Numerical Matrix Analysis
Lecture Notes #6 — The QR-Factorization and Least Squares
Problems: Orthogonality and Projections

Peter Blomgren,
⟨blomgren.peter@gmail.com⟩

Department of Mathematics and Statistics
Dynamical Systems Group
Computational Sciences Research Center
San Diego State University
San Diego, CA 92182-7720

http://terminus.sdsu.edu/

Spring 2016
Outline

1 Recap
   - Checking the Roadmap

2 Projectors
   - Idempotent Matrices; Range & Nullspace; Complementary
   - Orthogonal Projectors
   - Orthonormal and Non-orthonormal Basis

3 The QR-Factorization
   - The Full and Reduced QR-Factorizations
   - Gram-Schmidt Orthogonalization
   - QR: Existence and Uniqueness
So far we have reviewed (or quickly introduced) basic linear algebra concepts, e.g.

- Vector and Matrix operations, including norms.
- Matrix properties (vocabulary): rank, range, nullspace, domain, Hermitian conjugate (adjoint), unitary...

Then we introduced the idea — from a geometrical perspective — of the Singular Value Decomposition $A = U\Sigma V^*$ of a matrix.

Finally, we connected the SVD and its properties to the majority of the concepts introduced.

In a sense, with the SVD we have extracted all information from the matrix $A$ and we are “done.”
Problem #1: We do not have a stable algorithm to compute the SVD. (We don’t even know what “stable” means!)

Problem #2: Even when we have such an algorithm (later in the semester), it will turn out to be quite computationally expensive.

The Approach: We will now start building our computational toolbox so that in the end we can implement a stable, effective algorithm for the SVD.

Along the way we will study other decompositions which may not be as complete as the SVD, but are cheaper to compute and are quite useful in certain applications.
A Quick Check of the Roadmap

- **Projectors**: Orthogonal and non-orthogonal projection matrices.

- **The QR-Factorization**
  - As an idea...
  - Computed using Gram-Schmidt orthogonalization
  - Computed using Householder triangularization
  - Alternative **not** discussed: Computed using Givens rotation
  - Solving least-squares problems using the QR-factorization
Figure: **LEFT** — Projecting a geometrical shape onto different planes (the figure itself is a 2D projection of this 3D-to-2D projection!); **RIGHT** — Map projections; $S^2 \to \mathbb{R}^2$, and $S^2 \to \mathbb{R} \times [−\pi, \pi]$. 
Definition (Projector)

A **projector** is a square matrix $P$ that satisfies

$$P^2 = P.$$ 

Think, for instance of

$$P = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix},$$

as the projection of a vector in $\mathbb{R}^3$ onto $\mathbb{R}^2$: — find a corner in the room; put a broom-stick in the corner and let it point into the room; observe the shadow on the floor. (This is making the assumption that the lighting is arranged so that all light-rays go straight from ceiling-to-floor...)
More generally, $P : \vec{v} \rightarrow P\vec{v}$ maps the vector $\vec{v}$ onto $\text{range}(P)$.

Clearly, once the $\vec{p} = P\vec{v}$ is on the range of $P$, another projection has no effect, hence

$$P\vec{p} = P^2\vec{v} = P\vec{v} \quad \Leftrightarrow \quad P(P\vec{v} - \vec{v}) = P^2\vec{v} - P\vec{v} = 0$$

Thus $(P\vec{v} - \vec{v}) \in \text{null}(P)$. If we think in terms of the projection being the shadow of a light-source illuminating $\vec{v}$, it means that the direction of the light-rays are described by a vector in $\text{null}(P)$.
If $P$ is a projector, then $(I - P)$ is also a projector

$$(I - P)^2 = I^2 - IP - PI + P^2 = I - 2P + P = I - P$$

$(I - P)$ is the **complementary projector** to $P$.

We have the following properties

$$\begin{align*}
\text{range}(I - P) &= \text{null}(P) \\
\text{null}(I - P) &= \text{range}(P) \\
\text{null}(I - P) \cap \text{null}(P) &= \{0\} \\
\text{range}(P) \cap \text{null}(P) &= \{0\}
\end{align*}$$

$\text{range}(I - P) \supseteq \text{null}(P)$, since if $P\vec{v} = 0$, then $(I - P)\vec{v} = \vec{v}$

$\text{range}(I - P) \subseteq \text{null}(P)$, since $\forall \vec{v}$, $(I - P)\vec{v} = \vec{v} - P\vec{v} \in \text{null}(P)$. 

