Time-Variant and Quasi-separable Systems*
supplementary reading
.
- Matrix Decompositions -

Klaus Diepold, Patrick Dewilde

Spring 2025

1 Introduction

In this chapter we will cover two matrix factorization, which are of central importance for our topic. We will discuss the QR Decomposition (QRD) and the Singular Value Decomposition (SVD). Of course, there exists a large number of various factorizations such as LU-factorization, Polar Decomposition, Cholesky-Factorization, Eigenvalue Decomposition, to name the most prominent examples. However, we will not dive into the general topic of matrix factorizations.

2 QR Decomposition

2.1 Basic Factorization Properties

QR decomposition and its several variants (QL, RQ, LQ) is one of the most effective, most common and most useful techniques of matrix calculus. It has many purposes, which will become clear in this and the remaining chapters. Given a matrix T, it consists in finding a factorization $T = Q \cdot \tilde{R}$, in which Q is unitary (hence square with $Q'Q = QQ' = 1$), the entries of R are concentrated in the upper right corner, and so that

$$T = \underbrace{\left[\begin{array}{c|c} Q_1 & Q_2 \end{array}\right]}_{Q} \cdot \underbrace{\left[\begin{array}{c} R \\ \hline 0 \end{array}\right]}_{\tilde{R}} = Q_1 \cdot R,$$

with the result that the columns of Q_1 span an orthonormal basis for the range of T, Q_2 provides for an orthonormal basis of the co-kernel of T (i.e., the kernel of T^T) and the rows of R provide a basis for the co-range of T. In other words, the factorization characterizes T's global 'geometry'. We have already

*P. Dewilde, K. Diepold, A.-J. v.d. Veen. Time-Variant ans Quasi-separable Systems, Cambridge University Press, 2024

seen the importance of such factorizations in realization theory, in this and the next lecture we explore the numerical methods to compute it efficiently, starting with T just a matrix, and then later moving to allow T to have a state space realization.

2.2 Using QR Decomposition to Find Least Squares Solutions

Besides the determination of ranges and kernels mentioned above, QR can be used effectively to produce minimal least squares solutions for overdetermined systems of equations.

Let us assume that we are given an $m \times n$ matrix T and an m-vector y. We intend to determine an n-vector u such that

$$Tu = y.$$

Distinguish the following cases

$$\begin{cases} m > n & \text{overdetermined} \\ n = m & \text{square} \\ m < n & \text{underdetermined.} \end{cases}$$

For ease of discussion we shall look only at the case $m \geq n$, that is, we want to compute the least squares solution for the overdetermined system of equations, and let us assume, in addition, that the n columns of T are linearly independent. In that case, R will be a square, non-singular matrix, and the system of equations becomes

$$\left[\begin{array}{c|c} Q_1 & Q_2 \end{array}\right] \cdot \left[\begin{array}{c} R \\ \hline 0 \end{array}\right] u = b$$

Premultiplying with Q' and putting $\begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix} := \begin{bmatrix} Q_1^T \\ Q_2^T \end{bmatrix} b$ reduces the system to the equivalent system

$$\begin{bmatrix} Ru \\ 0 \end{bmatrix} = \begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix},$$

which will only have an exact solution iff $\beta_2 = 0$ (i.e. if b belongs to the range of T), in which case $u = R^{-1}\beta_1$. More generally, for any u, the error e made by using u as a solution will be

$$\begin{bmatrix} e_1 \\ e_2 \end{bmatrix} = \begin{bmatrix} \beta_1 - Ru \\ \beta_2 \end{bmatrix}$$

which has norm squared $\|e\|^2 = \|\beta_1 - Ru\|^2 + \|\beta_2\|^2$, which will evidently be minimal when $Ru = \beta_1$, and will then be equal to the minimal possible quadratic error β_2. Hence $u = R^{-1}Q_1^T b$ solves the quadratic minimization problem for the present case of overdetermined data.

3 QR Decomposition using Householder Reflections

3.1 Householder Reflections

We consider the Householder matrix H, which is defined as

$$H = 1 - 2P_u, \quad P_u = u(u'u)^{-1}u',$$

i.e. the matrix P_u is a projection matrix ($P_u^2 = P_u, P_u' = P_u$), which projects any vector x onto the direction of the vector $u = x - y$. The direction spanned by the vector u is sometimes denoted by $path\,(H)$, whereas the direction spanned by the vector $v = x + y$ is denoted by $fix\,(H)$ because we can see that $Hv = v$.

The Householder matrix H satisfies the orthogonality property $H'H = 1$ and has the property that $\det H = -1$, which indicates that H is a reflection which reverses orientation, i.e. we have $Hu = -u$.

We search for an orthogonal transformation H which transforms a given vector x onto a given vector $y = Hx$, where $\|x\| = \|y\|$ holds. We compute the vector u specifying the projection direction for P_u as $u = y - x$. The construction idea behind the Householder transformation as a reflection is depicted as geometric construction in Figure 1.

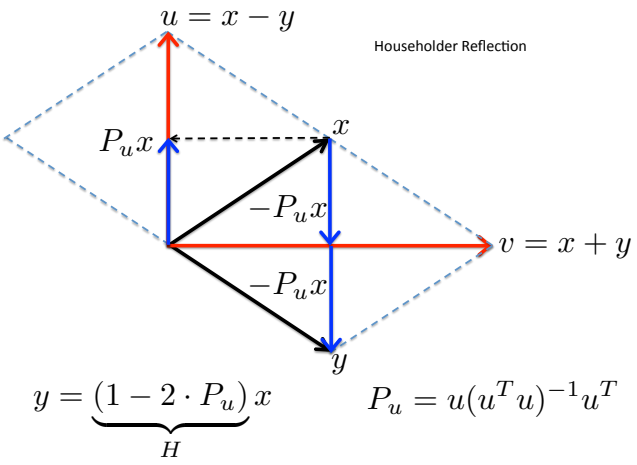

Figure 1: Geometric visualization for the construction of a Householder reflection.

Note that $\det H = -1$ holds as the transformation H is a reflection, which flips orientation.

3.2 Generalized Rotations

Although Householder reflections are very handy for computing the QR decomposition (that's what Matlab does), it is often useful to dispose of a one-shot generalized rotation that rotates a column vector

$$u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}, \quad u_1 \in \mathcal{R}, u_2 \in \mathcal{R}^n$$

of dimension $n + 1$, with

$$\|u\|^2 = |u_1|^2 + \sum_{i=1}^{n} |u_{2,i}|^2 = 1$$

and with non-negative real first element u_1 to the positive first axis, which we call e_1 generically (ignoring its dimension)—e_1 is then a vector of dimension $n + 1$ here.

The following unitary matrix pulls the trick (notice that u_1 may very well be equal to zero)

$$Q_u := \begin{bmatrix} u_1 & u_2' \\ -u_2 & 1 - u_2 \frac{1}{1+u_1} u_2' \end{bmatrix}, \quad Q_u \in \mathcal{SO}(n+1) \tag{1}$$

and we shall have $Q_u u = e_1$, which is easily verified directly. The matrix Q_u is an orthogonal matrix with $\det Q_u = 1$, it is a generalized rotation matrix. One can show that it can be produced by a sequence of elementary Givens rotations, but a direct application of such a matrix on an arbitrary vector

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix},$$

with x_1 scalar, produces the following 'efficient' computation

$$Q_u x = \begin{bmatrix} u_1 x_1 + u_2' x_2 \\ x_2 - u_2 (x_1 + \frac{u_2' x_2}{1+u_1}) \end{bmatrix}, \tag{2}$$

in which the inner product $u_2' x_2$ should be executed only once.

