Overdetermined Linear Systems

Contents

Introduction

Opening Example: Polynomial Approximation

Introduction

Problem: Fitting Functions to Data

Given data points $\{(x_i, y_i) \mid i \in \mathbb{N}[1, m]\}$, pick a form for the "fitting" function $f(x)$ and minimize its total error in representing the data.

- With real-world data, interpolation is often not the best method.
- Instead of finding functions lying exactly on given data points, we look for ones which are "close" to them.
- In the most general terms, the fitting function takes the form

$$f(x) = c_1 f_1(x) + \cdots + c_n f_n(x),$$

where $f_1, \ldots, f_n$ are known functions while $c_1, \ldots, c_n$ are to be determined.

Linear Least Squares Approximation

In this discussion:

- use a polynomial fitting function $p(x) = c_1 + c_2 x + \cdots + c_n x^{n-1}$ with $n < m$;
- minimize the 2-norm of the error $r_i = y_i - p(x_i)$:

$$\|\mathbf{r}\|_2 = \sqrt{\sum_{i=1}^{m} r_i^2} = \sqrt{\sum_{i=1}^{m} \big(y_i - p(x_i)\big)^2}.$$

Since the fitting function is linear in unknown coefficients and the 2-norm is minimized, this method of approximation is called the **linear least squares (LLS) approximation**.

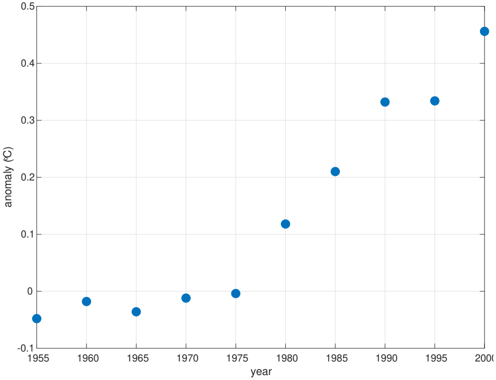
Example: Temperature Anomaly

Below are 5-year averages of the worldwide temperature anomaly as compared to the 1951-1980 average (source: NASA).

Year	Anomaly ($^\circ C$)
1955	-0.0480
1960	-0.0180
1965	-0.0360
1970	-0.0120
1975	-0.0040
1980	0.1180
1985	0.2100
1990	0.3320
1995	0.3340
2000	0.4560

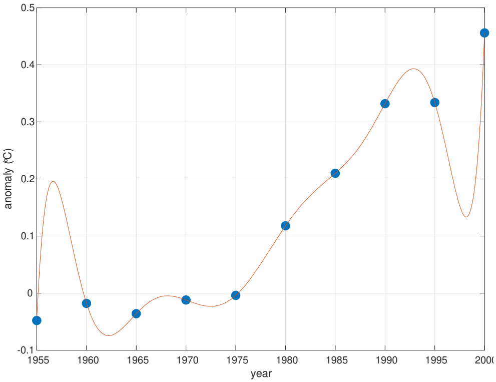
Example: Import and Plot Data

```
t = (1955:5:2000)';
y = [-0.0480; -0.0180;
     -0.0360; -0.0120;
     -0.0040;  0.1180;
      0.2100;  0.3320;
      0.3340;  0.4560];
plot(t, y, '.')
```



Example: Interpolation

```
t = (t-1950)/10;
n = length(t);
V = t.^(0:n-1);
c = V\y;
p = @(x) polyval(flip(c),
      (x-1950)/10);
hold on
fplot(p, [1955 2000])
```



Fitting by a Straight Line

Suppose that we are fitting data to a linear polynomial: $p(x) = c_1 + c_2 x$.

- If it were to pass through all data points:

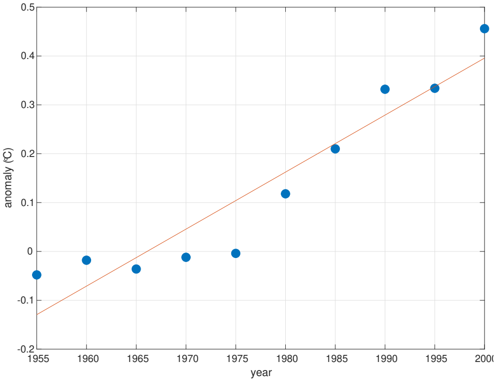
$$\begin{cases} y_1 = p(x_1) = c_1 + c_2 x_1 \\ y_2 = p(x_2) = c_1 + c_2 x_2 \\ \quad \vdots \qquad\quad \vdots \qquad \vdots \\ y_m = p(x_m) = c_1 + c_2 x_m \end{cases} \xrightarrow{\text{matrix equation}} \underbrace{\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}}_{\mathbf{y}} = \underbrace{\begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_m \end{bmatrix}}_{V} \underbrace{\begin{bmatrix} c_1 \\ c_2 \end{bmatrix}}_{\mathbf{c}}$$

