Análisis Numérico

Primer cuatrimestre 2023

Descomposición QR

Outline

1 Descomposición QR reducida

Outline

1 Descomposición QR reducida

Descomposición QR reducida

Sea $A \in \mathbb{C}^{m \times n}$ con $\text{rank}\,(A) = n$ $(m \geq n)$.

Descomposición QR reducida

Sea $A \in \mathbb{C}^{m \times n}$ con $\operatorname{rank}(A) = n$ $(m \geq n)$.
Llamamos $a_1, \ldots, a_n$ a las columnas de A.

Descomposición QR reducida

Sea $A \in \mathbb{C}^{m \times n}$ con $\text{rank}\,(A) = n$ $(m \geq n)$.
Llamamos $a_1, \ldots, a_n$ a las columnas de A.

Idea. Hallar vectores $\{q_1, q_2, \ldots, q_n\}$ **ortonormales** tales que

$$\langle q_1, \ldots, q_j \rangle = \langle a_1, \ldots, a_j \rangle \qquad j = 1, 2, \ldots, n$$

Descomposición QR reducida

Sea $A \in \mathbb{C}^{m \times n}$ con $\text{rank}\,(A) = n$ $(m \geq n)$.
Llamamos $a_1, \ldots, a_n$ a las columnas de A.

Idea. Hallar vectores $\{q_1, q_2, \ldots, q_n\}$ **ortonormales** tales que

$$\langle q_1, \ldots, q_j \rangle = \langle a_1, \ldots, a_j \rangle \qquad j = 1, 2, \ldots, n$$

Por lo tanto tenemos

$$\begin{aligned}
a_1 &= r_{11} q_1 \\
a_2 &= r_{12} q_1 + r_{22} q_2 \\
a_3 &= r_{13} q_1 + r_{23} q_2 + r_{33} q_3 \\
&\vdots \\
a_n &= r_{1n} q_1 + r_{2n} q_2 + r_{3n} q_3 + \cdots + r_{nn} q_n
\end{aligned}$$

Descomposición QR reducida

En forma matricial tenemos

$$\left[\begin{array}{c|c|c|c} a_1 & a_2 & \cdots & a_n \end{array}\right] = \left[\begin{array}{c|c|c|c} q_1 & q_2 & \cdots & q_n \end{array}\right] \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ 0 & r_{22} & \cdots & r_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & r_{nn} \end{bmatrix}$$

Descomposición QR reducida

En forma matricial tenemos

$$\left[\begin{array}{c|c|c|c} a_1 & a_2 & \cdots & a_n \end{array}\right] = \left[\begin{array}{c|c|c|c} q_1 & q_2 & \cdots & q_n \end{array}\right] \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ 0 & r_{22} & \cdots & r_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & r_{nn} \end{bmatrix}$$

o

Descomposición QR reducida

En forma matricial tenemos

$$\left[\begin{array}{c|c|c|c} a_1 & a_2 & \cdots & a_n \end{array}\right] = \left[\begin{array}{c|c|c|c} q_1 & q_2 & \cdots & q_n \end{array}\right] \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ 0 & r_{22} & \cdots & r_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & r_{nn} \end{bmatrix}$$

o

$$A = \widehat{Q}\widehat{R}$$

Descomposición QR reducida

Los vectores $q_1, q_2, \ldots, q_n$ pueden obtenerse aplicando la ortogonalización de Gram-Schmidt a $a_1, a_2, \ldots, a_n$:

Descomposición QR reducida

Los vectores $q_1, q_2, \ldots, q_n$ pueden obtenerse aplicando la ortogonalización de Gram-Schmidt a $a_1, a_2, \ldots, a_n$:

$$q_1 = \frac{a_1}{r_{11}}, \qquad \text{con } r_{11} = \pm\|a_1\|_2$$

Descomposición QR reducida

Los vectores $q_1, q_2, \ldots, q_n$ pueden obtenerse aplicando la ortogonalización de Gram-Schmidt a $a_1, a_2, \ldots, a_n$:

$$q_1 = \frac{a_1}{r_{11}}, \qquad \text{con } r_{11} = \pm \|a_1\|_2$$

y si $q_1, \ldots, q_{j-1}$ ya fueron hallados, entonces

$$q_j = \frac{a_j - r_{1j} q_1 - \cdots - r_{j-1,j} q_{j-1}}{r_{jj}},$$

con

$$r_{ij} = q_i^* a_j, \qquad i = 1, 2, \ldots, j-1$$

$$r_{jj} = \pm \left\| \frac{a_j - r_{1j} q_1 - \cdots - r_{j-1,j} q_{j-1}}{r_{jj}} \right\|_2$$

