Householder Triangularization and Least Squares Problem

Emre Mengi

Department of Mathemtics
Koç University
Istanbul, Turkey

October 26, 2011

Outline

Step k of the algorithm: $(k = 1, \ldots, n-1)$

$$\underbrace{\begin{bmatrix} R & B \\ 0 & A^{(k)} \end{bmatrix}}_{Q_{k-1}\ldots Q_1 A} \longrightarrow \underbrace{\begin{bmatrix} R & B \\ 0 & \hat{Q}_k A^{(k)} \end{bmatrix}}_{Q_k Q_{k-1}\ldots Q_1 A}$$

- $Q_k = \begin{bmatrix} I_{k-1} & 0 \\ 0 & \hat{Q}_k \end{bmatrix} \in \mathbb{C}^{m \times m}$, $R \in \mathbb{C}^{(k-1)\times(k-1)}$ is upper triangular
- $\hat{Q}_k \in \mathbb{C}^{(m-k+1)\times(m-k+1)}$ is the HH reflector assoc with $a_1^{(k)}$ so that

$$A^{(k)} = \begin{bmatrix} x & x & x & \ldots & x \\ x & x & x & \ldots & x \\ \vdots & \vdots & & & \vdots \\ x & x & x & \ldots & x \end{bmatrix} \longrightarrow \hat{Q}_k A^{(k)} = \begin{bmatrix} x & x & x & \ldots & x \\ 0 & x & x & \ldots & x \\ \vdots & \vdots & & & \vdots \\ 0 & x & x & \ldots & x \end{bmatrix}$$

Step k of the algorithm: $(k = 1, \dots, n-1)$

$$\underbrace{\begin{bmatrix} R & B \\ 0 & A^{(k)} \end{bmatrix}}_{Q_{k-1} \dots Q_1 A} \longrightarrow \underbrace{\begin{bmatrix} R & B \\ 0 & \hat{Q}_k A^{(k)} \end{bmatrix}}_{Q_k Q_{k-1} \dots Q_1 A}$$

- $Q_k = \begin{bmatrix} I_{k-1} & 0 \\ 0 & \hat{Q}_k \end{bmatrix} \in \mathbb{C}^{m \times m}$, $R \in \mathbb{C}^{(k-1)\times(k-1)}$ is upper triangular
- $\hat{Q}_k \in \mathbb{C}^{(m-k+1)\times(m-k+1)}$ is the HH reflector assoc with $a_1^{(k)}$ so that

$$A^{(k)} = \begin{bmatrix} x & x & x & \dots & x \\ x & x & x & \dots & x \\ \vdots & \vdots & & & \vdots \\ x & x & x & \dots & x \end{bmatrix} \longrightarrow \hat{Q}_k A^{(k)} = \begin{bmatrix} x & x & x & \dots & x \\ 0 & x & x & \dots & x \\ \vdots & \vdots & & & \vdots \\ 0 & x & x & \dots & x \end{bmatrix}$$

Step k of the algorithm: $(k = 1, \ldots, n-1)$

$$\underbrace{\begin{bmatrix} R & B \\ 0 & A^{(k)} \end{bmatrix}}_{Q_{k-1}\ldots Q_1 A} \longrightarrow \underbrace{\begin{bmatrix} R & B \\ 0 & \hat{Q}_k A^{(k)} \end{bmatrix}}_{Q_k Q_{k-1}\ldots Q_1 A}$$

- $Q_k = \begin{bmatrix} I_{k-1} & 0 \\ 0 & \hat{Q}_k \end{bmatrix} \in \mathbb{C}^{m\times m}$, $R \in \mathbb{C}^{(k-1)\times(k-1)}$ is upper triangular
- $\hat{Q}_k \in \mathbb{C}^{(m-k+1)\times(m-k+1)}$ is the HH reflector assoc with $a_1^{(k)}$ so that

$$A^{(k)} = \begin{bmatrix} x & x & x & \ldots & x \\ x & x & x & \ldots & x \\ \vdots & \vdots & & & \vdots \\ x & x & x & \ldots & x \end{bmatrix} \longrightarrow \hat{Q}_k A^{(k)} = \begin{bmatrix} x & x & x & \ldots & x \\ 0 & x & x & \ldots & x \\ \vdots & \vdots & & & \vdots \\ 0 & x & x & \ldots & x \end{bmatrix}$$

Algorithm

- **Input:** $A \in \mathbb{C}^{m \times n}$ with $m \geq n$
- **Output:** Upper triangular $R \in \mathbb{C}^{m \times n}$ and the HH vectors $u_1, \dots, u_{n-1} \in \mathbb{C}^m$. The unitary factor $Q \in \mathbb{C}^{m \times m}$ can be formed from the HH vectors so that $A = QR$ is a full QR factorization.

for $k = 1, n$ **do**
  $v \leftarrow A_{k:m,k}$
  $u_k \leftarrow v - \|v\| e_1$
  $u_k \leftarrow u_k / \|u_k\|$
  $A_{k:m,k:n} \leftarrow A_{k:m,k:n} - 2u_k(u_k^* A_{k:m,k:n})$
end for
$R \leftarrow A$
Return R

