# Innotec Lectures
# Masaryk University Brno

# The Singular Value Decomposition

Prof. Walter Gander
ETH Zurich

Decenber 1, 2008

## 1 The Singular Value Decomposition

The singular value decomposition (SVD) of a matrix $A$ is very useful in the context of least squares problems. It also very helpful for analyzing properties of a matrix. With the SVD one x-rays a matrix!

**Theorem 1.1** *(The Singular Value Decomposition, SVD). Let $A$ be an $(m \times n)$ matrix with $m \geq n$. Then there exist orthogonal matrices $U$ $(m \times m)$ and $V$ $(n \times n)$ and a diagonal matrix $\Sigma = \operatorname{diag}(\sigma_1, \ldots, \sigma_n)$ $(m \times n)$ with $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n \geq 0$, such that*

$$A = U \Sigma V^T$$

*holds. If $\sigma_r > 0$ is the smallest* singular value *greater than zero then the matrix $A$ has rank $r$.*

**Definition 1.1** *The column vectors of $U = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_m]$ are called the* left singular vectors *and similarly $V = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n]$ are the* right singular vectors*. The values $\sigma_i$ are called the* singular values *of $A$.*

**Proof** The 2-norm of $A$ is defined by $\|A\|_2 = \max_{\|\boldsymbol{x}\|=1} \|A\boldsymbol{x}\|$. Thus there exists a vector $\boldsymbol{x}$ with $\|\boldsymbol{x}\| = 1$ such that

$$\boldsymbol{z} = A\boldsymbol{x}, \quad \|\boldsymbol{z}\| = \|A\|_2 =: \sigma.$$

Let $\boldsymbol{y} := \boldsymbol{z}/\|\boldsymbol{z}\|$. We have obtained $A\boldsymbol{x} = \sigma \boldsymbol{y}$ with $\|\boldsymbol{x}\| = \|\boldsymbol{y}\| = 1$.

Now we construct vectors orthogonal to $\boldsymbol{x}$ and similarly to $\boldsymbol{y}$ and form the matrices $V = [\boldsymbol{x}, V_1]$ and $U = [\boldsymbol{y}, U_1]$. This could e.g. be done in Matlab by the command

```
> V = orth([x, rand(n, n-1)]), U = orth([y, rand(n, n-1)])
```

Now

$$A_1 = U^T A V = \left[\begin{array}{c} \boldsymbol{y}^T \\ U_1^T \end{array}\right] A\,[\boldsymbol{x}, V_1] = \left[\begin{array}{cc} \boldsymbol{y}^T A\boldsymbol{x} & \boldsymbol{y}^T A V_1 \\ U_1^T A\boldsymbol{x} & U_1^T A V_1 \end{array}\right] = \left[\begin{array}{cc} \sigma & \boldsymbol{w}^T \\ 0 & B \end{array}\right]$$

because $\boldsymbol{y}^T A\boldsymbol{x} = \boldsymbol{y}^T \sigma \boldsymbol{y} = \sigma \boldsymbol{y}^T \boldsymbol{y} = \sigma$ and $U_1^T A\boldsymbol{x} = \sigma U_1^T \boldsymbol{y} = 0$ since $U_1 \perp \boldsymbol{y}$.

We claim that $\boldsymbol{w}^T := \boldsymbol{y}^T A V_1 = 0$. In order to prove that we compute

$$A_1 \begin{pmatrix} \sigma \\ \boldsymbol{w} \end{pmatrix} = \begin{pmatrix} \sigma^2 + \|\boldsymbol{w}\|^2 \\ B\boldsymbol{w} \end{pmatrix}$$

and conclude from that equation that

$$\left\| A_1 \begin{pmatrix} \sigma \\ \boldsymbol{w} \end{pmatrix} \right\|^2 = (\sigma^2 + \|\boldsymbol{w}\|^2))^2 + \|B\boldsymbol{w}\|^2 \geq (\sigma^2 + \|\boldsymbol{w}\|^2)^2.$$

Now since $V$ and $U$ are orthogonal $\|A_1\|_2 = \|U^T A V\|_2 = \|A\|_2 = \sigma$ holds and

$$\sigma^2 = \|A_1\|^2 = \max_{\|\boldsymbol{x}\| \neq 0} \|A_1 \boldsymbol{x}\|^2 \geq \frac{\left\| A_1 \binom{\sigma}{\boldsymbol{w}} \right\|^2}{\left\| \binom{\sigma}{\boldsymbol{w}} \right\|^2} \geq \frac{(\sigma^2 + \|\boldsymbol{w}\|^2))^2}{\sigma^2 + \|\boldsymbol{w}\|^2}.$$

The last equation reads

$$\sigma^2 \geq \sigma^2 + \|\boldsymbol{w}\|^2,$$

and we conclude that $\boldsymbol{w} = 0$. Thus we have obtained

$$A_1 = U^T A V = \left[\begin{array}{cc} \sigma & 0 \\ 0 & B \end{array}\right].$$

We can now apply the same construction to the sub-matrix $B$ and thus finally end up with a diagonal matrix.

Though this proof is constructive the singular value decomposition is not computed in this way. An effective algorithm was designed by Golub and Reinsch [6]. They first transform the matrix by orthogonal Householder-transformations to bidiagonal form. Then the bidiagonal matrix is further diagonalized in a iterative process.

Let $r = \operatorname{rank}(A)$. Then $\sigma_{r+1} = \ldots = \sigma_n = 0$. Partition $U = [U_1, U_2]$ and $V = [V_1, V_2]$ where $U_1 = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_r]$ and $V_1 = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_r]$ have $r$ columns. Then with $\Sigma_r := \operatorname{diag}(\sigma_1, \ldots, \sigma_r)$:

$$\begin{aligned} A &= [U_1, U_2] \begin{pmatrix} \Sigma_r & 0 \\ 0 & 0 \end{pmatrix} [V_1, V_2]^T && (1) \\ &= U_1 \Sigma_r V_1^T && (2) \\ &= \sum_{i=1}^{r} \boldsymbol{u}_i \boldsymbol{v}_i^T \, \sigma_i && (3) \end{aligned}$$

Equation (1) is the *full decomposition* with square matrices $U$ and $V$. When making use of the zeros we obtain the *"economy"* or *"reduced" version* of the SVD given in Equations (2) and (3). In MATLAB there are two variants to compute the SVD:

```
> [U S V ] = svd(A)   % gives the full decomposition
> [U S V ] = svd(A,0) % gives an m by n matrix U
```

The call `svd(A,0)` computes a version between full and economic with a non-square matrix $U$.

