

# MATH 350: Introduction to Computational Mathematics

## Chapter V: Least Squares Problems

Greg Fasshauer

Department of Applied Mathematics  
Illinois Institute of Technology

Spring 2011



# Outline

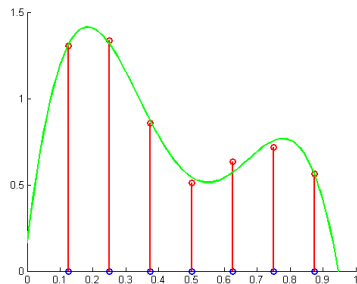
- 1 Motivation and Applications
- 2 The MATLAB Function `polyfit`
- 3 The QR Decomposition
- 4 The SVD



# Data Fitting

Earlier we discussed the problem of fitting a given set of data by interpolation. However, if the data contain some noise (such as measurement errors in a physical experiment), then we **may not want to exactly fit** the data.

Instead we may want to **approximate the data via a linear least squares (or linear regression) fit**.

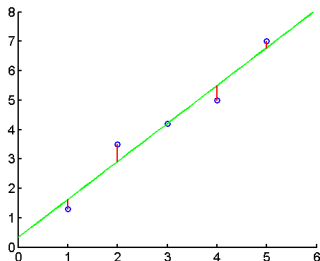


## Example

Fit the data

x	1	2	3	4	5
y	1.3	3.5	4.2	5.0	7.0

with a best fitting line.



## Idea

Minimize the sum of the squares of the vertical distances of line from the data points.

# The Normal Equations

Assume the line is of the form

$$L(x) = c_1x + c_2.$$

If we were to interpolate the data, i.e., enforce

$$L(x_i) = y_i, \quad i = 1, \dots, m,$$

we would get an **overdetermined** system of linear equations (as soon as  $m > 2$ ):

$$\mathbf{A}\mathbf{c} = \mathbf{y},$$

where

$$\mathbf{A} = \begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_m & 1 \end{bmatrix}, \quad \mathbf{c} = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}.$$



## The Normal Equations (cont.)

Minimizing the sums of the squares of the vertical distances from the line to the data points can be expressed as

$$\min_{c_1, c_2} \sum_{i=1}^m [(c_1 x_i + c_2) - y_i]^2 \iff \min_{c_1, c_2} \|\mathbf{A}\mathbf{c} - \mathbf{y}\|_2^2$$

Since the norm is related to an inner product by

$$\|\mathbf{v}\|_2 = \sqrt{\mathbf{v}^T \mathbf{v}}$$

we want to find the coefficient vector  $\mathbf{c}$  that minimizes

$$\begin{aligned} (\mathbf{A}\mathbf{c} - \mathbf{y})^T (\mathbf{A}\mathbf{c} - \mathbf{y}) &= (\mathbf{c}^T \mathbf{A}^T - \mathbf{y}^T) (\mathbf{A}\mathbf{c} - \mathbf{y}) \\ &= \mathbf{c}^T \mathbf{A}^T \mathbf{A}\mathbf{c} - \mathbf{c}^T \mathbf{A}^T \mathbf{y} - \mathbf{y}^T \mathbf{A}\mathbf{c} + \mathbf{y}^T \mathbf{y} \\ &= \mathbf{c}^T \mathbf{A}^T \mathbf{A}\mathbf{c} - 2\mathbf{c}^T \mathbf{A}^T \mathbf{y} + \mathbf{y}^T \mathbf{y}. \end{aligned}$$

The last equality holds because  $\mathbf{c}^T \mathbf{A}^T \mathbf{y} = \mathbf{y}^T \mathbf{A}\mathbf{c}$  is a scalar quantity (and therefore equal to its transpose).



## The Normal Equations (cont.)

From Calculus we know that a necessary<sup>1</sup> condition for the minimum is that the partials with respect to  $c_1$  and  $c_2$  are zero.

In vector notation this means

$$2A^T A \mathbf{c} - 2A^T \mathbf{y} = 0$$

or

$$A^T A \mathbf{c} = A^T \mathbf{y}.$$

This system of linear equations is known as the **normal equations**.

