Department of Engineering Sciences and Mathematics
January 23, 2013
Ove Edlund

C0002M - Numerical Analysis

Some Notes on Least Squares, QR-factorization, SVD and Fitting

Contents

1 Introduction

While Chapra (2012) makes a good job in general, the book lacks in the linear algebra part. Doing LU-factorization without a permutation matrix is a bad idea. Cholesky factorization of a symmetric matrix is only possible if the matrix is positive definite. Those particular problems are handled in the slides of C0002M. When it comes to least squares though, I would like the course to go deeper than the book with a slightly different focus. This document serves as a complement in that respect and is part of the course literature.

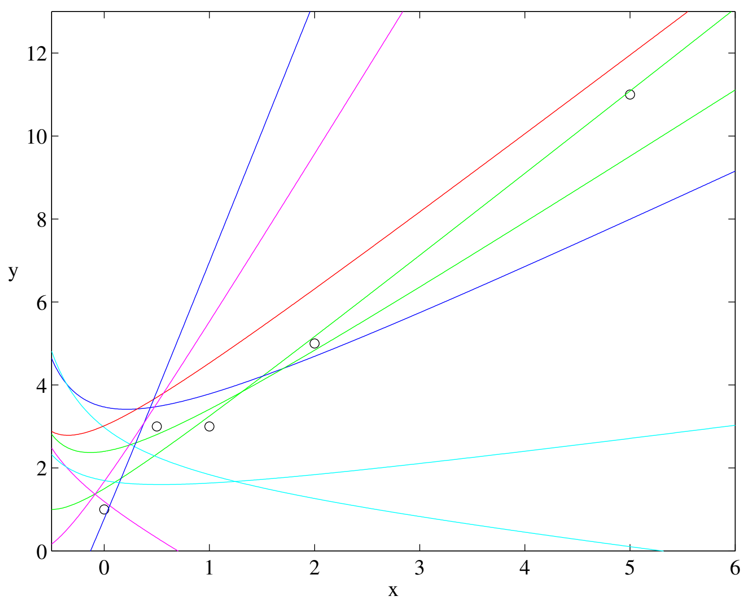

Figure 1: The data points and the model function $y = a_1 + a_2\,x + a_3/(x+1)$ for different choices of a_1, a_2 and a_3.

2 The Least squares problem

The least squares solution of a system of linear equations is defined as follows

Definition 1 *Given a matrix* $\mathbf{Z} \in \mathbb{R}^{n\times p}$ *and a right hand side* $\mathbf{y} \in \mathbb{R}^n$*, the least squares solution* $\mathbf{a} \in \mathbb{R}^p$ *is given by*

$$\min_{\mathbf{a}} \|\mathbf{y} - \mathbf{Z}\,\mathbf{a}\|_2 \,.$$

What we are trying to do is to find some kind of solution to $\mathbf{Z}\,\mathbf{a} = \mathbf{y}$ when the equality is not possible to fulfill. So we minimize the size of the difference between the left hand and the right hand side. The size that we minimize is found with the l_2-norm. Other norms could be used, but then we would not find the least squares solution. Other choices also require a much greater computational effort.

A typical situation where a least squares problem shows up is when a curve is fitted to some data points, i.e. when we do regression. For instance, consider the data points

x	0	0.5	1	2	5
y	1	3	3	5	11

to which we are trying to fit a model function with unknown parameters a_1, a_2 and a_3.

$$y = a_1 + a_2\,x + a_3\frac{1}{x+1}\,.$$

Figure 1 shows the data points and the model function for some different choices of the unknown parameters. We want to find the choice of parameters that give the "best fit" of the model function to the data points. To accomplish this, we plug the data points into the model function and receive a system of linear equations:

$$\begin{cases} 1 = a_1 + a_2 \cdot 0 \quad + a_3/(0+1) \\ 3 = a_1 + a_2 \cdot 0.5 + a_3/(0.5+1) \\ 3 = a_1 + a_2 \cdot 1 \quad + a_3/(1+1) \\ 5 = a_1 + a_2 \cdot 2 \quad + a_3/(2+1) \\ 11 = a_1 + a_2 \cdot 5 \quad + a_3/(5+1) \end{cases} \Leftrightarrow \begin{bmatrix} 1 & 0 & 1 \\ 1 & 0.5 & 1/1.5 \\ 1 & 1 & 1/2 \\ 1 & 2 & 1/3 \\ 1 & 5 & 1/6 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 3 \\ 3 \\ 5 \\ 11 \end{bmatrix}$$