**Example 1.1** *The matrix $A$ has rank one and its economic SVD is given by*

$$A = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \end{pmatrix} (2\sqrt{3}) \begin{pmatrix} \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} \end{pmatrix}$$

# 2 Applications of the SVD

In Equation (3) we have decomposed the matrix $A$ as a sum of matrices $\boldsymbol{u}_i\boldsymbol{v}_i^T$ of rank one. Since

$$||\boldsymbol{u}_i\boldsymbol{v}_i^T||_2^2 = \max_{\|\boldsymbol{x}\|=1} ||\|\boldsymbol{u}_i\boldsymbol{v}_i^T\boldsymbol{x}\|^2 = \max_{\|\boldsymbol{x}\|=1} \boldsymbol{x}^T\boldsymbol{v}_i\boldsymbol{u}_i^T\boldsymbol{u}_i\boldsymbol{v}_i^T\boldsymbol{x} = \max_{\|\boldsymbol{x}\|=1} \|\boldsymbol{v}_i^T\boldsymbol{x}\|^2 = \|\boldsymbol{v}_i^T\boldsymbol{v}_i\|^2 = 1$$

we see from Equation (3) that the matrix $A$ is computed by a weighted sum of matrices for which the value of the norm is the same. The singular values are the weights. The main contribution in the sum is given by the terms with the largest singular values. *We see that we may approximate $A$ by a lower rank matrix by dropping the smallest singular values, i.e., changing their values to zero.* In fact let $\mathcal{M}$ denote the set of $m \times n$ matrices with rank $p$. The solution of

$$\min_{X \in \mathcal{M}} ||A - X||_F$$

is given by

$$A_p = \sum_{i=1}^{p} \boldsymbol{u}_i\boldsymbol{v}_i^T \, \sigma_i. \tag{4}$$

A proof of this fact is given e.g. in [2].

**Theorem 2.1** *If $A = U\Sigma V^T$. The column vectors of $V$ are the eigenvectors of the matrix $A^TA$ to the eigenvalues $\sigma_i^2$, $i = 1, \ldots, n$. The column vectors of $U$ are the eigenvectors of the matrix $AA^T$.*

**Proof**

$$A^TA = (U\Sigma V^T)^T U\Sigma V^T = VDV^T, \quad D = \Sigma^T\Sigma = \mathrm{diag}(\sigma_1^2, \ldots, \sigma_n^2). \tag{5}$$

Thus $A^TAV = VD$ and $\sigma_i^2$ is an eigenvalue of $A^TA$. Similarly

$$AA^T = U\Sigma V(U\Sigma V^T)^T = U^T\Sigma\Sigma^T U^T, \tag{6}$$

where $\Sigma\Sigma^T = \mathrm{diag}(\sigma_1^2, \ldots, \sigma_n^2, 0, \ldots, 0)$.

**Theorem 2.2** *Let $A = U\Sigma V^T$. Then*

$$\|A\|_2 = \sigma_1, \qquad \textit{and } \|A\|_F = \sqrt{\sum_{i=1}^{n} \sigma_i^2}.$$

**Proof** Since $U$ and $V$ are orthogonal we have $\|A\|_2 = \|U\Sigma V^T\|_2 = \|\Sigma\|_2$. Now

$$\|\Sigma\|_2^2 = \max_{\|\boldsymbol{x}\|=1} (\sigma_1^2 x_1^2 + \cdots + \sigma_n^2 x_n^2) \leq \sigma_1^2 (x_1^2 + \cdots + x_n^2) = \sigma_1^2$$

and since the max is attained for $\boldsymbol{x} = \boldsymbol{e}_1$ it follows $\|A\|_2 = \sigma_1$.

For the Frobenius norm we have

$$\|A\|_F = \sqrt{\sum_{i,j} a_{ij}^2} = \sqrt{\operatorname{trace}(A^T A)} = \sqrt{\sum_{i=1}^{n} \sigma_i^2}.$$

**Theorem 2.3** *Let $A = U\Sigma V^T$. Then the problem*

$$\|A\boldsymbol{x}\|_2 = \min, \quad \textit{subject to } \|\boldsymbol{x}\|_2 = 1 \tag{7}$$

*has the solution $\boldsymbol{x} = \boldsymbol{v}_n$ and the value of the minimum is $\min_{\|\boldsymbol{x}\|_2=1} \|A\boldsymbol{x}\|_2 = \sigma_n$.*

**Proof** We make use of the fact that for orthogonal $V$ and $V^T\boldsymbol{x} = \boldsymbol{y}$ we have $||\boldsymbol{x}|| = ||VV^T\boldsymbol{x}|| = ||V\boldsymbol{y}|| = ||\boldsymbol{y}||$:

$$\begin{aligned}
\min_{\|\boldsymbol{x}\|_2=1} \|A\boldsymbol{x}\|_2 &= \min_{\|\boldsymbol{x}\|_2=1} \|U\Sigma V^T\boldsymbol{x}\|_2 = \min_{\|VV^T\boldsymbol{x}\|_2=1} \|U\Sigma(V^T\boldsymbol{x})\|_2 \\
&= \min_{\|\boldsymbol{y}\|_2=1} \|\Sigma\boldsymbol{y}\|_2 = \min_{\|\boldsymbol{y}\|_2=1} (\sigma_1^2 y_1^2 + \cdots + \sigma_n^2 y_n^2) \geq \sigma_n^2
\end{aligned}$$

The minimum is attained for $\boldsymbol{y} = \boldsymbol{e}_n$ thus for $\boldsymbol{x} = V\boldsymbol{y} = \boldsymbol{v}_n$.

## 2.1 The Condition of the Linear Least Squares Problem

The principle of Wilkinson states that *the result of a numerical computation is the result of an exact computation for a slightly perturbed problem.* This result allows us to estimate the influence of finite arithmetic. A problem is said to be well conditioned if the results do not differ too much when solving a perturbed problem. For an ill conditioned problem the solution of a perturbed problem may be very different.

Consider a system of linear equations $A\boldsymbol{x} = \boldsymbol{b}$ with $A$ $n \times n$ nonsingular and a perturbed system $(A + \epsilon E)\boldsymbol{x}(\epsilon) = \boldsymbol{b}$ where $\epsilon$ is small like e.g. the machine precision.

How do the solutions $\boldsymbol{x}(\epsilon)$ and $\boldsymbol{x} = \boldsymbol{x}(0)$ differ? We want to expand $\boldsymbol{x}(\epsilon) \simeq \boldsymbol{x}(0) + \dot{\boldsymbol{x}}(0)\epsilon$. The derivative $\dot{\boldsymbol{x}}(0)$ is obtained by differentiating:

$$\begin{aligned}
(A + \epsilon E)\boldsymbol{x}(\epsilon) &= \boldsymbol{b} \\
E\boldsymbol{x}(\epsilon) + (A + \epsilon E)\dot{\boldsymbol{x}}(\epsilon) &= 0 \\
\Rightarrow \dot{\boldsymbol{x}}(0) &= -A^{-1}E\boldsymbol{x}(0) \\
\Rightarrow \boldsymbol{x}(\epsilon) &\simeq \boldsymbol{x}(0) - A^{-1}E\boldsymbol{x}(0)\,\epsilon \\
||\boldsymbol{x}(\epsilon) - \boldsymbol{x}(0)|| &\simeq ||A^{-1}||\;||\epsilon\, E||\;||\boldsymbol{x}(0)||.
\end{aligned}$$

From the last equation we conclude for the relative error

$$\frac{||\boldsymbol{x}(\epsilon)-\boldsymbol{x}||}{||\boldsymbol{x}||} \simeq \underbrace{||A^{-1}||\,||A||}_{\substack{\kappa \\ \text{condition number}}} \frac{||\epsilon\, E||}{||A||}. \tag{8}$$

If we use the 2-norm as matrix norm then

$$||A|| := \max_{\boldsymbol{x}\neq 0} \frac{||A\boldsymbol{x}||_2}{||\boldsymbol{x}||_2} = \sigma_{\max}(A), \quad ||A^{-1}|| = \frac{1}{\sigma_{\min}(A)}$$

and the condition number is computed by

$$\kappa = \frac{\sigma_{\max}(A)}{\sigma_{\min}(A)} = \texttt{cond(A)} \text{ in } \textsc{Matlab}.$$

Thus according to the principle of Wilkinson we have to expect that the numerical solution may deviate by about $\kappa$ units in the last digit from the exact solution.