We notice that a projector $P$ separates $\mathbb{C}^m$ into two spaces. Conversely, if $S_1, S_2 \subseteq \mathbb{C}^m$ such that $S_1 \cap S_2 = \{\overline{0}\}$, and $S_1 + S_2 = \mathbb{C}^m$, then $S_1$ and $S_2$ are **complementary subspaces** and there exists a projector $P$ **onto** $S_1$ **along** $S_2$ such that $\text{range}(P) = S_1$, and $\text{null}(P) = S_2$.

An **orthogonal projector** is a projector that projects onto a subspace $S_1$ along a space $S_2$, where $S_1$ and $S_2$ are orthogonal.
Warning!!!
Orthogonal projectors are not orthogonal matrices!!!

An **orthogonal projector** is a projector that is also Hermitian, *i.e.*

\[ P^* = P, \quad \text{and} \quad P^2 = P \]

If \( P = P^* \), then the inner product between \( P\bar{x} \in S_1 \) and \( (I - P)\bar{y} \in S_2 \) is zero:

\[
\bar{x}^* P^* (I - P)\bar{y} = \bar{x}^* (P - P^2)\bar{y} = 0
\]
Recap
Projectors

The QR-Factorization
Idempotent Matrices; Range & Nullspace; Complementary
Orthogonal Projectors
Orthonormal and Non-orthonormal Basis

Orthogonal Projectors

We now show that if $P$ projects onto $S_1$ along $S_2$ ($S_1 \perp S_2$, and $S_1$ has dimension $n$), then $P = P^*$ — the proof will give us a very simple characterization of the projector in terms of the SVD!

We construct the SVD of $P$ as follows: Let $\{\bar{q}_1, \bar{q}_2, \ldots, \bar{q}_m\}$ be an orthonormal basis for $\mathbb{C}^m$, where $\{\bar{q}_1, \bar{q}_2, \ldots, \bar{q}_n\}$ is a basis for $S_1$, and $\{\bar{q}_{n+1}, \bar{q}_{n+2}, \ldots, \bar{q}_m\}$ is a basis for $S_2$. We have

$$
\begin{cases}
P\bar{q}_j = \bar{q}_j, & j \leq n \\
P\bar{q}_j = 0, & j > n
\end{cases}
$$

Now, let $Q$ be the unitary $m \times m$ matrix whose $j$th column is $\bar{q}_j$. 

Peter Blomgren, blomgren.peter@gmail.com
QR & LSQ: Orthogonality and Projections — (12/26)
Orthogonal Projectors

With this constriction we have

\[
PQ = \begin{bmatrix}
\bar{q}_1 & \ldots & \bar{q}_n & \bar{0} & \ldots
\end{bmatrix}
\]

and

\[
Q^*PQ = \text{diag}(1, \ldots, 1, 0, \ldots, 0) = \Sigma
\]

Thus we have constructed an SVD of \(P\):

\[
P = Q\Sigma Q^*
\]

and clearly \(P\) is Hermitian

\[
P^* = (Q\Sigma Q^*)^* = Q\Sigma^* Q^* = Q\Sigma Q^* = P.
\]

\[\Box\]
Projection with an Orthonormal Basis

Since some singular values (in $\Sigma$) are zero, we can use the reduced SVD instead, \( i.e. \) we only keep the first \( n \) columns in $Q$, and we end up with

$$P = \hat{Q}\hat{Q}^*$$

where the columns of $Q \in \mathbb{C}^{m \times n}$ are orthonormal.

There is nothing magic about orthonormal vectors coming from the SVD — as long as the columns, $\bar{q}_j \in \mathbb{C}^m$, of $\hat{Q}$ are orthonormal, $P = QQ^*$ defines an orthogonal projection.
The projection

\[ \tilde{v} \rightarrow P\tilde{v} = QQ^*\tilde{v} = \sum_{i=1}^{n}(\tilde{q}_i\tilde{q}_i^*)\tilde{v} \]

can be viewed as a sum on \( n \) rank-one projections,

\[ P_i = \tilde{q}_i\tilde{q}_i^* \]

where each such projection isolates the component in a single direction of \( \tilde{q}_i \). These rank-one projectors will show up as building blocks in future algorithms.

For completeness, we note that the complement of a rank-one projector is a rank-\((m-1)\) projector that eliminates the component in the direction of \( \tilde{q}_i \)

\[ P_{\perp\tilde{q}_i} = I - \tilde{q}_i\tilde{q}_i^* \]
Projection with a Non-Orthonormal Basis

We can build an orthogonal projector from an arbitrary (not necessarily orthogonal) basis.

