QR Algorithm is a form of Power Method and Inverse PM

while NOT YET CONVERGED **do**
  Compute QR factorization $A^{(k)} = Q^{(k)} R^{(k)}$.
  Compute $A^{(k+1)} = R^{(k)} Q^{(k)}$, $\overline{Q}^{(k+1)} = \overline{Q}^{(k)} Q^{(k)}$
  $k = k + 1$
end while

$$
\begin{aligned}
A^k &= Q^{(0)} R^{(0)} Q^{(0)} R^{(0)} \cdots Q^{(0)} R^{(0)} \\
&= Q^{(0)} \left(A^{(1)} A^{(1)} \cdots A^{(1)} \right) R^{(0)} \\
&= Q^{(0)} \left(Q^{(1)} R^{(1)} \cdots Q^{(1)} R^{(1)} \right) R^{(0)} \\
&= Q^{(0)} Q^{(1)} \left(A^{(2)} \cdots A^{(2)} \right) R^{(1)} R^{(0)} \\
&= \cdots \\
&= \left(Q^{(0)} Q^{(1)} \cdots Q^{(k-1)} \right) \left(R^{(k-1)} R^{(k-2)} \cdots R^{(0)} \right) \\
&\overset{def}{=} \overline{Q}^{(k)} \overline{R}^{(k)},
\end{aligned}
$$

QR Algorithm is a form of Power Method and Inverse PM

while NOT YET CONVERGED **do**
  Compute QR factorization $A^{(k)} = Q^{(k)} R^{(k)}$.
  Compute $A^{(k+1)} = R^{(k)} Q^{(k)}$, $\overline{Q}^{(k+1)} = \overline{Q}^{(k)} Q^{(k)}$
  $k = k + 1$
end while

$$
\begin{aligned}
A^k &= Q^{(0)} R^{(0)} Q^{(0)} R^{(0)} \cdots Q^{(0)} R^{(0)} \\
&= Q^{(0)} \left(A^{(1)} A^{(1)} \cdots A^{(1)} \right) R^{(0)} \\
&= Q^{(0)} \left(Q^{(1)} R^{(1)} \cdots Q^{(1)} R^{(1)} \right) R^{(0)} \\
&= Q^{(0)} Q^{(1)} \left(A^{(2)} \cdots A^{(2)} \right) R^{(1)} R^{(0)} \\
&= \cdots \\
&= \left(Q^{(0)} Q^{(1)} \cdots Q^{(k-1)} \right) \left(R^{(k-1)} R^{(k-2)} \cdots R^{(0)} \right) \\
&\overset{def}{=} \overline{Q}^{(k)} \overline{R}^{(k)}, \quad \text{and} \quad \left(A^{-1} \right)^k = \overline{Q}^{(k)} \left(\overline{R}^{(k)} \right)^{-T}.
\end{aligned}
$$

Big eigenvalues converge to top, small eigenvalues bottom

while NOT YET CONVERGED **do**
  Compute QR factorization $A^{(k)} = Q^{(k)} R^{(k)}$.
  Compute $A^{(k+1)} = R^{(k)} Q^{(k)}$, $\overline{Q}^{(k+1)} = \overline{Q}^{(k)} Q^{(k)}$
  $k = k + 1$
end while

$$A^k = \overline{Q}^{(k)} \overline{R}^{(k)}, \quad \text{and} \quad \left(A^{-1}\right)^k = \overline{Q}^{(k)} \left(\overline{R}^{(k)}\right)^{-T}.$$

$$\text{Define} \quad \mathbf{e}_1 = \begin{pmatrix} 1 \\ \mathbf{0} \end{pmatrix}, \quad \mathbf{e}_n = \begin{pmatrix} \mathbf{0} \\ 1 \end{pmatrix},$$

$$\overline{Q}^{(k)} = \left(\overline{\mathbf{q}}_1^{(k)}, \cdots, \overline{\mathbf{q}}_n^{(k)}\right), \quad \overline{R}^{(k)} = \begin{pmatrix} \alpha^{(k)} & * & * \\ & * & * \\ & & \zeta^{(k)} \end{pmatrix}.$$

$$\text{Then} \quad A^k \mathbf{e}_1 = \alpha^{(k)} \overline{\mathbf{q}}_1^{(k)}, \quad \left(A^{-1}\right)^k \mathbf{e}_n = \left(\zeta^{(k)}\right)^{-1} \overline{\mathbf{q}}_n^{(k)}.$$

Thm: $A^{(k)} = \left(\overline{Q}^{(k)}\right)^T A\, \overline{Q}^{(k)}$

Proof

- $A^{(1)} = R^{(0)}\, Q^{(0)} = \left(Q^{(0)}\right)^T Q^{(0)}\, R^{(0)}\, Q^{(0)} = \left(\overline{Q}^{(1)}\right)^T A \overline{Q}^{(1)}$.
- By induction,

$$\begin{aligned} A^{(k)} &= R^{(k-1)}\, Q^{(k-1)} = \left(Q^{(k-1)}\right)^T Q^{(k-1)}\, R^{(k-1)}\, Q^{(k-1)} \\ &= \left(Q^{(k-1)}\right)^T A^{(k-1)}\, Q^{(k-1)} \\ &= \left(Q^{(k-1)}\right)^T \left(\overline{Q}^{(k-1)}\right)^T A\, \overline{Q}^{(k-1)}\, Q^{(k-1)} \\ &= \left(\overline{Q}^{(k)}\right)^T A \overline{Q}^{(k)}. \qquad \square \end{aligned}$$

If Algorithm converges, so does

$$\left(\overline{\mathbf{q}}_n^{(k)}\right)^T A\, \overline{\mathbf{q}}_n^{(k)} = \mathbf{e}_n^T\, A^{(k)}\, \mathbf{e}_n = A^{(k)}\,(n,n)\,.$$

Assume the eigenvalues of A are all distinct, and

$$|\lambda_1| > |\lambda_2| > \cdots > |\lambda_{n-1}| > |\lambda_n| .$$

- QR Algorithm converges in $A^{(k)}(n,n)$ to λ_n, at rate $\left| \frac{\lambda_n}{\lambda_{n-1}} \right|^k$.

