Numerical Matrix Analysis
Notes #15 — Conditioning and Stability
Least Squares Problems: Stability

Peter Blomgren
⟨blomgren@sdsu.edu⟩

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 2023
(Revised: March 8, 2023)
Outline

1. Least Squares Problems
   - Recap: Conditioning
   - Recap: Solution Strategies
   - Experiment: Test Problem

2. LSQ + Householder Triangularization
   - Theorem
   - Relative Error
   - Comparison with Gram-Schmidt

3. Conditioning
   - Normal Equations vs. Householder QR?!?
   - The SVD
   - Comments & Rank-Deficient Problems
Theorem (Conditioning of Linear Least Squares Problems)

Let $\vec{b} \in \mathbb{C}^m$ and $A \in \mathbb{C}^{m \times n}$ of full rank be given. The least squares problem, $\min_{\vec{x} \in \mathbb{C}^n} \| \vec{b} - A\vec{x} \|$ has the following 2-norm relative condition numbers describing the sensitivities of $\vec{y} = P\vec{b} \in \text{range}(A)$ and $\vec{x}$ to perturbations in $\vec{b}$ and $A$:

<table>
<thead>
<tr>
<th>Input, Output $\rightarrow$</th>
<th>$\vec{y}$</th>
<th>$\vec{x}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\vec{b}$</td>
<td>$\frac{1}{\cos \theta}$</td>
<td>$\frac{\kappa(A)}{\eta \cos \theta}$</td>
</tr>
<tr>
<td>$A$</td>
<td>$\frac{\kappa(A)}{\cos \theta}$</td>
<td>$\kappa(A) + \frac{\kappa(A)^2 \tan \theta}{\eta}$</td>
</tr>
</tbody>
</table>

$$\kappa(A) = \frac{\sigma_1}{\sigma_n} \in [1, \infty), \quad \cos(\theta) = \frac{\|\vec{y}\|}{\|\vec{b}\|} \in [0, 1], \quad \eta = \frac{\|A\| \|\vec{x}\|}{\|A\vec{x}\|} \in [1, \kappa(A))$$
Deconstructing $\eta$…

$$\eta = \frac{||A|| \cdot ||\vec{x}||}{||A\vec{x}||} \in [1, \kappa(A))$$

Without loss of generality, rescale $\vec{x}$ so that $||\vec{x}|| = 1$.

Now with $A = U\Sigma V^*$, the extreme cases correspond to

$$\vec{x} = \vec{v}_1 \implies \eta = \frac{||A||}{||A\vec{v}_1||} = \frac{\sigma_1}{\sigma_1} = 1,$$

$$\vec{x} = \vec{v}_n \implies \eta = \frac{||A||}{||A\vec{v}_n||} = \frac{\sigma_1}{\sigma_n} = \kappa(A).$$

So, we get the best conditioning of the Least Squares Problem when the formulation and model conspires such that the projection of the right-hand-side is parallel to the minor semi-axis of the ellipsoid $A\mathbb{S}^{n-1}$.

“Obviously!”

“But, why?!?” — It’s a bit counter-intuitive: the problem is most sensitive to perturbations along that semi-axis (by the argument from the previous lecture), so if we maximize the “signal-to-noise-ratio” (the relative error along that semi-axis) by having significant model-action there, we get better behavior. It means that adding “irrelevant” parts to the model can significantly reduce the accuracy of the computation.

“Careful Modeling Matters!”
Currently, we have four candidate methods for solving least squares problems:

- **The Normal Equations**
  \[ \hat{x} = (A^* A)^{-1} A^* \vec{b} \]

- **Gram-Schmidt Orthogonalization** (QR-factorization)
  \[ \hat{x} = R^{-1}(Q^* \vec{b}) \]

- **Householder Triangularization** (QR-factorization)
  \[ \hat{x} = R^{-1}(Q^* \vec{b}) \]

- **The Singular Value Decomposition**
  \[ \hat{x} = V(\Sigma^{-1}(U^* \vec{b})) \]
% The Dimensions of the problem
m = 100;
n = 15;

% The time-vector --- samples in [0,1]
t = (0:(m-1))’ / (m-1);

% Build the matrix A
A = [];
for p = 0:(n-1)
    A = [ A t.^p ];
end

% Build the right-hand side
b = exp(sin(4*t)) / 2006.787453080206;
Our Test Problem: Visualized

Figure: The rows of the matrix $A$, the columns of the matrix $A$, and the vector $\vec{b}$.
The normalization

\[
\text{b} = \frac{\exp(\sin(4t))}{2006.787453080206};
\]

Is chosen so that the correct (exact) value of the last component is \(x_{15} = 1\).