### Remark

*The normal equations characterize the **least squares solution** of the system  $A \mathbf{c} = \mathbf{y}$ . They are important for theoretical purposes and provide simple notation.*

*However, they are not recommended for serious computations!*

<sup>1</sup>and since the function is quadratic in  $\mathbf{c}$  also sufficient

# Do not compute with the normal equations!

## Example

Fit the data

x	10	10.2	10.4	10.6	10.8	11
y	0	0.004	0.016	0.036	0.064	0.1

with a quadratic polynomial  $Q(x) = c_1x^2 + c_2x + c_3$ .  
Note that the data actually come from the function

$$f(x) = \frac{x^2}{10} - 2x + 10,$$

so that we should be able to recover the function exactly.

The MATLAB script `LSQquad.m` illustrates what happens when we use the normal equations to solve the problem in single and double precision.



The derivation of the normal equations works **analogously** for any **linear approximation** of the form

$$P(x) = c_1 b_1(x) + c_2 b_2(x) + \dots + c_n b_n(x),$$

where  $\{b_1, \dots, b_n\}$  is some set of **basis functions** we want to compute the least squares fit with. This could be polynomials, trigonometric functions, logarithms, exponentials, etc.

The matrix  $A$  is a **Vandermonde-like matrix** of the form

$$A = \begin{bmatrix} b_1(x_1) & b_2(x_1) & \dots & b_n(x_1) \\ b_1(x_2) & b_2(x_2) & \dots & b_n(x_2) \\ \vdots & \vdots & & \vdots \\ b_1(x_m) & b_2(x_m) & \dots & b_n(x_m) \end{bmatrix}.$$

Usually, this is a **tall and skinny matrix**, i.e.,  $m > n$ . In this case we have **more data points than unknown coefficients**, i.e., an **over-determined linear system**.

The normal equations are always given by

$$A^T A \mathbf{c} = A^T \mathbf{y}.$$



## Question

How can we improve the conditioning of the least squares normal equations  $A^T A \mathbf{c} = A^T \mathbf{y}$ ?

## Answer

Use **orthogonal basis functions** to represent the least squares approximation.

## Example

Instead of using the monomial basis  $\{x^2, x, 1\}$  as we did in `LSQquad.m`, we can use the **orthogonal basis**

$$\left\{ \left(x - \frac{21}{2}\right)^2 - \frac{7}{60}, x - \frac{21}{2}, 1 \right\}.$$

The MATLAB script `LSQquadOrtho.m` illustrates what happens when we use the normal equations based on an orthogonal basis to solve the previous fitting problem in single precision.

The **QR decomposition** (see below) produces an orthogonal matrix that helps us solve the least squares problem in a stable way.



**Polynomial** least squares fits can be computed in MATLAB directly from the data (without having to set up any intermediate linear systems).

The two commands required for this are

`p = polyfit(x, y, n)`: returns the coefficients  $p$  of the degree  $n$  polynomial

$$p(x) = p(1)x^n + p(2)x^{n-1} + \dots + p(n)x + p(n+1)$$

that provides the best least squares fit to the data in  $x$  and  $y$ .

`v = polyval(p, u)`: evaluates the polynomial  $p$  with coefficients  $p$  at all the values specified in  $u$ .

The MATLAB script `PolyfitDemo.m` illustrates the use of these two functions for the data from the previous example and for the same data contaminated with 10% noise.



The MATLAB function `censusgui` from [NCM] illustrates the use of normalized data for polynomial least squares fitting of U.S. population data.

It also uses `pchip` along with `spline` fitting, and an exponential fit of the type

$$y = ce^{\alpha x}.$$

Note that **it is not a good idea to use polynomials for extrapolation.**

### Remark

*If you are interested in **statistics**, then `[p, S] = polyfit(x, y, n)` will also provide information for **error bounds on the prediction** in `S`, and `[p, S, mu] = polyfit(x, y, n)` will produce the coefficients of a fitting polynomial in the normalized variable  $\hat{x} = (x - \mu)/\sigma$ , where  $\mu = \text{mu}(1)$  is the **mean** of the data sites `x` and  $\sigma = \text{mu}(2)$  is the **standard deviation**.*



## Theorem

Every real  $m \times n$  matrix  $A$  has a *QR decomposition*

$$A = QR,$$

where  $Q$  is an  $m \times m$  *orthogonal matrix* and  $R$  is an  $m \times n$  upper triangular matrix.

