

Tema 4.

Cálculo de valores propios: Caso no simétrico

Cálculo de valores propios: Caso no simétrico

El algoritmo iterativo QR: versiones secuencial y paralelas

Lecturas recomendadas:

- G.Golub, C. Van Loan. “*Matrix Computations*”. Capítulo 7
- G.Henry, R. Van der Geijn. “*Parallelizing the QR Algorithm for the Unsymmetric Algebraic Eigenvalue Problem: Myths and Reality*”. SIAM J. On Scientific Computing, Vol. 14, No. 4, pp.870-88, (1996)
- Z.Bai and J.Demmel. “*On a Block Implementation of Hessenberg Multishift QR Iteration*”. Int. J. of High Speed Computing. Vol.1 (1989)
- A.Dubrulle and G.Golub. “*A Multishift QR Iteration without Computation of the Shifts*”. Numerical Algorithms. Vol. 7 (1994)

Idea básica del *Algoritmo Iterativo QR*

Sea $A \in \mathbb{C}^{n \times n}$

Consideremos el siguiente algoritmo:


$$A_1 = A$$

Para $s = 1, 2, \dots$, hasta convergencia

Calcular Q_s , unitaria, y R_s , triangular superior tales que $A_s = Q_s R_s$ (Desc. QR de A_s)

$$A_{s+1} = R_s Q_s$$

Se puede probar que, en $O(n^3)$ x num. de iteraciones,

$$* \quad \lim_{s \rightarrow \infty} A_s \rightarrow$$


$$* \quad \text{Si } A \in \mathbb{R}^{n \times n} \quad \lim_{s \rightarrow \infty} A_s \rightarrow \begin{array}{l} \text{Forma} \\ \text{real de} \\ \text{Schur} \end{array}$$

Convergencia del Algoritmo iterativo QR

Sea $A \in \mathbb{C}^{n \times n}$ con $|\lambda|_1 > |\lambda|_2 > \dots > |\lambda|_n$

Sea $X = [x_1, x_2, \dots, x_n]$ invertible, con $Ax_i = \lambda_i x_i$

Consideremos el algoritmo

$$A_1 := A$$

Para $s = 1, 2, \dots, \infty$

$$A_s = Q_s R_s$$

$$A_{s+1} := R_s Q_s = Q_s^H A_s Q_s$$

Se tiene que:

$$A = XDX^{-1} \text{ con } D = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_n)$$

Construimos:

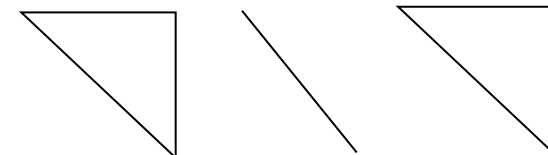
$$P_s := Q_1 Q_2 \dots Q_s ; \quad U_s := R_s R_{s-1} \dots R_1$$

Entonces

$$A_{s+1} = P_s^H A P_s = P_s^H XDX^{-1} P_s = P_s^H Q_X R_X D R_X^{-1} Q_X^H P_s$$

y en el límite $\lim_{s \rightarrow \infty} P_s = Q_X$ y por tanto:

$$\lim_{s \rightarrow \infty} A_{s+1} = Q_X^H Q_X R_X D R_X^{-1} Q_X^H Q_X = R_X D R_X^{-1}$$



En efecto, se tiene

$$A_{s+1} = Q_s^H A_s Q_s = Q_s^H Q_{s-1}^H A_{s-1} Q_{s-1} Q_s = \dots = P_s^H A_1 P_s \Rightarrow P_s A_{s+1} = A_1 P_s$$

Y también

$$P_s U_s = Q_1 Q_2 \dots Q_s R_s R_{s-1} \dots R_1 = Q_1 Q_2 \dots Q_{s-1} A_s R_{s-1} \dots R_1 = P_{s-1} A_s U_{s-1}$$

Por tanto

$$P_s U_s = A_1 P_{s-1} U_{s-1} = A_1^2 P_{s-2} U_{s-2} = \dots = A_1^{s-1} P_1 U_1 = A_1^s (\text{Desc. QR})$$

Además

$$P_s U_s = A_1^s = (XDX^{-1})(XDX^{-1})\dots(XDX^{-1}) = XD^s X^{-1}$$

Tomando límites

$$\lim_{s \rightarrow \infty} P_s U_s = \lim_{s \rightarrow \infty} (XD^s X^{-1}) = \lim_{s \rightarrow \infty} (Q_X R_X D^s L_{X^{-1}} U_{X^{-1}}) = \lim_{s \rightarrow \infty} (Q_X R_X D^s L_{X^{-1}} D^{-s} D^s U_{X^{-1}})$$

$$\text{donde } X = Q_X R_X \text{ (Desc. QR de } X), \text{ y } X^{-1} = L_{X^{-1}} U_{X^{-1}} \text{ (Desc. LU de } X^{-1})$$

Puesto que $\lim_{s \rightarrow \infty} (D^s L_{X^{-1}} D^{-s}) = I$

$$\lim_{s \rightarrow \infty} P_s U_s = \lim_{s \rightarrow \infty} (Q_X R_X D^s U_{X^{-1}}) \quad \Rightarrow \quad \lim_{s \rightarrow \infty} P_s = Q_X$$

