A Guide to the STML MATLAB Package

Amir Beck and Yonina C. Eldar

November 1, 2009

1 Overview

This short note briefly describes the usage of the functions in the STML package which are based on the paper

> Amir Beck and Yonina C. Eldar, "Structured Total Maximum Likelihood: An Alternative to Structured Total Least-Squares", submitted for publication.

Our goal is to find a "solution" of an approximate linear system

$$\mathbf{A}\mathbf{x} \approx \mathbf{b},$$

where both $\mathbf{A}$ and $\mathbf{b}$ are noisy. We further assume that $\mathbf{A}$ possesses some linear structure. In particular, the assumed linear model is:

$$\left(\mathbf{A}_0 + \sum_{i=1}^{p} e_i \mathbf{A}_i\right)\mathbf{x} = \mathbf{b}_0 + \mathbf{w},$$

where

- $\mathbf{A}_0 \in \mathbb{R}^{m\times n}, \mathbf{b}_0 \in \mathbb{R}^m$. $\mathbf{A}_0$ is the nominal measurement matrix and $\mathbf{b}_0$ is the observed data vector (both are known).
- $\mathbf{A}_1, \ldots, \mathbf{A}_p$ are the structure matrices describing the corresponding linear structure.
- $e_1, \ldots, e_p$ are unknown perturbations of the structure components assumed to be distributed as $N(0, \sigma_e^2)$.
- $w_1, \ldots, w_m$ are the components of the vector $\mathbf{w}$ which is the perturbation to the right-hand side vector $\mathbf{b}$. It is assumed that $w_1, \ldots, w_m$ are distributed as $N(0, \sigma_w^2)$.

An underlying assumption is that the $m + p$ random variables $w_1, \ldots, w_m, e_1, \ldots, e_p$ are independently distributed. The STLS solution to the problem is an optimal solution to the following nonconvex optimization problem

$$(\text{STLS}): \quad \min_{\mathbf{x}} (\mathbf{A}\mathbf{x} - \mathbf{b})^T \mathbf{\Sigma}_{\mathbf{x}}^{-1} (\mathbf{A}\mathbf{x} - \mathbf{b}), \tag{1.1}$$

where

$$\mathbf{\Sigma_x} = \sigma_e^2 \sum_{i=1}^{p} \mathbf{A}_i \mathbf{x}\mathbf{x}^T \mathbf{A}_i^T + \sigma_w^2 \mathbf{I}. \tag{1.2}$$

The STML estimate is an optimal solution of the problem

$$\text{(STML)}: \quad \min_{\mathbf{x}} \{(\mathbf{Ax}-\mathbf{b})^T \mathbf{\Sigma_x}^{-1} (\mathbf{Ax}-\mathbf{b}) + \log\det \mathbf{\Sigma_x}\}. \tag{1.3}$$

2 General Structures

In this section the following functions are described:

```
mstls.m
stls.m
detect_structure.m
detect_structure_unweighted.m
```

Suppose that $\mathbf{A}_0$ is a 3×2 matrix of the form

$$\begin{pmatrix} a_1 & a_1 \\ a_2 & a_1 \\ 0 & a_2 \end{pmatrix}$$

For example, consider the following observed matrix and righthand side vector:

```
>>A_0=[0.234,0.234;0.14,0.234;0,0.14]
A_0 =
    0.2340    0.2340
    0.1400    0.2340
         0    0.1400
>>b_0=[1;2;3]
b_0 =
     1
     2
     3
```

If the true structure components are assumed to be given by

$$g_1 = a_1 + \varepsilon_1, g_2 = a_2 + \varepsilon_2, \quad (a_1 = 0.234, a_2 = 0.14)$$

where $\varepsilon_i \sim N(0, \sigma_e^2)$, then the structure matrices can be defined by

```
>>A(:,:,1)=[1,1;0,1;0,0];
>>A(:,:,2)=[0,0;1,0;0,1]

A(:,:,1) =
```

```
     1     1
     0     1
     0     0
A(:,:,2) =
     0     0
     1     0
     0     1
```