A similar but more difficult computation shows (see [1]) for

$$||\boldsymbol{b}-A\boldsymbol{x}||_2 = \min \quad \text{and} \quad ||\boldsymbol{b}-(A+\epsilon E)\boldsymbol{x}||_2 = \min$$

the estimate

$$\frac{||\boldsymbol{x}(\epsilon)-\boldsymbol{x}||}{||\boldsymbol{x}||} \simeq \left(2\kappa + \kappa^2 \frac{||\boldsymbol{r}||}{||A||\,||\boldsymbol{x}||}\right) \frac{||\epsilon\, E||}{||A||} \quad \text{(Golub-Pereyra)}, \tag{9}$$

where

$$\kappa := ||A||\,||A^+|| = \frac{\sigma_1(A)}{\sigma_r(A)},$$

with $A^+$ the *Pseudoinverse*, see the next section.

Equation (9) tells us again what accuracy we can expect from the numerical solution. We have to distinguish between good and bad models. For good models, i.e. if the residual $||\boldsymbol{r}||$ is small, the situation is similar as for linear equations (8): the error in the solution may deviate by about $\kappa$ units in the last digit from the exact solution. However, when the model is bad, i.e. when $||\boldsymbol{r}||$ is large then also the condition is worse and we must expect a larger error in the computed solution.

## 2.2 Normal Equations and Condition

Using the normal equations we have to expect worse numerical results as predicted by Equation (9) also for good models. Forming $A^TA$ leads to a matrix with a squared condition number compared to the original matrix $A$. If $A = U\Sigma V^T$ has rank $n$ then

$$\kappa(A^TA) = \kappa(V^T\Sigma^TUU^T\Sigma V) = \kappa(V^T\Sigma^T\Sigma V) = \frac{\sigma_1^2}{\sigma_n^2} = \kappa(A)^2.$$

Forming $A^TA$ may result in a loss of information as a famous example by P. Läuchli shows:

$$A = \begin{pmatrix} 1 & 1 \\ \delta & 0 \\ 0 & \delta \end{pmatrix}, \quad A^TA = \begin{pmatrix} 1+\delta^2 & 1 \\ 1 & 1+\delta^2 \end{pmatrix}.$$

If $\delta < \sqrt{\epsilon}$ (with $\epsilon$ machine precision) then numerically $1+\delta^2 = 1$ and the matrix of the normal equations becomes singular though $A$ has also numerically rank 2.

## 2.3 The Pseudoinverse

**Definition 2.1** *Let $A = U\Sigma V^T$ be the singular value decomposition with*

$$\Sigma = \begin{pmatrix} \Sigma_r \\ 0 \end{pmatrix} \in \mathbb{R}^{m\times n}, \quad \Sigma_r := \operatorname{diag}(\sigma_1, \ldots, \sigma_r, 0, \ldots, 0) \in \mathbb{R}^{n\times n}$$

*The the matrix $A^+ = V\Sigma^+ U^T$ with*

$$\Sigma^+ = (\Sigma_r^+ \;\; 0) \in \mathbb{R}^{n\times m}, \quad \Sigma_r^+ := \operatorname{diag}(\frac{1}{\sigma_1}, \ldots, \frac{1}{\sigma_r}, 0, \ldots, 0) \in \mathbb{R}^{n\times n} \tag{10}$$

*is called the pseudo-inverse of $A$.*

We have discussed the SVD only for the case in which $A \in \mathbb{R}^{m\times n}$ with $m \geq n$. This was mainly for simplicity. The SVD exists for any matrix. It is usually computed such that the singular values are ordered decreasingly. The representation $A^+ = V\Sigma^+ U^T$ of the pseudo-inverse is also already a SVD, except that the singular values $\frac{1}{\sigma_1}, \cdots, \frac{1}{\sigma_r}$ are ordered increasingly. By simultaneously permuting rows and columns one can reorder the decomposition and bring it into standard form with decreasing elements in $\Sigma^+$.

**Theorem 2.4** *(Penrose Equations). $Y = A^+$ is the only solution of the matrix equations*

$$\begin{array}{rrcl rrcl} (i) & AYA & = & A & (ii) & YAY & = & Y \\ (iii) & (AY)^T & = & AY & (iv) & (YA)^T & = & YA \end{array}$$

**Proof** It is simple to verify that $A^+$ is a solution. Inserting the SVD e.g. in (i) we get

$$AA^+A = U\Sigma V^T V\Sigma^+ U^T U\Sigma V^T = U\Sigma\Sigma^+\Sigma V^T = U\Sigma V^T = A$$

More challenging is to prove uniqueness. To do this assume that $Y$ is also a solution. Then

$$\begin{aligned}
Y &= YAY \quad \text{because of (ii)} \\
&= (YA)^T Y = A^T Y^T Y \quad \text{because of (iv)} \\
&= (AA^+A)^T Y^T Y = A^T (A^+)^T A^T Y^T Y \quad \text{because of (i)} \\
&= A^T (A^+)^T YAY \quad \text{because of (iv)} \\
&= A^T (A^+)^T Y = (A^+A)^T Y \quad \text{because of (ii)} \\
&= A^+AY \quad \text{because of (iv)} \\
&= A^+AA^+AY \quad \text{because of (i)} \\
&= A^+(AA^+)^T (AY)^T = A^+(A^+)^T A^T Y^T A^T \quad \text{because of (iii)} \\
&= A^+(A^+)^T A^T \quad \text{because of (i)} \\
&= A^+(AA^+)^T = A^+AA^+ \quad \text{because of (iii)} \\
Y &= A^+ \quad \text{because of (ii)}
\end{aligned}$$

## 2.4 Fundamental Subspaces

Associated with a matrix $A \in \mathbb{R}^{m \times n}$ are four fundamental subspaces:

**Definition 2.2** *1.* $\mathcal{R}(A) = \{\boldsymbol{y} | \boldsymbol{y} = A\boldsymbol{x}, \quad \boldsymbol{x} \in \mathbb{R}^n\} \subset \mathbb{R}^m$ *the* range *or* column space*.*

*2.* $\mathcal{R}(A)^\perp$ *the orthogonal complement of* $\mathcal{R}(A)$*. If* $\boldsymbol{z} \in \mathcal{R}(A)^\perp$ *then* $\boldsymbol{z}^T\boldsymbol{y} = 0, \quad \forall \boldsymbol{y} \in \mathcal{R}(A)$*.*

*3.* $\mathcal{R}(A^T) = \{\boldsymbol{z} | \boldsymbol{z} = A^T\boldsymbol{y}, \quad \boldsymbol{y} \in \mathbb{R}^m\} \subset \mathbb{R}^n$ *the* row space*.*

*4.* $\mathcal{N}(A) = \{\boldsymbol{x} | A\boldsymbol{x} = 0\}$ *the* null space*.*

