1.13 Total Least Squares (TLS)

1.13.1 Theory

Total Least Squares solves a nonlinear, overconstrained system of equations by minimizing the squared residuals of each observation. A Taylor Series expansion is utilized to linearize the equation and iteratively calculate the gradient of the function to determine a local minima. A Weight Matrix, with a column/row for each observation rather than observation equation, is used to allow for error in all dimensions. Traditionally, this will be a predicted variance for each observation.

*Note that if the scale of the variance-covariance in the weight matrix is known to be 1, then the computed reference variance should be inspected to ensure it passes the χ^2 goodness of fit test. If it passes the test, the Covariance matrix should NOT be multiplied by the reference variance. See definition of reference variance for the reasoning.

1.13.2 Assumptions

- No Outliers/Blunders. Nonlinear Least Squares is not robust to outliers (consider RANSAC/Robust Weighting if outliers)
- System is over-constrained (eg. Number of Observation Equations > Number of Unknowns)
- Error is in all measurements variable (eg. mx+b = y + v → error only in x and y dimension)
- X_0 must be a reasonable guess, otherwise the solution might settle on an incorrect local minima, rather than the global minimum. If a linear problem, one method is to solve the unweighted OLS, then use that to initialize X_0 in TLS.

1.13.3 Equations

$$AX = L \qquad \text{(note: residuals in both X and L)}$$

$$BV + J\Delta X = K$$

$$m = \text{number of observations} \qquad n = \text{number of unknowns}$$

$$p = \text{number of observation equations for each observation}$$

$$q = \text{number of input variables}$$

$$dof = \text{degrees of freedom (\# of redundant observations)} = m - n$$

$$i = \text{loop iteration}$$

$$\text{Observation Equations} = F_m(x_1, x_2, ..., x_n) = l_m \text{ (note: } AX \text{ is obs eqn, L is } [l_1, l_2..., l_m] \text{)}$$

$$b(r) = \begin{bmatrix} \frac{\delta F_1}{\delta v_{1r}} & \frac{\delta F_1}{\delta v_{2r}} & \dots & \frac{\delta F_1}{\delta v_{pr}} \\ \frac{\delta F_2}{\delta v_{1r}} & \frac{\delta F_2}{\delta v_{2r}} & \dots & \frac{\delta F_2}{\delta v_{pr}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\delta F_k}{\delta v_{1r}} & \frac{\delta F_k}{\delta v_{2r}} & \dots & \frac{\delta F_k}{\delta v_{pr}} \end{bmatrix} \qquad B = \begin{bmatrix} b(1) & 0 & \dots & 0 \\ 0 & b(2) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & b(m) \end{bmatrix} \qquad V = \begin{bmatrix} v_{11} \\ v_{21} \\ \vdots \\ v_{p1} \\ v_{12} \\ v_{22} \\ \vdots \\ v_{p2} \\ \vdots \\ v_{1m} \\ v_{2m} \\ \vdots \\ v_{pm} \end{bmatrix}$$

$$J = \begin{bmatrix} \frac{\delta F_1}{\delta x_1} & \frac{\delta F_1}{\delta x_2} & \cdots & \frac{\delta F_1}{\delta x_n} \\ \frac{\delta F_2}{\delta x_1} & \frac{\delta F_2}{\delta x_2} & \cdots & \frac{\delta F_2}{\delta x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\delta F_m}{\delta x_1} & \frac{\delta F_m}{\delta x_2} & \cdots & \frac{\delta F_m}{\delta x_n} \end{bmatrix} \qquad \Delta X = \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \\ \vdots \\ \Delta x_n \end{bmatrix} \qquad K = \begin{bmatrix} l_1 - F_1(X_i) \\ l_2 - F_2(X_i) \\ \vdots \\ l_m - F_m(X_i) \end{bmatrix}$$

$$\Sigma = \begin{bmatrix} \sigma^2_{11} & \sigma^2_{12} & \cdots & \sigma^2_{1(n\times m)} \\ \sigma^2_{21} & \sigma^2_{22} & \cdots & \sigma^2_{2(n\times m)} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma^2_{(n\times m)1} & \sigma^2_{(n\times m)2} & \cdots & \sigma^2_{(n\times m)(n\times m)} \end{bmatrix} \qquad \text{Initial Guess } X_0 = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

Loop Equations:

