Numerical Matrix Analysis

Notes #8

The QR-Factorization: — Least Squares Problems

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 2024
(Revised: February 20, 2024)

Outline

1. Student Learning Targets, and Objectives
   - SLOs: QR-Factorization Least Squares Problems
2. Recap
3. Least Squares Problems
   - Problem, Language...
   - Problem Set-up: the Vandermonde Matrix
   - Formal Statement
4. LSQ: The Solution
   - Pseudo-Inverse
   - The Moore-Penrose Matrix Inverse
   - 3.5 Algorithms for the LSQ Problem

Student Learning Targets, and Objectives

Target Linear Least Squares Problems
- Objective Discrepancy Measure: Residual
- Objective Relation to the Maximum Likelihood Estimate
- Objective Polynomial Fitting, and the Vandermonde Matrix
- Objective The Moore-Penrose Pseudo-Inverse of a Matrix

Target Approaches
- Objective Normal Equations
- Objective Pseudo-Inverse Solution based on the SVD
- Objective Pseudo-Inverse Solution based on the QR-Factorization

Previously (Gram-Schmidt and Householder)

Computing the QR-factorization 3 ways:

Gram-Schmidt Orthogonalization — Modified vs. Classical.

Householder Triangularization

Modified Gram-Schmidt	**Householder**
Numerically stable* **Useful for iterative methods**	**Even better stability** Not as useful for iterative methods
"Triangular Orthogonalization" $AR_1R_2\ldots R_n = \widehat{Q}$	**"Orthogonal Triangularization"** $Q_n\ldots Q_2Q_1A = R$
Work $\sim 2mn^2$ flops	**Work** $\left(\sim 2mn^2 - \frac{2n^3}{3}\right)$ flops **Note:** No Q at this lower cost!!!

Least Squares

Least squares data/model fitting is used everywhere; — social sciences, engineering, statistics, mathematics, “data science” ...

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

$$A\vec{x} = \vec{b}, \quad A \in \mathbb{C}^{m\times n}, \; m \gg n.$$

Since A is “tall and skinny,” we have more equations than unknowns. ⇝ Very likely to be inconsistent.

The least squares solution is defined by

$$\vec{x}_{\text{LS}} = \arg\min_{\vec{x}\in\mathbb{C}^n} \left\| \vec{b} - A\vec{x} \right\|_2^2.$$

Least Squares: Some Language

The quantity $\vec{r}(\vec{x}) = \vec{b} - A\vec{x}$ is known as the **residual**, and since our problem is overdetermined, we cannot (in general) hope to find an $\vec{x}^*$ such that $\vec{r}(\vec{x}^*) = \vec{0}$.

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

This (among other things, like *e.g.* checking that large matrices contain zeros) is why we needed the discussion of norms back in [Lecture#3].

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

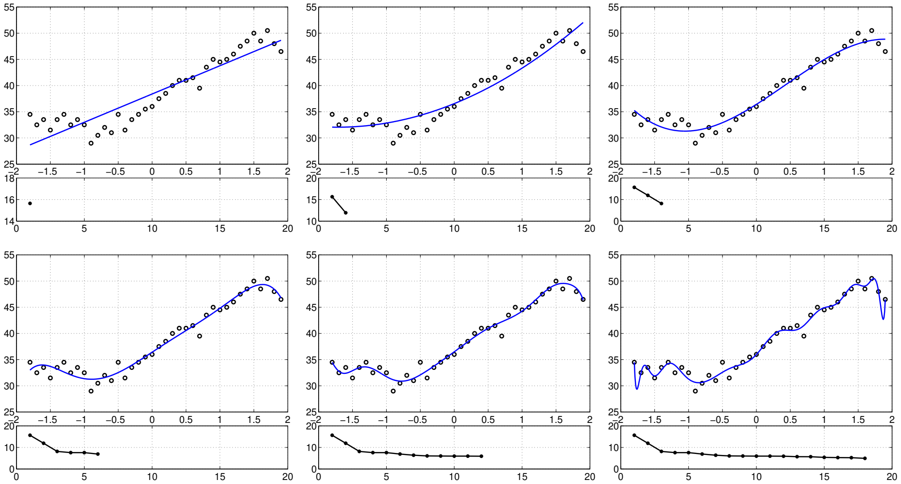


Figure: Illustrating the least-squares polynomial fit of degrees 1, 2, 3, 6, 12, and 18 to a data-set containing 38 points. The top panel of each figure shows the data-set and the fitted polynomial; the bottom panel shows the residual (as a function of the polynomial degree).