Descomposición QR reducida

Los vectores $q_1, q_2, \ldots, q_n$ pueden obtenerse aplicando la ortogonalización de Gram-Schmidt a $a_1, a_2, \ldots, a_n$:

$$q_1 = \frac{a_1}{r_{11}}, \qquad \text{con } r_{11} = \pm \|a_1\|_2$$

y si $q_1, \ldots, q_{j-1}$ ya fueron hallados, entonces

$$q_j = \frac{a_j - r_{1j} q_1 - \cdots - r_{j-1,j} q_{j-1}}{r_{jj}},$$

con

$$r_{ij} = q_i^* a_j, \qquad i = 1, 2, \ldots, j-1$$

$$r_{jj} = \pm \left\| \frac{a_j - r_{1j} q_1 - \cdots - r_{j-1,j} q_{j-1}}{r_{jj}} \right\|_2$$

Notar que el signo de r_{jj} no está definido

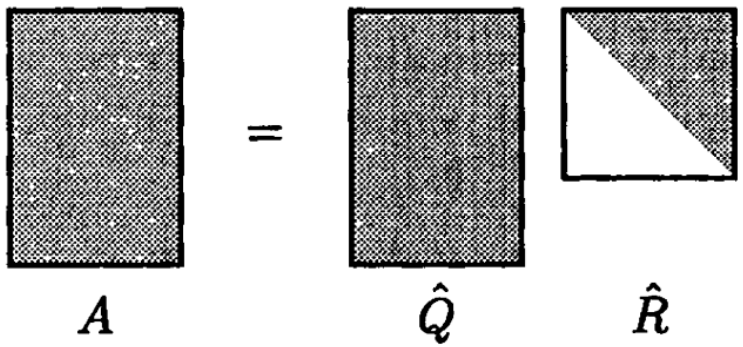
Descomposición QR reducida

Algoritmo: Gram–Schmidt clásico ($\text{rank}\,(A) = n$)

```
1  for j = 1 : n
2     v_j   = a_j
3     for i = 1 : j-1
4        r_ij  = q_i^* a_j
5        v_j   = v_j  -  r_ij q_i
6     r_jj  = ||v_j||_2
7     q_j   = v_j  /  r_jj
8
```