Loop until ΔX is small, or more robustly, loop until S_0^2 increases. S_0^2 will increase slightly when you get down to really really small numbers and the cpu starts rounding. Caveat: it will also increase if you have a really bad initial guess, and it starts diverging.

$$\text{Equivalent Weight} = W_{eq} = inv(B\Sigma B^T)$$
$$\text{Loop Delta Estimate} = \Delta X = inv(J^T W_{eq} J) J^T W_{eq} K$$
$$\text{Loop Estimate} = \hat{X}_i = X_{i-1} + \Delta X$$
$$\text{Equivalent Residuals} = V_{eq} = K_i$$
$$\text{Reference Variance} = S_0^2 = \frac{V_{eq}^T W_{eq} V_{eq}}{dof}$$

make sure V = K in all nonlinear

Final Calculations

$$\text{Observation Residuals} = V = \Sigma B^T W_{eq} V_{eq}$$
$$\text{Unknowns} = \hat{X} = \hat{X}_i \text{ (Final Loop Estimate)}$$
$$\text{Cofactor Matrix} = Q_{xx} = inv(J^T W_{eq} J)$$
$$\text{Covariance Matrix of Unkowns} = \Sigma_{xx} = S_0^2 \times Q_{xx}$$
$$\text{Covariance Matrix of Observations} = \Sigma_{\hat{l}\hat{l}} = J \Sigma_{xx} J^T$$
$$\text{Standard Deviation of Solved Unknowns} = \sigma_{\hat{X}} = \sqrt{diag(\Sigma_{xx})}$$
$$\text{RMSE} = \sqrt{\frac{V_{eq} V_{eq}^T}{m}}$$

1.13.4 Sample Problem

Given the points and covariance matrix for each observation:

$$x = [10, 20, 60, 40, 85] \qquad y = [0, 15, 23, 25, 40]$$

$$\Sigma = \begin{bmatrix}
\sigma^2_{x_1x_1} & \sigma^2_{x_1y_1} & \sigma^2_{x_1x_2} & \sigma^2_{x_1y_2} & \sigma^2_{x_1x_3} & \sigma^2_{x_1y_3} & \sigma^2_{x_1x_4} & \sigma^2_{x_1y_4} & \sigma^2_{x_1x_5} & \sigma^2_{x_1y_5} \\
\sigma^2_{x_1y_1} & \sigma^2_{y_1y_1} & \sigma^2_{y_1x_2} & \sigma^2_{y_1y_2} & \sigma^2_{y_1x_3} & \sigma^2_{y_1y_3} & \sigma^2_{y_1x_4} & \sigma^2_{y_1y_4} & \sigma^2_{y_1x_5} & \sigma^2_{y_1y_5} \\
\sigma^2_{x_1x_2} & \sigma^2_{y_1x_2} & \sigma^2_{x_2x_2} & \sigma^2_{x_2y_2} & \sigma^2_{x_2x_3} & \sigma^2_{x_2y_3} & \sigma^2_{x_2x_4} & \sigma^2_{x_2y_4} & \sigma^2_{x_2x_5} & \sigma^2_{x_2y_5} \\
\sigma^2_{x_1y_2} & \sigma^2_{y_1y_2} & \sigma^2_{x_2y_2} & \sigma^2_{y_2y_2} & \sigma^2_{y_2x_3} & \sigma^2_{y_2y_3} & \sigma^2_{y_2x_4} & \sigma^2_{y_2y_4} & \sigma^2_{y_2x_5} & \sigma^2_{y_2y_5} \\
\sigma^2_{x_1x_3} & \sigma^2_{y_1x_3} & \sigma^2_{x_2x_3} & \sigma^2_{y_2x_3} & \sigma^2_{x_3x_3} & \sigma^2_{x_3y_3} & \sigma^2_{x_3x_4} & \sigma^2_{x_3y_4} & \sigma^2_{x_3x_5} & \sigma^2_{x_3y_5} \\
\sigma^2_{x_1y_3} & \sigma^2_{y_1y_3} & \sigma^2_{x_2y_3} & \sigma^2_{y_2y_3} & \sigma^2_{x_3y_3} & \sigma^2_{y_3y_3} & \sigma^2_{y_3x_4} & \sigma^2_{y_3y_4} & \sigma^2_{y_3x_5} & \sigma^2_{y_3y_5} \\
\sigma^2_{x_1x_4} & \sigma^2_{y_1x_4} & \sigma^2_{x_2x_4} & \sigma^2_{y_2x_4} & \sigma^2_{x_3x_4} & \sigma^2_{y_3x_4} & \sigma^2_{x_4x_4} & \sigma^2_{x_4y_4} & \sigma^2_{x_4x_5} & \sigma^2_{x_4y_5} \\
\sigma^2_{x_1y_4} & \sigma^2_{y_1y_4} & \sigma^2_{x_2y_4} & \sigma^2_{y_2y_4} & \sigma^2_{x_3y_4} & \sigma^2_{y_3y_4} & \sigma^2_{x_4y_4} & \sigma^2_{y_4y_4} & \sigma^2_{y_4x_5} & \sigma^2_{y_4y_5} \\
\sigma^2_{x_1x_5} & \sigma^2_{y_1x_5} & \sigma^2_{x_2x_5} & \sigma^2_{y_2x_5} & \sigma^2_{x_3x_5} & \sigma^2_{y_3x_5} & \sigma^2_{x_4x_5} & \sigma^2_{y_4x_5} & \sigma^2_{x_5x_5} & \sigma^2_{x_5y_5} \\
\sigma^2_{x_1y_5} & \sigma^2_{y_1y_5} & \sigma^2_{x_2y_5} & \sigma^2_{y_2y_5} & \sigma^2_{x_3y_5} & \sigma^2_{y_3y_5} & \sigma^2_{x_4y_5} & \sigma^2_{y_4y_5} & \sigma^2_{x_5y_5} & \sigma^2_{y_5y_5}
\end{bmatrix}$$

