Available online at www.sciencedirect.com
SCIENCE DIRECT®
Journal of Computational and Applied Mathematics 180 (2005) 311–331
JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS
www.elsevier.com/locate/cam

High-performance numerical algorithms and software for structured total least squares

Ivan Markovsky*, Sabine Van Huffel

K.U.Leuven, ESAT-SCD, Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium

Received 15 June 2004; accepted 2 November 2004

Abstract

We present a software package for structured total least-squares approximation problems. The allowed structures in the data matrix are block-Toeplitz, block-Hankel, unstructured, and exact. Combination of blocks with these structures can be specified. The computational complexity of the algorithms is $O(m)$, where m is the sample size. We show simulation examples with different approximation problems. Application of the method for multivariable system identification is illustrated on examples from the database for identification of systems DAISY.

Keywords: Parameter estimation; Structured total least squares; Deconvolution; System identification; Numerical linear algebra

1. Introduction

The structured total least squares (STLS) problem is defined as the total least squares (TLS) problem [10,27]

$$\min_{\Delta A, \Delta B, X} \|[\Delta A \;\; \Delta B]\|_{\mathrm{F}}^2 \quad \text{s.t.} \quad (A - \Delta A)X = B - \Delta B \tag{1}$$

with the additional constraint that the correction matrix $[\Delta A \;\; \Delta B]$ has the same structure as the data matrix $[A \;\; B]$. A typical example where a structured system of equations arises is the difference equation

$$R_0 w(t) + R_1 w(t+1) + \cdots + R_l w(t+l) = 0. \tag{2}$$

* Corresponding author. Tel.: +32 1632 17 10; fax: +32 1632 19 70.
E-mail address: Ivan.Markovsky@esat.kuleuven.ac.be (I. Markovsky).

0377-0427/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.cam.2004.11.003

For $t = 1, \ldots, T - l$, the difference equation (2) is equivalent to the structured system of equations $R\mathscr{H}_{l+1}(w) = 0$, where $R := [R_0 \;\; R_1 \;\; \cdots \;\; R_l]$ and $\mathscr{H}_{l+1}(w)$ is the block-Hankel matrix

$$\mathscr{H}_{l+1}(w) := \begin{bmatrix} w(1) & w(2) & \cdots & w(T-l) \\ w(2) & w(3) & \cdots & w(T-l+1) \\ \vdots & \vdots & & \vdots \\ w(l+1) & w(l+2) & \cdots & w(T) \end{bmatrix}.$$

1.1. History of the problem

The origin of the STLS problem dates back to the work of Aoki and Yue [3], although the name "structured total least squares" appeared only 23 years later in the literature [7]. Aoki and Yue consider a single input single output system identification problem, where both the input and the output are noisy (errors-in-variables setting) and derive a maximum likelihood solution. Under the normality assumption for the measurement errors, a maximum likelihood estimate turns out to be a solution of the STLS problem. Furthermore, Aoki and Yue approach the optimization problem in a similar way to the one we adopt: they use classical nonlinear least-squares minimization methods for solving an equivalent unconstrained problem.

The STLS problem occurs frequently in signal processing applications. Cadzow [5], Bresler and Macovski [4] propose heuristic solution methods that turn out to be suboptimal [8, Section V] with respect to the STLS criterion. These methods, however, became popular because of their simplicity. For example, the method of Cadzow is an iterative method that alternates between unstructured low rank approximation and structure enforcement, thereby only requiring singular value decomposition (SVD) computations and manipulation of the matrix entries.

Abatzoglou et al. [2] are considered to be the first who formulated an STLS problem. They called their approach constrained total least squares (CTLS) and motivate the problem as an extension of the TLS method to matrices with structure. The solution approach adopted in [2] is closely related to the one of Aoki and Yue. Again an equivalent optimization problem is derived, but it is solved numerically via a Newton-type optimization method.

Shortly after the publication of the work on the CTLS problem, De Moor lists many applications of the STLS problem and outlines a new framework for deriving analytical properties and numerical methods [7]. His approach is based on the Lagrange multipliers and the basic result is an equivalent problem, called Riemannian singular value decomposition (RiSVD), that can be considered as a "nonlinear" extension of the classical SVD. As an outcome of the new problem formulation, an iterative solution method based on the inverse power iteration is proposed.

Another algorithm for solving the STLS problem (even with ℓ_1 and ℓ_∞ norm in the cost function), called structured total least norm (STLN), is proposed by Rosen et al. [23]. In contrast to the approaches of Aoki et al., Rosen et al. solve the problem in its original formulation. The constraint is linearized around the current iteration point, which results in a linearly constrained least-squares problem. In the algorithm of [23], the constraint is incorporated in the cost function by adding a multiple of its residual norm.

All problem formulations and solution methods cited above have been proven to be equivalent [14] but only consider univariate STLS problems, i.e., STLS problems with one right-hand side vector B. They all aim to reduce the rank of the augmented data matrix $C := [A \;\; B]$ with one. A multivariate version of the

algorithm of Rosen et al. is proposed in [25]. It involves, however, Kronecker products that unnecessary inflate the dimension of the involved matrices.

When dealing with a general affine structure the CTLS, RiSVD, and STLN methods have computational complexity $O(m^3)$. Fast algorithms with computational complexity $O(m)$ are proposed by Mastronardi et al. [15,21] for special STLS problems: $[A \;\; b]$ Hankel and A Toeplitz, b unstructured. They use the STLN approach but recognize that a matrix appearing in the kernel subproblem of the algorithm has low displacement rank. This is exploited via the Schur algorithm.

1.2. Motivation for our work

The STLS methods outlined above point out the following issues:

Structure: the structure specification for the augmented data matrix $[A \;\; B]$ varies from general affine [2,7,23] to specific affine, like Hankel/Toeplitz [15], or Hankel/Toeplitz block augmented with an unstructured column [21],

Rank reduction: all published methods, except the one of [25], reduce the rank of the augmented data matrix $[A \;\; B]$ by one,

Computational efficiency: the efficiency varies from $O(m^3)$ for the methods that use a general affine structure to $O(m)$ for the fast methods of [15,21] that use a Hankel/Toeplitz-type structure.

No efficient algorithms exist for multivariate problems. In addition, the proposed methods lack a numerically reliable and robust software implementation that would enhance their use in real-life applications. Due to the above reasons the STLS methods, although attractive for theoretical studies and relevant for applications, did not penetrate in real-life problems.

The motivation for our work is to make the STLS method practically useful by deriving algorithms that are general enough for various applications and computationally efficient for non-toy examples. We complement the theoretical study by a robust software implementation. Therefore, we present in this paper a numerically efficient algorithm together with a robust software implementation for solving a variety of STLS problems.

1.3. Outline of the paper

In Section 2, we define formally the class of problems solved by the package and describe the solution method used. Section 3 outlines the proposed algorithm. The implementation details necessary to use the software package are given in Appendix A. In Section 4, we compare the performance of the STLS package with that of alternative solution methods in several classical estimation problems. Section 5 describes an application of the package in system identification and shows simulation results with real-life and simulated data sets from the data base for system identification DAISY.