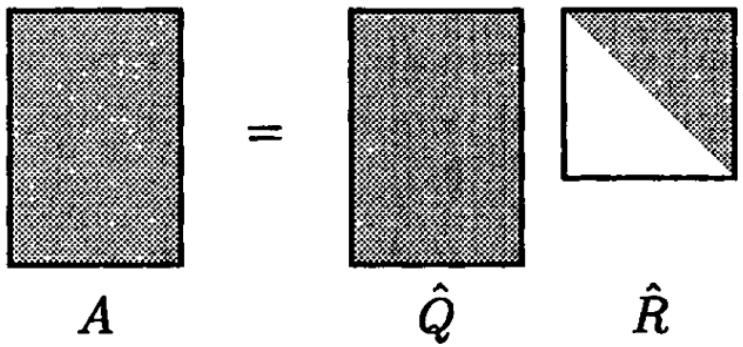
Descomposición QR completa

Descomposición QR reducida



Descomposición QR completa

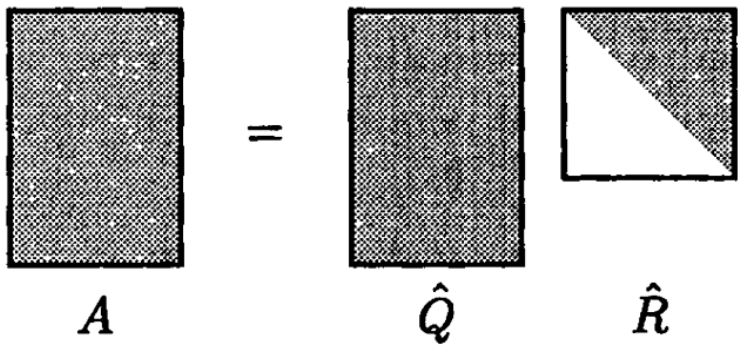
Descomposición QR reducida



Descomposición QR completa

Descomposición QR completa

Descomposición QR reducida

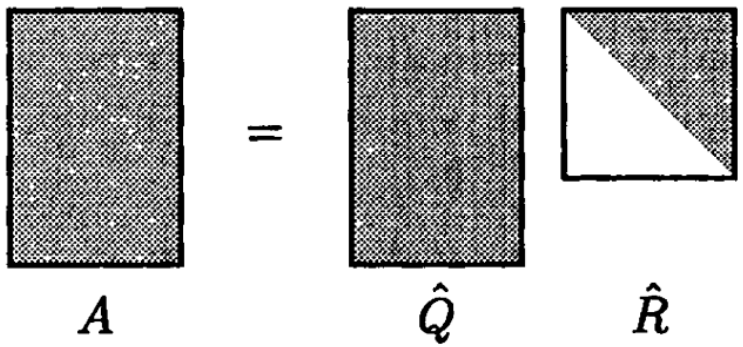


Descomposición QR completa

- Completamos $\{q_1, \ldots, q_n\}$ a una base ortonormal de $\mathbb{C}^m$

Descomposición QR completa

Descomposición QR reducida

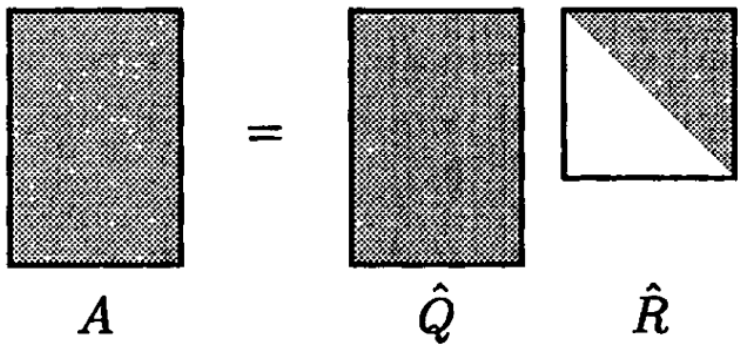


Descomposición QR completa

- Completamos $\{q_1, \ldots, q_n\}$ a una base ortonormal de $\mathbb{C}^m$
- Agregamos $m - n$ filas de ceros a $\widehat{R}$

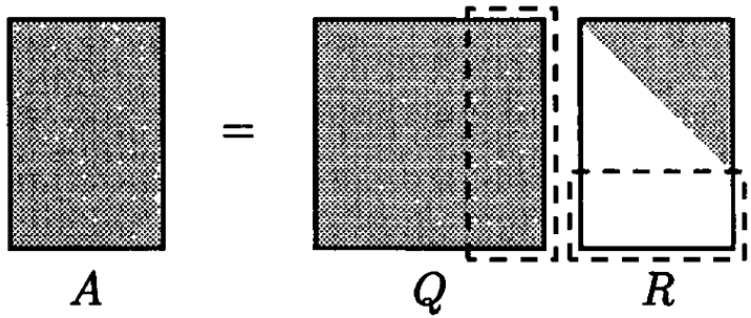
Descomposición QR completa

Descomposición QR reducida



Descomposición QR completa

- Completamos $\{q_1, \ldots, q_n\}$ a una base ortonormal de $\mathbb{C}^m$
- Agregamos $m - n$ filas de ceros a $\widehat{R}$



Descomposición QR completa

Teorema. Toda matriz en $\mathbb{C}^{m\times n}$ $(m \geq n)$ tiene una factorización QR (completa y reducida)

Descomposición QR completa

Teorema. Toda matriz en $\mathbb{C}^{m\times n}$ $(m \geq n)$ tiene una factorización QR (completa y reducida)

Proof. Si $\operatorname{rank}(A) = n$ basta ver que el algoritmo GS clásico termina. En este caso las columnas $a_1, \ldots, a_n$ son l.i., y por lo tanto ningún v_j es nulo (si $v_j = 0$ entonces $a_j \in \langle a_1, \ldots, a_{j-1} \rangle$ que es absurdo.)

Descomposición QR completa

Teorema. Toda matriz en $\mathbb{C}^{m\times n}$ $(m \geq n)$ tiene una factorización QR (completa y reducida)

Proof. Si $\operatorname{rank}(A) = n$ basta ver que el algoritmo GS clásico termina. En este caso las columnas $a_1, \ldots, a_n$ son l.i., y por lo tanto ningún v_j es nulo (si $v_j = 0$ entonces $a_j \in \langle a_1, \ldots, a_{j-1}\rangle$ que es absurdo.)
Si $\operatorname{rank}(A) < n$ entonces uno o más v_j son nulos. En este caso elegimos q_j arbitrario, ortonormal con $q_i, i = 1, \ldots, j-1$. Entonces el algoritmo continúa normalmente.

Descomposición QR completa

Teorema. Toda matriz en $\mathbb{C}^{m\times n}$ $(m \geq n)$ tiene una factorización QR (completa y reducida)

Proof. Si $\operatorname{rank}(A) = n$ basta ver que el algoritmo GS clásico termina. En este caso las columnas $a_1, \ldots, a_n$ son l.i., y por lo tanto ningún v_j es nulo (si $v_j = 0$ entonces $a_j \in \langle a_1, \ldots, a_{j-1}\rangle$ que es absurdo.)
Si $\operatorname{rank}(A) < n$ entonces uno o más v_j son nulos. En este caso elegimos q_j arbitrario, ortonormal con $q_i, i = 1, \ldots, j-1$. Entonces el algoritmo continúa normalmente.

La factorización completa resulta de completar las matrices $\widehat{Q}$ y $\widehat{R}$ como ya mencionamos.