Assume the eigenvalues of A are all distinct, and

$$|\lambda_1| > |\lambda_2| > \cdots > |\lambda_{n-1}| > |\lambda_n| .$$

- QR Algorithm converges in $A^{(k)}(n,n)$ to λ_n, at rate $\left| \dfrac{\lambda_n}{\lambda_{n-1}} \right|^k$.
- QR iteration with SHIFT σ:

$$A - \sigma I = Q R, \quad \text{and} \quad \widehat{A} = R Q + \sigma I \quad \left(= Q^T A Q. \right)$$

Assume the eigenvalues of A are all distinct, and

$$|\lambda_1| > |\lambda_2| > \cdots > |\lambda_{n-1}| > |\lambda_n| .$$

- QR Algorithm converges in $A^{(k)}(n,n)$ to λ_n, at rate $\left| \dfrac{\lambda_n}{\lambda_{n-1}} \right|^k$.
- QR iteration with SHIFT σ:

$$A - \sigma I = Q R, \quad \text{and} \quad \widehat{A} = R Q + \sigma I \quad \left(= Q^T A Q. \right)$$

Algorithm 3 QR Algorithm with shift σ

Initialize: $A^{(0)} = A$, $k = 0$, $\overline{Q}^{(0)} = I$.
while NOT YET CONVERGED **do**
  Compute QR factorization $A^{(k)} - \sigma I = Q^{(k)} R^{(k)}$.
  Compute $A^{(k+1)} = R^{(k)} Q^{(k)} + \sigma I$, $\overline{Q}^{(k+1)} = \overline{Q}^{(k)} Q^{(k)}$
  $k = k + 1$
end while

Assume the eigenvalues of A are all distinct, and

$$|\lambda_1| > |\lambda_2| > \cdots > |\lambda_{n-1}| > |\lambda_n| .$$

- QR Algorithm converges in $A^{(k)}(n,n)$ to λ_n, at rate $\left| \dfrac{\lambda_n}{\lambda_{n-1}} \right|^k$.
- QR iteration with SHIFT σ:

$$A - \sigma I = Q R, \quad \text{and} \quad \widehat{A} = R Q + \sigma I \quad \left(= Q^T A Q. \right)$$

Algorithm 4 QR Algorithm with shift σ

Initialize: $A^{(0)} = A$, $k = 0$, $\overline{Q}^{(0)} = I$.
while NOT YET CONVERGED **do**
  Compute QR factorization $A^{(k)} - \sigma I = Q^{(k)} R^{(k)}$.
  Compute $A^{(k+1)} = R^{(k)} Q^{(k)} + \sigma I$, $\overline{Q}^{(k+1)} = \overline{Q}^{(k)} Q^{(k)}$
  $k = k + 1$
end while

- Let λ_i and λ_k be eigenvalues closest and second closest to σ.
  Algorithm 4 converges in $A^{(k)}(n,n)$ to λ_i, at rate $\left| \dfrac{\lambda_i - \sigma}{\lambda_k - \sigma} \right|^k$.

- QR Algorithm converges in $A^{(k)}(n,n)$ to λ_n, at rate $\left|\frac{\lambda_n}{\lambda_{n-1}}\right|^k$.

- QR Algorithm converges in $A^{(k)}(n,n)$ to λ_n, at rate $\left|\frac{\lambda_n}{\lambda_{n-1}}\right|^k$.
- Let λ_i and λ_k be eigenvalues closest and second closest to σ. Algorithm 4 converges in $A^{(k)}(n,n)$ to λ_i, at rate $\left|\frac{\lambda_i - \sigma}{\lambda_k - \sigma}\right|^k$.

- QR Algorithm converges in $A^{(k)}(n,n)$ to λ_n, at rate $\left|\frac{\lambda_n}{\lambda_{n-1}}\right|^k$.
- Let λ_i and λ_k be eigenvalues closest and second closest to σ. Algorithm 4 converges in $A^{(k)}(n,n)$ to λ_i, at rate $\left|\frac{\lambda_i - \sigma}{\lambda_k - \sigma}\right|^k$.
- Let $E^{(k)} = \begin{pmatrix} \alpha_{n-1} & \beta_{n-1} \\ \beta_{n-1} & \alpha_n \end{pmatrix}$ be the bottom 2×2 submatrix of $A^{(k)}$, choose WILKINSON SHIFT

$$\sigma = \text{eigenvalue of } E^{(k)} \text{ closest to } \alpha_n = A^{(k)}(n,n).$$

- Wilkinson (1919 – 1986)

QR Algorithm

Algorithm 5 The QR Algorithm with Wilkinson shift

Initialize: $A^{(0)} = A$, $k = 0$, $\overline{Q}^{(0)} = I$.

while NOT YET CONVERGED **do**

$E^{(k)} = \begin{pmatrix} \alpha_{n-1} & \beta_{n-1} \\ \beta_{n-1} & \alpha_n \end{pmatrix} \stackrel{def}{=}$ bottom 2×2 submatrix of $A^{(k)}$.

Choose shift $\sigma =$ eigenvalue of $E^{(k)}$ closest to α_n.

Compute QR factorization $A^{(k)} - \sigma I = Q^{(k)} R^{(k)}$.