ALGORITHM

- **Input:** $A \in \mathbb{C}^{m \times n}$ with $m \geq n$
- **Output:** Upper triangular $R \in \mathbb{C}^{m \times n}$ and the HH vectors $u_1, \dots, u_{n-1} \in \mathbb{C}^m$. The unitary factor $Q \in \mathbb{C}^{m \times m}$ can be formed from the HH vectors so that $A = QR$ is a full QR factorization.

for $k = 1, n$ **do**

$v \leftarrow A_{k:m,k}$

$u_k \leftarrow v - \|v\| e_1$

$u_k \leftarrow u_k / \|u_k\|$

$A_{k:m,k:n} \leftarrow A_{k:m,k:n} - 2u_k(u_k^* A_{k:m,k:n})$

end for

$R \leftarrow A$

Return R

Algorithm

- **Input:** $A \in \mathbb{C}^{m \times n}$ with $m \geq n$
- **Output:** Upper triangular $R \in \mathbb{C}^{m \times n}$ and the HH vectors $u_1, \dots, u_{n-1} \in \mathbb{C}^m$. The unitary factor $Q \in \mathbb{C}^{m \times m}$ can be formed from the HH vectors so that $A = QR$ is a full QR factorization.

for $k = 1, n$ **do**

$v \leftarrow A_{k:m,k}$

$u_k \leftarrow v - \|v\| e_1$

$u_k \leftarrow u_k / \|u_k\|$

$A_{k:m,k:n} \leftarrow A_{k:m,k:n} - 2u_k(u_k^* A_{k:m,k:n})$

end for

$R \leftarrow A$

Return R

ALGORITHM

- **Input:** $A \in \mathbb{C}^{m \times n}$ with $m \geq n$
- **Output:** Upper triangular $R \in \mathbb{C}^{m \times n}$ and the HH vectors $u_1, \dots, u_{n-1} \in \mathbb{C}^m$. The unitary factor $Q \in \mathbb{C}^{m \times m}$ can be formed from the HH vectors so that $A = QR$ is a full QR factorization.

for $k = 1, n$ **do**
  $v \leftarrow A_{k:m,k}$
  $u_k \leftarrow v - \|v\| e_1$
  $u_k \leftarrow u_k / \|u_k\|$
  $A_{k:m,k:n} \leftarrow A_{k:m,k:n} - 2u_k(u_k^* A_{k:m,k:n})$
end for
$R \leftarrow A$
Return R

ALGORITHM

- **Input:** $A \in \mathbb{C}^{m \times n}$ with $m \geq n$
- **Output:** Upper triangular $R \in \mathbb{C}^{m \times n}$ and the HH vectors $u_1, \dots, u_{n-1} \in \mathbb{C}^m$. The unitary factor $Q \in \mathbb{C}^{m \times m}$ can be formed from the HH vectors so that $A = QR$ is a full QR factorization.

for $k = 1, n$ **do**
  $v \leftarrow A_{k:m,k}$
  $u_k \leftarrow v - \|v\| e_1$
  $u_k \leftarrow u_k / \|u_k\|$
  $A_{k:m,k:n} \leftarrow A_{k:m,k:n} - 2u_k(u_k^* A_{k:m,k:n})$
end for
$R \leftarrow A$
Return R

ALGORITHM

- **Input:** $A \in \mathbb{C}^{m \times n}$ with $m \geq n$
- **Output:** Upper triangular $R \in \mathbb{C}^{m \times n}$ and the HH vectors $u_1, \dots, u_{n-1} \in \mathbb{C}^m$. The unitary factor $Q \in \mathbb{C}^{m \times m}$ can be formed from the HH vectors so that $A = QR$ is a full QR factorization.

for $k = 1, n$ **do**

$v \leftarrow A_{k:m,k}$

$u_k \leftarrow v - \|v\| e_1$

$u_k \leftarrow u_k / \|u_k\|$

$A_{k:m,k:n} \leftarrow A_{k:m,k:n} - 2u_k(u_k^* A_{k:m,k:n})$

end for

$R \leftarrow A$

Return R

The unitary factor Q such that $A = QR$ can be recovered from the HH vectors u_k.

$$Q_n \cdots Q_1 A = R \text{ where } Q_k = \begin{bmatrix} I_{k-1} & 0 \\ 0 & I_{m-k+1} - 2u_k u_k^* \end{bmatrix}$$

equivalently

$$A = Q_1^* Q_2^* \cdots Q_n^* R = \underbrace{Q_1 Q_2 \cdots Q_n}_{Q} R.$$

The unitary factor Q such that $A = QR$ can be recovered from the HH vectors u_k.

$$Q_n \cdots Q_1 A = R \text{ where } Q_k = \begin{bmatrix} I_{k-1} & 0 \\ 0 & I_{m-k+1} - 2u_k u_k^* \end{bmatrix}$$

equivalently

$$A = Q_1^* Q_2^* \cdots Q_n^* R = \underbrace{Q_1 Q_2 \cdots Q_n}_{Q} R.$$

The unitary factor Q such that $A = QR$ can be recovered from the HH vectors u_k.

$$Q_n \cdots Q_1 A = R \text{ where } Q_k = \begin{bmatrix} I_{k-1} & 0 \\ 0 & I_{m-k+1} - 2u_k u_k^* \end{bmatrix}$$

equivalently

$$A = Q_1^* Q_2^* \cdots Q_n^* R = \underbrace{Q_1 Q_2 \cdots Q_n}_{Q} R.$$

- A very common use of the QR factorization is the numerical solution of the least squares problem.

