

Outline

- 1 Least Squares Data Fitting
- 2 Existence, Uniqueness, and Conditioning
- 3 Solving Linear Least Squares Problems



Method of Least Squares

- Measurement errors are inevitable in observational and experimental sciences
- Errors can be smoothed out by averaging over many cases, i.e., taking more measurements than are strictly necessary to determine parameters of system
- Resulting system is *overdetermined*, so usually there is no exact solution
- In effect, higher dimensional data are projected into lower dimensional space to suppress irrelevant detail
- Such projection is most conveniently accomplished by method of *least squares*



Linear Least Squares

- For linear problems, we obtain *overdetermined* linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$, with $m \times n$ matrix \mathbf{A} , $m > n$
- System is better written $\mathbf{A}\mathbf{x} \cong \mathbf{b}$, since equality is usually not exactly satisfiable when $m > n$
- Least squares solution \mathbf{x} minimizes squared Euclidean norm of residual vector $\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x}$,

$$\min_{\mathbf{x}} \|\mathbf{r}\|_2^2 = \min_{\mathbf{x}} \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2^2$$



Least Squares Idea

Given $\underline{b} \in \mathbb{R}^m$, with $m > n$, find:

$$\begin{aligned}\underline{y} := A\underline{x} &= \underline{a}_1x_1 + \underline{a}_2x_2 + \cdots + \underline{a}_n\underline{x}_n \approx \underline{b} \\ \underline{r} &:= \underline{b} - A\underline{x} = \underline{b} - \underline{y}\end{aligned}$$

Least squares:

$$\text{Minimize } \|\underline{r}\|_2 = \left[\sum_{i=1}^m (b_i - y_i)^2 \right]^{\frac{1}{2}}$$

This system is *overdetermined*.

There are more equations than unknowns.

Least Squares Idea

With $m > n$, we have:

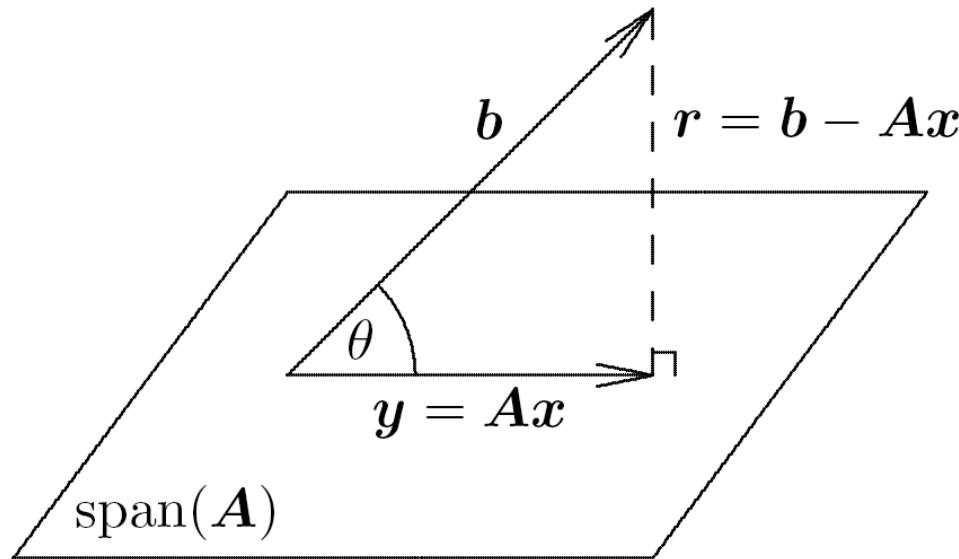
- Lots of data ($\underline{b} \in \mathbb{R}^m$)
- A few model parameters (x_1, x_2, \dots, x_n)
- A few candidate basis vectors ($\underline{a}_1, \underline{a}_2, \dots, \underline{a}_n$)
- Our estimate, $\underline{y} = A\underline{x}$

The matrix A is tall and thin.

$$\begin{array}{c|c|c|c} \underline{y} & = & A & \underline{x} \approx \underline{b} \\ \hline & & & \end{array}$$

Most Important Picture

Geometric relationships among b , r , and $\text{span}(A)$ are shown in diagram



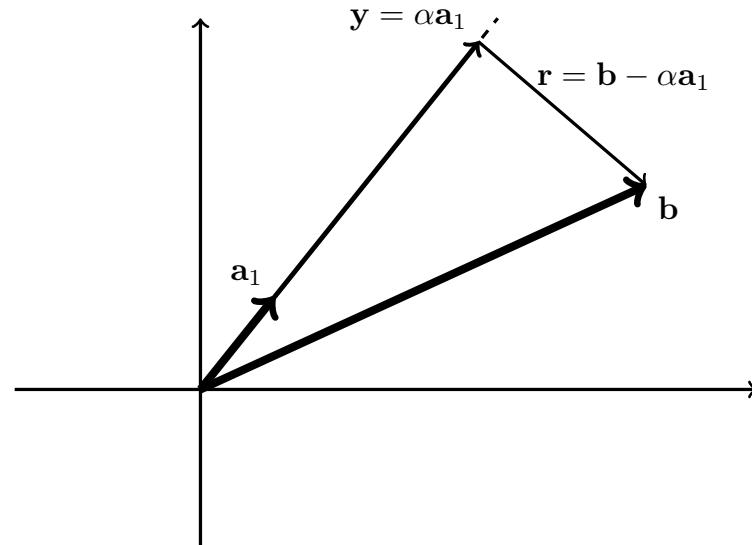
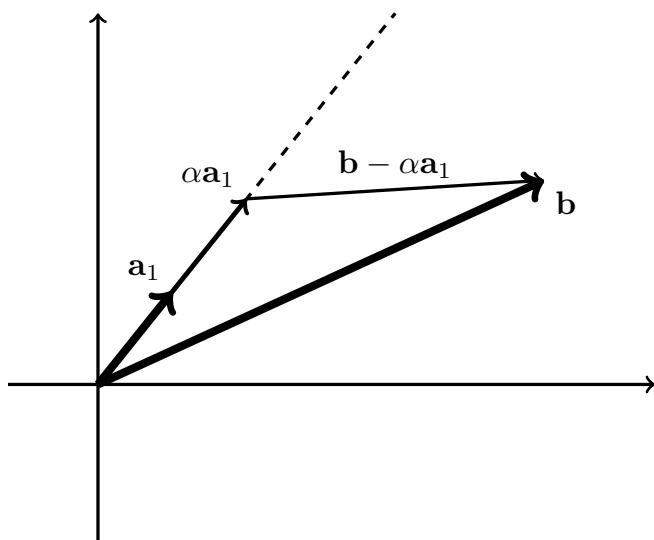
- The vector y is the *orthogonal projection* of b onto $\text{span}(A)$.
- The projection results in minimization of $\|r\|_2$, which, as we shall see, is equivalent to having $r := b - Ax \perp \text{span}(A)$

1D Projection

- Consider the 1D subspace of \mathbb{R}^2 spanned by \mathbf{a}_1 :

$$\alpha \mathbf{a}_1 \in \text{span}\{\mathbf{a}_1\}.$$

- The *projection* of a point $\mathbf{b} \in \mathbb{R}^2$ onto $\text{span}\{\mathbf{a}_1\}$ is the point on the line $\mathbf{y} = \alpha \mathbf{a}_1$ that is closest to \mathbf{b} .
- To find the projection, we look for the value α that minimizes $\|\mathbf{r}\| = \|\alpha \mathbf{a}_1 - \mathbf{b}\|$ in the 2-norm. (Other norms are also possible.)



1D Projection

- Minimizing the square of the residual with respect to α , we have

$$\begin{aligned}\frac{d}{d\alpha} \|\mathbf{r}\|^2 &= \\ &= \frac{d}{d\alpha} (\mathbf{b} - \alpha \mathbf{a}_1)^T (\mathbf{b} - \alpha \mathbf{a}_1) \\ &= \frac{d}{d\alpha} [\mathbf{b}^T \mathbf{b} + \alpha^2 \mathbf{a}_1^T \mathbf{a}_1 - 2\alpha \mathbf{a}_1^T \mathbf{b}] \\ &= 2\alpha \mathbf{a}_1^T \mathbf{a}_1 - 2 \mathbf{a}_1^T \mathbf{b} = 0\end{aligned}$$

- For this to be a minimum, we require the last expression to be zero, which implies

$$\alpha = \frac{\mathbf{a}_1^T \mathbf{b}}{\mathbf{a}_1^T \mathbf{a}_1}, \quad \Rightarrow \quad \mathbf{y} = \alpha \mathbf{a}_1 = \frac{\mathbf{a}_1^T \mathbf{b}}{\mathbf{a}_1^T \mathbf{a}_1} \mathbf{a}_1.$$

- We see that \mathbf{y} points in the direction of \mathbf{a}_1 and has magnitude that scales as \mathbf{b} (but not with \mathbf{a}_1).

Projection in Higher Dimensions

- Here, we have basis coefficients x_i written as $\mathbf{x} = [x_1 \dots x_n]^T$.
- As before, we minimize the square of the norm of the residual

$$\begin{aligned} \|\mathbf{r}\|^2 &= \|A\mathbf{x} - \mathbf{b}\|^2 \\ &= (A\mathbf{x} - \mathbf{b})^T (A\mathbf{x} - \mathbf{b}) \\ &= \mathbf{b}^T \mathbf{b} - \mathbf{b}^T A\mathbf{x} - (A\mathbf{x})^T \mathbf{b} + \mathbf{x}^T A^T A\mathbf{x} \\ &= \mathbf{b}^T \mathbf{b} + \mathbf{x}^T A^T A\mathbf{x} - 2\mathbf{x}^T A^T \mathbf{b}. \end{aligned}$$

- As in the 1D case, we require stationarity with respect to all coefficients

$$\frac{d}{dx_i} \|\mathbf{r}\|^2 = 0$$

- The first term is constant.
- The second and third are more complex.

Projection in Higher Dimensions

- Define $\mathbf{c} = A^T \mathbf{b}$ and $H = A^T A$ such that

$$\mathbf{x}^T A^T \mathbf{b} = \mathbf{x}^T \mathbf{c} = x_1 c_1 + x_2 c_2 + \dots x_n c_n.$$

$$\mathbf{x}^T A^T A \mathbf{x} = \mathbf{x}^T H \mathbf{x} = \sum_{j=1}^n \sum_{k=1}^n x_k H_{kj} x_j$$

- Differentiating with respect to x_i ,

$$\frac{d}{dx_i} (\mathbf{x}^T A^T \mathbf{b}) = c_i = (A^T \mathbf{b})_i, \quad \text{and}$$

$$\begin{aligned} \frac{d}{dx_i} (\mathbf{x}^T H \mathbf{x}) &= \sum_{j=1}^n H_{ij} x_j + \sum_{k=1}^n x_k H_{ki} \\ &= 2 \sum_{j=1}^n H_{ij} x_j = 2 (H \mathbf{x})_i. \end{aligned}$$

Projection in Higher Dimensions

- From the preceding pages, the minimum is realized when

$$0 = \frac{d}{dx_i} (\mathbf{x}^T A^T A \mathbf{x} - 2\mathbf{x}^T A^T \mathbf{b}) = 2(A^T A \mathbf{x} - A^T \mathbf{b})_i, \quad i = 1, \dots, n$$

- Or, in matrix form:

$$\mathbf{x} = (A^T A)^{-1} A^T \mathbf{b}.$$

- As in the 1D case, our *projection* is

$$\mathbf{y} = A\mathbf{x} = A(A^T A)^{-1} A^T \mathbf{b}.$$

- \mathbf{y} has units and length that scale with \mathbf{b} , but it lies in the range of A .
- It is the projection of \mathbf{b} onto $R(A)$.

Important Example: Weighted Least Squares

- Standard inner-product:

$$(u, v)_2 := \sum_{i=1}^m u_i v_i = \mathbf{u}^T \mathbf{v},$$

$$\|\mathbf{r}\|_2^2 = \sum_{i=1}^m r_i^2 = \mathbf{r}^T \mathbf{r},$$

- Consider ***weighted*** inner-product:

$$(u, v)_W := \sum_{i=1}^m u_i w_i v_i = \mathbf{u}^T W \mathbf{v}, \text{ where}$$

$$W = \begin{bmatrix} w_1 & & & \\ & w_2 & & \\ & & \ddots & \\ & & & w_m \end{bmatrix}, \quad w_i > 0.$$

$$\|\mathbf{r}\|_w^2 = \sum_{i=1}^m w_i r_i^2 = \mathbf{r}^T W \mathbf{r},$$

- If we want to minimize in a weighted norm:

Find $\mathbf{x} \in \mathbb{R}^n$ such that $\|\mathbf{r}\|_W^2$ is minimized.

- Require

$$\begin{aligned}
& \frac{d}{dx_i} \left[(\mathbf{b} - A\mathbf{x})^T W (\mathbf{b} - A\mathbf{x}) \right] \\
&= \frac{d}{dx_i} [\mathbf{b}^T W \mathbf{b} + \mathbf{x}^T A^T W A \mathbf{x} - \mathbf{x}^T A^T W \mathbf{b} - \mathbf{b}^T W A \mathbf{x}] \\
&= \frac{d}{dx_i} [\mathbf{x}^T A^T W A \mathbf{x} - 2\mathbf{x}^T A^T W \mathbf{b}] \\
&= 0.
\end{aligned}$$

- Thus, $\mathbf{x} = (A^T W A)^{-1} A^T W \mathbf{b}$,
- $$\mathbf{y} = A\mathbf{x} = A(A^T W A)^{-1} A^T W \mathbf{b}, \approx \mathbf{b}.$$

- \mathbf{y} is the **weighted** least-squares approximation to \mathbf{b} .
- Works for *any* SPD W , not just (positive) diagonal ones.
- Can be used to solve linear systems.