The generalized rotation provides the same basic functionality than the Householder reflection, i.e. it rotates a given vector in the direction of the first unit direction e_1. If we need a general orthogonal mapping between two vectors x and y with $Q_{xy} \cdot x = y$. where both vectors have the same Euclidean length $\|x\| = \|y\|$, then we need to compose two rotations Q_x and Q_y with

$$Q_x \cdot x = e_1, \quad Q_y \cdot y = e_1, \quad \Rightarrow \quad \underbrace{Q_y' \cdot Q_x}_{Q_{xy}} \cdot x = y$$

The main feature of the generalized rotations is that we have $\det Q = +1$, whereas we have $\det H = -1$ for Householder reflections.

3.3 Elimination Scheme Based on Householder Reflections

A cascade of unitary transformations is still a unitary transformation. Let H_i be a transformation on the *rows* i until and n which annihilates all appropriate elements in rows $i+1 : n$ (as indicated further in the example)

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad y = \begin{bmatrix} \sqrt{x'x} \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

Successive eliminations on a 6×4 matrix (in which the $\cdot$ indicates a relevant element of the matrix)

$$\begin{bmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{bmatrix} (H_{16}) \mapsto \begin{bmatrix} \star & \star & \star & \star \\ 0 & \cdot & \cdot & \cdot \\ 0 & \cdot & \cdot & \cdot \\ 0 & \cdot & \cdot & \cdot \\ 0 & \cdot & \cdot & \cdot \\ 0 & \cdot & \cdot & \cdot \end{bmatrix} (H_{26}) \mapsto \begin{bmatrix} \star & \star & \star & \star \\ 0 & \star & \star & \star \\ 0 & 0 & \cdot & \cdot \\ 0 & 0 & \cdot & \cdot \\ 0 & 0 & \cdot & \cdot \\ 0 & 0 & \cdot & \cdot \end{bmatrix} (H_{36}) \mapsto \begin{bmatrix} \star & \star & \star & \star \\ 0 & \star & \star & \star \\ 0 & 0 & \star & \star \\ 0 & 0 & 0 & \cdot \\ 0 & 0 & 0 & \cdot \\ 0 & 0 & 0 & \cdot \end{bmatrix} \mapsto \begin{bmatrix} \star & \star & \star & \star \\ 0 & \star & \star & \star \\ 0 & 0 & \star & \star \\ 0 & 0 & 0 & \cdot \\ 0 & 0 & 0 & \cdot \\ 0 & 0 & 0 & \cdot \end{bmatrix}$$

$$(H_{46}) \mapsto \begin{bmatrix} \star & \star & \star & \star \\ 0 & \star & \star & \star \\ 0 & 0 & \star & \star \\ 0 & 0 & 0 & \star \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \mapsto \left[\begin{array}{cccc} \star & \star & \star & \star \\ 0 & \star & \star & \star \\ 0 & 0 & \star & \star \\ 0 & 0 & 0 & \star \\ \hline 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right] = \left[\begin{array}{c} R \\ \hline 0 \end{array}\right]$$

The elements that have changed during the last transformation are denoted by a '$\star$'. There are no fill-ins, a purposeful zero does not get modified later on. The end result is

$$\underbrace{H_{46}H_{36}H_{26}H_{16}}_{Q^T} T = \left[\begin{array}{c} R \\ \hline 0 \end{array}\right]$$

in which R is upper triangular.

The elimination scheme looks exactly the same if we were to use Generalized rotations instead of Householder reflections. The only difference is that the final orthogonal transformation Q will have $\det Q = +1$, when rotations are used.

4 QR Decomposition Using Givens Rotations

4.1 The Givens Elementary Elimination Strategy

The general strategy for solution is an orthogonal transformations on rows. Let a and b be two rows in a matrix, then we can generate linear combinations of these rows by applying a transformation matrix to the left according to

$$\begin{bmatrix} t_{11} & t_{12} \\ t_{21} & t_{22} \end{bmatrix} \begin{bmatrix} - & a & - \\ - & b & - \end{bmatrix} = \begin{bmatrix} t_{11}a + t_{12}b \\ t_{21}a + t_{22}b \end{bmatrix}.$$

We can represent this transformation equivalently by embedding the 2×2 transformation matrix into a $m \times m$-matrix according to

$$\begin{bmatrix} 1 & & & & & \\ & 1 & & & & \\ & & t_{11} & - & t_{12} & \\ & & \vert & 1 & \vert & \\ & & t_{21} & - & t_{22} & \\ & & & & & 1 \end{bmatrix} \begin{bmatrix} - & \cdot & - \\ - & \cdot & - \\ - & a & - \\ - & \cdot & - \\ - & b & - \\ - & \cdot & - \end{bmatrix} = \begin{bmatrix} - & \cdot & - \\ - & \cdot & - \\ t_{11}a + t_{12}b & & \\ - & \cdot & - \\ t_{21}a + t_{22}b & & \\ - & \cdot & - \end{bmatrix}.$$

It is the goal of such a row transformation to eliminate one pre-specified entry in the transformed rows. One thought goes into choosing the type for the elementary transformation and then we need to determine the parameter values of such a transformation to achieve the transformation goal.

4.2 Givens (Jacobi) Rotation

One particular choice for the elementary transformation is the Givens transformation, which is also often referred to by the name Jacobi transformation. The Givens elementary transformation is given as a special

form for the elementary 2×2-matrix

$$\begin{bmatrix} t_{11} & t_{12} \\ t_{21} & t_{22} \end{bmatrix} = \underbrace{\begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix}}_{R(\phi)}.$$

The Givens transformation $R(\phi)$ represents an elementary rotation over an angle ϕ. We can choose the value for this parameter ϕ such that the resulting Jacobi transformation R annihilates a predetermined element in a row (with $c \doteq \cos\phi$ and $s \doteq \sin\phi$), that is we get

$$\begin{bmatrix} c & -s \\ s & c \end{bmatrix} \begin{bmatrix} a_1 & a_2 & \cdots & a_m \\ b_1 & b_2 & \cdots & b_m \end{bmatrix} = \begin{bmatrix} ca_1 - sb_1 & ca_2 - sb_2 & \cdots \\ sa_1 + cb_1 & sa_2 + cb_2 & \cdots \end{bmatrix}$$

$$\doteq \begin{bmatrix} \sqrt{|a_1|^2 + |b_1|^2} & \star & \cdots & \star \\ 0 & \star & \cdots & \star \end{bmatrix}$$

when $sa_1 + cb_1 = 0$ or

$$\tan\phi = -\frac{b_1}{a_1}$$

which can always be done.

Note that the Jacobi transformation has the interpretation of a rotation in one plane, which corresponds with the observation that $\det R = 1$.