- The above is unsolvable; instead, find $\mathbf{c}$ which makes the *residual* $\mathbf{r} = \mathbf{y} - V\mathbf{c}$ “as small as possible” in the sense of vector 2-norm.
- **Notation:** $\mathbf{y}$ “=” $V\mathbf{c}$

MATLAB Implementation

Revisiting the temperature anomaly example again:

```
year = (1955:5:2000)';
t = year - 1955;
V = t.^(0:1);
c = V\y;
p = @(x) polyval(flip(c),
      x-1955);
plot(year, y, '.')
hold on
fplot(p, [1955, 2000])
```



Fitting by a General Polynomial

In general, when fitting data to a polynomial

$$p(x) = c_1 + c_2 x + c_3 x^2 + \cdots + c_n x^{n-1},$$

we need to solve

$$\underbrace{\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}}_{\mathbf{y}} \text{“=”} \underbrace{\begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \cdots & x_2^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_m & x_m^2 & \cdots & x_m^{n-1} \end{bmatrix}}_{V} \underbrace{\begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix}}_{\mathbf{c}}.$$

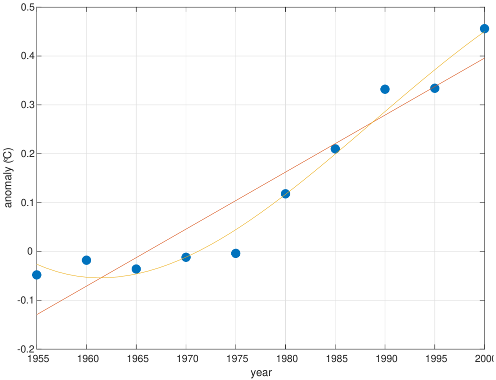
- The solution $\mathbf{c}$ of $\mathbf{y}$ “=” $V\mathbf{c}$ turns out to be the solution of the <u>*normal equation*</u>

$$V^{\mathrm{T}} V \mathbf{c} = V^{\mathrm{T}} \mathbf{y}\,.$$

MATLAB Implementation

Revisiting the temperature anomaly example again:

```
V = t.^(0:3);
c = V\y;
q = @(x) polyval(flip(c),
      x-1955);
hold on
fplot(q, [1955 2000])
```



Backslash Again

The Versatile Backslash

In MATLAB, the generic linear equation $A\mathbf{x} = \mathbf{b}$ is solved by `x = A\b`.

- When A is a square matrix, Gaussian elimination is used.
- When A is NOT a square matrix, the normal equation $A^{\mathrm{T}}A\mathbf{x} = A^{\mathrm{T}}\mathbf{b}$ is solved instead.

- As long as $A \in \mathbb{R}^{m\times n}$ where $m \geqslant n$ has rank n, the square matrix $A^{\mathrm{T}}A$ is nonsingular. (unique solution)
- Though $A^{\mathrm{T}}A$ is a square matrix, MATLAB does not use Gaussian elimination to solve the normal equation.
- Rather, a faster and more accurate algorithm is used.

The Normal Equations

LLS and Normal Equation

Big Question: How is the least square solution $\mathbf{x}$ to $A\mathbf{x}$ "=" $\mathbf{b}$ equivalent to the solution of the normal equation $A^{\mathrm{T}}A\mathbf{x} = A^{\mathrm{T}}\mathbf{b}$?

Theorem (Normal Equation)

Let $A \in \mathbb{R}^{m\times n}$ with $m \geqslant n$. If $\mathbf{x} \in \mathbb{R}^{n}$ satisfies $A^{\mathrm{T}}A\mathbf{x} = A^{\mathrm{T}}\mathbf{b}$, then $\mathbf{x}$ solves the LLS problems, *i.e.*, $\mathbf{x}$ minimizes $\|\mathbf{b} - A\mathbf{x}\|_2$.

Proof of the Theorem

Idea of Proof[1]**.** Enough show to that $\left\|\mathbf{b} - A(\mathbf{x} + \mathbf{y})\right\|_2 \geqslant \left\|\mathbf{b} - A\mathbf{x}\right\|_2$ for any $\mathbf{y} \in \mathbb{R}^n$.

- Useful algebra:

$$(\mathbf{u} + \mathbf{v})^{\mathrm{T}}(\mathbf{u} + \mathbf{v}) = \mathbf{u}^{\mathrm{T}}\mathbf{u} + \mathbf{u}^{\mathrm{T}}\mathbf{v} + \mathbf{v}^{\mathrm{T}}\mathbf{v} + \mathbf{v}^{\mathrm{T}}\mathbf{v} = \mathbf{u}^{\mathrm{T}}\mathbf{u} + 2\mathbf{v}^{\mathrm{T}}\mathbf{u} + \mathbf{v}^{\mathrm{T}}\mathbf{v}.$$

- **Exercise:** Prove it.