If we could solve this system, we would get a function curve that passed all the data points. It is however not possible to solve the system, so we do the “second best”, we find the least squares solution to minimize the difference between the left hand and the right hand side. The following command sequence in Matlab finds our solution and makes a plot of it as shown in Figure 2.

```
>> x = [0 0.5 1 2 5]'
x =
         0
    0.5000
    1.0000
    2.0000
    5.0000
>> y = [1 3 3 5 11]'
y =
     1
     3
     3
     5
    11
>> Z = [x.^0 x 1./(x+1)]
Z =
    1.0000         0    1.0000
    1.0000    0.5000    0.6667
    1.0000    1.0000    0.5000
    1.0000    2.0000    0.3333
    1.0000    5.0000    0.1667
>> a = Z\y
a =
    1.5772
    1.8792
   -0.3221
>> xs = -0.2:0.01:5.2;
>> plot(xs,a(1)+a(2)*xs+a(3)./(xs+1),x,y,'o')
>> xlabel('x'), ylabel('y')
```

As seen in Figure 2 the least squares solution fits nicely to the general behavior of the data points, without passing through them. Also note that the least squares solution is found easily with “\” in Matlab.

The success of all this is due to that the unknowns a_1, a_2 and a_3 are weights in a linear combination of “basis functions”. This is also true for fitting polynomials for example. With this in mind we can express a general model

$$y = a_1\phi_1(x) + a_2\,\phi_2(x) + \cdots + a_p\,\phi_p(x) \tag{1}$$

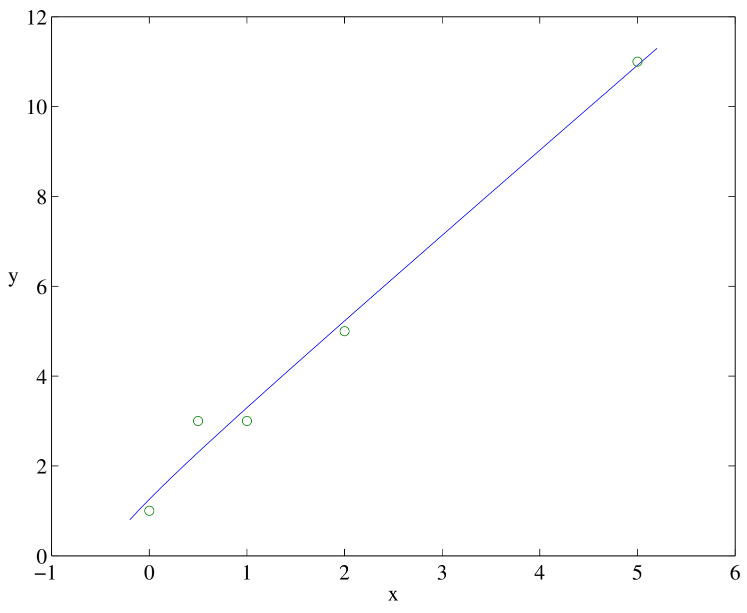

Figure 2: The least squares solution

with known "basis functions" $\phi_k(x)$. In the previous example we had $\phi_1(x) = 1$, $\phi_2(x) = x$ and $\phi_3(x) = 1/(x+1)$. Inserting the data points (x_i, y_i) into the model generates the system

$$\begin{bmatrix} \phi_1(x_1) & \phi_2(x_1) & \dots & \phi_p(x_1) \\ \phi_1(x_2) & \phi_2(x_2) & \dots & \phi_p(x_2) \\ \vdots & \vdots & & \vdots \\ \phi_1(x_n) & \phi_2(x_n) & \dots & \phi_p(x_n) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_p \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

which is solved in least square sense to find the weights a_k. If the model function does not have the form (1) it may still be possible to transform it to the right shape. This is the case for the model functions $y = a_1\, e^{a_2\, t}$ and $y = a_1\, t^{a_2}$, where the form (1) is realized by taking the logarithm of the expressions.

In other cases, for example with $y = a_1 \sin(a_2\, t) + a_3$, there is no cure, and more computationally demanding methods are necessary to find the least squares solution. There is a Matlab command **lsqcurvefit** to handle those non-linear cases.

3 Finding the least squares solution with the normal equations

There are several methods for finding the least squares solution. The one appearing most frequently in Linear algebra textbooks are the normal equations.

