Total Least Squares (TLS): multivariate example with 5 random variables

This code successfully executed on Matlab **R2022a**

Objective: illustrating a Total Least Squares case study involving five correlated variables.

Presentations in video:

http://personales.upv.es/asala/YT/V/tls51EN.html , http://personales.upv.es/asala/YT/V/tls52EN.html .

Table of Contents

Data generation (we are God)

Let us consider a "hidden" model where two variables (y_1, y_2) are a linear function of other three ones (x_1, x_2, x_3). Well, TLS will seek finding correlations in the vector $(y_1, y_2, x_1, x_2, x_3)$.

The model will be in the form

$$(y_1 \;\; y_2)_{N\times 2} = (x_1 \;\; x_2 \;\; x_3)_{N\times 3} \cdot \Theta_{3\times 2}$$

so each sample of data has the variables in a "row", being N the number of samples.

We may express an equivalent model in "column" form, of course, just transposing.

```
Xclean=randn(90000,3)*diag([25 1.7 18]);%random X data
ThetaClean=[2 40 5;-2 10 3]'
```

```
ThetaClean = 3×2
     2    -2
    40    10
     5     3
```

```
Yclean=Xclean*ThetaClean;
%TLS assumes clean linearly-related data are corrupted by independent
%measurement noise in all channels, let's do it...
stdx1=6; stdx2=0.4; stdx3=5; stdy1=0.5; stdy2=7.5;
stdX=diag([stdx1 stdx2 stdx3]); stdY=diag([stdy1 stdy2]);
X=Xclean+randn(size(Xclean))*stdX; %measurement noise in X
```

```
Y=Yclean+randn(size(Yclean))*stdY;%measurement noise in Y
%of course, only X and Y are actually "accessible" to the data engineer in the code bel
```

Ordinary LS (biased)

$Y = X\theta$, we may obtain θ with:

```
Th_LS_biased=pinv(X)*Y
```

```
Th_LS_biased = 3×2
    1.8891   -1.8880
   37.8767    9.4249
    4.6395    2.7845
```

```
ThetaClean
```

```
ThetaClean = 3×2
     2    -2
    40    10
     5     3
```

Biased because we do not recover the correct theta even with "a lot" of data.

TLS estimate

*We'll assume that standard deviations of the random additive noise are known... that assumption may actually be difficult in practice!.

Data in rows

```
size(X)
```

```
ans = 1×2
       90000           3
```

```
size(Y)
```

```
ans = 1×2
       90000           2
```

```
stdX=diag([stdx1 stdx2 stdx3]); %we repeat because we cannot access (supposedly) the da
stdY=diag([stdy1 stdy2]); %we repeat because we cannot access (supposedly) the data ger
Xesc=(X-mean(X))*inv(stdX); %we scale to zero mean and unit variance measurement noise
Yesc=(Y-mean(Y))*inv(stdY); %we scale to zero mean and unit variance measurement noise
Data_esc=[Yesc Xesc];  %Now all five columns have "unit variance measurement noise"
[N,m]=size(Data_esc)
```

```
N = 90000
m = 5
```

*Note: in the video, I first divided by standard deviation and I subtracted the mean later on... It's OK but, well, what everybody does to center data is first subtracting the mean and then dividing by standard deviation, so I changed it in the materials.

```
tic
[U,S,V]=svd(Data_esc/sqrt(N-1),'econ'); %TLS and SVD are the same with the proposed sca
%dividing by sqrt(N-1), S has units of standard deviation
toc %svd is fast.
```

```
Elapsed time is 0.008457 seconds.
```

The standard deviation of each principal component is:

```
diag(S)' %I should find the model as "unit noise" singular values, given the above scal
```

```
ans = 1×5
  247.0704   10.1631    4.2263    0.9999    0.9987
```

```
size(U)
```

```
ans = 1×2
       90000           5
```

```
size(V) %the 5x5 matrix is the one to use in TLS
```

```
ans = 1×2
     5     5
```

```
Model_scaled=V(:,4:5) %model relating scaled variables
```

```
Model_scaled = 5×2
   -0.0163    0.0062
    0.2895    0.2725
    0.8526    0.2881
    0.3664   -0.3414
    0.2340   -0.8522
```

So we have "[Yesc Xesc]*Model_scaled ≈ 0". TLS does not know whether any of these variables are "inputs" or "outputs". As we generated data with $Y = X\theta$, let us solve for Y to check what we get. Once we undo the scaling, we will have

$YM_1 + XM_2 = 0$, thus $YM_1 = -XM_2$, and finally $Y = -X \cdot M_2 M_1^{-1}$.

```
ModX_sc=Model_scaled(3:5,:)
```

```
ModX_sc = 3×2
    0.8526    0.2881
    0.3664   -0.3414
    0.2340   -0.8522
```

```
ModY_sc=Model_scaled(1:2,:)
```

```
ModY_sc = 2×2
```

```
   -0.0163    0.0062
    0.2895    0.2725
```

```
ModX=stdX\ModX_sc %matrix multiplying X in original units in the found model (M2 above)
```

```
ModX = 3×2
    0.1421    0.0480
    0.9160   -0.8534
    0.0468   -0.1704
```

```
ModY=stdY\ModY_sc  %matrix multiplying Y in original units in the found model (M1 above
```

```
ModY = 2×2
   -0.0325    0.0123
    0.0386    0.0363
```

```
ThetaEstimTLS=-ModX*inv(ModY)  %estimated TLS parameter, solving for Y explicitly
```

```
ThetaEstimTLS = 3×2
    1.9968   -1.9982
   39.9640    9.9556
    4.9960    2.9992
```