[1] Alternately, one can derive the normal equation using calculus. See Appendix.

Appendix: Derivation of Normal Equation

Derivation of Normal Equation

Consider $A\mathbf{x}$ "=" $\mathbf{b}$ where $A \in \mathbb{R}^{m\times n}$ where $m \geqslant n$.

- **Requirement:** minimize the 2-norm of the residual $\mathbf{r} = \mathbf{b} - A\mathbf{x}$:

$$g(x_1, x_2, \ldots, x_n) := \|\mathbf{r}\|_2^2 = \sum_{i=1}^{m} \left(b_i - \sum_{j=1}^{n} a_{ij} x_j \right)^2 .$$

- **Strategy:** using calculus, find the minimum by setting

$$\mathbf{0} = \nabla g(x_1, x_2, \ldots, x_n)$$

which yields n equations in n unknowns $x_1, x_2, \ldots, x_n$.

Derivation of Normal Equation (cont')

Noting that $\partial x_j / \partial x_k = \delta_{j,k}$, the n equations $\partial g / \partial x_k = 0$ are written out as

$$0 = \sum_{i=1}^{m} 2\Big(b_i - \sum_{j=1}^{n} a_{ij} x_j\Big)(-a_{ik}), \qquad \text{for } k \in \mathbb{N}[1,n]\,,$$

which can be rearranged into

$$\sum_{i=1}^{m} a_{ik} b_i = \sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij} a_{ik} x_j, \qquad \text{for } k \in \mathbb{N}[1,n]\,.$$

One can see that the two sides correspond to the k^{th} elements of $A^{\mathrm{T}}\mathbf{b}$ and $A^{\mathrm{T}}A\mathbf{x}$ respectively:

$$A^{\mathrm{T}}A\mathbf{x} = A^{\mathrm{T}}\mathbf{b}\,,$$

showing the desired equivalence.

QR Factorization

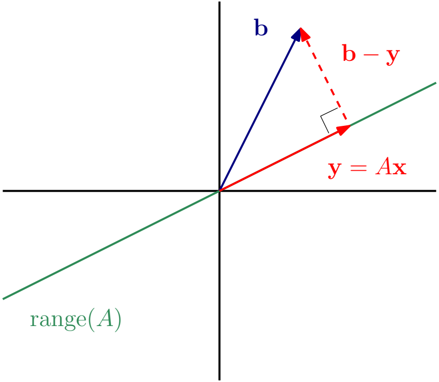
Preliminary: Orthogonality

Normal Equation Revisited

Alternate perspective on the "normal equation":

$$A^{\mathrm{T}}(\mathbf{b} - A\mathbf{x}) = \mathbf{0} \quad \Longleftrightarrow \quad \mathbf{z}^{\mathrm{T}}(\underbrace{\mathbf{b} - A\mathbf{x}}_{\text{residual } = \mathbf{r}}) = 0 \quad \text{for all } \mathbf{z} \in \mathcal{R}(A),$$

i.e., $\mathbf{x}$ solves the normal equation if and only if the residual is orthogonal to the range of A.



Orthogonal Vectors

Recall that the angle θ between two vectors $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$ satisfies

$$\cos(\theta) = \frac{\mathbf{u}^{\mathrm{T}}\mathbf{v}}{\|\mathbf{u}\|_2 \, \|\mathbf{v}\|_2}.$$

Definition 1

- Two vectors $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$ are **orthogonal** if $\mathbf{u}^{\mathrm{T}}\mathbf{v} = 0$.
- Vectors $\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_k \in \mathbb{R}^n$ are **orthogonal** if $\mathbf{q}_i^{\mathrm{T}}\mathbf{q}_j = 0$ for all $i \neq j$.
- Vectors $\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_k \in \mathbb{R}^n$ are **orthonormal** if $\mathbf{q}_i^{\mathrm{T}}\mathbf{q}_j = \delta_{i,j}$.

Notation. (Kronecker delta function)

$$\delta_{i,j} = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}$$

Matrices with Orthogonal Columns

Let $Q = \left[\ \mathbf{q}_1 \mid \mathbf{q}_2 \mid \cdots \mid \mathbf{q}_k\ \right] \in \mathbb{R}^{n\times k}$. Note that

$$Q^{\mathrm{T}}Q = \begin{bmatrix} \mathbf{q}_1^{\mathrm{T}} \\ \hline \mathbf{q}_2^{\mathrm{T}} \\ \hline \vdots \\ \hline \mathbf{q}_k^{\mathrm{T}} \end{bmatrix} \left[\begin{array}{c|c|c|c} \mathbf{q}_1 & \mathbf{q}_2 & \cdots & \mathbf{q}_k \end{array}\right] = \begin{bmatrix} \mathbf{q}_1^{\mathrm{T}}\mathbf{q}_1 & \mathbf{q}_1^{\mathrm{T}}\mathbf{q}_2 & \cdots & \mathbf{q}_1^{\mathrm{T}}\mathbf{q}_k \\ \mathbf{q}_2^{\mathrm{T}}\mathbf{q}_1 & \mathbf{q}_2^{\mathrm{T}}\mathbf{q}_2 & \cdots & \mathbf{q}_2^{\mathrm{T}}\mathbf{q}_k \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{q}_k^{\mathrm{T}}\mathbf{q}_1 & \mathbf{q}_k^{\mathrm{T}}\mathbf{q}_2 & \cdots & \mathbf{q}_k^{\mathrm{T}}\mathbf{q}_k \end{bmatrix}.$$

