# YACS

## Yet Another Companion Solver

J.L. Aurentz, T. Mach, L. Robol, R. Vandebril, and D.S. Watkins
Raf.Vandebril@cs.kuleuven.be

Dept. of Computer Science, University of Leuven, Belgium

Back To The Roots – February – 2023

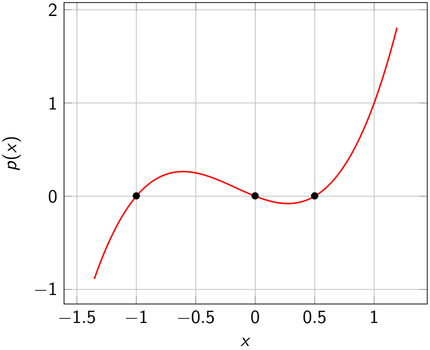
# Outline

# About this Lecture

- We address the Rootfinding Problem.
- Given ($a_i \in \mathbb{C}$ or $a_i \in \mathbb{R}$)

$$p(x) = a_n x^n + a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \ldots + a_1 x + a_0,$$

- find all $\lambda$ such that $p(\lambda) = 0$.

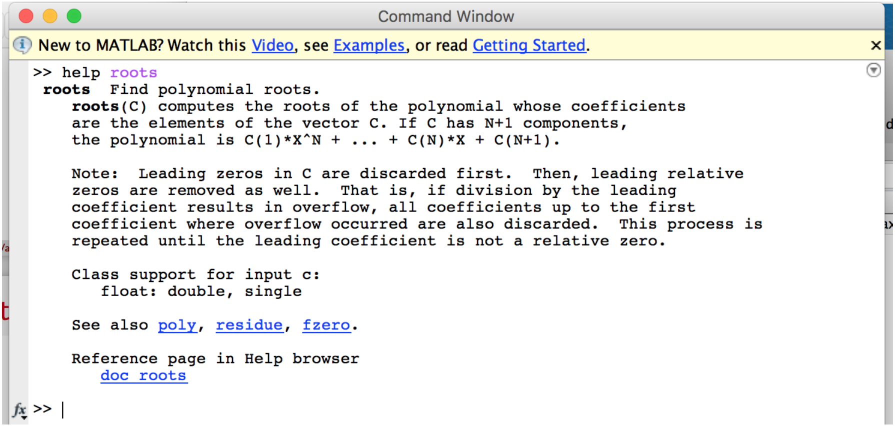


# About Rootfinding

- How would you solve this?

# About Rootfinding

- How would you solve this?
- Use, e.g., "roots" in "Matlab" or "Octave".



# About Matlab's Roots

- What does "roots" do?

# About Matlab's Roots

- What does "roots" do?

**More About**

› Tips

⌄ Algorithms

The algorithm simply involves computing the eigenvalues of the companion matrix:

```
A = diag(ones(n-1,1),-1);
A(1,:) = -c(2:n+1)./c(1);
eig(A)
```

- Coefficients put in first row or last column.
- So let's for simplicity only consider monic ones.
- We'll come back to this later on, in the backward error analysis!

# About Matlab's Roots

- What does "roots" do?
- Given a (complex) monic polynomial

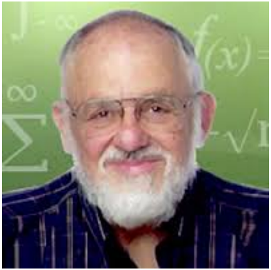
$$p(x) = x^n + a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \ldots + a_0.$$

- Form the companion matrix

$$A = \begin{bmatrix} & & & & & -a_0 \\ 1 & & & & & -a_1 \\ & 1 & & & & -a_2 \\ & & \ddots & & & \vdots \\ & & & 1 & & -a_{n-2} \\ & & & & 1 & -a_{n-1} \end{bmatrix}.$$

- Get the zeros of $p(x) = \det(A - xI)$ by computing the eigenvalues of $A$.

# Comments from Cleve Moler




Clever Moler stated in the original documentation for "roots" the following: (Mathworks Newsletter 1991)

> *It uses order $n^2$ storage and order $n^3$ time. An algorithm designed specifically for polynomial roots might use order n storage and $n^2$ time.*

# Cost of Roots

- MATLAB's roots – xclassical non-structure exploiting algorithm:
  - $O(n^2)$ storage
  - $O(n^3)$ flops
  - Absolute backward error on the polynomial coefficients $\leq \|p\|^2 u$
  - Francis's implicitly-shifted QR algorithm

- Since a decade, structure exploiting algorithms:
  - $O(n)$ storage
  - $O(n^2)$ flops
  - Absolute backward error on the polynomial coefficients $\leq \|p\|^{2,3,4} u$
  - Data-sparse representation and adjusted version of Francis's algorithm
  - Methods proposed by many authors (overview follows).

# Outline

# About "This Lecture"

We designed

a norwise backward stable algorithm for companion matrices

satisfying Cleve's requests: $\mathcal{O}(n)$ storage and $\mathcal{O}(n^2)$ flops.

# About “This Lecture”

We designed

a norwise backward stable algorithm for companion matrices

satisfying Cleve’s requests: $\mathcal{O}(n)$ storage and $\mathcal{O}(n^2)$ flops.

# YACS

Yet Another Companion Solver

# About "The Paper"

- Jared L. Aurentz, Thomas Mach, Raf Vandebril, and David S. Watkins, *Fast and backward stable computation of roots of polynomials*, SIAM J. Matrix Anal. Appl., 36, 2015.

# About "The Paper"

- Jared L. Aurentz, Thomas Mach, Raf Vandebril, and David S. Watkins,
  *Fast and backward stable computation of roots of polynomials*,
  SIAM J. Matrix Anal. Appl., 36, 2015.
- We received a 15-page long review report.
  And the paper was rejected.

# About "The Paper"

- Jared L. Aurentz, Thomas Mach, Raf Vandebril, and David S. Watkins, *Fast and backward stable computation of roots of polynomials*, SIAM J. Matrix Anal. Appl., 36, 2015.
- We received a 15-page long review report.
  And the paper was rejected.
- But in 2017, we received Siam's outstanding paper prize.