Formas de mejorar las prestaciones del Algoritmo Iterativo QR

1. Disminuir el coste por iteración:

Reducción unitaria a la forma de Hessenberg superior

$$\begin{bmatrix} x & x & x & x \\ x & x & x & x \\ x & x & x & x \\ x & x & x & x \end{bmatrix} \quad \begin{array}{l} \text{Coste de la QR:} \\ O(n^3) \end{array}$$

$$\begin{bmatrix} x & x & x & x \\ x & x & x & x \\ 0 & x & x & x \\ 0 & 0 & x & x \end{bmatrix} \quad \begin{array}{l} \text{Coste de la QR:} \\ O(n^2) \end{array}$$

2. Aceleración de la convergencia:

Utilización de desplazamientos

$$\lim_{s \rightarrow \infty} (D^s L_{X^{-1}} D^{-s}) = I$$

Algoritmo Iterativo QR con Desplazamiento

Sea $A \in \mathbb{C}^{n \times n}$

$H_1 = P_0^H A P_0$ (* H_1 en forma de Hessenberg Superior*)

Para $s = 1, 2, \dots$, hasta Convergencia

Elegir (κ_s)

Calcular Q_s, R_s / $H_s - \kappa_s I = Q_s R_s$ (*Desc. QR*)

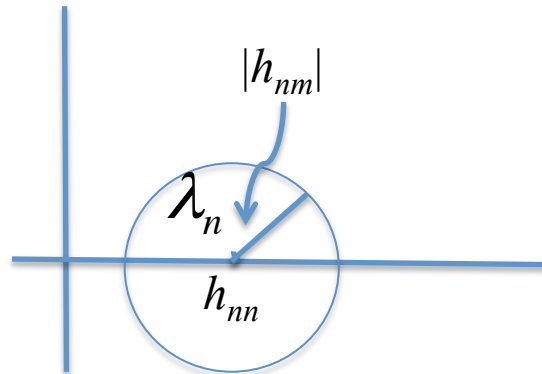
$H_{s+1} = R_s Q_s + \kappa_s I$

finpara

Elección del desplazamiento simple

$m = n - 1$

$$\begin{bmatrix} x & x & x & x \\ x & x & x & x \\ 0 & x & h_{mm} & h_{mn} \\ 0 & 0 & h_{nm} & h_{nn} \end{bmatrix}$$



Tratar de aproximar : $\kappa_s \approx \lambda_n$

Elegir : $\kappa_s \approx h_{nn}$

Desplazamiento de Stewart

Algoritmo Iterativo QR con Doble Desplazamiento

Si $A \in \mathbb{R}^{n \times n}$ tiene valores propios complejos conjugados. $\lambda_m, \overline{\lambda_m}$

$$\begin{bmatrix} x & x & x & x \\ x & x & x & x \\ 0 & \boxed{x} & h_{mm} & h_{mn} \\ 0 & 0 & h_{nm} & h_{nn} \end{bmatrix}$$

Elegir $\kappa_s \approx h_{nn}$ no garantiza que

$$m=n-1$$

$$s \rightarrow \infty \Rightarrow h_{nm} \rightarrow 0$$

$$\text{ya que podr\u00eda ocurrir } s \rightarrow \infty \Rightarrow h_{n-1,m-1} \rightarrow 0$$

¡¡Podr\u00eda converger a la forma Real de Schur!!

¡¡No hay garant\u00eda de que $s \rightarrow \infty \Rightarrow h_{nn} \rightarrow \lambda_n$!!

En cambio si que hay garant\u00eda de que

$$s \rightarrow \infty \Rightarrow \lambda \left(\begin{bmatrix} h_{mm} & h_{mn} \\ h_{nm} & h_{nn} \end{bmatrix} \right) \rightarrow \{\lambda_m, \overline{\lambda_m}\}$$

Posible elecci\u00f3n del desplazamiento

$Elegir(\kappa_s)$

$$S = h_{mm} + h_{nn}$$

$$d = h_{mm} * h_{nn} - h_{mn} * h_{nm}$$

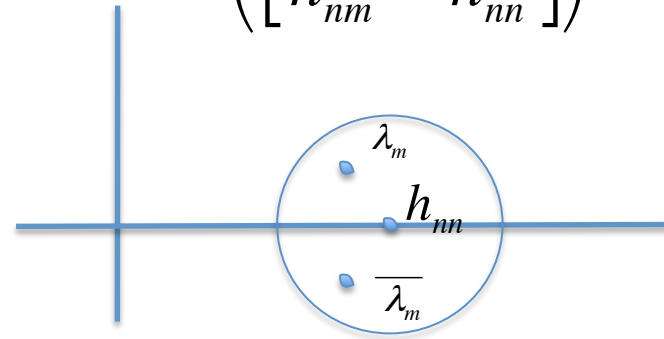
$$\text{Resolver } a^2 - Sa + d = 0$$

$$\text{Si } |a - h_{nn}| < |\overline{a} - h_{nn}|$$

$$\text{entonces } \kappa_s = a$$

$$\text{en otro caso } \kappa_s = \overline{a}$$

finsi



INCONVENIENTE : Si $A \in \mathbb{R}^{n \times n}$ se introduce aritm\u00e9tica compleja

