Incremental Singular Value Decomposition of Uncertain Data with Missing Values." In: *Computer Vision ECCV 2002*. Ed. by Anders Heyden et al. Vol. 2350. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2002, pp. 707–720.

[17] M. Brand. "Fast low-rank modifications of the thin singular value decomposition." In: *Linear algebra and its applications* 415.1 (2006), pp. 20–30.

[18] D. Kubus, T. Kroger, and F.M. Wahl. "On-line estimation of inertial parameters using a recursive total least-squares approach." In: *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*. Sept. 2008, pp. 3845–3852.

[19] C.E. Davila. "An efficient recursive total least squares algorithm for FIR adaptive filtering." In: *Signal Processing, IEEE Transactions on* 42.2 (Feb. 1994), pp. 268–280.

[20] J. Lim, N. Choi, and K. Sung. "Robust Recursive TLS (Total Least Square) Method Using Regularized UDU Decomposed for FNN (Feedforward Neural Network) Training." In: *Advances in Neural Networks - ISNN 2005*. Ed. by Jun Wang, Xiaofeng Liao, and Zhang Yi. Vol. 3496. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2005, pp. 133–137.

[21] S. Van Huffel. *Recent advances in total least squares techniques and errors-in-variables modeling*. Vol. 1996. Siam, 1997.

[22] S. Rhode and F. Gauterin. "Vehicle mass estimation using a total least-squares approach." In: *Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference on*. Sept. 2012, pp. 1584–1589.

[23] L. Ljung. *System identification : theory for the user*. 2. ed. Prentice-Hall information and system sciences series. Upper Saddle River, NJ: Prentice Hall, 1999.

[24] S. Saelid and B. Foss. "Adaptive controllers with a vector variable forgetting factor." In: *Decision and Control, 1983. The 22nd IEEE Conference on*. Vol. 22. Dec. 1983, pp. 1488–1494.

[25] W. D. Timmons et al. "Parameter-Constrained Adaptive Control." In: *Industrial & Engineering Chemistry Research* 36.11 (1997), pp. 4894–4905.

[26] M.A. Akella and K. Subbarao. "A novel parameter projection mechanism for smooth and stable adaptive control." In: *Systems & Control Letters* 54.1 (2005), pp. 43–51.

[27] E.M. Knorr and R.T. Ng. "Algorithms for mining distance-based outliers in large datasets." In: *Proceedings of the International Conference on Very Large Data Bases*. 1998, pp. 392–403.

[28] B.P. Welford. "Note on a Method for Calculating Corrected Sums of Squares and Products." In: *Technometrics* 4.3 (1962), pp. 419–420.

[29] S.J. Orfanidis. *Optimum signal processing: An introduction*. 2. nd. Rutgers University, 2007.