Therefore,

- $\mathbf{q}_1, \ldots, \mathbf{q}_k$ are orthogonal. $\iff$ $Q^{\mathrm{T}}Q$ is a $k \times k$ diagonal matrix.
- $\mathbf{q}_1, \ldots, \mathbf{q}_k$ are orthonormal. $\iff$ $Q^{\mathrm{T}}Q$ is the $k \times k$ identity matrix.

Matrices with Orthonormal Columns

Theorem 2

Let $Q = \left[\ \mathbf{q}_1 \mid \mathbf{q}_2 \mid \cdots \mid \mathbf{q}_k\ \right] \in \mathbb{R}^{n \times k}$ *and suppose that* $\mathbf{q}_1, \ldots, \mathbf{q}_k$ *are orthonormal. Then*

1. $Q^{\mathrm{T}} Q = I \in \mathbb{R}^{k \times k}$;
2. $\|Q\mathbf{x}\|_2 = \|\mathbf{x}\|_2$ *for all* $\mathbf{x} \in \mathbb{R}^k$;
3. $\|Q\|_2 = 1$.

Orthogonal Matrices

Definition 3

We say that $Q \in \mathbb{R}^{n\times n}$ is an **orthogonal matrix** if $Q^{\mathrm{T}}Q = I \in \mathbb{R}^{n\times n}$.

- A square matrix with orthogonal columns is not, in general, an orthogonal matrix!

Properties of Orthogonal Matrices

Theorem 4

Let $Q \in \mathbb{R}^{n \times n}$ be orthogonal. Then

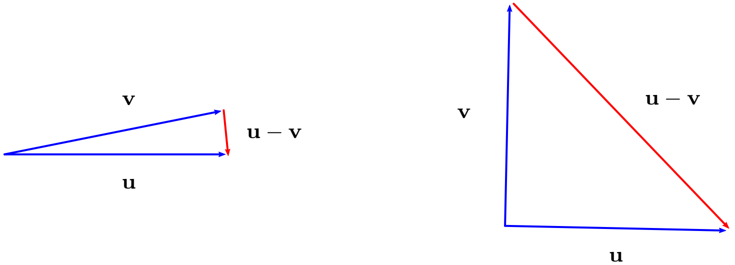
1. $Q^{-1} = Q^{\mathrm{T}}$;
2. Q^{T} *is also an orthogonal matrix;*
3. $\kappa_2(Q) = 1$;
4. *For any* $A \in \mathbb{R}^{n \times n}$, $\|AQ\|_2 = \|A\|_2$;
5. *if* $P \in \mathbb{R}^{n \times n}$ *is another orthogonal matrix, then* PQ *is also orthogonal.*

Why Do We Like Orthogonal Vectors?

- If $\mathbf{u}$ and $\mathbf{v}$ are orthogonal, then

$$\|\mathbf{u} \pm \mathbf{v}\|_2^2 =$$

- Without orthogonality, it is possible that $\|\mathbf{u} - \mathbf{v}\|_2$ is much smaller than $\|\mathbf{u}\|_2$ and $\|\mathbf{v}\|_2$.
- The addition and subtraction of orthogonal vectors are guaranteed to be well-conditioned.



QR Factorization

The QR Factorization

The following matrix factorization plays a role in solving linear least squares problems similar to that of LU factorization in solving linear systems.

Theorem 5

Let $A \in \mathbb{R}^{m\times n}$ where $m \geqslant n$. Then $A = QR$ where $Q \in \mathbb{R}^{m\times m}$ is orthogonal and $R \in \mathbb{R}^{m\times n}$ is upper triangular.

$$\underbrace{\left[\begin{array}{c|c|c|c} \mathbf{a}_1 & \mathbf{a}_2 & \cdots & \mathbf{a}_n \end{array}\right]}_{A} = \underbrace{\left[\begin{array}{c|c|c|c} \mathbf{q}_1 & \mathbf{q}_2 & \cdots & \mathbf{q}_m \end{array}\right]}_{Q} \underbrace{\begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ 0 & r_{22} & \cdots & r_{2n} \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & r_{nn} \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}}_{R}$$

Thick VS Thin QR Factorization

- Here is the QR factorization again.