Descomposición QR completa

Teorema. Toda matriz en $\mathbb{C}^{m\times n}$ $(m \geq n)$ tiene una factorización QR (completa y reducida)

Proof. Si $\text{rank}\,(A) = n$ basta ver que el algoritmo GS clásico termina. En este caso las columnas $a_1, \ldots, a_n$ son l.i., y por lo tanto ningún v_j es nulo (si $v_j = 0$ entonces $a_j \in \langle a_1, \ldots, a_{j-1}\rangle$ que es absurdo.)
Si $\text{rank}\,(A) < n$ entonces uno o más v_j son nulos. En este caso elegimos q_j arbitrario, ortonormal con $q_i, i = 1, \ldots, j-1$. Entonces el algoritmo continúa normalmente.
La factorización completa resulta de completar las matrices $\widehat{Q}$ y $\widehat{R}$ como ya mencionamos.

Teorema. Toda matriz $m \times n$ de rango n tiene una única factorización QR reducida, con $\widehat{R}$ con entradas > 0 en la diagonal

Algoritmo GS modificado

Con la notación del algoritmo GS clásico tenemos

$$q_j = \frac{v_j}{\|v_j\|_2}$$

Algoritmo GS modificado

Con la notación del algoritmo GS clásico tenemos

$$q_j = \frac{v_j}{\|v_j\|_2}$$

con

$$v_j = a_j - \sum_{i=1}^{j-1} (q_i^* a_j)\, q_i$$

Algoritmo GS modificado

Con la notación del algoritmo GS clásico tenemos

$$q_j = \frac{v_j}{\|v_j\|_2}$$

con

$$v_j = a_j - \sum_{i=1}^{j-1} (q_i^* a_j)\, q_i$$

Entonces

Algoritmo GS modificado

Con la notación del algoritmo GS clásico tenemos

$$q_j = \frac{v_j}{\|v_j\|_2}$$

con

$$v_j = a_j - \sum_{i=1}^{j-1} (q_i^* a_j)\, q_i$$

Entonces

- $v_j \perp \langle q_1, \ldots, q_{j-1} \rangle$

Algoritmo GS modificado

Con la notación del algoritmo GS clásico tenemos

$$q_j = \frac{v_j}{\|v_j\|_2}$$

con

$$v_j = a_j - \sum_{i=1}^{j-1} (q_i^* a_j)\, q_i$$

Entonces

- $v_j \perp \langle q_1, \ldots, q_{j-1} \rangle$
- Si $w \perp \langle q_1, \ldots, q_{j-1} \rangle$ entonces $v_j^* w = a_j^* w$

Algoritmo GS modificado

Con la notación del algoritmo GS clásico tenemos

$$q_j = \frac{v_j}{\|v_j\|_2}$$

con

$$v_j = a_j - \sum_{i=1}^{j-1} (q_i^* a_j)\, q_i$$

Entonces

- $v_j \perp \langle q_1, \ldots, q_{j-1} \rangle$
- Si $w \perp \langle q_1, \ldots, q_{j-1} \rangle$ entonces $v_j^* w = a_j^* w$

Si P_j es la proyección ortogonal sobre $\langle q_1, \ldots, q_{j-1} \rangle^\perp$ entonces $v_j = P_j a_j$

Algoritmo GS modificado

Con la notación del algoritmo GS clásico tenemos

$$q_j = \frac{v_j}{\|v_j\|_2}$$

con

$$v_j = a_j - \sum_{i=1}^{j-1} (q_i^* a_j)\, q_i$$

Entonces

- $v_j \perp \langle q_1, \ldots, q_{j-1} \rangle$
- Si $w \perp \langle q_1, \ldots, q_{j-1} \rangle$ entonces $v_j^* w = a_j^* w$

Si P_j es la proyección ortogonal sobre $\langle q_1, \ldots, q_{j-1} \rangle^\perp$ entonces $v_j = P_j a_j$ y

$$q_j = \frac{P_j a_j}{\|P_j a_j\|_2}$$

Algoritmo GS modificado

Proyección ortogonal

Algoritmo GS modificado

Proyección ortogonal

Si $\{q_1, q_2, \ldots, q_j\}$ son ortonormales, formamos la matriz

$$\widehat{Q}_j = \left[\; q_1 \;\middle|\; q_2 \;\middle|\; \cdots \;\middle|\; q_n \; \right]$$

Algoritmo GS modificado

Proyección ortogonal

Si $\{q_1, q_2, \ldots, q_j\}$ son ortonormales, formamos la matriz

$$\widehat{Q}_j = \left[\; q_1 \;\middle|\; q_2 \;\middle|\; \cdots \;\middle|\; q_n \; \right]$$

Entonces

- $P = \widehat{Q}_j \widehat{Q}_j^*$ es la proyección sobre $\langle q_1, \ldots, q_j \rangle$