**Theorem 2.5** *The following relations hold*

*1.* $\mathcal{R}(A)^\perp = \mathcal{N}(A^T)$*. Thus* $\mathbb{R}^m = \mathcal{R}(A) \oplus \mathcal{N}(A^T)$*.*

*2.* $\mathcal{R}(A^T)^\perp = \mathcal{N}(A)$*. Thus* $\mathbb{R}^n = \mathcal{R}(A^T) \oplus \mathcal{N}(A)$*.*

**Proof** Let $\boldsymbol{y} \in \mathcal{R}(A)$ and $\boldsymbol{z} \in \mathcal{R}(A)^\perp$. The by definition

$$0 = \boldsymbol{y}^T\boldsymbol{z} = (A\boldsymbol{x})^T\boldsymbol{z} = \boldsymbol{x}^T(A^T\boldsymbol{z}), \quad \forall \boldsymbol{x}.$$

Thus it follows that $A^T\boldsymbol{z} = 0$ which means $\boldsymbol{z} \in \mathcal{N}(A^T)$ and therefore $\mathcal{R}(A)^\perp \subset \mathcal{N}(A^T)$.

On the other hand let $\boldsymbol{y} \in \mathcal{R}(A)$ and $\boldsymbol{z} \in \mathcal{N}(A^T)$. Then we have

$$\boldsymbol{y}^T\boldsymbol{z} = (A\boldsymbol{x})^T\boldsymbol{z} = \boldsymbol{x}^T(A^T\boldsymbol{z}) = \boldsymbol{x}^T 0 = 0$$

which means that $\boldsymbol{z} \in \mathcal{R}(A)^\perp$. Thus also $\mathcal{N}(A^T) \subset \mathcal{R}(A)^\perp$. The second statement is verified in the same way.

With the help of the pseudo-inverse we can describe *projectors* on these subspaces.

**Theorem 2.6**

$$\begin{array}{llll} 1. & P_{\mathcal{R}(A)} = AA^+ & 2. & P_{\mathcal{R}(A^T)} = A^+A \\ 3. & P_{\mathcal{N}(A^T)} = I - AA^+ & 4. & P_{\mathcal{N}(A)} = I - A^+A \end{array}$$

**Proof** We proof only the first relation. The other proofs are similar. Because of Relation (iii) in Theorem 2.4 we have $(AA^+)^T = AA^+$. Thus $P_{\mathcal{R}(A)}$ is symmetric. Furthermore $(AA^+)(AA^+) = (AA^+A)A^+ = AA^+$ because of (i). Thus $P_{\mathcal{R}(A)}$ is also idempotent and is a projector. Now let $\boldsymbol{y} = A\boldsymbol{x} \in \mathcal{R}(A)$; then $P_{\mathcal{R}(A)}\boldsymbol{y} = AA^+\boldsymbol{y} = AA^+A\boldsymbol{x} = A\boldsymbol{x} = \boldsymbol{y}$. So elements in $\mathcal{R}(A)$ are projected on themselves. Finally take $\boldsymbol{z} \perp \mathcal{R}(A) \iff A^T\boldsymbol{z} = 0$ then $P_{\mathcal{R}(A)}\boldsymbol{z} = AA^+\boldsymbol{z} = (AA^+)^T\boldsymbol{z} = (A^+)^TA^T\boldsymbol{z} = 0$.

Note that the projectors can be computed using the SVD. Let $U_1 \in \mathbb{R}^{m \times r}$, $U_2 \in \mathbb{R}^{m \times n-r}$, $V_1 \in \mathbb{R}^{\times r}$, $V_2 \in \mathbb{R}^{n \times n-r}$ and $\Sigma_r \in \mathbb{R}^{r \times r}$ in the following SVD

$$A = \begin{pmatrix} U_1 & U_2 \end{pmatrix} \begin{pmatrix} \Sigma_r & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} V_1^T \\ V_2^T \end{pmatrix}.$$

Then inserting this decomposition in the expressions for the projectors of Theorem 2.6 we obtain:

$$\begin{array}{llll} 1. & P_{\mathcal{R}(A)} = U_1U_1^T & 2. & P_{\mathcal{R}(A^T)} = V_1V_1^T \\ 3. & P_{\mathcal{N}(A^T)} = U_2U_2^T & 4. & P_{\mathcal{N}(A)} = V_2V_2^T \end{array}$$

## 2.5 General Solution of the Linear Least Squares Problem

We are now ready to describe the general solution for the linear least squares problem. We are given a system of equations with more equations than unknowns

$$A\boldsymbol{x} \approx \boldsymbol{b}.$$

In general $\boldsymbol{b}$ will not be in $\mathcal{R}(A)$ therefore the system will not have a solution. A consistent system can be obtained if we project $\boldsymbol{b}$ on $\mathcal{R}(A)$ :

$$A\boldsymbol{x} = AA^{+}\boldsymbol{b} \iff A(\boldsymbol{x} - A^{+}\boldsymbol{b}) = 0.$$

We conclude that $\boldsymbol{x} - A^{+}\boldsymbol{b} \in \mathcal{N}(A)$. That means

$$\boldsymbol{x} - A^{+}\boldsymbol{b} = (I - A^{+}A)\boldsymbol{w}$$

where we have generated an element in $\mathcal{N}(A)$ by projecting an arbitrary vector $\boldsymbol{w}$ onto it. Thus we have shown

**Theorem 2.7** *The general solution of the linear least squares problem $A\boldsymbol{x} \approx \boldsymbol{b}$ is*

$$\boldsymbol{x} = A^{+}\boldsymbol{b} + (I - A^{+}A)\boldsymbol{w}, \quad \boldsymbol{w} \text{ arbitrary.} \tag{11}$$

Using the expressions for projectors from the SVD we obtain for the general solution

$$\boldsymbol{x} = V_1 \Sigma_r^{-1} U_1^T \boldsymbol{b} + V_2 \boldsymbol{c} \tag{12}$$

where we have introduced the arbitrary vector $\boldsymbol{c} := V_2^T \boldsymbol{w}$.

Thus we have obtain an algorithm for computing the general solution of the linear least squares problem with (possibly) rank deficient coefficient matrix:

Algorithm 2.1: General solution of the linear least squares problem $A\boldsymbol{x} \approx \boldsymbol{b}$

1. Compute the SVD: `[U S V] = svd(A)`.
2. Make a rank decision, i.e. choose $r$ such that $\sigma_r > 0$ and $\sigma_{r+1} = \cdots = \sigma_n = 0$. This decision is necessary because rounding errors will prevent the zero singular values to be exactly zero.
3. Set `V1=V(:,1:r), V2= V(:,r+1:n), Sr=S(1:r,1:r), U1=U(:,1:r).`
4. The solution with minimal norm is `xm=V1*(Sr\U1'*b).`
5. The general solution is `x = xm + V2*c` with an arbitrary $\boldsymbol{c} \in \mathbb{R}^{n-r}$.

