Fast Hessenberg QR Iteration for Companion Matrices

David Bindel

Ming Gu

David Garmire

James Demmel

Shivkumar Chandrasekaran

Motivation: Polynomial root finding

A standard algorithm: QR iteration on a companion matrix

- Robust software exists
- It's normwise backward stable
- It's used in Matlab
- But it takes $O(n^3)$ time and $O(n^2)$ storage

Companion matrices

Given

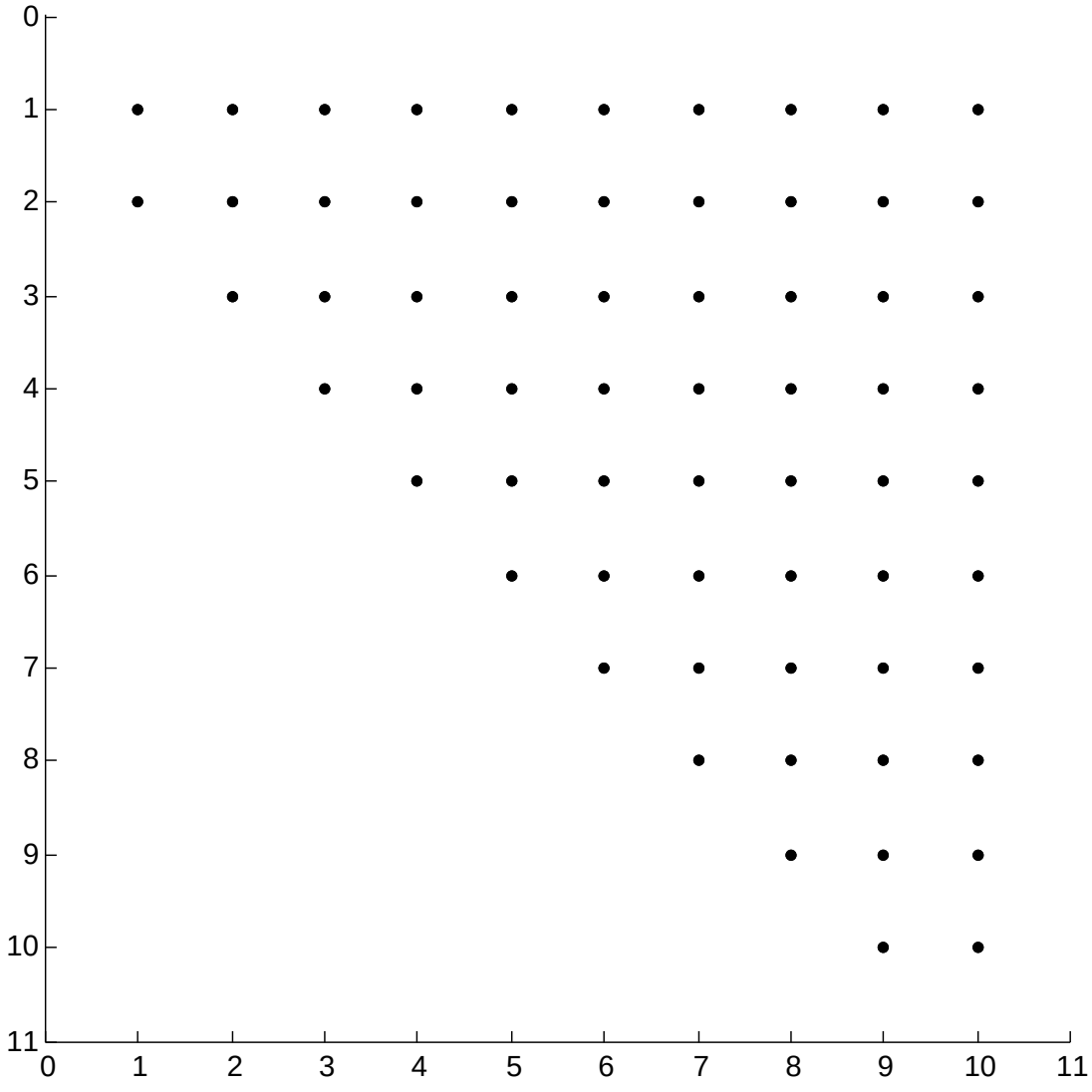
$$p(\lambda) = \lambda^n + a_{n-1}\lambda^{n-1} + \ldots + a_1\lambda + a_0$$

Define a companion matrix C:

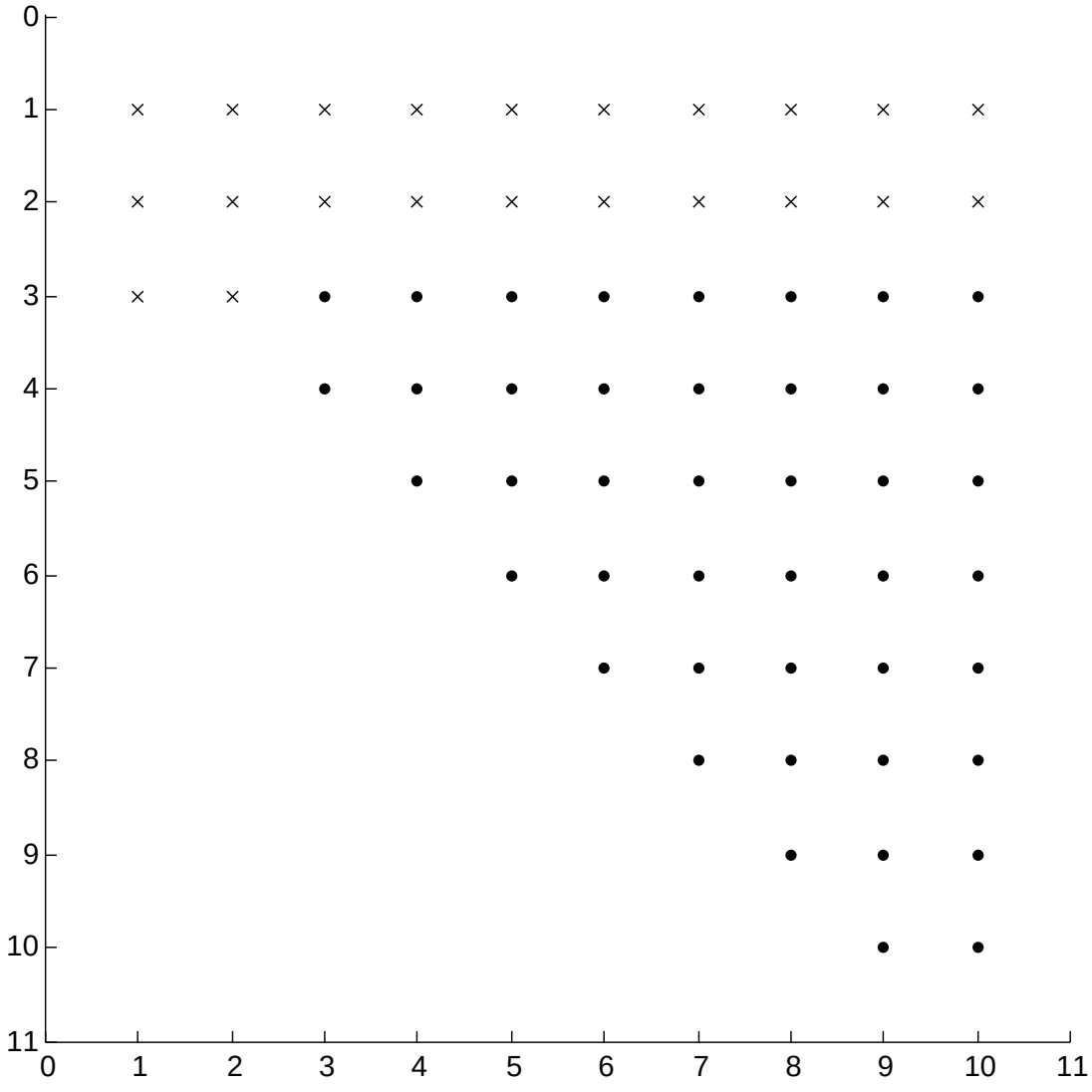
$$C = \begin{bmatrix} -a_{n-1} & -a_{n-2} & -a_{n-3} & \ldots & -a_1 & -a_0 \\ 1 & 0 & 0 & \ldots & 0 & 0 \\ 0 & 1 & 0 & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & 1 & 0 \end{bmatrix}$$

For matrix polynomials, replace a_i by A_i and 1 by I.