## Remark

There are many different ways in which the matrices  $Q$  and  $R$  can be found. E.g., one can use

- Gram-Schmidt orthogonalization (and modifications thereof),
- Householder reflections, or
- Givens rotations.

All of these are discussed in MATH 477. We will only investigate the *use* of the QR decomposition.

# Orthogonal Matrices

The  $m \times m$  matrix  $Q$  in the QR decomposition is said to be **orthogonal**. This means that

- its **columns are orthogonal** to each other, i.e.,  $\mathbf{q}_i^T \mathbf{q}_j = 0$  if  $i \neq j$  and  $\mathbf{q}_i$  is the  $i$ th column of  $Q$  (see the example involving `LSQquadOrtho.m` above),
- its **columns are normalized**, i.e.,  $\|\mathbf{q}_i\|_2 = 1$ ,  $i = 1, \dots, n$  (this is not satisfied in `LSQquadOrtho.m`).

This can be summarized as

$$Q^T Q = Q Q^T = I,$$

where  $I$  is the  $m \times m$  identity matrix.

Note that this implies that

$$Q^{-1} = Q^T \quad \text{— a very handy property.}$$



## Orthogonal Matrices (cont.)

### Example

Verify that the matrix

$$Q = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{3}} & \frac{-1}{\sqrt{6}} \\ 0 & \frac{1}{\sqrt{3}} & \frac{2}{\sqrt{6}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{3}} & \frac{1}{\sqrt{6}} \end{bmatrix}$$

is an orthogonal matrix.

We need to verify that

$$\begin{aligned} \mathbf{q}_1^T \mathbf{q}_2 &= 0, & \mathbf{q}_1^T \mathbf{q}_3 &= 0, & \mathbf{q}_2^T \mathbf{q}_3 &= 0, \\ \mathbf{q}_1^T \mathbf{q}_1 &= 1, & \mathbf{q}_2^T \mathbf{q}_2 &= 1, & \mathbf{q}_3^T \mathbf{q}_3 &= 1. \end{aligned}$$

Note that the conditions in the second row are equivalent to  $\|\mathbf{q}_i\|_2 = 1$ ,  $i = 1, 2, 3$ .

**Example** (cont.)

Columns are pairwise perpendicular:

$$\mathbf{q}_1^T \mathbf{q}_2 = \left[ \frac{1}{\sqrt{2}}, 0, \frac{1}{\sqrt{2}} \right] \begin{bmatrix} \frac{1}{\sqrt{3}} \\ \frac{2}{\sqrt{6}} \\ \frac{-1}{\sqrt{3}} \end{bmatrix} = \frac{1}{\sqrt{6}} - \frac{1}{\sqrt{6}} = 0,$$

$$\mathbf{q}_1^T \mathbf{q}_3 = \left[ \frac{1}{\sqrt{2}}, 0, \frac{1}{\sqrt{2}} \right] \begin{bmatrix} \frac{-1}{\sqrt{6}} \\ \frac{2}{\sqrt{6}} \\ \frac{1}{\sqrt{6}} \end{bmatrix} = -\frac{1}{\sqrt{12}} + \frac{1}{\sqrt{12}} = 0,$$

$$\mathbf{q}_2^T \mathbf{q}_3 = \left[ \frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, \frac{-1}{\sqrt{3}} \right] \begin{bmatrix} \frac{-1}{\sqrt{6}} \\ \frac{2}{\sqrt{6}} \\ \frac{1}{\sqrt{6}} \end{bmatrix} = -\frac{1}{\sqrt{18}} + \frac{2}{\sqrt{18}} - \frac{1}{\sqrt{18}} = 0.$$