4.3 Elimination Scheme Based on Givens Transformations

A cascade of unitary transformations is still a unitary transformation: let Q_{ij} be a transformation on *rows* i and j which annihilates an appropriate element (as indicated further in the example). Successive eliminations on a 4×3 matrix (in which the $\cdot$ indicates a relevant element of the matrix):

$$\left[\begin{array}{ccc} \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \hline \cdot & \cdot & \cdot \end{array}\right] (Q_{14}) \mapsto \left[\begin{array}{ccc} \star & \star & \star \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \hline 0 & \star & \star \end{array}\right] (Q_{13}) \mapsto \left[\begin{array}{ccc} \star & \star & \star \\ \cdot & \cdot & \cdot \\ 0 & \star & \star \\ \hline 0 & \star & \star \end{array}\right] (Q_{12}) \mapsto$$

$$\mapsto \left[\begin{array}{ccc} \star & \star & \star \\ 0 & \star & \star \\ 0 & \star & \star \\ \hline 0 & \star & \star \end{array}\right] (Q_{24}) \mapsto \left[\begin{array}{ccc} \star & \star & \star \\ 0 & \star & \star \\ 0 & \star & \star \\ \hline 0 & 0 & \star \end{array}\right] (Q_{23}) \mapsto \left[\begin{array}{ccc} \star & \star & \star \\ 0 & \star & \star \\ 0 & 0 & \star \\ \hline 0 & 0 & \star \end{array}\right] \mapsto (Q_{34}) \mapsto \left[\begin{array}{ccc} \star & \star & \star \\ 0 & \star & \star \\ 0 & 0 & \star \\ \hline 0 & 0 & 0 \end{array}\right] = \left[\begin{array}{c} R \\ \hline 0 \end{array}\right]$$

(The elements that have changed during the last transformation are denoted by a '$\star$'. There are no fill-ins, a purposeful zero does not get modified lateron.) The end result is:

$$\underbrace{Q_{34}Q_{23}Q_{24}Q_{12}Q_{13}Q_{14}}_{Q^T} T = \begin{bmatrix} R \\ \hline 0 \end{bmatrix}$$

in which R is upper triangular.

There exist a certain degree of freedom concerning the sequencing of elimination steps. The previous schematic example displayed a column-wise elimination strategy. Equivalently, a other elimination strategies are possible.

4.4 Parallel Processing for Elimination Scheme

Successive elimination of matrix entries in the lower triangular part of a matrix. Some elimination steps are independent from other steps and allow for a scheduling of the elimination for parallel processing. See how the 0 can be created in the schematic example below

$$\begin{bmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{bmatrix} \mapsto \begin{bmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ 0 & \cdot & \cdot & \cdot \end{bmatrix} \mapsto \begin{bmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ 0 & \cdot & \cdot & \cdot \\ 0 & \cdot & \cdot & \cdot \end{bmatrix} \mapsto \begin{bmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ 0 & \cdot & \cdot & \cdot \\ 0 & \cdot & \cdot & \cdot \\ 0 & 0 & \cdot & \cdot \end{bmatrix} \mapsto \begin{bmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ 0 & \cdot & \cdot & \cdot \\ 0 & \cdot & \cdot & \cdot \\ 0 & 0 & \cdot & \cdot \\ 0 & 0 & \cdot & \cdot \end{bmatrix}$$

$$\begin{bmatrix} \cdot & \cdot & \cdot & \cdot \\ 0 & \cdot & \cdot & \cdot \\ 0 & \cdot & \cdot & \cdot \\ 0 & 0 & \cdot & \cdot \\ 0 & 0 & \cdot & \cdot \\ 0 & 0 & 0 & \cdot \end{bmatrix} \mapsto \begin{bmatrix} \cdot & \cdot & \cdot & \cdot \\ 0 & \cdot & \cdot & \cdot \\ 0 & 0 & \cdot & \cdot \\ 0 & 0 & \cdot & \cdot \\ 0 & 0 & 0 & \cdot \\ 0 & 0 & 0 & \cdot \end{bmatrix} \mapsto \begin{bmatrix} \cdot & \cdot & \cdot & \cdot \\ 0 & \cdot & \cdot & \cdot \\ 0 & 0 & \cdot & \cdot \\ 0 & 0 & 0 & \cdot \\ 0 & 0 & 0 & \cdot \\ 0 & 0 & 0 & 0 \end{bmatrix} \mapsto \begin{bmatrix} \cdot & \cdot & \cdot & \cdot \\ 0 & \cdot & \cdot & \cdot \\ 0 & 0 & \cdot & \cdot \\ 0 & 0 & 0 & \cdot \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Analyzing this elimination pattern gives us an opportunity to sequence the elimination steps systematically according to the following pattern to exploit a maximum of data parallelism

$$\begin{bmatrix} \cdot & \cdot & \cdot & \cdot \\ 5 & \cdot & \cdot & \cdot \\ 4 & 6 & \cdot & \cdot \\ 3 & 5 & 7 & \cdot \\ 2 & 4 & 6 & 8 \\ 1 & 3 & 5 & 7 \end{bmatrix}.$$

4.5 Uniqueness of QR Decomposition

In spite of all the various approaches to compute a QR decomposition, i.e. using Householder reflections or different sequences of Givens rotations, the result turns out the same. Under regular conditions of having a matrix A with full column rank we can show that the factorization is unique within a signature matrix. To this end we look at a factorization for such a matrix in terms of

$$A = Q \cdot R.$$

We want to check if there exist alternative matrices $\widehat{Q}$ and $\widehat{R}$ to achieve this decomposition for A. To this end, we squeeze the identity between the original factors Q and R. Then we re-group the factors to have an expression for these alternative factors. This way we have

$$A = Q \cdot M \cdot M^{-1} R = \underbrace{(Q \cdot M)}_{\widehat{Q}} \cdot \underbrace{(M^{-1} \cdot R)}_{\widehat{R}}.$$

In order for the factor $\widehat{Q} = Q \cdot M$ to be again an orthogonal matrix, the matrix M needs to be an orthogonal matrix. In order for the factor $\widehat{R} = M^{-1} \cdot R$ to be upper triangular, the matrix M needs to be upper triangular. So, the matrix M is required to e orthogonal and upper triangular at the same time.

The upper triangular matrix M needs to satisfy $M'M = 1$, which implies that the columns are scaled to unit length and are mutually orthogonal. We can look at

$$M = \begin{bmatrix} m_{11} & m_{12} & \cdots & m_{1n} \\ & m_{22} & \cdots & m_{2n} \\ & & \ddots & \vdots \\ & & & m_{nn} \end{bmatrix} \quad \longrightarrow \quad \begin{array}{l} m_{11} = \pm 1 \\ m_{11}m_{12} = 0 \longrightarrow m_{12} = 0 \\ m_{22} = \pm 1 \\ \vdots \\ m_{nn} = \pm 1 \end{array}$$

and conclude that M can only be

$$M = \begin{bmatrix} \pm 1 & & & \\ & \pm 1 & & \\ & & \ddots & \\ & & & \pm 1 \end{bmatrix}.$$

This means in essence that all the algorithmic variations to compute the QR decomposition arrive at the same result. This is a reassuring result, even if it keeps me surprised observing the very different elimination strategies.