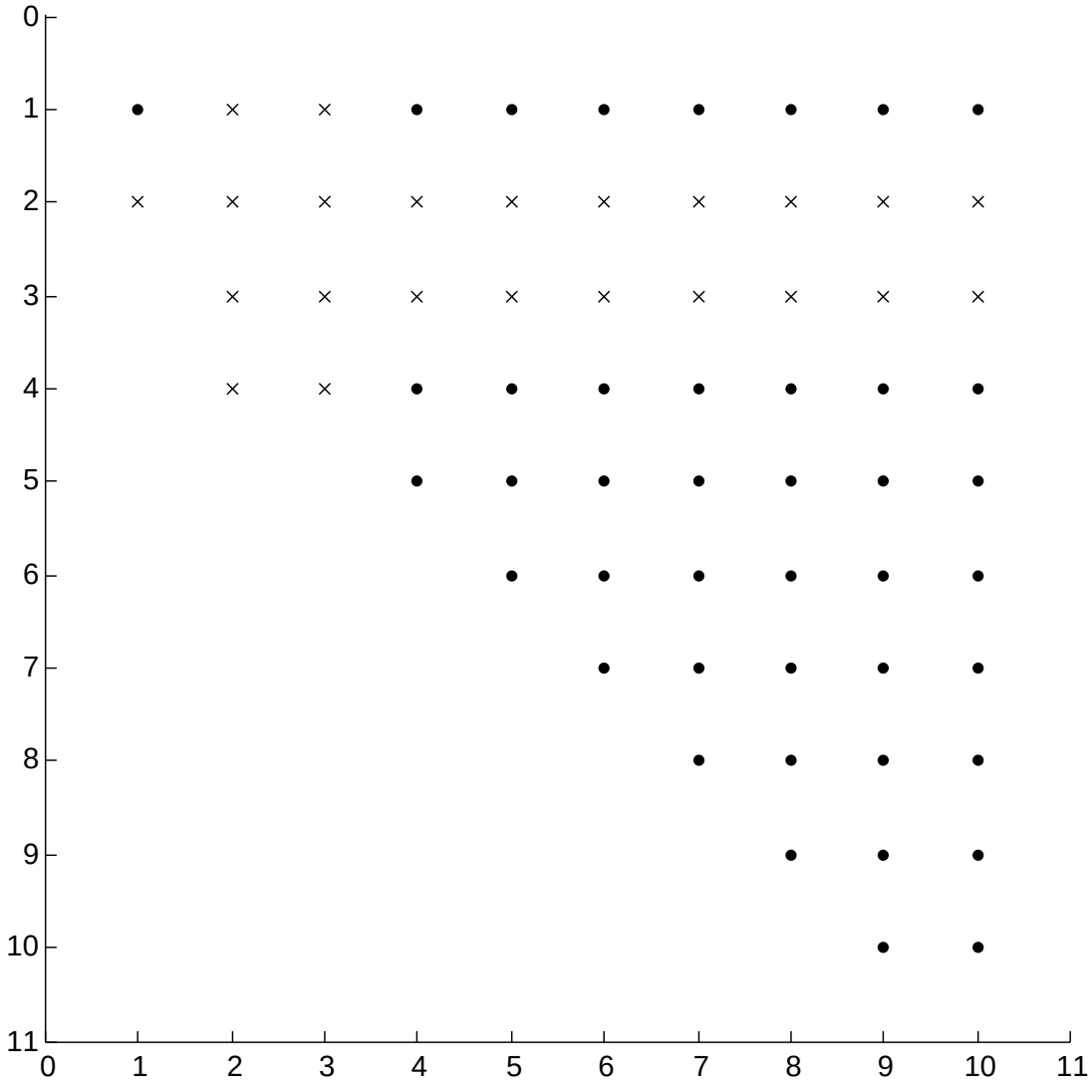
Hessenberg QR iteration



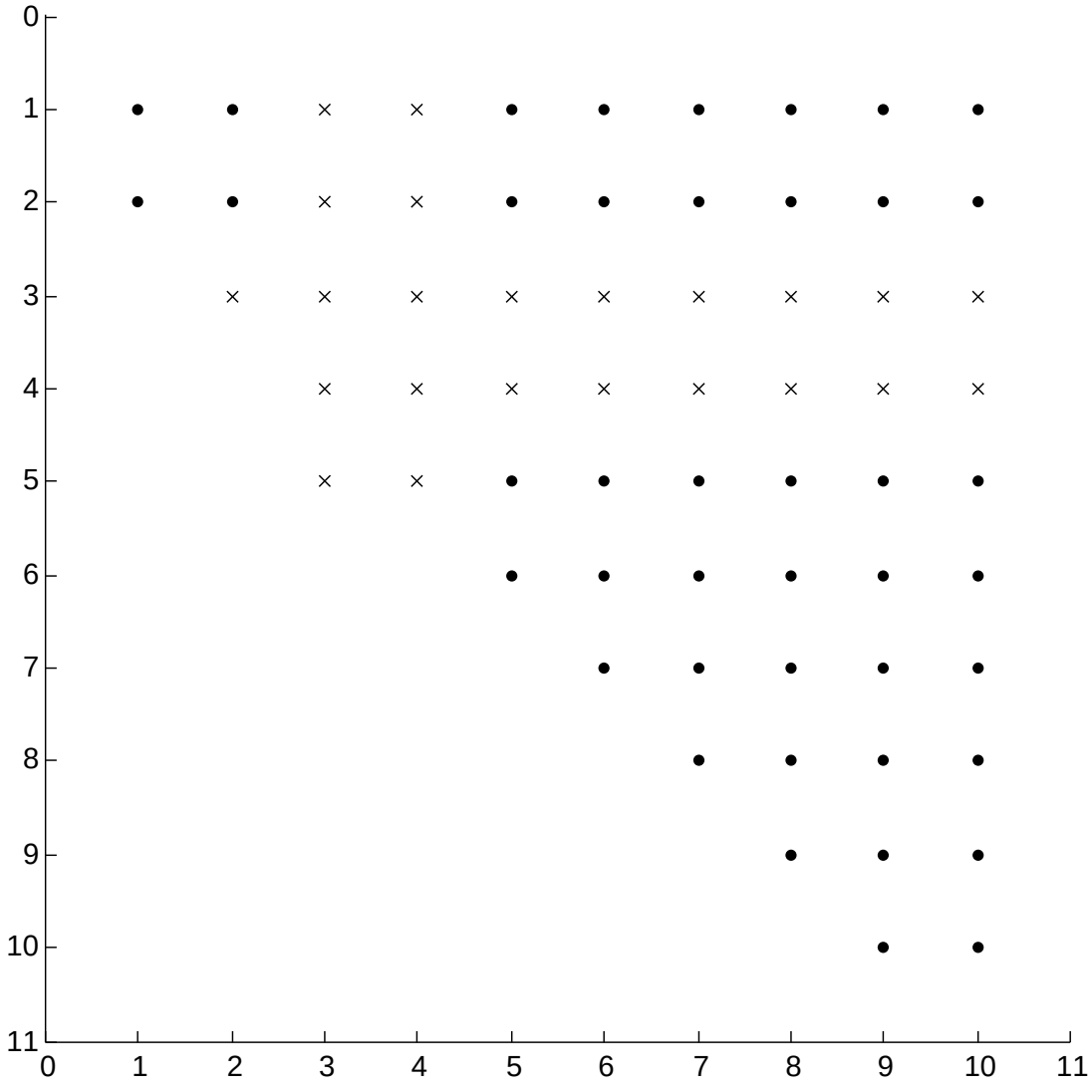
Hessenberg QR iteration



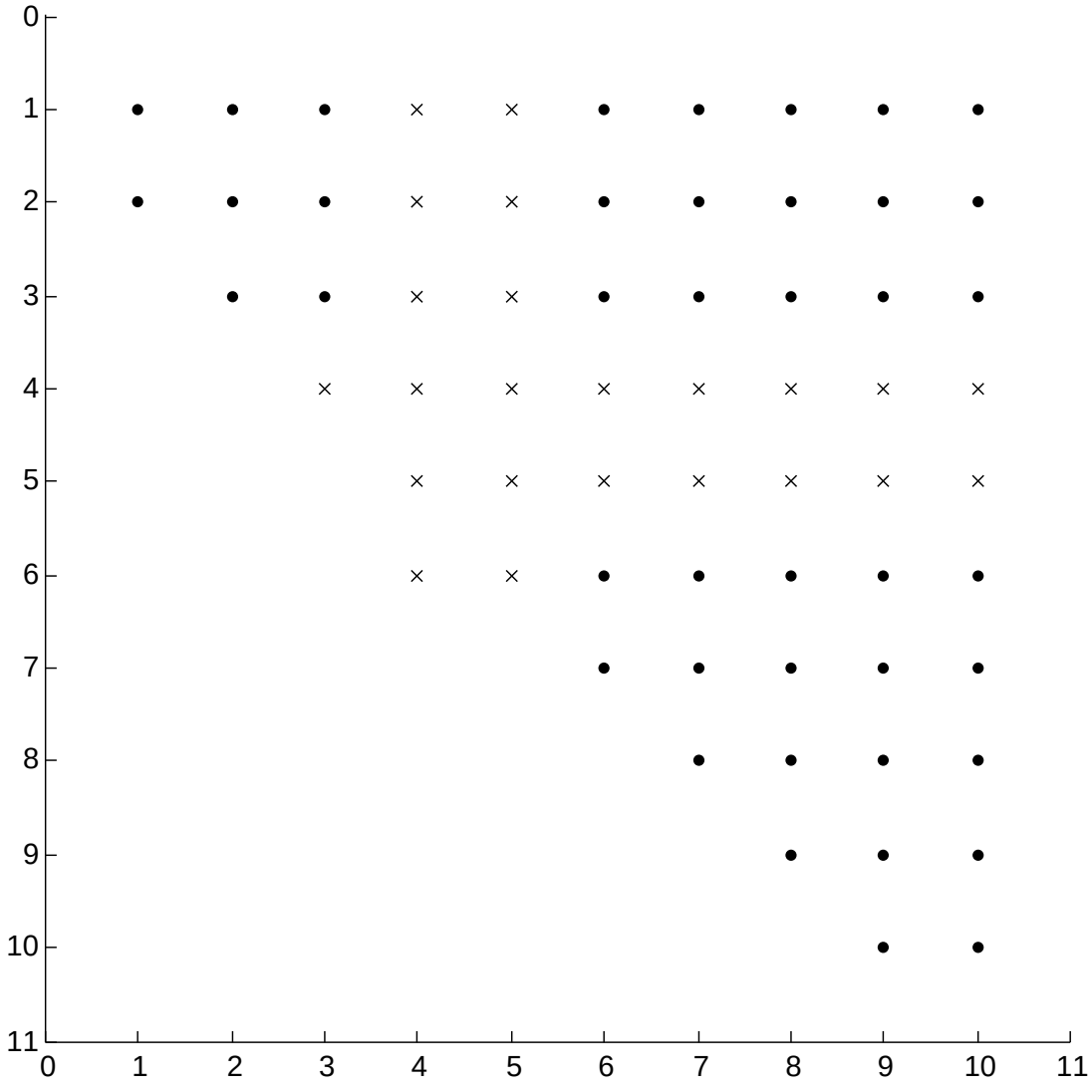
Hessenberg QR iteration



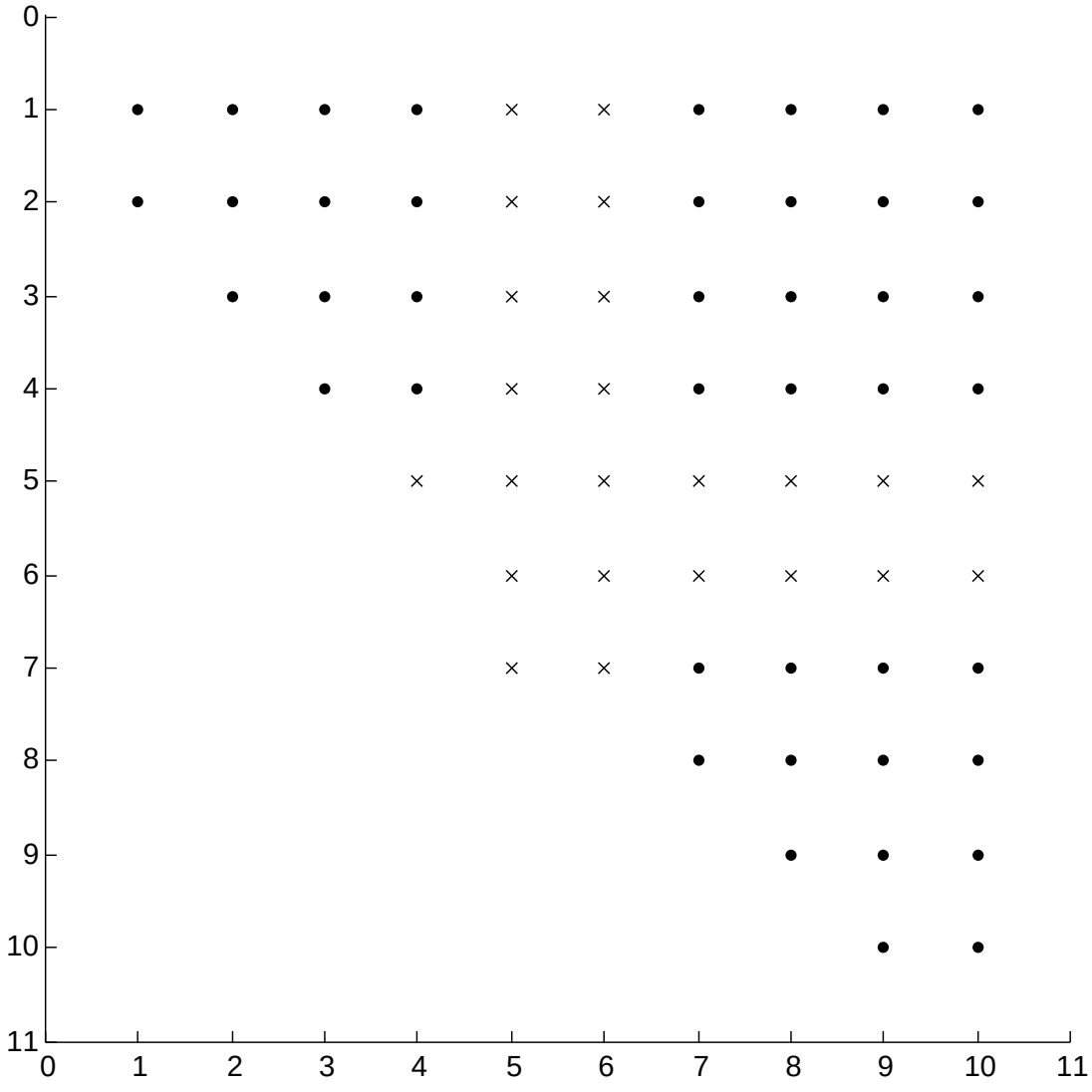
Hessenberg QR iteration



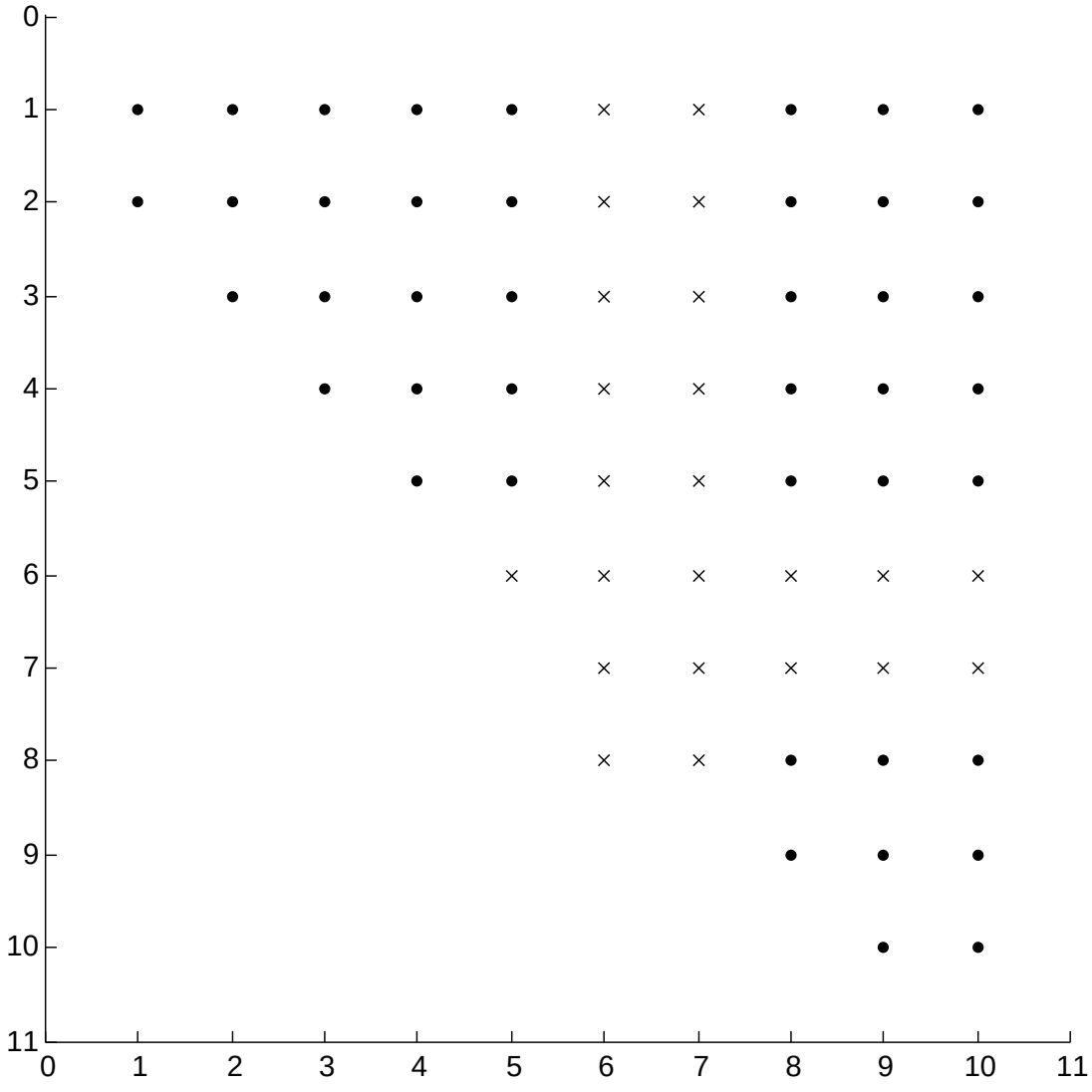
Hessenberg QR iteration



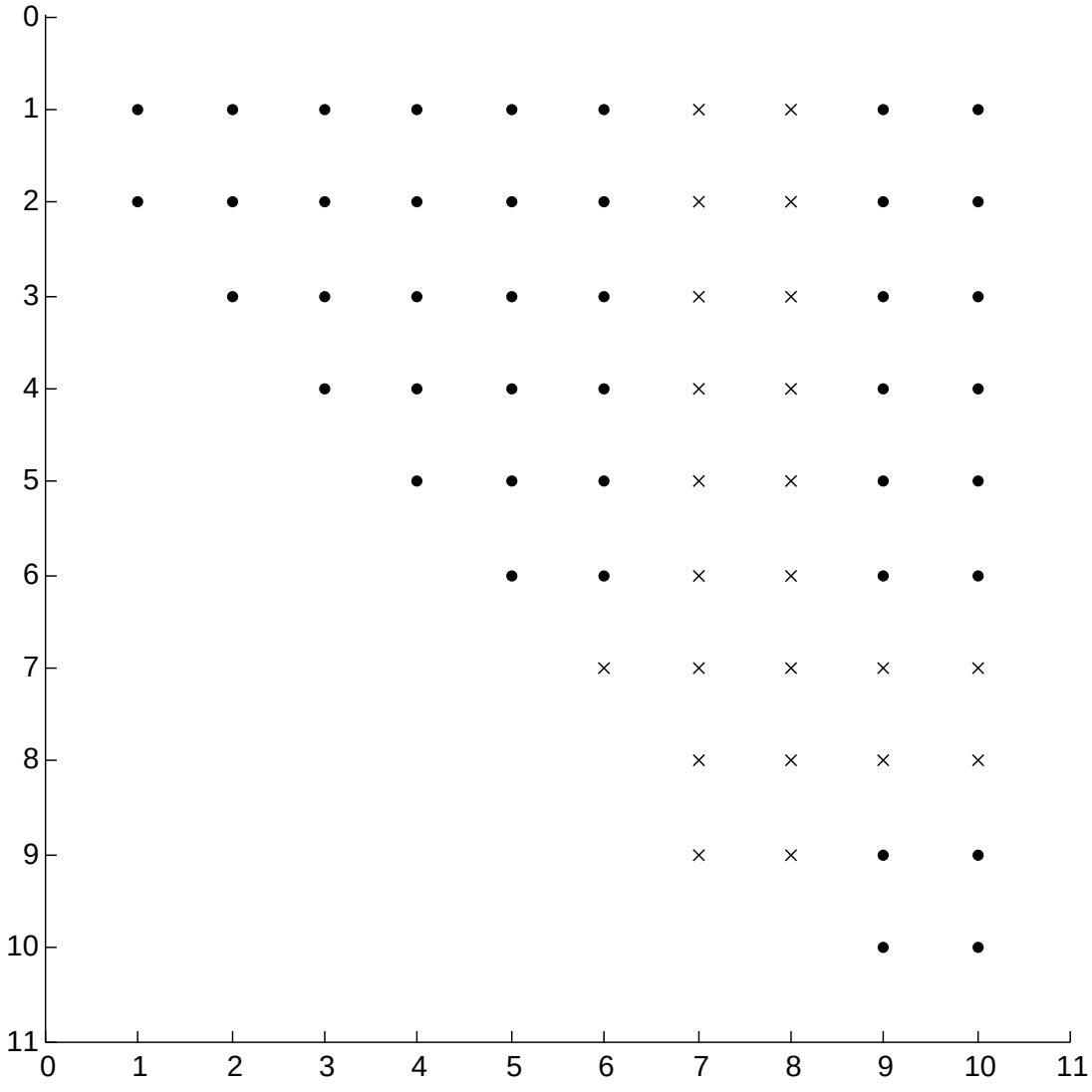
Hessenberg QR iteration



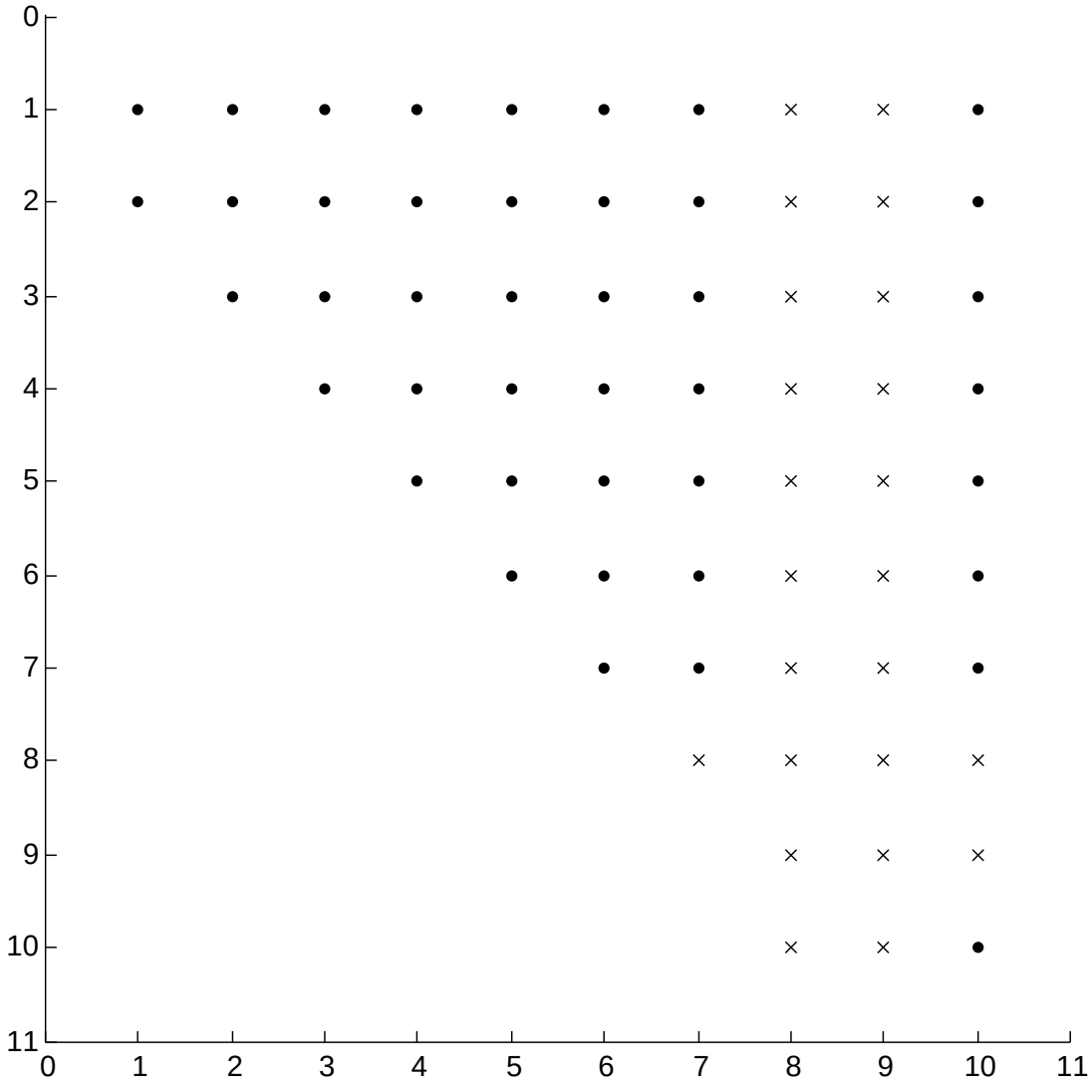
Hessenberg QR iteration



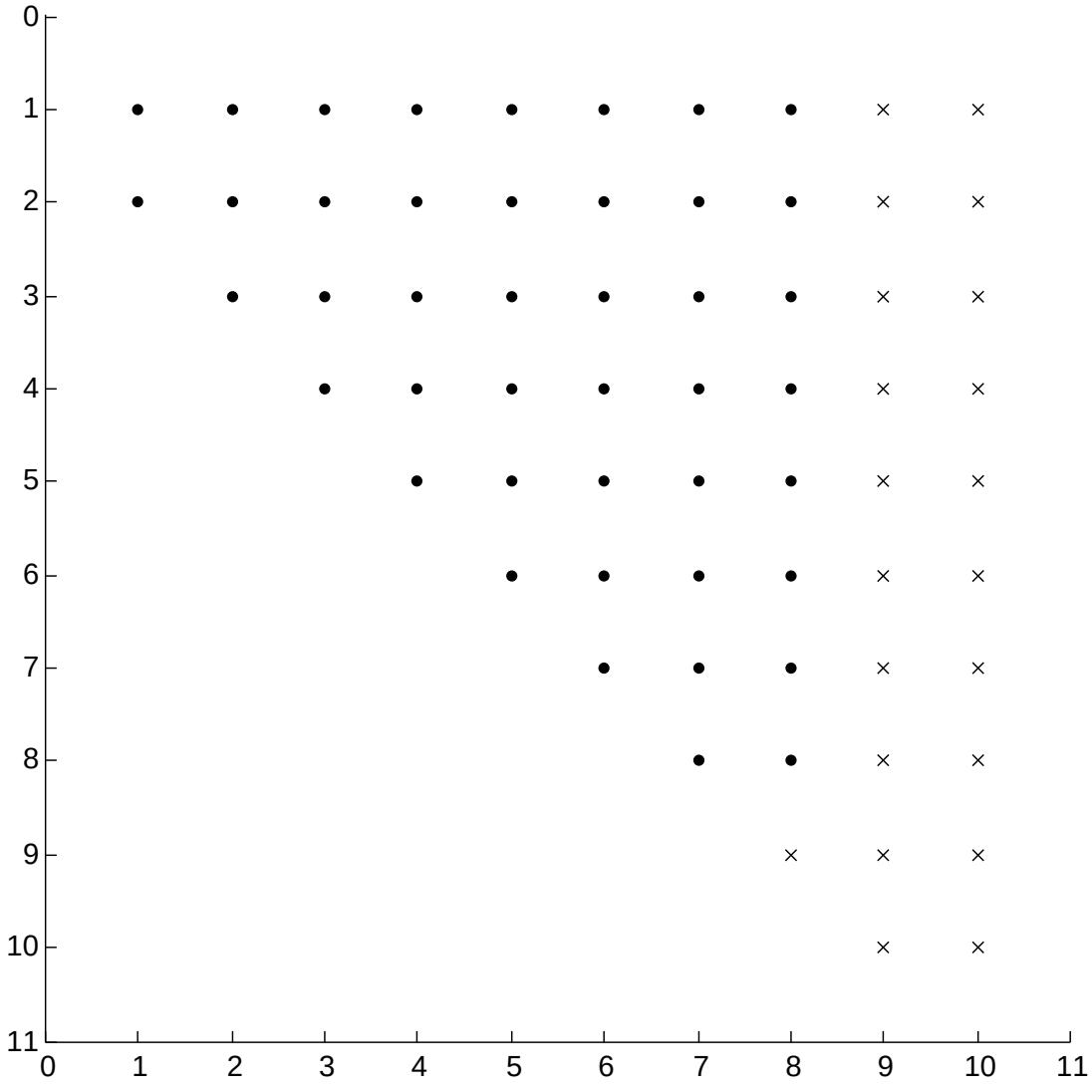