Algoritmo GS modificado

Proyección ortogonal

Si $\{q_1, q_2, \ldots, q_j\}$ son ortonormales, formamos la matriz

$$\widehat{Q}_j = \left[\; q_1 \;\middle|\; q_2 \;\middle|\; \cdots \;\middle|\; q_n \; \right]$$

Entonces

- $P = \widehat{Q}_j \widehat{Q}_j^*$ es la proyección sobre $\langle q_1, \ldots, q_j \rangle$
- $I - P = I - \widehat{Q}_j \widehat{Q}_j^*$ es la proyección sobre $\langle q_1, \ldots, q_j \rangle^\perp$

Algoritmo GS modificado

Proyección ortogonal

Si $\{q_1, q_2, \ldots, q_j\}$ son ortonormales, formamos la matriz

$$\widehat{Q}_j = \left[\; q_1 \;\middle|\; q_2 \;\middle|\; \cdots \;\middle|\; q_n \; \right]$$

Entonces

- $P = \widehat{Q}_j \widehat{Q}_j^*$ es la proyección sobre $\langle q_1, \ldots, q_j \rangle$
- $I - P = I - \widehat{Q}_j \widehat{Q}_j^*$ es la proyección sobre $\langle q_1, \ldots, q_j \rangle^\perp$
- Si P_q y $P_{\perp q}$ denotan las proyecciones sobre $\langle q \rangle$ y $\langle q \rangle^\perp$, entonces

$$P_q = qq^* \qquad \text{y} \qquad P_{\perp q} = I - qq^* \qquad \text{(q unitario)}$$

Algoritmo GS modificado

Proyección ortogonal

Si $\{q_1, q_2, \ldots, q_j\}$ son ortonormales, formamos la matriz

$$\widehat{Q}_j = \left[\; q_1 \;\middle|\; q_2 \;\middle|\; \cdots \;\middle|\; q_n \; \right]$$

Entonces

- $P = \widehat{Q}_j \widehat{Q}_j^*$ es la proyección sobre $\langle q_1, \ldots, q_j \rangle$
- $I - P = I - \widehat{Q}_j \widehat{Q}_j^*$ es la proyección sobre $\langle q_1, \ldots, q_j \rangle^\perp$
- Si P_q y $P_{\perp q}$ denotan las proyecciones sobre $\langle q \rangle$ y $\langle q \rangle^\perp$, entonces

$$P_q = qq^* \qquad \text{y} \qquad P_{\perp q} = I - qq^* \qquad \text{(q unitario)}$$

- Como en la slide anterior, sea P_j la proyección sobre $\langle q_1, \ldots, q_{j-1} \rangle$, entonces tenemos

$$P_j = I - \widehat{Q}_{j-1} \widehat{Q}_{j-1}^* = P_{\perp q_{j-1}} P_{\perp q_{j-2}} \cdots P_{\perp q_1}$$

Algoritmo GS modificado

Entonces $v_j = P_j a_j$ puede calcularse iterativamente por

$$v_j^{(1)} = a_j$$

Algoritmo GS modificado

Entonces $v_j = P_j a_j$ puede calcularse iterativamente por

$$v_j^{(1)} = a_j$$
$$v_j^{(2)} = P_{\perp q_1} v_j^{(1)} = v_j^{(1)} - q_1 q_1^* v_j^{(1)}$$

Algoritmo GS modificado

Entonces $v_j = P_j a_j$ puede calcularse iterativamente por

$$
\begin{aligned}
v_j^{(1)} &= a_j \\
v_j^{(2)} &= P_{\perp q_1} v_j^{(1)} = v_j^{(1)} - q_1 q_1^* v_j^{(1)} \\
v_j^{(3)} &= P_{\perp q_2} v_j^{(2)} = v_j^{(2)} - q_2 q_2^* v_j^{(2)}
\end{aligned}
$$

Algoritmo GS modificado

Entonces $v_j = P_j a_j$ puede calcularse iterativamente por

$$
\begin{aligned}
v_j^{(1)} &= a_j \\
v_j^{(2)} &= P_{\perp q_1} v_j^{(1)} = v_j^{(1)} - q_1 q_1^* v_j^{(1)} \\
v_j^{(3)} &= P_{\perp q_2} v_j^{(2)} = v_j^{(2)} - q_2 q_2^* v_j^{(2)} \\
&\vdots
\end{aligned}
$$

Algoritmo GS modificado

Entonces $v_j = P_j a_j$ puede calcularse iterativamente por

$$
\begin{aligned}
v_j^{(1)} &= a_j \\
v_j^{(2)} &= P_{\perp q_1} v_j^{(1)} = v_j^{(1)} - q_1 q_1^* v_j^{(1)} \\
v_j^{(3)} &= P_{\perp q_2} v_j^{(2)} = v_j^{(2)} - q_2 q_2^* v_j^{(2)} \\
&\vdots \\
v_j &= v_j^{(j)} = P_{\perp q_{j-1}} v_j^{(j-1)} = v_j^{(j-1)} - q_{j-1} q_{j-1}^* v_j^{(j-1)}
\end{aligned}
$$