Least-Squares: Problem Set-Up

So... How do we fit (polynomial) models to data?!? We flip back to [Lecture#2] and express our system using the **Vandermonde matrix**

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

where the fitting polynomial is described using the coefficients $\vec{c}$

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

Given the locations of the points $\vec{x}$, and a particular set of coefficients $\vec{c}$, the matrix-vector product $\vec{p} = A\vec{c}$ evaluates the polynomial in those points, *i.e.* $\vec{p}^T = (p(x_1), p(x_2), \ldots, p(x_m))$.

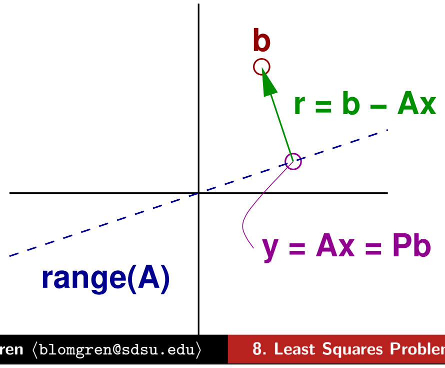
Least-Squares: Thinking About Projectors

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

Since we are measuring using the 2-norm, "closest" $\stackrel{\text{def}}{=}$ 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 $\mathrm{range}(A)$.



Least Squares: Formal Statement

Theorem (Linear Least Squares)

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

$$\underbrace{A^*\vec{r} = 0}_{\vec{r}\in\mathrm{null}(A^*)}, \quad \Leftrightarrow \quad A^*A\vec{x} = A^*\vec{b}, \quad \Leftrightarrow \quad A\vec{x} = P\vec{b}$$

where the orthogonal projector $P \in \mathbb{C}^{m\times m}$ maps $\mathbb{C}^m$ onto $\mathrm{range}(A)$. The $(n \times n)$ system $A^*A\vec{x} = A^*\vec{b}$ (the **normal equations***), is non-singular if and only if A has full rank $\Leftrightarrow$ The solution $\vec{x}^*$ is unique if and only if A has full rank.*

Language: The Pseudo-Inverse

Hence, if A has full rank, the least squares-solution $\vec{x}_{\mathrm{LS}}$ is uniquely determined by

$$\vec{x}_{\mathrm{LS}} = (A^*A)^{-1}A^*\,\vec{b}.$$

The matrix

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

is known as a **pseudo-inverse** of A.

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

$$\vec{x} = A^\dagger\vec{b}, \qquad \vec{y} = P\vec{b}$$

We will look at $\left(3 + \frac{1}{2}\right)$ algorithms for accomplishing this.

The Moore-Penrose Matrix Inverse — Pseudo-Inverse

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...

A Note on the Case $(m < n)$

When $A \in \mathbb{C}^{m\times n}$, $(m < n)$, we must have $\mathrm{rank}(A) \le m < n$, and $A^*A \in \mathbb{C}^{n\times n}$. Since $(n > m)$ this matrix cannot have full rank $\rightsquigarrow$ it is not invertible.

The rank-deficient scenario, where $\mathrm{rank}(A) < n$ requires "some" more thought.

The Normal Equations Matrix (A^*A) is not invertible $\rightsquigarrow$ we lose the "infinite precision" pseudo-inverse $(A^*A)^{-1}A^*$; and with it the uniqueness of "the" solution.

In order to make progress we have to (yet again) re-define what we mean by finding a solution... but that's a story for a different day.