Hessenberg QR iteration



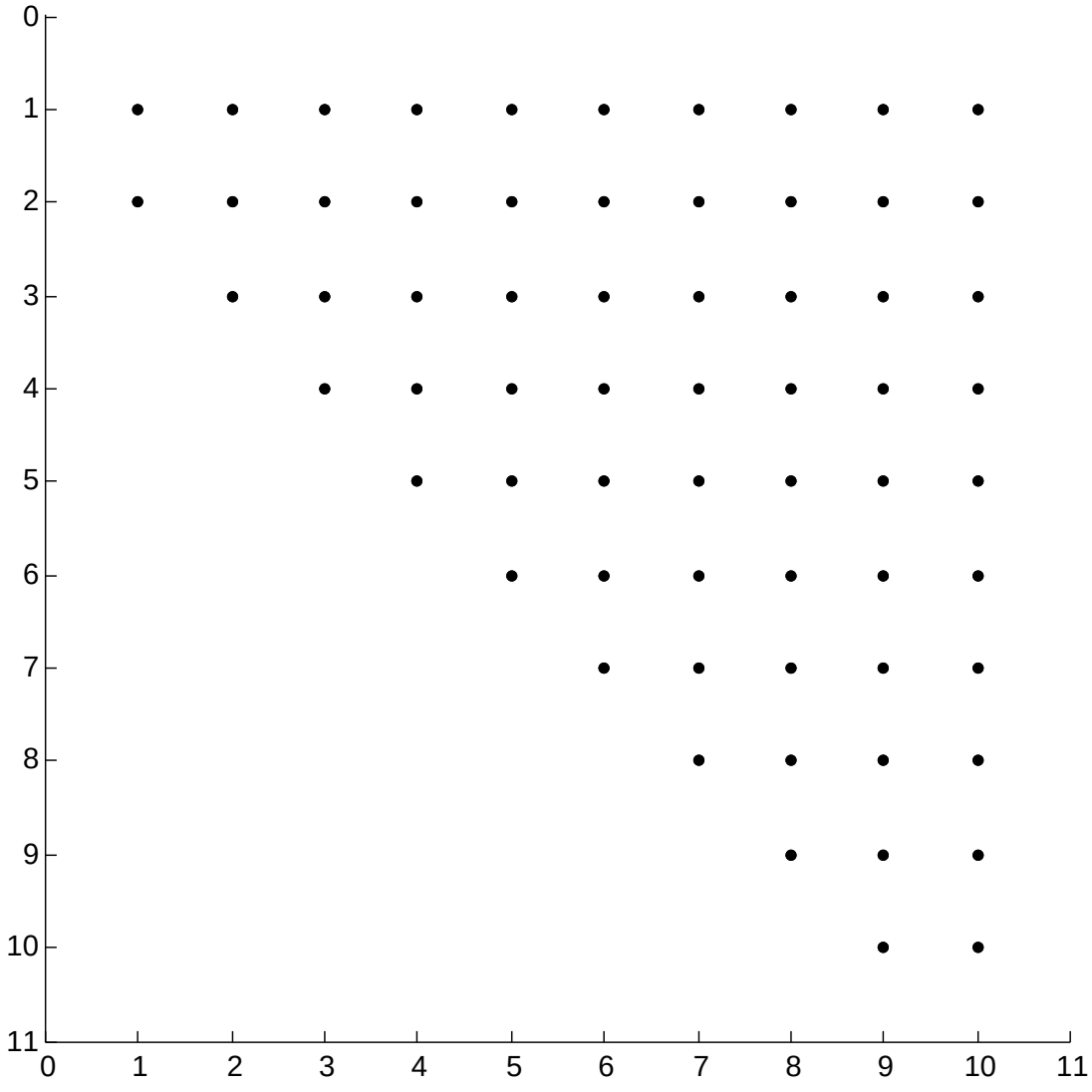
Hessenberg QR iteration



Hessenberg QR iteration



Hessenberg QR iteration



Accelerating QR

Have examples of how to do QR faster:

- Naive implementation $\Longrightarrow$ Hessenberg QR
- Tridiagonal QR (symmetric and skew problems)
- Unitary Hessenberg QR

Keys to fast QR:

- Find a structure preserved by the iteration
- Find a low-complexity representation for that structure

Rank symmetry