2. Problem formulation and solution method

The STLS problem was introduced in Section 1 as the TLS problem with an additional constraint on the correction matrix $[\Delta A \;\; \Delta B]$. Now we further specify this informal definition. The general STLS problem

that we are going to consider is

$$\hat{X} = \arg \min_{X,\ \Delta p} \|\Delta p\|_2^2 \quad \text{s.t.} \quad \mathscr{S}(p - \Delta p) \begin{bmatrix} X \\ -I_d \end{bmatrix} = 0, \tag{3}$$

where $X \in \mathbb{R}^{n\times d}$ is a parameter of interest, $p \in \mathbb{R}^{n_p}$ is a data vector, and $\mathscr{S}(p) \in \mathbb{R}^{m\times(n+d)}$ is a structured matrix

$$\mathscr{S}(p) = [C^{(1)} \cdots C^{(q)}], \quad \text{for all } p \in \mathbb{R}^{n_p}, \text{ with } C^{(l)},\ l = 1, \ldots, q, \begin{cases} \mathsf{T} & \text{block-Toeplitz,} \\ \mathsf{H} & \text{block-Hankel,} \\ \mathsf{U} & \text{unstructured, or} \\ \mathsf{E} & \text{exact.} \end{cases} \tag{4}$$

All block-Toeplitz and block-Hankel structured blocks $C^{(l)}$ have blocks of the same row dimension K and possibly different column dimensions t_l. The parameter vector p uniquely specifies the augmented data matrix. For example, if $[A\ \ B]$ is Toeplitz

$$[A\ \ B] = \begin{bmatrix} p_{m+d+n} & p_{m+d+n-1} & \cdots & p_2 & p_1 \\ p_{m+d+n+1} & p_{m+d+n} & & & p_2 \\ \vdots & & & & \vdots \\ p_{m-1} & & & & p_{m-n-d} \\ p_m & p_{m-1} & \cdots & & p_{m-n-d+1} \end{bmatrix},$$

where $p = \operatorname{col}(p_1, p_2, \ldots, p_{m+n+d-1}) \in \mathbb{R}^{n_p}$ with $n_p = m + n + d - 1$.

The link with the formulation in (1) is

$$[A\ \ B] = \mathscr{S}(p) \quad \text{and} \quad [\Delta A\ \ \Delta B] = \mathscr{S}(\Delta p). \tag{5}$$

The constraint that $[A\ \ B]$ and $[\Delta A\ \ \Delta B]$ have the same structure is enforced by (5) and the search for the optimal solution is over the parameters Δp that uniquely specify the correction.

The structure of $\mathscr{S}(p)$ is specified by the scalar K, and the array $\mathscr{D} \in ((\mathsf{T}, \mathsf{H}, \mathsf{U}, \mathsf{F}) \times \mathbb{N} \times \mathbb{N})^q$ that describes the structure of the blocks $\{C^{(l)}\}_{l=1}^q$; $\mathscr{D}_l$ specifies the block $C^{(l)}$ by giving its type $\mathscr{D}_l(1)$, the number of columns $n_l = \mathscr{D}_l(2)$, and if applicable, the number of columns $t_l = \mathscr{D}_l(3)$ of a block in a block-Toeplitz/Hankel block $C^{(l)}$. The input data for the problem is the vector p (alternatively the matrix $\mathscr{S}(p)$) and the structure specification K, $\mathscr{D}$. The desired solution is a value $\hat{X}$ of X at a global optimum point of the optimization problem (3).

An equivalent *unconstrained* optimization problem to (3)–(4) is derived by minimizing analytically over the correction Δp, see [17, Section 2],

$$\min_X f_0(X), \quad f_0(X) := r^\top(X)\Gamma^{-1}(X)r(X), \quad r(X) := \operatorname{vec}\left(\left(\mathscr{S}(p)\begin{bmatrix} X \\ -I_d \end{bmatrix}\right)^\top\right), \tag{6}$$

where the weight matrix $\Gamma(X)$ has block-Toeplitz and block-banded structure, see [17, Section 3],

$$\Gamma(X) = \begin{bmatrix} \mathbf{\Gamma}_0 & \mathbf{\Gamma}_{-1} & \cdots & \mathbf{\Gamma}_{-s} & & 0 \\ \mathbf{\Gamma}_1 & \ddots & \ddots & \ddots & \ddots & \\ \vdots & \ddots & \ddots & \ddots & \ddots & \mathbf{\Gamma}_{-s} \\ \mathbf{\Gamma}_s & \ddots & \ddots & \ddots & \ddots & \vdots \\ & \ddots & \ddots & \ddots & \ddots & \mathbf{\Gamma}_{-1} \\ 0 & & \mathbf{\Gamma}_s & \cdots & \mathbf{\Gamma}_1 & \mathbf{\Gamma}_0 \end{bmatrix} \in \mathbb{R}^{md \times md}.$$

The block size is dK and the upper/lower block bandwidth s equals the maximum number of block columns in a block-Hankel/Toeplitz structured block $C^{(i)}$.

Transforming the original problem (3)–(4) to the equivalent one (2) is a key step in our solution approach. The equivalent problem is an unconstrained optimization problem that has as decision variables only the nd element of X. In contrast, the original problem is a constrained optimization problem and has as additional decision variables the n_p elements of Δp. For the applications that we aim at, $n_p \gg nd$, which makes the elimination of Δp a huge complexity reduction. The same elimination step is used also in the CTLS and RiSVD approaches for the case of univariate STLS problems.

The second key step in our solution approach is the exploitation of the structure in the weight matrix $\Gamma(X)$ for efficient cost function and first derivative evaluation. The cost function evaluation requires solving the system of equations $\Gamma(X)y(X) = r(X)$ and computing the inner product $f_0(X) = r^\top(X)y(X)$. Straightforward implementation of these steps results in $\mathrm{O}(m^3)$ floating point operations (flops). By solving the system of equations, taking into account the structure of $\Gamma(X)$, we reduce the computational complexity to $\mathrm{O}(m)$. In addition, the first derivative $f_0'(X)$ can be computed with the same computational complexity. The numerical efficiency of the method with respect to n and d, however, is $\mathrm{O}((nd)^3)$. Therefore, it is suitable for applications with $m \gg nd$.

Caveat: The STLS problem is nonconvex. By using a local optimization method there is no guarantee that a global minimum point is found. Upon convergence, the computed solution is (up to the provided convergence tolerance) *locally* optimal. The convergence to one local minimum or another depends on the initial approximation used.

3. Algorithm and implementation

The input data for the STLS algorithm is the structure specification $K, \mathscr{D}$ and the data matrix C. The flexible structure specification $\mathscr{D}$ is utilized in the computations for the construction of the weight matrix $\Gamma(X)$. Because of its structure, $\Gamma(X)$ is specified by the nonzero part of its first block row, i.e., by the $s+1$ matrices $\{\mathbf{\Gamma}_k(X)\}_{k=0}^{s}$. In [17, Section 4], we prove that the $\mathbf{\Gamma}_k$'s are quadratic in X. Define the shift matrix

$$J_{n_l} := \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 1 \\ 0 & 0 & \cdots & 0 & 0 \end{bmatrix} \in \mathbb{R}^{n_l \times n_l}$$

and let δ be the Kronecker delta function: $\delta(0) = 1$ and $\delta(k) = 0$ for $k \neq 0$. It can be shown that

$$\mathbf{\Gamma}_k(X) = \mathbf{X_{ext}} W_{\tilde{\mathbf{c}},k} \mathbf{X}_{\mathbf{ext}}^\top, \quad \text{for } k = 0, \ldots, s, \qquad \text{where } \mathbf{X_{ext}} := (I_K \otimes [X^\top \ \ -I]), \tag{7}$$

$$W_{\tilde{\mathbf{c}},k} := \operatorname{blk\,diag}(W_k^{(1)}, \ldots, W_k^{(q)}), \quad \text{and} \quad W_k^{(l)} = \begin{cases} (J_{n_l}^\top)^{t_l k} & \text{if } C^{(l)} \text{ is Toeplitz,} \\ (J_{n_l})^{t_l k} & \text{if } C^{(l)} \text{ is Hankel,} \\ \delta(k) I_{n_l} & \text{if } C^{(l)} \text{ is unstructured,} \\ 0_{n_l} & \text{if } C^{(l)} \text{ is exact.} \end{cases} \tag{8}$$

By computing the $W_{\tilde{\mathbf{c}},k}$'s defined in (8), $\Gamma(X)$ can be constructed for any $X \in \mathbb{R}^{n\times d}$ via (7). Expressions (7) and (8) completely "decode" the structure specification $\mathscr{D}$ and utilize it in the subsequent computations.

