TANA15/Lecture 2 - Contents

Linear Systems of Equations

- The *LU* Decomposition. Cholesky Decomposition.
- Sensitivity, Condition number, Residual.

Least Squares Problems

- Normal Equations. Data fitting.
- Gram-Schmidt ortogonalization.
- The *QR* decomposition. Projections.

Linear Systems of Equations

A *linear system of equations* can be written as

$$Ax = b,$$

where A is a matrix, x is the solution, and b is the right hand side.

Lemma A *linear system of equations* $Ax = b$ has a solution if $b \in \text{Range}(A)$.

Remark If A has a non-trivial null-space then if x_1 is a solution and $x_2 \in \text{null}(A)$ we have $A(x_1 + x_2) = Ax_1 + 0 = b$ so $x_1 + x_2$ is a also solution.

Lemma Let $A \in \mathbb{R}^{n\times n}$. If $\text{Null}(A) = \{0\}$ then A^{-1} exists and A is called *non-singular*.

Remark Suppose $A \in \mathbb{R}^{m\times n}$, $m > n$, then A^{-1} does not exist. If $b \in \text{Range}(A)$ a solution to $Ax = b$ exists. If $\text{null}(A) = \{0\}$ then the solution is unique.

Lemma Let $A \in \mathbb{R}^{n\times n}$. Then then following are equivalent: $\det(A) \neq 0$, A^{-1} exists, and $\text{Rank}(A) = n$.

Remark Not very useful for checking if $Ax = b$ has a solution.

Solving Linear Systems of Equations

Solve $Ax = b$ where

$$\begin{pmatrix} 1 & 2 & 2 \\ 4 & 4 & 2 \\ 4 & 6 & 4 \end{pmatrix} x = \begin{pmatrix} 3 \\ 6 \\ 10 \end{pmatrix}$$

Method Reduce A to upper triangular form using row operations and partial pivoting.

Following the pivoting strategy we exchange rows one and two:

$$\left(\begin{array}{ccc|c} 1 & 2 & 2 & 3 \\ 4 & 4 & 2 & 6 \\ 4 & 6 & 4 & 10 \end{array}\right) \sim \left(\begin{array}{ccc|c} 4 & 4 & 2 & 6 \\ 1 & 2 & 2 & 3 \\ 4 & 6 & 4 & 10 \end{array}\right)$$

Use multipliers $m_{21} = 0.25$ and $m_{31} = 1$ to eliminate a_{21} and a_{31}. Pivot again by exchanging rows 2 and 3.

$$\left(\begin{array}{ccc|c} 4 & 4 & 2 & 6 \\ 0 & 1 & 1.5 & 1.5 \\ 0 & 2 & 2 & 4 \end{array}\right) \sim \left(\begin{array}{ccc|c} 4 & 4 & 2 & 6 \\ 0 & 2 & 2 & 4 \\ 0 & 1 & 1.5 & 1.5 \end{array}\right)$$

Now use a multiplier $m_{32} = 0.5$ to eliminate a_{32}. Then solve the triangular system using backwards substitution.

$$\left(\begin{array}{ccc|c} 4 & 4 & 2 & 6 \\ 0 & 2 & 2 & 4 \\ 0 & 0 & 0.5 & -0.5 \end{array}\right) \Longrightarrow x = \begin{pmatrix} -1 \\ 3 \\ -1 \end{pmatrix}$$

This is called *Gaussian Elimination*!

Definition Row operations use a *Gauss transformation matrix M* and row exchanges use *Permutation matrix P*.

Example A Gauss-transformation has the structure

$$M \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 - m_{21}x_1 \\ x_3 - m_{31}x_1 \end{pmatrix} \Longrightarrow M = \begin{pmatrix} 1 & 0 & 0 \\ -m_{21} & 1 & 0 \\ -m_{31} & 0 & 1 \end{pmatrix}.$$

and an example of a permutation matrix is

$$P_{23} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} x_1 \\ x_3 \\ x_2 \end{pmatrix} \Longrightarrow P_{23} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}.$$

Both P^{-1} and M^{-1} exists. What is M^{-1}?

Repeat the steps taken to reduce A to upper triangular form using Gauss transformations and permutation matrices.

