Numerical Matrix Analysis
Lecture Notes #8
The QR-Factorization: — Least Squares Problems

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

2 Least Squares Problems
   - Problem, Language...
   - Problem Set-up: the Vandermonde Matrix
   - Formal Statement

3 LSQ: The Solution
   - Pseudo-Inverse
   - The Moore-Penrose Matrix Inverse
   - 3.5 Algorithms for the LSQ Problem
Previously...

Computing the QR-factorization 3 ways:

**Gram-Schmidt Orthogonalization** — Modified vs. Classical.

**Householder Triangularization**

<table>
<thead>
<tr>
<th>Modified Gram-Schmidt</th>
<th>Householder</th>
</tr>
</thead>
<tbody>
<tr>
<td>Numerically stable</td>
<td><strong>Even better stability</strong></td>
</tr>
<tr>
<td><strong>Useful for iterative methods</strong></td>
<td>Not as useful for iterative methods</td>
</tr>
<tr>
<td>“Triangular Orthogonalization”</td>
<td>“Orthogonal Triangularization”</td>
</tr>
<tr>
<td>$AR_1 R_2 \ldots R_n = \hat{Q}$</td>
<td>$Q_n \ldots Q_2 Q_1 A = R$</td>
</tr>
<tr>
<td><strong>Work</strong> ~ $2mn^2$ flops</td>
<td><strong>Work</strong> ~ $2mn^2 - \frac{2n^3}{3}$ flops</td>
</tr>
</tbody>
</table>
| **Note:** No $Q$ at this lower cost!!!
Least squares data/model fitting is used everywhere; — social sciences, engineering, statistics, mathematics, ....

In our language, the problem is expressed as an **overdetermined system**

\[ A\bar{x} = \bar{b}, \quad A \in \mathbb{C}^{m \times n}, \quad m \gg n. \]

Since \( A \) is “tall and skinny,” we have more equations than unknowns.

The least squares solution is defined by

\[ \bar{x}_{LS} = \arg \min_{\bar{x} \in \mathbb{C}^n} \| \bar{b} - A\bar{x} \|^2_2. \]
The quantity \( \bar{r} = \bar{b} - A\bar{x} \) is known as the residual, and since our problem is overdetermined, we cannot (in general) hope to find an \( \bar{x}^* \) such that \( \bar{r}(\bar{x}^*) = \bar{0} \).

Minimizing some norm of \( \bar{r}(\bar{x}) \) is a close second best.

The choice of the 2-norm leads to a problem that is easy to work with, and it is usually the correct choice for statistical reasons — computing the least squares solution yields the Maximum Likelihood estimate (under certain conditions — independent identically distributed variables, etc...).
Example: Polynomial Data-Fitting

**Figure:** Illustrating the least-squares polynomial fit of degrees 1, 2, 3, 6, 12, and 18 to a data-set containing 38 points. The top panel of each figure shows the data-set and the fitted polynomial; the bottom panel shows the residual (as a function of the polynomial degree).

Peter Blomgren, ⟨blomgren.peter@gmail.com⟩  Least Squares Problems — (6/18)
So... How do we achieve this miracle of data fitting?!? We flip back to lecture #2 and express our system using the Vandermonde matrix

\[
A = \begin{bmatrix}
1 & x_1 & x_1^2 & \cdots & x_1^d \\
1 & x_2 & x_2^2 & \cdots & x_2^d \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & x_m & x_m^2 & \cdots & x_m^d 
\end{bmatrix}, \quad \tilde{c} = \begin{bmatrix}
c_0 \\
c_1 \\
c_2 \\
\vdots \\
c_d 
\end{bmatrix}, \quad \tilde{b} = \begin{bmatrix}
b_0 \\
b_1 \\
b_2 \\
\vdots \\
b_m 
\end{bmatrix},
\]

where the fitting polynomial is described using the coefficients \( \tilde{c} \)

\[
p(x) = c_0 + c_1 x + c_2 x^2 + \cdots + c_d x^d.
\]

Given the locations of the points \( \tilde{x} \), and a particular set of coefficients \( \tilde{c} \), the matrix-vector product \( \tilde{p} = A\tilde{c} \) evaluates the polynomial in those points, i.e. \( \tilde{p}^T = \{ p(x_1), p(x_2), \ldots, p(x_m) \} \).
Least-Squares: Thinking About Projectors

We can think of the least squares problem in as the problem of finding the vector in $\text{range}(A)$ which is closest to $\vec{b}$.

Since we are measuring using the 2-norm, “closest” $\overset{\text{def}}{=} \text{closest in the sense of Euclidean distance}$.

We look to minimize the residual, $\vec{r} = \vec{b} - A\vec{x}$.

The minimum residual must be orthogonal to $\text{range}(A)$. 

$$
\begin{align*}
\text{range}(A) \\
y = Ax = Pb
\end{align*}
$$

$$
\begin{align*}
\vec{b} \\
r = \vec{b} - Ax
\end{align*}
$$
Theorem (Linear Least Squares)

Let \( A \in \mathbb{C}^{m \times n} (m \geq n) \), and \( \bar{b} \in \mathbb{C}^m \) be given. A vector \( \bar{x} \in \mathbb{C}^n \) minimizes the residual norm \( \|\bar{r}\|_2 = \|\bar{b} - A\bar{x}\|_2 \), thereby solving the least squares problem, if and only if \( \bar{r} \perp \text{range}(A) \), that is

\[
A^*\bar{r} = 0, \quad \Leftrightarrow \quad A^*A\bar{x} = A^*\bar{b}, \quad \Leftrightarrow \quad A\bar{x} = P\bar{b}
\]

where the orthogonal projector \( P \in \mathbb{C}^{m \times m} \) maps \( \mathbb{C}^m \) onto \( \text{range}(A) \). The \( n \times n \) system \( A^*A\bar{x} = A^*\bar{b} \) (the normal equations), is non-singular if and only if \( A \) has full rank \( \Leftrightarrow \) The solution \( \bar{x}^* \) is unique if and only if \( A \) has full rank.
Hence, if $A$ has full rank, the least squares-solution $\tilde{x}_{LS}$ is uniquely determined by

$$\tilde{x}_{LS} = (A^* A)^{-1} A^* \bar{b}.$$  

The matrix

$$A^\dagger \overset{\text{def}}{=} (A^* A)^{-1} A^*$$

is known as the **pseudo-inverse** of $A$.