SOLUCI\u00d3N : Desplazamientos consecutivos con a y \overline{a}

**Algoritmo Iterativo QR con Doble Desplazamiento Implícito
(caso real)**

Construimos

$$H_1 - aI = Q_1 R_1$$

$$H_2 = R_1 Q_1 + aI \rightarrow H_2 = Q_1^H H_1 Q_1$$

$$H_2 - \bar{a}I = Q_2 R_2$$

$$H_3 = R_2 Q_2 + \bar{a}I \rightarrow H_3 = Q_2^H H_2 Q_2 = Q_2^H Q_1^H H_1 Q_1 Q_2$$

$$\begin{aligned} Q_1 Q_2 R_2 R_1 &= Q_1 (H_2 - \bar{a}I) R_1 = Q_1 (Q_1^H H_1 Q_1 - \bar{a}I) R_1 = \\ &= H_1 Q_1 R_1 - \bar{a} Q_1 R_1 = (H_1 - \bar{a}I) Q_1 R_1 = (H_1 - \bar{a}I)(H_1 - aI) \end{aligned}$$

Si llamamos

$$H \equiv (H_1 - \bar{a}I)(H_1 - aI) = H_1^2 - 2\operatorname{Re}(a)H_1 + |a|^2 I \in \Re^{n \times n}$$

$$Q \equiv Q_1 Q_2 \quad R \equiv R_2 R_1$$

Se tiene

$$H = QR \quad (*\text{Descomposición QR de } H*)$$

$$H_3 = Q^T H_1 Q$$

Algoritmo Iterativo QR con Doble Desplazamiento Implícito y Aritmética Real

$$H_1 = P_0^T A P_0$$

Para $i = 1, 2, \dots$, hasta convergencia

$$m = n - 1;$$

$$S = h_{nn} + h_{mm};$$

$$d = h_{nn} * h_{mm} - h_{nm} * h_{nm};$$

$$H = H_i^2 - S * H_i + dI;$$

Calcular Q_i, R_i / $H = Q_i R_i$ (Desc. QR de H^*)*

$$H_{i+1} = Q_i^T H_i Q_i$$

finpara

Inconveniente: El coste de calcular H y su QR es $O(n^3)$

Teorema de la Q implícita

Sean $A, B, Q \in \mathbb{R}^{n \times n}$ con B Hessenberg superior irreducible y Q ortogonal

Si $B = Q^T A Q$, con B y Q están unívocamente determinadas por la primera columna de Q

(* $B = Q^T A Q$, $B' = Q'^T A Q'$, y $Q e_1 = Q' e_1 \Rightarrow B = B'$, $Q = Q'$ *)

Demostración

$$B = Q^T A Q \Rightarrow QB = AQ$$

Si $Q = [q_1, q_2, \dots, q_n]$ y $b_{ij} = 0, i > j + 1$, igualando la columna k de los dos miembros de $QB = AQ$, se tiene:

$$b_{1k}q_1 + b_{2k}q_2 + \dots + b_{kk}q_k + b_{k+1,k}q_{k+1} = Aq_k$$

Puesto que Q es ortogonal $\Rightarrow q_i^T q_j = \delta_{ij}$, si se premultiplican ambos miembros por q_i^T se tiene

$$b_{ik} = q_i^T A q_k, i = 1, 2, \dots, k$$

$$b_{k+1,k} \neq 0 \Rightarrow q_{k+1} = \frac{\left(Aq_k - \sum_{i=1}^k b_{ik}q_i \right)}{b_{k+1,k}}$$

$$q_{k+1}^T q_{k+1} = 1 \Rightarrow b_{k+1,k} = \left\| Aq_k - \sum_{i=1}^k b_{ik}q_i \right\|_2$$

Es decir, conocido q_1 quedan determinados q_2 y b_1

Por tanto, conocido q_1, q_2, \dots, q_k quedan determinados q_{k+1} y b_k

Como evitar el cálculo de H y la QR en O(n³)

Operaciones anteriores :

$$H = H_i^2 - 2\text{Re}(a)H_i + |a|^2 I \in \mathbb{R}^{n \times n}$$

Calcula QR de H

$$H_{i+2} = Q^T H_i Q$$

O(n³)

Operaciones propuestas :

Calcula P₁ de Householder / Qe₁ = P₁e₁

Calcular P₁^T H_i P₁

Devolver a P₁^T H_i P₁ la forma de Hessenberg superior :

$$H_{i+2} = (P_{n-1}^T \dots P_3^T P_2^T)(P_1^T H_i P_1)(P_2 P_3 \dots P_{n-1})$$

O(n²)

Operaciones anteriores :

$$H = \begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \\ & x & x & x & x \\ & & x & x & x \\ & & & x & x \end{bmatrix} \begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \\ & x & x & x & x \\ & & x & x & x \\ & & & x & x \end{bmatrix} - S \begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \\ & x & x & x & x \\ & & x & x & x \\ & & & x & x \end{bmatrix} + \begin{bmatrix} d & & & & \\ & d & & & \\ & & d & & \\ & & & d & \\ & & & & d \end{bmatrix} = \begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \\ & x & x & x & x \\ & & x & x & x \end{bmatrix}$$

Como se calcularía la Desc. QR de H

$$Q_4^T Q_3^T Q_2^T Q_1^T H = \begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \\ & x & x & x & x \\ & & x & x & x \end{bmatrix} = Q_4^T Q_3^T Q_2^T \begin{bmatrix} x & x & x & x & x \\ & x & x & x & x \\ & x & x & x & x \\ & & x & x & x \\ & & & x & x \end{bmatrix} \text{ con } Q_1^T = \begin{bmatrix} q & q & q \\ q & q & q \\ q & q & q \\ & & & 1 \\ & & & & 1 \end{bmatrix} \text{ y } Q_1^T (H e_1) = Q_1^T \begin{bmatrix} \alpha \\ \beta \\ \gamma \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} w \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \text{ y } Q_1 e_1 = (Q_1 Q_2 Q_3 Q_4) e_1$$