First exchange rows 1 and 2

$$P_{12}A = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 2 & 2 \\ 4 & 4 & 2 \\ 4 & 6 & 4 \end{pmatrix} = \begin{pmatrix} 4 & 4 & 2 \\ 1 & 2 & 2 \\ 4 & 6 & 4 \end{pmatrix}$$

Second use a Gauss transformation M_1 to eliminate a_{21} and a_{31}.

$$M_1(P_{12}A) = \begin{pmatrix} 1 & 0 & 0 \\ -0.25 & 1 & 0 \\ -1 & 0 & 1 \end{pmatrix} \begin{pmatrix} 4 & 4 & 2 \\ 1 & 2 & 2 \\ 4 & 6 & 4 \end{pmatrix} = \begin{pmatrix} 4 & 4 & 2 \\ 0 & 1 & 1.5 \\ 0 & 2 & 2 \end{pmatrix}$$

Continue and exchange rows 2 and 3

$$P_{23}(M_1P_{12}A) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 4 & 4 & 2 \\ 0 & 1 & 1.5 \\ 0 & 2 & 2 \end{pmatrix} = \begin{pmatrix} 4 & 4 & 2 \\ 0 & 2 & 2 \\ 0 & 1 & 1.5 \end{pmatrix}$$

Lastly use a Gauss transformation M_2 to eliminate a_{32}

$$M_2(P_{23}M_1P_{12}A) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -0.5 & 1 \end{pmatrix} \begin{pmatrix} 4 & 4 & 2 \\ 0 & 2 & 2 \\ 0 & 1 & 1.5 \end{pmatrix} = \begin{pmatrix} 4 & 4 & 2 \\ 0 & 2 & 2 \\ 0 & 0 & 0.5 \end{pmatrix} = U$$

We now have $M_2P_{23}M_1P_{12}A = U$ or $P_{12}A = M_1^{-1}P_{23}^TM_2^{-1}U$.

Multiply both sides by P_{23} to obtain

$$P_{23}P_{12}A = P_{23}M_1^{-1}P_{23}^TM_2^{-1}U,$$

or $PA = LU$ where,

$$P = P_{23}P_{12} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}, \quad U = \begin{pmatrix} 4 & 4 & 2 \\ 0 & 2 & 2 \\ 0 & 0 & 0.5 \end{pmatrix},$$

$$L = P_{23}M_1^{-1}P_{23}^TM_2^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0.25 & -0.5 & 1 \end{pmatrix}.$$

This is called the LU decomposition!

The LU Decomposition

Theorem Every non-singular matrix A can be written $PA = LU$, where P is a permutation matrix, L and U are triangular, non–singular, and $|L| \leq 1$.

Remarks Requires $2n^3/3$ multiply/additions to compute. Most efficient way to check if a matrix A is non-singular.

Example Use the LU decomposition for solving a linear system $Ax = b$. In Matlab

```
>> [L,U,P]=lu(A);
>> y=L\(P*b); x = U\y;
```


The Cholesky decomposition

Definition A matrix $A \in \mathbb{R}^{n\times n}$ is *positive definite* if $x^TAx > 0$, for every $x \neq 0$.

Proposition If A is symmetric and positive definite then pivoting is not needed and $A = R^TR$ is the *Cholesky decomposition*.

Remark Exactly half the work and memory compared to regular LU-decomposition. In Matlab we use `chol`.

Sensitivity of Linear systems

Lemma Suppose $Ax = b$ and we are given inexact data $b_\delta = b + \delta b$. The resulting error is

$$\frac{||\delta x||}{||x||} \leq ||A||||A^{-1}||.\frac{||\delta b||}{||b||}.$$

Definition The *condition number* is $\kappa(A) = ||A||||A^{-1}||$.

Remark The condition number is a measure of how sensitive a linear system is with respect to errors in the right hand side.

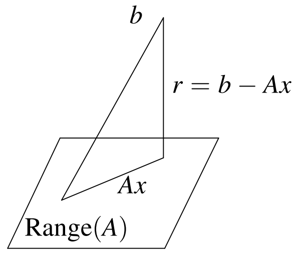
The Residual

Definition The *residual* of an approximate solution $\hat{x}$ to a linear system $Ax = b$ is $r = b - A\hat{x}$.

Lemma The error in an approximate solution $\hat{x}$ can be estimated as $\|x - \hat{x}\| \leq \|A^{-1}\|\|r\|$.

Remark If a system is *well-conditioned* and the *residual* is small then the solution is accurate.

The Least Squares Problem



Definition Let $A \in \mathbb{R}^{m\times n}$, $m > n$. The *least squares problem* is to find the $x \in \mathbb{R}^n$ that minimize

$$\|Ax - b\|_2.$$

Remark The least squares problem always has a solution.

Proposition The following are equivalent

(1) $x = \text{argmin}\|Ax - b\|_2$.

(2) $r = b - Ax \perp \text{Range}(A)$.

(3) $A^TAx = A^Tb$.