# About "The Paper"

- Jared L. Aurentz, Thomas Mach, Raf Vandebril, and David S. Watkins, *Fast and backward stable computation of roots of polynomials*, SIAM J. Matrix Anal. Appl., 36, 2015.
- We received a 15-page long review report.
  And the paper was rejected.
- But in 2017, we received Siam's outstanding paper prize.
- In 2017 we (im)proved

  Absolute backward error on the polynomial coefficients $\leq \|p\|^2 u$

  to

  Absolute backward error on the polynomial coefficients $\leq \|p\|^1 u$
- Jared L. Aurentz, Thomas Mach, Leonardo Robol, Raf Vandebril, and David S. Watkins, *Fast and Backward Stable Computation of Roots of Polynomials, Part II: Backward Error Analysis; Companion Matrix and Companion Pencil*, SIAM J. Matrix Anal. Appl., 39, 2018.

# About The Authors


Celebration DW75 – May 9 and 10 here in Leuven!

# The Rootfinding Problem

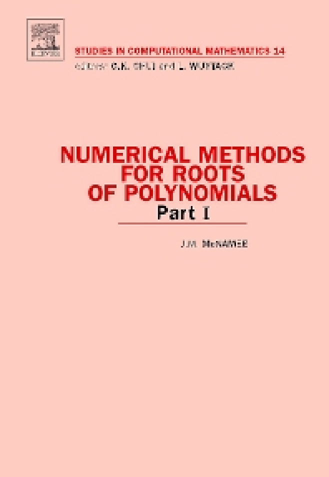
- $p(x) = x^n + a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \ldots + a_0 = 0.$

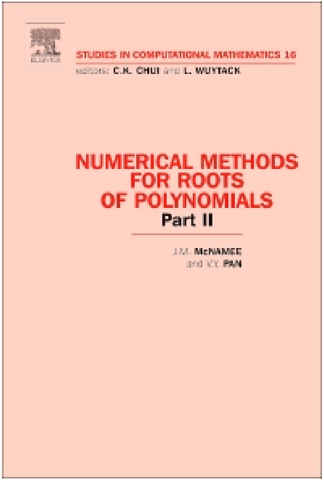
- Already 3000 B.C. people were solving such equations.

- This basis, because it is "one of" the simplest polynomial basis.
  (Other bases lead to, e.g., confederate, companion, fellow,... matrices.)

- Already thousands of methods exists.

J.M. McNamee and V.Y. Pan



# Some Particular Monic Cases

- Case $n = 1$: $p(x) = x^1 + a_0 = 0$.
  - Left as an exercise to the audience.

# Some Particular Monic Cases

- Case $n = 1$: $p(x) = x^1 + a_0 = 0$.
  - Left as an exercise to the audience.

- Case $n = 2$: $p(x) = x^2 + a_1 x^1 + a_0 = 0$.
  - $x_{1/2} = -\frac{a_1}{2} \pm \sqrt{\frac{a_1^2}{4} - a_0}$.

# Some Particular Monic Cases

- Case $n = 1$: $p(x) = x^1 + a_0 = 0$.
  - Left as an exercise to the audience.

- Case $n = 2$: $p(x) = x^2 + a_1x^1 + a_0 = 0$.
  - $x_{1/2} = -\frac{a_1}{2} \pm \sqrt{\frac{a_1^2}{4} - a_0}$.

- Case $n = 3$: $p(x) = x^3 + a_2x^2 + a_1x^1 + a_0 = 0$.
  1. Substitute $x = z - \frac{a_2}{3}$.
  2. This gives $z^3 + uz + v = 0$, with $u = a_1 - \frac{a_2^2}{3}$ and $v = \frac{2a_2^3}{27} - \frac{a_2 a_1}{3} + a_0$.
  3. Compute $\Delta = \frac{v^2}{4} + \frac{u^3}{27}$.
  4. Solve $f = \sqrt[3]{-\frac{v}{2} + \sqrt{\Delta}}$, $g = \sqrt[3]{-\frac{v}{2} - \sqrt{\Delta}}$, with $fg = -\frac{u}{3}$.
  5. $z_1 = f + g$, $z_2 = f\alpha_1 + g\alpha_2$, and $z_3 = f\alpha_2 + g\alpha_1$, with $\alpha_{1/2} = -\frac{1}{2} \pm \frac{1}{2}\imath\sqrt{3}$.
  6. Back substitution.

  (Proof of correctness left again to the attentive listener.)

# Some Particular Cases

- Case $n = 4$: $p(x) = x^4 + a_3x^3 + a_2x^2 + a_1x^1 + a_0 = 0$.
  1. Substitute $x = z - \frac{a_3}{4}$.
  2. This gives $z^4 + uz^2 + vz + w = 0$, with $u = \frac{3a_3^2}{8} + a_2, \ldots$.
  3. ...

  (The whole solution method fills a page.)

# Some Particular Cases

- Case $n = 4$: $p(x) = x^4 + a_3x^3 + a_2x^2 + a_1x^1 + a_0 = 0$.
  1. Substitute $x = z - \frac{a_3}{4}$.
  2. This gives $z^4 + uz^2 + vz + w = 0$, with $u = \frac{3a_3^2}{8} + a_2, \ldots$.
  3. ...

  (The whole solution method fills a page.)

- Case $n = 5$: $p(x) = x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x^1 + a_0 = 0$.
  - Not possible anymore: Abel–Ruffini theorem.

# Some Particular Cases

- Case $n = 4$: $p(x) = x^4 + a_3x^3 + a_2x^2 + a_1x^1 + a_0 = 0$.
  1. Substitute $x = z - \frac{a_3}{4}$.
  2. This gives $z^4 + uz^2 + vz + w = 0$, with $u = \frac{3a_3^2}{8} + a_2, \ldots$.
  3. ...

  (The whole solution method fills a page.)

- Case $n = 5$: $p(x) = x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x^1 + a_0 = 0$.
  - Not possible anymore: Abel–Ruffini theorem.

At least that is what I thought for a very long time.