It's unbiased when comparing to our ideal "clean" parameter we used when generating the data:

```
ThetaClean
```

```
ThetaClean = 3×2
     2    -2
    40    10
     5     3
```

Data arranged in "columns"

```
X=X';Y=Y';
size(X)
```

```
ans = 1×2
           3       90000
```

```
size(Y)
```

```
ans = 1×2
           2       90000
```

We would now have

```
stdX=diag([stdx1 stdx2 stdx3]);
stdY=diag([stdy1 stdy2]);
Xesc=inv(stdX)*X;
Yesc=inv(stdY)*Y;
Data_esc=[Yesc; Xesc];
```

```
[m,N]=size(Data_esc)
```

```
m = 5
N = 90000
```

```
Data_esc=Data_esc-sum(Data_esc,2)/N; %subtract the mean, now summing along dimension 2
[U,S,V]=svd(Data_esc/sqrt(N-1),'econ');
diag(S)'
```

```
ans = 1×5
  247.0704   10.1631    4.2263    0.9999    0.9987
```

```
size(U) %choose 5x5 for TLS
```

```
ans = 1×2
     5     5
```

```
Model_scaled=U(:,4:5)' %new model, now it's the transpose of the earlier one, but it's
```

```
Model_scaled = 2×5
   -0.0163    0.2895    0.8526    0.3664    0.2340
    0.0062    0.2725    0.2881   -0.3414   -0.8522
```

```
ModX_sc=Model_scaled(:,3:5);
ModY_sc=Model_scaled(:,1:2);
ModX=ModX_sc/stdX
```

```
ModX = 2×3
    0.1421    0.9160    0.0468
    0.0480   -0.8534   -0.1704
```

```
ModY=ModY_sc/stdY
```

```
ModY = 2×2
   -0.0325    0.0386
    0.0123    0.0363
```

```
ThetaEstimTLS=-inv(ModY)*ModX %this is how we now must solve Y=\theta X
```

```
ThetaEstimTLS = 2×3
    1.9968   39.9640    4.9960
   -1.9982    9.9556    2.9992
```

```
ThetaClean'
```

```
ans = 2×3
     2    40     5
    -2    10     3
```

So it's irrelevant whether I arrange data in rows or columns, results are equivalent, of course.

Addenda: what would happen if my scaling is wrong?... well, TLS won't work well!

```
DataWrong=[Y; X]; %NO scaling, because I (wrongly) thought measurement noise was alread
DataWrong=DataWrong-sum(DataWrong,2)/N;
[U,S,V]=svd(DataWrong/sqrt(N-1),'econ');
```

```
diag(S)' %only one equation relating things? (significantly smaller singular value)?
```

```
ans = 1×5
  129.0442   71.6624   12.0066    5.8477    0.7345
```

```
Model_wrong_scaling=U(:,4:5)' %no need of undoing the scaling step, because there was n
```

```
Model_wrong_scaling = 2×5
   -0.0709    0.3061    0.7511    0.0200   -0.5803
    0.0221   -0.0021   -0.0457   -0.9940   -0.0972
```

```
Theta_wrong_scaling=-inv(Model_wrong_scaling(:,1:2))*Model_wrong_scaling(:,3:5) %biased
```

```
Theta_wrong_scaling = 2×3
    1.8763   45.8866    4.6662
   -2.0191   10.5645    2.9768
```

```
ThetaClean'
```

```
ans = 2×3
     2    40     5
    -2    10     3
```

```
idealModel=[eye(2) -ThetaClean'];
Model_good_scaling=[ModY ModX] %the "good TLS one", in original un-scaled coordinates
```

```
Model_good_scaling = 2×5
   -0.0325    0.0386    0.1421    0.9160    0.0468
    0.0123    0.0363    0.0480   -0.8534   -0.1704
```

Saying that $Model * Data \approx 0$ amounts to saying that the allowed data according to the model are the subspace given by `null(Model)`. Let us compute the "angle" between these subspaces to summarise the modelling error in a single number:

```
subspace(null(Model_wrong_scaling),null(idealModel))*180/pi %angle in degrees
```

```
ans = 3.0736
```

```
subspace(null(Model_good_scaling),null(idealModel))*180/pi %angle in degrees
```

```
ans = 0.1118
```

```
subspace(null([-eye(2) Th_LS_biased']),null(idealModel))*180/pi
```

```
ans = 1.1526
```

So, the "angle" bias due to wrong TLS scaling is even **worse** than the bias of the standard LS solution. Wow! Nevertheless, the bias less than 3 degrees might not be so much in applications (I would have, say, more severe "variance-related" problems if I had 50 data instead of 90000). Note however, that "subspace angle" depends on the "chosen" scaling, so changing a variable from volts to milivolts would change the angle output above.

Conclusions

If data are "very abundant", and generated in accordance with theory and I know the "measuring noise variance", then everything works as theory predicts.

If I have not so many data, or I make a wrong guess with the scaling, then TLS may be biased. Real life is hard.