$$\Sigma = \begin{bmatrix}
45 & -30 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-30 & 30 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 20 & -10 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -10 & 70 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 80 & 4 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 4 & 4 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 40 & -13 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -13 & 60 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 30 & -25 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -25 & 30
\end{bmatrix}$$

Fit a line given the observation equation:

$$mx + b = y$$

With residuals for each observation:

$$F: \qquad m(x + v_x) + b - (y + v_y) = 0$$

$$\frac{\delta F}{\delta v_x} = m$$
$$\frac{\delta F}{\delta v_y} = -1$$

Solving for Partial Derivatives:

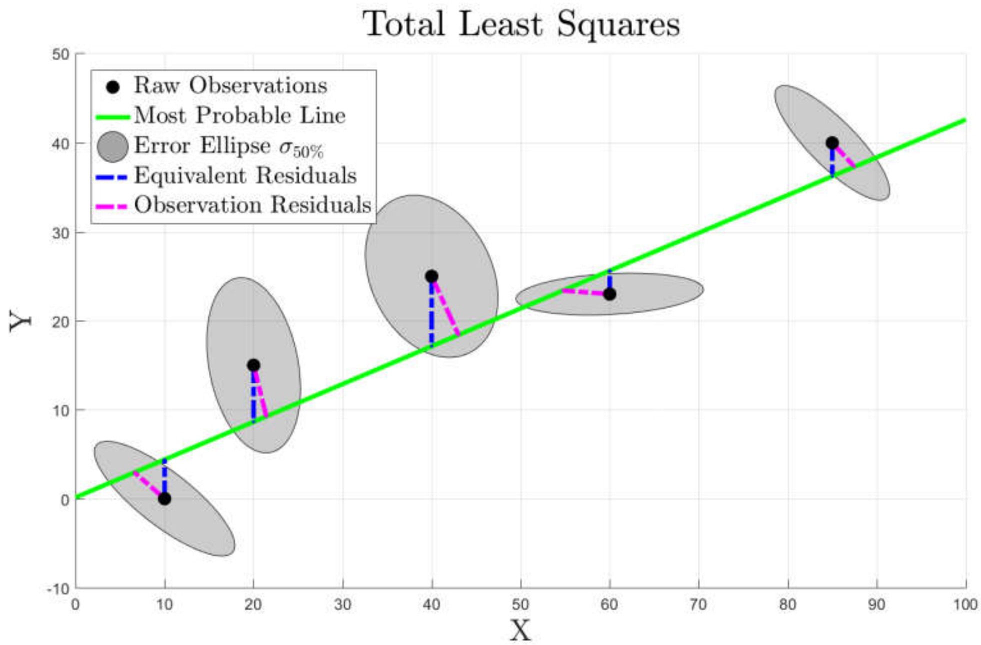
$$B = \begin{bmatrix}
\frac{\delta F}{\delta v_{x_1}} & \frac{\delta F}{\delta v_{y_1}} & 0 & 0 & \dots & 0 & 0 \\
0 & 0 & \frac{\delta F}{\delta v_{x_2}} & \frac{\delta F}{\delta v_{y_2}} & \dots & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \ddots & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \frac{\delta F}{\delta v_{x_5}} & \frac{\delta F}{\delta v_{y_5}}
\end{bmatrix} = \begin{bmatrix}
m & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & m & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & m & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & m & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & m & -1
\end{bmatrix}$$