# Beyond $n = 4$

- I was told that it was impossible and iterative procedures are required.
- Because of:

*The Abel–Ruffini theorem*

- But, there is a small glitch here.
- Abel–Ruffini states:

# Beyond $n = 4$

- I was told that it was impossible and iterative procedures are required.
- Because of:

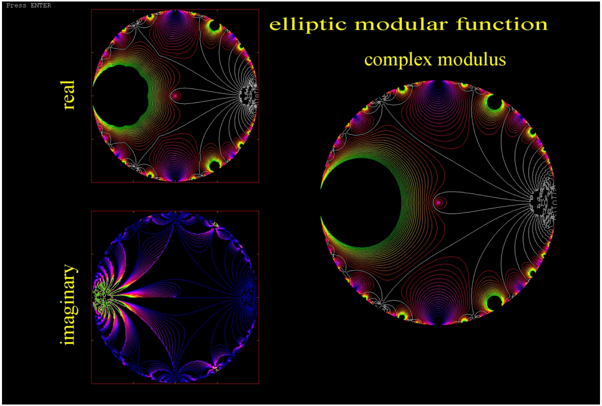
  *The Abel–Ruffini theorem*

- But, there is a small glitch here.
- Abel–Ruffini states:

  *There is no solution only using the coefficients and the following operations*
  - addition,
  - subtraction,
  - multiplication,
  - division,
  - and mth roots.

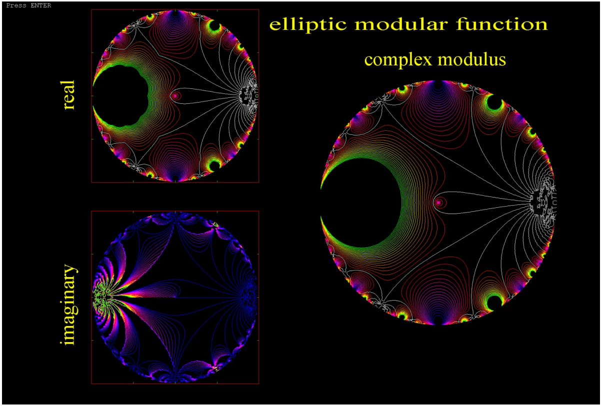
# Beyond $n = 4$

- Case $n = 5$: $p(x) = x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x^1 + a_0 = 0$.
- There is a direct solution method using elliptic modular functions.
- The description fills several pages.



# Beyond $n = 4$

- Case $n = 5$: $p(x) = x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x^1 + a_0 = 0$.
- There is a direct solution method using elliptic modular functions.
- The description fills several pages.




- Before I continue:
  *please do not ask me later on what an elliptic modular function is ...*

# Beyond $n = 4$

- Case $n = 6$: The sextic equation
$$p(x) = x^6 + a_5 x^5 + a_4 x^4 + a_3 x^3 + a_2 x^2 + a_1 x^1 + a_0 = 0.$$
- There is a solution method using Kampé-de-Fériet functions.

# Beyond $n = 4$

- Case $n = 6$: The sextic equation
$$p(x) = x^6 + a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x^1 + a_0 = 0.$$
- There is a solution method using Kampé-de-Fériet functions.
- But, even though for $n = 4, \ldots, 6$ direct methods exists, the complexity grows too fast.
- This was already stated by Gauss.

# Beyond $n = 4$

- Case $n = 6$: The sextic equation
$$p(x) = x^6 + a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x^1 + a_0 = 0.$$
- There is a solution method using Kampé-de-Fériet functions.

- But, even though for $n = 4$, ..., 6 direct methods exists, the complexity grows too fast.

- This was already stated by Gauss.

- So typically iterative methods to approximate the roots.

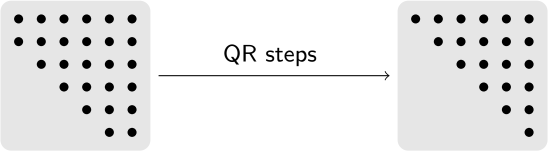
# Outline

# Classical QR algorithm

- Given a Hessenberg matrix $A$, iteratively compute the Schur decomposition

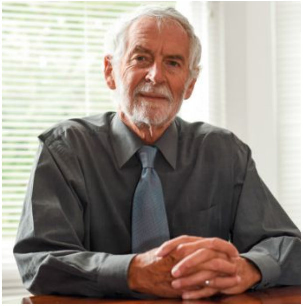
$$Q^{\star}AQ = S,$$

with $Q$ unitary and $S$ upper triangular having the eigenvalues on the diagonal.
- Each iteration is named a QR step.
- So graphically several QR steps lead to



# Implicitly Shifted QR algorithm

- John G.F. Francis and Vera N. Kublanovskaya.
- Also Rutishauser, Wilkinson, ...
- Published in 1961.
- 1962 Francis left for industry.



# An Implicitly Single Shifted QR Step

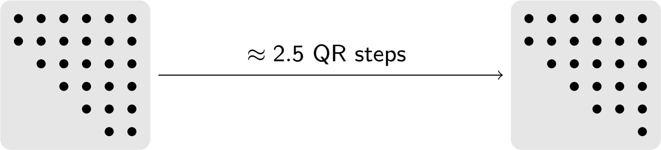
- We execute $n-1$ similarity transformations with rotations.

# An Implicitly Single Shifted QR Step

- We execute $n-1$ similarity transformations with rotations.
- Flow:
  1. Compute a good initial rotation $G_1$ (acts on rows 1 and 2).
  2. Apply it on $A_1 = A$:
  $$G_1^\star A_1 G_1 = A_2.$$
  3. $A_2$ has lost its Hessenberg structure, it has a bulge.
  4. Chase the bulge via similarities with rotations $G_2, \ldots, G_{n-1}$.

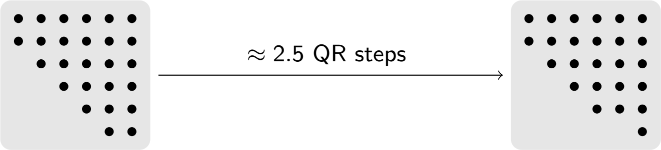
# An Implicitly Single Shifted QR Step

- We execute $n-1$ similarity transformations with rotations.
- Flow:
  1. Compute a good initial rotation $G_1$ (acts on rows 1 and 2).
  2. Apply it on $A_1 = A$:
     $$G_1^{\star} A_1 G_1 = A_2.$$
  3. $A_2$ has lost its Hessenberg structure, it has a bulge.
  4. Chase the bulge via similarities with rotations $G_2, \ldots, G_{n-1}$.
- On average 2.5 QR steps needed to get a subdiagonal element zero. Thus on average 2.5 QR steps per eigenvalue.