We are trying to compute the 14\textsuperscript{th} degree polynomial \(p_{14}(t)\) which fits \(\exp(\sin(4t))\) on the interval \([0, 1]\).
Finding 2006.787453080206 — Using Maple

Some Maple Action...

```maple
with(linalg);
Digits := 512;
m := 100;
n := 15;
f := (i,j) -> ((i-1)/(m-1))^(j-1);
A := Matrix(m,n,f);
g := (i) -> exp(sin(4*(i-1)/(m-1)));
b := Vector(100,g);
x := leastsqr(A,b);
evalf(x[15]);
```

Gives

\[ x_{15} = 2006.7874531048518338 \ldots \]

Curious... However, using this value instead didn’t change anything significantly in the following slides...
Approximation of Associated Condition Numbers

We use the best available solution \( (x = A\backslash b; \ y = A\times x) \) to estimate the dimensionless parameters, and condition numbers

<table>
<thead>
<tr>
<th>( \kappa(A) )</th>
<th>( \cos \theta )</th>
<th>( \eta )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{cond}(A) )</td>
<td>( \text{norm}(y) / \text{norm}(b) )</td>
<td>( \text{norm}(A) \times \text{norm}(x) / \text{norm}(y) )</td>
</tr>
<tr>
<td>( 2.27 \times 10^{10} )</td>
<td>( 0.99999999999426 )</td>
<td>( 2.10 \times 10^5 )</td>
</tr>
</tbody>
</table>

\[
\begin{array}{c|c|c}
\downarrow \text{Input, Output} & \vec{y} & \vec{x} \\
\hline
\vec{b} & 1.00 & 1.08 \times 10^5 \\
A & 2.27 \times 10^{10} & 3.10 \times 10^{10} \\
\end{array}
\]

\textbf{Bottom Line:} If we get 6 correct digits (error \( \sim 10^{-6} \)) in matlab (\( \varepsilon_{\text{mach}} \sim 10^{-16} \)) then we are doing as well as we can.
We have three ways of solving the least squares problem using the Matlab built-in Householder Triangularization

\[
\begin{align*}
[Q,R] &= qr(A,0); \\
x &= R\backslash (Q'*b); \\
e1 &= \text{abs}(x(15)-1);
\end{align*}
\]

\[
\begin{align*}
[~,R] &= qr([A \ b],0); \\
QstarB &= R(1:n,n+1); \\
R &= R(1:n,1:n); \\
x &= R\backslash QstarB; \\
e2 &= \text{abs}(x(15)-1);
\end{align*}
\]

\[
\begin{align*}
x &= A\backslash b; \\
e3 &= \text{abs}(x(15)-1);
\end{align*}
\]

- In the first approach, we explicitly form and use the matrix \( Q \).
- In the second approach, we extract the “action” \( Q^*\vec{b} \), by appending \( \vec{b} \) as an additional column in \( A \), and then identifying the appropriate components of the computed \( \tilde{R} \) as \( R \) and \( Q^*\vec{b} \).
- In the third approach, we rely on Matlab’s implementation... It uses Householder triangularization with column pivoting, for maximal accuracy.
The approaches described above gives us the following errors

\[ e_1 = 3.16387 \times 10^{-7}, \quad e_2 = 3.16371 \times 10^{-7}, \quad e_3 = 2.18674 \times 10^{-7} \]

Implicitly forming \( Q^* \bar{b} \) improves the result marginally, which means that the errors introduced in the explicit formation of \( Q^* \bar{b} \) are small compared to the errors introduced by the QR-factorization itself.

The Matlab solver, which includes all the bells and whistles, improves the result a little more;

All three variants are backward stable.
Householder Triangularization: Theorem

Theorem (Finding the Least Squares Solution Using Householder QR-Factorization is Backward Stable)

Let the full-rank least squares problem be solved by Householder triangularization in a floating-point environment satisfying the floating point axioms. This algorithm is backward stable in the sense that the computed solution $\tilde{x}$ has the property

$$\| (A + \delta A)\tilde{x} - \tilde{b} \| = \min_{\bar{x} \in \mathbb{C}^n} \| \tilde{b} - A\bar{x} \|,$$

$$\frac{\| \delta A \|}{\| A \|} = O(\varepsilon_{mach})$$

for some $\delta A \in \mathbb{C}^{m \times n}$. This is true whether $\hat{Q}^* \tilde{b}$ is formed explicitly or implicitly. Further, the theorem is true for Householder triangularization with arbitrary column pivoting.
Householder Triangularization: Relative Error

Figure: The relative error \( \frac{p(x) - b(x)}{b(x)} \) on the interval \([0, 1]\).
From homework, we have two ways of solving the least squares problem using modified Gram-Schmidt orthogonalization:

\[
\begin{align*}
\text{[Q,R]} &= \text{qr.mgs(A)}; \\
x &= R\backslash (Q'*b); \\
e_4 &= \text{abs}(x(15)-1);
\end{align*}
\]

\[
\begin{align*}
\text{[~,R]} &= \text{qr.mgs([A b])}; \\
Q\star b &= R(1:n,n+1); \\
R &= R(1:n,1:n); \\
x &= R\backslash Q\star b; \\
e_5 &= \text{abs}(x(15)-1);
\end{align*}
\]

- The explicit formation of \( Q \) in the first approach suffers from forward errors, and the result is quite disastrous

\[ e_4 = 0.03024 \]

- If instead we form \( Q\star \vec{b} \) implicitly (the second approach), the result is much better

\[ e_5 = 2.4854 \times 10^{-8} \]
The fact that $e_5 < e_{1,2,3}$ in this example is not an indication of anything in particular — it is just luck.