Using Least Squares to Solve Linear Systems

- In particular, suppose $W\mathbf{b} = \mathbf{z}$.
- Linear system — \mathbf{z} is right-hand side, *known*.
 - \mathbf{b} is *unknown*.
- Want to find weighted least-squares fit, $\mathbf{y} \approx \mathbf{b}$, minimizing $\|\mathbf{y} - \mathbf{b}\|_W^2$ with $\mathbf{y} \in \mathcal{R}(A)$.
- Answer:

$$\begin{aligned}\mathbf{y} &= A(A^T W A)^{-1} A^T W \mathbf{b} \\ &= A(A^T W A)^{-1} A^T \mathbf{z} \\ &= A\mathbf{x}\end{aligned}$$

Using Least Squares to Solve Linear Systems

- Suppose W is a sparse $m \times m$ matrix with (say) $m > 10^6$.
- Factor cost is likely very large (superlinear in m).
- If $A = (\mathbf{a}_1 \mathbf{a}_2 \cdots \mathbf{a}_n)$, $n \ll m$, can form n vectors,

$$WA = (W\mathbf{a}_1 \ W\mathbf{a}_2 \ \cdots \ W\mathbf{a}_n),$$

and the *Gram* matrix, $\tilde{W} = A^T W A = [\mathbf{a}_i^T W \mathbf{a}_j]$, and solve

$$\tilde{W}\mathbf{x} = A^T \mathbf{z} = \begin{pmatrix} \mathbf{a}_1^T \mathbf{z} \\ \mathbf{a}_2^T \mathbf{z} \\ \vdots \\ \mathbf{a}_n^T \mathbf{z} \end{pmatrix},$$

which requires solution of a small $n \times n$ system, \tilde{W} .

Using Least Squares to Solve Linear Systems

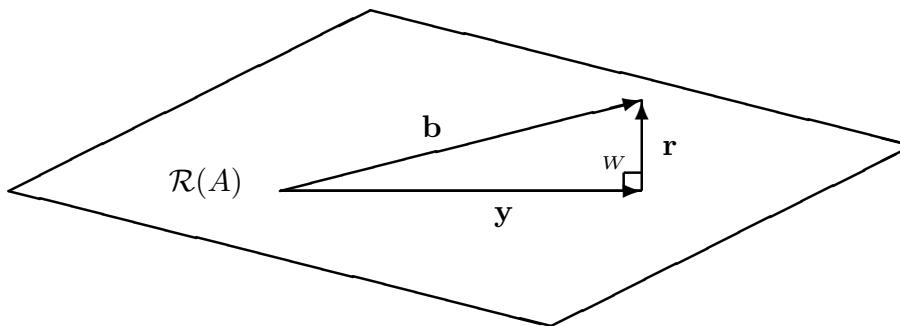
- Once we have \mathbf{x} ,

$$\mathbf{y} = A\mathbf{x} = \sum_{j=1}^n \mathbf{a}_j x_j \approx \mathbf{b} := W^{-1}\mathbf{z}.$$

- So, *weighted inner-product* allows us to approximate \mathbf{b} , the solution to $W\mathbf{b} = \mathbf{z}$, without knowing \mathbf{b} !
- Approximate solution $\mathbf{y} \in \mathcal{R}(A) = \text{span}\{\mathbf{a}_1 \mathbf{a}_2 \cdots \mathbf{a}_n\}$:

$$\mathbf{y} = A(A^T W A)^{-1} A^T \mathbf{z}$$

- \mathbf{y} is the *projection* of \mathbf{b} onto $\mathcal{R}(A)$,
 - the *closest approximation* or *best fit* in $\mathcal{R}(A)$ in the W -norm.



- \mathbf{r} is W -orthogonal to $\mathcal{R}(A)$.

Using Least Squares to Solve Linear Systems

- Very often can have accurate approximations with $n \ll m$.
- In particular, if $\kappa := \text{cond}(W)$, and

$$\begin{aligned}\mathcal{R}(A) &= \text{span}\{W\mathbf{b}, W^2\mathbf{b}, \dots, W^k\mathbf{b}\} \\ &= \text{span}\{\mathbf{z}, W\mathbf{z}, \dots, W^{k-1}\mathbf{z}\},\end{aligned}$$

then can have an accurate answer with $k \approx \sqrt{\kappa}$.

- Can keep increasing $\mathcal{R}(A)$ with additional matrix-vector products.
- This method corresponds to *conjugate gradient iteration* applied to the SPD system $W\mathbf{b} = \mathbf{z}$.

Back to Standard Least Squares

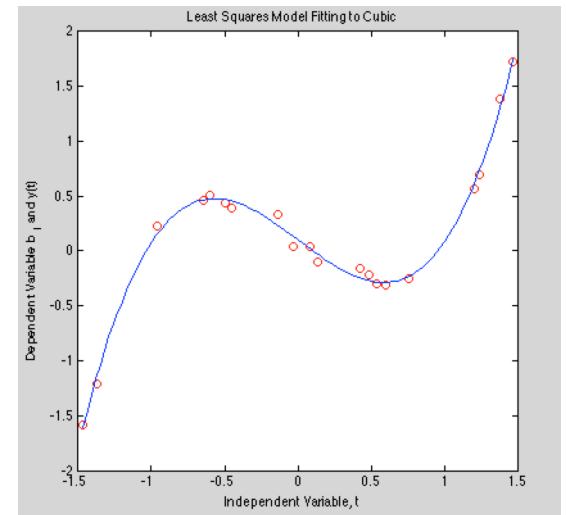
- ❑ Suppose we have observational data, $\{ b_i \}$ at some independent times $\{ t_i \}$ (red circles).
- ❑ The t_i s do not need to be sorted and can in fact be repeated.
- ❑ We wish to fit a smooth model (blue curve) to the data so we can compactly describe (and perhaps integrate or differentiate) the functional relationship between $b(t)$ and t .

A common model is of the form:

$$y(t) = \phi_1(t)x_1 + \phi_2(t)x_2 + \dots + \phi_n(t)x_n$$

The $\phi_j(t)$ s are the basis functions and x_j s the unknown basis coefficients.

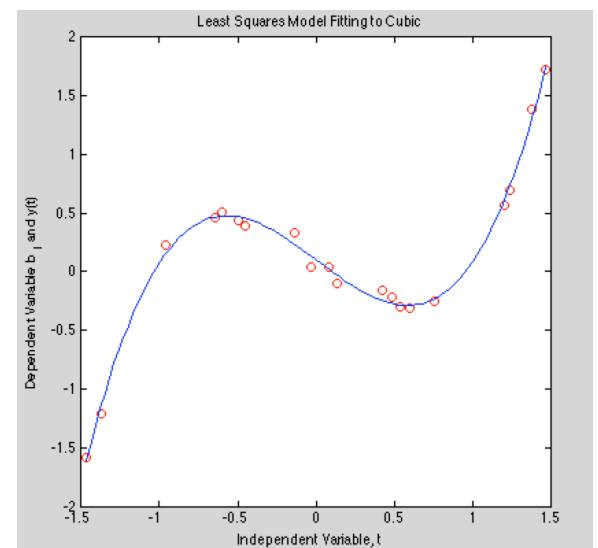
The system is *linear* with respect to the unknowns, hence, these are *linear least squares* problems.



Example

- To proceed, we assume b_i represents a function at time points t_i , which we are trying to model.
- We select basis functions, e.g., $\phi_j(t) = t^{j-1}$ would span the space of polynomials of up to degree $n - 1$.
(This might not be the best basis for the polynomials...)
- We then set $\{\underline{a}_j\}_i = \phi_j(t_i)$ for each column $j = 1, \dots, n$.
- We then solve the linear least squares problem: $\min ||\underline{b} - A\underline{x}||^2$
- Once we have the x_j s, we can reconstruct the smooth function:

$$y(t) = \sum_{j=1}^n \phi_j(t)x_j$$



Matlab Example

```
% Linear Least Squares Demo

degree=3; m=20; n=degree+1;

t=3*(rand(m,1)-0.5);
b = t.^3 - t; b=b+0.2*rand(m,1); %% Expect: x =~ [ 0 -1 0 1 ]

plot(t,b,'ro'), pause

%%%% DEFINE a_ij = phi_j(t_i)

A=zeros(m,n); for j=1:n; A(:,j) = t.^^(j-1); end;

A0=A; b0=b; % Save A & b.

%%%%% SOLVE LEAST SQUARES PROBLEM via Normal Equations &&&&

x = (A'*A) \ A'*b

plot(t,b0,'ro',t,A0*x,'bo',t,1*(b0-A0*x),'kx'), pause
plot(t,A0*x,'bo'), pause

%% CONSTRUCT SMOOTH APPROXIMATION

tt=(0:100)'/100; tt=min(t) + (max(t)-min(t))*tt;
S=zeros(101,n); for k=1:n; S(:,k) = tt.^^(k-1); end;
s=S*x;

plot(t,b0,'ro',tt,s,'b-')
title('Least Squares Model Fitting to Cubic')
xlabel('Independent Variable, t')
ylabel('Dependent Variable b_i and y(t)')
```

Python Least Squares Example

```
# % Linear Least Squares Demo
import numpy as np
import scipy as sp
import matplotlib
matplotlib.use('MacOSX')
import matplotlib.pyplot as plt
##import pylab

degree=3; m=20; n=degree+1;

t=3*(np.random.rand(m,1)-0.5);

b = t**3 - t;
b = b+0.2*np.random.rand(m,1); ##Expect: x =~ [ 0 -1 0 1 ]

plt.plot(t,b,'ro')
plt.show()                                     # %%% SOLVE LEAST SQUARES PROBLEM via Normal Equations
x = np.linalg.solve(np.dot(A.T, A), np.dot(A.T,b))

plt.figure()
plt.plot(t,b0,'ro')
plt.plot(t,np.dot(A0,x),'bo')
plt.plot(t,b0-np.dot(A0,x),'kx')
plt.show()

plt.figure()
plt.plot(t,np.dot(A0,x),'bo')
plt.show()

# %% CONSTRUCT SMOOTH APPROXIMATION
tt=np.linspace(0,100,101)/100
tt=min(t) + (max(t)-min(t))*tt;

S=np.zeros((101,n))
for k in range(n):
    S[:,k] = tt***(k)
s= np.dot(S, x)

plt.figure()
plt.plot(t,b0,'ro')
plt.plot(tt,s,'b-')
plt.title('Least Squares Model Fitting to Cubic')
plt.xlabel('Independent Variable, t')
plt.ylabel('Dependent Variable b_i and y(t)')
plt.show()
```

Note on the text examples

- Note, the text uses similar examples.
- The notation in the examples is a bit different from the rest of the derivation... so be sure to pay attention.

Data Fitting

- Given m data points (t_i, y_i) , find n -vector \mathbf{x} of parameters that gives “best fit” to model function $f(t, \mathbf{x})$,

$$\min_{\mathbf{x}} \sum_{i=1}^m (y_i - f(t_i, \mathbf{x}))^2$$

- Problem is *linear* if function f is linear in components of \mathbf{x} ,

$$f(t, \mathbf{x}) = x_1 \phi_1(t) + x_2 \phi_2(t) + \cdots + x_n \phi_n(t)$$

where functions ϕ_j depend only on t

- Problem can be written in matrix form as $A\mathbf{x} \cong \mathbf{b}$, with $a_{ij} = \phi_j(t_i)$ and $b_i = y_i$



Data Fitting

- Polynomial fitting

$$f(t, \mathbf{x}) = x_1 + x_2 t + x_3 t^2 + \cdots + x_n t^{n-1}$$

is linear, since polynomial linear in coefficients, though nonlinear in independent variable t

- Fitting sum of exponentials

$$f(t, \mathbf{x}) = x_1 e^{x_2 t} + \cdots + x_{n-1} e^{x_n t}$$

is example of nonlinear problem

- For now, we will consider only linear least squares problems



Example: Data Fitting

- Fitting quadratic polynomial to five data points gives linear least squares problem

$$\mathbf{A}\mathbf{x} = \begin{bmatrix} 1 & t_1 & t_1^2 \\ 1 & t_2 & t_2^2 \\ 1 & t_3 & t_3^2 \\ 1 & t_4 & t_4^2 \\ 1 & t_5 & t_5^2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \cong \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{bmatrix} = \mathbf{b}$$

- Matrix whose columns (or rows) are successive powers of independent variable is called *Vandermonde matrix*



Example, continued

- For data

t	-1.0	-0.5	0.0	0.5	1.0
y	1.0	0.5	0.0	0.5	2.0

overdetermined 5×3 linear system is

$$A\mathbf{x} = \begin{bmatrix} 1 & -1.0 & 1.0 \\ 1 & -0.5 & 0.25 \\ 1 & 0.0 & 0.0 \\ 1 & 0.5 & 0.25 \\ 1 & 1.0 & 1.0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \cong \begin{bmatrix} 1.0 \\ 0.5 \\ 0.0 \\ 0.5 \\ 2.0 \end{bmatrix} = \mathbf{b}$$