Call A *rank-symmetric* if rank$(A_{12}) =$ rank(A_{21}) for any 2-by-2 blocking of A with square A_{11} and A_{22}. Examples:

- Symmetric matrices
- Skew-symmetric matrices
- Orthogonal matrices (by the CS decomposition)
- J-orthogonal matrices (by hyperbolic CS decomposition)

Rank symmetry

Call A *rank-symmetric* if rank$(A_{12}) =$ rank(A_{21}) for any 2-by-2 blocking of A with square A_{11} and A_{22}. Examples:

- Symmetric matrices
- Skew-symmetric matrices
- Orthogonal matrices (by the CS decomposition)
- J-orthogonal matrices (by hyperbolic CS decomposition)

Examples of rank-symmetric + low rank:

- Companion matrices
- Problems which lose self-adjointness at boundary
- Linear oscillators with low-rank damping

Rank symmetric + low rank

Th: For A rank symmetric and L with rank k, both square block 2-by-2:

$$|\text{rank}(A_{12} + L_{12}) - \text{rank}(A_{21} + L_{21})| \leq 2k$$

Proof:

$\text{rank}(L_{ij}) \leq k$, so $|\text{rank}(A_{ij} + L_{ij}) - \text{rank}(A_{ij})| \leq k$.
Note $\text{rank}(A_{ij}) = \text{rank}(A_{ji})$ and apply triangle inequality.