Let $S_1$ be the subspace spanned by the linearly independent vectors $\{\bar{a}_1, \ldots, \bar{a}_n\}$ and let $A$ be the matrix with columns $\bar{a}_j$.

\[
\begin{align*}
\bar{v} \quad &\xrightarrow{P} \quad \bar{y} \in \text{range}(A), \quad \bar{y} = A\bar{x}, \text{ some } \bar{x} \in \mathbb{C}^m \\
\bar{y} - \bar{v} \quad &\perp \quad \text{range}(A) \\
\iff & \quad \bar{a}_j^*(\bar{y} - \bar{v}) = 0, \forall j \\
\iff & \quad \bar{a}_j^*(A\bar{x} - \bar{v}) = 0, \forall j \\
\iff & \quad A^*(A\bar{x} - \bar{v}) = 0 \\
\iff & \quad A^*A\bar{x} = A^*\bar{v} \\
\iff & \quad \bar{x} = (A^*A)^{-1}A^*\bar{v} \\
\iff & \quad \bar{y} = A(A^*A)^{-1}A^*\bar{v} = P\bar{v}
\end{align*}
\]
The key thing we bring from the discussion on projections is the ability to identify how much of the “action” is directed in a certain set of directions, or subspace.

These ideas will be used, explicitly or implicitly, in many algorithms presented in this (and other) classes.

We now turn our attention to one of the “heavy-hitters” among numerical algorithms — the QR-factorization.
The Reduced QR-Factorization

In many application we are interested in the column spaces spanned by a matrix $A$, i.e. the spaces

$$\text{span}\langle \bar{a}_1 \rangle \subseteq \text{span}\langle \bar{a}_1, \bar{a}_2 \rangle \subseteq \text{span}\langle \bar{a}_1, \bar{a}_2, \bar{a}_3 \rangle \subseteq \ldots$$

We may, for instance, be looking for a minimum, or maximum of some quantity over each subspace.

**The QR-factorization** generates a sequence of orthonormal vectors $\{\bar{q}_1, \bar{q}_2, \bar{q}_3, \ldots\}$ that spans these spaces, i.e.

$$\text{span}\langle \bar{q}_1, \bar{q}_2, \ldots, \bar{q}_j \rangle = \text{span}\langle \bar{a}_1, \bar{a}_2, \ldots, \bar{a}_j \rangle, \quad j = 1, \ldots, n$$

The reason for doing this is that it is much easier to work in an orthonormal basis.
The Reduced QR-Factorization

**The Idea**

\[
\begin{align*}
\text{span}\langle \bar{q}_1 \rangle &= \text{span}\langle \bar{a}_1 \rangle \quad \Rightarrow \quad \bar{a}_1 = r_{11}\bar{q}_1 \\
\text{span}\langle \bar{q}_1, \bar{q}_2 \rangle &= \text{span}\langle \bar{a}_1, \bar{a}_2 \rangle \quad \Rightarrow \quad \bar{a}_2 = r_{12}\bar{q}_1 + r_{22}\bar{q}_2 \\
\text{span}\langle \bar{q}_1, \ldots, \bar{q}_3 \rangle &= \text{span}\langle \bar{a}_1, \ldots, \bar{a}_3 \rangle \quad \Rightarrow \quad \bar{a}_3 = r_{13}\bar{q}_1 + r_{23}\bar{q}_2 + r_{33}\bar{q}_3 \\
&\quad \vdots \\
\text{span}\langle \bar{q}_1, \ldots, \bar{q}_n \rangle &= \text{span}\langle \bar{a}_1, \ldots, \bar{a}_n \rangle \quad \Rightarrow \quad \bar{a}_n = r_{1n}\bar{q}_1 + \cdots + r_{nn}\bar{q}_n
\end{align*}
\]

In matrix notation, with \( A \in \mathbb{C}^{m \times n} \), \( \hat{Q} \in \mathbb{C}^{m \times n} \) with orthonormal columns, \( \hat{R} \in \mathbb{C}^{n \times n} \)

\[A = \hat{Q}\hat{R}\]
As for the SVD, we can extend the QR-factorization by “fleshing out” $\hat{Q}$ with an additional $(m - n)$ orthonormal columns, and zero-padding $\hat{R}$ with an additional $(m - n)$ rows of zeros:

\[
\begin{array}{c|c|c|c}
\hline
\text{Figure: The Reduced QR-Factorization, } A = \hat{Q}\hat{R} \\
\hline
\end{array}
\begin{array}{c|c|c|c}
\hline
\text{Figure: The Full QR-Factorization, } A = QR \\
\hline
\end{array}
\]

In the full QR-factorization, the columns $\bar{q}_j, j > n$ are orthogonal to $\text{range}(A)$. If $\text{rank}(A) = n$, they are an orthonormal basis for $\text{range}(A)^\perp = \text{null}(A^*)$, the space orthogonal to $\text{range}(A)$. 
The equations on slide 19 outline a method for computing reduced QR-factorizations.