Assuming that $\sigma_e = \sigma_w = 0.1$, the STLS solution of the problem can be found by the function stls.m as follows:

```
x_stls=stls(2,A,A_0,b_0,0.1,0.1)
```

and the output is

```
 Optimization terminated: relative infinity-norm of gradient less than
options.TolFun.
 x_stls =
  -14.8060
   19.2849
```

To find the STML solution, the following command should be invoked:

```
>>x_stml=stml(2,A,A_0,b_0,0.1,0.1)

Optimization terminated: relative infinity-norm of gradient less
than options.TolFun.

x_stml =
  -12.0923
   15.7693
```

The function `stml.m` invokes the MATLAB function `fmincon` and uses the BFGS method. A message describing the stopping criteria used to terminate the algorithm used by `fmincon` is given.

Now consider the following symmetric Toeplitz matrix

```
>>A_0=toeplitz([100,3,1,0,0])

A_0 =
   100     3     1     0     0
     3   100     3     1     0
     1     3   100     3     1
     0     1     3   100     3
     0     0     1     3   100
```

This matrix has three structure components. In some scenarios it is more logical to assume that the perturbations of the structure components are proportional to their nominal value. That is, instead of assuming that the "true" structured matrix is

$$\begin{pmatrix} 100+\varepsilon_1 & 3+\varepsilon_2 & 1+\varepsilon_3 & 0 & 0 \\ 3+\varepsilon_2 & 100+\varepsilon_1 & 3+\varepsilon_2 & 1+\varepsilon_3 & 0 \\ 1+\varepsilon_3 & 3+\varepsilon_2 & 100+\varepsilon_1 & 3+\varepsilon_2 & 1+\varepsilon_3 \\ 0 & 1+\varepsilon_3 & 3+\varepsilon_2 & 100+\varepsilon_1 & 3+\varepsilon_2 \\ 0 & 0 & 1+\varepsilon_3 & 3+\varepsilon_2 & 100+\varepsilon_1 \end{pmatrix},$$

where $\varepsilon_i \sim N(0, \sigma_e^2), i = 1, 2, 3$, it is more reasonable to assume that the"true" matrix is

$$\begin{pmatrix} 100(1+\varepsilon_1) & 3(1+\varepsilon_2) & 1(1+\varepsilon_3) & 0 & 0 \\ 3(1+\varepsilon_2) & 100(1+\varepsilon_1) & 3(1+\varepsilon_2) & 1(1+\varepsilon_3) & 0 \\ 1(1+\varepsilon_3) & 3(1+\varepsilon_2) & 100(1+\varepsilon_1) & 3(1+\varepsilon_2) & 1(1+\varepsilon_3) \\ 0 & 1(1+\varepsilon_3) & 3(1+\varepsilon_2) & 100(1+\varepsilon_1) & 3(1+\varepsilon_2) \\ 0 & 0 & 1(1+\varepsilon_3) & 3(1+\varepsilon_2) & 100(1+\varepsilon_1) \end{pmatrix},$$

The only difference is that the structure matrices should include the nominal value and not just zeros and ones:

```
>>clear A;
>>A(:,:,1)=toeplitz([100,0,0,0,0]);
>>A(:,:,2)=toeplitz([0,3,0,0,0]);
>>A(:,:,3)=toeplitz([0,0,1,0,0])

A(:,:,1) =
   100     0     0     0     0
     0   100     0     0     0
     0     0   100     0     0
     0     0     0   100     0
     0     0     0     0   100
A(:,:,2) =
     0     3     0     0     0
     3     0     3     0     0
     0     3     0     3     0
     0     0     3     0     3
     0     0     0     3     0
A(:,:,3) =
     0     0     1     0     0
     0     0     0     1     0
     1     0     0     0     1
     0     1     0     0     0
     0     0     1     0     0
```

If the righthand side vector $\mathbf{b}_0$ is given by

```
>>b_0=[1;2;3;4;5]
b_0 =
     1
     2
     3
     4
     5
```