- Solution, which we will see later how to compute, is

$$\mathbf{x} = [0.086 \quad 0.40 \quad 1.4]^T$$

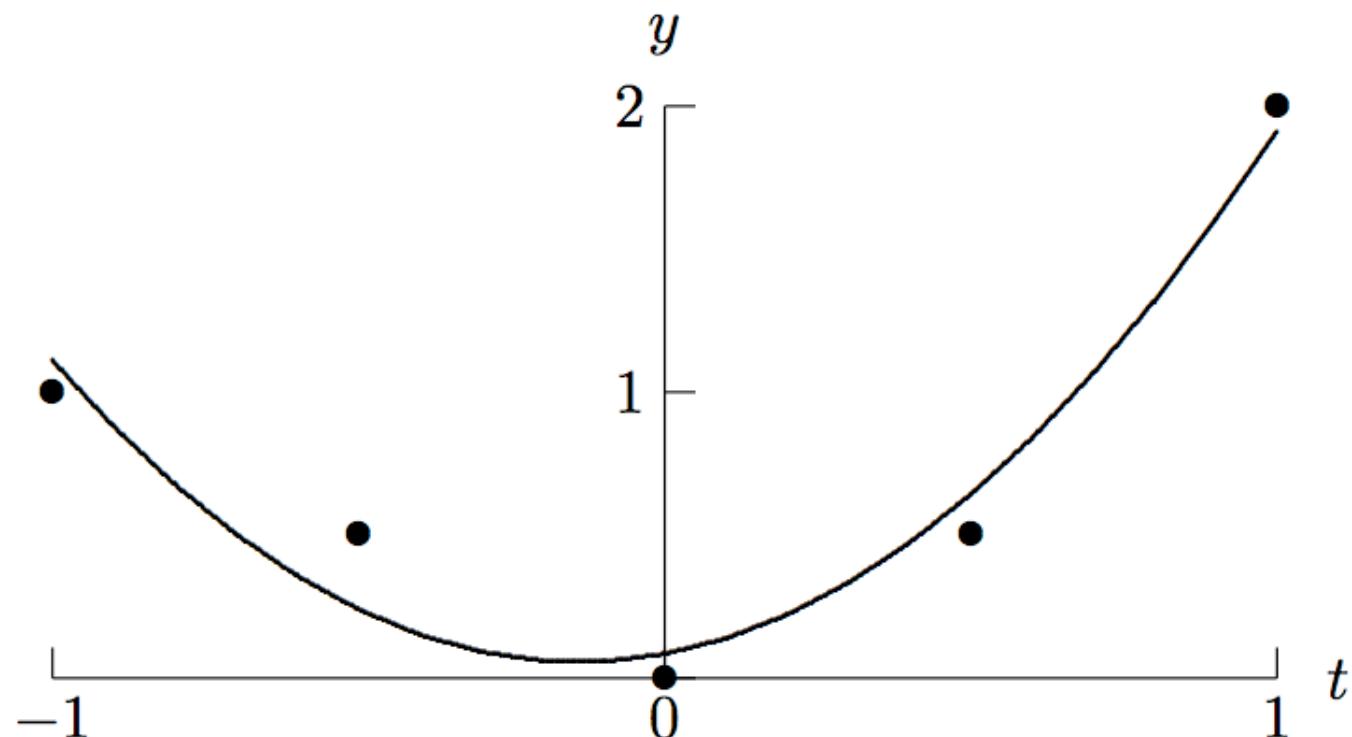
so approximating polynomial is

$$p(t) = 0.086 + 0.4t + 1.4t^2$$



Example, continued

- Resulting curve and original data points are shown in graph



Existence and Uniqueness

- Linear least squares problem $Ax \cong b$ **always** has solution
- Solution is **unique** if, and only if, columns of A are **linearly independent**, i.e., $\text{rank}(A) = n$, where A is $m \times n$
- If $\text{rank}(A) < n$, then A is **rank-deficient**, and solution of linear least squares problem is not unique
- For now, we assume A has full column rank n



Normal Equations

- To minimize squared Euclidean norm of residual vector

$$\begin{aligned}\|r\|_2^2 &= r^T r = (\mathbf{b} - \mathbf{A}\mathbf{x})^T(\mathbf{b} - \mathbf{A}\mathbf{x}) \\ &= \mathbf{b}^T \mathbf{b} - 2\mathbf{x}^T \mathbf{A}^T \mathbf{b} + \mathbf{x}^T \mathbf{A}^T \mathbf{A}\mathbf{x}\end{aligned}$$

take derivative with respect to \mathbf{x} and set it to 0,

$$2\mathbf{A}^T \mathbf{A}\mathbf{x} - 2\mathbf{A}^T \mathbf{b} = \mathbf{0}$$

which reduces to $n \times n$ linear system of *normal equations*

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



Orthogonality

- Vectors v_1 and v_2 are *orthogonal* if their inner product is zero, $v_1^T v_2 = 0$
- Space spanned by columns of $m \times n$ matrix A , $\text{span}(A) = \{Ax : x \in \mathbb{R}^n\}$, is of dimension at most n
- If $m > n$, b generally does not lie in $\text{span}(A)$, so there is no exact solution to $Ax = b$
- Vector $y = Ax$ in $\text{span}(A)$ closest to b in 2-norm occurs when residual $r = b - Ax$ is *orthogonal* to $\text{span}(A)$,

$$\mathbf{0} = A^T r = A^T(b - Ax)$$

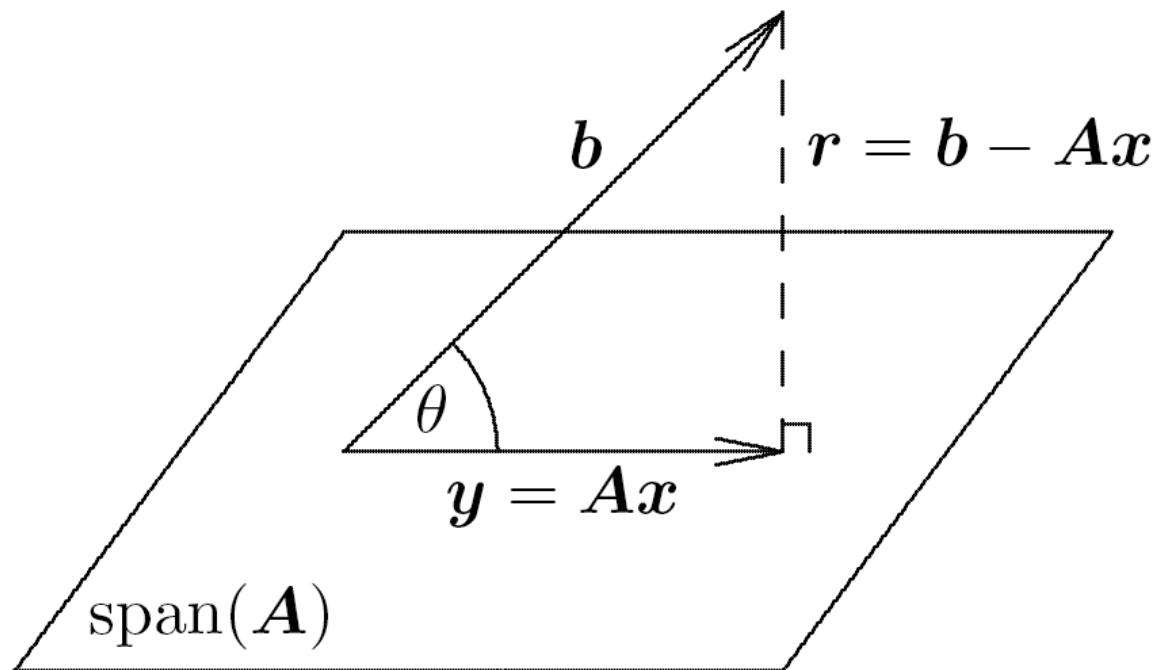
again giving system of *normal equations*

$$A^T A x = A^T b$$



Orthogonality, continued

- Geometric relationships among b , r , and $\text{span}(A)$ are shown in diagram



Orthogonal Projectors

- Matrix P is *orthogonal projector* if it is *idempotent* ($P^2 = P$) and *symmetric* ($P^T = P$)
- Orthogonal projector onto orthogonal complement $\text{span}(P)^\perp$ is given by $P_\perp = I - P$
- For any vector v ,

$$v = (P + (I - P)) v = Pv + P_\perp v$$

- For least squares problem $Ax \cong b$, if $\text{rank}(A) = n$, then

$$P = A(A^T A)^{-1} A^T$$

is orthogonal projector onto $\text{span}(A)$, and

$$b = Pb + P_\perp b = Ax + (b - Ax) = y + r$$



Pseudoinverse and Condition Number

- Nonsquare $m \times n$ matrix \mathbf{A} has no inverse in usual sense
- If $\text{rank}(\mathbf{A}) = n$, *pseudoinverse* is defined by

$$\mathbf{A}^+ = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T$$

and condition number by

$$\text{cond}(\mathbf{A}) = \|\mathbf{A}\|_2 \cdot \|\mathbf{A}^+\|_2$$

- By convention, $\text{cond}(\mathbf{A}) = \infty$ if $\text{rank}(\mathbf{A}) < n$
- Just as condition number of square matrix measures closeness to singularity, condition number of rectangular matrix measures closeness to rank deficiency
- Least squares solution of $\mathbf{Ax} \cong \mathbf{b}$ is given by $\mathbf{x} = \mathbf{A}^+ \mathbf{b}$



Sensitivity and Conditioning

- Sensitivity of least squares solution to $\mathbf{Ax} \cong \mathbf{b}$ depends on \mathbf{b} as well as \mathbf{A}
- Define angle θ between \mathbf{b} and $\mathbf{y} = \mathbf{Ax}$ by

$$\cos(\theta) = \frac{\|\mathbf{y}\|_2}{\|\mathbf{b}\|_2} = \frac{\|\mathbf{Ax}\|_2}{\|\mathbf{b}\|_2}$$

- Bound on perturbation Δx in solution x due to perturbation Δb in b is given by

$$\frac{\|\Delta x\|_2}{\|x\|_2} \leq \text{cond}(A) \frac{1}{\cos(\theta)} \frac{\|\Delta b\|_2}{\|b\|_2}$$



Sensitivity and Conditioning, continued

- Similarly, for perturbation E in matrix A ,

$$\frac{\|\Delta x\|_2}{\|x\|_2} \lesssim ([\text{cond}(A)]^2 \tan(\theta) + \text{cond}(A)) \frac{\|E\|_2}{\|A\|_2}$$

- Condition number of least squares solution is about $\text{cond}(A)$ if residual is small, but can be squared or arbitrarily worse for large residual



Normal Equations Method

- If $m \times n$ matrix A has rank n , then symmetric $n \times n$ matrix $A^T A$ is positive definite, so its Cholesky factorization

$$A^T A = LL^T$$

can be used to obtain solution x to system of normal equations

$$A^T A x = A^T b$$

which has same solution as linear least squares problem
 $Ax \cong b$

- Normal equations method involves transformations

rectangular \longrightarrow square \longrightarrow triangular



Spoiler: Normal Equations *not* Recommended

- So far, our examples have used normal equations approach, as do the next examples.
- After the introduction, most of this chapter is devoted to better methods in which columns of A are first *orthogonalized*.
- Orthogonalization methods of choice:
 - Householder transformations (very stable)
 - Givens rotations (stable, and often cheaper than Householder)
 - Gram-Schmidt (better than normal eqns, but not great)
 - Modified Gram-Schmidt (better than “classical” Gram-Schmidt)

Example: Normal Equations Method

- For polynomial data-fitting example given previously, normal equations method gives

$$A^T A = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ -1.0 & -0.5 & 0.0 & 0.5 & 1.0 \\ 1.0 & 0.25 & 0.0 & 0.25 & 1.0 \end{bmatrix} \begin{bmatrix} 1 & -1.0 & 1.0 \\ 1 & -0.5 & 0.25 \\ 1 & 0.0 & 0.0 \\ 1 & 0.5 & 0.25 \\ 1 & 1.0 & 1.0 \end{bmatrix}$$

$$= \begin{bmatrix} 5.0 & 0.0 & 2.5 \\ 0.0 & 2.5 & 0.0 \\ 2.5 & 0.0 & 2.125 \end{bmatrix},$$

$$A^T b = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ -1.0 & -0.5 & 0.0 & 0.5 & 1.0 \\ 1.0 & 0.25 & 0.0 & 0.25 & 1.0 \end{bmatrix} \begin{bmatrix} 1.0 \\ 0.5 \\ 0.0 \\ 0.5 \\ 2.0 \end{bmatrix} = \begin{bmatrix} 4.0 \\ 1.0 \\ 3.25 \end{bmatrix}$$



Example, continued

- Cholesky factorization of symmetric positive definite matrix $\mathbf{A}^T \mathbf{A}$ gives

$$\begin{aligned}\mathbf{A}^T \mathbf{A} &= \begin{bmatrix} 5.0 & 0.0 & 2.5 \\ 0.0 & 2.5 & 0.0 \\ 2.5 & 0.0 & 2.125 \end{bmatrix} \\ &= \begin{bmatrix} 2.236 & 0 & 0 \\ 0 & 1.581 & 0 \\ 1.118 & 0 & 0.935 \end{bmatrix} \begin{bmatrix} 2.236 & 0 & 1.118 \\ 0 & 1.581 & 0 \\ 0 & 0 & 0.935 \end{bmatrix} = \mathbf{L} \mathbf{L}^T\end{aligned}$$