- For the least squares problem Q does not need to be formed explicitly.

- Let $b \in \mathbb{C}^m$. We will need the product $Q^* b$, which can be computed by means of the vectors u_k, since

$$Q_k^* b = \underbrace{\begin{bmatrix} I_{k-1} & 0 \\ 0 & I_{m-k+1} - 2u_k u_k^* \end{bmatrix}}_{Q_k^* = Q_k} \underbrace{\begin{bmatrix} \hat{b} \in \mathbb{C}^{k-1} \\ \tilde{b} \in \mathbb{C}^{m-k+1} \end{bmatrix}}_{b} = \begin{bmatrix} \hat{b} \\ \tilde{b} - 2u_k(u_k^* \tilde{b}) \end{bmatrix}.$$

- A very common use of the QR factorization is the numerical solution of the least squares problem.
- For the least squares problem Q does not need to be formed explicitly.
- Let $b \in \mathbb{C}^m$. We will need the product Q^*b, which can be computed by means of the vectors u_k, since

$$Q_k^* b = \underbrace{\begin{bmatrix} I_{k-1} & 0 \\ 0 & I_{m-k+1} - 2u_k u_k^* \end{bmatrix}}_{Q_k^* = Q_k} \underbrace{\begin{bmatrix} \hat{b} \in \mathbb{C}^{k-1} \\ \tilde{b} \in \mathbb{C}^{m-k+1} \end{bmatrix}}_{b} = \begin{bmatrix} \hat{b} \\ \tilde{b} - 2u_k(u_k^* \tilde{b}) \end{bmatrix}.$$

- A very common use of the QR factorization is the numerical solution of the least squares problem.

- For the least squares problem Q does not need to be formed explicitly.

- Let $b \in \mathbb{C}^m$. We will need the product $Q^* b$, which can be computed by means of the vectors u_k, since

$$Q_k^* b = \underbrace{\begin{bmatrix} I_{k-1} & 0 \\ 0 & I_{m-k+1} - 2u_k u_k^* \end{bmatrix}}_{Q_k^* = Q_k} \underbrace{\begin{bmatrix} \hat{b} \in \mathbb{C}^{k-1} \\ \tilde{b} \in \mathbb{C}^{m-k+1} \end{bmatrix}}_{b} = \begin{bmatrix} \hat{b} \\ \tilde{b} - 2u_k(u_k^* \tilde{b}) \end{bmatrix}.$$

Remarks

The algorithm based on HH reflectors shows the existence of a QR factorization.

Theorem

Every matrix $A \in \mathbb{C}^{m \times n}$ with $m \geq n$ has a QR factorization.

Pay attention to the order of operation to perform $2u_k u_k^* A_{k:m,k:n}$.

- Inefficient way: $2(u_k u_k^*) A_{k:m,k:n}$
  $\#FLOPS = 2(m-k+1)^2 \times (n-k+1) + O(mn) + O(n^2)$
- Efficient way: $2u_k (u_k^* A_{k:m,k:n})$
  $\#FLOPS = 3(m-k+1) \times (n-k+1) + O(n)$

Remarks

The algorithm based on HH reflectors shows the existence of a QR factorization.

Theorem

Every matrix $A \in \mathbb{C}^{m \times n}$ with $m \geq n$ has a QR factorization.

Pay attention to the order of operation to perform $2u_k u_k^* A_{k:m,k:n}$.

- Inefficient way: $2(u_k u_k^*)A_{k:m,k:n}$
  $\#FLOPS = 2(m-k+1)^2 \times (n-k+1) + O(mn) + O(n^2)$
- Efficient way: $2u_k(u_k^* A_{k:m,k:n})$
  $\#FLOPS = 3(m-k+1) \times (n-k+1) + O(n)$

Remarks

The algorithm based on HH reflectors shows the existence of a QR factorization.

Theorem

Every matrix $A \in \mathbb{C}^{m \times n}$ with $m \geq n$ has a QR factorization.

Pay attention to the order of operation to perform $2u_k u_k^* A_{k:m,k:n}$.

- Inefficient way: $2(u_k u_k^*) A_{k:m,k:n}$
  $\#FLOPS = 2(m-k+1)^2 \times (n-k+1) + O(mn) + O(n^2)$
- Efficient way: $2u_k(u_k^* A_{k:m,k:n})$
  $\#FLOPS = 3(m-k+1) \times (n-k+1) + O(n)$

Remarks

The algorithm based on HH reflectors shows the existence of a QR factorization.

Theorem

Every matrix $A \in \mathbb{C}^{m \times n}$ with $m \geq n$ has a QR factorization.