Then with $\sigma_e = \sigma_w = 0.1$, the MSTLS solution can be computed by[1]

```
>> x_stml=stml(3,S,A_0,b_0,0.1,0.1)
Optimization cannot make further progress:
 relative change in x less than options.TolX.

 x_stml =
    0.0091
    0.0184
    0.0275
    0.0372
    0.0481
```

It is also possible to extract the structure matrices using the function `detect_structure.m`. Just write

```
>> [p,S]=detect_structure(A_0)

p =

     3


S(:,:,1) =

     0     0     1     0     0
     0     0     0     1     0
     1     0     0     0     1
     0     1     0     0     0
     0     0     1     0     0
```

[1]In our demonstration the standard deviations σ_w, σ_w are the same, but it is perfectly legitimate to use different standard deviations.

```
S(:,:,2) =

     0     3     0     0     0
     3     0     3     0     0
     0     3     0     3     0
     0     0     3     0     3
     0     0     0     3     0


S(:,:,3) =

   100     0     0     0     0
     0   100     0     0     0
     0     0   100     0     0
     0     0     0   100     0
     0     0     0     0   100
```

This function will work only when values of different structure components are not the same. If *un-weighted* structure matrices are required, then the function

`detect_structure_unweighted.m`

should be invoked:

```
>>[p,S]=detect_structure_unweighted(A_0)
p =
     3
S(:,:,1) =
     0     0     1     0     0
     0     0     0     1     0
     1     0     0     0     1
     0     1     0     0     0
     0     0     1     0     0
S(:,:,2) =
     0     1     0     0     0
     1     0     1     0     0
     0     1     0     1     0
     0     0     1     0     1
     0     0     0     1     0
S(:,:,3) =
     1     0     0     0     0
     0     1     0     0     0
     0     0     1     0     0
     0     0     0     1     0
     0     0     0     0     1
```

3 The Matrix-Restricted structure

The function

`mstls_dec.m`

can be used when the perturbation matrix has the matrix-structure DEC where

- $\mathbf{E} \in \mathbb{R}^{p\times q}$ is an unknown matrix whose components are independently and normally distributed as $N(0, \sigma_e^2)$.
- $\mathbf{D} \in \mathbb{R}^{m\times p}, \mathbf{C} \in \mathbb{R}^{n\times q}$ are known.

The linear model here is therefore

$$(\mathbf{A}_0 + DEC)\mathbf{x} = \mathbf{b}_0 + \mathbf{w}$$

For example if $\mathbf{A}_0, \mathbf{b}_0, \mathbf{D}$ and $\mathbf{C}$ are given by

```
>>A_0=[1,2;3,4]
A_0 =
     1     2
     3     4

>>b_0=[5;6]
b_0 =
     5
     6

>>D=[1,-1;2,-3]
D =
     1    -1
     2    -3

>>C=[1,0;1,2]
C =
     1     0
     1     2
```

then the STML solution with $\sigma_e = \sigma_w = 0.1$ can be found by

```
>>x_stml1=stml_dec(A_0,b_0,D,C,0.1,0.1)

 Func-count     x          f(x)         Procedure
    1        38.1966     -1.41606        initial
    2        61.8034      3.17932        golden
```

```
    3        23.6068      11.1174        golden
    4        46.4718     -0.77517        parabolic
    5        37.2693     -1.30881        parabolic
    6        40.4887     -1.48735        parabolic
    7         42.774     -1.33908        golden
    8        40.0843     -1.49262        parabolic
    9        40.0081     -1.49282        parabolic
   10        39.9858     -1.49283        parabolic
   11         39.987     -1.49283        parabolic
   12        39.9871     -1.49283        parabolic
   13        39.9871     -1.49283        parabolic
   14        39.9871     -1.49283        parabolic

Optimization terminated:
 the current x satisfies the termination criteria using OPTIONS.TolX  of 1.000e-006
x_stml1 =
   -4.0025
    4.4491
```