Además, α , β y γ se pueden calcular en 8 flops :

$$\alpha = (H_i)_{11}^2 + (H_i)_{21}^* (H_i)_{12} - S^* (H_i)_{11} + d ; \beta = (H_i)_{21}^* (H_i)_{11} + (H_i)_{22}^* (H_i)_{21} - S^* (H_i)_{21} ; \gamma = (H_i)_{32}^* (H_i)_{21}$$

Como evitar el cálculo de H y la QR en $O(n^3)$

$$1. P_1 = Q_1 \text{ tal que } Q_1^T \begin{bmatrix} \alpha \\ \beta \\ \gamma \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} w \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Operaciones propuestas :

1. Calcular P_1 de Householder / $Qe_1 = P_1e_1$

2. Calcular $P_1^T H_i P_1$

3. Devolver a $P_1^T H_i P_1$ la forma de Hessenberg superior :

$$H_{i+2} = (P_{n-1}^T \dots P_3^T P_2^T)(P_1^T H_i P_1)(P_2 P_3 \dots P_{n-1})$$

$$2. P_1^T H_i P_1 = \begin{bmatrix} q & q & q & & \\ q & q & q & & \\ q & q & q & & \\ & & & 1 & \\ & & & & 1 \end{bmatrix} \begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \\ & x & x & x & x \\ & & x & x & x \\ & & & x & x \end{bmatrix} \begin{bmatrix} q & q & q & & \\ q & q & q & & \\ q & q & q & & \\ & & & 1 & \\ & & & & 1 \end{bmatrix} = \begin{bmatrix} y & y & y & y & y \\ y & y & y & y & y \\ y & y & y & y & y \\ y & y & y & x & x \\ & & & x & x \end{bmatrix}$$

$$3. (P_{n-1}^T \dots P_3^T P_2^T)(P_1^T H_i P_1)(P_2 P_3 \dots P_{n-1}) = \begin{bmatrix} y & y & y & y & y \\ y & y & y & y & y \\ y & y & y & y & y \\ y & y & y & x & x \\ & z & z & x & x \end{bmatrix}$$

Coste aproximado: $20n^2 \text{ flops}$

A 12x12 matrix of 'x' symbols. The matrix is lower triangular, with 'x' symbols filling the entire lower triangle and the diagonal. The upper triangle is empty. A series of blue rectangles are placed along the diagonal, starting from the top-left and moving towards the bottom-right. Each rectangle is positioned such that it covers the 'x' symbols on the diagonal and the 'x' symbols immediately below it. The rectangles are of varying heights, with the first rectangle on the diagonal being the tallest and the last one being the shortest. Red dots are placed at the bottom-right corner of each blue rectangle.

Algoritmo iterativo QR (paso básico de Francis)

$$H_1 = P_0^T A P_0 \quad (* P_0 \text{ ortogonal, } H_1 \text{ Hessenberg superior} *)$$

Para $i = 1, 2, \dots$, *hasta convergencia*

$$m = n - 1;$$

$$S = h_{nn} + h_{mm};$$

$$d = h_{nn} * h_{mm} - h_{nm} * h_{nm};$$

$$\text{Calcular } He_1 \quad (*\alpha, \beta, \gamma \text{ en } 8 \text{ flops}*)$$

$$\text{Calcular } \bar{P}_1 \in \Re^{3 \times 3} \quad / \quad \begin{bmatrix} \bar{P}_1^T & \\ & I \end{bmatrix} (He_1) = ke_1 \quad (* \text{Triangulariza } [\alpha \beta \gamma]^T *)$$

$$H_{i+1} = (P_{n-1}^T \dots P_3^T P_2^T) (P_1^T H_i P_1) (P_2 P_3 \dots P_{n-1})$$