Compute $A^{(k+1)} = R^{(k)} Q^{(k)} + \sigma I$, $\overline{Q}^{(k+1)} = \overline{Q}^{(k)} Q^{(k)}$

$k = k + 1$

end while

QR Algorithm: pseudo-code, in four components

- Algorithm 1: Given a symmetric 2×2 matrix, find Wilkinson shift.
- Algorithm 2: Given a shift σ, perform a QR iteration.
- Algorithm 3: Compute next symmetric tridiagonal A.
- Algorithm 4: QR Algorithm to (recursively) compute all eigenvalues of symmetric tridiagonal matrix A.

Input matrix $E = \begin{pmatrix} \alpha_1 & \beta_1 \\ \beta_1 & \alpha_2 \end{pmatrix}$

Algorithm 1 Compute Wilkinson shift

$$\sigma = \begin{cases} \frac{\alpha_1+\alpha_2}{2} + \sqrt{\left(\frac{\alpha_1-\alpha_2}{2}\right)^2 + \beta_1^2}, & \text{if } \alpha_1 \leq \alpha_2, \\ \frac{\alpha_1+\alpha_2}{2} - \sqrt{\left(\frac{\alpha_1-\alpha_2}{2}\right)^2 + \beta_1^2}, & \text{if } \alpha_1 > \alpha_2. \end{cases}$$

Input matrix $A = \begin{pmatrix} \alpha_1 & \beta_1 & & & \\ \beta_1 & \alpha_2 & \beta_2 & & \\ & \beta_2 & \alpha_3 & \beta_3 & \\ & & \ddots & \ddots & \ddots \\ & & & \beta_{n-1} & \alpha_n \end{pmatrix}$, shift σ

- Compute $d_j = \alpha_j - \sigma,\ j = 1, \cdots, n$.
- Initialize $x_1 = d_1, y_1 = \beta_1$.
- QR Iteration stencil for $j = 1, \cdots, n-1$

$$\begin{pmatrix} c_{j+1} & s_{j+1} \\ -s_{j+1} & c_{j+1} \end{pmatrix} \begin{pmatrix} x_j & y_j & \\ \beta_j & d_{j+1} & \beta_{j+1} \end{pmatrix} = \begin{pmatrix} z_j & q_j & r_j \\ 0 & x_{j+1} & y_{j+1} \end{pmatrix}$$

Input matrix $A = \begin{pmatrix} \alpha_1 & \beta_1 & & & \\ \beta_1 & \alpha_2 & \beta_2 & & \\ & \beta_2 & \alpha_3 & \beta_3 & \\ & & \ddots & \ddots & \ddots \\ & & & \beta_{n-1} & \alpha_n \end{pmatrix}$, shift σ

Algorithm 2 QR Iteration

Set $d_j = \alpha_j - \sigma$, $j = 1, \cdots, n$. Initialize $x_1 = d_1, y_1 = \beta_1$.

for $j = 1, \cdots, n-1$ **do**

Compute $z_j = \sqrt{x_j^2 + \beta_j^2}, \begin{pmatrix} c_{j+1} \\ s_{j+1} \end{pmatrix} = \begin{pmatrix} x_j \\ \beta_j \end{pmatrix} / z_j,$

Compute $\begin{pmatrix} q_j \\ x_{j+1} \end{pmatrix} = \begin{pmatrix} c_{j+1} & s_{j+1} \\ -s_{j+1} & c_{j+1} \end{pmatrix} \begin{pmatrix} y_j \\ d_{j+1} \end{pmatrix},$

if $j < n-1$ **then**

$$\begin{pmatrix} r_j \\ y_{j+1} \end{pmatrix} = \begin{pmatrix} c_{j+1} & s_{j+1} \\ -s_{j+1} & c_{j+1} \end{pmatrix} \begin{pmatrix} 0 \\ \beta_j \end{pmatrix}.$$

end for

$z_n = x_n$.

- Algorithm 2 computes

$$G_n \cdots G_3\, G_2 A = R, \quad A = (G_n \cdots G_3\, G_2)^T\, R.$$

$$\text{where} \quad R = \begin{pmatrix} z_1 & q_1 & r_1 & & & & \\ & z_2 & q_2 & r_2 & & & \\ & & z_3 & q_3 & r_3 & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & z_{n-2} & q_{n-2} & r_{n-2} \\ & & & & & z_{n-1} & q_{n-1} \\ & & & & & & z_n \end{pmatrix}.$$

- $$\textsc{Next}\ A\ = R\,(G_n \cdots G_3\, G_2)^T = R\, G_2^T\, G_3^T \cdots G_n^T.$$

- $G_2^T = \begin{pmatrix} c_2 & -s_2 \\ s_2 & c_2 \end{pmatrix}$. In first two columns of $\textsc{Next}\ A$

$$\begin{pmatrix} z_1 & q_1 \\ 0 & z_2 \end{pmatrix} \begin{pmatrix} c_2 & -s_2 \\ s_2 & c_2 \end{pmatrix} = \begin{pmatrix} c_2\, z_1 + s_2\, q_1 & * \\ s_2\, z_2 & c_2\, z_2 \end{pmatrix} \stackrel{def}{=} \begin{pmatrix} \widehat{\alpha}_1 & * \\ \widehat{\beta}_1 & c_2\, z_2 \end{pmatrix}.$$

- Algorithm 2 computes

$$G_n \cdots G_3\, G_2 A = R, \quad A = (G_n \cdots G_3\, G_2)^T\, R.$$

$$\text{where} \quad R = \begin{pmatrix} z_1 & q_1 & r_1 & & & & \\ & z_2 & q_2 & r_2 & & & \\ & & z_3 & q_3 & r_3 & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & z_{n-2} & q_{n-2} & r_{n-2} \\ & & & & & z_{n-1} & q_{n-1} \\ & & & & & & z_n \end{pmatrix}.$$