Algorithm 1 outlines the step for the construction of the $W_{\tilde{\mathbf{c}},k}$'s. It requires arithmetic operation only for indexing matrix–vector elements. The $s+1$ matrices $\{W_{\tilde{\mathbf{c}},k}\}_{k=0}^s$ are sparse. For the typical applications that we address, however, their dimension $(n+d)K \times (n+d)K$ is relatively small (compared with the row dimension m of the data matrix), so that we do not take into account their structure.

Algorithm 1 From structure specification K, $\mathscr{D}$ to $\{W_{\tilde{\mathbf{c}},k}\}$.

Input structure specification K, $\mathscr{D}$.
Define $s := \max_{l=1,\ldots,q}(\mathbf{n}_l - 1)$, where $\mathbf{n}_l := n_l/t_l$, for block-Toeplitz/Hankel structured block $C^{(l)}$, and $\mathbf{n}_l := 1$, otherwise.
for $k = 1, \ldots, s$ **do**
  for $l = 1, \ldots, q$ **do**
    if $\mathscr{D}_l(1) = =\mathsf{T}$ **then**
      $W_k^{(l)} = (J_{n_l}^\top)^{t_l k}$
    else if $\mathscr{D}_l(1) = =\mathsf{H}$ **then**
      $W_k^{(l)} = (J_{n_l})^{t_l k}$
    else if $\mathscr{D}_l(1) = =\mathsf{U}$ **then**
      $W_k^{(l)} = \delta(k) I_{n_l}$
    else
      $W_k^{(l)} = 0_{n_l}$
    end if
  end for
  $W_{\tilde{\mathbf{c}},k} := \operatorname{blk\,diag}(W_k^{(1)}, \ldots, W_k^{(q)})$
end for
Output $\{W_{\tilde{\mathbf{c}},k}\}_{k=0}^s$ and stop.

Algorithm 2 specifies the steps needed for the cost function and its first derivative evaluation. For details, see [17, Section 5]. The flops per step for Algorithm 2 are:

2. $(n+d)(n+2d)dK^3$
3. $m(n+1)d$
4. msd^2K^2
5. md

6. $msd^2K - s(s+1)d^2K^2/2$
7. $mnd + (2s+1)(nd+n+1)dK^2$

Thus in total $\mathrm{O}(md(sdK^2+n) + n^2dK^3 + 3nd^2K^3 + 2d^3K^3 + 2snd^2K^2)$ flops are required for cost function and first derivative evaluation. Note that the flop counts depend on the structure through s.

Algorithm 2 Cost function and first derivative evaluation.

1: Input: A, B, X, $\{W_{\tilde{\mathbf{c}},k}\}_{k=0}^{s}$.
2: $\mathbf{\Gamma}_k = (I_K \otimes [X^\top \ \ -I])W_{\tilde{\mathbf{c}},k}(I_K \otimes [X^\top \ \ -I])^\top$, for $k = 0, 1, \ldots, s$.
3: $r = \mathrm{vec}((AX-B)^\top)$.
4: Solve $\Gamma y_r = r$ exploiting the block-banded/Toeplitz structure of Γ, e.g., by using the routines `MB02GD` from the SLICOT library and `DPBTRS` from the LAPACK library.
5: $f_0 = r^\top y_r$.
6: If only the cost function evaluation is required, output f_0 and stop.
7: Define $\mathrm{col}(y_{r,1}, \ldots, y_{r,m}) := y_r$, where $y_{r,i} \in \mathbb{R}^d$; $\mathrm{col}(\mathbf{y}_{r,1}, \ldots, \mathbf{y}_{r,\mathbf{m}}) =: y_r$, where $\mathbf{y}_{r,i} \in \mathbb{R}^{dK}$, $\mathbf{m} := m/K$; and $Y_r^\top := [y_{r,1} \ \cdots \ y_{r,m}]$.
8: $\mathbf{N}_k = \sum_{i=1}^{\mathbf{m}-k} \mathbf{y}_{r,i+k}\mathbf{y}_{r,i}^\top$, for $k = 0, 1, \ldots, s$.
9: $f_0' = 2A^\top Y_r - 2\sum_{k=-s}^{s}\sum_{i,j=1}^{K}(W_{\tilde{\mathbf{a}},k,ij}X - W_{\tilde{\mathbf{a}}\tilde{\mathbf{b}},k,ij})\mathbf{N}_{k,ij}^\top$, where $W_{\tilde{\mathbf{c}},k,ij} \in \mathbb{R}^{(n+d)\times(n+d)}$ is the (i,j)th block of $W_{\tilde{\mathbf{c}},k} \in \mathbb{R}^{K(n+d)\times K(n+d)}$, $W_{\tilde{\mathbf{a}},k,ij} \in \mathbb{R}^{n\times n}$; $W_{\tilde{\mathbf{a}}\tilde{\mathbf{b}},k,ij} \in \mathbb{R}^{n\times d}$ are defined as blocks of $W_{\tilde{\mathbf{c}},k,ij}$ as follows

$$W_{\tilde{\mathbf{c}},k,ij} =: \begin{bmatrix} W_{\tilde{\mathbf{a}},k,ij} & W_{\tilde{\mathbf{a}}\tilde{\mathbf{b}},k,ij} \\ W_{\tilde{\mathbf{b}}\tilde{\mathbf{a}},k,ij} & W_{\tilde{\mathbf{b}},k,ij} \end{bmatrix};$$

and $\mathbf{N}_{k,ij} \in \mathbb{R}^{d\times d}$ is the (i,j)th block of $\mathbf{N}_k \in \mathbb{R}^{dK\times dK}$.
10: Output f_0, f_0' and stop.

The overall algorithm for the computation of the STLS solution is Algorithm 3.

Algorithm 3 Algorithm for solving the STLS problem (3).

1: Input: the structure specification K, $\mathscr{D}$ and the matrices A and B.
2: Compute the matrices $\{W_{\tilde{\mathbf{c}},k}\}$ via Algorithm 1.
3: Compute the TLS solution $X^{(0)}$ of $AX \approx B$, by, e.g., the function `MB02MD` from the SLICOT library.
4: Execute a standard optimization algorithm, e.g., the BFGS (Broyden, Fletcher, Goldfarb, and Shanno) quasi-Newton one, for the minimization of f_0 over X with initial approximation $X^{(0)}$ and with cost function and first derivative evaluation, performed via Algorithm 2. Let $\hat{X}$ be the approximation found by the optimization algorithm upon convergence.
5: Output $\hat{X}$ and stop.

The implementation details are given in Appendix A.

4. Simulation examples

In this section, we illustrate the application of the STLS package on standard estimation problems. Our goal is to show the flexibility of the STLS problem formulation (3)–(4). A more realistic application of the STLS package is described in Section 5, where simulation examples on real-life data sets are shown.