**Example** (cont.)

Columns are normalized:

$$\mathbf{q}_1^T \mathbf{q}_1 = \left[ \frac{1}{\sqrt{2}}, 0, \frac{1}{\sqrt{2}} \right] \begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ \frac{1}{\sqrt{2}} \end{bmatrix} = \frac{1}{2} + \frac{1}{2} = 1,$$

$$\mathbf{q}_2^T \mathbf{q}_2 = \left[ \frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, \frac{-1}{\sqrt{3}} \right] \begin{bmatrix} \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{3}} \\ \frac{-1}{\sqrt{3}} \end{bmatrix} = \frac{1}{3} + \frac{1}{3} + \frac{1}{3} = 1,$$

$$\mathbf{q}_3^T \mathbf{q}_3 = \left[ \frac{-1}{\sqrt{6}}, \frac{2}{\sqrt{6}}, \frac{1}{\sqrt{6}} \right] \begin{bmatrix} \frac{-1}{\sqrt{6}} \\ \frac{2}{\sqrt{6}} \\ \frac{1}{\sqrt{6}} \end{bmatrix} = \frac{1}{6} + \frac{4}{6} + \frac{1}{6} = 1.$$



## Orthogonal Matrices (cont.)

Another important property of any  $m \times m$  orthogonal matrix  $Q$  is that it leaves the **2-norm invariant**, i.e., for any  $m$ -vector  $\mathbf{x}$  we have

$$\|Q\mathbf{x}\|_2 = \|\mathbf{x}\|_2.$$

Proof.

$$\begin{aligned} \|Q\mathbf{x}\|_2^2 &= (Q\mathbf{x})^T(Q\mathbf{x}) \\ &= \mathbf{x}^T Q^T Q \mathbf{x} \\ &\stackrel{Q^T Q = I}{=} \mathbf{x}^T \mathbf{x} = \|\mathbf{x}\|_2^2 \end{aligned}$$



### Remark

*In fact, orthogonal matrices/transformations represent either a **rotation** (if  $\det Q = 1$ ) or a **reflection** ( $\det Q = -1$ ), i.e., a length-preserving (or isometric) geometric transformation. Angles are also preserved (just compare  $(Q\mathbf{x})^T(Q\mathbf{y})$  with  $\mathbf{x}^T \mathbf{y}$ ).*

Consider first the square non-singular linear system  $A\mathbf{x} = \mathbf{b}$ , and assume that  $A$  has the QR decomposition  $A = QR$ .

Then

$$A\mathbf{x} = \mathbf{b} \iff QR\mathbf{x} = \mathbf{b}.$$

Multiplication by  $Q^{-1} = Q^T$  yields

$$R\mathbf{x} = Q^T\mathbf{b}.$$

This is an **upper triangular system** for the unknown vector  $\mathbf{x}$  which can be easily solved by back substitution (note a certain similarity with the procedure used with the LU decomposition earlier).

In fact, QR decomposition is in most cases the **preferred method for the solution of linear systems**. It provides a **good balance of numerical stability** (better than LU) **and efficiency** (worse than LU).



Recall that the least squares solution of the overdetermined linear system  $\mathbf{A}\mathbf{c} = \mathbf{y}$  is given by the coefficient vector  $\mathbf{c}$  that minimizes the norm of the 2-norm residual  $\|\mathbf{A}\mathbf{c} - \mathbf{y}\|_2$ .

Using the QR decomposition of  $\mathbf{A}$  this becomes

$$\|\mathbf{QR}\mathbf{c} - \mathbf{y}\|_2 \stackrel{\mathbf{Q}\mathbf{Q}^T = \mathbf{I}}{=} \|\mathbf{QR}\mathbf{c} - \mathbf{Q}\mathbf{Q}^T\mathbf{y}\|_2 = \|\mathbf{Q}(\mathbf{R}\mathbf{c} - \mathbf{Q}^T\mathbf{y})\|_2.$$

Now remember that the orthogonal matrix  $\mathbf{Q}$  leaves the 2-norm invariant ( $\|\mathbf{Q}\mathbf{x}\|_2 = \|\mathbf{x}\|_2$ ). Therefore we have

$$\|\mathbf{Q}(\mathbf{R}\mathbf{c} - \mathbf{Q}^T\mathbf{y})\|_2 = \|\mathbf{R}\mathbf{c} - \mathbf{Q}^T\mathbf{y}\|_2.$$

Finally, in the overdetermined system case (with  $m > n$ ) the last  $m - n$  rows of the upper triangular matrix  $\mathbf{R}$  are in fact all zero.