At the $j$th step, we are looking to construct $\bar{q}_j \in \text{span}\langle \bar{a}_1, \ldots, \bar{a}_j \rangle$ such that $\bar{q}_j \perp \text{span}\langle \bar{q}_1, \ldots, \bar{q}_{j-1} \rangle$

We simply take $\bar{a}_j$, and subtract all the projections onto the directions $\bar{q}_1, \ldots, \bar{q}_{j-1}$, and then normalize the resulting vector

\[
(*) \quad \bar{v}_j = \bar{a}_j - (\bar{q}_1 \bar{q}^*_1)\bar{a}_j - \cdots - (\bar{q}_{j-1} \bar{q}^*_{j-1})\bar{a}_j
\]

\[
\bar{q}_j = \bar{v}_j / \|\bar{v}_j\|_2
\]

Computationally, it is more efficient to compute

\[
(*)' \quad \bar{v}_j = \bar{a}_j - \bar{q}_1(\bar{q}_1^* \bar{a}_j) - \cdots - \bar{q}_{j-1}(\bar{q}_{j-1}^* \bar{a}_j)
\]
We summarize our findings in an algorithm:

**Algorithm (Classical Gram-Schmidt)**

```plaintext
for j = 1:n
    \( \vec{v}_j = \vec{a}_j \)
    for i=1:(j-1)
        \( r_{ij} = \vec{q}_i^* \vec{a}_j \)
        \( \vec{v}_j = \vec{v}_j - r_{ij} \vec{q}_i \)
    endfor
    \( r_{jj} = \| \vec{v}_j \|_2 \)
    \( \vec{q}_j = \vec{v}_j / r_{jj} \)
endfor
```

Mathematically, we are done. Numerically, however, we can run into trouble due to roundoff errors.
Theorem (Existence of the QR-Factorization)

Every $A \in \mathbb{C}^{m \times n} (m \geq n)$ has a full QR-factorization, hence also a reduced QR-factorization.

Theorem (Uniqueness of the QR-Factorization)

Every $A \in \mathbb{C}^{m \times n} (m \geq n)$ of full rank has a unique reduced QR-factorization $A = \hat{Q}\hat{R}$, with $r_{jj} > 0$. 
1. If $A$ is full rank, the Gram-Schmidt algorithm gives the unique reduced QR-factorization.

2. If $A$ does not have full rank, then $\vec{v}_j = 0$ can occur during the iteration; if it does set $\vec{q}_j$ to be an arbitrary vector orthogonal to span$\langle \vec{q}_1, \ldots, \vec{q}_{j-1} \rangle$, and proceed.

3. If $m > n$, follow Gram-Schmidt as described until $j = n$, then take an addition $m - n$ steps, introducing arbitrary orthogonal $\vec{q}_j$ in each step.
Solving \( A\tilde{x} = \tilde{b} \) by QR-Factorization

If we have a QR-factorization algorithm handy, then we have the following “algorithm” for solving \( A\tilde{x} = \tilde{b} \)

1. Compute the QR-factorization \( A = QR \).
2. Compute \( \tilde{y} = Q^*\tilde{b} \).
3. Solve \( R\tilde{x} = \tilde{y} \) for \( \tilde{x} \).

**Note:** Computing \( Q^*\tilde{b} \) is just a multiplication with a unitary matrix. Since \( |\det(Q^*)| = 1 \) this completely numerically stable in the sense that errors will not be magnified. (WE WILL QUANTIFY THIS SOON.)

**Note:** Solving \( R\tilde{x} = \tilde{y} \) is very easy (backward substitution) since \( R \) is upper triangular.

**Note:** The bulk of the work is in computing the QR-factorization (twice that of Gaussian Elimination).
Implement the reduced QR-factorization by Classical Gram-Schmidt orthogonalization.

Write a function which given an $A \in \mathbb{C}^{m \times n}$ computes $Q \in \mathbb{C}^{m \times n}$, and $R \in \mathbb{C}^{n \times n}$ — in matlab you want to start something like this (file: qr_cgs.m):

```matlab
function [Q,R] = qr_cgs(A)
  ...
```

See `help function` for help on writing functions.

Validate your function — test that $A - QR \approx 0$, that $Q$ is unitary and $R$ upper triangular. Compare the result with the built-in version of the QR-factorization (`help qr`). Can you find a matrix where your QR-factorization breaks?

Hand in your code, and your validation and test-cases.

Peter Blomgren, ⟨blomgren.peter@gmail.com⟩

QR & LSQ: Orthogonality and Projections — (26/26)