The problems listed below are special cases of the block-Toeplitz/Hankel STLS problem, for particular choices of the structure specification $K, \mathscr{D}$. If not given, K and the third element of $\mathscr{D}_l$ are by default equal to ones. In all but one of the examples, we show the computed solution by the STLS package and by an alternative method. Special problems like least squares (LS), TLS, and mixed LS-TLS [27, Section 3.5] should be solved in practice by the corresponding special methods. Still they serve as benchmarks for the STLS package.

All examples are performed in MATLAB 6.0, running on a Linux i686 PC. The scripts, listed below, are included in the STLS package as the demo file `demo.m`.

4.1. Least squares

The least squares problem $AX \approx B$, where $A \in \mathbb{R}^{m\times n}$ is exact and unstructured, and $B \in \mathbb{R}^{m\times d}$ is perturbed and unstructured, is solved as an STLS problem with structure specification $\mathscr{D} = [[\mathtt{F}\ n], [\mathtt{U}\ d]]$. Next we show a simulation example, in which the solution of the STLS package is checked by MATLAB's least squares solver /.

```
>> % Define dimensions and generate random data
>> m = 100; n = 5; d= 2;
>> a = rand(m,n); b = rand(m,d);
>> % Find the LS estimate by Matlab's \
>> tic, x_ls = a\b; t_ls = toc
t_ls =
   0.00157700000000
>> disp(x_ls(1,1:d))
   0.05737144079627 0.22486444701677
>> % Define and solve the LS problem as an STLS problem
>> s_ls = [4 n 1; 3 d 1];
>> tic, x_stls = stls(a,b,s_ls); t_stls = toc
t_stls =
   0.00730900000000
>> disp(x_stls(1,1:d))
   0.05737144079627 0.22486444701677
```

4.2. Total least squares

The TLS problem $AX \approx B$, where the data matrix $C := [A\ B] \in \mathbb{R}^{m\times(n+d)}$ is perturbed and unstructured, is solved as an STLS problem with structure specification $\mathscr{D} = [\mathtt{U}\ n + d]$. Next we show

a simulation example, in which the solution of the STLS package is checked by the function `tls.m` that implements an SVD method for the computation of the TLS solution [10].

```
>> % The data is a,b used above.
>> % Solve the TLS problem via SVD
>> tic, x_tls = tls(a,b); t_tls = toc
t_tls =
   0.00375500000000
>> disp(x_tls(1,1:d))
   −0.49791037472338 0.03277992784515
>> % Define and solve the TLS problem as an STLS problem
>> s_tls= [3 n+d 1];
>> tic, i_stls = stls(a,b,s_tls); t_stls = toc
t_stls=
   0.00504600000000
>> disp(x_stls(1,1:d))
   −0.49791037472338 0.03277992784515
```

4.3. Mixed least squares total least squares

The mixed LS-TLS problem [27, Section 3.5] is defined as follows: $AX \approx B$, where $A = [A_\mathrm{f}\ A_\mathrm{p}]$, $A_\mathrm{p} \in \mathbb{R}^{m\times n_1}$ and $B \in \mathbb{R}^{m\times d}$ are perturbed and unstructured, and $A_\mathrm{f} \in \mathbb{R}^{m\times n_2}$ is exact and unstructured. This problem is solved as an STLS problem with structure specification $\mathscr{D} = [[\mathtt{U}\ n_1], [\mathtt{F}\ n_2], [\mathtt{U}\ d]]$. In [27, Section 3.5] an (exact) SVD-based method for the computation of the mixed LS-TLS solution is proposed. Next we show a simulation example, in which the solution of the STLS package is checked by a MATLAB implementation `lstls.m` of the exact mixed LS-TLS solution method.

```
>> % The data is a,b used above.
>> n1 = 5; % # of column of a1, where a =: [a1 a2] with a1 exact
>> % Solve the mixed LS-TLS problem via exact algorithm
>> tic, x_lstls = ls_tls(a(:,1:n1), a(:, n1+1:end),b); t_lstls = toc
t_lstls=
   0.03171700000000
>> disp(x_lstls(1,1:d))
   0.05737144079627 0.22486444701677
>> % Define and solve the mixed LS-TLS problem as an STLS problem
>> s_lstls = [4 n1 1; 3 n+d-n1 1];
>> tic, [x_stls,i_stls]=stls(a,b,s_lstls); t_stls = toc
t_stls=
   0.00724800000000
>> disp(x_stls(1,1:d))
   0.05737144079627 0.22486444701677
```

In the simulation example above $A = A_\mathrm{f}$. This problem is called data least-squares problem and is studied in [12,6].

4.4. Hankel low-rank approximation

The Hankel low-rank approximation problem [7, Section 4.524], is defined as follows:

$$\min_{\Delta p} \|\Delta p\|_2^2 \quad \text{s.t.} \quad \mathscr{H}(p - \Delta p) \text{ has given rank } n. \tag{9}$$

Here $\mathscr{H}$ is a mapping from the parameter space $\mathbb{R}^{n_p}$ to the set of the $m \times (n + d)$ block-Hankel matrices, with block size $n_y \times n_u$. If the rank constraint is expressed as $\mathscr{H}(\hat{p})\left[\begin{smallmatrix} X \\ -I \end{smallmatrix}\right] = 0$, where $X \in \mathbb{R}^{n \times d}$ is an additional variable, then (9) becomes an STLS problem with $K = n_y$ and $\mathscr{D} = [\mathtt{H}\ n + d\ n_u]$.

Next we show a simulation example for the scalar Hankel low-rank approximation problem, i.e., when $n_y = n_u = 1$. The closely related STLS problem with Toeplitz structured data matrix $\mathscr{H}(p)$ is studied in [15], where an efficient $O(m)$ solution method is proposed. For comparison with the STLS package we use a MATLAB implementation `faststln2` of the method of [15].

```
>> % Generate data
>> np = 12;                        % number of parameters
>> p0 = (1:np)';                   % true value of the parameter vector
>> p = p0 + [5; zeros(np-1,1)];% add disturbance
>> c = hankel(p(1:10),p(10:np)); a = c(:,1:2); b = c(:,3);
>> % Define the structure and solve the problem via STLS
>> s = [2 3 1];
>> tic, [xh_stls, i_stls]=stls(a,b,s); t_stls = toc
t_stls=
   0.00395000000000
>> disp(xh_stls(1:2)')
   0.30331872971326 0.87809000348994
>> disp(i_stls.fmin)  % value of the cost function at xh_stls
   2.88924164814028
>> % Solve via an alternative STLS method
>> ct = fliplr(c);  % ct is Toeplitz structured
>> tic, xh_stln = faststln2(c(:,1:2),c(:,3)); t_stln = toc
t_stln=
   0.19313500000000
>> % recover the solution of the Hankel structured problem
>> x_ext = [xh_stln;-1]; x_ext = flipud(x_ext);
         xh_stln = -x_ext(1:2)/x_ext(3);
>> disp(xh_stln(1:2)')
   0.30320645782842 0.87819047149399
>> disp(cost(xh_stln,a,b,s))  % value of the cost function at xh_stln
   2.88924181032173
```

The difference in the computed solutions `xh_stls` and `xh_stln` for the example above is due to the different convergence tolerances of the two methods. The cost function f_0 of the equivalent problem (6)

is evaluated at `xh_stls` and `xh_stln` and it turns out that the results coincide up to the 6th digit after the decimal point. Better approximation of the minimum point can be achieved (at the expense of extra computation time) by decreasing the convergence tolerances.