-

$$\textsc{Next}\ A\ = R\,(G_n \cdots G_3\, G_2)^T = R\, G_2^T\, G_3^T \cdots G_n^T.$$

- $\textsc{Next}\ A$ stencil for $j = 2, \cdots, n-1$

$$\begin{pmatrix} c_j\, z_j & q_j \\ 0 & z_{j+1} \end{pmatrix} \begin{pmatrix} c_{j+1} & -s_{j+1} \\ s_{j+1} & c_{j+1} \end{pmatrix}$$

$$= \begin{pmatrix} c_{j+1}\, c_j\, z_j + s_{j+1}\, q_j & * \\ s_{j+1}\, z_{j+1} & c_{j+1}\, z_{j+1} \end{pmatrix} \stackrel{def}{=} \begin{pmatrix} \widehat{\alpha}_j & * \\ \widehat{\beta}_j & c_{j+1}\, z_{j+1} \end{pmatrix}$$

Compute Next A matrix

Algorithm 3 Compute Next A matrix

Compute $\alpha_1 = c_2\, z_1 + s_2\, q_1 + \sigma$

$\beta_1 = s_2\, z_1$.

for $j = 2, \cdots, n-1$ **do**

  Compute $\alpha_j = c_{j+1}\, c_j\, z_j + s_{j+1}\, q_j + \sigma$

  Compute $\beta_j = s_{j+1}\, z_j$

end for

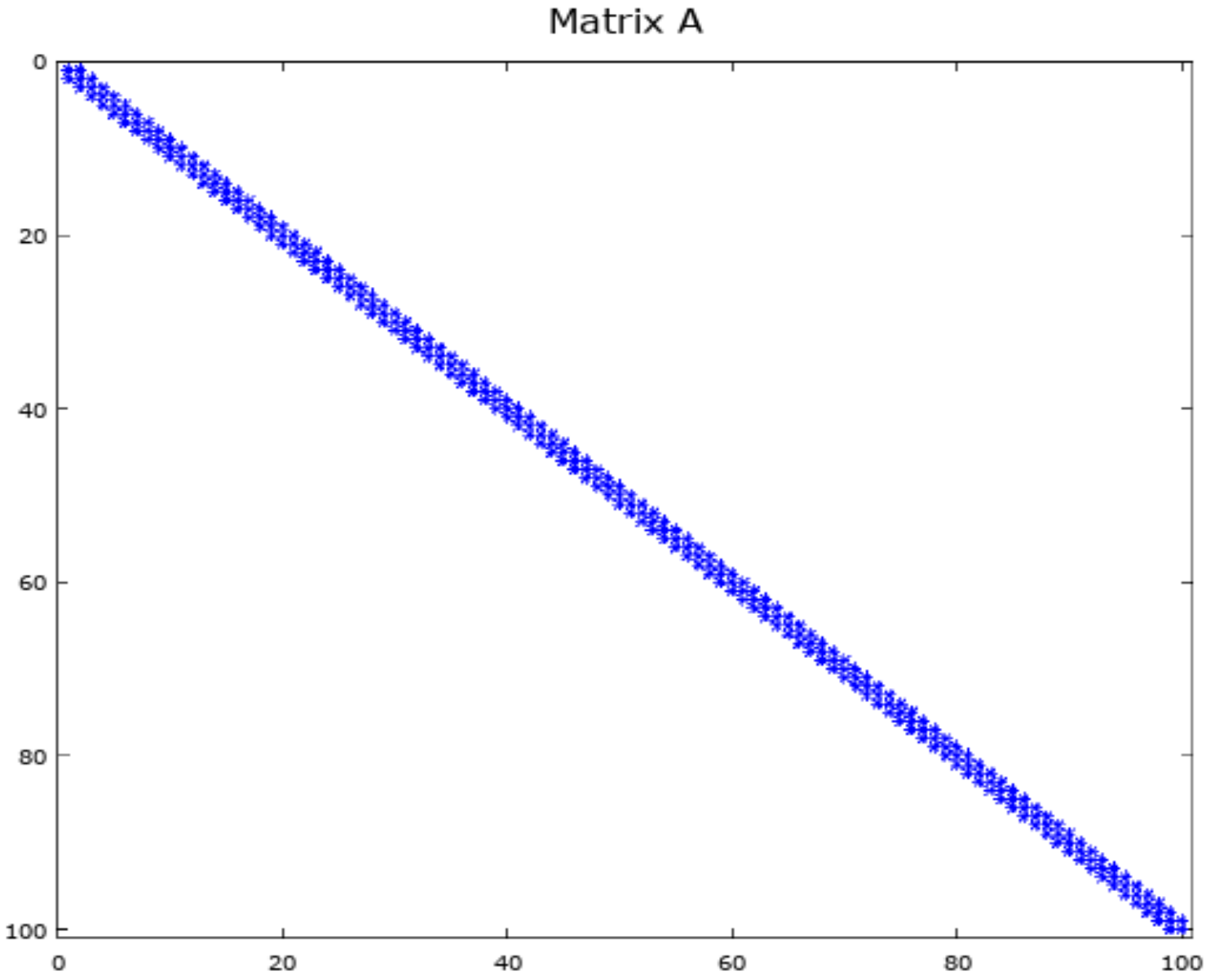
$\alpha_n = c_n\, z_n + \sigma$.