If $A$ has full rank ($\text{rank}(A) = n$) then the solution of the linear least squares problem is unique:

$$\boldsymbol{x} = A^{+}\boldsymbol{b} = V\Sigma^{+}U^T\boldsymbol{b}.$$

The matrix $A^{+}$ is called pseudo-inverse because in the full rank case the analogies of $A\boldsymbol{x} = \boldsymbol{b} \Rightarrow \boldsymbol{x} = A^{-1}\boldsymbol{b}$ and $A\boldsymbol{x} \approx \boldsymbol{b} \Rightarrow \boldsymbol{x} = A^{+}\boldsymbol{b}$ are obvious.

The general least squares solution presented in Theorem 2.7 is also valid for a system of equations $A\boldsymbol{x} = \boldsymbol{b}$ where $m \leq n$, i.e. an under-determined linear system with fewer equations than unknowns. In this case the $\boldsymbol{x} = V_1 \Sigma_r^{-1} U_1^T \boldsymbol{b}$ solves the problem

$$\min \|\boldsymbol{x}\| \quad \text{subject to } A\boldsymbol{x} = \boldsymbol{b}.$$

## 2.6 Fitting Lines

We consider the problem of fitting lines by minimizing the sum of squares of the distances to given points (see Chapter 6 in [3]). In the plane we can represent a straight line uniquely by the equations

$$c + n_1 x + n_2 y = 0, \quad n_1^2 + n_2^2 = 1. \tag{13}$$

The unit vector $(n_1, n_2)$ is orthogonal to the line. A point is on the line if its coordinates $(x, y)$ satisfy the first equation. On the other hand if $P = (x_P, y_P)$ is some point not on the line and we compute

$$r = c + n_1 x_P + n_2 y_P$$

then $|r|$ is its distance from the line. Therefore if we want to determine the line for which the sum of squares of the distances to given points is minimal, we have to solve the constrained least squares problem

$$\begin{pmatrix} 1 & x_{P_1} & y_{P_1} \\ 1 & x_{P_2} & y_{P_2} \\ \vdots & \vdots & \vdots \\ 1 & x_{P_m} & y_{P_m} \end{pmatrix} \begin{pmatrix} c \\ n_1 \\ n_2 \end{pmatrix} \approx \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \quad \text{subject to } n_1^2 + n_2^2 = 1. \tag{14}$$

Let $A$ be the matrix of the linear system (14). Using the QR decomposition $A = QR$ we can reduce the the linear system to $R\mathbf{x} \approx 0$, i.e., the problem becomes

$$\begin{pmatrix} r_{11} & r_{12} & r_{13} \\ 0 & r_{22} & r_{23} \\ 0 & 0 & r_{33} \end{pmatrix} \begin{pmatrix} c \\ n_1 \\ n_2 \end{pmatrix} \approx \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \quad \text{subject to } n_1^2 + n_2^2 = 1. \tag{15}$$

Since the nonlinear constraint only involves two unknowns; we now have to solve

$$\begin{pmatrix} r_{22} & r_{23} \\ 0 & r_{33} \end{pmatrix} \begin{pmatrix} n_1 \\ n_2 \end{pmatrix} \approx \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad \text{subject to} \quad n_1^2 + n_2^2 = 1. \tag{16}$$

The solution is obtained using Theorem 2.3. Inserting the values into the first component of (15) and setting it to zero, we then can compute $c$.

The constraint least squares problem

$$A \begin{pmatrix} \boldsymbol{c} \\ \boldsymbol{n} \end{pmatrix} \approx \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad \text{subject to} \quad ||\boldsymbol{n}||_2 = 1$$

is therefore solved by the following MATLAB function:

Algorithm 2.2: Constrained Linear Least Squares

```
function [c,n] = clsq(A,dim);
% [c,n] = CLSQ(A,dim)
% solves the constrained least squares Problem
% A (c n)' ~ 0 subject to norm(n,2)=1
% length(n) = dim
[m,p] = size(A);
if p < dim+1, error ('not enough unknowns'); end;
if m < dim, error ('not enough equations'); end;
m = min (m, p);
R = triu (qr (A));
[U,S,V] = svd(R(p-dim+1:m,p-dim+1:p));
n = V(:,dim);
c = -R(1:p-dim,1:p-dim)\R(1:p-dim,p-dim+1:p)*n;
```

## 2.7 Fitting Ellipses

We want to fit ellipses to measured points by minimizing the "algebraic distance" (see [4]). The solutions $\boldsymbol{x} = [x_1, x_2]$ of a quadratic equation

$$\boldsymbol{x}^T A \boldsymbol{x} + \boldsymbol{b}^T \boldsymbol{x} + c = 0 \tag{17}$$

are points on an ellipse if $A$ is symmetric and definite (i.e. if $\det(A) = a_{11}a_{22} - a_{12}^2 > 0$). For each measured points $\boldsymbol{x}_i$ we obtain by inserting it in (17) an equation for the unknown coefficients $\boldsymbol{u} = [a_{11}, a_{12}, a_{22}, b_1, b_2, c]$. Since (17) is homogeneous in the coefficients, we need some normalizing condition in order to make the solution unique. A possibility that includes all cases is to normalize the coefficients by $\|\boldsymbol{u}\| = 1$. We obtain the problem

$$||B\boldsymbol{u}|| = \min \quad \text{subject to } ||\boldsymbol{u}|| = 1$$

which can be solved using the SVD. If we write the measured points as rows in the matrix $X$:

```
X =[-2.8939    4.1521
    -2.0614    2.1684
    -0.1404    1.9764
     2.6772    3.0323
     5.1746    5.7199
     3.2535    8.1196
    -0.1724    6.8398 ]
```

then

```
B =  [X(:,1).^2  X(:,1).*X(:,2) X(:,2).^2 ...
           X(:,1) X(:,2) ones(size(X(:,1)))]
```

The solution is obtained using Theorem 2.3:

```
 [U S V]= svd(B);
 u = V(:,6);
 A = [u(1) u(2)/2; u(2)/2 u(3)];
 b  = [u(4); u(5)]; c = u(6);
```

We obtain the coefficients:

$$A = \begin{pmatrix} -0.0316 & 0.0227 \\ 0.0227 & -0.0589 \end{pmatrix} \qquad b = \begin{pmatrix} -0.1484 \\ 0.5316 \end{pmatrix} \qquad c = -0.8300$$

If the coefficients $\boldsymbol{u} = [a_{11}, a_{12}, a_{22}, b_1, b_2, c]$ are known, we can compute the geometric quantities, the axes and the center point, as follows. To find a coordinate system in which then axes of the ellipse are parallel to the coordinate axis, we compute the eigenvalue decomposition:

$$A = QDQ^T, \quad Q \quad \text{orthogonal} \quad D = \operatorname{diag}(\lambda_1, \lambda_2). \tag{18}$$

Introducing the new variable $\bar{\boldsymbol{x}} = Q^T\boldsymbol{x}$, Equation (17) becomes

$$\bar{\boldsymbol{x}}^T Q^T A Q \bar{\boldsymbol{x}} + (Q^T\boldsymbol{b})^T \bar{\boldsymbol{x}} + c = 0,$$

or written in components (using $\bar{\boldsymbol{b}} = Q^T\boldsymbol{b}$):

$$\lambda_1 \bar{x_1}^2 + \lambda_2 \bar{x_2}^2 + \bar{b_1}\bar{x_1} + \bar{b_2}\bar{x_2} + c = 0.$$