Theorem 1 (The Normal equations) *Given the matrix* $\mathbf{Z} \in \mathbb{R}^{n\times p}$ *and the right hand side* $\mathbf{y} \in \mathbb{R}^n$*, the solution set of the least squares problem*

$$\min_{\mathbf{a}\in\mathbb{R}^p} \|\mathbf{y} - \mathbf{Z}\,\mathbf{a}\|_2$$

is identical to the solution set of the normal equations

$$\mathbf{Z}^T\mathbf{Z}\,\mathbf{a} = \mathbf{Z}^T\mathbf{y}\,.$$

Proof See e.g. Lay (2003). □

So finding the least squares solution simply reduces to solving a system of linear equations with system matrix $\mathbf{Z}^T\mathbf{Z}$ and right hand side $\mathbf{Z}^T\mathbf{y}$. If $\mathbf{Z}$ has full column rank it holds that $\mathbf{Z}^T\mathbf{Z}$ is

symmetric : $\left(\mathbf{Z}^T\mathbf{Z}\right)^T = \mathbf{Z}^T\left(\mathbf{Z}^T\right)^T = \mathbf{Z}^T\mathbf{Z}$, and

positive definite : Let $\mathbf{q} = \mathbf{Z}\,\mathbf{x}$. Then

$$\mathbf{x}^T\mathbf{Z}^T\mathbf{Z}\,\mathbf{x} = (\mathbf{Z}\,\mathbf{x})^T\mathbf{Z}\,\mathbf{x} = \mathbf{q}^T\mathbf{q} = \|\mathbf{q}\|_2^2 > 0, \text{ if } \mathbf{q} \neq \mathbf{0} \Leftrightarrow \mathbf{x} \neq \mathbf{0}$$

With $\mathbf{Z}^T\mathbf{Z}$ as a symmetric and positive definite matrix, obviously it is favorable to use a Cholesky factorization to solve the normal equations. Continuing with the previous example, we thus find the least squares solution with the normal equations in this way in Matlab:

```
>> Matrix = Z'*Z
Matrix =
    5.0000    8.5000    2.6667
    8.5000   30.2500    2.3333
    2.6667    2.3333    1.8333
>> rhs = Z'*y
rhs =
   23.0000
   69.5000
    8.0000
>> U = chol(Matrix)
U =
    2.2361    3.8013    1.1926
         0    3.9749   -0.5535
         0         0    0.3237
>> a = U\(U'\rhs)
a =
    1.5772
    1.8792
   -0.3221
```

The restriction with full column rank makes sense from a numerical perspective, since the numerical algorithms we use depend on the existence of a unique solution. It is however possible to handle rank-deficient problems gracefully using the singular value decomposition (SVD), though we use it for other purposes in these notes.

One can show that the condition number

$$\mathrm{Cond}(\mathbf{Z}) = \max_{\mathbf{x}\neq\mathbf{0}} \frac{\|\mathbf{Z}\,\mathbf{x}\|_2}{\|\mathbf{x}\|_2} \Big/ \min_{\mathbf{x}\neq\mathbf{0}} \frac{\|\mathbf{Z}\,\mathbf{x}\|_2}{\|\mathbf{x}\|_2}$$

describes the relative perturbation sensitivity in the least squares solution $\mathbf{a}$, when $\mathbf{e} = \mathbf{y} - \mathbf{Z}\,\mathbf{a}$, the residual, is "small". Moreover

$$\mathrm{Cond}(\mathbf{Z}^T\mathbf{Z}) = \left(\mathrm{Cond}(\mathbf{Z})\right)^2$$

so, the normal eqations have a relative sensitivity that is *squared* compared to the original problem formulation. From this we draw the conclusion that any numerical method using the normal equations will be unstable, since the rounding errors will correspond to $\big(\mathrm{Cond}(\mathbf{Z})\big)^2$, while the mathematical problem has condition number $\mathrm{Cond}(\mathbf{Z})$. This bad numerical behaviour for the normal equations indeed does happen when we use the Cholesky factorization. We need better options!

4 The QR factorization and least squares

We get a stable numerical algorithm, by using the QR-factorization. To understand the factorization we need to recapture some special classes of matrices and some of their properties.

Definition 2 (**Orthonormal columns**) *A matrix $\mathbf{Q} \in \mathbb{R}^{n \times p}$ with $p \leq n$, has orthonormal columns if all columns in $\mathbf{Q}$ are orthogonal to every other column, and are normalized. More specificly, for*

$$\mathbf{Q} = [\, \mathbf{q}_1 \;\; \mathbf{q}_2 \;\; \cdots \;\; \mathbf{q}_p \,]$$

where $\mathbf{q}_i$ are the columns of $\mathbf{Q}$, it holds that $\mathbf{q}_i^T \mathbf{q}_k = 0$ for all $i \neq k$, and $\|\mathbf{q}_i\|_2 = 1$ for all i.