Pay attention to the order of operation to perform $2u_k u_k^* A_{k:m,k:n}$.

- Inefficient way: $2(u_k u_k^*)A_{k:m,k:n}$
  $\#FLOPS = 2(m-k+1)^2 \times (n-k+1) + O(mn) + O(n^2)$
- Efficient way: $2u_k(u_k^* A_{k:m,k:n})$
  $\#FLOPS = 3(m-k+1) \times (n-k+1) + O(n)$

Remarks

The algorithm based on HH reflectors shows the existence of a QR factorization.

Theorem

Every matrix $A \in \mathbb{C}^{m \times n}$ with $m \geq n$ has a QR factorization.

Pay attention to the order of operation to perform $2 u_k u_k^* A_{k:m,k:n}$.

- Inefficient way: $2(u_k u_k^*) A_{k:m,k:n}$
  $\#FLOPS = 2(m-k+1)^2 \times (n-k+1) + O(mn) + O(n^2)$
- Efficient way: $2u_k(u_k^* A_{k:m,k:n})$
  $\#FLOPS = 3(m-k+1) \times (n-k+1) + O(n)$

Remarks

The algorithm based on HH reflectors shows the existence of a QR factorization.

Theorem

Every matrix $A \in \mathbb{C}^{m \times n}$ with $m \geq n$ has a QR factorization.

Pay attention to the order of operation to perform $2u_k u_k^* A_{k:m,k:n}$.

- Inefficient way: $2(u_k u_k^*) A_{k:m,k:n}$
  $\#FLOPS = 2(m-k+1)^2 \times (n-k+1) + O(mn) + O(n^2)$
- Efficient way: $2u_k(u_k^* A_{k:m,k:n})$
  $\#FLOPS = 3(m-k+1) \times (n-k+1) + O(n)$

Outline

for $k = 1, n$ **do**

$v \leftarrow A_{k:m,k}$

$\underbrace{u_k \leftarrow v - \|v\| e_1}_{O(m) \text{ flops}}$

$\underbrace{u_k \leftarrow u_k / \|u_k\|}_{O(m) \text{ flops}}$

$\underbrace{A_{k:m,k:n} \leftarrow A_{k:m,k:n} - 2u_k(u_k^* A_{k:m,k:n})}_{4(m-k+1)\times(n-k+1)+O(n) \text{ flops}}$

end for

$R \leftarrow A$

Return R

for $k = 1, n$ **do**

$v \leftarrow A_{k:m,k}$

$\underbrace{u_k \leftarrow v - \|v\| e_1}_{O(m) \;\; flops}$

$\underbrace{u_k \leftarrow u_k / \|u_k\|}_{O(m) \;\; flops}$

$\underbrace{A_{k:m,k:n} \leftarrow A_{k:m,k:n} - 2u_k(u_k^* A_{k:m,k:n})}_{4(m-k+1)\times(n-k+1)+O(n) \;\; flops}$

end for

$R \leftarrow A$

Return R

for $k = 1, n$ **do**

$v \leftarrow A_{k:m,k}$

$\underbrace{u_k \leftarrow v - \|v\| e_1}_{O(m) \ \ flops}$

$\underbrace{u_k \leftarrow u_k / \|u_k\|}_{O(m) \ \ flops}$

$\underbrace{A_{k:m,k:n} \leftarrow A_{k:m,k:n} - 2u_k(u_k^* A_{k:m,k:n})}_{4(m-k+1)\times(n-k+1)+O(n) \ \ flops}$

end for

$R \leftarrow A$

Return R

for $k = 1, n$ **do**

$$v \leftarrow A_{k:m,k}$$

$$\underbrace{u_k \leftarrow v - \|v\| e_1}_{O(m)\ \ \text{flops}}$$

$$\underbrace{u_k \leftarrow u_k / \|u_k\|}_{O(m)\ \ \text{flops}}$$

$$\underbrace{A_{k:m,k:n} \leftarrow A_{k:m,k:n} - 2u_k(u_k^* A_{k:m,k:n})}_{4(m-k+1)\times(n-k+1)+O(n)\ \ \text{flops}}$$

end for

$R \leftarrow A$

Return R

for $k = 1, n$ **do**

$v \leftarrow A_{k:m,k}$

$$\underbrace{u_k \leftarrow v - \|v\| e_1}_{O(m) \ \ flops}$$

$$\underbrace{u_k \leftarrow u_k / \|u_k\|}_{O(m) \ \ flops}$$

$$\underbrace{A_{k:m,k:n} \leftarrow A_{k:m,k:n} - 2u_k(u_k^* A_{k:m,k:n})}_{4(m-k+1)\times(n-k+1)+O(n) \ \ flops}$$

end for

$R \leftarrow A$

Return R

$$\begin{aligned}\text{Total } \# \text{ FLOPS} = & \quad \textstyle\sum_{k=1}^{n} (4(m-k+1)(n-k+1) + O(m)) \\ = & \quad 2mn^2 - \frac{2n^3}{3} + O(m^2)\end{aligned}$$

(Recall that Gram-Schmidt requires $2mn^2$ flops.)