We transform this equation into the “normal form”:

$$\frac{(\bar{x_1} - \bar{z}_1)^2}{a^2} + \frac{(\bar{x_2} - \bar{z}_2)^2}{b^2} = 1.$$

$(\bar{z}_1, \bar{z}_2)$ is the center in the rotated coordinate system:

$$(\bar{z}_1, \bar{z}_2) = \left(-\frac{\bar{b_1}}{2\lambda_1}, -\frac{\bar{b_2}}{2\lambda_2}\right)$$

To obtain the center in the non rotated system we have to transform the coordinates: $\boldsymbol{z} = Q\bar{\boldsymbol{z}}$. The axis are given by

$$a = \sqrt{\frac{\bar{b_1^2}}{4\lambda_1^2} + \frac{\bar{b_2^2}}{4\lambda_1\lambda_2} - \frac{c}{\lambda_1}}, \quad b = \sqrt{\frac{\bar{b_1^2}}{4\lambda_1\lambda_2} + \frac{\bar{b_2^2}}{4\lambda_2^2} - \frac{c}{\lambda_2}}.$$

We have now all the elements to write a Matlab function to fit an ellipse to measured points:

Algorithm 2.3: Algebraic Ellipse Fit

```
function [z, a, b, alpha] = algellipse(X);
%  [z, a, b, alpha] = algellipse(X)
% fits an ellipse by minimizing the ‘‘algebraic distance’’
% to given points  Pi = [X(i,1), X(i,2)]
% in the least squares sense   x’A x + bb’x + c = 0
% z is the center.  a,b the main axes
% alpha angle between a and x-axe
```

```
[U S V]= svd([X(:,1).^2  X(:,1).*X(:,2) X(:,2).^2 ...
            X(:,1) X(:,2) ones(size(X(:,1)))]);
u = V(:,6);
A = [u(1) u(2)/2; u(2)/2 u(3)];
bb  = [u(4); u(5)]; c = u(6);

[Q D] = eig(A);
alpha = atan2(Q(2,1), Q(1,1));
bs = Q'*bb;
zs = -(2*D)\bs;  z = Q*zs;
h = -bs'*zs/2-c;
a = sqrt(h/D(1,1));
b = sqrt(h/D(2,2));
```

If we want to draw the ellipse we do this best using polar coordinates:

Algorithm 2.4: Drawing an Ellipse

```
function drawellipse(C, a, b, alpha)
 % drawellipse(C, a, b, alpha)
 % draw ellipse with center C  semiaxis a  and b
 % alpha angle between a and x-axe
 %
   s = sin(alpha); c = cos(alpha);
  Q =[c -s; s c];
   theta = [0:0.02:2*pi];
   u = diag(C)*ones(2,length(theta)) + ...
       Q*[a*cos(theta); b*sin(theta)];
   plot(u(1,:),u(2,:));
   plot(C(1),C(2),'+');
```

Finally here is the main program for computing and drawing Figure 1:

```
X =[-2.8939   4.1521
    -2.0614   2.1684
    -0.1404   1.9764
     2.6772   3.0323
     5.1746   5.7199
     3.2535   8.1196
    -0.1724   6.8398 ]

axis([0 10 0 10]); axis('equal'); hold
plot(X(:,1), X(:,2),'o');
[z, a, b, alpha] = algellipse(X)
drawellipse(z, a, b, alpha)
```

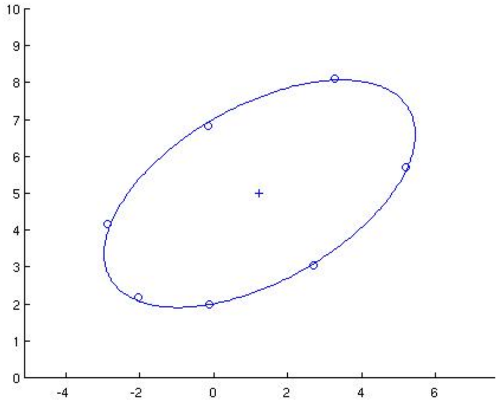

Figure 1: Fitting an Ellipse to Measured Points

# 3 Fitting Hyperplanes–Collinearity Test

Function `clsq` can be used to fit an $(n-1)$-dimensional hyperplane in $\mathbb{R}^n$ to given points. Let the rows of the matrix $X = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m]^T$ contain the coordinates of the given points, i.e. point $P_i$ has the coordinates $\mathbf{x}_i = X(i,:)$, $i = 1, \ldots, m$. Then the call

```
    >> [c, N] = clsq ([ones(m,1) X], n);
```

determines the hyperplane in normal form $c + N_1 y_1 + N_2 y_2 + \ldots + N_n y_n = 0$.

In this section, we will show how we can compute also best fit hyperplanes of lower dimensions $s$, where $1 \le s \le n-1$. We follow the theory developed in [5]. An $s$-dimensional hyperplane $\alpha$ in $\mathbb{R}^n$ can be represented in parameter form:

$$\alpha: \quad \mathbf{y} = \mathbf{p} + \mathbf{a}_1 t_1 + \mathbf{a}_2 t_2 + \cdots \mathbf{a}_s t_s = \mathbf{p} + A\mathbf{t}. \tag{19}$$

In this equation $\mathbf{p}$ is a point on the plane and $\mathbf{a}_i$ are linearly independent direction vectors, thus the hyperplane is determined by the data $\mathbf{p}$ and $A = [\mathbf{a}_1, \ldots, \mathbf{a}_s]$.

Without loss of generality, we assume that $A$ is orthogonal, i.e., $A^T A = I_s$. If we now want to fit a hyperplane to the given set of points $X$, then we have to minimize the distance of the points to the plane. The distance $d_i$ of point $P_i = \mathbf{x}_i$ to the hyperplane is given by

$$d_i = \min_{\mathbf{t}} ||\mathbf{p} - \mathbf{x}_i + A\mathbf{t}||_2.$$

To determine the minimum we solve $\operatorname{grad} d_i^2 = 2A^T(\mathbf{p} - \mathbf{x}_i + A\mathbf{t}) = \mathbf{0}$ for $\mathbf{t}$, and, since $A$ is orthogonal, we obtain

$$\mathbf{t} = A^T(\mathbf{x}_i - \mathbf{p}). \tag{20}$$

Therefore the distance becomes

$$d_i^2 = ||\mathbf{p} - \mathbf{x}_i + AA^T(\mathbf{x}_i - \mathbf{p})||_2^2 = ||\mathcal{P}(\mathbf{x}_i - \mathbf{p})||_2^2,$$

where we denoted by $\mathcal{P} = I - AA^T$, the projector onto the complement of the range of $A$, i.e. on the null space of $A^T$.