Input matrix $A = \begin{pmatrix} \alpha_1 & \beta_1 & & & \\ \beta_1 & \alpha_2 & \beta_2 & & \\ & \beta_2 & \alpha_3 & \beta_3 & \\ & & \ddots & \ddots & \ddots \\ & & & \beta_{n-1} & \alpha_n \end{pmatrix}$, tolerance τ

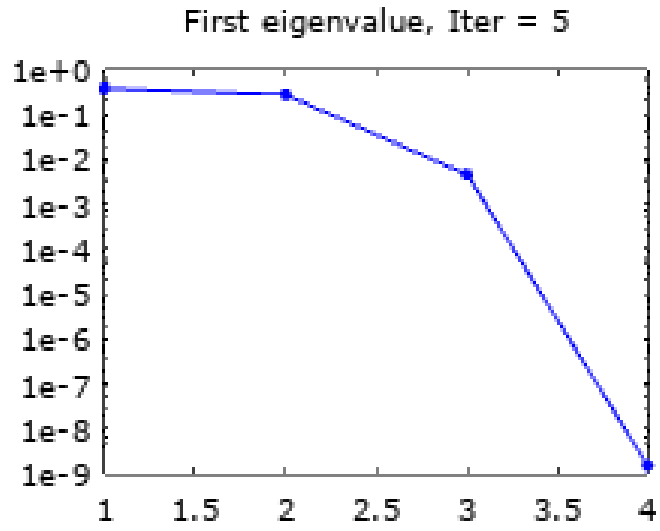
Algorithm 4 QR Algorithm with shift

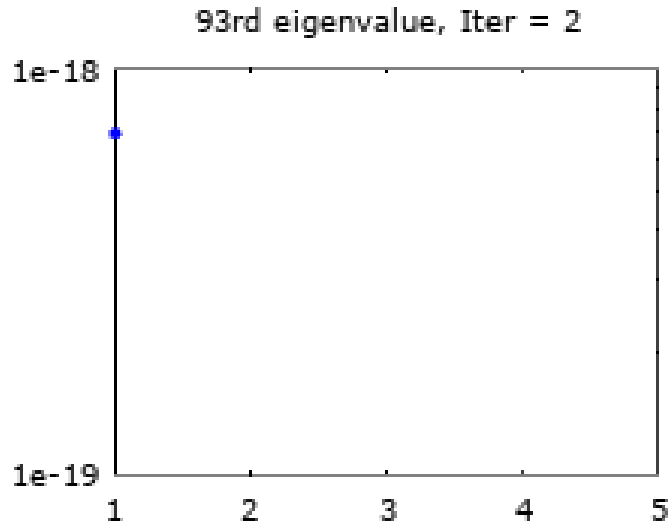
while NOT YET CONVERGED **do**
  if $n = 1$ **then** return eigenvalue $\lambda_1 = \alpha_1$
  for $j = 1, \cdots, n-1$ **do**
    if $|\beta_j| \leq \tau$ **then**
      Apply Algorithm 4 to $A(1:j, 1:j)$ and $A(j+1:n, j+1:n)$, respectively, for eigenvalues, return all eigenvalues
    end if
  end for
  Using Algorithm 1 to compute shift σ on matrix $\begin{pmatrix} \alpha_{n-1} & \beta_{n-1} \\ \beta_{n-1} & \alpha_n \end{pmatrix}$.
  Using Algorithm 2 to compute QR Iteration.
  Using Algorithm 3 to compute next A.
end while

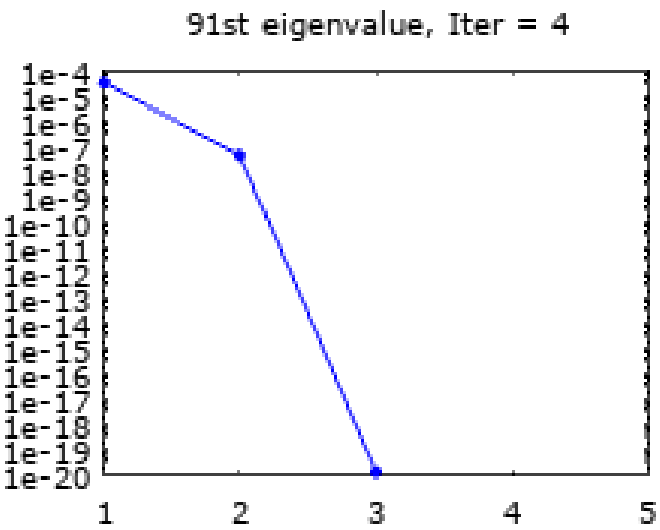
Ex: $A \in \mathbb{R}^{100\times 100}$ random, symmetric, tridiagonal