# An Implicitly Single Shifted QR Step

- We execute $n-1$ similarity transformations with rotations.
- Flow:
  1. Compute a good initial rotation $G_1$ (acts on rows 1 and 2).
  2. Apply it on $A_1 = A$:
  $$G_1^\star A_1 G_1 = A_2.$$
  3. $A_2$ has lost its Hessenberg structure, it has a bulge.
  4. Chase the bulge via similarities with rotations $G_2, \ldots, G_{n-1}$.
- On average 2.5 QR steps needed to get a subdiagonal element zero. Thus on average 2.5 QR steps per eigenvalue.




- Continue with the remaining unconverged upper part.

# Shorthand Notation for a Rotation

The active part of the rotation is retained.

$$= \begin{bmatrix} 1 & & & & & \\ & 1 & & & & \\ & & \times & \times & & \\ & & \times & \times & & \\ & & & & 1 & \\ & & & & & 1 \end{bmatrix}$$

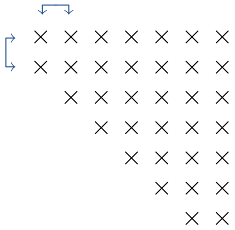
# A Classical QR Step

The original Hessenberg matrix.

$$
\begin{array}{ccccccc}
\times & \times & \times & \times & \times & \times & \times \\
\times & \times & \times & \times & \times & \times & \times \\
 & \times & \times & \times & \times & \times & \times \\
 & & \times & \times & \times & \times & \times \\
 & & & \times & \times & \times & \times \\
 & & & & \times & \times & \times \\
 & & & & & \times & \times
\end{array}
$$

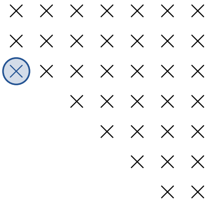
# A Classical QR Step

Executing the similarity with $G_1$ giving $G_1^* A_1 G_1 = A_2$.



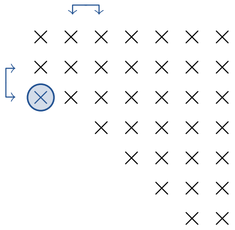
# A Classical QR Step

A bulge is created.



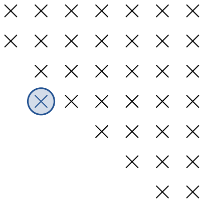
# A Classical QR Step

Remove the bulge via a similarity with $G_2$ giving $G_2^* A_2 G_2 = A_3$.



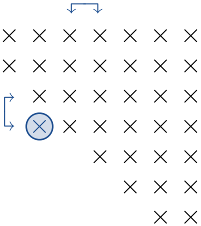
# A Classical QR Step

The bulge has moved down.



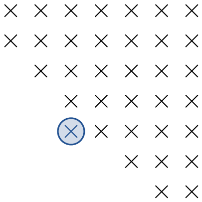
# A Classical QR Step

Continue the procedure.



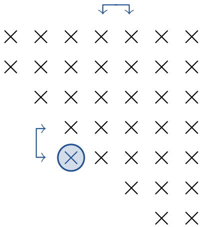
# A Classical QR Step

Continue the procedure.



# A Classical QR Step

Continue the procedure.



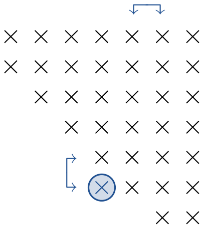
# A Classical QR Step

Continue the procedure.

$$
\begin{array}{ccccccc}
\times & \times & \times & \times & \times & \times & \times \\
\times & \times & \times & \times & \times & \times & \times \\
 & \times & \times & \times & \times & \times & \times \\
 & & \times & \times & \times & \times & \times \\
 & & & \times & \times & \times & \times \\
 & & & \otimes & \times & \times & \times \\
 & & & & & \times & \times
\end{array}
$$

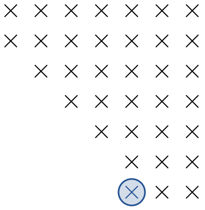
# A Classical QR Step

Continue the procedure.



# A Classical QR Step

Continue the procedure.



# A Classical QR Step

Continue the procedure.

# A Classical QR Step

We have a new, similar Hessenberg matrix.

$$\begin{matrix}
\times & \times & \times & \times & \times & \times & \times \\
\times & \times & \times & \times & \times & \times & \times \\
 & \times & \times & \times & \times & \times & \times \\
 & & \times & \times & \times & \times & \times \\
 & & & \times & \times & \times & \times \\
 & & & & \times & \times & \times \\
 & & & & & \times & \times
\end{matrix}$$

# Deflation

- After sufficient of these steps we typically get

$$A = \begin{bmatrix} \times & \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times & \times \\ & \times & \times & \times & \times & \times & \times \\ & & \times & \times & \times & \times & \times \\ & & & \times & \times & \times & \times \\ & & & & \times & \times & \times \\ & & & & & 0 & \times \end{bmatrix}.$$

# Deflation

- After sufficient of these steps we typically get

$$
A = \left[\begin{array}{cccccc|c}
\times & \times & \times & \times & \times & \times & \times \\
\times & \times & \times & \times & \times & \times & \times \\
 & \times & \times & \times & \times & \times & \times \\
 & & \times & \times & \times & \times & \times \\
 & & & \times & \times & \times & \times \\
 & & & & \times & \times & \times \\
\hline
 & & & & & 0 & \times
\end{array}\right].
$$

- One continues with QR steps, on the upper left part.
  The other parts have converged and are ignored.

# Outline

# New Setting

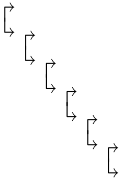
- We do not work on the Hessenberg matrix.
- We work directly on the QR factorization of the Hessenberg.
- Instead of chasing bulges, we chase rotations.
- So we need some tools to manipulate rotations.
- Important: theoretically identical.

# A QR Factored Hessenberg Matrix

- The QR factorization, for $A$ Hessenberg, looks like

$$A \quad = \quad QR$$

$$\begin{bmatrix} \times & \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times & \times \\ & \times & \times & \times & \times & \times & \times \\ & & \times & \times & \times & \times & \times \\ & & & \times & \times & \times & \times \\ & & & & \times & \times & \times \\ & & & & & \times & \times \end{bmatrix} =$$

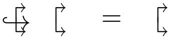


$$\begin{bmatrix} \times & \times & \times & \times & \times & \times & \times \\ & \times & \times & \times & \times & \times & \times \\ & & \times & \times & \times & \times & \times \\ & & & \times & \times & \times & \times \\ & & & & \times & \times & \times \\ & & & & & \times & \times \\ & & & & & & \times \end{bmatrix}.$$

- If $A$ would be unitary Hessenberg, $R$ can be made chosen the identity.