4.5. Deconvolution problem

The convolution of the sequences $(\ldots, a_{-1}, a_0, a_1, \ldots)$ with the sequence $(\ldots, x_{-1}, x_0, x_1, \ldots)$ is a sequence $(\ldots, b_{-1}, b_0, b_1, \ldots)$ defined as follows:

$$b_i = \sum_{j=-\infty}^{\infty} x_j a_{i-j}. \tag{10}$$

Assume that $x_j = 0$ for all $j < 1$ and for all $j > n$. Then (10) for $i = 1, \ldots, m$ can be written as the following structured system of equations:

$$\underbrace{\begin{bmatrix} a_0 & a_{-1} & \cdots & a_{1-n} \\ a_1 & a_0 & \cdots & a_{2-n} \\ \vdots & \vdots & & \vdots \\ a_{m-1} & a_{m+n-2} & \cdots & a_{m-n} \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}}_{b}. \tag{11}$$

Note that the matrix A is Toeplitz structured and is parameterized by the vector $a = \mathrm{col}(a_{1-n}, \ldots, a_{m-1}) \in \mathbb{R}^{m+n-1}$.

In the deconvolution problem we aim to find x, given a and b. With exact data the problem boils down to solving the system of equations (11). By construction it has a solution equal to x. Moreover the solution is unique whenever A is of full column rank, which can be translated to a condition on a (persistency of excitation).

The deconvolution problem is more realistic (and more challenging) when the data a, b is perturbed. We assume that $m > n$, so that the system of equations (11) is overdetermined. Because both a and b are perturbed and the A matrix is structured the deconvolution problem is similar to an STLS problem with the structure specification $\mathscr{D} = [[\mathtt{T}\ n], [\mathtt{U}\ 1]]$. A rigorous motivation for using the STLS method is the fact that under the additional assumption that the observations are obtained from true values with additive noise that is zero mean, normal, with covariance matrix a multiple of the identity, the STLS method provides a maximum likelihood estimate of the true values.

The STLS problem with the structure $\mathscr{D} = [[\mathtt{T}\ n], [\mathtt{U}\ 1]]$ is studied in [21], where an efficient $\mathrm{O}(m)$ method is proposed. We compare the solution obtained with the STLS package with the solution obtained with the MATLAB implementation `fastst ln1` of the method of [21]. In the particular simulation example shown below, the STLS package computes better approximation of a minimum point (i.e., the value of

the cost function f_0 at the computed solution is smaller), using about 200 times less computation time. This tremendous difference in the computation times can be attributed to the MATLAB implementation of `faststln1`. (M-files extensively using `for` loops are executed slowly in MATLAB versions 6.0 or smaller.)

```
>> m = 200; n = 2; % m = length (x), n = length (b)
>> % Generate true data: a0, b0, and x0
>> a0 = rand(n + m-1);A0 = toeplitz(a0(n:n + m − 1),a0(n:-1:1));
>> x0 = rand(n,1); b0 = A0*x0;
>>% Add noise: a = a0 + noise, b = b0 + noise
>> v_n = 0.25; % noise level
>> a = a0 + v_n * randn(n + m-1); b = b0 + v_n * randn(m,1);
>> A = toeplitz (a(n:n + m-1),a(n:-1:1));
>> % Define the structure and solve the deconvolution problem via STLS
>> s = [1 n 1; 3 1 1];
>> tic, [xh_stls,i_stls] = stls(A, b,s); t_stls = toc
t_stls=
   0.08655400000000
>> disp(xh_stls(1:2)')
   0.26594446871296 0.31420369470136
>> disp(i_stls.fmin) % value of the cost function at xh_stls
   15.23654290180259
>> % Solve via an alternative STLS method
>> tic, xh_stln = faststln1(A,b); t_stln = toc
t_stls=
   16.09171600000000
>> disp(xh_stln(1:2)')
   0.26594792311858 0.31420021871824
>> disp(cost1(xh_stln,a,b,s)) %value of the cost function at xh_stln
   15.23654290217436
```

4.6. Transfer function estimation

Consider the single input u, single output y linear time-invariant (LTI) system described by the difference equation

$$y_t + \sum_{\tau=1}^{n} a_\tau y_{t+\tau} = \sum_{\tau=0}^{n} b_\tau u_{t+\tau} \tag{12}$$

and define the parameter vector $x := \operatorname{col}(b_0, \ldots, b_n, -a_0, \ldots, -a_{n-1}) \in \mathbb{R}^{2n+1}$. The transfer function estimation problem is to find the parameter vector x, given a set of input/output measurements $(u_t, y_t)_{t=1}^{T}$ and the order n.

For the time horizon $t = 1, \ldots, T$, (12) can be written as the structured system of equations

$$\left[\begin{array}{cccc|cccc} u_1 & u_2 & \cdots & u_{n+1} & y_1 & y_2 & \cdots & y_n \\ u_2 & u_3 & \cdots & u_{n+2} & y_2 & y_3 & \cdots & y_{n+1} \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ u_m & u_{m+1} & \cdots & u_T & y_m & y_{m+1} & \cdots & y_{T-1} \end{array}\right] x = \begin{bmatrix} y_{n+1} \\ y_{n+2} \\ \vdots \\ y_T \end{bmatrix}, \tag{13}$$

where $m = T - n$. We assume that the time horizon is large enough to ensure $m \gg 2n + 1$. System (13) is satisfied for the exact input/output and a solution is the true value of the parameter x. Moreover, under additional assumption on the input (persistency of excitation) the solution is unique.

For perturbed input/output data an approximate solution is sought and the fact that the system of Eq. (13) is structured and all elements are perturbed suggests the use of the STLS method. Again under appropriate conditions for the data generating mechanism an STLS solution provides a maximum likelihood estimator.

The structure arising in this problem is $\mathscr{D} = [[\mathtt{H}\ n+1], [\mathtt{H}\ n+1]]$, where n is the order of the system. Unfortunately in this case we do not have an alternative method by which the result of the STLS package can be verified.

```
>> % True model
>> n = 3;
>> num = 0.151*[1 0.9 0.49 0.145]; den = [1 -1.2 0.81 -0.27]; % a, b
>> % True data
>> T = 1000; u0 = randn(T,1); [y0,x0] = dlsim(num, den, u0);
>> % Noisy data
>> v_n = .1; y = y0 + v_n * randn(T,1); u = u0 + v_n * randn(T,1);
>> % Define the system of equations
>> m = length(y) - n;
>> a = [hankel(u(1:m),u(m:end)) hankel(y(1:m),y(m:end)) ];
>> b = a(:,end); a(:,end) = [];
>> % Ignore the structure and solve the identification problem
   via LS and TLS
>> tic, xh_ls = a\b; t_ls = toc
t_ls =
   0.00329800000000
>> tic, xh_tls = tls(a,b); t_tls = toc
t_tls=
   0.00638900000001
>>%Define the structure and solve the identification problem via STLS
>> s = [2 n+1 1; 2 n+1 1];
>> tic, [xh_stls,i_stls] = stls(a,b,s); t_stls = toc
t_stls=
   1.27913800000000
>> % Extract the estimates
>> num_ls=fliplr(xh_ls(1:n+1)'); den_ls=[1 fliplr - (xh_ls(n+2:end)')];
```

```
>> num_tls = fliplr(xh_tls(1:n+1)');
           den_tls = [1 fliplr(- xh_tls(n+2:end)')];
>> num_stls = fliplr(xh_stls(1:n+1)');
            den_stls = [1 fliplr(- xh_stls (n+2:end)')];
>> % Compare the relative errors of estimation
>> e_ls = norm([num - num_ls, den - den_ls])/norm([num,den]); disp(e_ls)
   0.74534028390667
>> e_tls= norm([num - num_tls,den - den_tls]) /norm([num, den]); disp
 (e_tls)
   0.02825246589733
>> e_stls = norm([num-num_stls, den - den_stls]) / norm([num,den]);
          disp(e_stls)
   0.02381490382674
```

The relative error of estimation $e := \|\bar{x} - \hat{x}\|/\|\bar{x}\|$, where $\bar{x}$ is the vector of true values of the parameters and $\hat{x}$ is the vector of their estimates is largest for the LS method and is smallest for the STLS method. This relation of the estimation errors can be expected with high probability for large sample size ($T \to \infty$) due to the statistical consistency of the TLS and STLS methods and the inconsistency of the LS method. In addition, the STLS method being a maximum likelihood method is statistically more efficient than the TLS method.