The following properties hold for matrices $\mathbf{Q} \in \mathbb{R}^{n \times p}$ with orthonormal columns:

- $\mathbf{Q}^T\mathbf{Q} = \mathbf{I}_p$ as a consequence of the definition.
- $\mathbf{Q}\mathbf{Q}^T$ is the transformation that does orthogonal projection from $\mathbb{R}^n$ onto $\mathrm{Col}(\mathbf{Q})$ (Lay 2003).

The point being made here is that $\mathbf{Q}\mathbf{Q}^T$ is *not* the identity matrix. If we let $p = n$ the situation changes though:

Definition 3 (**Orthogonal matrix**) *A square matrix with orthonormal columns is referred to as an orthogonal matrix. Thus if $\mathbf{Q}$ has orthonormal columns and $\mathbf{Q} \in \mathbb{R}^{n \times n}$, then $\mathbf{Q}$ is an orthogonal matrix.*

For orthogonal matrices it holds that $\mathbf{Q}^T\mathbf{Q} = \mathbf{Q}\mathbf{Q}^T = \mathbf{I}_n$. (The last part of the equality is true since $\mathrm{Col}(\mathbf{Q}) = \mathbb{R}^n$ and the orthogonal projection from $\mathbb{R}^n$ onto $\mathbb{R}^n$ is the unit transformation $\mathbf{I}_n$.) From this it also is clear that if $\mathbf{Q}$ is an orthogonal matrix, then $\mathbf{Q}^T$ is orthogonal as well. It is also obvious that $\mathbf{Q}^{-1} = \mathbf{Q}^T$, a property that only holds for orthogonal matrices.

Now that we have consulted our memory banks, we are ready for the QR-factorization.

Theorem 2 (**QR-factorization**) *Any matrix $\mathbf{Z} \in \mathbb{R}^{n \times p}$ with $p \leq n$ can be factorized as*

$$\mathbf{Z} = \mathbf{Q}\mathbf{R}$$

where $\mathbf{Q} \in \mathbb{R}^{n \times p}$ has orthonormal columns, and $\mathbf{R} \in \mathbb{R}^{p \times p}$ is upper triangular.

The proof is by construction. It is always possible to construct this factorization by the Grahm-Schmidt orthogonalization procedure, which is bad to use with floating point numbers, or by Householder reflections, which is good (Demmel 1997, Higham 1996). We will take a look at Householder reflections in Section 5.

We use this factorization for the least squares problem instead of the normal equations, as stated in the following theorem:

Theorem 3 (**QR-factorization and least squares**) *Given the matrix* $\mathbf{Z} \in \mathbb{R}^{n\times p}$ *and the right hand side* $\mathbf{y} \in \mathbb{R}^n$, *the solution set of the least squares problem*

$$\min_{\mathbf{a}\in\mathbb{R}^p} \|\mathbf{y} - \mathbf{Z}\,\mathbf{a}\|_2$$

is identical to the solution set of

$$\mathbf{R}\,\mathbf{a} = \mathbf{Q}^T\mathbf{y}$$

where the matrix $\mathbf{Q} \in \mathbb{R}^{n\times p}$ *with orthonormal columns and the upper triangular* $\mathbf{R} \in \mathbb{R}^{p\times p}$, *is a QR-factorization of* $\mathbf{Z}$.

So we get a system matrix that is upper triangular. We easily solve this system with backwards substitution. Yet again, we go to our example and do this in Matlab:

```
>> [Q,R] = qr(Z,0)
Q =
   -0.4472   -0.4277    0.7104
   -0.4472   -0.3019   -0.1043
   -0.4472   -0.1761   -0.4041
   -0.4472    0.0755   -0.4888
   -0.4472    0.8302    0.2868
R =
   -2.2361   -3.8013   -1.1926
         0    3.9749   -0.5535
         0         0    0.3237
>> rhs = Q'*y
rhs =
  -10.2859
    7.6480
   -0.1043
>> a = R\rhs
a =
    1.5772
    1.8792
   -0.3221
```

Obviously this works as stated in the theorem. We get the same solution as before. Note that, to get the QR-factorization described in this document, that we are interested in for solving least squares problems, we have to add an extra "`0`" argument to `qr(Z,0)`.