Off-diagonal rank

Any matrix unitarily similar to C is orthogonal + rank 1. Therefore:

- Hessenberg forms similar to C have rank(super-diagonal block) ≤ 3.
- Real Schur forms similar to C have rank(super-diagonal block) ≤ 2.

(For polynomial matrices, make that $3d$ and $2d$).

SSS structure

Write Hessenberg matrices with low off-diagonal rank as narrow band matrix + SSS matrix.

$$H = B + \begin{bmatrix} 0 & U_1 V_2^T & U_1 W_2 V_3^T & U_1 W_2 W_3 V_4^T \\ 0 & 0 & U_2 V_3^T & U_2 W_3 V_4^T \\ 0 & 0 & 0 & U_3 V_4^T \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Write $H_{ij} = U_i W_{i+1} \dots W_{j-1} V_j$ for $i < j$ where $U_i \in \mathbb{R}^{m_b \times 2k+1}$, $V_i \in \mathbb{R}^{m_b \times 2k+1}$, and $W_i \in \mathbb{R}^{2k+1 \times 2k+1}$. Make the block size $m_b \geq 2k+1$.

Total storage cost: $O(nk)$.

SSS structure transformed

$$\begin{array}{cccc} B_{11} & U_1V_2^T & U_1W_2V_3^T & U_1W_2W_3V_4^T \\ B_{21} & B_{22} & U_2V_3^T & U_2W_3V_4^T \\ & B_{32} & B_{33} & U_3V_4^T \\ & & B_{43} & B_{44} \end{array}$$

- Start with an SSS representation

SSS structure transformed

$$
\begin{array}{cccc}
\mathbf{B_{11}} & \mathbf{U_1}V_2^T & \mathbf{U_1}W_2V_3^T & \mathbf{U_1}W_2W_3V_4^T \\
B_{21} & B_{22} & U_2V_3^T & U_2W_3V_4^T \\
 & B_{32} & B_{33} & U_3V_4^T \\
 & & B_{43} & B_{44}
\end{array}
$$

- Start with an SSS representation
- Bulge chase in B_{11}: changes B_{11} and U_1

SSS structure transformed

$$
\begin{array}{cccc}
\mathbf{B_{11}} & \mathbf{B_{12}} & \mathbf{\hat{U}_1} V_3^T & \mathbf{\hat{U}_1} W_3 V_4^T \\
\mathbf{B_{21}} & \mathbf{B_{22}} & \mathbf{\hat{U}_2} V_3^T & \mathbf{\hat{U}_2} W_3 V_4^T \\
 & B_{32} & B_{33} & U_3 V_4^T \\
 & & B_{43} & B_{44}
\end{array}
$$

- Start with an SSS representation
- Bulge chase in B_{11}: changes B_{11} and U_1
- *Merge* first two blocks and chase bulge across boundary

SSS structure transformed

$$\begin{array}{cccc} B_{11} & U_1 \mathbf{V_2}^T & U_1 W_2 V_3^T & U_1 W_2 W_3 V_4^T \\ B_{21} & \mathbf{B_{22}} & \mathbf{U_2} V_3^T & \mathbf{U_2} W_3 V_4^T \\ & B_{32} & B_{33} & U_3 V_4^T \\ & & B_{43} & B_{44} \end{array}$$

- Start with an SSS representation
- Bulge chase in B_{11}: changes B_{11} and U_1
- *Merge* first two blocks and chase bulge across boundary
- *Split* blocks once bulge is within B_{22}

SSS structure transformed

$$
\begin{array}{cccc}
B_{11} & U_1 \hat{\mathbf{V}}_{\mathbf{2}}^T & U_1 \hat{\mathbf{V}}_{\mathbf{3}}^T & U_1 W_2 W_3 V_4^T \\
B_{21} & \mathbf{B_{22}} & \mathbf{B_{23}} & \hat{\mathbf{U}}_{\mathbf{2}} V_4^T \\
 & \mathbf{B_{32}} & \mathbf{B_{33}} & \hat{\mathbf{U}}_{\mathbf{3}} V_4^T \\
 & & B_{43} & B_{44}
\end{array}
$$

- Start with an SSS representation
- Bulge chase in B_{11}: changes B_{11} and U_1
- *Merge* first two blocks and chase bulge across boundary
- *Split* blocks once bulge is within B_{22}
- Continue until Hessenberg form is restored

SSS structure transformed

$$
\begin{array}{cccc}
B_{11} & U_1 V_2^T & U_1 W_2 \mathbf{V_3}^T & U_1 W_2 W_3 V_4^T \\
B_{21} & B_{22} & U_2 \mathbf{V_3}^T & U_2 W_3 V_4^T \\
 & B_{32} & \mathbf{B_{33}} & \mathbf{U_3} V_4^T \\
 & & B_{43} & B_{44}
\end{array}
$$

- Start with an SSS representation
- Bulge chase in B_{11}: changes B_{11} and U_1
- *Merge* first two blocks and chase bulge across boundary
- *Split* blocks once bulge is within B_{22}
- Continue until Hessenberg form is restored

SSS structure transformed

$$
\begin{array}{cccc}
B_{11} & U_1 V_2^T & U_1 W_2 \hat{\mathbf{V}}_{\mathbf{3}}^T & U_1 W_2 \hat{\mathbf{V}}_{\mathbf{4}}^T \\
B_{21} & B_{22} & U_2 \hat{\mathbf{V}}_{\mathbf{3}}^T & U_2 \hat{\mathbf{V}}_{\mathbf{4}}^T \\
 & B_{32} & \mathbf{B_{33}} & \mathbf{B_{34}} \\
 & & \mathbf{B_{43}} & \mathbf{B_{44}}
\end{array}
$$

- Start with an SSS representation
- Bulge chase in B_{11}: changes B_{11} and U_1
- *Merge* first two blocks and chase bulge across boundary
- *Split* blocks once bulge is within B_{22}
- Continue until Hessenberg form is restored

SSS structure transformed

$$
\begin{array}{cccc}
B_{11} & U_1 V_2^T & U_1 W_2 V_3^T & U_1 W_2 W_3 \mathbf{V_4}^T \\
B_{21} & B_{22} & U_2 V_3^T & U_2 W_3 \mathbf{V_4}^T \\
 & B_{32} & B_{33} & U_3 \mathbf{V_4}^T \\
 & & B_{43} & \mathbf{B_{44}}
\end{array}
$$

- Start with an SSS representation
- Bulge chase in B_{11}: changes B_{11} and U_1
- *Merge* first two blocks and chase bulge across boundary
- *Split* blocks once bulge is within B_{22}
- Continue until Hessenberg form is restored

SSS structure transformed

$$\begin{array}{cccc} B_{11} & U_1V_2^T & U_1W_2V_3^T & U_1W_2W_3V_4^T \\ B_{21} & B_{22} & U_2V_3^T & U_2W_3V_4^T \\ & B_{32} & B_{33} & U_3V_4^T \\ & & B_{43} & B_{44} \end{array}$$

- Start with an SSS representation
- Bulge chase in B_{11}: changes B_{11} and U_1
- *Merge* first two blocks and chase bulge across boundary
- *Split* blocks once bulge is within B_{22}
- Continue until Hessenberg form is restored

Row and column bases

Define C_j and G_j by the recurrences

$$C_1 = U_1 \text{ and } C_j = \begin{bmatrix} C_{j-1}W_j \\ U_j \end{bmatrix} \text{ for } j > 1$$

$$G_n = V_n \text{ and } G_j = \begin{bmatrix} V_v \\ G_{j+1}W_j^T \end{bmatrix} \text{ for } j < n.$$

C_j and G_j are row and column bases for the submatrix left of block column j and above block row j.

All C_j orthonormal: left proper form.
All G_j orthonormal: right proper form.