With this notation and language, the least squares problem comes down to computing one or both of

$$\tilde{x} = A^\dagger \bar{b}, \quad \bar{y} = P\bar{b}.$$  

We will look at $3\frac{1}{2}$ algorithms for accomplishing this.
Given $B \in \mathbb{C}^{m \times n}$, the Moore-Penrose generalized matrix inverse is a unique pseudo-inverse $B^\dagger$, satisfying

(i) $BB^\dagger B = B$

(ii) $B^\dagger BB^\dagger = B^\dagger$

(iii) $(BB^\dagger)^* = BB^\dagger$

(iv) $(B^\dagger B)^* = B^\dagger B$

The Moore-Penrose inverse is often referred to in the literature, so it is a good thing to know what it is...
The classical / straight-forward / bone-headed(?) way to solve the least squares problem is to solve the normal equations

\[ A^* A \bar{x} = A^* \bar{b}. \]

The preferred way of doing this is by computing the **Cholesky factorization** (details to follow at a later date)

\[ A^* A \overset{\text{Cholesky}}{\rightarrow} R^* R, \]

where \( R \) is an upper triangular matrix.

Hence, the equivalent system

\[ R^* R \bar{x} = A^* \bar{b}, \quad (A^\dagger = (R^* R)^{-1} A^*), \]

can be solved by a forward and a backward substitution sweep.
If (when) we can compute the reduced SVD

\[ A = \hat{U} \Sigma V^* \]

we can use \( \hat{U} \) to express the projector \( P = \hat{U} \hat{U}^* \), and end up with the linear system of equations

\[ \hat{U} \Sigma V^* \bar{x} = \hat{U} \hat{U}^* \bar{b}. \]

and we get \( \hat{x}_{LS} \) by

\[ \hat{x}_{LS} = V \Sigma^{-1} \hat{U}^* \bar{b}. \]

Here, the pseudo-inverse is

\[ A^\dagger = V \Sigma^{-1} \hat{U}^*. \]
With the reduced QR factorization, the game unfolds like this...

Given \( A = \hat{Q}\hat{R} \), we can project \( \bar{b} \) to the range of \( A \) using \( P = \hat{Q}\hat{Q}^* \), then the system

\[
\hat{Q}\hat{R}\tilde{x} = \hat{Q}\hat{Q}^*\bar{b}.
\]

has a unique solution, given by

\[
\tilde{x}_{\text{LS}} = \hat{R}^{-1}\hat{Q}^*\bar{b}, \quad (A^\dagger = \hat{R}^{-1}\hat{Q}^*).
\]

Note that we do not need \( Q \) explicitly, only the action \( Q^*\bar{b} \), which we can get cheaply from the \( Q \)-less version of Householder triangularization.
Take #3$\frac{1}{2}$ — The Q-less QR-Factorization

Say we computed $\hat{R}$ using the Householder Q-less QR-factorization, but “forgot” to compute $Q^*\bar{b}$, is everything lost?!?

No, we can still compute $\bar{x}_{LS}$ using the following sequence

$$\bar{x} = R^{-1}R^{-*}(A^*\bar{b})$$
$$\bar{r} = \bar{b} - A\bar{x}$$
$$\bar{e} = R^{-1}R^{-*}(A^*\bar{r})$$
$$\bar{x} = \bar{x} + \bar{e}.$$

The first step solves the **semi-normal equations**

$$R^*R\bar{x} = A^*\bar{b}.$$  

The remaining three steps takes one step of iterative refinement to reduce roundoff error.
## Algorithms for Least Squares: Comments

<table>
<thead>
<tr>
<th>Method</th>
<th>Work (flops)</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal Equations</td>
<td>$\sim mn^2 + \frac{n^3}{3}$</td>
<td>Fastest, sensitive to roundoff errors. Not recommended.</td>
</tr>
<tr>
<td>QR-Factorization</td>
<td>$\sim 2mn^2 - \frac{2n^3}{3}$</td>
<td>Your everyday choice. Can break when $A$ is close to rank-deficient.</td>
</tr>
<tr>
<td>SVD</td>
<td>$\sim 2mn^2 + 11n^3$</td>
<td>The Big Hammer™ more stable than the QR approach, but requires more computational work.</td>
</tr>
</tbody>
</table>

**Additional Comment:** If $m \gg n$, then the work for QR and SVD are both dominated by the first term, $2mn^2$, and the computational cost of the SVD is not excessive. However, when $m \approx n$ the cost of the SVD is roughly 10 times that of the QR-factorization.
Looking Forward

We can now compute (and use) one of the big important tools of numerical linear algebra — the QR-factorization.

Next, we finally(?) formalize the discussion on “numerical stability,” and then we take another look at some of our algorithms in the light of stability considerations.
HW#4 — Exploratory
Implement modified Gram-Schmidt QR-factorization. Work through experiment #1 and #2 in “lecture 9” of Trefethen & Bau. Make sure your versions of classical and modified GS can reproduce figure 9.1. Do exercises 9.1(a,b), and 9.2(a,b). For additional (non-mandatory) fun do exercises 9.1(c) and 9.2(c).