Remark The *normal equations* can be solved using the Cholesky decomposition, but

$$\kappa_2(A^TA) = (\kappa_2(A))^2,$$

so that should be avoided.

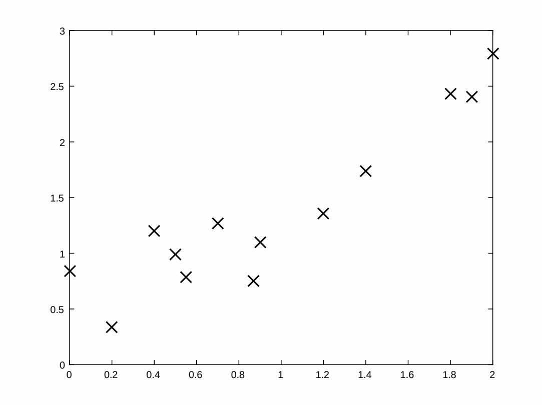
Definition If $\text{rank}(A) = n$ then $A^+ = (A^TA)^{-1}A^T$ is called the *pseudo inverse*.

Lemma If $\text{rank}(A) = n$ then the solution to the least squares problem is given by $x = A^+b$.

Remark If $\text{Rank}(A) < n$ the least squares problem does not have a unique solution.

Data fitting

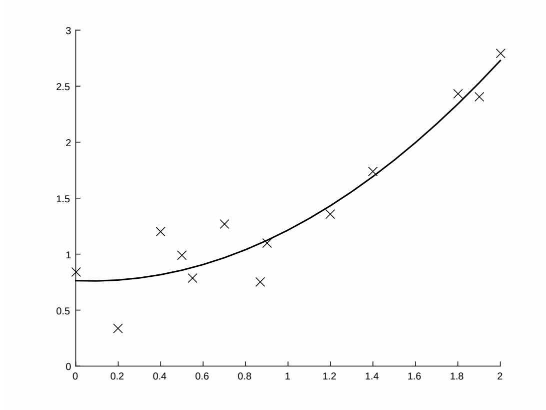
Example Fit a polynomial $p(t) = c_0 + c_1 t + c_2 t^2$ to a data $\{(t_i, y_i)\}_{i=1}^m$.



How can we formulate this as a least squares problem?

Matlab Suppose the data is stored in two vectors `t` and `y`.

```
>> A=[ t.^0 t.^1 t.^2]; b=y; c=(A'*A)\(A'*b);
>> tt=0:0.1:2;yy=c(1)+c(2)*tt+c(3)*tt.^2;
>> plot(t,y,'x',tt,yy);
```



Geometrical Solution

Let P_A be the *orthogonal projection* onto Range(A). Then

$$Ax = P_A b.$$

The residual is

$$r = (I - P_A)b.$$

How to compute the projection onto Range(A)?

Lemma Suppose $Q = (q_1, \ldots, q_n)$ is an orthogonal basis for Range(A). Then

$$P_A = QQ^T.$$

Gram-Schmidt Orthogonalization

Algorithm Compute an *orthogonal basis* for Range(A), $A = (a_1, \ldots, a_n)$, by

$r_{11} = \|a_1\|_2, \ q_1 = a_1/r_{11}.$

`for` $j = 2, \ldots, n$

  $\tilde{q}_j = a_j.$

  `for` $i = 2, \ldots, j-1$

    $r_{ij} = q_i^T \tilde{q}_j.$

    $\tilde{q}_j = \tilde{q}_j - r_{ij} q_i.$

  `end`

  $r_{jj} = \|\tilde{q}_j\|_2, \ q_j = \tilde{q}_j / r_{jj}.$

`end`

Remark The solution to the least squares problem is obtained by solving $Ax = P_A b = QQ^T b \in$ Range(A).

The QR Decomposition

Lemma Let $A \in \mathbb{R}^{m\times n}$, $m > n$. Then A can be factorized as

$$A = Q\begin{pmatrix} R \\ 0 \end{pmatrix},$$

where $Q \in \mathbb{R}^{m\times m}$ is *orthogonal* and $R \in \mathbb{R}^{n\times n}$ is upper triangular. If $\mathrm{rank}(A) = n$ then R is non-singular.

Definition Let $Q = (Q_1, Q_2)$ where $Q_1 \in \mathbb{R}^{m\times n}$. Then $A = Q_1 R$ is called the *reduced QR decomposition*.

Remark The columns of Q_1 form an orthonormal basis for $\mathrm{Range}(A)$.