Algoritmo GS modificado

Algoritmo: Gram–Schmidt modificado ($\text{rank}\,(A) = n$)

```
1  for i = 1 : n
2     v_i  = a_i
3  for i = 1 : n
4     r_ii = ||v_i||
5     q_i  = v_i / r_ii
6     for j = i+1 : n
7        r_ij = q_i^* v_j
8        v_j  = v_j - r_ij q_i
9
```

Algoritmo GS modificado

Ortogonalización Triangular

Algoritmo GS modificado

Ortogonalización Triangular

Puede verse que el algoritmo GS modificado transforma la matriz

$$\left[\begin{array}{c|c|c|c} v_1 & v_2 & \cdots & v_n \end{array}\right]$$

en

$$\left[\begin{array}{c|c|c|c} v_1 & v_2 & \cdots & v_n \end{array}\right] \underbrace{R_1 R_2 \ldots R_n}_{\widehat{R}^{-1}} = \underbrace{\left[\begin{array}{c|c|c|c} q_1 & q_2 & \cdots & q_n \end{array}\right]}_{\widehat{Q}}$$

siendo R_i la matriz triangular superior

$$R_i = \begin{bmatrix} 1 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \cdots & \vdots \\ 0 & \cdots & \frac{1}{r_{ii}} & -\frac{r_{i,i+1}}{r_{ii}} & \cdots & -\frac{r_{i,n}}{r_{ii}} \\ 0 & \cdots & 0 & 1 & \cdots & 0 \\ \vdots & \cdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & 1 \end{bmatrix}$$

Algoritmo de Householder

Triangulación ortogonal

Algoritmo de Householder

Triangulación ortogonal

Idea. multiplicar a izquierda a A por matrices unitarias Q_k tal que

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

con R triangular superior

Algoritmo de Householder

Triangulación ortogonal

Idea. multiplicar a izquierda a A por matrices unitarias Q_k tal que

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

con R triangular superior

Cada matriz Q_k introduce ceros bajo la diagonal en la columna k (preservando los que ya fueron introducidos)

$$\underset{A}{\begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{bmatrix}} \xrightarrow{Q_1} \underset{Q_1A}{\begin{bmatrix} \times & \times & \times \\ 0 & \times & \times \\ 0 & \times & \times \\ 0 & \times & \times \\ 0 & \times & \times \end{bmatrix}} \xrightarrow{Q_2} \underset{Q_2Q_1A}{\begin{bmatrix} \times & \times & \times \\ & \times & \times \\ & 0 & \times \\ & 0 & \times \\ & 0 & \times \end{bmatrix}} \xrightarrow{Q_3} \underset{Q_3Q_2Q_1A}{\begin{bmatrix} \times & \times & \times \\ & \times & \times \\ & & \times \\ & & 0 \\ & & 0 \end{bmatrix}}$$

Algoritmo de Householder

- Q_k opera sobre las filas $k, k+1, \ldots, m$ introduciendo 0 en los lugares $(k+1, k), \ldots, (m, k)$

Algoritmo de Householder

- Q_k opera sobre las filas $k, k+1, \ldots, m$ introduciendo 0 en los lugares $(k+1, k), \ldots, (m, k)$
- Se elige Q_k como

$$Q_k = \begin{bmatrix} I & 0 \\ 0 & F \end{bmatrix} \qquad \begin{array}{l} I = \mathrm{id}(k-1, k-1) \\ F \text{ matriz } m-k+1 \times m-k+1 \end{array}$$

Algoritmo de Householder

- Q_k opera sobre las filas $k, k+1, \ldots, m$ introduciendo 0 en los lugares $(k+1, k), \ldots, (m, k)$
- Se elige Q_k como

$$Q_k = \begin{bmatrix} I & 0 \\ 0 & F \end{bmatrix} \qquad \begin{array}{l} I = \mathrm{id}(k-1, k-1) \\ F \text{ matriz } m-k+1 \times m-k+1 \end{array}$$

- Si $x \in \mathbb{C}^{m-k+1}$ son las entradas $(k:m, k)$ de $Q_{k-1} \cdots Q_1 A$ entonces F actúa sobre x como

$$x = \begin{bmatrix} * \\ * \\ * \\ \vdots \\ * \end{bmatrix} \xrightarrow{F} Fx = \begin{bmatrix} \pm\|x\|_2 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \pm\|x\|_2 e_1$$