$$A = \underbrace{\left[\begin{array}{c|c|c|c} & & & \\ \mathbf{q}_1 & \mathbf{q}_2 & \cdots & \mathbf{q}_m \\ & & & \end{array}\right]}_{Q} \underbrace{\begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ 0 & r_{22} & \cdots & r_{2n} \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & r_{nn} \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}}_{R} \qquad \text{(thick)}$$

- When m is much larger than n, it is much more efficient to use the *thin* or *compressed* QR factorization.

$$A = \underbrace{\left[\begin{array}{c|c|c|c} & & & \\ \mathbf{q}_1 & \mathbf{q}_2 & \cdots & \mathbf{q}_n \\ & & & \end{array}\right]}_{\hat{Q}} \underbrace{\begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ 0 & r_{22} & \cdots & r_{2n} \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & r_{nn} \end{bmatrix}}_{\hat{R}} \qquad \text{(thin)}$$

QR Factorization in MATLAB

Either type of QR factorization is computed by `qr` command.

- Thick/Full QR factorization

```
[Q, R] = qr(A)
```

- Thin/Compressed QR factorization

```
[Q, R] = qr(A, 0)
```

Test the orthogonality of `Q` by calculating the norm of $Q^{\mathrm{T}}Q - I$ where I is the identity matrix with *suitable* dimensions.

```
norm(Q'*Q - eye(m))        % full QR
norm(Q'*Q - eye(n))        % thin QR
```

Least Squares and QR Factorization

Substitute the thin factorization $A = \widehat{Q}\widehat{R}$ into the normal equation $A^{\mathrm{T}}A\mathbf{x} = A^{\mathrm{T}}\mathbf{b}$ and simplify.

Least Squares and QR Factorization (cont')

Summary: Algorithm for LLS Approximation

If A has rank n, the normal equation $A^{\mathrm{T}}A\mathbf{x} = A^{\mathrm{T}}\mathbf{b}$ is consistent and is equivalent to $\widehat{R}\mathbf{x} = \widehat{Q}^{\mathrm{T}}\mathbf{b}$.

1. Factor $A = \widehat{Q}\widehat{R}$.
2. Let $\mathbf{z} = \widehat{Q}^{\mathrm{T}}\mathbf{b}$.
3. Solve $\widehat{R}\mathbf{x} = \mathbf{z}$ for $\mathbf{x}$ using backward substitution.

Least Squares and QR Factorization (cont')

```
function x = lsqrfact(A,b)
% LSQRFACT x = lsqrfact(A,b)
% Sove linear least squares by QR factorization
% Input:
%   A    coefficient matrix (m-by-n, m>n)
%   b    right-hand side (m-by-1)
% Output:
%   x    minimizer of || b - Ax || (2-norm)
    [Q,R] = qr(A,0);             % thin QR fact.
    z = Q'*b;
    x = backsub(R,c);
end
```

Appendix: Gram-Schmidt Orthogonalization

The Gram–Schmidt Procedure

Problem: Orthogonalization

Given $\mathbf{a}_1, \ldots, \mathbf{a}_n \in \mathbb{R}^m$, construct orthonormal vectors $\mathbf{q}_1, \ldots, \mathbf{q}_n \in \mathbb{R}^m$ such that

$$\operatorname{span}\{\mathbf{a}_1, \ldots, \mathbf{a}_k\} = \operatorname{span}\{\mathbf{q}_1, \ldots, \mathbf{q}_k\}, \quad \text{for any } k \in \mathbb{N}[1, n].$$

- **Strategy.** At the jth step, find a unit vector $\mathbf{q}_j \in \operatorname{span}\{\mathbf{a}_1, \ldots, \mathbf{a}_j\}$ that is orthogonal to $\mathbf{q}_1, \ldots, \mathbf{q}_{j-1}$.
- **Key Observation.** The vector $\mathbf{v}_j$ defined by

$$\mathbf{v}_j = \mathbf{a}_j - \left(\mathbf{q}_1^{\mathrm{T}} \mathbf{a}_j\right) \mathbf{q}_1 - \left(\mathbf{q}_2^{\mathrm{T}} \mathbf{a}_j\right) \mathbf{q}_2 - \cdots - \left(\mathbf{q}_{j-1}^{\mathrm{T}} \mathbf{a}_j\right) \mathbf{q}_{j-1}$$

  satisfies the required properties.