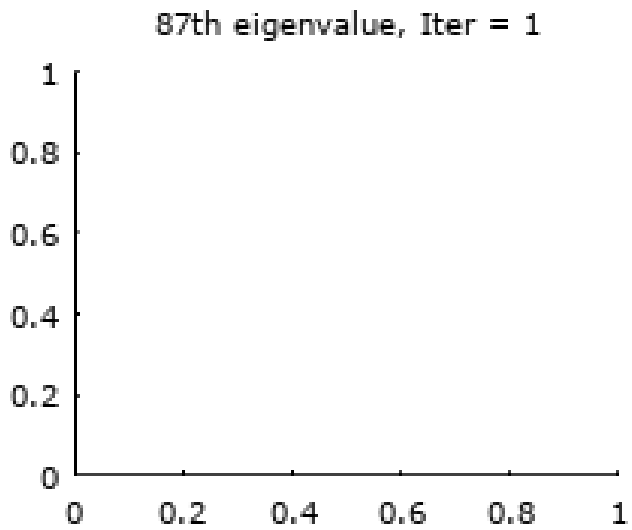


Ex: $A \in \mathbb{R}^{100\times 100}$ random, symmetric, tridiagonal

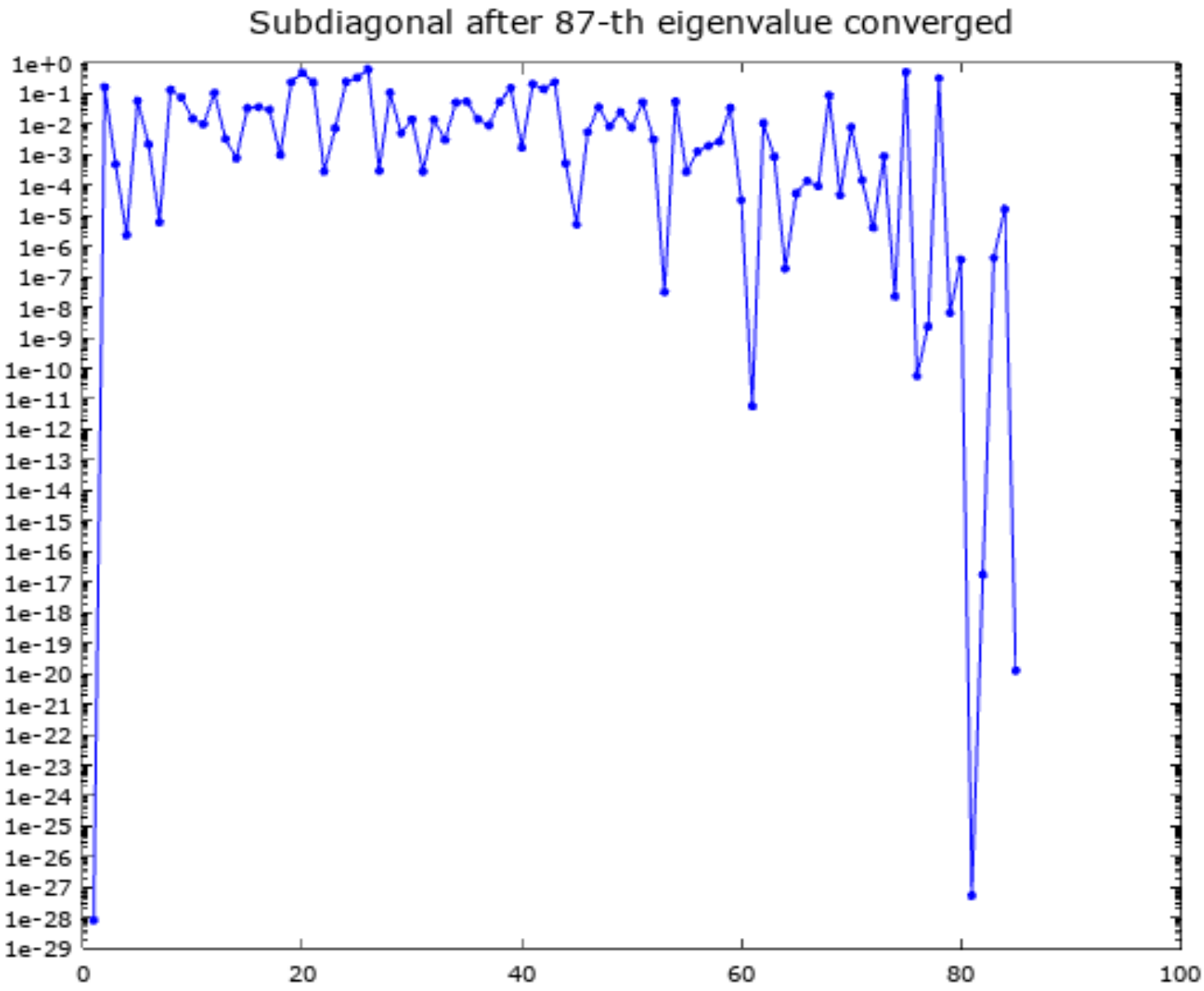








Ex: $A \in \mathbb{R}^{100\times 100}$ random, symmetric, tridiagonal



QR Algorithm for non-symmetric, upper Hessenberg matrix

Given: *Non-symmetric, upper Hessenberg matrix*

$$A = \begin{pmatrix} x & x & x & \cdots & x \\ x & x & x & \cdots & x \\ & x & x & \cdots & x \\ & & \ddots & \ddots & \vdots \\ & & & x & x \end{pmatrix} \in \mathbb{R}^{n \times n}.$$

Want: *All eigenvalues of A.*

Method: *The non-symmetric QR Algorithm,*
Repeatedly compute QR FACTORIZATION

$$A = Q\,R = Q \begin{pmatrix} x & x & x & x & \cdots & x \\ & x & x & x & \cdots & x \\ & & \ddots & \ddots & \ddots & \vdots \\ & & & x & x & x \\ & & & & x & x \\ & & & & & x \end{pmatrix}.$$

$O(n^2)$ Algorithm to eliminate sub-diagonal entries (I)

$$\text{Notation:} \quad G_j = \begin{pmatrix} I_{j-1} & & \\ & \begin{pmatrix} c_j & s_j \\ -s_j & c_j \end{pmatrix} & \\ & & I_{n-j} \end{pmatrix}, \quad j = 1, \cdots, n.$$

Eliminate first sub-diagonal entry with G_1

$$G_1 \begin{pmatrix} x & x & x & \cdots & x & x \\ x & x & x & \cdots & x & x \\ & x & x & \cdots & x & x \\ & & x & \cdots & x & x \\ & & & \ddots & \vdots & \vdots \\ & & & & x & x \end{pmatrix} = \begin{pmatrix} \overline{x} & \overline{x} & \overline{x} & \cdots & \overline{x} & \overline{x} \\ 0 & \overline{x} & \overline{x} & \cdots & \overline{x} & \overline{x} \\ & x & x & \cdots & x & x \\ & & x & \cdots & x & x \\ & & & \ddots & \vdots & \vdots \\ & & & & x & x \end{pmatrix},$$