- If A is square ($m = n$)

$$\text{Total } \# \text{ FLOPS} = \frac{4n^3}{3} + O(n^2)$$

(Gram-Schmidt would require $2n^3 + O(n^2)$ flops.)

$$\begin{aligned}\text{Total \# FLOPS} &= \sum_{k=1}^{n} (4(m-k+1)(n-k+1) + O(m)) \\ &= 2mn^2 - \frac{2n^3}{3} + O(m^2)\end{aligned}$$

(Recall that Gram-Schmidt requires $2mn^2$ flops.)

- If A is square ($m = n$)

$$\text{Total \# FLOPS} = \frac{4n^3}{3} + O(n^2)$$

(Gram-Schmidt would require $2n^3 + O(n^2)$ flops.)

$$\begin{aligned}\text{Total } \# \text{ FLOPS} = & \textstyle\sum_{k=1}^{n} (4(m-k+1)(n-k+1) + O(m)) \\ = & \; 2mn^2 - \tfrac{2n^3}{3} + O(m^2)\end{aligned}$$

(Recall that Gram-Schmidt requires $2mn^2$ flops.)

- If A is square ($m = n$)

$$\text{Total } \# \text{ FLOPS} = \frac{4n^3}{3} + O(n^2)$$

(Gram-Schmidt would require $2n^3 + O(n^2)$ flops.)

$$\begin{aligned}\text{Total } \# \text{ FLOPS} &= \sum_{k=1}^{n} (4(m-k+1)(n-k+1) + O(m)) \\ &= 2mn^2 - \frac{2n^3}{3} + O(m^2)\end{aligned}$$

(Recall that Gram-Schmidt requires $2mn^2$ flops.)

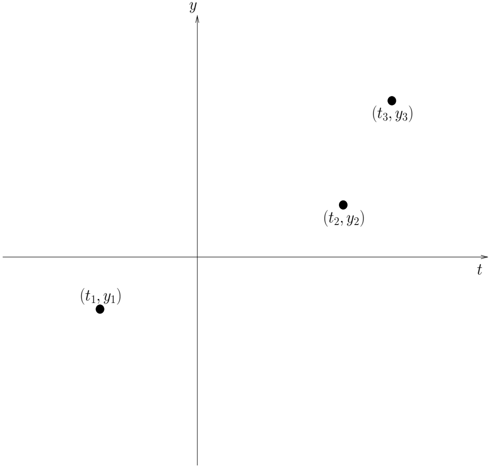
- If A is square ($m = n$)

$$\text{Total } \# \text{ FLOPS} = \frac{4n^3}{3} + O(n^2)$$

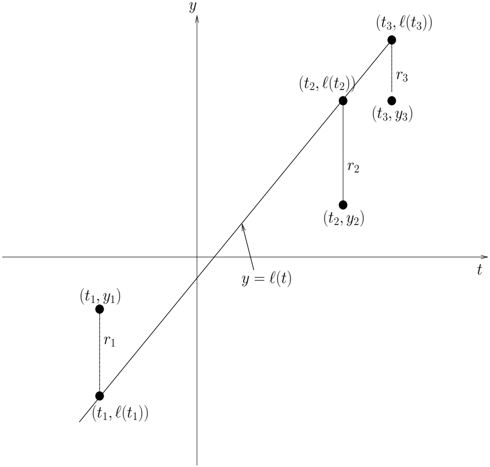
(Gram-Schmidt would require $2n^3 + O(n^2)$ flops.)

Outline

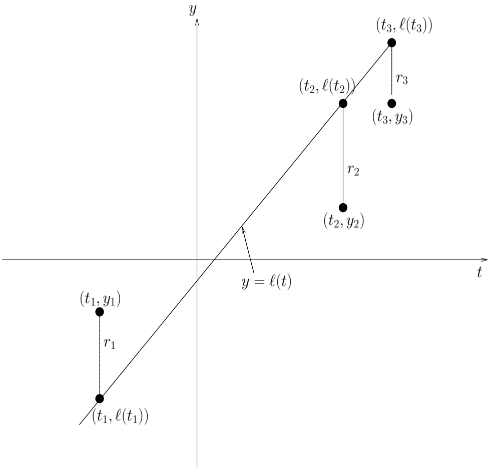
Let $p_1 = (t_1, y_1) = (-2, -1)$, $p_2 = (t_2, y_2) = (3, 1)$, $p_3 = (t_3, y_3) = (4, 3)$.



Let $p_1 = (t_1, y_1) = (-2, -1)$, $p_2 = (t_2, y_2) = (3, 1)$, $p_3 = (t_3, y_3) = (4, 3)$.



Let $p_1 = (t_1, y_1) = (-2, -1)$, $p_2 = (t_2, y_2) = (3, 1)$, $p_3 = (t_3, y_3) = (4, 3)$.



- Find the line $\ell(t) = x_1 t + x_0$ that best fits the points p_1, p_2, p_3. (The unknowns are x_0, x_1.)