GS Algorithm

The Gram–Schmidt iteration is outlined below:

$$
\begin{aligned}
\mathbf{q}_1 &= \frac{\mathbf{a}_1}{r_{11}}, \\
\mathbf{q}_2 &= \frac{\mathbf{a}_2 - r_{12}\mathbf{q}_1}{r_{22}}, \\
\mathbf{q}_3 &= \frac{\mathbf{a}_3 - r_{13}\mathbf{q}_1 - r_{23}\mathbf{q}_2}{r_{33}}, \\
&\vdots \\
\mathbf{q}_n &= \frac{\mathbf{a}_n - \sum_{i=1}^{n-1} r_{in}\mathbf{q}_i}{r_{nn}},
\end{aligned}
$$

where

$$
r_{ij} = \begin{cases} \mathbf{q}_i^{\mathrm{T}}\mathbf{a}_j, & \text{if } i \neq j \\ \pm\left\|\mathbf{a}_j - \sum_{k=1}^{j-1} r_{kj}\mathbf{q}_k\right\|_2, & \text{if } i = j \end{cases}.
$$

GS Procedure and Thin QR Factorization

- The GS algorithm, written as a matrix equation, yields a **thin QR factorization**:

$$A = \underbrace{\left[\begin{array}{c|c|c} \mathbf{a}_1 & \cdots & \mathbf{a}_n \end{array}\right]}_{A} = \underbrace{\left[\begin{array}{c|c|c} \mathbf{q}_1 & \cdots & \mathbf{q}_n \end{array}\right]}_{\hat{Q}} \underbrace{\begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ 0 & r_{22} & \cdots & r_{2n} \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & r_{nn} \end{bmatrix}}_{\hat{R}} = \widehat{Q}\widehat{R}$$

- While it is an important tool for theoretical work, the GS algorithm is numerically unstable.

QR Algorithm

Revisiting Least Squares

Moore-Penrose Pseudoinverse

Let $A \in \mathbb{R}^{m \times n}$ with $m \geqslant n$ and suppose that columns of A are linearly independent.

- The least square problem $A\mathbf{x}$ "=" $\mathbf{b}$ is equivalent to the normal equation $A^{\mathrm{T}}A\mathbf{x} = A^{\mathrm{T}}\mathbf{b}$, which is a square matrix equation.
- The solution can be written as
$$\mathbf{x} = \left(A^{\mathrm{T}}A\right)^{-1}A^{\mathrm{T}}\mathbf{b}.$$
- The matrix
$$A^{+} = \left(A^{\mathrm{T}}A\right)^{-1}A^{\mathrm{T}} \in \mathbb{R}^{n \times m},$$
is called the **(Moore-Penrose) pseudoinverse**.
- MATLAB's backslash is mathematically equivalent to left-multiplication by the inverse or pseudoinverse of a matrix.
- MATLAB's `pinv` calculates the pseudoinverse, but it is rarely used in practice, just as `inv`.

Moore-Penrose Pseudoinverse (cont')

- A^+ can be calculated by using the thin QR factorization[2] $A = \widehat{Q}\widehat{R}$.

$$A^+ = \widehat{R}^{-1}\widehat{Q}^{\mathrm{T}}.$$

[2] It can be done using the thick QR factorization as seen on p. 1624 of the text.

Least Squares and QR Factorization

Substitute the thin factorization $A = \widehat{Q}\widehat{R}$ into the normal equation $A^{\mathrm{T}} A\mathbf{x} = A^{\mathrm{T}}\mathbf{b}$ and simplify.

Summary: Algorithm for LLS Approximation

If A has rank n, the normal equation $A^{\mathrm{T}}A\mathbf{x} = A^{\mathrm{T}}\mathbf{b}$ is consistent and is equivalent to $\widehat{R}\mathbf{x} = \widehat{Q}^{\mathrm{T}}\mathbf{b}$.

1. Factor $A = \widehat{Q}\widehat{R}$.
2. Let $\mathbf{z} = \widehat{Q}^{\mathrm{T}}\mathbf{b}$.
3. Solve $\widehat{R}\mathbf{x} = \mathbf{z}$ for $\mathbf{x}$ using backward substitution.

Least Squares and QR Factorization (cont')

```
function x = lsqrfact(A,b)
% LSQRFACT x = lsqrfact(A,b)
% Sove linear least squares by QR factorization
% Input:
%   A    coefficient matrix (m-by-n, m>n)
%   b    right-hand side (m-by-1)
% Output:
%   x    minimizer of || b - Ax || (2-norm)
    [Q,R] = qr(A,0);                % thin QR fact.
    z = Q'*b;
    x = backsub(R,c);
end
```

Householder Transformation and QR Algorithm

Motivation

Problem

Given $\mathbf{z} \in \mathbb{R}^m$, find an orthogonal matrix $H \in \mathbb{R}^{m \times m}$ such that $H\mathbf{z}$ is nonzero only in the first element.

- Since orthogonal matrices preserve the 2-norm, H must satisfy

$$H\mathbf{z} = \begin{bmatrix} \pm \|\mathbf{z}\|_2 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \pm \|\mathbf{z}\|_2 \, \mathbf{e}_1.$$

- The **Householder transformation matrix** H defined by

$$H = I - 2\frac{\mathbf{v}\mathbf{v}^{\mathrm{T}}}{\mathbf{v}^{\mathrm{T}}\mathbf{v}}, \quad \text{where } \mathbf{v} = \pm \|\mathbf{z}\|_2 \, \mathbf{e}_1 - \mathbf{z},$$

solves the problem. See Theorem 6 on the next slide.