The following is a provable result:

**Theorem**

The solution of the full-rank least squares problem by modified Gram-Schmidt orthogonalization is also backward stable, provided that $Q^*\tilde{b}$ is formed implicitly, as indicated on the previous slide.
Even though the condition number for the least squares problem

\[ \kappa_{LS} = \kappa(A) + \frac{\kappa(A)^2 \tan \theta}{\eta} \]

contains \( \kappa(A)^2 \), we have successfully found the solution with \( \sim 6 \) correct digits.

Using the **normal equations** \( \tilde{x} = (A^*A)^{-1}(A^*\tilde{b}) \), we are subject to the full “force” of \( \kappa(A)^2 \), since

\[ \kappa(A^*A) \sim \kappa(A)\kappa(A^*) \sim \kappa(A)^2. \]

Matlab “barks” at us, if we try — \( x = (A'*A)\backslash(A'*b) \);

Warning: Matrix is close to singular or badly scaled.
Results may be inaccurate.  RCOND = 1.512821e-19.

and \( |\tilde{x}_{15} - x_{15}| = 1.678. \)
Even though the worst-case conditioning for the least squares problem is $\kappa(A)^2$, that is almost never realized.

In our test problem

$$\tan \theta \sim 3 \times 10^{-6}, \quad \eta \sim 2 \times 10^5$$

so, whereas

$$\kappa(A)^2 = 5.16 \times 10^{20}, \quad \frac{\kappa(A)^2 \tan \theta}{\eta} = 3.10 \times 10^{10}$$

But for $A^*A$ there are no mitigating factors, and

$$\kappa_{est}(A^*A) = 2.0 \times 10^{18} \quad \text{underestimate?}$$

SO

$$\kappa_{est}(A^*A) \cdot \varepsilon_{mach} = 4.4 \times 10^2$$
Theorem

The solution of the full-rank least squares problem via the normal equations is **unstable**. Stability can be achieved, however, by restriction to a class of problems in which $\kappa(A)$ is uniformly bounded above or $\frac{\tan \theta}{\eta}$ is uniformly bounded below.

**Bottom Line:** The normal equations only work for “easy” least squares problems, a.k.a. ”Friendly Homework problems.”
Solving the least squares problem using the SVD is the most expensive, but also the most stable method; here we get our error to be of the same order of magnitude as the other backward stable methods

\[ e_6 = 3.16383 \times 10^{-7} \]

**Theorem**

The solution of the full-rank least squares problem by the SVD is backward stable.
At this point we have four working backward stable approaches to solving the full rank least squares problem

- Householder triangularization
- Householder triangularization with column pivoting
- Modified Gram-Schmidt with implicit $Q^*\vec{b}$ calculation
- The SVD

The differences, in terms of classical norm-wise stability, among these algorithms are minor.

For everyday use, select the simplest one — Householder triangularization — as your default algorithm. If you are working in matlab use $A\backslash\vec{b}$ — Householder triangularization with column pivoting.
When \( \text{rank}(A) < n \), quite possibly with \( m < n \), the least squares problem is **under-determined**.

No unique solution exists, unless we add additional constraints. Usually, we look for the **minimum norm** solution \( \vec{x} \); i.e. among the infinitely many solutions we select the one with smallest norm.
When \( \text{rank}(A) < n \), quite possibly with \( m < n \), the least squares problem is **under-determined**.

No unique solution exists, unless we add additional constraints. Usually, we look for the **minimum norm** solution \( \vec{x} \); i.e. among the infinitely many solutions we select the one with smallest norm.

The solution depends (strongly) on \( \text{rank}(A) \), and determining numerical rank is non-trivial. Is \( 10^{-14} = 0 \)???
When \( \text{rank}(A) < n \), quite possibly with \( m < n \), the least squares problem is under-determined.

No unique solution exists, unless we add additional constraints. Usually, we look for the minimum norm solution \( \vec{x} \); i.e. among the infinitely many solutions we select the one with smallest norm.

The solution depends (strongly) on \( \text{rank}(A) \), and determining numerical rank is non-trivial. Is \( 10^{-14} = 0 \)???

For this class of problems, the only fully stable algorithms are based on the SVD.

Householder triangularization with column pivoting is stable for “almost all” such problems.

Rank-deficient least squares problems are a completely different class of problems, and we sweep all the details under the rug...