- Find the line $\ell(t) = x_1 t + x_0$ so that
$$\sqrt{\sum_{i=1}^{3}(\ell(t_i) - y_i)^2} = \sqrt{(-2x_1 + x_0 - (-1))^2 + (3x_1 + x_0 - 1)^2 + (4x_1 + x_0 - 3)^2}$$
is small as possible.

- Define
$$r = \begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix} = \begin{bmatrix} \ell(t_1) - y_1 \\ \ell(t_2) - y_2 \\ \ell(t_3) - y_3 \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & -2 \\ 1 & 3 \\ 1 & 4 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_0 \\ x_1 \end{bmatrix}}_{x} - \underbrace{\begin{bmatrix} -1 \\ 1 \\ 3 \end{bmatrix}}_{b}$$

- The problem can be posed as
$$\text{find } x = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} \text{ such that } \|r\|_2 = \|Ax - b\|_2 \text{ is as small as possible.}$$

- Find the line $\ell(t) = x_1 t + x_0$ so that

$$\sqrt{\sum_{i=1}^{3}(\ell(t_i) - y_i)^2} = \sqrt{(-2x_1 + x_0 - (-1))^2 + (3x_1 + x_0 - 1)^2 + (4x_1 + x_0 - 3)^2}$$

  is small as possible.

- Define

$$r = \begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix} = \begin{bmatrix} \ell(t_1) - y_1 \\ \ell(t_2) - y_2 \\ \ell(t_3) - y_3 \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & -2 \\ 1 & 3 \\ 1 & 4 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_0 \\ x_1 \end{bmatrix}}_{x} - \underbrace{\begin{bmatrix} -1 \\ 1 \\ 3 \end{bmatrix}}_{b}$$

- The problem can be posed as

  find $x = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix}$ such that $\|r\|_2 = \|Ax - b\|_2$ is as small as possible.

- Find the line $\ell(t) = x_1 t + x_0$ so that
$$\sqrt{\sum_{i=1}^{3}(\ell(t_i) - y_i)^2} = \sqrt{(-2x_1 + x_0 - (-1))^2 + (3x_1 + x_0 - 1)^2 + (4x_1 + x_0 - 3)^2}$$
is small as possible.

- Define
$$r = \begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix} = \begin{bmatrix} \ell(t_1) - y_1 \\ \ell(t_2) - y_2 \\ \ell(t_3) - y_3 \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & -2 \\ 1 & 3 \\ 1 & 4 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_0 \\ x_1 \end{bmatrix}}_{x} - \underbrace{\begin{bmatrix} -1 \\ 1 \\ 3 \end{bmatrix}}_{b}$$

- The problem can be posed as
$$\text{find } x = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} \text{ such that } \|r\|_2 = \|Ax - b\|_2 \text{ is as small as possible.}$$

- More generally given m points in $\mathbb{R}^2$

$$p_i = (t_i, y_i), \quad i = 1, \ldots, m$$

- Suppose you want to find the polynomial of degree $n-1$ $(n < m)$ in the form

$$P(t) = x_{n-1}t^{n-1} + x_{n-2}t^{n-2} + \cdots + x_1 t + x_0$$

minimizing

$$\sqrt{\sum_{i=1}^{m} (P(t_i) - y_i)^2}.$$

- More generally given m points in $\mathbb{R}^2$

$$p_i = (t_i, y_i), \quad i = 1, \ldots, m$$

- Suppose you want to find the polynomial of degree $n-1$ $(n < m)$ in the form

$$P(t) = x_{n-1}t^{n-1} + x_{n-2}t^{n-2} + \cdots + x_1 t + x_0$$

minimizing

$$\sqrt{\sum_{i=1}^{m} (P(t_i) - y_i)^2}.$$

- More generally given m points in $\mathbb{R}^2$

$$p_i = (t_i, y_i), \quad i = 1, \dots, m$$

- Suppose you want to find the polynomial of degree $n - 1$ ($n < m$) in the form

$$P(t) = x_{n-1} t^{n-1} + x_{n-2} t^{n-2} + \cdots + x_1 t + x_0$$

minimizing

$$\sqrt{\sum_{i=1}^{m} (P(t_i) - y_i)^2}.$$

- Define

$$\underbrace{\begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_m \end{bmatrix}}_{r} = \begin{bmatrix} P(t_1) - y_1 \\ P(t_2) - y_2 \\ \vdots \\ P(t_m) - y_m \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & \cdots & t_1^{n-2} & t_1^{n-1} \\ 1 & \cdots & t_2^{n-2} & t_2^{n-1} \\ & & \vdots & \vdots \\ 1 & \cdots & t_m^{n-2} & t_m^{n-1} \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n-1} \end{bmatrix}}_{x} - \underbrace{\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}}_{b}$$

Remark: The matrix A is called the *Vandermonde* matrix.

- We want to find $x = \begin{bmatrix} x_0 & x_1 & \cdots & x_{n-1} \end{bmatrix}^T$ minimizing

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

- Define

$$\underbrace{\begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_m \end{bmatrix}}_{r} = \begin{bmatrix} P(t_1) - y_1 \\ P(t_2) - y_2 \\ \vdots \\ P(t_m) - y_m \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & \cdots & t_1^{n-2} & t_1^{n-1} \\ 1 & \cdots & t_2^{n-2} & t_2^{n-1} \\ & & \vdots & \vdots \\ 1 & \cdots & t_m^{n-2} & t_m^{n-1} \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n-1} \end{bmatrix}}_{x} - \underbrace{\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}}_{b}$$

Remark: The matrix A is called the *Vandermonde* matrix.