5 Echelon Form

Suppose now that one disposes of a collection of vectors $A = \begin{bmatrix} a_1 & \cdots & a_m \end{bmatrix}$ of dimension $1 + n$ (say columns of a matrix), and suppose that we are entitled to apply rotations to them (i.e., generalized rotation matrix of dimension $n + 1$ to the left of A). Suppose a_k is the non-zero vector with the smallest k and that its first element is $a_{k,1} = |a_{k,1}|e^{j\phi_1}$ (the latter may very well be equal to zero). Applying Q_{a_k} to the stack now produces the following typical form:

$$\begin{aligned} Q_{a_k}A &= Q_{a_k}\begin{bmatrix} a_1 & \cdots & a_{k-1} & a_k & a_{k+1} & \cdots & a_m \end{bmatrix} && (3) \\ &= \begin{bmatrix} 0 & \cdots & 0 & e_1\|a_k\|e^{j\phi_1} & Q_{a_k}a_{k+1} & \cdots & Q_{a_k}a_m \end{bmatrix} && (4) \\ &= \begin{bmatrix} 0 & \cdots & 0 & \|a_k\|e^{j\phi_1} & * & \cdots & * \\ 0 & \cdots & 0 & 0 & b_{k+1} & \cdots & b_m \end{bmatrix} && (5) \end{aligned}$$

where the "$*$" indicate entries that have been modified (and will remain unchanged later one), and the $\begin{bmatrix} b_{k+1} & \cdots & b_m \end{bmatrix}$ is a new collection of vectors, now of dimension n, and on which the procedure can be repeated without producing new fill-ins in the zero elements obtained so far, now with one dimension less (some of the zeros shown above may disappear, e.g., when a_1 is already non-zero.). Continuing this way, now on the b's and realizing that products of rotation matrices remain orthogonal or unitary, after a number of steps one obtains a so called *echelon* form

$$A = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix}\begin{bmatrix} R_1 \\ 0 \end{bmatrix} \tag{6}$$

in which

$$R_1 = \begin{bmatrix} 0 & \cdots & 0 & R_{1,k_1} & \cdots & * & * & \cdots & * & * & \cdots \\ 0 & \cdots & 0 & 0 & \cdots & 0 & R_{2,k_2} & \cdots & * & * & \cdots \\ & & & & \vdots & & & & & & \\ 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & R_{\delta,k_\delta} & \cdots \end{bmatrix}, \tag{7}$$

δ is the rank of A, Q is orthogonal or unitary. One sees easily that the columns of Q_1 form a basis for the range of A, while the columns of R_1^T form a basis for the co-range (i.e., the range of A^T), and the columns of Q_2 for the co-kernel.

A similar, even more powerful result could have been obtained by SVD (Singular Value Decomposition) of A as

$$A = \begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} \Sigma & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} V_1^T \\ V_2^T \end{bmatrix} = \begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} \Sigma V_1^T \\ 0 \end{bmatrix} \tag{8}$$

at the cost of more computations. Here

$$\Sigma = \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_\delta \end{bmatrix}$$

are the singular values of A in order: $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_\delta > 0$, the columns of U_1 form a basis for the range of A while the columns of V_1 and $V_1\Sigma$ for the co-range.

The example in the next section (and many in subsequent chapters) actually uses a variant of the QR-algorithm just presented, namely an algorithm that starts at the bottom right corner and produces an RQ factorization, with R again an echelon matrix and Q an orthogonal or unitary matrix. The procedure now starts out with a collection of rows rather than columns, it is dual to the preceding

$$A = \begin{bmatrix} 0 & R_2 \end{bmatrix} \cdot \begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix} \tag{9}$$

R_2 is obtained by compressing towards the last column starting with the bottom row (skipping it when zero), it will look like

$$\begin{bmatrix} \vdots & & \vdots & \vdots \\ R_{\delta,k_\delta} & & * & * \\ 0 & & * & * \\ \vdots & & \vdots & \vdots \\ 0 & & R_{2,k_2} & * \\ 0 & & 0 & * \\ \vdots & \cdots & \vdots & \vdots \\ 0 & & 0 & R_{1,k_1} \\ 0 & & 0 & 0 \\ \vdots & & \vdots & \vdots \\ 0 & & 0 & 0 \end{bmatrix} \tag{10}$$

these columns now forming a base for the range of A. Also in this case, a more accurate result can be obtained through SVD, when needed.

6 The Family of QR Decompositions

6.1 Alternative Schemes

Besides the standard set up for the QR decomposition we can easily conceive alternative schemes by changing between left and right multiplication of T with the orthogonal factor Q and by choosing between

L and R according to

- $T = QR$

$$\longrightarrow \begin{bmatrix} q & q & q & q \\ q & q & q & q \\ q & q & q & q \\ q & q & q & q \end{bmatrix} \cdot \begin{bmatrix} r & r & r & r \\ & r & r & r \\ & & r & r \\ & & & r \end{bmatrix}$$

  compress columns upwards (bottom to top) going from from right to left, using orthogonal multiplications from the left side (pre-multitplication), start elimination with first column progressing from left to right.

- $T = QL$

$$\longrightarrow \begin{bmatrix} q & q & q & q \\ q & q & q & q \\ q & q & q & q \\ q & q & q & q \end{bmatrix} \cdot \begin{bmatrix} l & & & \\ l & l & & \\ l & l & l & \\ l & l & l & l \end{bmatrix}$$

  compress columns downwards (top to bottom) going from left to right using orthogonal pre-multiplication, using orthgonal multiplications from the left side (pre-multiplication), start elimination with the last column and progress from right to left.

- $T = RQ$

$$\longrightarrow \begin{bmatrix} r & r & r & r \\ & r & r & r \\ & & r & r \\ & & & r \end{bmatrix} \cdot \begin{bmatrix} q & q & q & q \\ q & q & q & q \\ q & q & q & q \\ q & q & q & q \end{bmatrix}$$

  compress rows towards the right, using orthogonal transformation from the right side (post-multiplication); start elimination with last row and progress upward.

- $T = LQ$

$$\longrightarrow \cdot \begin{bmatrix} l & & & \\ l & l & & \\ l & l & l & \\ l & l & l & l \end{bmatrix} \cdot \begin{bmatrix} q & q & q & q \\ q & q & q & q \\ q & q & q & q \\ q & q & q & q \end{bmatrix}$$

compress rows downwards starting from the top, where R denotes an upper echelon type matrix and L a corresponding lower echelon type matrix. All this variations of the QR decomposition can be determined by appropriately modified sequences of Givens rotations or general rotations applied from the left or from the right. Also, note that the matrices R, L and Q used in this section may all be different in spite of using the same symbols and in spite of starting the same T. When the matrix T is square non-singular, then the R are upper triangular and the L lower triangular and both are invertible.

6.2 Elimination Scheme for QL factorization

We briefly sketch the elimination scheme for computing a QL factorization of a given matrix. The computational tool, i.e. Householder reflections is exactly the same as used for computing the QR factorization

we discussed earlier. The difference lies in the sequence of elimination steps and the choice of vectors for constructing the reflection.