In order to prove Theorem 3, we first need to establish a property of the l_2-norm.

Lemma 1 *Multiplying a vector with an orthogonal matrix does not change its* l_2*-norm. Thus if* $\mathbf{Q}$ *is orthogonal it holds that*

$$\|\mathbf{Q}\mathbf{x}\|_2 = \|\mathbf{x}\|_2$$

Proof $\|\mathbf{Q}\mathbf{x}\|_2^2 = (\mathbf{Q}\mathbf{x})^T\mathbf{Q}\mathbf{x} = \mathbf{x}^T\mathbf{Q}^T\mathbf{Q}\mathbf{x} = \mathbf{x}^T\mathbf{x} = \|\mathbf{x}\|_2^2 \;\square$

Using this lemma we prove Theorem 3

Proof of Theorem 3 Given the QR-factorization $\mathbf{Z} = \mathbf{QR}$ we do an orthogonal extension of $\mathbf{Q}$, i.e. we add additional orthonormal columns to it to get an orthogonal matrix: $[\mathbf{Q}\ \widehat{\mathbf{Q}}] \in \mathbb{R}^{n\times n}$. The columns of $\widehat{\mathbf{Q}}$ represent the added orthonormal columns. Note that $\widehat{\mathbf{Q}}^T\mathbf{Q} = \mathbf{0}$. Also note that $[\mathbf{Q}\ \widehat{\mathbf{Q}}]^T$ also is an orthogonal matrix. Now

$$\begin{aligned}
\|\mathbf{y} - \mathbf{Z}\,\mathbf{a}\|_2^2 &= \|\mathbf{y} - \mathbf{Q}\mathbf{R}\,\mathbf{a}\|_2^2 = \left\|[\mathbf{Q}\ \widehat{\mathbf{Q}}]^T(\mathbf{y} - \mathbf{Q}\mathbf{R}\,\mathbf{a})\right\|_2^2 \\
&= \left\|\begin{bmatrix}\mathbf{Q}^T\\ \widehat{\mathbf{Q}}^T\end{bmatrix}(\mathbf{y} - \mathbf{Q}\mathbf{R}\,\mathbf{a})\right\|_2^2 = \left\|\begin{bmatrix}\mathbf{Q}^T(\mathbf{y} - \mathbf{Q}\mathbf{R}\,\mathbf{a})\\ \widehat{\mathbf{Q}}^T(\mathbf{y} - \mathbf{Q}\mathbf{R}\,\mathbf{a})\end{bmatrix}\right\|_2^2 \\
&= \left\|\begin{bmatrix}\mathbf{Q}^T\mathbf{y} - \mathbf{Q}^T\mathbf{Q}\mathbf{R}\,\mathbf{a}\\ \widehat{\mathbf{Q}}^T\mathbf{y} - \widehat{\mathbf{Q}}^T\mathbf{Q}\mathbf{R}\,\mathbf{a}\end{bmatrix}\right\|_2^2 = \left\|\begin{bmatrix}\mathbf{Q}^T\mathbf{y} - \mathbf{R}\,\mathbf{a}\\ \widehat{\mathbf{Q}}^T\mathbf{y}\end{bmatrix}\right\|_2^2 \\
&= \|\mathbf{Q}^T\mathbf{y} - \mathbf{R}\,\mathbf{a}\|_2^2 + \|\widehat{\mathbf{Q}}^T\mathbf{y}\|_2^2 \geq \|\widehat{\mathbf{Q}}^T\mathbf{y}\|_2^2
\end{aligned}$$

So the minimum value of $\|\mathbf{y} - \mathbf{Z}\,\mathbf{a}\|_2^2$ is $\|\widehat{\mathbf{Q}}^T\mathbf{y}\|_2^2$, and this minimum is realized when $\|\mathbf{Q}^T\mathbf{y} - \mathbf{R}\,\mathbf{a}\|_2^2 = 0$, i.e. when $\mathbf{Q}^T\mathbf{y} - \mathbf{R}\,\mathbf{a} = \mathbf{0}$. □

5 Calculating the QR-factorization

This section gives a rough sketch on how to calculate the QR-factorization in a way that is numerically stable.