Properties of Householder Transformation

Theorem 6

Let $\mathbf{v} = \|\mathbf{z}\|_2 \, \mathbf{e}_1 - \mathbf{z}$ *and let* H *be the Householder transformation defined by*

$$H = I - 2\frac{\mathbf{v}\mathbf{v}^{\mathrm{T}}}{\mathbf{v}^{\mathrm{T}}\mathbf{v}}.$$

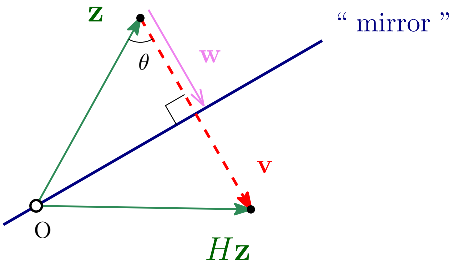
Then

1. H *is symmetric;*
2. H *is orthogonal;*
3. $H\mathbf{z} = \|\mathbf{z}\|_2 \, \mathbf{e}_1$.

- H is invariant under scaling of $\mathbf{v}$.
- If $\|\mathbf{v}\|_2 = 1$, then $H = I - \mathbf{v}\mathbf{v}^{\mathrm{T}}$.

Geometry Behind Householder Transformation (cont')

The Householder transformation matrix H can be thought of as a *reflector*[3].



[3] See Supplementary 1 for review on projection and reflection operators

Factorization Algorithm

- The Gram-Schmidt orthogonalization (thin QR factorization) is unstable in floating-point calculations.
- **Stable alternative:** Find orthogonal matrices $H_1, H_2, \ldots, H_n$ so that

$$\underbrace{H_n H_{n-1} \cdots H_2 H_1}_{=:Q^{\mathrm{T}}} A = R\,.$$

  introducing zeros one column at a time below diagonal terms.
- As a product of orthogonal matrices, Q^{T} is also orthogonal and so $(Q^{\mathrm{T}})^{-1} = Q$. Therefore,

$$A = QR\,.$$

MATLAB Demonstration Code MYQR

```
function [Q, R] = myqr(A)
  [m, n] = size(A);
  A0 = A;
  Q = eye(m);
  for j = 1:min(m,n)
      Aj = A(j:m, j:n);
      z = Aj(:, 1);
      v = z + sign0(z(1))*norm(z)*eye(length(z), 1);
      Hj = eye(length(v)) - 2/(v'*v) * v*v';
      Aj = Hj*Aj;
      H = eye(m);
      H(j:m, j:m) = Hj;
      Q = Q*H;
      A(j:m, j:n) = Aj;
  end
  R = A;
end
```

MATLAB Demonstration Code MYQR (cont')

(continued from the previous page)

```
% local function
function sign0(x)
  y = ones(size(x));
  y(x < 0) = -1;
end
```

- The MATLAB command `qr` works similar to, but more efficiently than, this.
- The function finds the factorization in $\sim (2mn^2 - n^3/3)$ flops asymptotically.

Supplementary 1: Projection and Reflection

Projection and Reflection Operators

Let $\mathbf{u}, \mathbf{v} \in \mathbb{R}^m$ be nonzero vectors.

- Projection of $\mathbf{u}$ onto $\langle \mathbf{v} \rangle = \text{span}(\mathbf{v})$:

$$\frac{\mathbf{v}^{\mathrm{T}}\mathbf{u}}{\mathbf{v}^{\mathrm{T}}\mathbf{v}}\mathbf{v} = \underbrace{\left(\frac{\mathbf{v}\mathbf{v}^{\mathrm{T}}}{\mathbf{v}^{\mathrm{T}}\mathbf{v}}\right)}_{=:P}\mathbf{u} =: P\mathbf{u}.$$

- Projection of $\mathbf{u}$ onto $\langle \mathbf{v} \rangle^{\perp}$, the orthogonal complement of $\langle \mathbf{v} \rangle$:

$$\mathbf{u} - \frac{\mathbf{v}^{\mathrm{T}}\mathbf{u}}{\mathbf{v}^{\mathrm{T}}\mathbf{v}}\mathbf{v} = \left(I - \frac{\mathbf{v}\mathbf{v}^{\mathrm{T}}}{\mathbf{v}^{\mathrm{T}}\mathbf{v}}\right)\mathbf{u} =: (I - P)\mathbf{u}.$$

- Reflection of $\mathbf{u}$ across $\langle \mathbf{v} \rangle^{\perp}$:

$$\mathbf{u} - 2\frac{\mathbf{v}^{\mathrm{T}}\mathbf{u}}{\mathbf{v}^{\mathrm{T}}\mathbf{v}}\mathbf{v} = \left(I - 2\frac{\mathbf{v}\mathbf{v}^{\mathrm{T}}}{\mathbf{v}^{\mathrm{T}}\mathbf{v}}\right)\mathbf{u} =: (I - 2P)\mathbf{u}.$$