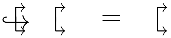
# Manipulating Rotations: Three Operations

- Fusion

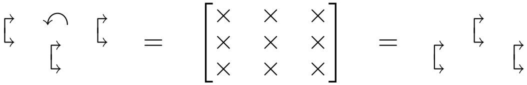


# Manipulating Rotations: Three Operations

- Fusion

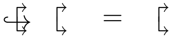


- Turnover

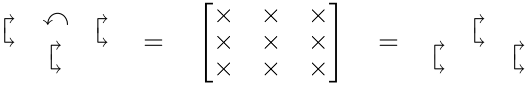


# Manipulating Rotations: Three Operations

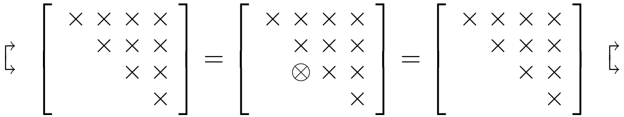
- Fusion



- Turnover

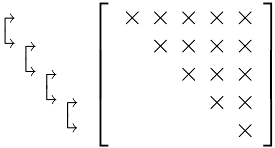


- Pass through an upper triangular



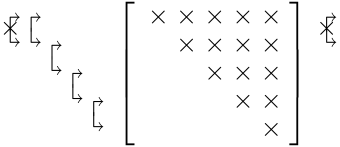
# New QR Step

- The original (factored Hessenberg matrix).



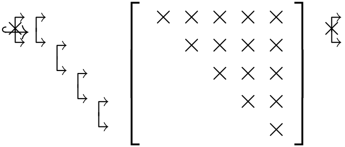
# New QR Step

- Initial similarity transformation with $G_1$ (marked with $\times$) $G_1^* A_1 G_1 = A_2$.



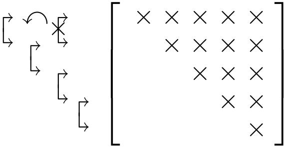
# New QR Step

- Fuse $G_1^*$ on the left.
- Pass $G_1$ (right) the through the upper triangular matrix.



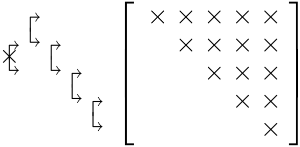
# New QR Step

- Turnover indicated.



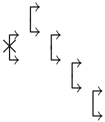
# New QR Step

- We get a perturbing rotator acting on rows 2 and 3.



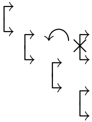
# New QR Step

- Suppress the triangular matrix (everything passes through).
- Start the chasing.
- Eliminate rotater in row 2 and 3 via a similarity:
  - removes the rotator on the left,
  - but add a new one on the right.



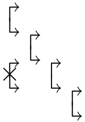
# New QR Step

- Similarity moves rotator to the right.
- Turnover indicated.



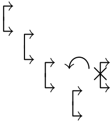
# New QR Step

- Eliminate rotator acting on rows 3 and 4, by similarity.



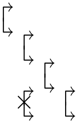
# New QR Step

- Turnover indicated.



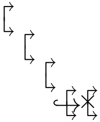
# New QR Step

- Eliminate by similarity the rotator marked with $\times$.



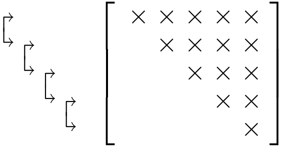
# New QR Step

- A final fusion.



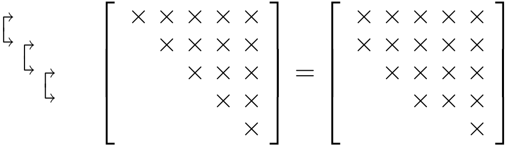
# New QR Step

- Again a Hessenberg matrix.



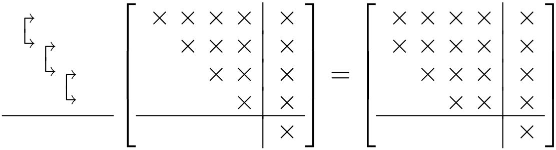
# New QR Step

- Deflation after a few steps.
- Search for diagonal rotations.



# New QR Step

- Deflation after a few steps.
- Continue operating on the upper part



- Remark: Rotation chasing is part of rational QR framework.

# Outline

# The Problem of Today

- Given the complex polynomial.

- $p(x) = x^n + a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \ldots + a_0 = 0.$

- Compute the eigenvalues of the companion matrix

$$A = \begin{bmatrix} & & & & & -a_0 \\ 1 & & & & & -a_1 \\ & 1 & & & & -a_2 \\ & & \ddots & & & \vdots \\ & & & 1 & & -a_{n-2} \\ & & & & 1 & -a_{n-1} \end{bmatrix}.$$

# Fast Companion QR Solvers

- Bini, Daddi, Gemignani (2004): explicit QR on $A = A^{-*} + UV^*$
- Bini, Eidelman, Gemignani, Gohberg (2007): explicit QR on quasisep. $A$
- Chandrasekaran, Gu, Xia, Zhu (2007): implicit QR on $A = QR$
- Delvaux, Frederix, Van Barel (2009/13): implicit QR on $A = QR$
  $R$ in Givens-weight representation
- Van Barel, Vandebril, Van Dooren, Frederix (2010): implicit QR
  unitary-plus-rank-one is preserved, Hessenberg structure is perturbed
- Bini, Boito, Eidelman, Gemignani, Gohberg (2010): now implicit
- Boito, Eidelman, Gemignani, Gohberg (2012): higher stability
- Eidelman, Gohberg, Haimovici (2013): three sequences of rotations

# Structural Fact 1

- Important fact:
  - Companion matrix is unitary-plus-rank-one

$$A = \begin{bmatrix} 0 & \cdots & 0 & e^{i\theta} \\ 1 & & & 0 \\ & \ddots & & \vdots \\ & & 1 & 0 \end{bmatrix} + \begin{bmatrix} 0 & \cdots & 0 & -e^{i\theta} - a_0 \\ 0 & & 0 & -a_1 \\ \vdots & & \vdots & \vdots \\ 0 & \cdots & 0 & -a_{n-1} \end{bmatrix}.$$

  - Unitary-plus-rank-one structure is preserved by unitary similarities:

$$\begin{aligned} A &= U + uv^H \\ Q^\star A Q &= Q^* U Q + (Q^* u)(Q^* v)^\star. \end{aligned}$$

# Structural Fact 2

- Important fact 2:
  - Companion matrix is also upper Hessenberg,
  - this is preserved by Francis's QR algorithm.
  - Remark:
    - the unitary matrix is initially of Hessenberg form too.
    - This is, however, not preserved.
    - Only the sum remains upper Hessenberg.