- Solving lower triangular system $\mathbf{L}z = \mathbf{A}^T \mathbf{b}$ by forward-substitution gives $z = [1.789 \quad 0.632 \quad 1.336]^T$
- Solving upper triangular system $\mathbf{L}^T x = z$ by back-substitution gives $x = [0.086 \quad 0.400 \quad 1.429]^T$



Shortcomings of Normal Equations

- Information can be lost in forming $\mathbf{A}^T \mathbf{A}$ and $\mathbf{A}^T \mathbf{b}$
- For example, take

$$\mathbf{A} = \begin{bmatrix} 1 & 1 \\ \epsilon & 0 \\ 0 & \epsilon \end{bmatrix}$$

where ϵ is positive number smaller than $\sqrt{\epsilon_{\text{mach}}}$

- Then in floating-point arithmetic

$$\mathbf{A}^T \mathbf{A} = \begin{bmatrix} 1 + \epsilon^2 & 1 \\ 1 & 1 + \epsilon^2 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

which is singular

- Sensitivity of solution is also worsened, since

$$\text{cond}(\mathbf{A}^T \mathbf{A}) = [\text{cond}(\mathbf{A})]^2$$



- ❑ Avoid normal equations:

$$\square A^T A x = A^T b$$

- ❑ Instead, orthogonalize columns of A

$$\square Ax = QRx = b$$

- ❑ Columns of Q are orthonormal
- ❑ R is upper triangular

Projection, QR Factorization, Gram-Schmidt

- Recall our linear least squares problem:

$$\mathbf{y} = A\mathbf{x} \approx \mathbf{b},$$

which is equivalent to minimization / orthogonal projection:

$$\mathbf{r} := \mathbf{b} - A\mathbf{x} \perp \mathcal{R}(A)$$

$$\|\mathbf{r}\|_2 = \|\mathbf{b} - \mathbf{y}\|_2 \leq \|\mathbf{b} - \mathbf{v}\|_2 \quad \forall \mathbf{v} \in \mathcal{R}(A).$$

- This problem has solutions

$$\mathbf{x} = (A^T A)^{-1} A^T \mathbf{b}$$

$$\mathbf{y} = A (A^T A)^{-1} A^T \mathbf{b} = P \mathbf{b},$$

where $P := A (A^T A)^{-1} A^T$ is the *orthogonal projector* onto $\mathcal{R}(A)$.

Observations

$$(A^T A) \mathbf{x} = A^T \mathbf{b} = \begin{pmatrix} \mathbf{a}_1^T \mathbf{b} \\ \mathbf{a}_2^T \mathbf{b} \\ \vdots \\ \mathbf{a}_n^T \mathbf{b} \end{pmatrix}$$

$$(A^T A) = \begin{pmatrix} \mathbf{a}_1^T \mathbf{a}_1 & \mathbf{a}_1^T \mathbf{a}_2 & \cdots & \mathbf{a}_1^T \mathbf{a}_n \\ \mathbf{a}_2^T \mathbf{a}_1 & \mathbf{a}_2^T \mathbf{a}_2 & \cdots & \mathbf{a}_2^T \mathbf{a}_n \\ \vdots & & & \vdots \\ \mathbf{a}_n^T \mathbf{a}_1 & \mathbf{a}_n^T \mathbf{a}_2 & \cdots & \mathbf{a}_n^T \mathbf{a}_n \end{pmatrix}.$$

Orthogonal Bases

- If the columns of A were *orthogonal*, such that $a_{ij} = \mathbf{a}_i^T \mathbf{a}_j = 0$ for $i \neq j$, then $A^T A$ is a diagonal matrix,

$$(A^T A) = \begin{pmatrix} \mathbf{a}_1^T \mathbf{a}_1 & & & \\ & \mathbf{a}_2^T \mathbf{a}_2 & & \\ & & \ddots & \\ & & & \mathbf{a}_n^T \mathbf{a}_n \end{pmatrix},$$

and the system is easily solved,

$$\mathbf{x} = (A^T A)^{-1} A^T \mathbf{b} = \begin{pmatrix} \frac{1}{\mathbf{a}_1^T \mathbf{a}_1} & & & \\ & \frac{1}{\mathbf{a}_2^T \mathbf{a}_2} & & \\ & & \ddots & \\ & & & \frac{1}{\mathbf{a}_n^T \mathbf{a}_n} \end{pmatrix} \begin{pmatrix} \mathbf{a}_1^T \mathbf{b} \\ \mathbf{a}_2^T \mathbf{b} \\ \vdots \\ \mathbf{a}_n^T \mathbf{b} \end{pmatrix}.$$

- In this case, we can write the projection in closed form:

$$\mathbf{y} = \sum_{j=1}^n x_j \mathbf{a}_j = \sum_{j=1}^n \frac{\mathbf{a}_j^T \mathbf{b}}{\mathbf{a}_j^T \mathbf{a}_j} \mathbf{a}_j. \quad (1)$$

- For *orthogonal* bases, (1) is the projection of \mathbf{b} onto $\text{span}\{\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n\}$.

Orthonormal Bases

- If the columns are orthogonal and *normalized* such that $\|\mathbf{a}_j\| = 1$, we then have $\mathbf{a}_j^T \mathbf{a}_j = 1$, or more generally

$$\mathbf{a}_i^T \mathbf{a}_j = \delta_{ij}, \text{ with } \delta_{ij} := \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases} \text{ the Kronecker delta,}$$

- In this case, $A^T A = I$ and the orthogonal projection is given by

$$\mathbf{y} = A A^T \mathbf{b} = \sum_{j=1}^n \mathbf{a}_j (\mathbf{a}_j^T \mathbf{b}).$$

Example: Suppose our model fit is based on sine functions, sampled uniformly on $[0, \pi]$:

$$\phi_j(t) = \sin j t_i, \quad t_i = \pi i / m, \quad i = 1, \dots, m.$$

In this case,

$$A = (\phi_1(t_i) \ \phi_2(t_i) \ \cdots \ \phi_n(t_i)),$$

$$A^T A = \frac{n}{2} I.$$

QR Factorization

- Generally, we don't *a priori* have orthonormal bases.
- We can construct them, however. The process is referred to as QR factorization.
- We seek factors Q and R such that $QR = A$ with Q orthogonal (or, *unitary*, in the complex case).
- There are two cases of interest:

Reduced QR

$$\begin{array}{c|c} Q_1 & R \\ \hline \end{array} = A$$

Full QR

$$\begin{array}{c|c} Q & R \\ \hline O & \\ \hline \end{array} = A$$

- Note that

$$A = Q \begin{bmatrix} R \\ O \end{bmatrix} = [Q_1 \quad Q_2] \begin{bmatrix} R \\ O \end{bmatrix} = Q_1 R.$$

- The columns of Q_1 form an orthonormal basis for $\mathcal{R}(A)$.
- The columns of Q_2 form an orthonormal basis for $\mathcal{R}(A)^\perp$.

QR Factorization: Gram-Schmidt

- We'll look at three approaches to QR :
 - Gram-Schmidt Orthogonalization,
 - Householder Transformations, and
 - Givens Rotations
- We start with Gram-Schmidt - which is most intuitive.
- We are interested in generating orthogonal subspaces that match the nested column spaces of A ,

$$\text{span}\{\mathbf{a}_1\} = \text{span}\{\mathbf{q}_1\}$$

$$\text{span}\{\mathbf{a}_1, \mathbf{a}_2\} = \text{span}\{\mathbf{q}_1, \mathbf{q}_2\}$$

$$\text{span}\{\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3\} = \text{span}\{\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3\}$$

$$\text{span}\{\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n\} = \text{span}\{\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_n\}$$

QR Factorization: Gram-Schmidt

- It's clear that the conditions

$$\text{span}\{\mathbf{a}_1\} = \text{span}\{\mathbf{q}_1\}$$

$$\text{span}\{\mathbf{a}_1, \mathbf{a}_2\} = \text{span}\{\mathbf{q}_1, \mathbf{q}_2\}$$

$$\text{span}\{\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3\} = \text{span}\{\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3\}$$

$$\text{span}\{\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n\} = \text{span}\{\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_n\}$$

are equivalent to the equations

$$\mathbf{a}_1 = \mathbf{q}_1 r_{11}$$

$$\mathbf{a}_2 = \mathbf{q}_1 r_{12} + \mathbf{q}_2 r_{22}$$

$$\mathbf{a}_3 = \mathbf{q}_1 r_{13} + \mathbf{q}_2 r_{23} + \mathbf{q}_3 r_{33}$$

$$\vdots = \vdots + \dots$$

$$\mathbf{a}_n = \mathbf{q}_1 r_{1n} + \mathbf{q}_2 r_{2n} + \dots + \mathbf{q}_n r_{nn}$$

$$\text{i.e., } A = QR$$

(For now, we drop the distinction between Q and Q_1 , and focus only on the reduced QR problem.)

Gram-Schmidt Orthogonalization

- The preceding relationship suggests the first algorithm.

Let $Q_{j-1} := [\mathbf{q}_1 \ \mathbf{q}_2 \ \dots \ \mathbf{q}_{j-1}]$, $P_{j-1} := Q_j Q_{j-1}^T$, $P_{\perp,j-1} := I - P_{j-1}$.

for $j = 2, \dots, n-1$

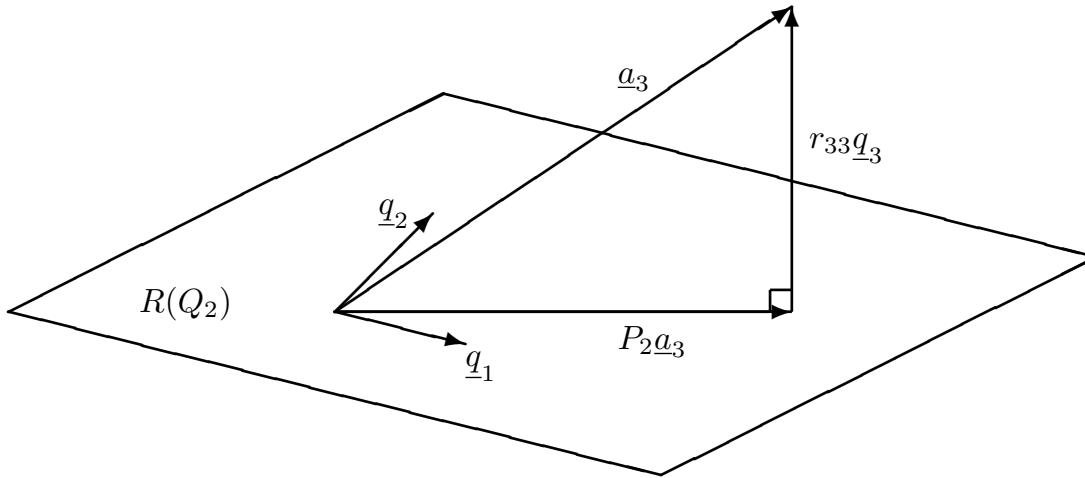
$$\mathbf{v}_j = \mathbf{a}_j - P_{j-1} \mathbf{a}_j = (I - P_{j-1}) \mathbf{a}_j = P_{\perp,j-1} \mathbf{a}_j$$

$$\mathbf{q}_j = \frac{\mathbf{v}_j}{\|\mathbf{v}_j\|} = \frac{P_{\perp,j-1} \mathbf{a}_j}{\|P_{\perp,j-1} \mathbf{a}_j\|}$$

end

- This is *Gram-Schmidt orthogonalization*.
- Each new vector \mathbf{q}_j starts with \mathbf{a}_j and subtracts off the projection onto $\mathcal{R}(Q_{j-1})$, followed by normalization.

Classical Gram-Schmidt Orthogonalization



$$\begin{aligned} P_2 \underline{a}_3 &= Q_2 Q_2^T \underline{a}_3 \\ &= \underline{q}_1 \frac{\underline{q}_1^T \underline{a}_3}{\underline{q}_1^T \underline{q}_1} + \underline{q}_2 \frac{\underline{q}_2^T \underline{a}_3}{\underline{q}_2^T \underline{q}_2} \\ &= \underline{q}_1 \underline{q}_1^T \underline{a}_3 + \underline{q}_2 \underline{q}_2^T \underline{a}_3 \end{aligned}$$

In general, if Q_k is an orthogonal matrix, then
 $P_k = Q_k Q_k^T$ is an orthogonal projector onto $R(Q_k)$

Gram-Schmidt Orthogonalization

- The preceding relationship suggests the first algorithm.

Let $Q_{j-1} := [\mathbf{q}_1 \ \mathbf{q}_2 \ \dots \ \mathbf{q}_{j-1}]$, $P_{j-1} := Q_j Q_{j-1}^T$, $P_{\perp,j-1} := I - P_{j-1}$.

for $j = 2, \dots, n-1$

$$\mathbf{v}_j = \mathbf{a}_j - P_{j-1} \mathbf{a}_j = (I - P_{j-1}) \mathbf{a}_j = P_{\perp,j-1} \mathbf{a}_j$$

$$\mathbf{q}_j = \frac{\mathbf{v}_j}{\|\mathbf{v}_j\|} = \frac{P_{\perp,j-1} \mathbf{a}_j}{\|P_{\perp,j-1} \mathbf{a}_j\|}$$

end

- This is *Gram-Schmidt orthogonalization*.
- Each new vector \mathbf{q}_j starts with \mathbf{a}_j and subtracts off the projection onto $\mathcal{R}(Q_{j-1})$, followed by normalization.