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad y = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ \pm\sqrt{x'x} \end{bmatrix}.$$

The change of the elimination strategy amounts to start on the far right side moving towards the left side and then creating zeros on the top. Successive eliminations on a 6×4 matrix (in which the $\cdot$ indicates a relevant element of the matrix)

$$\begin{bmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{bmatrix} (H_{16}) \mapsto \begin{bmatrix} \cdot & \cdot & \cdot & 0 \\ \cdot & \cdot & \cdot & 0 \\ \cdot & \cdot & \cdot & 0 \\ \cdot & \cdot & \cdot & 0 \\ \cdot & \cdot & \cdot & 0 \\ \star & \star & \star & \star \end{bmatrix} (H_{26}) \mapsto \begin{bmatrix} \cdot & \cdot & 0 & 0 \\ \cdot & \cdot & 0 & 0 \\ \cdot & \cdot & 0 & 0 \\ \cdot & \cdot & 0 & 0 \\ \star & \star & \star & 0 \\ \star & \star & \star & \star \end{bmatrix} (H_{36}) \mapsto \begin{bmatrix} \cdot & 0 & 0 & 0 \\ \cdot & 0 & 0 & 0 \\ \cdot & 0 & 0 & 0 \\ \star & \star & 0 & 0 \\ \star & \star & \star & 0 \\ \star & \star & \star & \star \end{bmatrix}$$

$$(H_{46}) \mapsto \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \star & 0 & 0 & 0 \\ \star & \star & 0 & 0 \\ \star & \star & \star & 0 \\ \star & \star & \star & \star \end{bmatrix} \mapsto \left[\begin{array}{cccc} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \hline \star & 0 & 0 & 0 \\ \star & \star & 0 & 0 \\ \star & \star & \star & 0 \\ \star & \star & \star & \star \end{array}\right] = \left[\begin{array}{c} 0 \\ \hline L \end{array}\right]$$

The elements that have changed during the last transformation are denoted by a '$\star$'. There are no fill-ins, a purposeful zero does not get modified later on. The end result is

$$\underbrace{H_{46}H_{36}H_{26}H_{16}}_{Q^T} T = \left[\begin{array}{c} 0 \\ \hline L \end{array}\right]$$

in which L is lower triangular.

The elimination scheme looks exactly the same if we were to use Generalized rotations instead of Householder reflections. The only difference is that the final orthogonal transformation Q will have $\det Q = +1$, when rotations are used.

7 Computing with Schur Complements

7.1 Schur Complements

We consider to eliminate the 21-block-entry of matrix by applying a lower triangular matrix from the left as

$$\begin{bmatrix} 1 & 0 \\ -\sigma & 1 \end{bmatrix} \cdot \begin{bmatrix} A & B \\ C & D \end{bmatrix} = \begin{bmatrix} A & B \\ C - \sigma A & D - \sigma B \end{bmatrix} = \begin{bmatrix} A & B \\ 0 & D - \sigma B \end{bmatrix}.$$

For the elimination of 21-block-entry B to work, we need to determine the necessary value of the matrix σ, which works as follows

$$C - \sigma A = 0 \quad \Longrightarrow \quad \sigma = CA^{-1},$$

as long as A is invertible. This choice for σ successfully eliminates the 21-block-entry and produces the value of the 22-entry as

$$D - \sigma B = D - CA^{-1}B,$$

which is called a *Schur Complement*.

Looking at the Schur Complement we can see that such an elimination process can be used to compute numerous elementary matrix operations such as addition and multiplication of matrices, but also the inversion and matrix-vector multiplications.

7.2 Transfer Function and Schur Complement

Using the formulas above we can see that the linear fractional map representing the transfer function T of a linear system expressed in terms of a given state-space realization $\{A, B, C, D\}$.

$$\begin{bmatrix} Z^{-1}x \\ y \end{bmatrix} = \underbrace{\begin{bmatrix} A & B \\ C & D \end{bmatrix}}_{\Sigma} \begin{bmatrix} x \\ u \end{bmatrix} \quad \longrightarrow \quad \begin{bmatrix} 0 \\ y \end{bmatrix} = \begin{bmatrix} ZA - 1 & ZB \\ C & D \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix}$$

after eliminating the state variable x the transfer function T comes as a Schur Complement, that is, we have

$$T(Z) = D + C(1 - ZA)^{-1}ZB.$$

7.3 Closed Form of the Inverse

A similar elimination process using an upper-triangular matrix from the right-hand side leads us to

$$\begin{bmatrix} A & B \\ 0 & D - CA^{-1}B \end{bmatrix} \cdot \begin{bmatrix} 1 & -\tau \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} A & B - A\tau \\ 0 & D - CA^{-1}B \end{bmatrix} = \begin{bmatrix} A & 0 \\ 0 & D - CA^{-1}B \end{bmatrix}$$

$$B - A\tau \quad \Longrightarrow \quad \tau = A^{-1}B$$

In summary, we can see the factorization of the original matrix into three factors

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \sigma & 1 \end{bmatrix} \begin{bmatrix} A & 0 \\ 0 & D - CA^{-1}B \end{bmatrix} \begin{bmatrix} 1 & \tau \\ 0 & 1 \end{bmatrix}.$$

This factorization can be used to provide a closed form for the inverse

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix}^{-1} = \begin{bmatrix} 1 & -\tau \\ 0 & 1 \end{bmatrix} \begin{bmatrix} A^{-1} & 0 \\ 0 & (D - CA^{-1}B)^{-1} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -\sigma & 1 \end{bmatrix} =$$

$$= \begin{bmatrix} A^{-1} + \tau(D - CA^{-1}B)^{-1}\sigma & -\tau(D - CA^{-1}B)^{-1} \\ -(D - CA^{-1}B)^{-1}\sigma & (D - CA^{-1}B)^{-1} \end{bmatrix} = \begin{bmatrix} A^{-1} + \tau\Delta\sigma & -\tau\Delta \\ -\Delta\sigma & \Delta \end{bmatrix},$$

where we have used the short-hand notation Δ for the inverse of the Schur Complement, that is, we have used

$$\Delta = (D - CA^{-1}B)^{-1}.$$

For this to hold we need to have a block-matrix a that is invertible. In case the original matrix is not invertible then the inverse can be replaced by the Moore-Penrose pseudo-inverse $A^{\dagger}$ for the overall formula to still hold.

7.4 Computing Determinants

We can use the Schur Complement also to compute the determinant of a block-partitioned matrix by exploiting the expression

$$\det \begin{bmatrix} A & B \\ C & D \end{bmatrix} = \det A \cdot \det (D - CA^{-1}B).$$

This is a particularily helpful formula of A happens to be a scalar values, as then, the computation of A^{-1} is trivial.

7.5 Fadeeva Algorithm

We consider the particular problem of using the Schur Complement computation as a means for solving a linear system of equation such as

$$Tu = y.$$

To this end we can plug in the coefficient matrix T as well as the right-hand side of the equation, that is y, in the block matrix, which is also called the 'pre-array'. The Schur Complement in the 22 entry of the post-array then carries the solution vector u according to

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} T & y \\ -1 & 0 \end{bmatrix} \quad \Longrightarrow \quad \begin{bmatrix} T & y \\ 0 & u \end{bmatrix}.$$

Such an elimination approach to actually work for general matrices T we need to also re-shape T to be upper triangular. This can be achieved if T is first modified by orthogonal transformations in a process similar to computing the QR decomposition of T, that is we compute

$$\begin{bmatrix} Q' & \\ & 1 \end{bmatrix} \begin{bmatrix} T & y \\ -1 & 0 \end{bmatrix} = \begin{bmatrix} R & y' \\ -1 & 0 \end{bmatrix},$$

using the expression $T = QR$ with $Q'Q = 1$ and $Q'y = y'$. The effect of the orthogonal matrix Q' is accomplished by applying Givens rotations from the left in order to eliminate all lower triangular entries in T, which will produce the factor R as a remainder. Schematically, we can represent this first phase of the algorithm as

$$\left[\begin{array}{ccc|c} t & t & t & y \\ t & t & t & y \\ t & t & t & y \\ t & t & t & y \\ \hline -1 & & & 0 \\ & -1 & & 0 \\ & & -1 & 0 \end{array}\right] \quad \overset{Q^T}{\longrightarrow} \quad \left[\begin{array}{ccc|c} r & r & r & y' \\ & r & r & y' \\ & & r & y' \\ & & 0 & y' \\ \hline -1 & & & 0 \\ & -1 & & 0 \\ & & -1 & 0 \end{array}\right].$$