Algoritmo de Householder

Reflector de Householder

Algoritmo de Householder

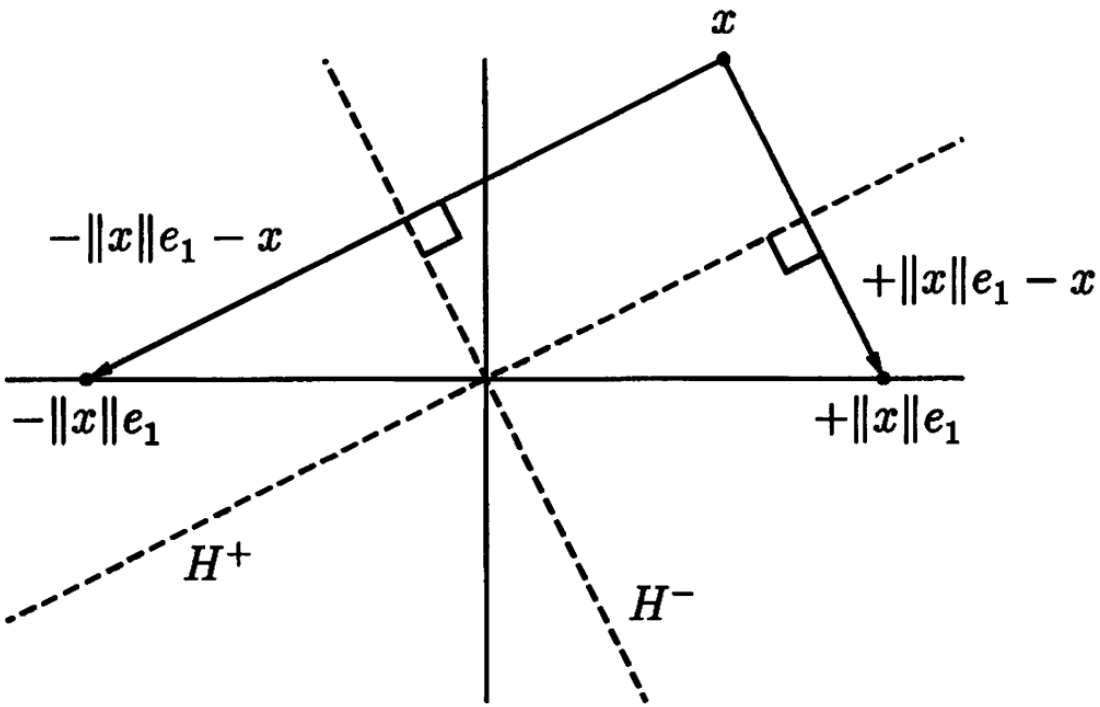
Reflector de Householder

F: refleja el espacio $\mathbb{C}^{m-k+1}$ en el hiperplano ortogonal a $v = \pm\|x\|_2 e_1 - x$

Algoritmo de Householder

Reflector de Householder

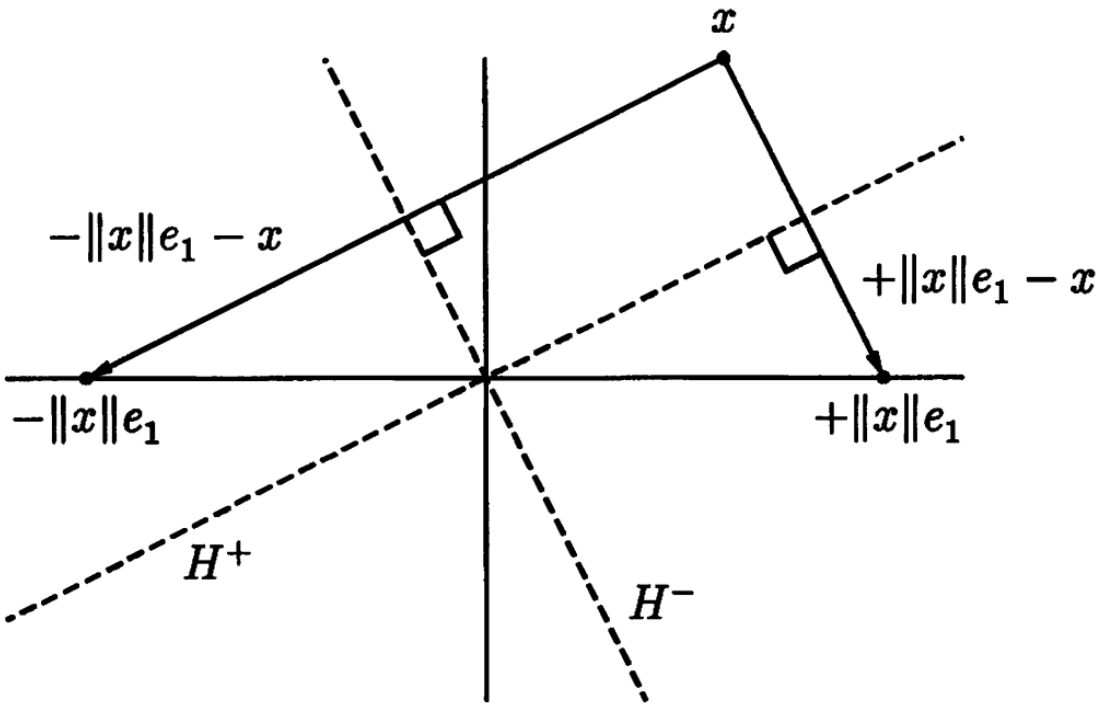
F: refleja el espacio $\mathbb{C}^{m-k+1}$ en el hiperplano ortogonal a $v = \pm\|x\|_2 e_1 - x$