Take#1 — The Normal Equations $\sim \left(mn^2 + \frac{n^3}{3}\right)$ flops

The classical / straight-forward / bone-headed(?) way to solve the least squares problem is to solve the normal equations

$$A^*A\vec{x} = A^*\vec{b}.$$

The preferred way of doing this is by computing the **Cholesky factorization** (essentially a symmetric row-reduction algorithm; details to follow in [Notes#17])

$$A^*A \quad \xrightarrow{\text{Cholesky}} \quad R^*R,$$

where R is an upper triangular matrix; The equivalent system

$$R^*R\vec{x} = A^*\vec{b}, \qquad (\, A^\dagger = (R^*R)^{-1}A^* \,),$$

can be solved by a forward and a backward substitution sweep.

Sidenote: There are specialized iterative schemes, *e.g.* CGNE (Conjugate Gradient on the Normal Equations) which are useful in certain circumstances (sparse A-matrix); see

- `https://en.wikipedia.org/wiki/Conjugate_gradient_method#Conjugate_gradient_on_the_normal_equations`
- `https://mathworld.wolfram.com/ConjugateGradientMethodontheNormalEquations.html`

Take#2 — The SVD $\sim \left(2mn^2 + 11n^3\right)$ flops

If we compute the reduced SVD

$$A = \widehat{U}\widehat{\Sigma}V^*,$$

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

$$\widehat{U}\widehat{\Sigma}V^*\vec{x} = \widehat{U}\widehat{U}^*\vec{b}.$$

and we get $\vec{x}_{\mathrm{LS}}$ by

$$\vec{x}_{\mathrm{LS}} = V\widehat{\Sigma}^{-1}\widehat{U}^*\vec{b}.$$

Here, the pseudo-inverse is expressed as

$$A^\dagger = V\widehat{\Sigma}^{-1}\widehat{U}^*.$$

Note: Since $\mathrm{rank}(A) = \mathrm{rank}(\widehat{\Sigma})$ this does not directly help with the rank-deficient problem.

Take#3 — The QR-Factorization $\sim \left(2mn^2 - \frac{2n^3}{3}\right)$ flops

With the reduced QR factorization, the game unfolds like this...

Given $A = \widehat{Q}\widehat{R}$, we can project $\vec{b}$ to the range of A using $P = \widehat{Q}\widehat{Q}^*$, then the system

$$\widehat{Q}\widehat{R}\vec{x} = \widehat{Q}\widehat{Q}^*\vec{b}.$$

has a unique solution, given by

$$\vec{x}_{\mathrm{LS}} = \widehat{R}^{-1}\widehat{Q}^*\vec{b}, \qquad (\, A^\dagger = \widehat{R}^{-1}\widehat{Q}^* \,).$$

Note: Again, $\mathrm{rank}(A) = \mathrm{rank}(R)$; i.e.we are not getting any direct help with the rank-deficient problem.

Comment

Note that we do not need Q explicitly, only the action $Q^*\vec{b}$, 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 $\widehat{R}$ using the Householder Q-less QR-factorization, but "forgot" to compute $Q^*\vec{b}$, is everything lost?!?

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

$$\begin{array}{rcl} \vec{x} & \leftarrow & R^{-1}R^{-*}(A^*\vec{b}) \\ \vec{r} & \leftarrow & \vec{b} - A\vec{x} \\ \vec{e} & \leftarrow & R^{-1}R^{-*}(A^*\vec{r}) \\ \vec{x} & \leftarrow & \vec{x} + \vec{e}. \end{array}$$

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

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

The remaining three steps takes one step of iterative refinement to reduce roundoff error.

Algorithms for Least Squares: Comments

Figures on Next Slides

Method	**Work** (flops)	**Comment**
Normal Equations	$\sim\left(mn^2 + \frac{n^3}{3}\right)$	Fastest, sensitive to roundoff errors. Not recommended.
QR-Factorization	$\sim\left(2mn^2 - \frac{2n^3}{3}\right)$	Your everyday choice. Can run into trouble when A is close to rank-deficient.
SVD	$\sim\left(2mn^2 + 11n^3\right)$	The Big Hammer$^{\text{TM}}$ more stable than the QR approach, but requires more computational work.