Therefore, the least squares solution of  $\mathbf{A}\mathbf{c} = \mathbf{y}$  is given by the solution of the square upper triangular system (provided  $\mathbf{A}$  has full (column) rank  $n$ )

$$\hat{\mathbf{R}}\mathbf{c} = \mathbf{z},$$

where  $\hat{\mathbf{R}} = \mathbf{R}(1:n, 1:n)$  and  $\mathbf{z} = \mathbf{Q}^T(1:n, :)\mathbf{y}$ .



## Example

Again we use the data

x	10	10.2	10.4	10.6	10.8	11
y	0	0.004	0.016	0.036	0.064	0.1

(without and with noise) and fit with a quadratic polynomial

$$P(x) = c_1 x^2 + c_2 x + c_3.$$

The MATLAB script `LSQquadQR.m` illustrates how we can use the method described above based on the QR decomposition of  $A$  to obtain the fit.



## Theorem

Let  $A$  be a complex  $m \times n$  matrix.  $A$  has a *singular value decomposition* of the form

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

where  $\Sigma$  is a uniquely determined  $m \times n$  (real) diagonal matrix,  $U$  is an  $m \times m$  unitary matrix, and  $V$  is an  $n \times n$  unitary matrix.

## Remark

- The entries  $\sigma_i$  of the diagonal matrix  $\Sigma$  are the *singular values* of  $A$  (recall our earlier discussion of the matrix 2-norm).
- Note that this theorem guarantees that *every matrix can be diagonalized!*
- The theorem is very general. If  $A$  is a *real* matrix, then “unitary” translates to “*orthogonal*” and “ $*$ ” (Hermitian conjugate) to “ $T$ ”.

## Using the SVD to solve $\|A\mathbf{c} - \mathbf{y}\|_2 \rightarrow \min$

Assume  $A$  is real  $m \times n$ ,  $m \geq n$ , and  $\text{rank}(A) = r = n$ .

Compute reduced SVD

$$A = \hat{U}\hat{\Sigma}V^T,$$

with  $\hat{U} = U(:, 1:r)$  an  $m \times r$  real matrix with orthogonal columns, real diagonal  $\hat{\Sigma} = \Sigma(1:r, 1:r)$ , and real orthogonal  $r \times r$  matrix  $V$ .

Use normal equations

$$\begin{aligned} A^T A \mathbf{c} &= A^T \mathbf{y} && \iff && (V \underbrace{\hat{\Sigma}^T \hat{U}^T}_{=\hat{\Sigma}}) (\underbrace{\hat{U} \hat{\Sigma} V^T}_{=I}) \mathbf{c} &= V \underbrace{\hat{\Sigma}^T \hat{U}^T}_{=\hat{\Sigma}} \mathbf{y} \\ &&& \iff && V \hat{\Sigma}^2 V^T \mathbf{c} &= V \hat{\Sigma} \hat{U}^T \mathbf{y} \\ (\hat{\Sigma}^{-1} V^T) \times &&& \iff && \hat{\Sigma} V^T \mathbf{c} &= \hat{U}^T \mathbf{y}. \end{aligned}$$

The least squares solution is given by

$$\mathbf{c} = \underbrace{V \hat{\Sigma}^{-1} \hat{U}^T}_{=A^\dagger, \text{ pseudoinverse}} \mathbf{y}.$$



According to the derivation presented on the previous slide we can follow the following **algorithm** to solve the least squares problem of minimizing  $\|\mathbf{A}\mathbf{c} - \mathbf{y}\|_2$ :

- 1 Solve the **diagonal system**

$$\hat{\Sigma}\mathbf{z} = \hat{\mathbf{U}}^T\mathbf{y}$$

for  $\mathbf{z}$ . Here  $\hat{\Sigma} = \Sigma(1:r, 1:r)$  and  $\hat{\mathbf{U}} = \mathbf{U}(:, 1:r)$  contains the first  $r$  columns of  $\mathbf{U}$  (**reduced SVD**).