Algoritmo de Householder

Reflector de Householder

F: refleja el espacio $\mathbb{C}^{m-k+1}$ en el hiperplano ortogonal a $v = \pm\|x\|_2 e_1 - x$



$$F = I - 2\frac{vv^*}{v^*v}$$

Algoritmo de Householder

- Entre $v = \|x\|_2 e_1 - x$ o $v = -\|x\|_2 e_1 - x$ se elige el de mayor módulo:

$$v = -\text{sign}\,(x_1)\|x\|_2 e_1 - x$$

Algoritmo de Householder

- Entre $v = \|x\|_2 e_1 - x$ o $v = -\|x\|_2 e_1 - x$ se elige el de mayor módulo:

$$v = -\text{sign}\,(x_1)\|x\|_2 e_1 - x$$

- Notar la diferencia entre

$$P = I - \frac{vv^*}{v^*v} \qquad \text{y} \qquad F = I - 2\frac{vv^*}{v^*v}$$

Algoritmo de Householder

- Entre $v = \|x\|_2 e_1 - x$ o $v = -\|x\|_2 e_1 - x$ se elige el de mayor módulo:

$$v = -\text{sign}\,(x_1)\|x\|_2 e_1 - x$$

- Notar la diferencia entre

$$P = I - \frac{vv^*}{v^*v} \qquad \text{y} \qquad F = I - 2\frac{vv^*}{v^*v}$$

P y F son matrices $(m-k+1)\times(m-k+1)$ que representan

- P: proyección sobre $\langle v\rangle^\perp$, con $\text{rank}\,(P) = m-k$
- F: reflector respecto del espacio $\langle v\rangle^\perp$, con $\text{rank}\,(F) = m-k+1$

Algoritmo de Householder

Algoritmo: Householder, $A \in \mathbb{C}^{m \times n}$ con $\text{rank}\,(A) = n$

```
for k = 1 : n
   x = A(k:m,k)
   v_k = sign(x_1)‖x‖_2 e_1 + x
   v_k = v_k / ‖v_k‖_2
   A(k:m,k:n) = A(k:m,k:n) - 2 v_k v_k^* A(k:m,k:n)

```

Algoritmo de Householder

Algoritmo: Householder, $A \in \mathbb{C}^{m \times n}$ con $\text{rank}\,(A) = n$

```
for k = 1 : n
   x = A(k:m,k)
   v_k = sign(x_1)‖x‖_2 e_1 + x
   v_k = v_k / ‖v_k‖_2
   A(k:m,k:n) = A(k:m,k:n) - 2 v_k v_k^* A(k:m,k:n)

```

- Al finalizar el algoritmo en A tenemos la matriz triangular $R = Q^* A$

Algoritmo de Householder

Algoritmo: Householder, $A \in \mathbb{C}^{m \times n}$ con $\text{rank}\,(A) = n$

```
for k = 1 : n
   x = A(k:m,k)
   v_k = sign(x_1)‖x‖_2 e_1 + x
   v_k = v_k / ‖v_k‖_2
   A(k:m,k:n) = A(k:m,k:n) - 2 v_k v_k^* A(k:m,k:n)

```

- Al finalizar el algoritmo en A tenemos la matriz triangular $R = Q^* A$
- Si queremos resolver $Ax = b$, necesitamos $Q^* b$, que se puede obtener así:

Algoritmo de Householder

Algoritmo: Householder, $A \in \mathbb{C}^{m \times n}$ con $\text{rank}(A) = n$

```
for k = 1 : n
   x = A(k:m,k)
   v_k = sign(x_1)||x||_2 e_1 + x
   v_k = v_k / ||v_k||_2
   A(k:m,k:n) = A(k:m,k:n) - 2 v_k v_k^* A(k:m,k:n)

```

- Al finalizar el algoritmo en A tenemos la matriz triangular $R = Q^*A$
- Si queremos resolver $Ax = b$, necesitamos Q^*b, que se puede obtener así:

```
for k = 1 : n
   b(k:m) = b(k:m) - 2 v_k (v_k^* b(k:m))

```