The Householder reflection matrix

$$\mathbf{H} = \mathbf{I} - 2\frac{\mathbf{v}\,\mathbf{v}^T}{\mathbf{v}^T\mathbf{v}}, \qquad \mathbf{H} \in \mathbb{R}^{n\times n}$$

is orthogonal ($\mathbf{H}^T\mathbf{H} = \mathbf{H}\,\mathbf{H}^T = \mathbf{I}$), and symmetric ($\mathbf{H}^T = \mathbf{H}$) for any vector $\mathbf{v} \in \mathbb{R}^n$. Now given the matrix $\mathbf{Z} \in \mathbb{R}^{n\times p}$ with first column $\mathbf{z}_1$ and by choosing

$$\mathbf{v} = \begin{bmatrix} z_{11} \pm \|\mathbf{z}_1\|_2 \\ z_{21} \\ z_{31} \\ \vdots \\ z_{n1} \end{bmatrix}$$

the Householder reflection $\mathbf{H}$, applied on $\mathbf{Z}$ will zero out the first column, except the first entry. The first column of $\mathbf{H}\,\mathbf{Z}$ will be

$$\begin{bmatrix} \mp\|\mathbf{z}_1\|_2 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Just as in Gauss elimination, we have translated everything below the diagonal to zero. We continue in the same way with the second, third etc column, but regard only the entries on the diagonal and below when we produce $\mathbf{v}$.

By applying p such Householder reflections to $\mathbf{Z}$ we can transform it to an upper triangular form

$$\mathbf{H}_p \cdots \mathbf{H}_3\mathbf{H}_2\mathbf{H}_1\mathbf{Z} = \mathbf{R}^*$$

If we collect the $\mathbf{H}_k$'s on the other side of the equality, we get the QR-factorization

$$\mathbf{Z} = \underbrace{\mathbf{H}_1\mathbf{H}_2\mathbf{H}_3 \cdots \mathbf{H}_p}_{\mathbf{Q}^*} \mathbf{R}^*$$

Picking the first p columns in $\mathbf{Q}^*$ as $\mathbf{Q}$, and the first p rows in $\mathbf{R}^*$ as $\mathbf{R}$ gives the QR-factorization we use for least-squares problems.

Below is a prototype QR-factorization routine for Matlab, where the description above is translated into code:

```
function [Q,R] = QRslow(A, zero)

%QRslow A perfect way to do QR-factorization, except that
%   the method is far from optimally implemented.
%   The calling sequence
%       [Q,R] = QRslow(A)
%   returns an orthogonal matrix Q and an upper triangular
%   matrix R of the same dimensions as A, such that A = Q*R.
%   The calling sequence
%       [Q,R] = QRslow(A,0)
%   returns the economy size QR-factorization, where Q has
%   orthonormal columns and the same dimensions as A, and R
%   is a square upper triangular matrix, such that A = Q*R.

if nargin == 2 && zero ~= 0
    error('Use QRslow(A,0) for compact form');
end

[n,p] = size(A);

Q = eye(n);
for i = 1:p
    v = zeros(n,1);
    v(i:n) = A(i:n,i);
    v(i) = v(i) + sign(v(i))*norm(v);
    H = eye(n) - 2*(v*v')/(v'*v);
    A = H*A;
    Q = Q*H;
end

if nargin == 2
    Q = Q(:,1:p);
    R = triu(A(1:p,:));
else
    R = triu(A);
end
```

6 SVD and fitting

This is some notes on how to use the singular value decomposition (SVD) for solving some fitting problems. The author of these notes is Inge Söderkvist.

6.1 The Singular Value Decomposition

The singular value decomposition of a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ is

$$\mathbf{A} = \mathbf{U}\,\mathbf{S}\,\mathbf{V}^T,$$

where $\mathbf{U} \in \mathbb{R}^{m \times m}$ and $\mathbf{V} \in \mathbb{R}^{n \times n}$ are orthogonal matrices, and $\mathbf{S} \in \mathbb{R}^{m \times n}$ is a diagonal matrix containing the singular values $\sigma_1 \geq \sigma_2 \geq \cdots, \geq \sigma_r \geq 0,\ r = \min(m, n)$.