After this step is completed, the algorithm switches to Gaussian elimination steps to annihilate the 21-block entry and to produce the Schur Complement

$$\begin{bmatrix} R & Q'y \\ -1 & 0 \end{bmatrix} \quad \Longrightarrow \quad \begin{bmatrix} R & Q'y \\ 0 & R^{-1}Q'y \end{bmatrix} = \begin{bmatrix} \star & \star \\ 0 & u \end{bmatrix}.$$

In more detail this second phase of the algorithm looks like this

$$\left[\begin{array}{ccc|c} r & r & r & y' \\ & r & r & y' \\ & & r & y' \\ & & 0 & y' \\ \hline -1 & & & 0 \\ & -1 & & 0 \\ & & -1 & 0 \end{array}\right] \quad \overset{L^{-1}}{\longrightarrow} \quad \left[\begin{array}{ccc|c} r & r & r & y' \\ & r & r & y' \\ & & r & y' \\ & & 0 & y' \\ \hline 0 & & & u \\ & 0 & & u \\ & & 0 & u \end{array}\right].$$

This algorithm is called a *Fadeeva algorithm*. If T is not invertible this approach will determine the least squares solution $u = T^{\dagger}y$, where $T^{\dagger}$ denotes the Moore-Penrose pseudo inverse of T.

The Fadeeva-algorithm is attractive if we have the task to design a hardware solution (e.g. application specific integrated circuits, parallel processing) dedicated for solving such systems in real-time.

7.6 Modified Fadeeva algorithm

An efficient and computationally attractive approach to solve a system of equations $Tu = y$, where T is simply an invertible square matrix of dimension $n \times n$ and y a given n-dimensional vector, uses orthogonal rotations as the sole elimination tool instead of the traditional and hazardous and complex Gaussian elimination or the inversion of a direct factorization $T = QR$ factorization as $T^{-1} = R^{-1}Q'$. It goes as follows for the system of equations $Ty = b$. Let us form the 'pre-array'

$$\left[\begin{array}{c|c|c} T^T & I & 0 \\ \hline -y^T & 0 & 1 \end{array}\right] = \left[\begin{array}{cccc|cccc} t & t & t & t & 1 & & & \\ t & t & t & t & & 1 & & \\ t & t & t & t & & & 1 & \\ \hline -y & -y & -y & -y & & & & 1 \end{array}\right]$$

and apply a single QR orthogonal elimination scheme to this pre-array (notice that the rotations only involve the first $n + 1$ columns). This results in an orthogonal matrix Q, applied from the left, which partitions conformably as

$$\left[\begin{array}{cc} Q_{11} & q_{12} \\ q_{21} & q_{22} \end{array}\right].$$

Applying this orthogonal matrix Q to the pre-array amounts to

$$\left[\begin{array}{cc} Q_{11} & q_{12} \\ q_{21} & q_{22} \end{array}\right] \left[\begin{array}{ccc} T' & I & 0 \\ -y' & 0 & 1 \end{array}\right] = \left[\begin{array}{ccc} R & Q_{11} & q_{12} \\ q_{21}T' - q_{22}y' & q_{21} & q_{22} \end{array}\right] = \left[\begin{array}{ccc} R & Q_{11} & q_{12} \\ 0 & q_{21} & q_{22} \end{array}\right].$$

Setting the 21-entry in the post-array has been made zero, we obtain the following identities

$$q_{21}T' - q_{22}y' = 0 \quad \Longrightarrow \quad q_{21} = q_{22}y^{-1'} = q_{22}u',$$

which produce the post-array to contain the data

$$\longrightarrow \quad \left[\begin{array}{c|c|c} R & Q_{11} & q_{12} \\ \hline 0 & q_{22}u' & q_{22} \end{array}\right] = \left[\begin{array}{cccc|ccc|c} 0 & r & r & r & q & q & q & q \\ & & r & r & q & q & q & q \\ & & & r & q & q & q & q \\ \hline 0 & 0 & 0 & 0 & u' & u' & u' & q_{22} \end{array}\right].$$

The post-array obviously contains a scaled version of the solution vector u' as the 22-entry of the post-array. The solution vector u' is multiplied by q_{22}, which is a purely scalar valued scaling factor. The post-array also carries this factor as its 23-entry.

8 Singular Value Decomposition

This section is adopted from Nicholas Higham's blog
`https://nhigham.com/2020/10/13/what-is-the-singular-value-decomposition`.

8.1 What is the Singular Value Decomposition?

A singular value decomposition (SVD) of a matrix $T \in \mathcal{R}^{m\times n}$ is a factorization

$$T = U \cdot \Sigma \cdot V', \quad \text{where } U \in \mathcal{R}^{m\times m} \text{ and } V \in \mathcal{R}^{n\times n} \text{ are orthogonal,} \tag{11}$$

and

$$\Sigma = \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_p \end{bmatrix} \in \mathcal{R}^{m\times n}, \quad \text{with } p = \min(m,n), \text{ and } \sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p \geq 0.$$

Partition $U = [u_1, \ldots, u_m]$ and $V = [v_1, \ldots, v_n]$. The σ_i are called the singular values of A and the u_i and v_i are the left and right singular vectors. We have

$$Tv_i = \sigma_i u_i,\ i = 1, 2 \ldots p.$$

The matrix Σ is unique, but U and V are not. The form of Σ is

$$\Sigma = \left[\begin{array}{ccc} \sigma_1 & & \\ & \ddots & \\ & & \sigma_n \\ \hline & 0 & \end{array}\right] \text{ for } m \geq n, \quad \Sigma = \left[\begin{array}{ccc|c} \sigma_1 & & & \\ & \ddots & & 0 \\ & & \sigma_m & \end{array}\right] \text{ for } m \leq n. \tag{12}$$

Here is an example, in which the entries of A have been specially chosen to give simple forms for the elements of the factors:

$$A = \begin{bmatrix} 0 & \frac{4}{3} \\ -1 & -\frac{5}{3} \\ -2 & -\frac{2}{3} \end{bmatrix} = \underbrace{\frac{1}{3}\begin{bmatrix} 1 & -2 & -2 \\ -2 & 1 & -2 \\ -2 & -2 & 1 \end{bmatrix}}_{U} \underbrace{\begin{bmatrix} 2\sqrt{2} & 0 \\ 0 & \sqrt{2} \\ 0 & 0 \end{bmatrix}}_{\Sigma} \underbrace{\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}}_{V'}. \tag{13}$$

The power of the SVD is that it reveals a great deal of useful information about norms, rank, and subspaces of a matrix and it enables many problems to be reduced to a trivial form.

