ENGR 695 Advanced Topics in Engineering Mathematics Fall 2023

Lecture Outline for Friday, Sept. 22

1. Example application of *QR* factorization

   Find linear fit to the following data set (seen before):

i	x_i	y_i
1	1.0	1.1
2	2.0	3.2
3	4.0	5.2

   Linear fit: $y = d_0 + d_1 x$, so the "function" matrix F and "data" vector $\mathbf{y}$ are ($M = 3$ and $N = 2$)

$$F = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 4 \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} 1.1 \\ 3.2 \\ 5.2 \end{bmatrix}.$$

   Apply Matlab command `[Q R] = qr(F)`.
   The resulting Q and R matrices are (to four decimal places of accuracy)

$$Q = \begin{bmatrix} -0.5774 & 0.6172 & 0.5345 \\ -0.5774 & 0.1543 & -0.8018 \\ -0.5774 & -0.7715 & 0.2673 \end{bmatrix} \qquad R = \begin{bmatrix} -1.7321 & -4.0415 \\ 0 & -2.1602 \\ 0 & 0 \end{bmatrix}$$

   Verify using *Matlab* that Q is orthogonal.
   To solve overdetermined system: $F\mathbf{c} = \mathbf{y} \rightarrow QR\mathbf{c} = \mathbf{y}$. Let $R\mathbf{c} = \mathbf{z} \rightarrow Q\mathbf{z} = \mathbf{y}$.
   Solve $\mathbf{z} = Q^T\mathbf{y}$ and then $R\mathbf{c} = \mathbf{z}$ for $\mathbf{c}$ using backward substitution.

   Solution should be $\mathbf{c} = \begin{bmatrix} 0.1000 \\ 1.3143 \end{bmatrix}$

2. Many other kinds of factorizations are available for special situations, such as Cholesky and LDL^T (both for symmetric matrices). The Cholesky and LDL^T factorizations are both available in *Matlab*.

3. For more information on the various factorizations, see

   - G. H. Golub and C. F. Van Loan, *Matrix Computations* (4th edition is latest)
   - W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes: The Art of Scientific Computing* (3rd edition is latest)

(*continued on next page*)

4. The greatest factorization of them all, singular value decomposition (SVD)

   a. Very good at handling difficult systems in which the matrix is very close to singular (ill conditioned), which can happen with large data sets, measurement errors, and/or noisy data, to name just a few issues often encountered in real problems.
   b. Recommended over the normal equation for solving difficult overdetermined systems. Main disadvantages are more memory storage (an extra matrix) and sometimes it's slower.
   c. Can also be used for data compression (demo soon).
   d. Using the so-called "economy-sized" or "thin" decomposition, an $M \times N$ matrix A can be expressed in the product form (assuming $M > N$ or $M = N$ for now)

$$A = U\Sigma V^H,$$

   where U is an $M \times N$ column-orthogonal matrix, Σ (sometimes labeled S) is an $N \times N$ diagonal matrix, V is an $N \times N$ orthogonal matrix, and H indicates complex conjugate transpose (V can be complex if A is complex):

$$U = \begin{bmatrix} \uparrow & \uparrow & & \uparrow \\ \mathbf{u}_1 & \mathbf{u}_2 & \cdots & \mathbf{u}_N \\ \downarrow & \downarrow & & \downarrow \end{bmatrix}_{M\times N} \qquad \Sigma = \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & \sigma_N \end{bmatrix}_{N\times N} \qquad V = \begin{bmatrix} \uparrow & \uparrow & & \uparrow \\ \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_N \\ \downarrow & \downarrow & & \downarrow \end{bmatrix}_{N\times N}$$

   where $\mathbf{u}_i^T\mathbf{u}_j = \delta_{ij}$ (orthogonal) and $\mathbf{v}_i^T\mathbf{v}_j = \delta_{ij}$ (orthogonal)

   e. In the full SVD, U is $M \times M$ and Σ is $M \times N$, but parts of U and Σ are not necessary for non-square ($M > N$) systems, hence the "economy-sized" decomposition.
   f. Matlab command (full SVD unless option is added): `[U S V] = svd(A)`
   g. The diagonal elements of Σ are called *singular values*. They are always real and either positive or zero, even if A has complex entries. They can repeat. Thus,

$$\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq \ldots \geq \sigma_N .$$

   Zero singular values, if any, occupy the highest index numbers (i.e., up to and including σ_N)