5. Application in multivariable system identification

The application described is this section is a generalization of the transfer function estimation example of Section 4.6 and is treated in detail in [19]. Here, we briefly describe the problem and show some simulation results on the data sets from DAISY.

5.1. Description of the identification problem

Let $\mathscr{M}$ be a user-specified model class, consisting of LTI systems with bounded complexity and let w be an observed time series of length $T \in \mathbb{N}$. We view a model $\mathscr{B} \in \mathscr{M}$ as a collection of legitimate time series. Within $\mathscr{M}$, we aim to find the model $\hat{\mathscr{B}}$ that best fits the data according to the criterion

$$M(w, \mathscr{B}) := \min_{\hat{w} \in \mathscr{B}} \|w - \hat{w}\|_{\ell_2}^2. \tag{14}$$

The resulting optimization problem is known as the *global total least squares* problem [22].

We consider a difference equation representation of the system, i.e.,

$$\mathscr{B} = \{w\colon\ \mathbb{N} \to \mathbb{R}^{\mathsf{w}} | (2) \text{ holds}\}.$$

Note that no a priori separation of the variables into inputs and outputs is imposed. The number of inputs and the number of outputs in an input/output representation of $\mathscr{B}$, however, are invariant. We denote by $\mathscr{L}_{m,l}$ the set of all LTI systems with m inputs and lag at most l. The natural numbers m and l specify the maximum complexity of a model in the model class $\mathscr{L}_{m,l}$.

The considered identification problem is defined as follows. For a given time series w and a complexity specification (m, l), where m is the number of inputs and l is the lag of the identified system, solve the optimization problem

$$\hat{\mathscr{B}} := \arg \min_{\mathscr{B} \in \mathscr{L}_{m,l}} M(w, \mathscr{B}). \tag{15}$$

In [18] the identification problem (14) is expressed as an STLS problem (3). The parameter $\hat{X}$, in the STLS problem formulation, gives a difference equation representation of the system $\hat{\mathscr{B}}$. Moreover a transfer function and an input/state/output representations of $\hat{\mathscr{B}}$ can be derived from $\hat{X}$.

5.2. Performance on data sets from DAISY

Currently the data base for system identification DAISY [9] contains 28 real-life and simulated data sets, which are used for verification and comparison of identification algorithms. In this section, we apply the described identification method, implemented by the software package described here, on data sets from DAISY that correspond to input/output identification problems. (The other data sets consist of output only time series. They can be modeled as a response of an autonomous linear time invariant system and treated in a similar way by the STLS method but we do not do this here.)

The first part of Table 1 gives information for the data sets (number of data points T, number of inputs m, the number of outputs p) and shows the selected lag l for the identified model. Since all data sets are with given input/output partitioning, the only user-defined parameter selecting the complexity of the model class $\mathscr{L}_{m,l}$ is the lag l.

The estimates obtained by the following methods are compared:

- `subid`, a MATLAB implementation of the robust combined subspace algorithm of [28, Fig. 4.8];
- `detss`, a MATLAB implementation of the deterministic balanced subspace algorithm of [18];
- `pem`, the prediction error method of the Identification Toolbox of MATLAB;
- `stls`, the proposed method based on STLS.

Note that $l + 1$ is the user-supplied parameter i in the combined subspace algorithm `subid`. The order specified for the methods `subid`, `detss`, and `pem` is pl (the maximum possible in the model class $\mathscr{L}_{m,l}$).

The comparison is in terms of the relative percentage misfit

$$M_{\mathrm{rel}}(w, \hat{\mathscr{B}}) := 100\ M(w, \hat{\mathscr{B}})/\|w(t)\|_{\ell_2}.$$

M_{rel} is computed by solving the smoothing problem $M(w, \hat{\mathscr{B}})$ for the estimated models $\hat{\mathscr{B}}$. For `detss` and `pem`, $\hat{\mathscr{B}}$ is the deterministic part of the identified stochastic system.

The second part of Table 1 shows the relative misfits M_{rel} and execution time for the compared methods. The STLS solver is initialized with the approximation of the non-iterative method `subid` or `detss` that achieves smaller misfit on the particular data set. The time needed for the computation of the initial approximation is *not* added in the timing of `stls`. The prediction error method is called with the data w and the order $n = pl$ specification only, so that the appropriate model structure and computational method are selected automatically by the function.

Table 1
Relative misfits M_{rel} and execution times t in seconds for the examples and the methods

#	Data set name	Parameters				subid		detss		pem		stls	
		T	m	p	l	t	M_{rel}	t	M_{rel}	t	M_{rel}	t	M_{rel}
1	Destillation column	90	5	3	1	0.11	0.0089	0.57	0.0306	1.66	0.0505	0.45	0.0029
2	Destillation column n10	90	5	3	1	0.10	0.0089	0.57	0.0306	1.58	0.0505	0.45	0.0029
3	Destillation column n20	90	5	3	1	0.10	0.4309	0.57	0.1187	0.54	1.8574	1.17	0.0448
4	Destillation column n30	90	5	3	1	0.10	0.4357	0.61	0.1848	0.57	7.3600	1.18	0.0522
5	Glass furnace (Philips)	1247	3	6	1	0.15	33.6782	1.72	29.3373	43	31.5416	84	11.4120
6	120 MW power plant	200	5	3	2	0.14	8.9628	0.63	4.2906	2.68	35.4524	0.77	1.2427
7	pH process	2001	2	1	6	0.23	4.2564	0.64	4.4113	5.66	9.8727	1.93	3.2203
8	Hair dryer	1000	1	1	5	0.12	1.0437	0.18	1.0359	3.33	0.8311	1.19	0.8208
9	Winding process	2500	5	2	2	0.29	11.4838	1.15	10.1473	17	20.2908	62	7.1731
10	Ball-and-beam setup	1000	1	1	2	0.10	2.8962	0.17	28.3637	0.55	2.7708	0.11	2.6718
11	Industrial dryer	867	3	3	1	0.11	0.5586	0.59	0.5519	2.35	0.5553	5.11	0.4447
12	CD-player arm	2048	2	2	1	0.13	9.4629	0.59	8.7653	4.79	11.3623	9.37	7.7980
13	Wing flutter	1024	1	1	5	0.12	20.2766	0.19	21.0214	3.14	35.2727	0.91	11.6501
14	Robot arm	1024	1	1	4	0.12	3.8855	0.21	26.0082	2.66	36.1531	0.08	1.3905
15	Lake Erie	57	5	2	1	0.11	0.1423	0.14	0.2205	0.52	2.1548	0.43	0.0908
16	Lake Erie n10	57	5	2	1	0.10	0.0505	0.14	0.0538	0.85	0.1992	0.41	0.0221
17	Lake Erie n20	57	5	2	1	0.10	0.0607	0.16	0.0671	0.67	0.2677	0.45	0.0268
18	Lake Erie n30	57	5	2	1	0.10	0.0798	0.17	0.0564	0.52	0.1862	0.41	0.0329
19	Heat flow density	1680	2	1	2	0.13	0.7779	0.30	0.5651	3.73	4.1805	0.49	0.4219
20	Heating system	801	1	1	2	0.10	0.4913	0.17	0.4441	0.94	0.4973	0.09	0.3658
21	Steam heat exchanger	4000	1	1	2	0.14	0.1521	0.54	0.1499	3.37	0.6723	0.40	0.0822
22	Industrial evaporator	6305	3	3	1	0.32	37.7809	3.27	27.6341	40	40.6798	15	24.0065
23	Tank reactor	7500	1	2	1	0.19	0.1768	1.89	0.1621	24	3.9620	2.18	0.0749
24	Steam generator	9600	4	4	1	0.66	0.4175	8.45	0.5341	132	0.5751	118	0.1704

Since M_{rel} is up to a scaling factor equal to the cost function of `stls`, it is not surprising that the proposed method outperforms with respect to this criterion the alternative methods. The purpose of doing the comparison is to verify that the numerical tool needed for the solution of the optimization problem (3) is robust and efficient.