6.2 The Rigid Body Movement Problem

Assume that we have n landmarks in a rigid body and let $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ be the 3-D positions of these landmarks before the movement and $\{\mathbf{y}_1, \ldots, \mathbf{y}_n\}$ be the positions after the movement. We want to determine a rotation matrix $\mathbf{R}$ and a translation vector $\mathbf{d}$ that map the points $\mathbf{x}_i$ to the points $\mathbf{y}_i, i = 1, \ldots, n$. Because of measurement errors the mapping is not exact and we use the following least-squares problem

$$\min_{\mathbf{R} \in \Omega, \mathbf{d}} \sum_{i=1}^{n} \|\mathbf{R}\,\mathbf{x}_i + \mathbf{d} - \mathbf{y}_i\|_2^2, \tag{2}$$

where

$$\Omega = \{\mathbf{R} \,|\, \mathbf{R}^T\mathbf{R} = \mathbf{R}\,\mathbf{R}^T = \mathbf{I}_3; \det(\mathbf{R}) = 1\},$$

is the set of orthogonal rotation matrices. Problem (2) is linear with respect to the vector $\mathbf{d}$ but it is non-linear with respect to the rotation matrix $\mathbf{R}$ (that is because of the orthogonality condition of $\mathbf{R}$). Introducing $\overline{\mathbf{x}} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i, \quad \overline{\mathbf{y}} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{y}_i$, and the matrices

$$\mathbf{A} = \left[\mathbf{x}_1 - \overline{\mathbf{x}}, \ldots, \mathbf{x}_n - \overline{\mathbf{x}}\right], \quad \mathbf{B} = \left[\mathbf{y}_1 - \overline{\mathbf{y}}, \ldots, \mathbf{y}_n - \overline{\mathbf{y}}\right],$$

the problem of determining the rotation matrix becomes

$$\min_{\mathbf{R} \in \Omega} \|\mathbf{R}\,\mathbf{A} - \mathbf{B}\|_F, \tag{3}$$

where the Frobenius norm of a matrix $\mathbf{Z}$ is defined as $\|\mathbf{Z}\|_F^2 = \sum_{i,j} z_{i,j}^2$. In e.g. (Arun, Huang & Blostein 1987, Hanson & Norris 1981) it is shown that this *Orthogonal Procrustes problem* can be solved using the singular value decomposition of the matrix $\mathbf{C} = \mathbf{B}\,\mathbf{A}^T$. An algorithm is presented below.

Algorithm 1 *Algorithm for solving problem (2) by using the SVD-method.*

```
1. $\bar{\mathbf{x}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i, \quad \bar{\mathbf{y}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{y}_i$
2. $\mathbf{A} = [\mathbf{x}_1 - \bar{\mathbf{x}}, \dots, \mathbf{x}_n - \bar{\mathbf{x}}], \quad \mathbf{B} = [\mathbf{y}_1 - \bar{\mathbf{y}}, \dots, \mathbf{y}_n - \bar{\mathbf{y}}]$
3. $\mathbf{C} = \mathbf{B}\,\mathbf{A}^T$
4. $\mathbf{U}\,\mathbf{S}\,\mathbf{V}^T = \mathbf{C}$ ( Computation of the singular decomposition of $\mathbf{C}$)
5. $\mathbf{R} = \mathbf{U}\,\mathrm{diag}(1, 1, \det(\mathbf{U}\,\mathbf{V}^T))\,\mathbf{V}^T$
6. $\mathbf{d} = \bar{\mathbf{y}} - \mathbf{R}\,\bar{\mathbf{x}}$
```

The solution to problem (3) is unique if $\sigma_2(\mathbf{C})$ is nonzero. (More strictly, the solution is not unique if $\sigma_2(\mathbf{C})$ equals zero or if $\sigma_2(\mathbf{C}) = \sigma_3(\mathbf{C})$ and $\det(\mathbf{U}\,\mathbf{V}^T) = -1$. But the latter situation implies that a reflection is the best orthogonal matrix describing the movement. This situation will seldom occur when studying rigid bodies and indicates that something is wrong.)

Algorithm 1 can be used also in 2-D settings when a movement in the plane is to be determined.

More about the rigid body movement problem and its sensitivity with respect to perturbations in the positions of the landmarks can be found in (Söderkvist 1993, Söderkvist & Wedin 1993, Söderkvist & Wedin 1994).

6.3 Fitting Planes and Lines by Orthogonal Distance Regression

Assume that we want to find the plane that are as close as possible to a set of n 3-D points $(\mathbf{p}_1, \dots, \mathbf{p}_n)$ and that the closeness is measured by the square sum of the orthogonal distances between the plane and the points.