Gram-Schmidt: Classical vs. Modified

- We take a closer look at the projection step, $\mathbf{v}_j = \mathbf{a}_j - P_{j-1} \mathbf{a}_j$.
- The classical (unstable) GS projection is executed as

$$\begin{aligned}\mathbf{v}_j &= \mathbf{a}_j \\ \text{for } k &= 1, \dots, j-1, \\ \mathbf{v}_j &= \mathbf{v}_j - \mathbf{q}_k (\mathbf{q}_k^T \mathbf{a}_j) \\ &\text{end}\end{aligned}$$

- The modified GS projection is executed as

$$\begin{aligned}\mathbf{v}_j &= \mathbf{a}_j \\ \text{for } k &= 1, \dots, j-1, \\ \mathbf{v}_j &= \mathbf{v}_j - \mathbf{q}_k (\mathbf{q}_k^T \mathbf{v}_j) \\ &\text{end}\end{aligned}$$

Mathematical Difference Between CGS and MGS

- Let $\tilde{P}_{\perp,j}, := I - \mathbf{q}_j \mathbf{q}_j^T$
- The CGS projection step amounts to

$$\begin{aligned}
 \mathbf{v}_j &= \left(\tilde{P}_{\perp,j-1} \tilde{P}_{\perp,j-2} \cdots \tilde{P}_{\perp,1} \right) \mathbf{a}_j \\
 &= \left(I - \tilde{P}_1 - \tilde{P}_2 - \cdots - \tilde{P}_{j-1} \right) \mathbf{a}_j \\
 &= \mathbf{a}_j - \tilde{P}_1 \mathbf{a}_j - \tilde{P}_2 \mathbf{a}_j - \cdots - \tilde{P}_{j-1} \mathbf{a}_j \\
 &= \mathbf{a}_j - \sum_{k=1}^{j-1} \tilde{P}_k \mathbf{a}_j.
 \end{aligned}$$

- The MGS projection step is equivalent to

$$\begin{aligned}
 \mathbf{v}_j &= \tilde{P}_{\perp,j-1} \left(\tilde{P}_{\perp,j-2} \left(\cdots \left(\tilde{P}_{\perp,1} \mathbf{a}_j \right) \cdots \right) \right) \\
 &= \left(I - \tilde{P}_{j-1} \right) \left(I - \tilde{P}_{j-2} \right) \cdots \left(I - \tilde{P}_1 \right) \mathbf{a}_j \\
 &= \prod_{k=1}^{j-1} \left(I - \tilde{P}_k \right) \mathbf{a}_j
 \end{aligned}$$

Mathematical Difference Between CGS and MGS

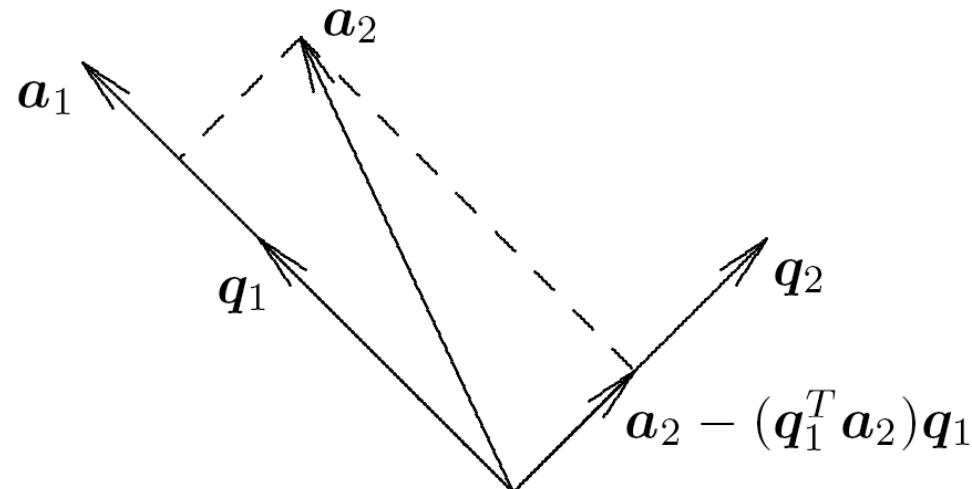
- Lack of associativity in floating point arithmetic drives the difference between CGS and MGS.
- Conceptually, MGS projects the residual, $\mathbf{r}_j := \mathbf{a}_j - P_{j-1}\mathbf{a}_j$.
- As we shall see, neither GS nor MGS are as robust as Householder transformations.
- Both, however, can be cleaned up with a second-pass through the orthogonalization process. (Just set $A = Q$ and repeat, once.)

MGS is an example of the idea that “small corrections are preferred to large ones:

Better to update \mathbf{v} by subtracting off the projection of \mathbf{v} , rather than the projection of \mathbf{a} .

Gram-Schmidt Orthogonalization

- Given vectors a_1 and a_2 , we seek orthonormal vectors q_1 and q_2 having same span
- This can be accomplished by subtracting from second vector its projection onto first vector and normalizing both resulting vectors, as shown in diagram



Gram-Schmidt Orthogonalization

- Process can be extended to any number of vectors $\mathbf{a}_1, \dots, \mathbf{a}_k$, orthogonalizing each successive vector against all preceding ones, giving *classical Gram-Schmidt* procedure

```

for  $k = 1$  to  $n$ 
     $\mathbf{q}_k = \mathbf{a}_k$ 
    for  $j = 1$  to  $k - 1$ 
         $r_{jk} = \mathbf{q}_j^T \mathbf{a}_k$             $\leftarrow$  Coefficient involves original  $\mathbf{a}_k$ 
         $\mathbf{q}_k = \mathbf{q}_k - r_{jk} \mathbf{q}_j$ 
    end
     $r_{kk} = \|\mathbf{q}_k\|_2$ 
     $\mathbf{q}_k = \mathbf{q}_k / r_{kk}$ 
end

```

- Resulting \mathbf{q}_k and r_{jk} form reduced QR factorization of A



Modified Gram-Schmidt

- Classical Gram-Schmidt procedure often suffers loss of orthogonality in finite-precision
- Also, separate storage is required for A , Q , and R , since original a_k are needed in inner loop, so q_k cannot overwrite columns of A
- Both deficiencies are improved by *modified Gram-Schmidt* procedure, with each vector orthogonalized in turn against all *subsequent* vectors, so q_k can overwrite a_k



Modified Gram-Schmidt QR Factorization

- Modified Gram-Schmidt algorithm

```
for k = 1 to n
     $r_{kk} = \|a_k\|_2$ 
     $q_k = a_k / r_{kk}$ 
    for j = k + 1 to n
         $r_{kj} = q_k^T a_j$            ← Coefficient involves modified  $a_j$ 
         $a_j = a_j - r_{kj} q_k$ 
    end
end
```

Matlab Demo: house.m



Classical & Modified GS: Notes

```
n=20;

A = rand(n,n); [Q,R]=qr(A);
for i=1:n; R(i,i)=R(i,i)/(1.2^i); end;
A=Q*R; [Q,R]=qr(A);

v=A; q=Q; a=A; % Classical GS
for j=1:n;
    for k=1:(j-1);
        v(:,j)=v(:,j)-q(:,k)*(q(:,k)'*a(:,j)); end;
        q(:,j)=v(:,j)/norm(v(:,j));
    end;
    qc=q;

v=A; q=Q; a=A; % Modified GS
for j=1:n;
    for k=1:(j-1);
        v(:,j)=v(:,j)-q(:,k)*(q(:,k)'*v(:,j)); end;
        q(:,j)=v(:,j)/norm(v(:,j));
    end;
    qm=q;
```

Classical & Modified GS: Notes

```
v=A; q=Q; a=A;      % Classical GS, text
for k=1:n;
    q(:,k)=a(:,k);
    for j=1:k-1; r(j,k)=q(:,j)'*a(:,k);
        q(:,k)=q(:,k)-r(j,k)*q(:,j); end;
    r(k,k)=norm(q(:,k));
    q(:,k)=q(:,k) / r(k,k);
end;
qct=q;

v=A; q=Q; a=A;      % Modified GS, text
for k=1:n;
    r(k,k)=norm(a(:,k));
    q(:,k)=a(:,k) / r(k,k);
    for j=k+1:n; r(k,j)=q(:,k)'*a(:,j);
        a(:,j)=a(:,j)-r(k,j)*q(:,k); end;
end;
qmt=q;
```

Householder Transformations: Notes

```
a=A;      % Householder, per textbook
I=eye(n); QH=I;
for k=1:n;
    v=a(:,k); v(1:k-1)=0;
    alphak=-sign(a(k,k))*norm(v);
    v(k)=v(k)-alphak;
    betak=v'*v;
    for j=k:n; gammaj=v'*a(:,j);
        a(:,j)=a(:,j)-(2*gammaj/betak)*v; end;
    QH=QH-(2/betak)*v*(v'*QH);
end;
QH=QH'; qht=QH;

nq =norm(Q'*Q-eye(n));
nc =norm(qc'*qc-eye(n));
nm =norm(qm'*qm-eye(n));
nct=norm(qct'*qct-eye(n));
nmt=norm(qmt'*qmt-eye(n));
nht=norm(qht'*qht-eye(n));

[nc nct nm nmt nht nq]
```

```
>> house
ans =
1.6971e-03  1.6971e-03  4.5031e-07  4.5031e-07  1.4232e-15  1.0825e-15
```

Orthogonal Transformations

- We seek alternative method that avoids numerical difficulties of normal equations
- We need numerically robust transformation that produces easier problem without changing solution
- What kind of transformation leaves least squares solution unchanged?
- Square matrix Q is *orthogonal* if $Q^T Q = I$
- Multiplication of vector by orthogonal matrix preserves Euclidean norm

$$\|Q\mathbf{v}\|_2^2 = (Q\mathbf{v})^T Q\mathbf{v} = \mathbf{v}^T Q^T Q\mathbf{v} = \mathbf{v}^T \mathbf{v} = \|\mathbf{v}\|_2^2$$

- Thus, multiplying both sides of least squares problem by orthogonal matrix does not change its solution



Triangular Least Squares Problems

- As with square linear systems, suitable target in simplifying least squares problems is triangular form
- Upper triangular overdetermined ($m > n$) least squares problem has form

$$\begin{bmatrix} \mathbf{R} \\ \mathbf{O} \end{bmatrix} \mathbf{x} \cong \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{bmatrix}$$

where \mathbf{R} is $n \times n$ upper triangular and \mathbf{b} is partitioned similarly

- Residual is

$$\|\mathbf{r}\|_2^2 = \|\mathbf{b}_1 - \mathbf{R}\mathbf{x}\|_2^2 + \|\mathbf{b}_2\|_2^2$$



Triangular Least Squares Problems, continued

- We have no control over second term, $\|b_2\|_2^2$, but first term becomes zero if x satisfies $n \times n$ triangular system

$$Rx = b_1$$

which can be solved by back-substitution

- Resulting x is least squares solution, and minimum sum of squares is

$$\|r\|_2^2 = \|b_2\|_2^2$$

- So our strategy is to transform general least squares problem to triangular form using orthogonal transformation so that least squares solution is preserved



QR Factorization

- Given $m \times n$ matrix A , with $m > n$, we seek $m \times m$ orthogonal matrix Q such that

$$A = Q \begin{bmatrix} R \\ O \end{bmatrix}$$

where R is $n \times n$ and upper triangular

- Linear least squares problem $Ax \cong b$ is then transformed into triangular least squares problem

$$Q^T A x = \begin{bmatrix} R \\ O \end{bmatrix} x \cong \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = Q^T b$$

which has same solution, since

$$\|r\|_2^2 = \|b - Ax\|_2^2 = \|b - Q \begin{bmatrix} R \\ O \end{bmatrix} x\|_2^2 = \|Q^T b - \begin{bmatrix} R \\ O \end{bmatrix} x\|_2^2$$



Orthogonal Bases

- If we partition $m \times m$ orthogonal matrix $\mathbf{Q} = [\mathbf{Q}_1 \ \mathbf{Q}_2]$, where \mathbf{Q}_1 is $m \times n$, then

$$\mathbf{A} = \mathbf{Q} \begin{bmatrix} \mathbf{R} \\ \mathbf{O} \end{bmatrix} = [\mathbf{Q}_1 \ \mathbf{Q}_2] \begin{bmatrix} \mathbf{R} \\ \mathbf{O} \end{bmatrix} = \boxed{\mathbf{Q}_1 \mathbf{R}}$$

is called *reduced* QR factorization of \mathbf{A}

- Columns of \mathbf{Q}_1 are orthonormal basis for $\text{span}(\mathbf{A})$, and columns of \mathbf{Q}_2 are orthonormal basis for $\text{span}(\mathbf{A})^\perp$
- $\mathbf{Q}_1 \mathbf{Q}_1^T$ is orthogonal projector onto $\text{span}(\mathbf{A})$
- Solution to least squares problem $\mathbf{Ax} \cong \mathbf{b}$ is given by solution to square system

$$\mathbf{Q}_1^T \mathbf{Ax} = \boxed{\mathbf{Rx} = \mathbf{c}_1} = \mathbf{Q}_1^T \mathbf{b}$$



QR for Solving Least Squares

- Start with $Ax \approx b$

$$Q \begin{bmatrix} R \\ O \end{bmatrix} x \approx b$$

$$Q^T Q \begin{bmatrix} R \\ O \end{bmatrix} x = \begin{bmatrix} R \\ O \end{bmatrix} x \approx Q^T b = [Q_1 Q_2] b = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}.$$

- Define the residual, $r := b - y = b - Ax$

$$\begin{aligned} \|r\| &= \|b - Ax\| \\ &= \|Q^T(b - Ax)\| \\ &= \left\| \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} - \begin{pmatrix} Rx \\ O \end{pmatrix} \right\| \\ &= \left\| \begin{pmatrix} c_1 - Rx \\ c_2 \end{pmatrix} \right\| \end{aligned}$$

$$\|r\|^2 = \|c_1 - Rx\|^2 + \|c_2\|^2$$

- Norm of residual is minimized when $Rx = c_1 = Q_1^T b$, and takes on value $\|r\| = \|c_2\|$.