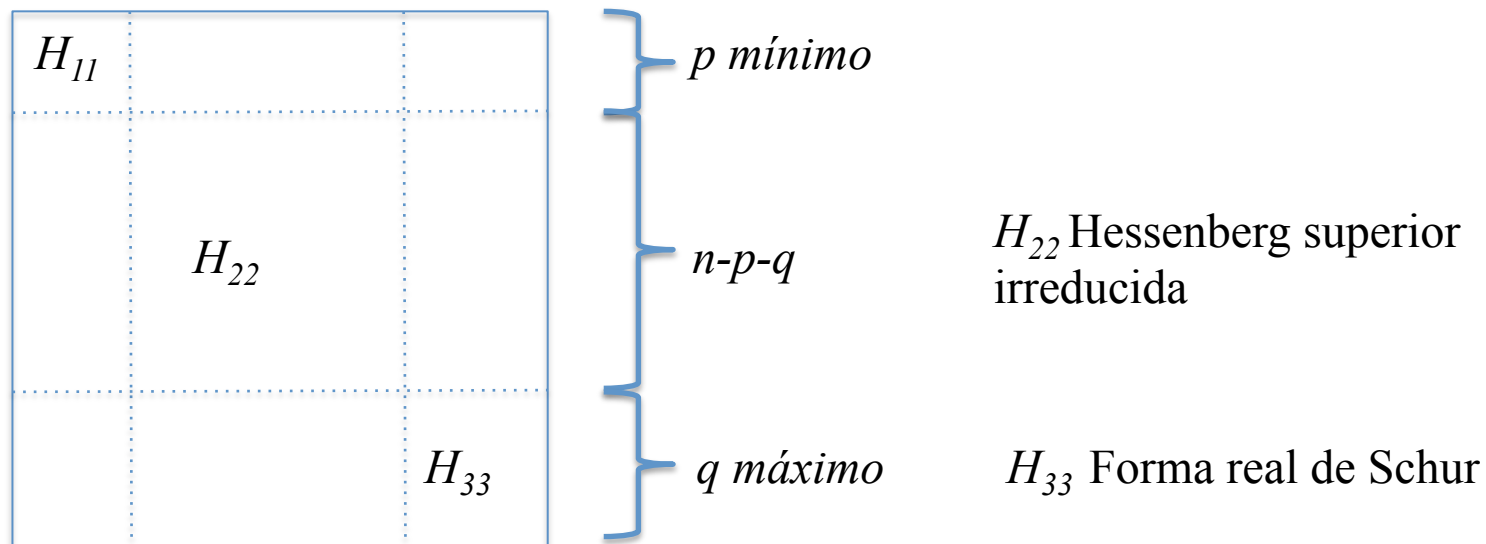
(*Reducción ortogonal de semejanza

de $(P_1^T H_i P_1)$ a la forma de Hessenberg superior*)

finpara

Coste aproximado: $20n^2$ flops por iteración

Algoritmo iterativo QR (completo con deflación)



Calcular p mínimo y q máximo para que

H_{22} sea Hessenberg superior irreducible

H_{33} esté en Forma Real de Schur

Mientras $q < n$

1. Aplicar un paso básico del Algoritmo Iterativo QR a H_{22}
2. Recalcular p y q

Paralelización del Algoritmo Iterativo QR en el modelo de Memoria Distribuida

[illegible]

Diagram illustrating a sparse matrix structure with two overlapping 5x5 subgrids (blue and red) and their intersection (3x3 subgrid). The intersection contains 'O' marks on the diagonal and 'X' marks elsewhere.

$$\begin{bmatrix} x & x & x & x & x & x & x & x \\ x & x & x & x & x & x & x & x \\ & x & x & x & x & x & x & x \\ & & x & x & x & x & x & x \\ & & O & x & x & x & x & x \\ & & O & \otimes & x & x & x & x \\ & & & \otimes & \otimes & x & x & x \\ & & & & & & x & x \end{bmatrix}$$

Por
filas

P_0

P_1

P_2

P_3

P_4

P_5

P_6

P_0

P_1

P_2

P_3

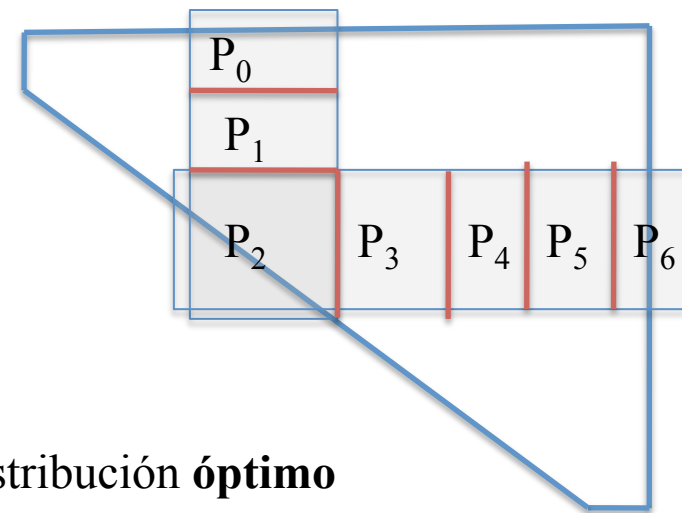
P_4

P_5

P_6

Por
columnas

Esquemas de distribución de datos



Esquema de distribución **óptimo**

Distribución por Antidiagonales (Hankel Wrapped)