Indeed, identification problems with a few thousands of data points can be solved with the STLS software package. Such problems are infeasible for direct application of optimization methods without exploiting the special structure. Also the computation time of `stls` is similar to that of `pem`, which is also an optimization based method. On all examples, initialization of the STLS solver with the estimate obtained by a subspace identification method, leads to an improved solution, in terms of the misfit criterion. Hence at the expense of some extra computation time, the subspace approximation is improved by `stls`.

6. Conclusions and future work

We considered an STLS problem with structure of the data matrix, specified block-wise. Each of the blocks can be block-Toeplitz/Hankel structured, unstructured, or exact. It was shown that such a formulation is flexible and covers as special cases many previously studied structured and unstructured matrix approximation problems.

The numerical solution method is based on an equivalent unconstrained optimization problem (6). Under our assumptions, the weight matrix Γ is block-Toeplitz and block-banded. These properties were used for cost function and first derivative evaluation with computational cost linear in the sample size.

The block-Toeplitz/Hankel structure is motivated by identification and model reduction problems for multivariable LTI systems. Planned further extensions are to include a diagonal weight matrix $W > 0$, $\Delta p^{\top} W \Delta p$, and a regularization term $\mathrm{vec}^{\top}(X) Q \mathrm{vec}(X)$ in the STLS cost function.

Acknowledgements

The approach implemented in this package is described in the sequence of papers [13,16,17]. We had insightful discussions on the STLS problem and its computation with Alexander Kukush and Rik Pintelon.

Dr. Sabine Van Huffel is a full professor at the Katholieke Universiteit Leuven, Belgium. Research supported by Research Council KUL: GOA-Mefisto 666, IDO /99/003 and /02/009 (Predictive computer models for medical classification problems using patient data and expert knowledge), several PhD/postdoc & fellow grants; Flemish Government: o FWO: PhD/postdoc grants, projects, G.0078.01 (structured matrices), G.0407.02 (support vector machines), G.0269.02 (magnetic resonance spectroscopic imaging), G.0270.02 (nonlinear Lp approximation), research communities (ICCoS, ANMMM); o IWT: PhD Grants; Belgian Federal Science Policy Office IUAP P5/22 (‘Dynamical Systems and Control: Computation, Identification and Modelling’); EU: PDT-COIL, BIOPATTERN, ETUMOUR.

Appendix A. Implementation

The package uses MINPACK’s Levenberg–Marquardt algorithm [20] for the solution of the STLS problem (3), (4) in its equivalent formulation (6). There is no closed form expression for the Jacobian

matrix $J = [\partial r_i / \partial x_j]$, where $x = \mathrm{vec}(X)$, so that the pseudo-Jacobian J_+ proposed in [11] is used instead of J. Its evaluation is done with computational complexity $O(m)$.

The software is written in ANSI C language. For the vector–matrix manipulations and for a C version of MINPACK's Levenberg–Marquardt algorithm, we use the *GNU Scientific library* (*GSL*) [1]. The computationally most intensive step of the algorithm—the Cholesky decomposition of the block-Toeplitz, block-banded weight matrix $\Gamma(X)$—is performed via the subroutine `MB02GD` from the SLICOT library [26]. By default the optimization algorithm is initialized with the TLS solution. Its computation is performed via the SLICOT subroutine `MB02MD`.

The package contains:

- C-source code: `stls.c` and `stls.h` (the function `stls` implements Algorithm 3), see Section A.1.
- MATLAB interface to the C function `stls` via C-mex file `stls.m`, see Section A.2.
- A demo file `demo.m` with examples that illustrate the application of the STLS solver, see Section 4.
- Documentation (this paper) and the related papers [17,16,19] describing the STLS problem in more details.

It is available from `http://www.esat.kuleuven.ac.be/~imarkovs/stls/stls.html`

A.1. C function

The function `stls` implements Algorithm 3 for solving the STLS problem (3)–(4). Its prototype is

```
int stls(gsl_matrix* a, gsl_matrix* b, const data_struct* s,
         gsl_matrix* x, gsl_matrix* v, opt_and_info* opt)
```

Description of the arguments:

A.1. `a` and `b` are the matrices $A \in \mathbb{R}^{m\times n}$, and $B \in \mathbb{R}^{m\times d}$, respectively, such that $[A \;\; B] = \mathscr{S}(p)$. We refer to the GSL reference manual for the definition of the type `gsl_matrix` and the functions needed to allocate and initialize variables of this type.

A.2. `s` is the structure description $K, \mathscr{D}$ of $\mathscr{S}(p)$. The type `data_struct` is defined in `stls.h` as

```
/* structure of the data matrix C = [A B] */
#define MAXQ 10 /* maximum number of blocks in C */
typedef struct {
   int K;              /* = rowdim(block in T/H blocks)*/
   int q;              /* number of blocks in C = [C1...Cq]*/
   struct{
          char type;   /* 'T'-Toeplitz, 'H'-Hankel, 'U'-unstructured,
                          'E'-exact */
          int ncol;    /* number of columns */
          int nb;      /* = coldim(block in T/H blocks)*/
   }      a[MAXQ];     /* q-element array describing C1,...,Cq; */
}  data_struct;
```

A.3. `x` on input contains the initial approximation for the Levenberg–Marquardt algorithm and on exit, upon convergence of the algorithm, a local minimum point of the cost function f_0.

A.4. `v` on exit, contains the error covariance matrix $(J_+^\top J_+)^{-1}$ of the vectorized estimate $\hat{x} = \mathrm{vec}(\hat{X})$. It is useful for deriving confidence bounds.

A.5. `opt` on input contains options that control the exit condition of the Levenberg–Marquardt algorithm and on exit contains information about the convergence of the algorithm. The exit condition is

$$|x_j^{(k+1)} - x_j^{(k)}| < \texttt{epsabs} + \texttt{epsrel}\ |x_j^{(k+1)}|, \quad \text{for all } j = 1, \ldots, nd, \tag{16}$$

where $x^{(k)}$, $k = 1, 2, \ldots,$ `iter` $\leqslant$ `maxiter` are the successive iterates, and `epsrel`, `epsabs`, `maxiter` are fields of `opt`. Convergence to the desired tolerance is indicated by a positive value of `opt.iter`. In this case `opt.iter` is the number of iterations performed. `opt.iter` $= -1$ indicates lack of convergence. `opt.time` and `opt.fmin` show the time in seconds used by the algorithm and the cost function f_0 value at the computed solution.