- We want to find $x = \begin{bmatrix} x_0 & x_1 & \cdots & x_{n-1} \end{bmatrix}^T$ minimizing

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

- Define

$$\underbrace{\begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_m \end{bmatrix}}_{r} = \begin{bmatrix} P(t_1) - y_1 \\ P(t_2) - y_2 \\ \vdots \\ P(t_m) - y_m \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & \dots & t_1^{n-2} & t_1^{n-1} \\ 1 & \dots & t_2^{n-2} & t_2^{n-1} \\ & & \vdots & \vdots \\ 1 & \dots & t_m^{n-2} & t_m^{n-1} \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n-1} \end{bmatrix}}_{x} - \underbrace{\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}}_{b}$$

<u>Remark:</u> The matrix A is called the *Vandermonde* matrix.

- We want to find $x = \begin{bmatrix} x_0 & x_1 & \cdots & x_{n-1} \end{bmatrix}^T$ minimizing

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

Definition

An $m \times n$ system $Ax = b$ is called *overdetermined* if $m > n$.

- Overdetermined systems are usually inconsistent. (*e.g.* It is unlikely that three lines in $\mathbf{R}^2$ intersect each other at a common point.)

Example:

$$[\,A\,|\,b\,] = \begin{bmatrix} 1 & -2 & -1 \\ 1 & 3 & 1 \\ 1 & 4 & 2 \end{bmatrix} \rightsquigarrow \underbrace{\begin{bmatrix} 1 & -2 & -1 \\ 0 & 5 & 2 \\ 0 & 0 & 3/5 \end{bmatrix}}_{\textit{inconsistent}}$$

Definition

An $m \times n$ system $Ax = b$ is called *overdetermined* if $m > n$.

- Overdetermined systems are usually inconsistent. (*e.g.* It is unlikely that three lines in $\mathbf{R}^2$ intersect each other at a common point.)

Example:

$$[\,A \mid b\,] = \begin{bmatrix} 1 & -2 & -1 \\ 1 & 3 & 1 \\ 1 & 4 & 2 \end{bmatrix} \rightsquigarrow \underbrace{\begin{bmatrix} 1 & -2 & -1 \\ 0 & 5 & 2 \\ 0 & 0 & 3/5 \end{bmatrix}}_{\textit{inconsistent}}$$

Definition

An $m \times n$ system $Ax = b$ is called *overdetermined* if $m > n$.

- Overdetermined systems are usually inconsistent. (*e.g.* It is unlikely that three lines in $\mathbf{R}^2$ intersect each other at a common point.)

Example:

$$[\, A \mid b \,] = \begin{bmatrix} 1 & -2 & -1 \\ 1 & 3 & 1 \\ 1 & 4 & 2 \end{bmatrix} \rightsquigarrow \underbrace{\begin{bmatrix} 1 & -2 & -1 \\ 0 & 5 & 2 \\ 0 & 0 & 3/5 \end{bmatrix}}_{\textit{inconsistent}}$$

Justification:

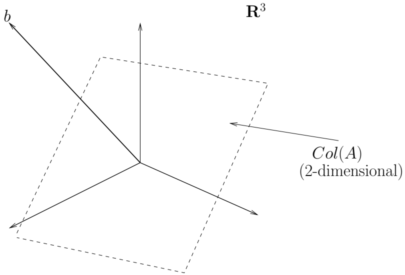
$\mathrm{range}(A) = \mathrm{span}\{a_1, a_2, \ldots, a_n\}$ is at most an n-dimen subspace in $\mathbb{C}^m$

$$\Longrightarrow$$

Most $b \in \mathbb{C}^m$ are not in $\mathrm{range}(A)$

$$\Longrightarrow$$

$Ax = b$ is inconsistent for most $b \in \mathbb{C}^m$



e.g. $m = 3$, $n = 2$

$$A = \begin{bmatrix} x & x \\ x & x \\ x & x \end{bmatrix}$$

Justification:

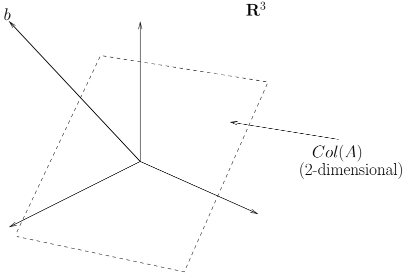
$\mathrm{range}(A) = \mathrm{span}\{a_1, a_2, \ldots, a_n\}$ is at most an n-dimen subspace in $\mathbb{C}^m$