Computing QR Factorization

- To compute QR factorization of $m \times n$ matrix A , with $m > n$, we annihilate subdiagonal entries of successive columns of A , eventually reaching upper triangular form
- Similar to LU factorization by Gaussian elimination, but use orthogonal transformations instead of elementary elimination matrices
- Possible methods include
 - Householder transformations
 - Givens rotations
 - Gram-Schmidt orthogonalization



Method 2: Householder Transformations

Householder Transformations

- *Householder transformation* has form

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

for nonzero vector \mathbf{v}

- H is orthogonal and symmetric: $H = H^T = H^{-1}$
- Given vector \mathbf{a} , we want to choose \mathbf{v} so that

$$H\mathbf{a} = \begin{bmatrix} \alpha \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \alpha \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \alpha \mathbf{e}_1$$

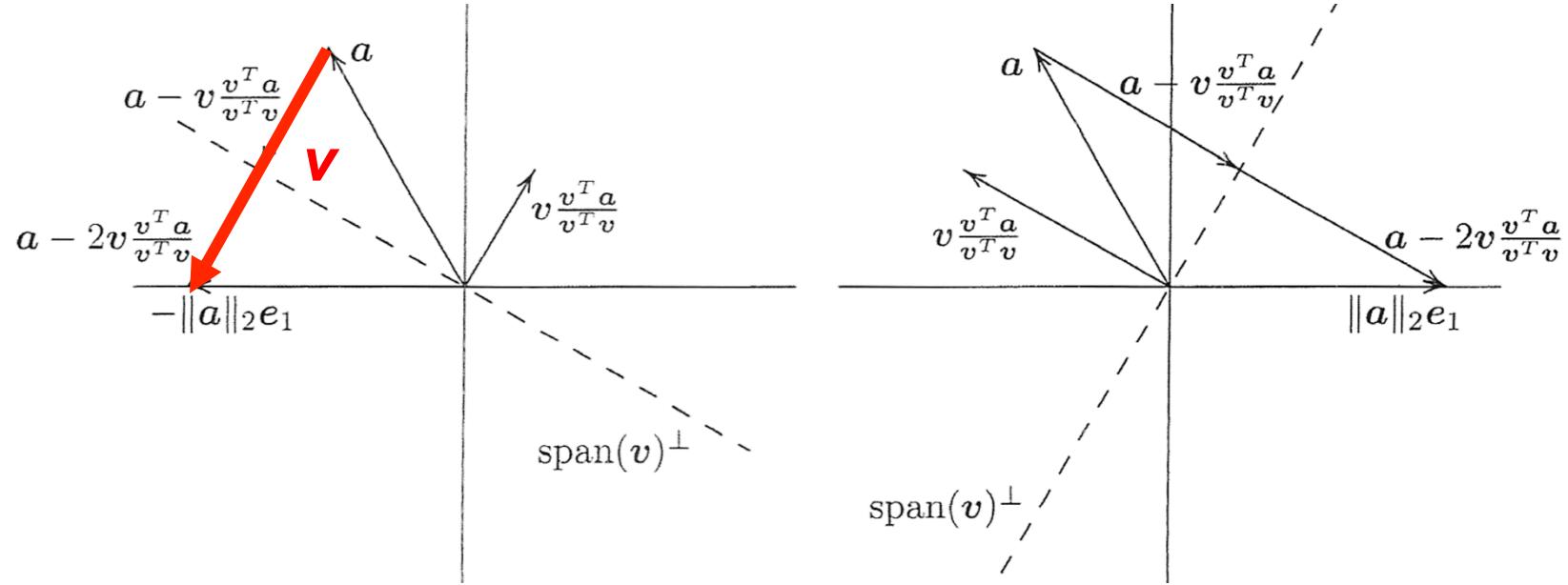
- Substituting into formula for H , we can take

$$\mathbf{v} = \mathbf{a} - \alpha \mathbf{e}_1$$

and $\alpha = \pm \|\mathbf{a}\|_2$, with sign chosen to avoid cancellation



Householder Reflection



Recall, $I - \underline{v}(\underline{v}^T \underline{v})^{-1} \underline{v}^T$ is a projector onto $R^\perp(\underline{v})$.

Therefore, $I - 2\underline{v}(\underline{v}^T \underline{v})^{-1} \underline{v}^T$ will reflect the transformed vector past $R^\perp(\underline{v})$.

With Householder, choose \underline{v} such that the reflected vector has all entries below the k th one set to zero.

Also, choose \underline{v} to avoid cancellation in k th component.

Householder Derivation

$$H\mathbf{a} = \mathbf{a} - 2 \frac{\mathbf{v}^T \mathbf{a}}{\mathbf{v}^T \mathbf{v}} \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_m \end{pmatrix} = \begin{pmatrix} \alpha \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

$\mathbf{v} = \mathbf{a} - \alpha \mathbf{e}_1 \leftarrow$ Choose α to get desired cancellation.

$$\mathbf{v}^T \mathbf{a} = \mathbf{a}^T \mathbf{a} - \alpha a_1, \quad \mathbf{v}^T \mathbf{v} = \mathbf{a}^T \mathbf{a} - 2\alpha a_1 + \alpha^2$$

$$\begin{aligned} H\mathbf{a} &= \mathbf{a} - 2 \frac{(\mathbf{a}^T \mathbf{a} - \alpha a_1)}{\mathbf{a}^T \mathbf{a} - 2\alpha a_1 + \alpha^2} (\mathbf{a} - \alpha \mathbf{e}_1) \\ &= \mathbf{a} - 2 \frac{||\mathbf{a}||^2 \pm ||\mathbf{a}|| a_1}{2||\mathbf{a}||^2 \pm 2||\mathbf{a}|| a_1} (\mathbf{a} - \alpha \mathbf{e}_1) \\ &= \mathbf{a} - (\mathbf{a} - \alpha \mathbf{e}_1) = \alpha \mathbf{e}_1. \end{aligned}$$

Choose $\alpha = -\text{sign}(a_1) ||\mathbf{a}|| = -\left(\frac{a_1}{|a_1|}\right) ||\mathbf{a}||$.

Example: Householder Transformation

- If $\mathbf{a} = [2 \ 1 \ 2]^T$, then we take

$$\mathbf{v} = \mathbf{a} - \alpha \mathbf{e}_1 = \begin{bmatrix} 2 \\ 1 \\ 2 \end{bmatrix} - \alpha \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \\ 2 \end{bmatrix} - \begin{bmatrix} \alpha \\ 0 \\ 0 \end{bmatrix}$$

where $\alpha = \pm \|\mathbf{a}\|_2 = \pm 3$

- Since a_1 is positive, we choose negative sign for α to avoid

cancellation, so $\mathbf{v} = \begin{bmatrix} 2 \\ 1 \\ 2 \end{bmatrix} - \begin{bmatrix} -3 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 5 \\ 1 \\ 2 \end{bmatrix}$

- To confirm that transformation works,

$$H\mathbf{a} = \mathbf{a} - 2 \frac{\mathbf{v}^T \mathbf{a}}{\mathbf{v}^T \mathbf{v}} \mathbf{v} = \begin{bmatrix} 2 \\ 1 \\ 2 \end{bmatrix} - 2 \frac{15}{30} \begin{bmatrix} 5 \\ 1 \\ 2 \end{bmatrix} = \begin{bmatrix} -3 \\ 0 \\ 0 \end{bmatrix}$$



Householder QR Factorization

- To compute QR factorization of A , use Householder transformations to annihilate subdiagonal entries of each successive column
- Each Householder transformation is applied to entire matrix, but does not affect prior columns, so zeros are preserved
- In applying Householder transformation H to arbitrary vector u ,

$$Hu = \left(I - 2 \frac{vv^T}{v^Tv} \right) u = u - \left(2 \frac{v^Tu}{v^Tv} \right) v$$

which is much cheaper than general matrix-vector multiplication and requires only vector v , not full matrix H



Householder QR Factorization, continued

- Process just described produces factorization

$$H_n \cdots H_1 A = \begin{bmatrix} R \\ O \end{bmatrix}$$

where R is $n \times n$ and upper triangular

- If $Q = H_1 \cdots H_n$, then $A = Q \begin{bmatrix} R \\ O \end{bmatrix}$
- To preserve solution of linear least squares problem, right-hand side b is transformed by same sequence of Householder transformations
- Then solve triangular least squares problem $\begin{bmatrix} R \\ O \end{bmatrix} x \cong Q^T b$



Householder QR Factorization, continued

- For solving linear least squares problem, product Q of Householder transformations need not be formed explicitly
- R can be stored in upper triangle of array initially containing A
- Householder vectors v can be stored in (now zero) lower triangular portion of A (almost)
- Householder transformations most easily applied in this form anyway



Example: Householder QR Factorization

- For polynomial data-fitting example given previously, with

$$\mathbf{A} = \begin{bmatrix} 1 & -1.0 & 1.0 \\ 1 & -0.5 & 0.25 \\ 1 & 0.0 & 0.0 \\ 1 & 0.5 & 0.25 \\ 1 & 1.0 & 1.0 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 1.0 \\ 0.5 \\ 0.0 \\ 0.5 \\ 2.0 \end{bmatrix}$$

- Householder vector v_1 for annihilating subdiagonal entries of first column of A is

$$v_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} -2.236 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 3.236 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$



Example, continued

- Applying resulting Householder transformation H_1 yields transformed matrix and right-hand side

$$H_1 A = \begin{bmatrix} -2.236 & 0 & -1.118 \\ 0 & -0.191 & -0.405 \\ 0 & 0.309 & -0.655 \\ 0 & 0.809 & -0.405 \\ 0 & 1.309 & 0.345 \end{bmatrix}, \quad H_1 b = \begin{bmatrix} -1.789 \\ -0.362 \\ -0.862 \\ -0.362 \\ 1.138 \end{bmatrix}$$

- Householder vector v_2 for annihilating subdiagonal entries of second column of $H_1 A$ is

$$v_2 = \begin{bmatrix} 0 \\ -0.191 \\ 0.309 \\ 0.809 \\ 1.309 \end{bmatrix} - \begin{bmatrix} 0 \\ 1.581 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ -1.772 \\ 0.309 \\ 0.809 \\ 1.309 \end{bmatrix}$$



Example, continued

- Applying resulting Householder transformation H_2 yields

$$H_2 H_1 A = \begin{bmatrix} -2.236 & 0 & -1.118 \\ 0 & 1.581 & 0 \\ 0 & 0 & -0.725 \\ 0 & 0 & -0.589 \\ 0 & 0 & 0.047 \end{bmatrix}, \quad H_2 H_1 b = \begin{bmatrix} -1.789 \\ 0.632 \\ -1.035 \\ -0.816 \\ 0.404 \end{bmatrix}$$

- Householder vector v_3 for annihilating subdiagonal entries of third column of $H_2 H_1 A$ is

$$v_3 = \begin{bmatrix} 0 \\ 0 \\ -0.725 \\ -0.589 \\ 0.047 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0.935 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -1.660 \\ -0.589 \\ 0.047 \end{bmatrix}$$



Example, continued

- Applying resulting Householder transformation \mathbf{H}_3 yields

$$\mathbf{H}_3 \mathbf{H}_2 \mathbf{H}_1 \mathbf{A} = \begin{bmatrix} -2.236 & 0 & -1.118 \\ 0 & 1.581 & 0 \\ 0 & 0 & 0.935 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{H}_3 \mathbf{H}_2 \mathbf{H}_1 \mathbf{b} = \begin{bmatrix} -1.789 \\ 0.632 \\ 1.336 \\ 0.026 \\ 0.337 \end{bmatrix}$$

- Now solve upper triangular system $\mathbf{R}\mathbf{x} = \mathbf{c}_1$ by back-substitution to obtain $\mathbf{x} = [0.086 \quad 0.400 \quad 1.429]^T$



kth Householder Transformation (Reflection)

kth column

↓

$$A_k = \begin{pmatrix} x & x & x & x & x & x \\ x & x & x & x & x & x \\ x & x & x & x & x & x \\ x & x & x & x & x & x \\ x & x & x & x & x & x \\ x & x & x & x & x & x \end{pmatrix} \quad \leftarrow k\text{th row}$$

Note: $H_k \underline{a}_j = \underline{a}_j$ for $j < k$.