$$V = \begin{bmatrix} v_{x_1} \\ v_{y_1} \\ v_{x_2} \\ v_{y_2} \\ v_{x_3} \\ v_{y_3} \\ v_{x_4} \\ v_{y_4} \\ v_{x_5} \\ v_{y_5} \end{bmatrix} \qquad J_i = \begin{bmatrix}
\frac{\delta F_1}{\delta m} & \frac{\delta F_1}{\delta b} \\
\frac{\delta F_2}{\delta m} & \frac{\delta F_2}{\delta b} \\
\frac{\delta F_3}{\delta m} & \frac{\delta F_3}{\delta b} \\
\frac{\delta F_4}{\delta m} & \frac{\delta F_4}{\delta b} \\
\frac{\delta F_5}{\delta m} & \frac{\delta F_5}{\delta b}
\end{bmatrix} = \begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ x_3 & 1 \\ x_4 & 1 \\ x_5 & 1 \end{bmatrix} \qquad \Delta X = \begin{bmatrix} \Delta m \\ \Delta b \end{bmatrix}$$

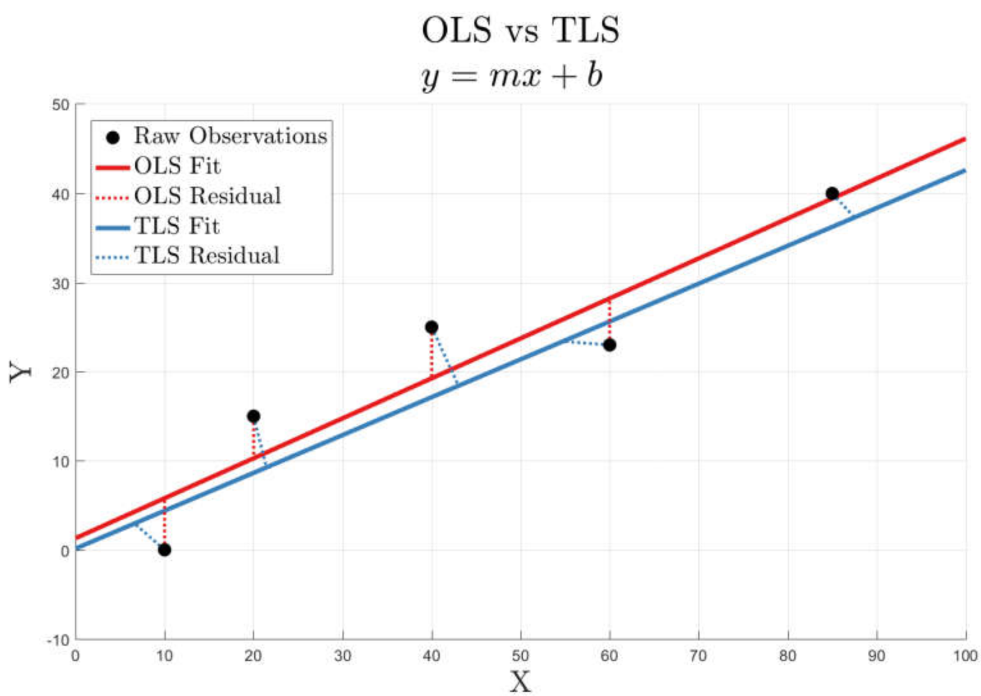
$$K_i = \begin{bmatrix} l_1 - F_1(X_i) \\ l_2 - F_2(X_i) \\ \vdots \\ l_m - F_m(X_i) \end{bmatrix} = \begin{bmatrix} 0 - (m_i x_1 + b_i - y_1) \\ 0 - (m_i x_2 + b_i - y_2) \\ 0 - (m_i x_3 + b_i - y_3) \\ 0 - (m_i x_4 + b_i - y_4) \\ 0 - (m_i x_5 + b_i - y_5) \end{bmatrix} \qquad \text{Initial Guess } X_0 = \begin{bmatrix} 2 \\ 5 \end{bmatrix}$$