$$\Longrightarrow$$

Most $b \in \mathbb{C}^m$ are not in $\mathrm{range}(A)$

$$\Longrightarrow$$

$Ax = b$ is inconsistent for most $b \in \mathbb{C}^m$

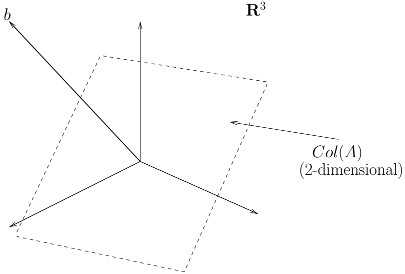


e.g. $m = 3$, $n = 2$

$$A = \begin{bmatrix} x & x \\ x & x \\ x & x \end{bmatrix}$$

Justification:

$$\boxed{\begin{array}{c}\mathrm{range}(A) = \mathrm{span}\{a_1, a_2, \ldots, a_n\} \text{ is at most an } n\text{-dimen subspace in } \mathbb{C}^m \\ \Longrightarrow \\ \text{Most } b \in \mathbb{C}^m \text{ are not in } \mathrm{range}(A) \\ \Longrightarrow \\ Ax = b \text{ is inconsistent for most } b \in \mathbb{C}^m\end{array}}$$



e.g. $m = 3$, $n = 2$

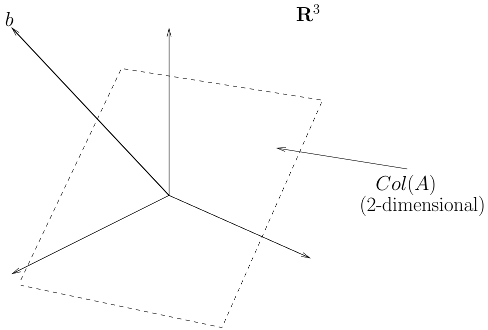
$$A = \begin{bmatrix} x & x \\ x & x \\ x & x \end{bmatrix}$$

Least Squares Problem

Given an overdetermined system $Ax = b$.

Find $x \in \mathbb{C}^n$ such that $\|Ax - b\|_2$ is as small as possible.

- Geometric interpretation: Find the point on the hyperplane $\mathrm{range}(A)$ that is closest to b.



- US population as a function of time

t	y (population)
1900	75.995
1910	91.972
1920	105.711
1930	123.203
1940	131.669
1950	150.697
1960	179.323
1970	203.212
1980	226.505
1990	249.633
2000	281.422

- Fit a cubic model $y \approx p(t) = x_3 t^3 + x_2 t^2 + x_1 t + x_0$ approximating the US population by solving the least squares problem. Use it to estimate population in 2020.

- US population as a function of time

t	y (population)
1900	75.995
1910	91.972
1920	105.711
1930	123.203
1940	131.669
1950	150.697
1960	179.323
1970	203.212
1980	226.505
1990	249.633
2000	281.422

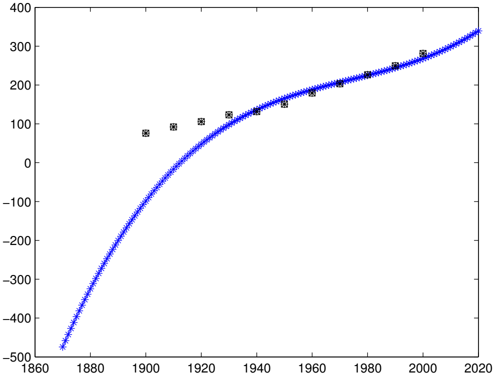
- Fit a cubic model $y \approx p(t) = x_3 t^3 + x_2 t^2 + x_1 t + x_0$ approximating the US population by solving the least squares problem. Use it to estimate population in 2020.

Need to find $x = \begin{bmatrix} x_0 & x_1 & x_2 & x_3 \end{bmatrix}^T \in \mathbb{C}^4$ minimizing

$$\left\| \underbrace{\begin{bmatrix} 1 & 1900 & 1900^2 & 1900^3 \\ 1 & 1910 & 1910^2 & 1910^3 \\ 1 & 1920 & 1920^2 & 1920^3 \\ 1 & 1930 & 1930^2 & 1930^3 \\ 1 & 1940 & 1940^2 & 1940^3 \\ 1 & 1950 & 1950^2 & 1950^3 \\ 1 & 1960 & 1960^2 & 1960^3 \\ 1 & 1970 & 1970^2 & 1970^3 \\ 1 & 1980 & 1980^2 & 1980^3 \\ 1 & 1990 & 1990^2 & 1990^3 \\ 1 & 2000 & 2000^2 & 2000^3 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}}_{x} - \underbrace{\begin{bmatrix} 75.995 \\ 91.972 \\ 105.711 \\ 123.203 \\ 131.669 \\ 150.697 \\ 179.323 \\ 203.212 \\ 226.505 \\ 249.633 \\ 281.422 \end{bmatrix}}_{b} \right\|$$

The optimal cubic polynomial solving the least squares problem

$$p(t) = 56.0821\left(\frac{t-1950}{50}\right)^3 + 127.3056\left(\frac{t-1950}{50}\right)^2 - 80.6311\left(\frac{t-1950}{50}\right) + 165.3947$$



Black squares - given pairs of (year,population) data; Blue curve - optimal cubic polynomial