Householder Transformations

$$H_1 A = \begin{pmatrix} x & x & x \\ x & x & \\ x & x & \\ x & x & \end{pmatrix}, \quad H_1 \mathbf{b} \longrightarrow \mathbf{b}^{(1)} = \begin{pmatrix} x \\ x \\ x \\ x \end{pmatrix}$$

$$H_2 H_1 A = \begin{pmatrix} x & x & x \\ & x & x \\ & & x \\ & & x \end{pmatrix}, \quad H_2 \mathbf{b}^{(1)} \longrightarrow \mathbf{b}^{(2)} = \begin{pmatrix} x \\ x \\ x \\ x \end{pmatrix}$$

$$H_3 H_2 H_1 A = \begin{pmatrix} x & x & x \\ & x & x \\ & & x \end{pmatrix}, \quad H_3 \mathbf{b}^{(2)} \longrightarrow \mathbf{b}^{(3)} = \begin{pmatrix} \mathbf{c}_1 \\ \mathbf{c}_2 \end{pmatrix}.$$

Questions: How does $H_3 H_2 H_1$ relate to Q or Q_1 ??

What is Q in this case?

Note: Householder Procedure

$$H_3 H_2 H_1 A = \begin{pmatrix} R \\ O \end{pmatrix}, \quad A = Q \begin{pmatrix} R \\ O \end{pmatrix}.$$

$$H_3 H_2 H_1 A = Q^{-1} Q \begin{pmatrix} R \\ O \end{pmatrix} = Q^T Q \begin{pmatrix} R \\ O \end{pmatrix} = Q^T A.$$

$$Q^T = H_3 H_2 H_1$$

$$Q = H_1^T H_2^T H_3^T = H_1 H_2 H_3.$$

- Technically, we usually don't need Q nor the action of Q .
- Just need the *action* of Q^T on a matrix or vector.
- Never form Q or H_k (large, $m \times m$ matrices), just apply H_k to vectors:

$$H_k \underline{u} = \underline{u} - 2 \left(\frac{\underline{v}_k^T \underline{u}}{\underline{v}_k^T \underline{v}_k} \right) \underline{v}_k.$$

Method 3: Givens Rotations

Givens Rotations

- *Givens rotations* introduce zeros one at a time
- Given vector $[a_1 \quad a_2]^T$, choose scalars c and s so that

$$\begin{bmatrix} c & s \\ -s & c \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} \alpha \\ 0 \end{bmatrix}$$

with $c^2 + s^2 = 1$, or equivalently, $\alpha = \sqrt{a_1^2 + a_2^2}$

- Previous equation can be rewritten

$$\begin{bmatrix} a_1 & a_2 \\ a_2 & -a_1 \end{bmatrix} \begin{bmatrix} c \\ s \end{bmatrix} = \begin{bmatrix} \alpha \\ 0 \end{bmatrix}$$

- Gaussian elimination yields triangular system

$$\begin{bmatrix} a_1 & a_2 \\ 0 & -a_1 - a_2^2/a_1 \end{bmatrix} \begin{bmatrix} c \\ s \end{bmatrix} = \begin{bmatrix} \alpha \\ -\alpha a_2/a_1 \end{bmatrix}$$



Givens Rotations, continued

- Back-substitution then gives

$$s = \frac{\alpha a_2}{a_1^2 + a_2^2} \quad \text{and} \quad c = \frac{\alpha a_1}{a_1^2 + a_2^2}$$

- Finally, $c^2 + s^2 = 1$, or $\alpha = \sqrt{a_1^2 + a_2^2}$, implies

$$c = \frac{a_1}{\sqrt{a_1^2 + a_2^2}}$$

and

$$s = \frac{a_2}{\sqrt{a_1^2 + a_2^2}}$$

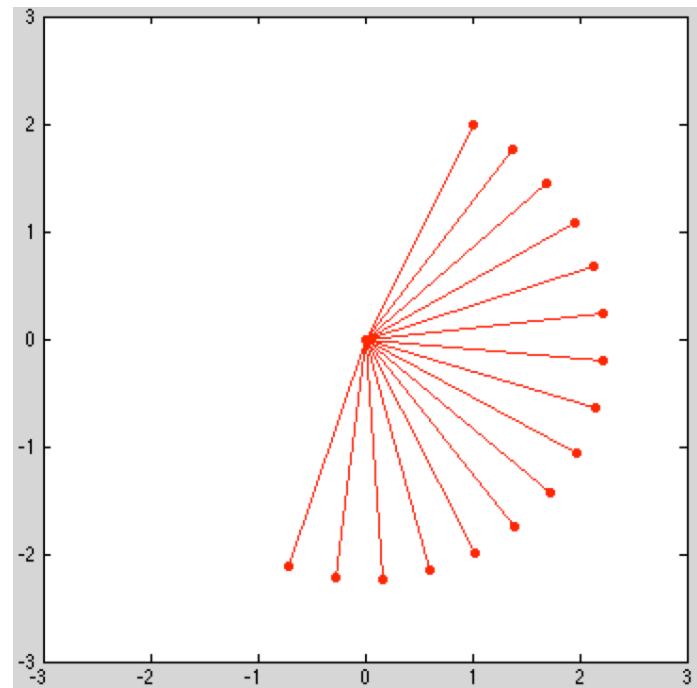


2 x 2 Rotation Matrices

```
% Rotation Matrix Demo

X=[ 0  1 ; ...      %   [ x0  x1
    0  2 ];           %       y0  y1  ]

hold off
X0=X;
for t=0:.2:3;
    c=cos(t); s=sin(t);
    R= [ c  s ; -s c ];
    X=R*X0;
    x=X(1,:); y=X(2,:);
    plot(x,y,'r.-');
    axis equal; axis ( [-3 3 -3 3] )
    hold on
    pause(.3)
end;
```



Example: Givens Rotation

- Let $\mathbf{a} = [4 \quad 3]^T$
- To annihilate second entry we compute cosine and sine

$$c = \frac{a_1}{\sqrt{a_1^2 + a_2^2}} = \frac{4}{5} = 0.8 \quad \text{and} \quad s = \frac{a_2}{\sqrt{a_1^2 + a_2^2}} = \frac{3}{5} = 0.6$$

- Rotation is then given by

$$\mathbf{G} = \begin{bmatrix} c & s \\ -s & c \end{bmatrix} = \begin{bmatrix} 0.8 & 0.6 \\ -0.6 & 0.8 \end{bmatrix}$$

- To confirm that rotation works,

$$\mathbf{G}\mathbf{a} = \begin{bmatrix} 0.8 & 0.6 \\ -0.6 & 0.8 \end{bmatrix} \begin{bmatrix} 4 \\ 3 \end{bmatrix} = \begin{bmatrix} 5 \\ 0 \end{bmatrix}$$



Givens QR Factorization

- More generally, to annihilate selected component of vector in n dimensions, rotate target component with another component

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & c & 0 & s & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & -s & 0 & c & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix} = \begin{bmatrix} a_1 \\ \alpha \\ a_3 \\ 0 \\ a_5 \end{bmatrix}$$

- By systematically annihilating successive entries, we can reduce matrix to upper triangular form using sequence of Givens rotations
- Each rotation is orthogonal, so their product is orthogonal, producing QR factorization



Givens QR Factorization

- Straightforward implementation of Givens method requires about 50% more work than Householder method, and also requires more storage, since each rotation requires two numbers, c and s , to define it
- These disadvantages can be overcome, but requires more complicated implementation
- Givens can be advantageous for computing QR factorization when many entries of matrix are already zero, since those annihilations can then be skipped

< interactive example >



Givens QR

- ❑ A particularly attractive use of Givens QR is when A is upper Hessenberg – A is upper triangular with one additional nonzero diagonal below the main one: $A_{ij} = 0$ if $i > j+1$

$$\begin{matrix} 0.1967 & 0.2973 & 0.0899 & 0.3381 & 0.5261 & 0.3965 & 0.1279 & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0.0934 & 0.0620 & 0.0809 & 0.2940 & 0.7297 & 0.0616 & 0.5495 & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0.2982 & 0.7772 & 0.7463 & 0.7073 & 0.7802 & 0.4852 & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 0.9051 & 0.0103 & 0.7814 & 0.3376 & 0.8905 & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & 0.0484 & 0.2880 & 0.6079 & 0.7990 & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & 0 & 0.6925 & 0.7413 & 0.7343 & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & 0 & 0 & 0.1048 & 0.0513 & \cdot & \cdot & \cdot & \cdot & \cdot \end{matrix}$$

- ❑ In this case, we require Givens row operations applied only n times, instead of $O(n^2)$ times.
- ❑ Work for Givens is thus $O(n^2)$, vs. $O(n^3)$ for Householder.
- ❑ Upper Hessenberg matrices arise when computing eigenvalues.

Successive Givens Rotations

As with Householder transformations, we apply successive Givens rotations, G_1, G_2 , etc.

$$G_1 A = \begin{pmatrix} x & x & x \\ x & x & x \\ x & x & x \\ x & x & x \end{pmatrix}, \quad H_1 \mathbf{b} \longrightarrow \mathbf{b}^{(1)} = \begin{pmatrix} x \\ x \\ x \\ x \end{pmatrix}$$

$$G_2 G_1 A = \begin{pmatrix} x & x & x \\ x & x & x \\ x & x & x \\ x & x & x \end{pmatrix}, \quad G_2 \mathbf{b}^{(1)} \longrightarrow \mathbf{b}^{(2)} = \begin{pmatrix} x \\ x \\ x \\ x \end{pmatrix}$$

$$G_3 G_2 G_1 A = \begin{pmatrix} x & x & x \\ x & x & x \\ x & x & x \\ x & x & x \end{pmatrix}, \quad G_3 \mathbf{b}^{(2)} \longrightarrow \mathbf{b}^{(3)} = \begin{pmatrix} x \\ x \\ x \\ x \end{pmatrix}$$

- How many Givens rotations (total) are required for the $m \times n$ case?
- How does $\dots G_3 G_2 G_1$ relate to Q or Q_1 ?
- What is Q in this case?

Rank Deficiency

- If $\text{rank}(\mathbf{A}) < n$, then QR factorization still exists, but yields singular upper triangular factor \mathbf{R} , and multiple vectors \mathbf{x} give minimum residual norm
- Common practice selects minimum residual solution \mathbf{x} having smallest norm
- Can be computed by QR factorization with column pivoting or by singular value decomposition (SVD)
- Rank of matrix is often not clear cut in practice, so relative tolerance is used to determine rank



Example: Near Rank Deficiency

- Consider 3×2 matrix

$$A = \begin{bmatrix} 0.641 & 0.242 \\ 0.321 & 0.121 \\ 0.962 & 0.363 \end{bmatrix}$$

- Computing QR factorization,

$$R = \begin{bmatrix} 1.1997 & 0.4527 \\ 0 & 0.0002 \end{bmatrix}$$

- R is extremely close to singular (exactly singular to 3-digit accuracy of problem statement)
- If R is used to solve linear least squares problem, result is highly sensitive to perturbations in right-hand side
- For practical purposes, $\text{rank}(A) = 1$ rather than 2, because columns are nearly linearly dependent



QR with Column Pivoting

- Instead of processing columns in natural order, select for reduction at each stage column of remaining unreduced submatrix having maximum Euclidean norm
- If $\text{rank}(\mathbf{A}) = k < n$, then after k steps, norms of remaining unreduced columns will be zero (or “negligible” in finite-precision arithmetic) below row k
- Yields orthogonal factorization of form

$$\mathbf{Q}^T \mathbf{A} \mathbf{P} = \begin{bmatrix} \mathbf{R} & \mathbf{S} \\ \mathbf{O} & \mathbf{O} \end{bmatrix}$$

where \mathbf{R} is $k \times k$, upper triangular, and nonsingular, and permutation matrix \mathbf{P} performs column interchanges



QR with Column Pivoting, continued

- *Basic solution* to least squares problem $Ax \cong b$ can now be computed by solving triangular system $Rz = c_1$, where c_1 contains first k components of $Q^T b$, and then taking

$$x = P \begin{bmatrix} z \\ 0 \end{bmatrix}$$

- *Minimum-norm solution* can be computed, if desired, at expense of additional processing to annihilate S
- $\text{rank}(A)$ is usually unknown, so rank is determined by monitoring norms of remaining unreduced columns and terminating factorization when maximum value falls below chosen tolerance



Comparison of Methods

- Forming normal equations matrix $A^T A$ requires about $n^2m/2$ multiplications, and solving resulting symmetric linear system requires about $n^3/6$ multiplications
- Solving least squares problem using Householder QR factorization requires about $mn^2 - n^3/3$ multiplications
- If $m \approx n$, both methods require about same amount of work
- If $m \gg n$, Householder QR requires about twice as much work as normal equations
- Cost of SVD is proportional to $mn^2 + n^3$, with proportionality constant ranging from 4 to 10, depending on algorithm used



Comparison of Methods, continued

- Normal equations method produces solution whose relative error is proportional to $[\text{cond}(\mathbf{A})]^2$
- Required Cholesky factorization can be expected to break down if $\text{cond}(\mathbf{A}) \approx 1/\sqrt{\epsilon_{\text{mach}}}$ or worse
- Householder method produces solution whose relative error is proportional to

$$\text{cond}(\mathbf{A}) + \|\mathbf{r}\|_2 [\text{cond}(\mathbf{A})]^2$$

which is best possible, since this is inherent sensitivity of solution to least squares problem

- Householder method can be expected to break down (in back-substitution phase) only if $\text{cond}(\mathbf{A}) \approx 1/\epsilon_{\text{mach}}$ or worse



Comparison of Methods, continued

- Householder is more accurate and more broadly applicable than normal equations
- These advantages may not be worth additional cost, however, when problem is sufficiently well conditioned that normal equations provide sufficient accuracy
- For rank-deficient or nearly rank-deficient problems, Householder with column pivoting can produce useful solution when normal equations method fails outright
- SVD is even more robust and reliable than Householder, but substantially more expensive



Singular Value Decomposition

- Singular value decomposition (SVD) of $m \times n$ matrix A has form

$$A = U\Sigma V^T$$

where U is $m \times m$ orthogonal matrix, V is $n \times n$ orthogonal matrix, and Σ is $m \times n$ diagonal matrix, with

$$\sigma_{ij} = \begin{cases} 0 & \text{for } i \neq j \\ \sigma_i \geq 0 & \text{for } i = j \end{cases}$$

- Diagonal entries σ_i , called *singular values* of A , are usually ordered so that $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_n$
- Columns u_i of U and v_i of V are called left and right *singular vectors*



SVD of Rectangular Matrix A

$$A = U \Sigma V^T$$

- $A = U\Sigma V^T$ is $m \times n$.
- U is $m \times m$, orthogonal.
- Σ is $m \times n$, diagonal, $\sigma_i > 0$.
- V is $n \times n$, orthogonal.