Splitting blocks

Start with C_{i-1} and G_{i+1} orthonormal.

$$\begin{bmatrix} H_{1,i} & \dots & H_{1,n} \\ \vdots & & \vdots \\ H_{i,i} & \dots & H_{i,n} \end{bmatrix} = \begin{bmatrix} C_{i-1} & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} V_i^T & W_i \\ H_{ii} & U_i \end{bmatrix} \begin{bmatrix} 0 & G_{i+1} \\ I & 0 \end{bmatrix}^T .$$

$$\begin{bmatrix} V_j^T & W_j \\ B_{jj} & U_j \end{bmatrix} \rightarrow \begin{bmatrix} V_j^{\alpha T} & W_j^{\alpha} V_j^{\beta T} & W_j^{\alpha} W_j^{\beta} \\ B_{jj}^{\alpha\alpha} & U_j^{\alpha} V_j^{\beta T} & U_j^{\alpha} W_j^{\beta} \\ B_{jj}^{\beta\alpha} & B_{jj}^{\beta\beta} & U_j^{\beta} \end{bmatrix}$$

Do pivoted QR or SVD decomposition to split U, V, W.

Choose $\begin{bmatrix} W_j^{\alpha} \\ U_j^{\alpha} \end{bmatrix}$ with orthonormal columns.

Balancing

Want to *balance* matrix to improve accuracy.
But general diagonal scaling destroys structure!

However: $|C|$ is a companion matrix with Perron vector:

$$x_i = \lambda^i$$

Let

$$D = \mathsf{diag}(x_i^{-1}) = \mathsf{diag}(\lambda^{-i}).$$

Then DCD^{-1}

- Is optimally balanced in the ∞-norm.
- Is $\lambda\times$ orthogonal + low rank
- Is scaling the polynomial indeterminate by λ.

Implementation

Modified DLAHQR (basic double-shift QR) to use the new data structure.

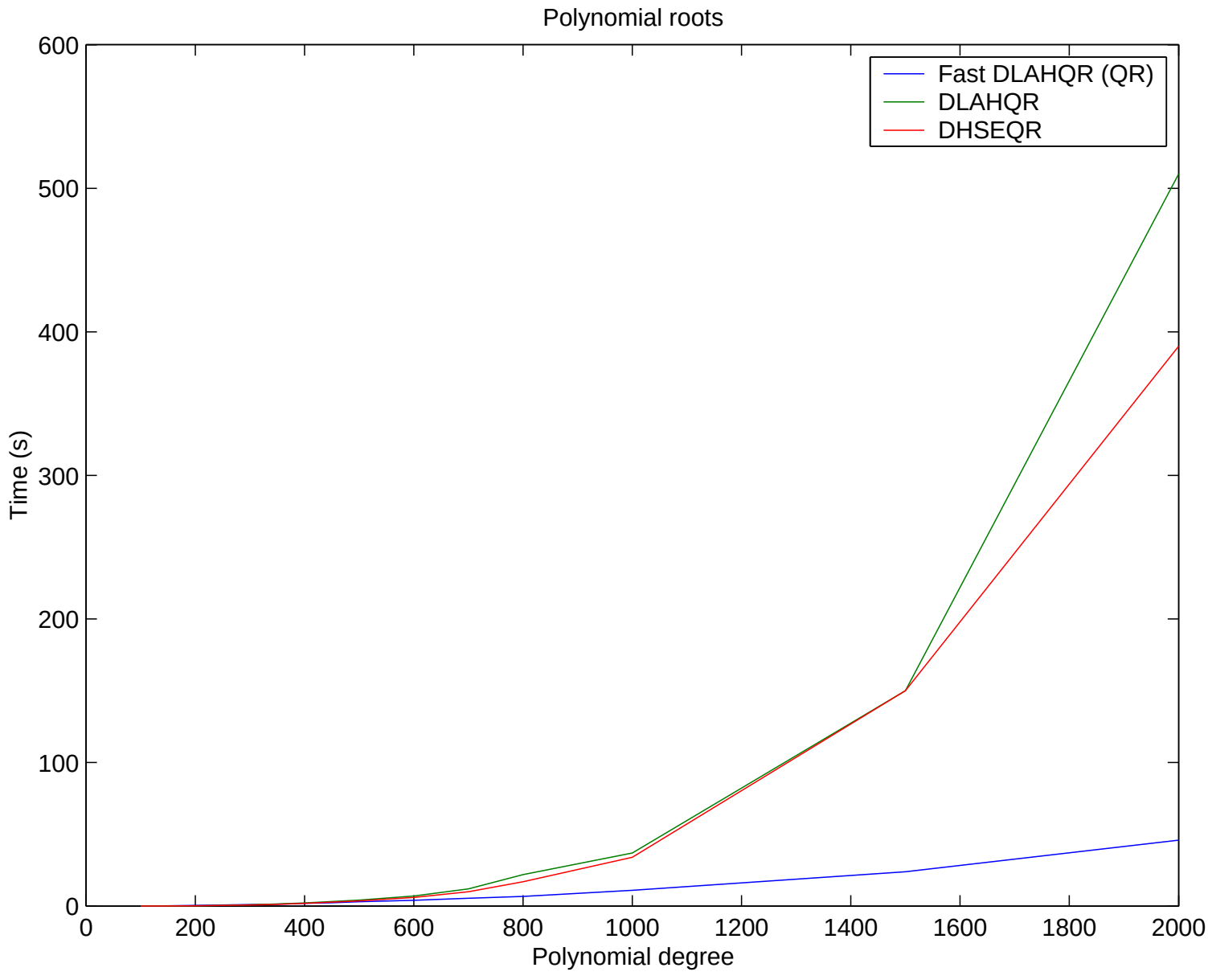
- Convert to right proper form.
- Choose shifts (logic from LAPACK).
- Bulge chase top to bottom / convert to left proper form.
- Check for convergence / deflate (logic from LAPACK).

Most of the code accesses data in the band part of the structure. This text is *identical* to the standard code – just change `LDA`.

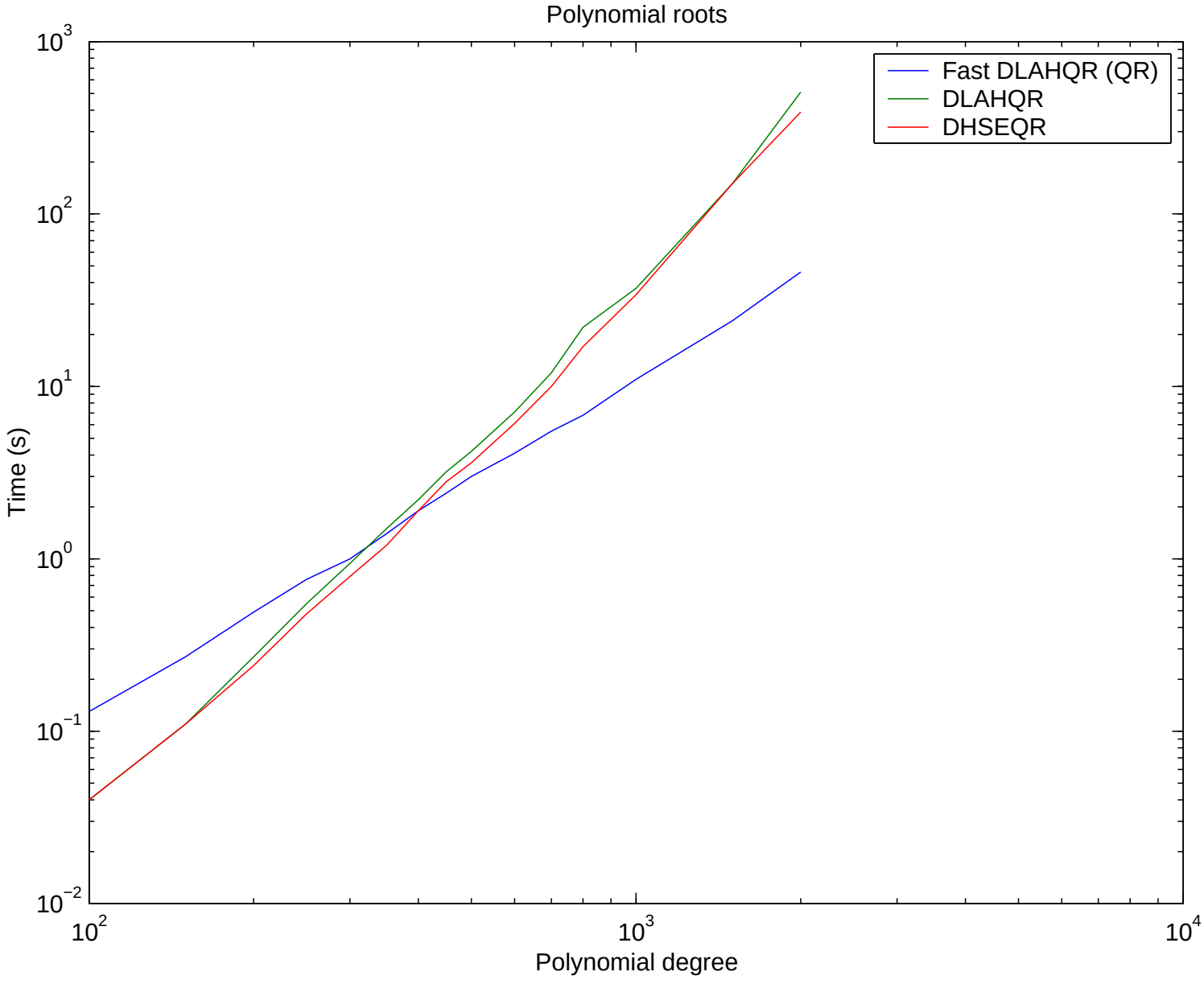
Performance

- Compared against LAPACK DLAHQR and DHSEQR
- Compiled using g77/gcc with -O2 flag
- LAPACK with standard optimizations, ATLAS BLAS
- P4 processor at 1.5 GHz
- Polynomials of degree 100-2000.

