

The purpose of this project is to illustrate the usefulness of the QR decomposition in two applications: the solution of underdetermined systems and constrained least squares.

Underdetermined systems.

Let A be an $m \times n$ matrix of rank m with $m \leq n$ and consider the linear system

$$Ax = b. \quad (1)$$

If $m > n$, elementary linear algebra considerations show that A has nonzero null vectors v satisfying $Av = 0$. In particular, if x is a solution of (1), so is $x + v$. Thus the underdetermined system (1) does not have a unique solution.

There are many kinds restrictions we can put on the solution x to make it unique.¹ But perhaps the most common one is to require that of all solutions $\|x\|$ is minimal. Here $\|\cdot\|$ is the vector 2-norm. We will now show how to use the QR decomposition to compute such a solution.

Let $U = (U_1 \ U_2)$ be an orthogonal matrix such that

$$U^T A^T = \begin{pmatrix} U_1^T \\ U_2^T \end{pmatrix} A^T = \begin{pmatrix} R \\ 0 \end{pmatrix}.$$

Here R is an $m \times m$ upper triangular matrix, which is necessarily nonsingular. If we set

$$y = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} U_1^T x \\ U_2^T x \end{pmatrix} = U^T x,$$

then

$$Ax = (AU)(U^T x) = (R^T \ 0) \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = R^T y_1 = b.$$

Thus we must have $R^T y_1 = b$ and y_2 can be anything, i.e., whatever the value of y_2 , $x = Uy$ is a solution of (1).

- a. Show that the choice $y_2 = 0$ gives the minimum norm solution. [Hint: Because U is orthogonal, $\|x\| = \|y\|$.]

Answer: There is a 1-1 correspondence between solutions in the y form and in the x form given by $x = Uy$. Because U is orthogonal, the norms of corresponding solutions are the same. But the solution y_* in which $y_2 = 0$ has minimal norm among all other solutions y . It follows that $x_* = Uy_*$ has minimal norm among all other solutions x .

You are to write a Matlab function

¹For example the Matlab backslash operator chooses (don't ask how) $m - n$ components of x and requires that they be zero. This leads to a system of order n for the remaining components.

```
x = MinNormSolve(A, b)
```

that returns the minimum norm solution of (1). It should handle the case where $m = n$ and give an error return if $m > n$. You do not need to check to see if A is of rank m , or equivalently if R is nonsingular. You should take advantage of the fact that if y_2 is zero, you do not need to compute U_2 —only U_1 . (Executing the command

```
>> help qr
```

will give you the skinny on how to compute only U_1 .)

An important problem is how to generate test cases to see if your function is working. Here is one way.

b. Show that if the rows of A are orthonormal, then $x = A^T b$ is the minimal norm solution.

Thus you can generate A using the Matlab function `qr` and for any b you can compare $A^T b$ with the solution from your `MinNormSolve`.

Answer: If A has orthonormal rows, $AA^T = I$ is upper triangular; i.e., we can take $U_1 = A^T$ and $R = I$ as the QR factorization of A . It follows that the equation $R^T y_1 = b$ becomes $y_1 = b$ and $x = U_1 y_1$ becomes $x = A^T b$.

Constrained least squares

Let A be an $m \times n$ matrix of rank n (so that $m \geq n$). We will be concerned with solving the following constrained least squares problem.

$$\begin{aligned} & \text{minimize} && \rho = \|b - Ax\| \\ & \text{subject to} && Cx = d. \end{aligned} \tag{2}$$

Here C is a $k \times n$ matrix with $k \leq n$.

To solve this problem we begin as above by finding U so that

$$U^T C^T = \begin{pmatrix} U_1^T \\ U_2^T \end{pmatrix} C^T = \begin{pmatrix} R \\ 0 \end{pmatrix}.$$

and setting

$$y = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} U_1^T x \\ U_2^T x \end{pmatrix} = U^T x.$$

Then as above $R^T y_1 = d$ and y_2 is undetermined.

We now write

$$\begin{aligned}\|b - Ax\| &= \left\| b - A(U_1 \ U_2) \begin{pmatrix} U_1^T \\ U_2^T \end{pmatrix} x \right\| \\ &= \left\| b - (AU_1 \ AU_2) \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \right\| \\ &= \|(b - B_1y_1) - B_2y_2\|,\end{aligned}$$

where $B_i = AU_i$ ($i = 1, 2$). Since y_1 is known, we can choose y_2 to

$$\text{minimize } \|(b - B_1y_1) - B_2y_2\|, \quad (3)$$

which is an unconstrained least squares problem. After y_1 and y_2 have been determined, we calculate $x = Uy$.

To solve the least squares problem (3) one can use the fact that if

$$(B_2 \ b - B_1y_1) = (Q, \ q) \begin{pmatrix} S & s \\ 0 & \sigma \end{pmatrix} \quad (4)$$

is the QR decomposition of $(B_2 \ b - B_1y_1)$, then $y_2 = S^{-1}s$ and $|\sigma|$ is the norm at the minimum.

Write a Matlab function

$$[\mathbf{x}, \mathbf{rho}] = \text{ConstrLsq}(\mathbf{A}, \mathbf{b}, \mathbf{C}, \mathbf{d})$$

that solves the constrained least squares problem (2). Here \mathbf{A} , \mathbf{b} , \mathbf{C} , \mathbf{d} are as above and \mathbf{x} and \mathbf{rho} are the values of \mathbf{x} and \mathbf{rho} at the minimum. The function should treat the case $k = n$ and should give an error return for inconsistencies among the matrices \mathbf{A} and \mathbf{C} . Do not compute \mathbf{rho} in the form $\|b - Ax\|$. Rather use (4) as described above.

Test cases with known answers are harder to find for this problem. However, there is only one x that produces the minimum value of ρ . Consequently, if you have followed instructions and used (4) to compute ρ you can compare the value with $\|b - Ax\|$. If they are different, you have problems. It is remotely possible that the two could agree when x is not the solution. But if you use `randn` to generate several random test cases and $\|b - Ax\| = \rho$ for all of them, you can be reasonably confident that your algorithm is working. You should also check that $Cx = d$ to working accuracy.

A couple of days before the project is due, you will be furnished with a script to run with your functions. Turn in your functions, your answers to problems a and b, and the results of running the script.