Thus, the STML solution is $\begin{pmatrix} -4.0025 \\ 4.4491 \end{pmatrix}$ and the optimal value is -1.49283. As a sanity check, let us compute the same solution using the function stml.m that can handle general structures. Denoting

$$\mathbf{E} = \begin{pmatrix} \varepsilon_1 & \varepsilon_2 \\ \varepsilon_3 & \varepsilon_4 \end{pmatrix},$$

$$\begin{aligned} \mathbf{DEC} &= \begin{pmatrix} 1 & -1 \\ 2 & -3 \end{pmatrix} \begin{pmatrix} \varepsilon_1 & \varepsilon_2 \\ \varepsilon_3 & \varepsilon_4 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 1 & 2 \end{pmatrix} \\ &= \varepsilon_1 \begin{pmatrix} 1 & 0 \\ 2 & 0 \end{pmatrix} + \varepsilon_2 \begin{pmatrix} 1 & 2 \\ 2 & 4 \end{pmatrix} + \varepsilon_3 \begin{pmatrix} -1 & 0 \\ -3 & 0 \end{pmatrix} + \varepsilon_4 \begin{pmatrix} -1 & -2 \\ -3 & -6 \end{pmatrix}. \end{aligned}$$

Therefore, the structure matrices can be defined:

```
 >>clear A;
>>A(:,:,1)=[1,0;2,0];
>>A(:,:,2)=[1,2;2,4];
>>A(:,:,3)=[-1,0;-3,0];
>>A(:,:,4)=[-1,-2;-3,-6]
A(:,:,1) =
     1     0
     2     0
A(:,:,2) =
     1     2
     2     4
```

```
A(:,:,3) =
    -1     0
    -3     0
A(:,:,4) =
    -1    -2
    -3    -6
```

and the STML estimate can now be found using `stml.m`:

```
>>[x_stml2,f]=stml(4,A,A_0,b_0,0.1,0.1)
Optimization terminated: relative infinity-norm of gradient less
than options.TolFun.

x_stml2 =
   -4.0025
    4.4491
f =
   -1.4928
```

Obviously this is the same solution as the one produced by `stml_dec.m`. It is important to note that for matrix-restricted structures, the function `stml_dec.m` is more recommended than the function `stml.m`. The latter function first reduces the problem into a one-dimensional problem and is more efficient and less likely to get stuck at a local minimum. In addition, when both $\mathbf{D}$ and $\mathbf{C}$ are the identity matrices (of appropriate sizes), we are dealing with the unstructured case and the function `stml_dec.m` is guaranteed to find the exact STML solution.

4 The Circulant Structure

The function

`stml_circulant.m`

can be used in order to find the STML estimate for the circulant structure. The function is guaranteed to produce a *global* optimal solution of the corresponding nonconvex problem. Let us begin with a small example. Suppose that we consider the 4×4 circulant matrix with first row

```
>>a_0=[1,2,3,4]
a_0 =
     1     2     3     4
```

We can compute the circulant matrix by using the following command:

```
>> A_0=gallery('circul',a_0)

A_0 =

     1     2     3     4
     4     1     2     3
     3     4     1     2
     2     3     4     1
```

The righthand side vector is defined by

```
>> b_0=[5;6;7;8]

b_0 =

     5
     6
     7
     8
```

Suppose that $\sigma_e = \sigma_w = 0.1$. Then the STML estimate can be computed via the command `stml_circulant.m`

```
>>x_stml=stml_circulant(a_0,b_0,0.1,0.1)

 #ITER      #1D-PROBLEMS      #MAX-INTERVAL
  0               4             1.127e+000
  1               4             7.511e-001
  2               4             5.007e-001
  3               4             3.338e-001
  4               4             2.226e-001
  5               4             1.484e-001
  6               4             9.891e-002
  7               4             6.594e-002
  8               4             4.396e-002
  .               .                 .
  .               .                 .
  .               .                 .
 42               4             4.529e-008
 43               1             3.019e-008
 44               1             2.013e-008
 45               1             1.342e-008
 46               1             8.945e-009
```

```
x_mstls =
    0.8973
    0.8997
    0.8973
   -0.0953
```