# Structural Fact 2

- Important fact 2:
  - Companion matrix is also upper Hessenberg,
  - this is preserved by Francis's QR algorithm.
  - Remark:
    - the unitary matrix is initially of Hessenberg form too.
    - This is, however, not preserved.
    - Only the sum remains upper Hessenberg.

- We will therefor run the QR algorithm preserving both
  - Hessenberg structure;
  - unitary-plus-low-rank structure.

# Structural Fact 2

- Important fact 2:
  - Companion matrix is also upper Hessenberg,
  - this is preserved by Francis's QR algorithm.
  - Remark:
    - the unitary matrix is initially of Hessenberg form too.
    - This is, however, not preserved.
    - Only the sum remains upper Hessenberg.

- We will therefor run the QR algorithm preserving both
  - Hessenberg structure;
  - unitary-plus-low-rank structure.

- Numerically this is, however, not feasible.

# Unitary Plus Low Rank

- Consider the splitting in more detail:

$$A \quad = \quad U + uv^T$$

$$\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & & \times & \times & \times \\ & & & \times & \times \end{bmatrix} = \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \boxtimes & \times & \times & \times & \times \\ \boxtimes & \boxtimes & \times & \times & \times \\ \boxtimes & \boxtimes & \boxtimes & \times & \times \end{bmatrix} + \begin{bmatrix} u_1 v_1 & u_1 v_2 & u_1 v_3 & u_1 v_4 & u_1 v_5 \\ u_2 v_1 & u_2 v_2 & u_2 v_3 & u_2 v_4 & u_2 v_5 \\ u_3 v_1 & u_3 v_2 & u_3 v_3 & u_3 v_4 & u_3 v_5 \\ u_4 v_1 & u_4 v_2 & u_4 v_3 & u_4 v_4 & u_4 v_5 \\ u_5 v_1 & u_5 v_2 & u_5 v_3 & u_5 v_4 & u_5 v_5 \end{bmatrix}$$

- The $\boxtimes$ must cancel out with the corresponding $u_i v_j$.

# Unitary Plus Low Rank

- Consider the splitting in more detail:

$$A = U + uv^T$$

$$\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & & \times & \times & \times \\ & & & \times & \times \end{bmatrix} = \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \boxtimes & \times & \times & \times & \times \\ \boxtimes & \boxtimes & \times & \times & \times \\ \boxtimes & \boxtimes & \boxtimes & \times & \times \end{bmatrix} + \begin{bmatrix} u_1v_1 & u_1v_2 & u_1v_3 & u_1v_4 & u_1v_5 \\ u_2v_1 & u_2v_2 & u_2v_3 & u_2v_4 & u_2v_5 \\ u_3v_1 & u_3v_2 & u_3v_3 & u_3v_4 & u_3v_5 \\ u_4v_1 & u_4v_2 & u_4v_3 & u_4v_4 & u_4v_5 \\ u_5v_1 & u_5v_2 & u_5v_3 & u_5v_4 & u_5v_5 \end{bmatrix}$$

- The $\boxtimes$ must cancel out with the corresponding $u_iv_j$.
- A pity: not enough information in $U$ to reconstruct $uv^T$.

# Outline

# Additional Zero Root

- We add an additional zero root to the polynomial.
- $xp(x) = x^{n+1} + a_{n-1}x^n + a_{n-2}x^{n-1} + \ldots + a_0 x + 0 = 0.$
- Companion matrix

$$A = \begin{bmatrix} & & & & & & 0 \\ 1 & & & & & & -a_0 \\ & 1 & & & & & -a_1 \\ & & 1 & & & & -a_2 \\ & & & \ddots & & & \vdots \\ & & & & 1 & & -a_{n-2} \\ & & & & & 1 & -a_{n-1} \end{bmatrix}$$

# Additional Zero Root

- We add an additional zero root to the polynomial.
- $xp(x) = x^{n+1} + a_{n-1}x^n + a_{n-2}x^{n-1} + \ldots + a_0 x + 0 = 0.$
- Companion matrix

$$A = \begin{bmatrix} & & & & & & 0 \\ 1 & & & & & & -a_0 \\ & 1 & & & & & -a_1 \\ & & 1 & & & & -a_2 \\ & & & \ddots & & & \vdots \\ & & & & 1 & & -a_{n-2} \\ & & & & & 1 & -a_{n-1} \end{bmatrix}$$

- In this form: still unable to get $uv^T$ from $U$ in $A = U + uv^T$.
- So: first do a special QR step (shift 0).

# Our Representation (Matrix $A$)

- We perform explicitly theoretically one QR step with shift 0.
- Since this is a perfect shift: theoretical convergence in one step!
- Explicit computation (on paper) without round-off since all rotations are flips.
- After the QR step we obtain (we have overwritten $A$)

$$A = \begin{bmatrix} 0 & & & -a_0 & 1 \\ 1 & & & -a_1 & 0 \\ & \ddots & & \vdots & \vdots \\ & & 1 & -a_{n-1} & 0 \\ & & & 0 & 0 \end{bmatrix}.$$

# Our Representation (Matrix $A$)

- We perform explicitly theoretically one QR step with shift 0.
- Since this is a perfect shift: theoretical convergence in one step!
- Explicit computation (on paper) without round-off since all rotations are flips.
- After the QR step we obtain (we have overwritten $A$)

$$A = \left[\begin{array}{cccc|c} 0 & & & -a_0 & 1 \\ 1 & & & -a_1 & 0 \\ & \ddots & & \vdots & \vdots \\ & & 1 & -a_{n-1} & 0 \\ \hline & & & 0 & 0 \end{array}\right].$$

- Extra zero root can be deflated immediately.
- We apparently end up with the same companion matrix.