Let the position of the plane be represented by a point $\mathbf{c}$ belonging to the plane and let the unit vector $\mathbf{n}$ be the normal to the plane determining its direction. The orthogonal distance between a point $\mathbf{p}_i$ and the plane is then $(\mathbf{p}_i - \mathbf{c})^T\mathbf{n}$. Thus the plane can be found by solving

$$\min_{\mathbf{c}, \|\mathbf{n}\|=1} \sum_{i=1}^{n} ((\mathbf{p}_i - \mathbf{c})^T\mathbf{n})^2. \tag{4}$$

Solving this for $\mathbf{c}$ gives $\mathbf{c} = 1/n\sum_{i=1}^{n}\mathbf{p}_i$, see e.g (Gander & Hrebicek 1993). Introducing the $3 \times n$ matrix

$$\mathbf{A} = [\mathbf{p}_1 - \mathbf{c}, \mathbf{p}_2 - \mathbf{c}, \dots, \mathbf{p}_n - \mathbf{c}]$$

problem (4) can be formulated as

$$\min_{\|\mathbf{n}\|=1} \|\mathbf{A}^T\mathbf{n}\|_2^2. \tag{5}$$

Using the singular decomposition $\mathbf{A} = \mathbf{U}\,\mathbf{S}\,\mathbf{V}^T$, of $\mathbf{A}$, we have

$$\|\mathbf{A}^T\mathbf{n}\|_2^2 = \|\mathbf{V}\,\mathbf{S}^T\mathbf{U}^T\mathbf{n}\|_2^2 = \|\mathbf{S}^T\mathbf{U}^T\mathbf{n}\|_2^2 = (\sigma_1 y_1)^2 + (\sigma_2 y_2)^2 + (\sigma_3 y_3)^2,$$

where $\mathbf{y}$ is the unit vector $\mathbf{y} = \mathbf{U}^T\mathbf{n}$. Thus, $\|\mathbf{A}^T\mathbf{n}\|_2^2$ is minimized for $\mathbf{y} = (0, 0, 1)^T$ or equivalently, for $\mathbf{n} = \mathbf{U}(:, 3)$. The minimal value of $\sum_{i=1}^{n}((\mathbf{p}_i - \mathbf{c})^T\mathbf{n})^2$ is σ_3^2. To summarize we present the following algorithm for fitting the plane.

Algorithm 2 *Algorithm for fitting a plane using orthogonal distance regression and singular value decomposition.*

```
1. $\mathbf{c} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{p}_i$
2. $\mathbf{A} = [\mathbf{p}_1 - \mathbf{c}, \ldots, \mathbf{p}_n - \mathbf{c}]$
3. $\mathbf{U}\,\mathbf{S}\,\mathbf{V}^T = \mathbf{A}$ ( Computation of the singular decomposition of $\mathbf{A}$)
4. $\mathbf{n} = \mathbf{U}(:, 3)$, i.e., the normal is given as the third column of $\mathbf{U}$.
```

Since the normal is given by $\mathbf{U}(:, 3)$ it follows from the orthogonality of $\mathbf{U}$ that the plane is spanned by the two first columns of $\mathbf{U}$.

Similarly, we can obtain the best fitted line as the first column of $\mathbf{U}$. The square sum of distances between the “best” plane and the points are given by σ_3^3 and the square sum of distances between the “best” line and the points is given by $\sigma_2^2 + \sigma_3^2$.

A similar technique can be used also for fitting a line in 2-D.

References

Arun, K. S., Huang, T. S. & Blostein, S. D. (1987), `Least-squares fitting of two 3-d point sets', *IEEE Trans. Patt. Anal. Machine Intell.* **9**(5), 698--700.

Chapra, S. C. (2012), *Applied Numerical Methods with MATLAB for Engineers and Scientists*, third edn, McGraw-Hill.

Demmel, J. W. (1997), *Applied Numerical Linear Algebra*, SIAM.

Gander, W. & Hrebicek, J. (1993), *Solving Problems in Scientific Computing using Matlab and Maple*, Springer Verlag.

Hanson, R.J. & Norris, M.J. (1981), `Analysis of measurements based on the singular value decomposition', *SIAM J. Sci. Stat. Comput.* **2**, 363--373.

Higham, N. J. (1996), *Accuracy and Stability of Numerical Algorithms*, SIAM.

Lay, D. C. (2003), *Linear Algebra and its Applications*, third edn, Addison-Wesley.

Söderkvist, I. (1993), `Perturbation analysis of the orthogonal procrustes problem', *BIT.* **33**, 687--694.

Söderkvist, I. & Wedin, P.-Å. (1993), `Determining the movements of the skeleton using well-configured markers', *Journal of Biomechanics.* **26**(12), 1473--1477.

Söderkvist, I. & Wedin, P.-Å. (1994), `On condition numbers and algorithms for determining a rigid body movement', *BIT.* **34**, 424--436.