- 2 Compute the coefficients  $\mathbf{c}$  via

$$\mathbf{V}^T\mathbf{c} = \mathbf{z} \iff \mathbf{c} = \mathbf{V}\mathbf{z}.$$

Alternatively, we can **directly use the pseudoinverse** to get the least squares solution.


Both approaches are illustrated in the MATLAB script `LSQquadSVD.m`. In MATLAB `[U S V] = svd(A)` produces the factors of the SVD, and the pseudoinverse is given by `pinv(A)`.

More details are provided in Chapters 5 and 10 of [NCM] as well as in MATH 477.





# Why is the SVD so fundamental?

- The fact that  $U$  and  $V$  are unitary (orthogonal) is
  - ▶ fundamental for geometric insights
- The fact that  $\Sigma$  is diagonal provides answers to important questions in linear algebra
  - number of non-zero singular values,  $r = \text{rank}(A)$
  - $\text{range}(A) = \text{range}(U(:, 1:r))$ ,  $\text{null}(A) = \text{range}(V(:, r+1:n))$
- The SVD is stable, i.e., small changes in  $A$  will cause only small changes in the SVD (in fact, this is the most stable matrix decomposition method).
- The SVD is optimal in the sense that it provides the
  - ▶ best low-rank approximations of  $A$
- Thanks to Gene Golub there are efficient and stable algorithms to compute the SVD.
- A new algorithm that very efficiently finds all singular values with high relative accuracy was found recently by [Drmač & Veselić]. 

# References I



C. Moler.

Numerical Computing with MATLAB.

SIAM, Philadelphia, 2004.

Also <http://www.mathworks.com/moler/>.



L. N. Trefethen and D. Bau, III.

Numerical Linear Algebra.

SIAM, Philadelphia, 1997.



Z. Drmač and K. Veselić.

New fast and accurate Jacobi SVD algorithm: Parts I and II.

SIAM J. Matrix Anal. Appl. **29** (2008), 1322–1362.



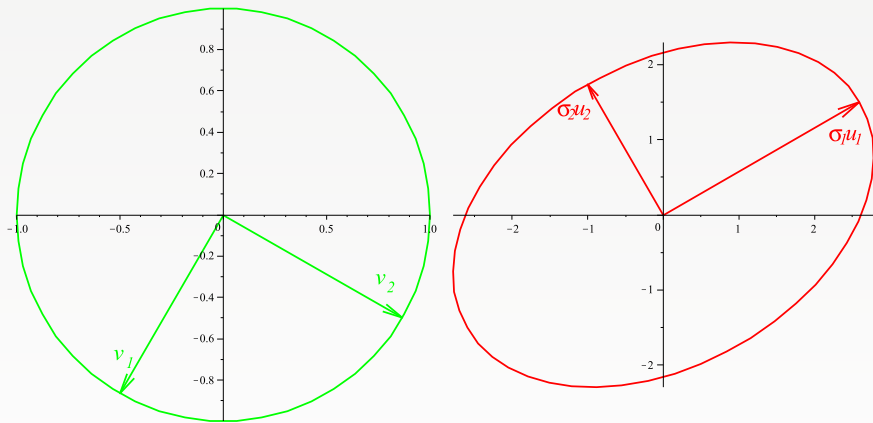


Figure: ON bases for row space and column space of  $A$ .



## Theorem

The  $m \times n$  matrix  $A$  can be decomposed into a sum of  $r$  rank-one matrices:

$$A = \sum_{j=1}^r \sigma_j \mathbf{u}_j \mathbf{v}_j^*.$$

Moreover, the best 2-norm approximation of rank  $\nu$  ( $0 \leq \nu \leq r$ ) to  $A$  is given by

$$A_\nu = \sum_{j=1}^{\nu} \sigma_j \mathbf{u}_j \mathbf{v}_j^*.$$

In fact,

$$\|A - A_\nu\|_2 = \sigma_{\nu+1}.$$

This theorem is very useful for applications such as image compression and is illustrated in the MATLAB script `SVD_movie.m` (see also `imagesvd.m` from [NCM]).

[Return](#)