$O(n^2)$ Algorithm to eliminate sub-diagonal entries (II)

Notation: $G_j = \begin{pmatrix} I_{j-1} & & \\ & \begin{pmatrix} c_j & s_j \\ -s_j & c_j \end{pmatrix} & \\ & & I_{n-j} \end{pmatrix}, \quad j = 1, \cdots, n.$

Eliminate second sub-diagonal entry with G_2

$$G_2 \begin{pmatrix} \overline{x} & \overline{x} & \overline{x} & \cdots & \overline{x} & \overline{x} \\ 0 & \overline{x} & \overline{x} & \cdots & \overline{x} & \overline{x} \\ & x & x & \cdots & x & x \\ & & x & \cdots & x & x \\ & & & \ddots & \vdots & \vdots \\ & & & & x & x \end{pmatrix} = \begin{pmatrix} \overline{x} & \overline{x} & \overline{x} & \cdots & \overline{x} & \overline{x} \\ 0 & \widehat{x} & \widehat{x} & \cdots & \widehat{x} & \widehat{x} \\ & 0 & \widehat{x} & \cdots & \widehat{x} & \widehat{x} \\ & & x & \cdots & x & x \\ & & & \ddots & \vdots & \vdots \\ & & & & x & x \end{pmatrix},$$

$O(n^2)$ Algorithm to eliminate sub-diagonal entries (III)

Eliminate all sub-diagonal entries with $G_{n-1} \cdots G_2 G_1$

$$G_{n-1} \cdots G_2 G_1 \begin{pmatrix} x & x & x & \cdots & x & x \\ x & x & x & \cdots & x & x \\ & x & x & \cdots & x & x \\ & & x & \cdots & x & x \\ & & & \ddots & \vdots & \vdots \\ & & & & x & x \end{pmatrix} = \begin{pmatrix} \overline{x} & \overline{x} & \overline{x} & \cdots & \overline{x} & \overline{x} \\ 0 & \widehat{x} & \widehat{x} & \cdots & \widehat{x} & \widehat{x} \\ & 0 & \breve{x} & \cdots & \breve{x} & \breve{x} \\ & & 0 & \cdots & \dot{x} & \dot{x} \\ & & & \ddots & \vdots & \vdots \\ & & & & 0 & \underline{x} \end{pmatrix}$$

$$\stackrel{def}{=} R.$$

$O(n^2)$ cost for QR factorization with Givens rotation (II)

$$A \;=\; (G_{n-1} \cdots G_2\, G_1)^T\; R \stackrel{def}{=} Q\, R.$$

Now define $\widehat{A} = R\, Q$.

- $\widehat{A} = Q^T\, A\, Q$. $\widehat{A}$ and A have the same eigenvalues.
- Assume A non-singular. $Q = A\, R^{-1}$ upper Hessenberg.
- $\widehat{A} = R\, Q$ must be upper Hessenberg as well.

QR Algorithm

Algorithm 1 The QR Algorithm

Initialize: $A^{(0)} = A$, $k = 0$, $\overline{Q}^{(0)} = I$.
while NOT YET CONVERGED **do**
  Compute QR factorization $A^{(k)} = Q^{(k)} R^{(k)}$.
  Compute $A^{(k+1)} = R^{(k)} Q^{(k)}$, $\overline{Q}^{(k+1)} = \overline{Q}^{(k)} Q^{(k)}$
  $k = k + 1$
end while

QR Algorithm is a form of Power Method and Inverse PM

---

while NOT YET CONVERGED **do**

Compute QR factorization $A^{(k)} = Q^{(k)} R^{(k)}$.

Compute $A^{(k+1)} = R^{(k)} Q^{(k)}$, $\overline{Q}^{(k+1)} = \overline{Q}^{(k)} Q^{(k)}$

$k = k + 1$

end while

$$
\begin{aligned}
A^k &= \left(Q^{(0)} Q^{(1)} \ldots Q^{(k-1)}\right) \left(R^{(k-1)} R^{(k-2)} \ldots R^{(0)}\right) \\
&\stackrel{def}{=} \overline{Q}^{(k)} \overline{R}^{(k)},
\end{aligned}
$$

QR Algorithm is a form of Power Method and Inverse PM

---

while NOT YET CONVERGED **do**

Compute QR factorization $A^{(k)} = Q^{(k)}\, R^{(k)}$.

Compute $A^{(k+1)} = R^{(k)}\, Q^{(k)}$, $\overline{Q}^{(k+1)} = \overline{Q}^{(k)}\, Q^{(k)}$

$k = k + 1$

end while

$$
\begin{aligned}
A^k &= \left(Q^{(0)}\, Q^{(1)}\, \ldots\, Q^{(k-1)}\right)\, \left(R^{(k-1)}\, R^{(k-2)}\, \ldots\, R^{(0)}\right) \\
&\stackrel{def}{=} \overline{Q}^{(k)}\, \overline{R}^{(k)}, \quad \text{and} \quad \left(A^{-1}\right)^k = \overline{Q}^{(k)}\, \left(\overline{R}^{(k)}\right)^{-T}.
\end{aligned}
$$