Projection and Reflection Operators (cont')

Summary: for given $\mathbf{v} \in \mathbb{R}^m$, a nonzero vector, let

$$P = \frac{\mathbf{v}\mathbf{v}^{\mathrm{T}}}{\mathbf{v}^{\mathrm{T}}\mathbf{v}} \in \mathbb{R}^{m \times m}.$$

Then the following matrices carry out geometric transformations

- Projection onto $\langle \mathbf{v} \rangle$: P
- Projection onto $\langle \mathbf{v} \rangle$: $I - P$
- Reflection across $\langle \mathbf{v} \rangle^{\perp}$: $I - 2P$

Note. If $\mathbf{v}$ were a unit vector, the definition of P simplifies to $P = \mathbf{v}\mathbf{v}^{\mathrm{T}}$.

Supplementary 2: Conditioning and Stability

Analytical Properties of Pseudoinverse

The matrix $A^{\mathrm{T}}A$ appearing in the definition of A^{+} satisfies the following properties.

Theorem 7

For any $A \in \mathbb{R}^{m \times n}$ with $m \geqslant n$, the following are true:

1. *$A^{\mathrm{T}}A$ is symmetric.*
2. *$A^{\mathrm{T}}A$ is singular if and only if rank$(A) < n$.*
3. *If $A^{\mathrm{T}}A$ is nonsingular, then it is positive definite.*

A symmetric positive definite (SPD) matrix S such as $A^{\mathrm{T}}A$ permits so-called the **Cholesky factorization**

$$S = R^{\mathrm{T}}R$$

where R is an upper triangular matrix.

Least Squares Using Normal Equation

One can solve the LLS problem $A\mathbf{x}$ "=" $\mathbf{b}$ by solving the normal equation $A^{\mathrm{T}}A\mathbf{x} = A^{\mathrm{T}}\mathbf{b}$ directly as below.

1. Compute $N = A^{\mathrm{T}}A$.
2. Compute $\mathbf{z} = A^{\mathrm{T}}\mathbf{b}$.
3. Solve the square linear system $N\mathbf{x} = \mathbf{z}$ for $\mathbf{x}$.

Step 3 is done using `chol` which implements the Cholesky factorization.

MATLAB Implementarion.

```
N = A'*A;
z = A'*b;
R = chol(N);
w = forelim(R',z);     % solve R'w = z
x = backsub(R,w);      % solve R x = w
```

Conditioning of Normal Equations

- Recall that the condition number of solving a square linear system $A\mathbf{x} = \mathbf{b}$ is $\kappa(A) = \|A\| \left\|A^{-1}\right\|$.
- Provided that the residual norm at the least square solution is relatively small, the conditioning of LLS problem is similar:

$$\kappa(A) = \|A\| \left\|A^{+}\right\|.$$

- If A is rank-deficient (columns are linearly dependent), then $\kappa(A) = \infty$.
- If an LLS problem is solved solving the normal equation, it can be shown that the condition number is

$$\kappa(A^{\mathrm{T}}A) = \kappa(A)^2.$$

Which Reflector Is Better?

- Recall:

$$H = I - 2\frac{\mathbf{v}\mathbf{v}^{\mathrm{T}}}{\mathbf{v}^{\mathrm{T}}\mathbf{v}}, \quad \text{where } \mathbf{v} = \pm \|\mathbf{z}\|_2 \, \mathbf{e}_1 - \mathbf{z},$$

- In `myqr.m`, the statement

```
v = z + sign0(z(1))*norm(z)*eye(length(z), 1);
```

defines $\mathbf{v}$ slightly differently[4], namely,

$$\mathbf{v} = \mathbf{z} \pm \|\mathbf{z}\|_2 \, \mathbf{e}_1.$$

[4]This does not cause any difference since H is invariant under scaling of $\mathbf{v}$; see p. 50

Which Reflector Is Better? (cont')

The sign of $\pm \|\mathbf{z}\|_2$ is chosen so as to avoid possible catastrophic cancellation in forming $\mathbf{v}$:

$$\mathbf{v} = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_m \end{bmatrix} + \begin{bmatrix} \pm \|\mathbf{z}\|_2 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} z_1 \pm \|\mathbf{z}\|_2 \\ z_2 \\ \vdots \\ z_m \end{bmatrix}$$

Subtractive cancellation may arise when $z_1 \approx \pm \|\mathbf{z}\|_2$.

- if $z_1 > 0$, use $z_1 + \|\mathbf{z}\|_2$;
- if $z_1 < 0$, use $z_1 - \|\mathbf{z}\|_2$;
- if $z_1 = 0$, either works.

For numerical stability, it is desirable to reflect $\mathbf{z}$ to the vector $s \|\mathbf{z}\|_2 \mathbf{e}_1$ that is not too close to $\mathbf{z}$ itself. (Trefethen & Bau)