Our objective is to minimize the sum of squares of the distances of all points to the hyperplane. We want to minimize the function

$$F(\mathbf{p}, A) = \sum_{i=1}^{m} ||\mathcal{P}(\mathbf{x}_i - \mathbf{p})||_2^2. \tag{21}$$

A necessary condition is $\operatorname{grad} F = \mathbf{0}$. We first consider the first part of the gradient, the partial derivative

$$\frac{\partial F}{\partial \mathbf{p}} = -\sum_{i=1}^{m} 2\mathcal{P}^T\mathcal{P}(\mathbf{x}_i - \mathbf{p}) = -2\mathcal{P}(\sum_{i=1}^{m} \mathbf{x}_i - m\mathbf{p}) = \mathbf{0},$$

where we made use of the property of a projector $\mathcal{P}^T\mathcal{P} = \mathcal{P}$. Since $\mathcal{P}$ projects the vector $\sum_{i=1}^{m} \mathbf{x}_i - m\mathbf{p}$ onto $\mathbf{0}$, this vector must be in the range of $A$, i.e.

$$\mathbf{p} = \frac{1}{m}\sum_{i=1}^{m} \mathbf{x}_i + A\tau. \tag{22}$$

Inserting this expression into Equation (21), the objective function to be minimized simplifies to

$$G(A) = \sum_{i=1}^{m} ||\mathcal{P}\hat{\mathbf{x}}_i||_2^2 = ||\mathcal{P}\hat{X}^T||_F^2, \tag{23}$$

where we put

$$\hat{\mathbf{x}}_i = \mathbf{x}_i - \frac{1}{m}\sum_{i=1}^{m} \mathbf{x}_i,$$

and where we used the *Frobenius norm* a matrix ($||A||_F^2 := \sum_{i,j} a_{ij}^2$). Now since $\mathcal{P}$ is symmetric, we may also write

$$G(A) = ||\hat{X}\mathcal{P}||_F^2 = ||\hat{X}(I - AA^T)||_F^2 = ||\hat{X} - \hat{X}AA^T||_F^2. \tag{24}$$

If we define $Y := \hat{X}AA^T$, which is a matrix of rank $s$, then we can consider the problem of minimizing

$$||\hat{X} - Y||_F^2 = \min, \quad \text{subject to} \quad \operatorname{rank}(Y) = s.$$

We discussed this problem already (see Equation 4). The solution is:

1. Compute the singular value decomposition of $\hat{X} = U\Sigma V^T$, with

   $$U, V \quad \text{orthogonal and} \quad \Sigma = diag(\sigma_1, \sigma_2 \ldots \sigma_n)$$

   and $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n$.

2. The minimizing matrix is then given by $Y = U\Sigma_s V^T$, where

   $$\Sigma_s = diag(\sigma_1, \sigma_2 \ldots, \sigma_s, 0, 0, \ldots, 0).$$

Now if $Y = U\Sigma_s V^T$, we have to find an orthogonal matrix $A$ such that $\hat{X}AA^T = Y$. It is easy to verify, that if we choose $A = V_1$ where $V_1 = V(:, 1{:}s)$, then $\hat{X}AA^T = U\Sigma_s V^T$. Thus the singular value decomposition of $\hat{X}$ gives us all the lower-dimensional hyperplanes that fit best the given points:

$$\mathbf{y} = \mathbf{p} + V_1\mathbf{t}, \quad \text{with} \quad \mathbf{p} = \frac{1}{m}\sum_{i=1}^{m}\mathbf{x}_i.$$

Notice that $V_2 = V(:, s+1{:}n)$ gives us also the normal form of the hyperplane: Here the hyperplane is described as the solution of the linear equations

$$V_2^T\mathbf{y} = V_2^T\mathbf{p}.$$

In order to compute the hyperplanes, we therefore essentially have to compute one singular value decomposition. This is done by the MATLAB function `hyper.m` (Algorithm 3.1).

Algorithm 3.1: Computation of Hyperplanes.

```
function [V,p] =  hyper(Q);
% Fits a hyperplane of dimension s <n
% to a set of given points Q(i,:) belonging to R^n.
% The hyperplane has the equation
% X = p + V(:,1:s)*tau (Parameter Form) or
% is defined as solution of the linear
% equations  V(:,s+1:n)'*(y - p)=0 (Normal form)
m = max(size(Q));
p = sum(Q)'/m;
Qt = Q - ones(size(Q))*diag(p);
[U S V] = svd(Qt, 0);
```

The reader should note that the statement `[U S V] = svd(Qt, 0)` computes the “economy size” singular value decomposition. If `Qt` is an $m$-by-$n$ matrix with $m > n$, then only the first $n$ columns of $U$ are computed, and $S$ is an $n$-by-$n$ matrix.

## 3.1 Total Least Squares

The linear least squares problem $A\boldsymbol{x} \approx \boldsymbol{b}$ has so far been solved by projecting the vector $\boldsymbol{b}$ on the range of $A$:

$$A\boldsymbol{x} = \mathcal{P}_{\mathcal{R}(A)}\boldsymbol{b} = AA^{+}\boldsymbol{b}.$$

*With “Total Least Squares” the system of equations is made consistent by changing both $A$ and $\boldsymbol{b}$.* We are looking for a matrix $\hat{A}$ and a vector $\hat{\boldsymbol{b}} \in \mathcal{R}(\hat{A})$ that differ as little as possible from the given data

$$||[A, \boldsymbol{b}] - [\hat{A}, \hat{\boldsymbol{b}}||_F = \min, \quad \text{subject to} \quad \hat{\boldsymbol{b}} \in \mathcal{R}(\hat{A}).$$

The constraint says that $\hat{C} := [\hat{A}, \hat{\boldsymbol{b}}]$ must have rank $n$. In general $C := [A, \boldsymbol{b}]$ will have rank $n+1$ thus our problem becomes

$$\min_{\operatorname{rank} \hat{C} = n} \|C - \hat{C}\|_F.$$

If we introduce $\Delta = C - \hat{C}$ and write the condition $\operatorname{rank} \hat{C} = n$ as $\hat{C}\boldsymbol{z} = 0$ with $\boldsymbol{z} = \binom{\boldsymbol{x}}{-1} \neq 0$ then the problem is

$$\|\Delta\|_F^2 = \min \quad \text{subject to} \quad (C + \Delta)\boldsymbol{z} = 0. \tag{25}$$

The solution is given by Equation (4). Let $[A, \boldsymbol{b}] = C = U\Sigma V^T$ be the SVD. Then

$$[\hat{A}, \hat{\boldsymbol{b}}] = \hat{C} = \sum_{i=1}^{n} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T = U\hat{\Sigma}V^T, \quad \text{with} \quad \hat{\Sigma} = diag(\sigma_1, \ldots, \sigma_n, 0),$$

or equivalent

$$\Delta = -\sigma_{n+1} \boldsymbol{u}_{n+1} \boldsymbol{v}_{n+1}^T.$$

We have in this case $\hat{C}\boldsymbol{v}_{n+1} = [\hat{A}, \hat{\boldsymbol{b}}]\boldsymbol{v}_{n+1} = 0$ and if $v_{n+1,n+1} \neq 0$ then the total least squares solution exists:

$$\hat{A}\hat{\boldsymbol{x}} = \hat{\boldsymbol{b}}, \quad \hat{\boldsymbol{x}} = -\frac{1}{v_{n+1,n+1}} \boldsymbol{v}(1:n, n+1).$$

This leads to the following MATLAB function:

Algorithm 3.2: Total Least Squares

```
function x = lsqtotal(A,b);
% x = LSQTOTAL(A,b) computes the total least squares solution.
 [m,n]=size(A);
 [U, Sigma, V] = svd([A,b]);
 s = V(n+1,n+1);
 if s == 0 ,
  error('Total Least Squares Solution does not exist')
 end
 x = -V(1:n,n+1)/s;
```