Big eigenvalues converge to top, small eigenvalues bottom

while NOT YET CONVERGED **do**
  Compute QR factorization $A^{(k)} = Q^{(k)} R^{(k)}$.
  Compute $A^{(k+1)} = R^{(k)} Q^{(k)}$, $\overline{Q}^{(k+1)} = \overline{Q}^{(k)} Q^{(k)}$
  $k = k + 1$
end while

$$A^k = \overline{Q}^{(k)} \overline{R}^{(k)}, \quad \text{and} \quad \left(A^{-1}\right)^k = \overline{Q}^{(k)} \left(\overline{R}^{(k)}\right)^{-T}.$$

$$\text{Define} \quad \mathbf{e}_1 = \begin{pmatrix} 1 \\ \mathbf{0} \end{pmatrix}, \quad \mathbf{e}_n = \begin{pmatrix} \mathbf{0} \\ 1 \end{pmatrix},$$

$$\overline{Q}^{(k)} = \left(\overline{\mathbf{q}}_1^{(k)}, \cdots, \overline{\mathbf{q}}_n^{(k)}\right), \quad \overline{R}^{(k)} = \begin{pmatrix} \alpha^{(k)} & * & * \\ & * & * \\ & & \zeta^{(k)} \end{pmatrix}.$$

$$\text{Then} \quad A^k \mathbf{e}_1 = \alpha^{(k)} \overline{\mathbf{q}}_1^{(k)}, \quad \left(A^{-1}\right)^k \mathbf{e}_n = \left(\zeta^{(k)}\right)^{-1} \overline{\mathbf{q}}_n^{(k)}.$$

Assume the eigenvalues of A are all distinct, and

$$|\lambda_1| > |\lambda_2| > \cdots > |\lambda_{n-1}| > |\lambda_n| .$$

- QR Algorithm converges in $A^{(k)}(n, n)$ to λ_n, at rate $\left| \frac{\lambda_n}{\lambda_{n-1}} \right|^k$.

Assume the eigenvalues of A are all distinct, and

$$|\lambda_1| > |\lambda_2| > \cdots > |\lambda_{n-1}| > |\lambda_n| .$$

- QR Algorithm converges in $A^{(k)}(n,n)$ to λ_n, at rate $\left|\dfrac{\lambda_n}{\lambda_{n-1}}\right|^k$.
- QR iteration with SHIFT σ:

$$A - \sigma I = Q R, \quad \text{and} \quad \widehat{A} = R Q + \sigma I \quad \left(= Q^T A Q.\right)$$

Assume the eigenvalues of A are all distinct, and

$$|\lambda_1| > |\lambda_2| > \cdots > |\lambda_{n-1}| > |\lambda_n| .$$

- QR Algorithm converges in $A^{(k)}(n,n)$ to λ_n, at rate $\left| \frac{\lambda_n}{\lambda_{n-1}} \right|^k$.
- QR iteration with SHIFT σ:

$$A - \sigma I = Q R, \quad \text{and} \quad \widehat{A} = R Q + \sigma I \quad \left(= Q^T A Q. \right)$$

Algorithm 3 QR Algorithm with shift σ

Initialize: $A^{(0)} = A$, $k = 0$, $\overline{Q}^{(0)} = I$.
while NOT YET CONVERGED **do**
  Compute QR factorization $A^{(k)} - \sigma I = Q^{(k)} R^{(k)}$.
  Compute $A^{(k+1)} = R^{(k)} Q^{(k)} + \sigma I$, $\overline{Q}^{(k+1)} = \overline{Q}^{(k)} Q^{(k)}$
  $k = k + 1$
end while

Assume the eigenvalues of A are all distinct, and

$$|\lambda_1| > |\lambda_2| > \cdots > |\lambda_{n-1}| > |\lambda_n| .$$

- QR Algorithm converges in $A^{(k)}(n,n)$ to λ_n, at rate $\left| \frac{\lambda_n}{\lambda_{n-1}} \right|^k$.
- QR iteration with SHIFT σ:

$$A - \sigma I = Q R, \quad \text{and} \quad \widehat{A} = R Q + \sigma I \quad \left(= Q^T A Q. \right)$$

Algorithm 4 QR Algorithm with shift σ

Initialize: $A^{(0)} = A$, $k = 0$, $\overline{Q}^{(0)} = I$.
while NOT YET CONVERGED **do**
  Compute QR factorization $A^{(k)} - \sigma I = Q^{(k)} R^{(k)}$.
  Compute $A^{(k+1)} = R^{(k)} Q^{(k)} + \sigma I$, $\overline{Q}^{(k+1)} = \overline{Q}^{(k)} Q^{(k)}$
  $k = k + 1$
end while

Biggest Issue: eigenvalues could be complex

- QR Algorithm converges in $A^{(k)}(n,n)$ to λ_n, at rate $\left|\frac{\lambda_n}{\lambda_{n-1}}\right|^k$.

- QR Algorithm converges in $A^{(k)}(n,n)$ to λ_n, at rate $\left|\frac{\lambda_n}{\lambda_{n-1}}\right|^k$.
- But if λ_n is complex, then $\lambda_{n-1} = \overline{\lambda}_n$ (complex conjugate), with $\left|\frac{\lambda_n}{\lambda_{n-1}}\right|^k = 1$.
- Let $E^{(k)} = \begin{pmatrix} \alpha_{n-1} & \beta_{n-1} \\ \beta_{n-1} & \alpha_n \end{pmatrix}$ be the bottom 2×2 submatrix of $A^{(k)}$, choose WILKINSON SHIFT

$$\sigma = \text{eigenvalue of } E^{(k)} \text{ closest to } \alpha_n = A^{(k)}(n,n).$$

- Wilkinson shift could be complex.
- If A is real, but σ complex, then $A - \sigma I$ becomes complex.

QR Algorithm with complex shift

Algorithm 1 The QR Algorithm with complex shift σ

Compute QR factorization $A - \sigma I = Q R$.
Compute $\widehat{A} = R Q + \sigma I$,
Compute QR factorization $\widehat{A} - \overline{\sigma} I = \widehat{Q} \widehat{R}$.
Compute $A^{\mathbf{new}} = \widehat{R} \widehat{Q} + \overline{\sigma} I$,

$\widehat{A}$ is complex, but $A^{\mathbf{new}}$ is real.

There is a procedure to directly compute $A^{\mathbf{new}}$ without $\widehat{A}$.