# Our Representation (Matrix $A$)

- We perform explicitly theoretically one QR step with shift 0.
- Since this is a perfect shift: theoretical convergence in one step!
- Explicit computation (on paper) without round-off since all rotations are flips.
- After the QR step we obtain (we have overwritten $A$)

$$A = \left[\begin{array}{cccc|c} 0 & & & -a_0 & 1 \\ 1 & & & -a_1 & 0 \\ & \ddots & & \vdots & \vdots \\ & & 1 & -a_{n-1} & 0 \\ \hline & & & 0 & 0 \end{array}\right].$$

- Extra zero root can be deflated immediately.
- We apparently end up with the same companion matrix.
- But, we will still consider the factorization of the entire matrix $A = U + uv^T$.
- Now we can reconstruct $uv^T$ from $U$.

# Advantages of the Additional Root

We will not explain all advantages in detail.

But summarized we have:

- We can reconstruct $uv^T$ from $U$.

# Advantages of the Additional Root

We will not explain all advantages in detail.

But summarized we have:

- We can reconstruct $uv^T$ from $U$.
- We do not need $uv^T$, only $U$.
  As a consequence:
  - faster QR steps, no need to update $u$ nor $v$, (saves 30%)
  - less storage.

# Advantages of the Additional Root

We will not explain all advantages in detail.

But summarized we have:

- We can reconstruct $uv^T$ from $U$.
- We do not need $uv^T$, only $U$.
  As a consequence:
  - faster QR steps, no need to update $u$ nor $v$, (saves 30%)
  - less storage.
- Strong theoretical backward stability results.

# Our Representation (Matrix $A$)

- We start as before, by factoring our $(n+1) \times (n+1)$ Hessenberg matrix $A$.
- Consider it's QR factorization: $A = QR$, where

$$
\begin{aligned}
A &= QR \\
\left[\begin{array}{cccc|c}
0 & & & -a_0 & 1 \\
1 & & & -a_1 & 0 \\
 & \ddots & & \vdots & \vdots \\
 & & 1 & -a_{n-1} & 0 \\
\hline
 & & & 0 & 0
\end{array}\right]
&=
\begin{array}{ccc}
\begin{smallmatrix}\ulcorner\!\rightarrow \\ \llcorner\!\rightarrow\end{smallmatrix} & & \\
 & \ddots & \\
 & & \begin{smallmatrix}\ulcorner\!\rightarrow \\ \llcorner\!\rightarrow\end{smallmatrix}
\end{array}
\left[\begin{array}{cccc|c}
1 & & & -a_1 & 0 \\
 & \ddots & & \vdots & \vdots \\
 & & 1 & -a_{n-1} & 0 \\
 & & & \pm a_0 & \mp 1 \\
\hline
 & & & 0 & 0
\end{array}\right] \\
&= Q_1 Q_2 \cdots Q_{n-1} R.
\end{aligned}
$$

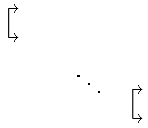
- The deflation is visible in $Q$ as well since $Q_n = I$.

# Our Representation (Matrix $A$)

- We start as before, by factoring our $(n+1) \times (n+1)$ Hessenberg matrix $A$.
- Consider it's QR factorization: $A = QR$, where

$$A \quad = \quad QR$$

$$\left[\begin{array}{cccc|c} 0 & & & -a_0 & 1 \\ 1 & & & -a_1 & 0 \\ & \ddots & & \vdots & \vdots \\ & & 1 & -a_{n-1} & 0 \\ \hline & & & 0 & 0 \end{array}\right] =$$



$$\left[\begin{array}{ccc|c} 1 & & -a_1 & 0 \\ & \ddots & \vdots & \vdots \\ & 1 & -a_{n-1} & 0 \\ & & \pm a_0 & \mp 1 \\ \hline & & 0 & 0 \end{array}\right]$$

$$= \quad Q_1 Q_2 \cdots Q_{n-1} R.$$

- The deflation is visible in $Q$ as well since $Q_n = I$.
- It remains to factor the upper triangular $R$.

# Our Representation (Matrix $R$)

$$R = \left[\begin{array}{cccc|c} 1 & & & -a_1 & 0 \\ & \ddots & & \vdots & \vdots \\ & & 1 & -a_{n-1} & 0 \\ & & & \pm a_0 & \mp 1 \\ \hline & & & 0 & 0 \end{array}\right]$$

- $R$ is unitary-plus-rank-one:

$$R = \left[\begin{array}{cccc|c} 1 & & & 0 & 0 \\ & \ddots & & \vdots & \vdots \\ & & 1 & 0 & 0 \\ & & & 0 & \mp 1 \\ \hline & & & \pm 1 & 0 \end{array}\right] + \left[\begin{array}{cccc|c} 0 & & & -a_1 & 0 \\ & \ddots & & \vdots & \vdots \\ & & 0 & -a_{n-1} & 0 \\ & & & \pm a_0 & 0 \\ \hline & & & \mp 1 & 0 \end{array}\right]$$

# Representation of $R$

- $R = U + xy^T$, where

$$xy^T = \begin{bmatrix} -a_1 \\ \vdots \\ -a_{n-1} \\ \pm a_0 \\ \hline \mp 1 \end{bmatrix} \left[ \begin{array}{cccc|c} 0 & \cdots & 0 & 1 & 0 \end{array} \right]$$

- Next step: Roll up $x$. Thus project $x$ onto $e_1$ with rotators.

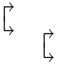
# Representation of $R$

$$\begin{bmatrix} \times \\ \times \\ \times \\ \times \end{bmatrix} = \begin{bmatrix} \times \\ \times \\ \times \\ \times \end{bmatrix}$$

# Representation of $R$

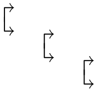
$$\begin{matrix} \\ \\ \ulcorner\!\!\to \\ \llcorner\!\!\to \end{matrix} \begin{bmatrix} \times \\ \times \\ \times \\ \times \end{bmatrix} = \begin{bmatrix} \times \\ \times \\ \times \\ 0 \end{bmatrix}$$

# Representation of $R$



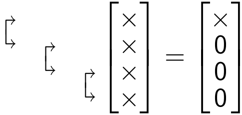
$$\begin{bmatrix} \times \\ \times \\ \times \\ \times \end{bmatrix} = \begin{bmatrix} \times \\ \times \\ {\color{red}0} \\ 0 \end{bmatrix}$$