The total least squares solution is unique only if $\sigma_n > \sigma_{n+1}$. Furthermore the total least squares solution solves the equation

$$(A^T A - \sigma_{n+1}^2 I)\hat{\boldsymbol{x}} = A^T \boldsymbol{b}. \tag{26}$$

**Proof** Since $C = [A, \boldsymbol{b}] = U\Sigma V^T$ we have $C\boldsymbol{v}_{n+1} = \sigma_{n+1}\boldsymbol{u}_{n+1}$ and therefore

$$C^T C \boldsymbol{v}_{n+1} = \sigma_{n+1} C^T \boldsymbol{u}_{n+1} = \sigma_{n+1}^2 \boldsymbol{v}_{n+1}.$$

Dividing the last equation by $v_{n+1,n+1}$ and replacing $C = [A, \boldsymbol{b}]$ we obtain

$$\left(\begin{array}{c|c} A^TA & A^T\boldsymbol{b} \\ \hline \boldsymbol{b}^TA & \boldsymbol{b}^T\boldsymbol{b} \end{array}\right) \begin{pmatrix} \hat{\boldsymbol{x}} \\ -1 \end{pmatrix} = \sigma_{n+1}^2 \begin{pmatrix} \hat{\boldsymbol{x}} \\ -1 \end{pmatrix}$$

and the first equation is our claim.

A variant of total least squares is given if some elements of the matrix $A$ have no errors. Let us consider $A\boldsymbol{x} \approx \boldsymbol{b}$ with $A = [A_1, A_2]$ where $A_1 \in \mathbb{R}^{m\times l}$ has no error. The total least squares problem $(A+E)\boldsymbol{x} = \boldsymbol{b}+\boldsymbol{r}$ subject to $\|E\|_F^2 + \|\boldsymbol{r}\|^2 = \min$ becomes:

$$\|\Delta\|_F^2 = \min \quad \text{subject to} \quad (C + \Delta)\boldsymbol{z} = 0$$

with $C = [A_1, A_2, \boldsymbol{b}]$, $\Delta = [0, \Delta_2]$ and $\Delta_2 = [E_2, \boldsymbol{r}]$. We can transform the constraint by pre-multiplying with an orthogonal matrix $Q^T$ such that

$$Q^T(C + \Delta)\boldsymbol{z} = \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix} \boldsymbol{u} + \begin{pmatrix} 0 & \tilde{\Delta}_{12} \\ 0 & \tilde{\Delta}_{22} \end{pmatrix} \boldsymbol{v} = 0, \quad R_{11} \in \mathbb{R}^{l\times l}, \boldsymbol{z} = \begin{pmatrix} \boldsymbol{u} \\ \boldsymbol{v} \end{pmatrix}.$$

Now $\|\Delta\|_F^2 = \|Q^T\Delta\|_F^2 = \|\tilde{\Delta}_{12}\|_F^2 + \|\tilde{\Delta}_{22}\|_F^2 = \min$ if we choose $\tilde{\Delta}_{12} = 0$ and minimize $\tilde{\Delta}_{22}$. So the algorithm for *constrained total least squares* becomes:

1. Reduce $C = [A_1, A_2, b]$ to

   $$Q^TC = \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix}$$

   by performing $l$ Householder transformations.

2. Compute $\hat{\boldsymbol{v}}$ the solution of the total least squares problem $(R_{22} + \tilde{\Delta}_{22})\boldsymbol{v} = 0$.

3. Solve $R_{11}\boldsymbol{u} = -R_{22}\hat{\boldsymbol{v}}$ for $\boldsymbol{u}$.

4. $\boldsymbol{z} = \binom{\boldsymbol{u}}{\hat{\boldsymbol{v}}}$ and $\boldsymbol{x} = [z_1, \ldots, z_n]$.

**Example 3.1** *We want to fit a line through given points $(\xi_i, \eta_i)$, $i = 1, \ldots, m$. The equation*

$$y = ax + b$$

*is used for regression and total least squares. The equations become*

$$\begin{pmatrix} \xi_1 & 1 \\ \vdots & \vdots \\ \xi_m & 1 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} \approx \begin{pmatrix} \eta_1 \\ \vdots \\ \eta_m \end{pmatrix}$$

*The geometric fit is done as explained in Section 2.6.*

```
xi = [1 2 4 5 6 7 9]'
eta = [4 1 5 6 5 7 9]'
axis([-1, 11 -1, 11])
hold
plot(xi,eta,'x')

% Minimizing geometric distance
A = [ones(size(xi)) xi eta]
[c, n] = clsq(A,2)
     xx = -1:11
plot(xx,-c/n(2)-n(1)*xx/n(2),'-')

% Regression
A =  [xi ones(size(xi))]
x = A\eta;
plot(xx,x(1)*xx+x(2),'-.')

% naive TLS
A =  [xi ones(size(xi))]
x =  lsqtotal(A,eta);
plot(xx,x(1)*xx+x(2),':')

% constrained TLS
C =  [ones(size(xi)) xi eta ];
m = max(size(C));
sqm = sqrt(m);
% construct Householder vector
u = ones(size(xi)); u(1) = u(1) +sqm;
u = u/sqrt(sqm*(1+sqm));
QC = C-u*u'*C;

AA = QC(2:m,2)
a =  lsqtotal(AA,QC(2:m,3))
b = -(QC(1,2)*a -QC(1,3))/ QC(1,1)
plot(xx, a*xx+b,'-')
```

We see from Figure 2 that naive total least squares is not useful to fit geometric elements. However, the solution with constrained total least squares, where the elements in the matrix that are 1 are treated as exact delivers the same solution as the geometric fit.

# 4 Bibliography

## References

[1] Åke Björck, *Numerical Methods for Least Squares Problems* SIAM, 1996.

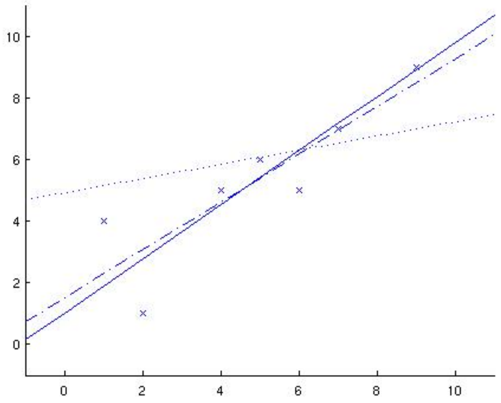

Figure 2: Line Fits: Geometric fit and constrained TLS −, Regression −· and naive TLS ··

[2] Gene H. Golub, Charles F. Van Loan, *Matrix Computations*, Johns Hopkins Series in the Mathematical Sciences, 3rd edition, 1996.

[3] W. Gander and J. Hřebíček, *Solving Problems in Scientific Computing using Maple and Matlab*, Springer Verlag, third edition 1997.

[4] Walter Gander, Gene H. Golub and Rolf Strebel, *Least-Squares Fitting of Circles and Ellipses*, BIT 34, 558-578, 1994.

[5] H. Späth, *Orthogonal least squares fitting with linear manifolds.* Numer. Math., 48, 1986, pp. 441–445.

[6] J. Wilkinson and Chr. Reinsch, *Linear Algebra*, Springer Verlag, 1971.