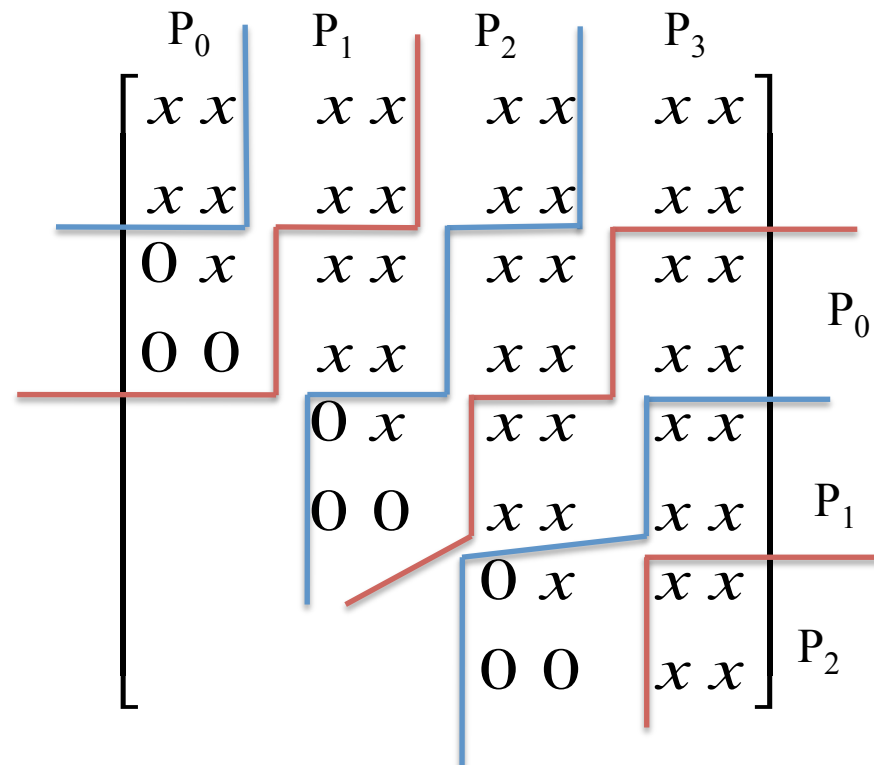
Particionado por bloques

$$\begin{bmatrix} A_{11}^{(0)} & A_{12}^{(1)} & A_{13}^{(2)} & \dots & A_{1r}^{(r-1)} \\ A_{21}^{(1)} & A_{22}^{(2)} & A_{23}^{(3)} & \dots & A_{2r}^{(r)} \\ & A_{32}^{(3)} & A_{33}^{(4)} & \dots & A_{3r}^{(r+1)} \\ & & \cdot & \cdot & \cdot \\ & & & \cdot & \cdot \\ & & & & A_{rr}^{(2r-2)} \end{bmatrix}$$

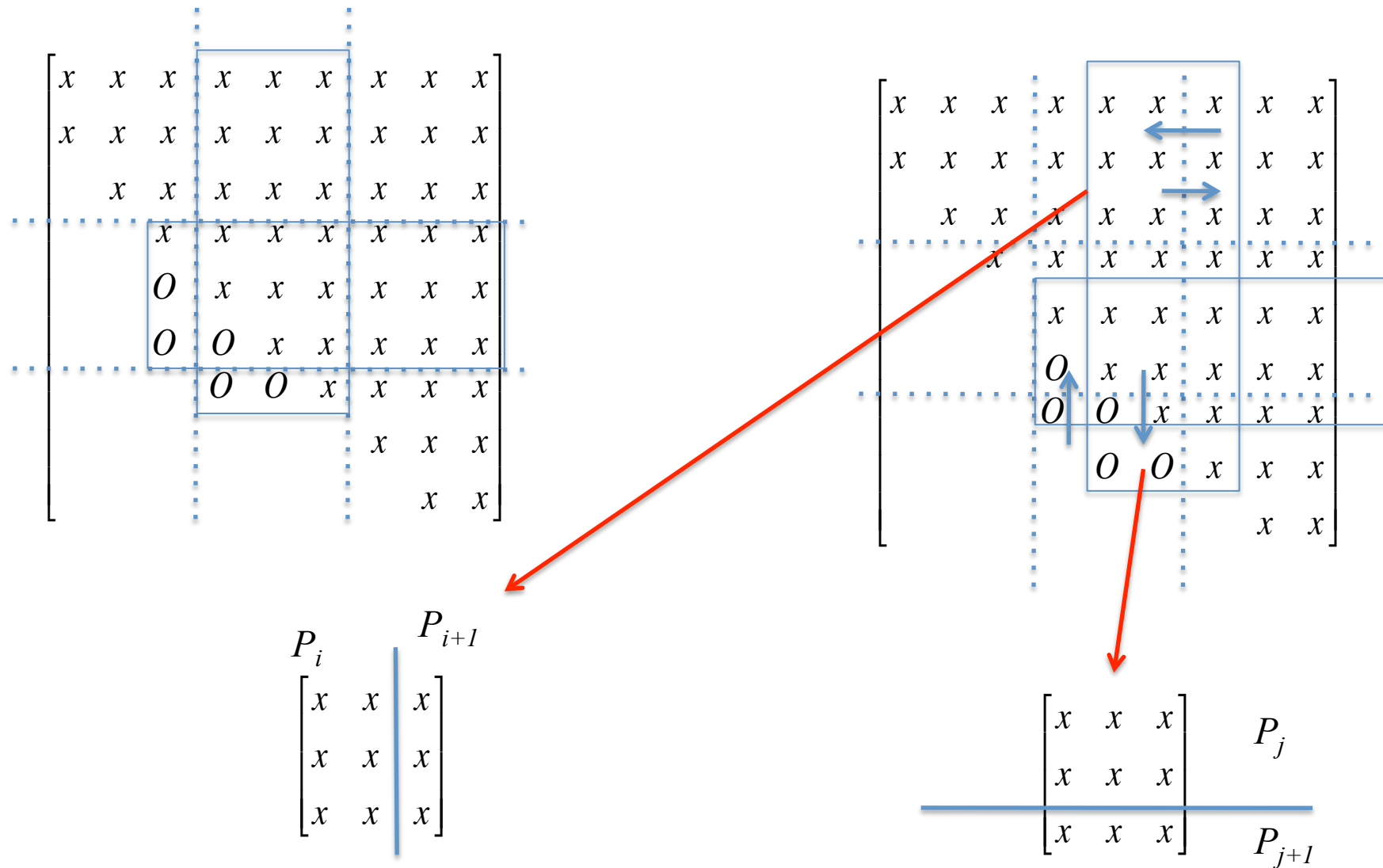
Ejemplo con $p=4$, $n=8$, $r=4$

Distribución en p procesadores por bloques de antidiagonales con $r=n/tb$

Bloque $A_{ij} \rightarrow \text{Procesador } (i + j - 2) \text{ MOD } p$



Distribución por Antidiagonales (Hankel Wrapped): Comunicaciones



*Si $A_{rr} \in \text{Pr}_i$
entonces
Calcula S y d
Envía S y d a Pr_0*

finsi

Si $i = 0$

entonces

Espera S y d

Calcula P_1

Difunde P_1

en otro caso

Espera P_1

finsi

Actualiza bloques correspondientes a P_1 que están en Pr_i

Para $j = 2, 3, \dots, n - 1$

Si $a_{j+1,j} \in \text{Pr}_i$

entonces

Calcula P_j (puede implicar comunicaciones *)*

Difunde P_j

en otro caso

Espera P_j

finsi

Actualiza bloques correspondientes a P_j (puede implicar comunicaciones *)*

finpara

**Paso del Algoritmo Iterativo QR paralelizado:
Código en el Procesador Pr_i**

Paso del Algoritmo Iterativo QR paralelizado:

Prestaciones (según artículo *Parallelizing the QR Algorithm for the Unsymmetric Algebraic Eigenvalue Problem: Myths and Reality*)

$$S_p = \frac{20n^2}{20\frac{n^2}{p} + k\frac{n^2}{rp} + O(n)} \approx \frac{p}{1 + O\left(\frac{p}{n}\right)} \quad E_p \approx \frac{1}{1 + O\left(\frac{p}{n}\right)}$$

Valores asintóticos

Si $p = \text{constante}$ AND $n \rightarrow \infty \Rightarrow E_p \rightarrow 1$

Escalabilidad

Si queremos que E_p sea constante $\forall p \Rightarrow (p \rightarrow \infty \Rightarrow n \rightarrow \infty)$

Por tanto la cantidad de memoria necesaria crecería con $O(n^2) = O(p^2)$

Algoritmo no escalable

Prestaciones experimentales en un Intel Paragon

p	n	Sp	Ep
4	2000	2.5	0.63
8	2800	5.2	0.65
16	4000	10.0	0.63
32	5600	17.1	0.53
64	8000	30.1	0.47
96	9600	37.4	0.39

Paralelización del Algoritmo Iterativo QR en el modelo de Memoria Compartida

Doble Desplazamiento Implícito

$$He_1 = (H_i - a_1 I)(H_i - a_2 I)e_1 \text{ con } a_1, a_2 \in \Re \text{ OR } a_1 = \overline{a_2}$$