We can also compute this solution using the general purpose function `stml.m`. For that, we first need to find the corresponding structure matrices:

```
>>[p,S]=detect_structure_unweighted(A_0)
p =
     4
S(:,:,1) =
     1     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1
S(:,:,2) =
     0     1     0     0
     0     0     1     0
     0     0     0     1
     1     0     0     0
S(:,:,3) =
     0     0     1     0
     0     0     0     1
     1     0     0     0
     0     1     0     0
S(:,:,4) =
     0     0     0     1
     1     0     0     0
     0     1     0     0
     0     0     1     0
```

and then invoke the function:

```
>>stml(p,S,A_0,b_0,0.1,0.1)
Optimization cannot make further progress:
 relative change in x less than options.TolX.
ans =
    0.8973
    0.8997
    0.8973
   -0.0953
```

Let us now take a larger example. Begin by fixing the seeds of randn and rand

```
>>randn('seed',314);
>>rand('seed',314);
```

Now, let us generate the "true" circulant matrix by randomly generating the first row:

```
>>a_t=rand(100,1);
```

Let us assume that the "true" signal is the vector of 100 ones:

```
>>x_t=ones(100,1);
```

The "true" righthand side is computed as follows:

```
>> A_t=gallery('circul',a_t);
>> b_t=A_t*x_t;
```

The observed righthand side and first row of the matrix are randomly generated as

```
>> a_0=a_t+0.1*randn(100,1);
>> b_0=b_t+0.1*randn(100,1);
>> A_0=gallery('circul',a_0);
```

The solution to the system $\mathbf{A}_0\mathbf{x} = \mathbf{b}_0$ is given by

```
>> x_ls=A_0\b_0;
```

The STML solution is

```
>> x_stml=stml_circulant(a_0,b_0,0.1,0.1);
```

This is a much better solution to the problem. To see this, we can just write

```
>> mean(abs(x_ls-x_t))

ans =

    0.3675

>> mean(abs(x_stml-x_t))

ans =

    0.0385
```

That is, the average error per component of the naive solution is one order of magnitude larger than the error of the STML solution. Although not recommended, we can solve the same problem with the general purpose algorithm `stml.m`:

```
>> [p,S]=detect_structure_unweighted(A_0);
>> x_stml2=stml(p,S,A_0,b_0,0.1,0.1);
```

In this case we get the same solution (up to the tolerance):

```
>> norm(x_stml2-x_stml)

ans =

  2.8989e-006
```

If we start with a random vector:

```
>> x_stml3=stml(p,S,A_0,b_0,0.1,0.1,10*randn(100,1));
```

then we also get the same solution, but this time the algorithm is much slower (since the initial guess was far). It is not difficult to find scenarios in which the algorithm of `stml.m` does not find the optimal solution since this is a general purpose algorithm for nonconvex optimization. On the other hand, the function `stml_circulant` is guaranteed to produce the global optimal solution.

5 The BCCB Structure

The function

`stml_BCCB.m`

can be used in order to find the STML estimate for the BCCB structure. Like in the circulant case, this function is guaranteed to produce a *global* optimal solution of the corresponding nonconvex problem. The input to this function fits the image deblurring application associated with the BCCB structure. We will reconstruct here the example from the paper. Let us begin with uploading the Lena test image by writing

```
>> X=double(imread('lena_gray.tif'));
>> X=X/255;
```

The second command is done for scaling reasons. The image is blurred with a Gaussian PSF generated by

```
>> [P,center]=psfGauss([31,31],2);
```

where the command psfGauss is taken from the HNO package of [21] (an alternative is to use the function fspecial from the MATLAB image processing toolbox).

To actually generate the blurred image we write

```
>> B=imfilter(X,P,'circular');
```

Note that the blurring was under the periodic boundary conditions assumption. We now set the seed of the randn function:

```
>> randn('seed',314);
```

The observed image is constructed by adding to each component of B a normally distributed zero-mean random variable with standard deviation 10^{-3}.

```
>> Bobs=B+1e-3*randn(size(B));
```

The observed PSF is constructed by adding to each component of P a normally distributed zero-mean random variable with standard deviation 10^{-4}.

```
>> Pobs=P+sigma_e*randn(size(P));
```

The STML solution is now found by writing:

```
>> Xstml=stml_BCCB(Pobs,center,Bobs,1e-3,1e-4);
```