Example: SVD

- SVD of $A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \end{bmatrix}$ is given by $U\Sigma V^T =$

$$\begin{bmatrix} .141 & .825 & -.420 & -.351 \\ .344 & .426 & .298 & .782 \\ .547 & .0278 & .664 & -.509 \\ .750 & -.371 & -.542 & .0790 \end{bmatrix} \begin{bmatrix} 25.5 & 0 & 0 \\ 0 & 1.29 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} .504 & .574 & .644 \\ -.761 & -.057 & .646 \\ .408 & -.816 & .408 \end{bmatrix}$$

In square matrix case, $U \Sigma V^T$ closely related to eigenpair, $X \Lambda X^{-1}$



Applications of SVD

- *Minimum norm solution* to $Ax \cong b$ is given by

$$x = \sum_{\sigma_i \neq 0} \frac{u_i^T b}{\sigma_i} v_i$$

For ill-conditioned or rank deficient problems, “small” singular values can be omitted from summation to stabilize solution

- *Euclidean matrix norm*: $\|A\|_2 = \sigma_{\max}$
- *Euclidean condition number of matrix*: $\text{cond}(A) = \frac{\sigma_{\max}}{\sigma_{\min}}$
- *Rank of matrix*: number of nonzero singular values



SVD for Linear Least Squares Problem: $A = U\Sigma V^T$

$$A\underline{x} \approx \underline{b}$$

$$U\Sigma V^T \approx \underline{b}$$

$$U^T U\Sigma V^T \approx U^T \underline{b}$$

$$\Sigma V^T \approx U^T \underline{b}$$

$$\begin{bmatrix} \tilde{R} \\ O \end{bmatrix} \underline{x} \approx \begin{pmatrix} \underline{c}_1 \\ \underline{c}_2 \end{pmatrix}$$

$$\tilde{R}\underline{x} = \underline{c}_1$$

$$\underline{x} = \sum_{j=1}^n \underline{v}_j \frac{1}{\sigma_j} (\underline{c}_1)_j = \sum_{j=1}^n \underline{v}_j \frac{1}{\sigma_j} \underline{u}_j^T \underline{b}$$

SVD for Linear Least Squares Problem: $A = U\Sigma V^T$

- SVD can also handle the rank deficient case.
- If there are only k singular values $\sigma_j > \epsilon$ then take only the first k contributions.

$$\underline{x} = \sum_{j=1}^k v_j \frac{1}{\sigma_j} u_j^T \underline{b}$$

Pseudoinverse

- Define pseudoinverse of scalar σ to be $1/\sigma$ if $\sigma \neq 0$, zero otherwise
- Define pseudoinverse of (possibly rectangular) diagonal matrix by transposing and taking scalar pseudoinverse of each entry
- Then *pseudoinverse* of general real $m \times n$ matrix A is given by

$$A^+ = V\Sigma^+U^T$$

- Pseudoinverse always exists whether or not matrix is square or has full rank
- If A is square and nonsingular, then $A^+ = A^{-1}$
- In all cases, minimum-norm solution to $Ax \cong b$ is given by $x = A^+ b$



Orthogonal Bases

- SVD of matrix, $A = U\Sigma V^T$, provides orthogonal bases for subspaces relevant to A
- Columns of U corresponding to nonzero singular values form orthonormal basis for $\text{span}(A)$
- Remaining columns of U form orthonormal basis for orthogonal complement $\text{span}(A)^\perp$
- Columns of V corresponding to zero singular values form orthonormal basis for null space of A
- Remaining columns of V form orthonormal basis for orthogonal complement of null space of A



Lower-Rank Matrix Approximation

- Another way to write SVD is

$$\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^T = \sigma_1 \mathbf{E}_1 + \sigma_2 \mathbf{E}_2 + \cdots + \sigma_n \mathbf{E}_n$$

with $\mathbf{E}_i = \mathbf{u}_i \mathbf{v}_i^T$

- \mathbf{E}_i has rank 1 and can be stored using only $m + n$ storage locations
- Product $\mathbf{E}_i \mathbf{x}$ can be computed using only $m + n$ multiplications
- Condensed approximation to \mathbf{A} is obtained by omitting from summation terms corresponding to small singular values
- Approximation using k largest singular values is closest matrix of rank k to \mathbf{A}
- Approximation is useful in image processing, data compression, information retrieval, cryptography, etc.



Low Rank Approximation to $A = U\Sigma V^T$

- Because of the diagonal form of Σ , we have

$$A = U\Sigma V^T = \sum_{j=1}^n \underline{u}_j \sigma_j \underline{v}_j^T$$

- A *rank k* approximation to A is given by

$$A \approx A_k := \sum_{j=1}^k \underline{u}_j \sigma_j \underline{v}_j^T$$

- A_k is the best approximation to A in the Frobenius norm,

$$\|M\|_F := \sqrt{m_{11}^2 + m_{21}^2 + \cdots + m_{mn}^2}$$

SVD for Image Compression

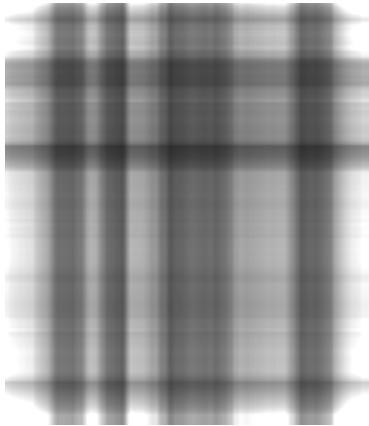
- ❑ If we view an image as an $m \times n$ matrix, we can use the SVD to generate a low-rank compressed version.
- ❑ Full image storage cost scales as $O(mn)$
- ❑ Compress image storage scales as $O(km) + O(kn)$, with $k < m$ or n .



$$A \approx A_k := \sum_{j=1}^k \underline{u}_j \sigma_j \underline{v}_j^T$$

Image Compression

- ❑ If we view an image as an $m \times n$ matrix, we can use the SVD to generate a low-rank compressed version.
- ❑ Full image storage cost scales as $O(mn)$
- ❑ Compress image storage scales as $O(km) + O(kn)$, with $k < m$ or n .

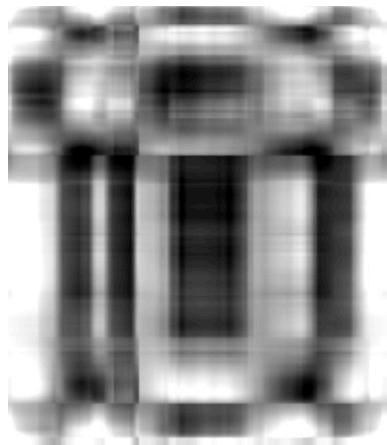
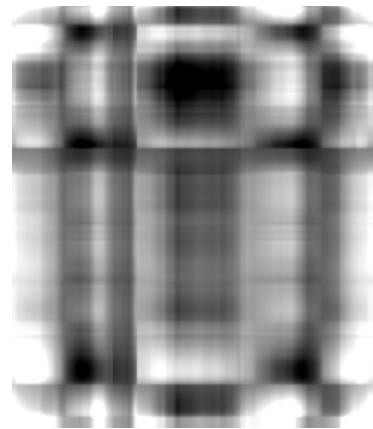
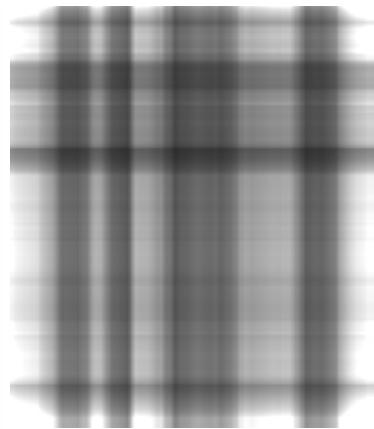
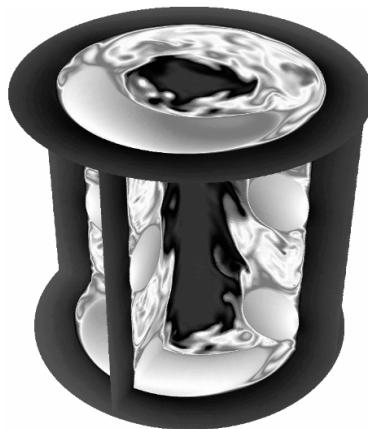


$k=1$

$$A \approx A_k := \sum_{j=1}^k \underline{u}_j \sigma_j \underline{v}_j^T$$

Image Compression

- ❑ If we view an image as an $m \times n$ matrix, we can use the SVD to generate a low-rank compressed version.
- ❑ Full image storage cost scales as $O(mn)$
- ❑ Compress image storage scales as $O(km) + O(kn)$, with $k < m$ or n .



Matlab code

```
[ X,A ]=imread('collins_img.gif'); [m,n]=size(X);  
  
Xo=X; imwrite(Xo,'oldfile.png')  
  
whos  
  
X=double(X); [ U,D,V ] = svd(X); % COMPUTE SVD  
  
  
X = 0*X;  
for k=1:min(m,n); k  
  
    X = X + U( :,k ) * D(k,k) * V( :,k )' ;  
  
    Xi = uint8(X); imwrite(Xi,'newfile.png'); spy(Xi>100);  
  
    pause  
  
end;
```

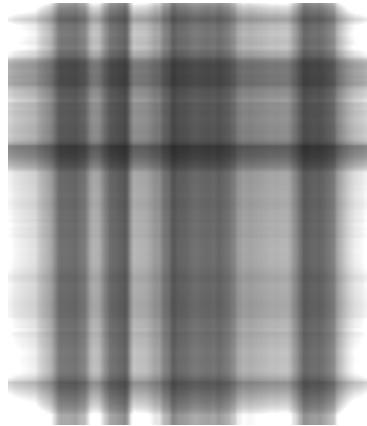
Image Compression

Compressed image storage scales as $O(km) + O(kn)$, with $k < m$ or n .

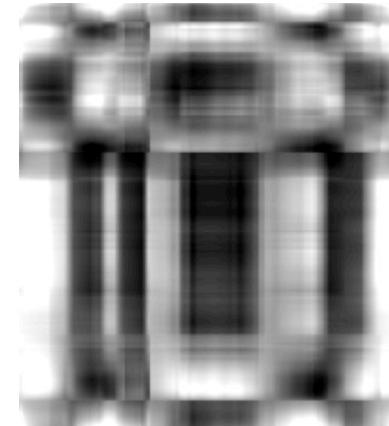
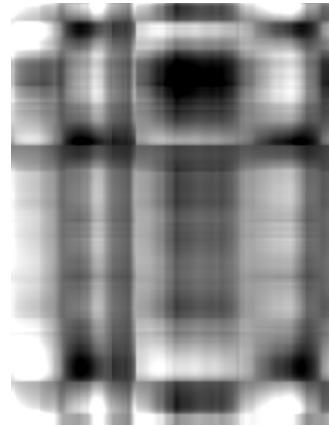
$k=1$



$k=2$



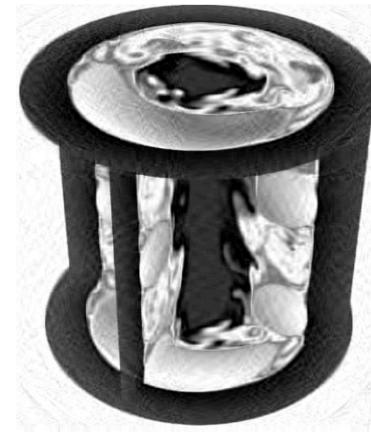
$k=3$



$k=10$



$k=20$



$k=50$

$(m=536, n=462)$

Low-Rank Approximations to Solutions of $\mathbf{A}\underline{x} = \underline{b}$

If $\sigma_1 \leq \sigma_2 \leq \dots \leq \sigma_n$,

$$\underline{x} \approx \sum_{j=1}^k \sigma_j^+ \underline{v}_j \underline{u}_j^T \underline{b}$$

- Other functions, aside from the inverse of the matrix, can also be approximated in this way, at relatively low cost, once the SVD is known.