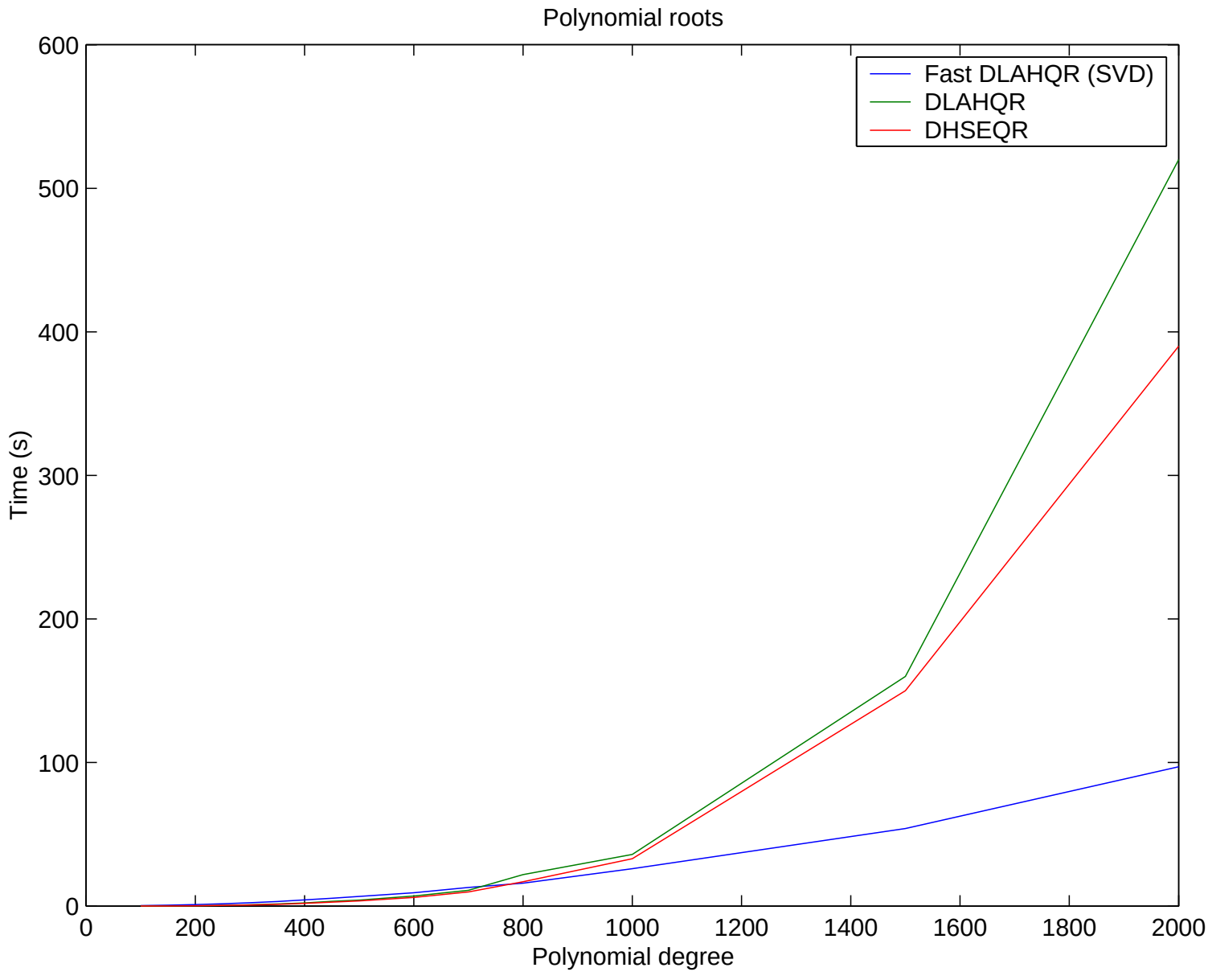
Performance: QR-based



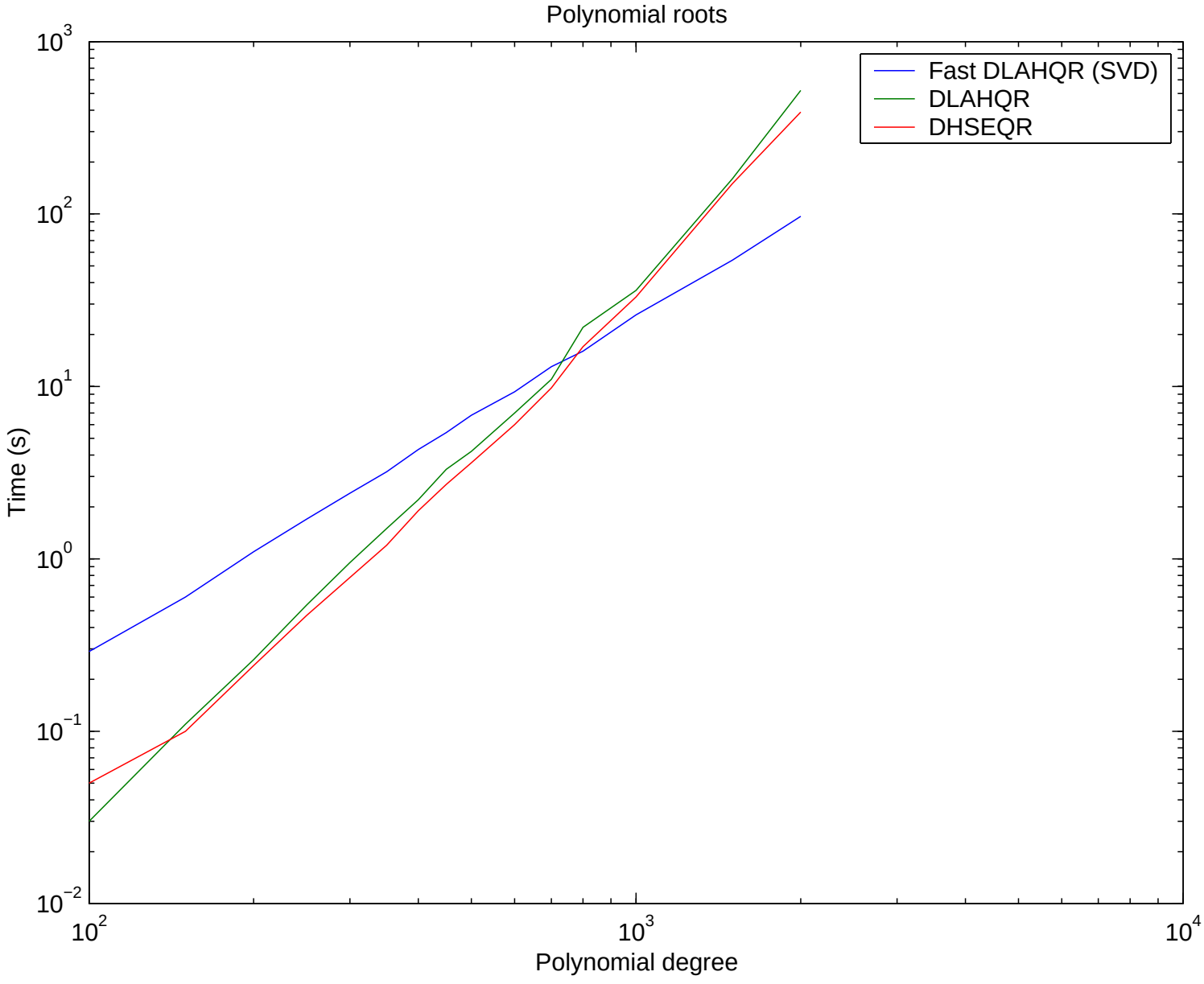
Performance: QR-based



Performance: SVD-based



Performance: SVD-based



Conclusions

- Hessenberg({ symmetric, skew, unitary } + low rank) has low off-diagonal rank.
- Applies to (matrix) polynomial root finding, some damped vibration calculations.
- Can use low-rank structure for fast bulge-chasing.
- Can use-use most existing QR lore and code.
- Appears backward stable with SVD-based splitting.