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.

Algorithms for Least Squares: Work Comparison 1/2

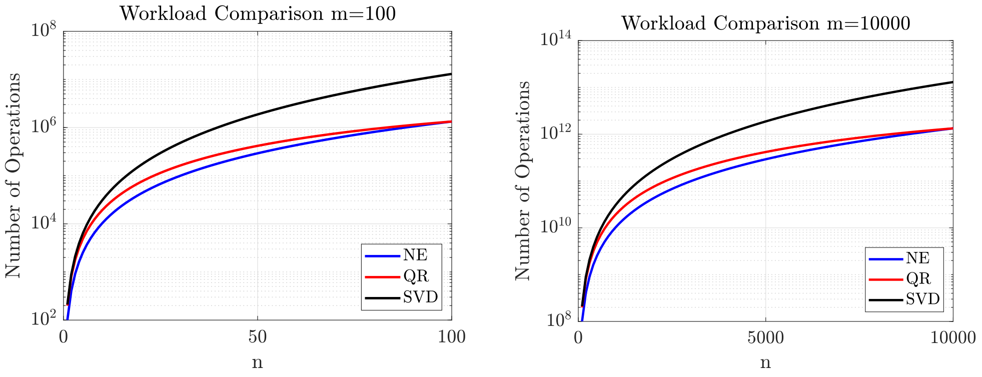


Figure: It is worth noting that the relative NE–QR–SVD work only depends on the aspect ratio — $\frac{n}{m} \in [0, 1]$

Algorithms for Least Squares: Work Comparison 2/2

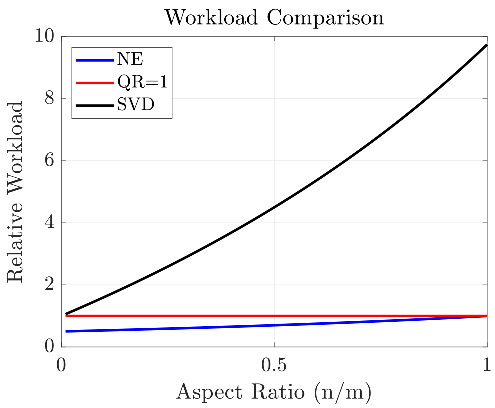


Figure: We have normalized so that the QR-workload is one; we notice that the NE "savings" are quite small (and come with extra instability issues); as the aspect ratio approaches one, the SVD-workload is about 10 times that of the QR-workload.

Looking Forward

We can now compute (and have a "serious" use for) 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 Due Date in Canvas/Gradescope

HW#4

1. Implement modified Gram-Schmidt QR-factorization.

   Write a function which given an $A \in \mathbb{C}^{m\times n}$ computes $Q \in \mathbb{C}^{m\times n}$, and $R \in \mathbb{C}^{n\times n}$ — $\texttt{qr\_mgs(A)} \to \texttt{Q}, \texttt{R}$.

   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.

   Note that depending on your coding environment, you may have to use larger (and worse conditioned) matrices to achieve the loss of orthogonality in classical Gram-Schmidt.
2. 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).

Homework AI-Policy Spring 2024

AI-era Policies — SPRING 2024

AI-3 Documented: *Students can use AI in any manner for this assessment or deliverable, but they must provide appropriate documentation for all AI use.*

This applies to ALL MATH-543 WORK during the SPRING 2024 semester.

The goal is to leverage existing tools and resources to generate HIGH QUALITY SOLUTIONS to all assessments.

You MUST document what tools you use and HOW they were used (including prompts); AND how results were VALIDATED.

BE PREPARED to DISCUSS homework solutions and AI-strategies. **Participation in the in-class discussions will be an essential component of the grade for each assessment.**