8.2 Four Fundamental Subspaces of a Matrix

Since U and V are nonsingular, $\text{rank}(T) = \text{rank}(\Sigma) = r$, where $r \leq p$ is the number of nonzero singular values. Since the 2-norm and Frobenius norm are invariant under orthogonal transformations, $\|T\| = \|\Sigma\|$ for both norms, giving

$$\|T\|_2 = \sigma_1, \quad \|T\|_F = \Big(\sum_{i=1}^{r} \sigma_i^2\Big)^{1/2}, \tag{14}$$

and hence $\|T\|_2 \le \|T\|_F \le r^{1/2}\|T\|_2$. The range space and null space of T are given in terms of the columns of U and V by

$$\begin{aligned} \text{null}(T) &= \text{span}\{v_{r+1}, \dots, v_n\}, \\ \text{col}(T) &= \text{span}\{u_1, u_2, \dots, u_r\} \\ \text{row}(T) &= \text{span}\{v_1, v_2, \dots, v_r\} \\ \text{null}(T') &= \text{span}\{u_{r+1}, u_{r+2}, \dots, u_n\}. \end{aligned} \tag{15}$$

8.3 Sum of Rank-1 Matrices

We can write the SVD as

$$T = \begin{bmatrix} u_1, u_2 \dots, u_r \end{bmatrix} \cdot \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_r \end{bmatrix} \cdot \begin{bmatrix} v_1^T \\ v_2^T \\ \vdots \\ v_r^T \end{bmatrix} = \sum_{i=1}^{r} \sigma_i u_i v_i', \qquad (*), \tag{16}$$

which expresses T as a sum of r rank-1 matrices, the i-th of which has 2-norm σ_i.

8.4 Matrix Approximation

The famous Eckart-Young theorem (1936) says that

$$\min_{\text{rank}(B)=k} \|A - B\|_q = \begin{cases} \sigma_{k+1}, & q = 2, \\ \left(\sum_{i=k+1}^{r} \sigma_i^2\right)^{1/2}, & q = F, \end{cases} \tag{17}$$

and that the minimum is attained at

$$A_k = U D_k V', \quad D_k = \text{diag}(\sigma_1, \dots, \sigma_k, 0, \dots, 0). \tag{18}$$

In other words, truncating the sum (*) after $k < r$ terms gives the best rank-k approximation to A in both the 2-norm and the Frobenius norm. In particular, this result implies that when A has full rank the distance from A to the nearest rank-deficient matrix is σ_r.

8.5 Moore-Penrose Pseudoinverse

The Moore-Penrose Pseudoinverse of a matrix $A \in \mathbb{R}^{n \times n}$ can be expressed in terms of the SVD as

$$A^+ = V \cdot \left[\begin{array}{ccc|ccc} \sigma_1^{-1} & & & & & \\ & \ddots & & & & \\ & & \sigma_r^{-1} & & & \\ \hline & & & 0 & & \\ & & & & \ddots & \\ & & & & & 0 \end{array}\right] \cdot U'. \tag{19}$$

The least squares problem $\min_x \|b - Ax\|_2$, where $A \in \mathbb{R}^{m \times n}$ with $m \ge n$ is solved by $x = A^+ b$, and when A is rank-deficient this is the solution of minimum 2-norm. For $m < n$ this is an underdetermined system and $x = A^+ b$ gives the minimum 2-norm solution.

8.6 Polar Decomposition

We can write $A = U\Sigma V' = UV' \cdot V\Sigma V' \equiv PQ$, where P is orthogonal and Q is symmetric positive semidefinite. This decomposition $A = PQ$ is the polar decomposition and $Q = (A'A)^{1/2}$ is unique. This connection between the SVD and the polar decomposition is useful both theoretically and computationally.

8.7 Relations with Symmetric Eigenvalue Problem

The SVD is not directly related to the eigenvalues and eigenvectors of A. However, for $m \geq n$, $\quad A = U\Sigma V'$ implies

$$A^TA = V \begin{bmatrix} \sigma_1^2 & & \\ & \ddots & \\ & & \sigma_n^2 \end{bmatrix} V', \qquad AA' = U \begin{bmatrix} \sigma_1^2 & & & & & \\ & \ddots & & & & \\ & & \sigma_n^2 & & & \\ & & & 0 & & \\ & & & & \ddots & \\ & & & & & 0 \end{bmatrix} U', \tag{20}$$

so the singular values of A are the square roots of the eigenvalues of the symmetric positive semidefinite matrices $A'A$ and AA' (modulo m-n zeros in the latter case), and the singular vectors are eigenvectors. Moreover, the eigenvalues of the $(m+n) \times (m+n)$ matrix

$$C = \begin{bmatrix} 0 & A \\ A' & 0 \end{bmatrix} \tag{21}$$

are plus and minus the singular values of A, together with $|m-n|$ additional zeros if $m \neq n$, and the eigenvectors of C and the singular vectors of A are also related.

Consequently, by applying results or algorithms for the eigensystem of a symmetric matrix to $A'A, AA'$, or C one obtains results or algorithms for the singular value decomposition of A.

8.8 History and Computation

The SVD was introduced independently by Beltrami in 1873 and Jordan in 1874. Golub popularized the SVD as an essential computational tool and developed the first reliable algorithms for computing it. The Golub-Reinsch algorithm, dating from the late 1960s and based on bi-diagonalization and the QR algorithm, is the standard way to compute the SVD. Various alternatives are available; see the references.

9 Recursive Matrix Decompositions

9.1 Introduction

At times we want to compute a matrix decomposition of a data matrix A, where the data matrix is continuously changing by either additional new rows a_i or additional new columns being added, that is

we have

$$\left[\begin{array}{c} A \\ \hline a_1 \\ \vdots \\ \vdots \end{array}\right] \longrightarrow \left[\begin{array}{c} A \\ \hline a_1 \\ a_2 \\ \vdots \end{array}\right] \longrightarrow \dots$$

For these scenarios we do not intend to preserve and store all data entries of the matrix and to re-compute the matrix factorization from scratch taking in the newly arrived rows. We'd rather compute recursively an update of the already known matrix factors. Of course, a similar approach is needed if we remove rows or columns from the data matrix. This is the essence of this section.

9.2 Recursive Computation of the QR Decomposition

9.2.1 Adding a Row Vector

Let's assume we have computed the QR decomposition of a given data matrix $A \in \mathcal{R}^{m \times n}$ as

$$A = Q \cdot R, \quad Q'Q = I, \quad R: \text{ upper triangular.}$$

Now, we add a new row vector $a \in \mathcal{R}^{1 \times n}$ to the matrix A and we want to determine the QR decomposition of this augmented matrix

$$\left[\begin{array}{c} A \\ \hline a \end{array}\right] = \hat{Q} \cdot \hat{R}.$$

We want to compute the matrices $\hat{Q}, \hat{R}$ not from scratch, but by reusing the factors of the QR decompositon of the original matrix Q, R.

$$\left[\begin{array}{c} A \\ \hline a \end{array}\right] = \left[\begin{array}{c|c} Q & \\ \hline & 1 \end{array}\right] \cdot \left[\begin{array}{c} R \\ \hline a \end{array}\right] = \underbrace{\left[\begin{array}{c|c} Q & \\ \hline & 1 \end{array}\right] \cdot Q_1}_{\hat{Q}} \cdot \left[\begin{array}{c} \hat{R} \\ \hline 0 \end{array}\right], \quad Q_1 \in \mathcal{R}^{(n+1)\times(n+1)} \tag{22}$$