Computing Projections

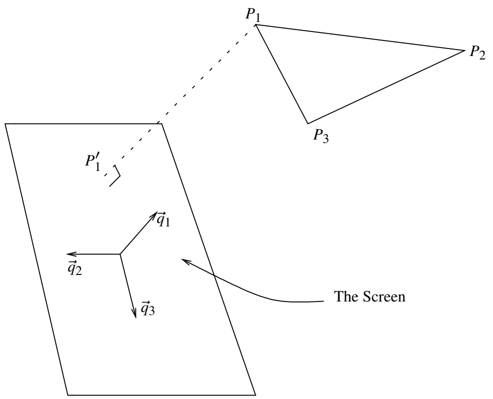
Lemma Suppose the columns of $Q_1 = (q_1, \dots, q_k)$ are orthogonal. Then the *orthogonal projection* on the subspace $\mathrm{span}(q_1, \dots, q_k)$ is

$$P = Q_1 Q_1^T.$$

Application In computer graphics an object is represented by a set of polygons. Each corner of the polygons have known coordinates in $\mathbb{R}^3$. In order to draw the object on screen we need to compute projections of the coordinate vectors onto the screen.

If we draw the polygons in the order *closest last* then we obtain a correct image. Thus we also need the distance z from the plane to the polygons. This is called z–buffer technique.

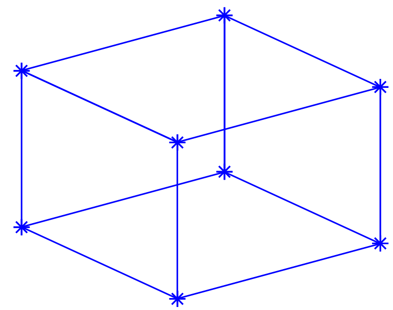
How to organize the computations?



The polygon, with corners, $\{P_1, P_2, P_3\}$, should be projected onto the screen $\mathrm{span}(\vec{q}_2, \vec{q}_3)$ in the direction given by the *normal vector* $\vec{q}_1$.

We obtain

$$P_k' = (q_2^T P_k)q_2 + (q_3^T P_k)q_3, \quad \text{and} \quad z_k = (\vec{q}_1^T P_k).$$



Example We create a matrix P containing the corners of a cube.

```
>> P =
     0     1     0     1     0     1     0     1
     0     0     1     1     0     0     1     1
     0     0     0     0     1     1     1     1
>> ind=[1 2 4 3 1 5 6 2 6 8 4 8 7 5 7 3];
>> plot3(P(1,ind),P(2,ind),P(3,ind),'b-*');
```

Can we recreate the same figure by projection and using a $2D$ plot?

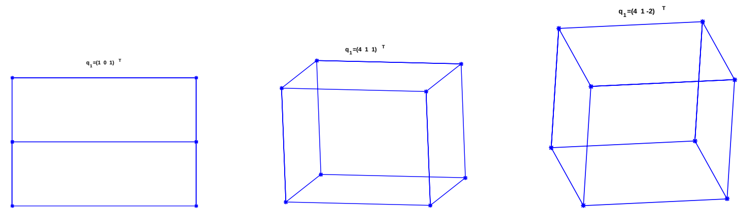
Let S_k be the projection of P_k on the screen and q_1 be the normal to the plane. In Matlab

```
>> q1=[1 1 0]';[Q,R]=qr(q1);
>> for k=1:8, S(:,k)=Q(:,2:3)'*P(:,k);,end
>> ind=[1 2 4 3 1 5 6 2 6 8 4 8 7 5 7 3];
>> plot(S(1,ind),S(2,ind),'b-*');
```

The distance from the screen to the points are

```
>> for k=1:8, z(k)=Q(:,1)'*P(:,k);,end
```

Not needed here since we draw hidden lines.



We view the cube from different directions q_1. What we see on the screen is the projection in the direction q_1.

If we draw surfaces we need to sort with respect to the distance from the screen. This called z-buffer technique.

The QR Decomposition and Least squares

Lemma If Q is *orthogonal* then, for any $x \in \mathbb{R}^n$, $\|Qx\|_2 = \|x\|_2$.

Proof This follows from

$$\|Qy\|_2^2 = (Qy)^T(Qy) = y^T Q^T Q y = y^T y = \|y\|_2^2.$$

Lemma Suppose $A = Q_1 R$ is the reduced QR decomposition. The least squares solution is

$$x = R^{-1}(Q_1^T b).$$

Matlab Compute the reduced QR decomposition and find the solution by

```
>> [Q,R]=qr(A,0);
>> x=R\(Q'*b);
```

Remark Typically $m >> n$. Dimensions $m = 10^3 - 10^5$ and $n = 5 - 50$ are not unusual.

Question How to compute the QR decomposition efficiently?