$$P_1^T (He_1) = ke_1, \quad P_1 \in \Re^{3 \times 3}$$

Reducir $P_1^T (H_i) P_1$ a la forma de Hessenberg superior

Múltiple Desplazamiento Implícito

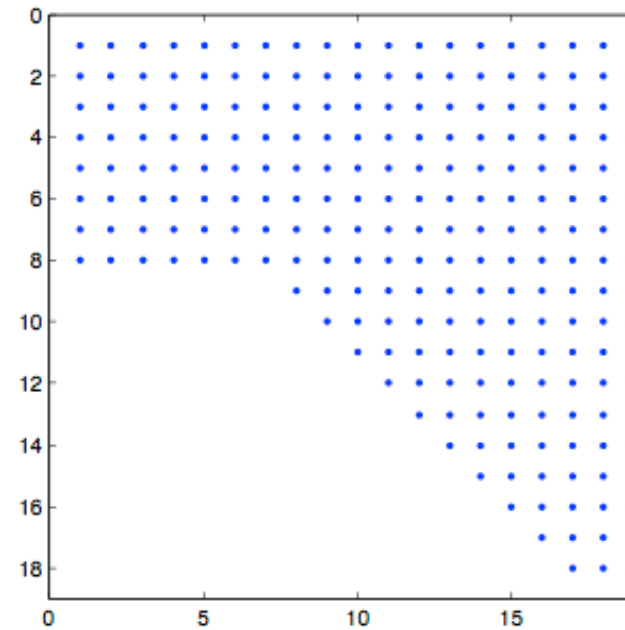
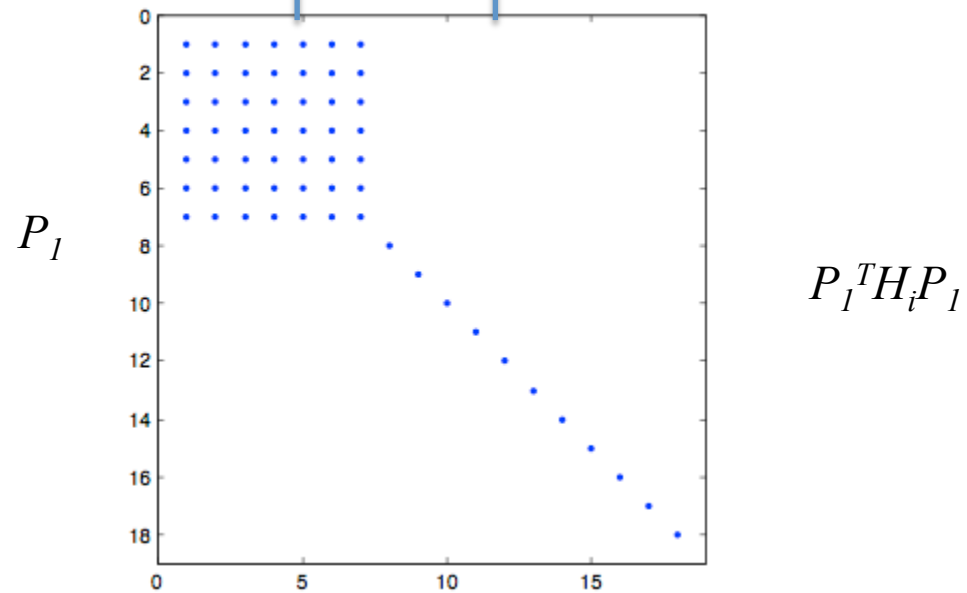
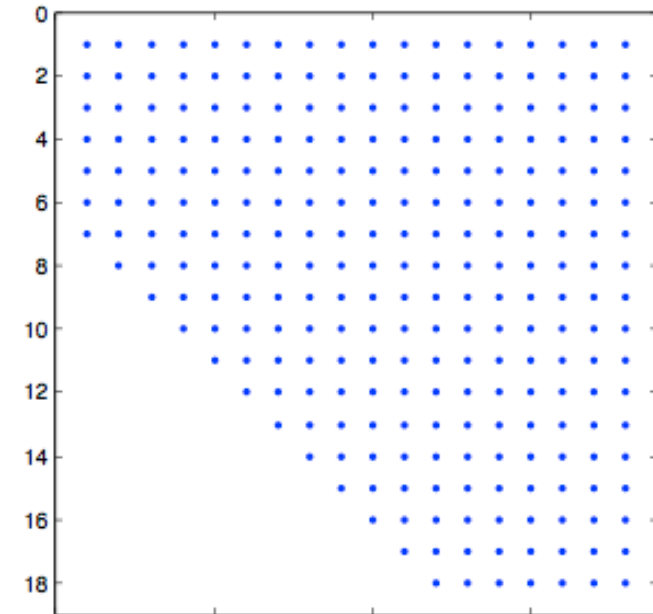
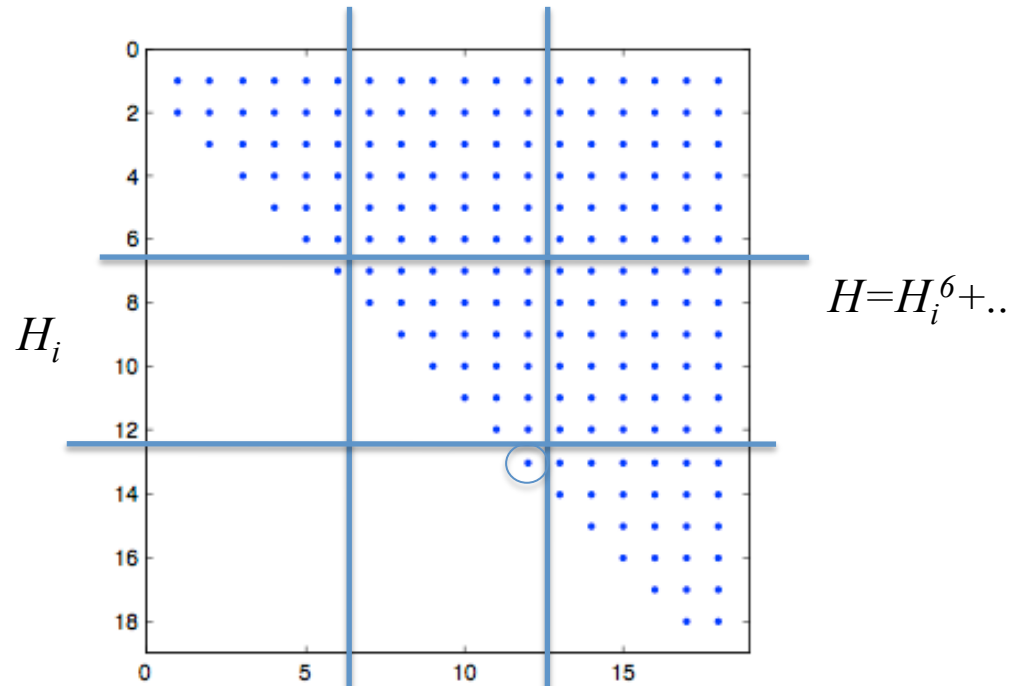
$$He_1 = (H_i - a_1 I)(H_i - a_2 I) \dots (H_i - a_m I)e_1 \text{ con } a_1, a_2, \dots, a_m \in \Re \text{ OR } a_i = \overline{a_{i+1}}$$

$$P_1^T (He_1) = ke_1, \quad P_1 \in \Re^{(m+1) \times (m+1)}$$

Reducir $P_1^T (H_i) P_1$ a la forma de Hessenberg superior

Por ejemplo, si $m=6$, se puede calcular He_1 en $O(1)$ utilizando únicamente 25 elementos.

Ejemplo de multidesplazamiento en un caso 6x6:



Paralelización del Algoritmo Iterativo QR en el modelo de Memoria Compartida

Algoritmo por bloques: Paso del Algoritmo

Particionado por bloques

$$H_i = \begin{bmatrix} A_{11} & A_{12} & A_{13} & \dots & A_{1r} \\ A_{21} & A_{22} & A_{23} & \dots & A_{2r} \\ & A_{32} & A_{33} & \dots & A_{3r} \\ & & \cdot & \cdot & \cdot \\ & & & \cdot & \cdot \\ & & & & A_{rr} \end{bmatrix}$$

1. *Calcula los Valores Propios de A_{rr} (* $a_1, a_2, \dots, a_{n/r}$ *)*
2. *Calcula $He_1 = \prod_{i=1}^{n/r} (H_i - a_i I) e_1$*
3. *Calcula P_1 tal que $P_1^T (He_1) = ke_1$*
4. *$H_i = P_1^T (H_i) P_1$*
5. *Obtener H_{i+1} reduciendo H_i a la forma de Hessenberg superior (* operacion por bloques *)*