The algorithmic step depicted in the right hand side of Equation 22 implies that we eliminate the elements of the new row vector a by means of Givens rotations and producing the updated triangular matrix $\hat{R}$. Schematically, this looks like

$$\left[\begin{array}{c} R \\ \hline a \end{array}\right] = \left[\begin{array}{ccccc} r & r & r & r & r \\ & r & r & r & r \\ & & r & r & r \\ & & & r & r \\ & & & & r \\ \hline x & x & x & x & x \end{array}\right] \xrightarrow{G'_{16}} \left[\begin{array}{ccccc} \hat{r} & \hat{r} & \hat{r} & \hat{r} & \hat{r} \\ & r & r & r & r \\ & & r & r & r \\ & & & r & r \\ & & & & r \\ \hline 0 & x & x & x & x \end{array}\right] \xrightarrow{G'_{26}} \left[\begin{array}{ccccc} \hat{r} & \hat{r} & \hat{r} & \hat{r} & \hat{r} \\ & \hat{r} & \hat{r} & \hat{r} & \hat{r} \\ & & r & r & r \\ & & & r & r \\ & & & & r \\ \hline 0 & 0 & x & x & x \end{array}\right] \longrightarrow$$

$$\xrightarrow{G'_{36}} \left[\begin{array}{ccccc} \hat{r} & \hat{r} & \hat{r} & \hat{r} & \hat{r} \\ & \hat{r} & \hat{r} & \hat{r} & \hat{r} \\ & & \hat{r} & \hat{r} & \hat{r} \\ & & & r & r \\ & & & & r \\ \hline 0 & 0 & 0 & x & x \end{array}\right] \xrightarrow{G'_{46}} \left[\begin{array}{ccccc} \hat{r} & \hat{r} & \hat{r} & \hat{r} & \hat{r} \\ & \hat{r} & \hat{r} & \hat{r} & \hat{r} \\ & & \hat{r} & \hat{r} & \hat{r} \\ & & & \hat{r} & \hat{r} \\ & & & & r \\ \hline 0 & 0 & 0 & 0 & x \end{array}\right] \xrightarrow{G'_{56}} \left[\begin{array}{ccccc} \hat{r} & \hat{r} & \hat{r} & \hat{r} & \hat{r} \\ & \hat{r} & \hat{r} & \hat{r} & \hat{r} \\ & & \hat{r} & \hat{r} & \hat{r} \\ & & & \hat{r} & \hat{r} \\ & & & & \hat{r} \\ \hline 0 & 0 & 0 & 0 & 0 \end{array}\right] = \left[\begin{array}{c} \hat{R} \\ \hline 0 \end{array}\right],$$

where G_{ij} denotes the Givens rotation between rows i and j with the purpose to eliminate the matrix entry j, i. This updating procedure requires only n Givens rotations to produce $\hat{R}$. Additional operations are necessary if one computes the update orthogonal matrix $\hat{Q}_1$ by collecting the all the involved Givens rotation matrices G_{ij}.

9.3 Recursive Computation of the SVD

Let's assume we have computed the SVD of a given data matrix $A \in \mathcal{R}^{m \times n}$ as

$$A = U \cdot \Sigma \cdot V'.$$

Now, we add a new row vector $a \in \mathcal{R}^{1 \times n}$ to the matrix A and we want to determine the SVD of this augmented matrix

$$\left[\begin{array}{c} A \\ \hline a \end{array}\right] = \hat{U} \cdot \hat{\Sigma} \cdot \hat{V}'.$$

We want to compute the matrices $\hat{U}, \hat{\Sigma}, \hat{V}$ not from scratch, but by reusing the factors of the SVD of the original matrix U, Σ, V

$$\left[\begin{array}{c} A \\ \hline a \end{array}\right] = \left[\begin{array}{c|c} U & \\ \hline & 1 \end{array}\right] \cdot \left[\begin{array}{c|c} \Sigma & \\ \hline & 1 \end{array}\right] \cdot \left[\begin{array}{c} V' \\ \hline a \end{array}\right]$$

With the following conventions

$$\hat{x} = a \cdot V, \quad a = e + \hat{x}V', \quad \sigma = \|e\|, \quad \hat{e} = \frac{1}{\sigma} \cdot e$$

we convert the task into

$$\left[\begin{array}{c} A \\ \hline a \end{array}\right] = \left[\begin{array}{c|c} U & \\ \hline & 1 \end{array}\right] \cdot \left[\begin{array}{c|c} \Sigma & \\ \hat{x} & \sigma \end{array}\right] \cdot \left[\begin{array}{c} V' \\ \hline \hat{e} \end{array}\right]. \tag{23}$$

The updating now requires us to compute the SVD of

$$\left[\begin{array}{c|c} \Sigma & \\ \hat{x} & \sigma \end{array}\right] = U' \cdot \hat{\Sigma} \cdot V'.$$

to arrive at

$$\left[\begin{array}{c} A \\ \hline a \end{array}\right] = \underbrace{\left[\begin{array}{c|c} U & \\ \hline & 1 \end{array}\right] \cdot U'}_{\hat{U}} \cdot \hat{\Sigma} \cdot \underbrace{V'^T \left[\begin{array}{c} V^T \\ \hline \hat{e} \end{array}\right]}_{\hat{V}^T}$$

Proceeding this way we can compute the SVD of a large matrix by starting with a single row and adding successively rows while updating the SVD.

9.4 Core Computation

Let's have a look at how to compute the middle factor in Equation (23), which looks like

$$\left[\begin{array}{c|c} \Sigma & \\ \hat{x} & \sigma \end{array}\right] = \left[\begin{array}{ccc|c} \ddots & & & \\ & \ddots & & \\ & & \ddots & \\ \hline \cdots & \cdots & \cdots & \cdot \end{array}\right].$$

Converting this bordered matrix into a lower bi-diagonal matrix (upper Hessenberg) form by orthogonal transformations

$$U'^T \cdot \left[\begin{array}{c|c} \Sigma & \\ \hline \hat{x} & \sigma \end{array}\right] \cdot V' = \left[\begin{array}{ccccc|c} \ddots & & & & & \\ \ddots & \ddots & & & & \\ & \ddots & \ddots & & & \\ & & \ddots & \ddots & & \\ \hline & & & \cdot & \cdot & \end{array}\right]$$

This lower Hessenberg matrix is not further processed until convergence towards a diagonal form, but it contains the updated singular values and corresponding matrices U' and V', which span the subspaces of interest.

Literatur

[1] I. Gohberg, T. Kailath and I. Koltracht. Linear complexity algorithms for semiseparable matrices. Integral Equations and Operator Theory, vol. 8, pp. 780-804, Birkhauser Verlag, 1985.

[2] P. Dewilde, A.-J. van der Veen. *Time-Varying Systems and Computations*. Kluwer, 1989.

[3] S. Chandrasekaran, P. Dewilde, M. Gu, T. Pals, A.-J. van der Veen, D. White. Fast Stable Solvers for Sequentially Semi-Separable Linear System of Equations. Lawrence Livermore Nationall Laboratory. Report UCRL-CR-151499, 2003.

[4] R.Vandebril, M. van Barel, N. Mastonardi. Matrix Computations and Semiseparable Matrices, Vol.1 Linear Systems. Johns Hopkins University Press, 2008.

[5] G. Strang. *Computational Science and Engineering*. Wellesley-Cambridge Press, 2007.

[6] G. Golub, Ch. van Loan. *Matrix Computations*. John Hopkins, 1992.