Solve Using Equations:

$n = 2$	$m = 5$	$dof = 3$
$\hat{X} = \begin{bmatrix} 0.42 \\ 0.15 \end{bmatrix}$	$V_{eq} = \begin{bmatrix} -4.39 \\ 6.36 \\ -2.63 \\ 7.86 \\ 3.75 \end{bmatrix}$	$V = \begin{bmatrix} -3.39 \\ 2.95 \\ 1.43 \\ -5.75 \\ -5.25 \\ 0.40 \\ 3.01 \\ -6.58 \\ 2.50 \\ -2.69 \end{bmatrix}$
$\Sigma_{xx} = \begin{bmatrix} 0.01 & -0.60 \\ -0.60 & 36.87 \end{bmatrix}$	$\sigma_{\hat{X}} = \begin{bmatrix} 0.11 \\ 6.07 \end{bmatrix}$	$\Sigma_{\hat{l}\hat{l}} = \begin{bmatrix} 26.05 & 21.22 & 1.92 & 11.57 & -10.14 \\ 21.22 & 17.57 & 2.94 & 10.25 & -6.20 \\ 1.92 & 2.94 & 7.01 & 4.98 & 9.56 \\ 11.57 & 10.25 & 4.98 & 7.62 & 1.68 \\ -10.14 & -6.20 & 9.56 & 1.68 & 19.41 \end{bmatrix}$
$Q_{xx} = \begin{bmatrix} 0.02 & -0.78 \\ -0.78 & 48.19 \end{bmatrix}$	$\hat{L} = \begin{bmatrix} 4.39 \\ 8.64 \\ 25.63 \\ 17.14 \\ 36.25 \end{bmatrix}$	$S_0^2 = 0.76$ $RMSE = 5.46$



Notice how the observation residuals are not just in the y dimension. OLS is calculated with the same data, and plotted below for comparison.



1.13.5 Example Matlab Code

Algorithm 1.9: exampleTLS.m

```matlab
%% Example TLS Script
%% Input data
x = [10 20 60 40 85];
y = [0 15 23 25 40];
S = blkdiag([45 -30;-30 30],[20 -10;-10 70],[80 4;4 4],[40 -13;-13 60],[30 -25;-25 30]);

%% Solve TLS
X = [0.5; 0];                       % initial unknowns guess

So2 = inf; dSo2 = 1; iter = 0;      % initialize for while loop

m = numel(x);                       % number of observations
n = numel(X);                       % number of unknowns
dof = m-n;                          % degrees of freedom
while dSo2>0 && iter<100 %loop until So2 increases or exceed 100 iteration
    B = kron(eye(numel(y)),[X(1) -1]); %B
    J = [x(:) ones(size(x(:)))];   % Jacobian
    K = -(X(1)*x(:) + X(2) - y(:));% K Matrix
    Weq = inv(B*S*B');
    dX = (J'*Weq*J)\J'*Weq*K;       % Loop Delta Estimate
    X = X + dX;                     % Loop Estimate
    Veq = K;                        % Residuals
    dSo2 = So2 - Veq'*Weq*Veq/dof; % Change in Reference Variance
    So2 = (Veq'*Weq*Veq)/dof;       % Reference Variance
    iter = iter + 1;
end

V = S * B' * Weq * Veq;            % Observation Residuals
Q = inv(J'*Weq*J);                 % cofactor
Sx = So2 * Q;                      % covariance of unknowns
Sl = J * Sx * J';                  % covariance of observations
stdX = sqrt(diag(Sx));             % std of solved unknowns
Lhat = J * X;                      % predicted L values
RMSE = sqrt(Veq'*Veq/m);           % RMSE
```

Note: Matlab does not have a built in TLS function, but I wrote a function called LSRTLS.m

Algorithm 1.10: exampleTLS.m: Using lsrtls.m and anonymous function handles to perform TLS.

```matlab
%% Example Using LSRTLS
Jfun = @(X)([x(:) ones(size(x(:)))]);
Kfun = @(X)(-(X(1)*x(:) + X(2) - y(:)));
Bfun = @(X)(kron(eye(numel(y)),[X(1) -1]));
[X2,Sx2,lsainfo] = lsrtls(Jfun,Kfun,Bfun,Xo,S);
```