The type `opt_and_info` is defined in `stls.h` as

```
/* optimization options and output information structure */
typedef struct {
            /* input options */
            int maxiter;
            double epsrel, epsabs;
            /* output information */
            int iter;
            double fmin;
            double time;
}           opt_and_info;
```

A.2. MATLAB mex-file

The provided C-mex file allows to call the C solver `stls` via the MATLAB command:

```
>>[xh, info, v] = stls(a,b,s,x,opt);
```

The input arguments `a`, `b`, and `s` are obligatory. `x` and `opt` are optional and can be skipped by the empty matrix `[]`. In these cases their default values are used.

Description of the arguments:

A.1. `a` and `b` are the matrices $A \in \mathbb{R}^{m\times n}$, and $B \in \mathbb{R}^{m\times d}$, respectively, where $[A\ \ B] = \mathscr{S}(p)$.

A.2. `s` is a $q \times 3$ matrix or a structure with scalar field `k` and a $q \times 3$ matrix field `a`. In the first case K is assumed to be 1, and in the second case it is specified by `s.k`. The array $\mathscr{D}$, introduced in Section 2, is specified by `s`, in the first case, and by `s.a`, in the second case. The first column of `s` (or `s.a`) defines the type of the blocks $C^{(1)}, \ldots, C^{(q)}$ (1 block-Toeplitz, 2 block-Hankel, 3 unstructured, 4 exact), the second column defines $n_1, \ldots, n_q$, and the third column defines $t_1, \ldots, t_q$.

A.3. `x` is a user-supplied initial approximation. Its default value is the TLS solution.

A.4. `opt` contains user supplied options for the exit conditions. `opt.maxiter` defines the maximum number of iterations (default 100), `opt.epsrel` defines the relative tolerance `epsrel` (default `1e-5`), and `opt.epsabs` defines the absolute tolerance ε_a `epsabs` (default `1e-5`), see (16).
A.5. `xh` is the computed solution.
A.6. `info` is a structure with fields `iter`, `time`, and `fmin` that gives information for the termination of the optimization algorithm. These fields are the ones returned from the C function, see item A.5.
A.7. `v` is the error covariance matrix $(J_+^\top J_+)^{-1}$ of the vectorized estimate $\hat{x} = \mathrm{vec}(\hat{X})$.

A.3. Compilation

The included make file, when called with argument `mex`, generates the MATLAB mex file. The GSL, BLAS, and LAPACK libraries have to be installed in advance. For their location and for the location of the `mex` command and options file, one has to edit the provided make file. Precompiled mex-files are included for Linux only.

Note 1. Due to particularities of the Windows operating system the compilation of the mex files for Windows is complicated. For UNIX-related operating systems the compilation is straightforward, once the supporting libraries are installed and the correct paths are specified in the make file.

References

[1] GSL - GNU Scientific Library, http://www.gnu.org/software/gsl/.
[2] T. Abatzoglou, J. Mendel, G. Harada, The constrained total least squares technique and its application to harmonic superresolution, IEEE Trans. Signal Process. 39 (1991) 1070–1087.
[3] M. Aoki, P.C. Yue, On a priori error estimates of some identification methods, IEEE Trans. Automatic Control 15 (5) (1970) 541–548.
[4] Y. Bresler, A. Macovski, Exact maximum likelihood parameter estimation of superimposed exponential signals in noise, IEEE Trans. Acoust. Speech Signal Process. 34 (1986) 1081–1089.
[5] J. Cadzow, Signal enhancement—a composite property mapping algorithm, IEEE Trans. Signal Process. 36 (1988) 49–62.
[6] G. Cirrincione, G. Ganesan, K. Hari, S. Van Huffel, Direct and neural techniques for the data least squares problem, in: International Symposium on the Mathematical Theory of Networks and Systems (MTNS), Perpignan, France, 2000.
[7] B. De Moor, Structured total least squares and L_2 approximation problems, Linear Algebra Appl. 188–189 (1993) 163–207.
[8] B. De Moor, Total least squares for affinely structured matrices and the noisy realization problem, IEEE Trans. Signal Process. 42 (11) (1994) 3104–3113.
[9] B. De Moor, Daisy: Database for the identification of systems, Department of Electrical Engineering, ESAT/SISTA, K.U. Leuven, Belgium, URL: http://www.esat.kuleuven.ac.be/sista/daisy/, 1998.
[10] G.H. Golub, C.F. Van Loan, An analysis of the total least squares problem, SIAM J. Numer. Anal. 17 (1980) 883–893.
[11] P. Guillaume, R. Pintelon, A Gauss–Newton-like optimization algorithm for "weighted" nonlinear least-squares problems, IEEE Trans. Signal Process. 44 (9) (1996) 2222–2228.
[12] W. Ku, R.H. Storer, Ch. Georgakis, Disturbance detection and isolation by dynamic principle component analysis, Chemometr. Intell. Lab. Systems 30 (1995) 179–196.
[13] A. Kukush, I. Markovsky, S. Van Huffel, Consistency of the structured total least squares estimator in a multivariate errors-in-variables model, J. Statist. Plann. Inference, to appear.
[14] P. Lemmerling, B. De Moor, S. Van Huffel, On the equivalence of constrained total least squares and structured total least squares, IEEE Trans. Signal Process. 44 (1996) 2908–2911.

[15] P. Lemmerling, N. Mastronardi, S. Van Huffel, Fast algorithm for solving the Hankel/Toeplitz structured total least squares problem, Numer. Algorithms 23 (2000) 371–392.
[16] I. Markovsky, S. Van Huffel, A. Kukush, On the computation of the structured total least squares estimator, Numer. Linear Algebra Appl. 11 (2004) 591–608.
[17] I. Markovsky, S. Van Huffel, R. Pintelon, Block-Toeplitz/Hankel structured total least squares, SIAM J. Matrix Anal. Appl., to appear.
[18] I. Markovsky, J.C. Willems, P. Rapisarda, B. De Moor, Algorithms for deterministic balanced subspace identification, Automatica, to appear.
[19] I. Markovsky, J.C. Willems, S. Van Huffel, B. De Moor, R. Pintelon, Application of structured total least squares for system identification and model reduction, Technical Report 04-51, Department of Electrical Engineering, K.U. Leuven, 2004.
[20] D. Marquardt, An algorithm for least squares estimation of nonlinear parameters, SIAM J. Appl. Math. 11 (1963) 431–441.
[21] N. Mastronardi, P. Lemmerling, S. Van Huffel, Fast structured total least squares algorithm for solving the basic deconvolution problem, SIAM J. Matrix Anal. 22 (2000) 533–553.
[22] B. Roorda, C. Heij, Global total least squares modeling of multivariate time series, IEEE Trans. Automat. Control 40 (1) (1995) 50–63.
[23] J.B. Rosen, H. Park, J. Glick, Total least norm formulation and solution of structured problems, SIAM J. Matrix Anal. 17 (1996) 110–128.
[24] M. Schuermans, P. Lemmerling, S. Van Huffel, Structured weighted low rank approximation, Numer. Linear. Algebra Appl. 11 (2004) 609–618.
[25] S. Van Huffel, H. Park, J.B. Rosen, Formulation and solution of structured total least norm problems for parameter estimation, IEEE Trans. Signal Process. 44 (10) (1996) 2464–2474.
[26] S. Van Huffel, V. Sima, A. Varga, S. Hammarling, F. Delebecque, High-performance numerical software for control, IEEE Control Systems Mag. 24 (2004) 60–76.
[27] S. Van Huffel, J. Vandewalle, The Total Least Squares Problem: Computational Aspects and Analysis, SIAM, Philadelphia, 1991.
[28] P. Van Overschee, B. De Moor, Subspace Identification for Linear Systems: Theory, Implementation, Applications, Kluwer Academic Publishers, Dordrecht, 1996.