# Representation of $R$



$$\begin{bmatrix} \times \\ \times \\ \times \\ \times \end{bmatrix} = \begin{bmatrix} \times \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

# Representation of $R$



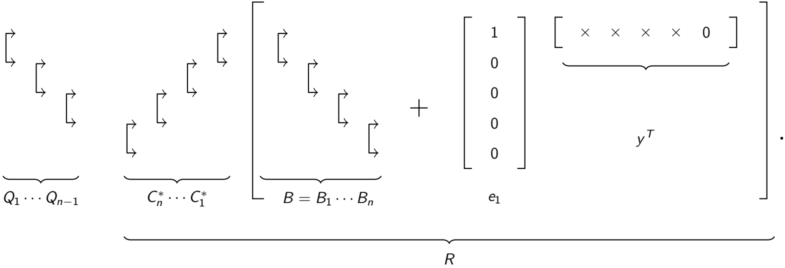
So we get (the vector is of length $n + 1$)

$$
\begin{aligned}
C_1 \cdots C_n \, x &= \alpha e_1 \qquad \text{(w.l.g. } \alpha = 1\text{)} \\
x &= C^* e_1 = C_n^* \ldots C_1^* \, e_1.
\end{aligned}
$$

# Representation of $A$

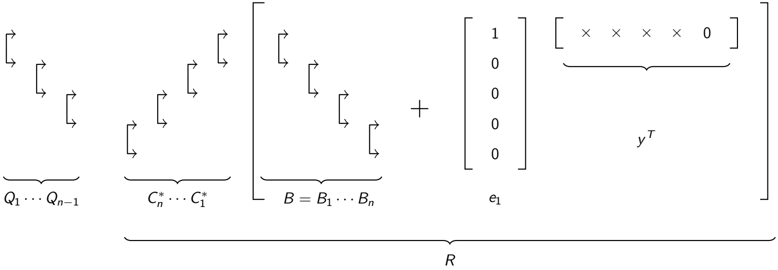
Altogether we have

- $A = QR = Q\ C^* (B + e_1 y^T)$, with $C^* B = U$.
- Again $B = CU$ is unitary Hessenberg: $B = B_1 \cdots B_n$.
- $A = Q_1 \cdots Q_{n-1}\ C_n^* \cdots C_1^* (B_1 \cdots B_n + e_1 y^T)$.



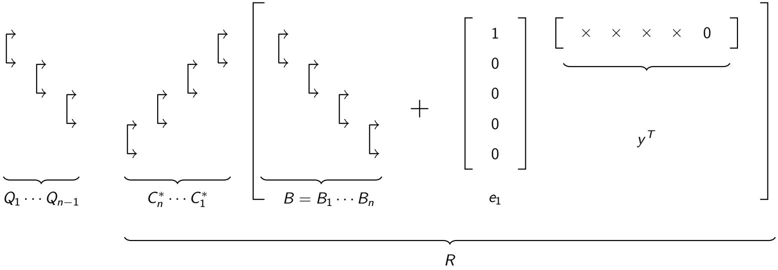
# Outline

# $B$ and $C$ contain the information in $y$




- Recall: $A$ is now of size $(n+1) \times (n+1)$.
- $C^*(B + e_1 y^T) = R$ and $R \in \mathbb{C}^{(n+1)\times(n+1)}$ is upper triangular

# $B$ and $C$ contain the information in $y$



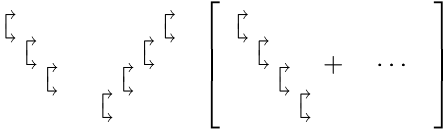

- Recall: $A$ is now of size $(n+1) \times (n+1)$.
- $C^*(B + e_1 y^T) = R$ and $R \in \mathbb{C}^{(n+1)\times(n+1)}$ is upper triangular
- The complete last row of $R$ is zero: $e_{n+1} R = 0 = e_{n+1}(C^*(B + e_1 y^T))$.
- Therefore $y^T = -\rho^{-1} e_{n+1}^T C^* B$, with $\rho = e_{n+1}^T C^* e_1$
- Only possible because of the additional root!

# Outline

# Original Hessenberg Matrix $A$

Altogether we have

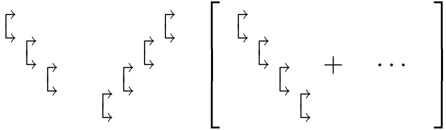
- $A = QR = Q\ C^* \, (B + e_1 y^T)$
- $A = Q_1 \cdots Q_{n-1}\, C_n^* \cdots C_1^* \, (B_1 \cdots B_n + e_1 y^T)$



# Original Hessenberg Matrix $A$

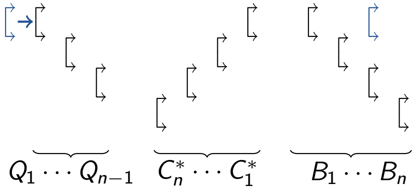
Altogether we have

- $A = QR = Q\ C^* (B + e_1 y^T)$
- $A = Q_1 \cdots Q_{n-1}\ C_n^* \cdots C_1^* (B_1 \cdots B_n + e_1 y^T)$



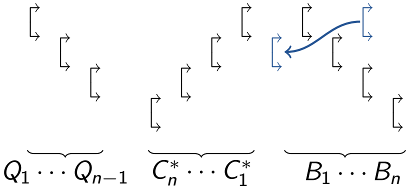
- We will ignore the rank one part!
- The rank one part is encoded in the unitary matrices.

# The Chase



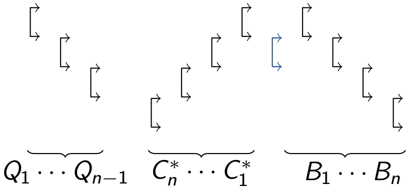

Similarity 1

# The Chase



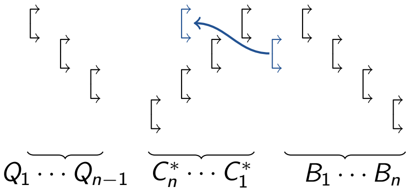

Similarity 1

# The Chase



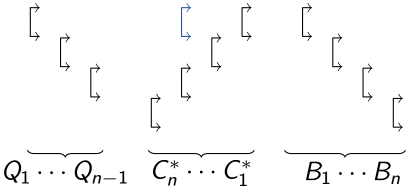

Similarity 1

# The Chase



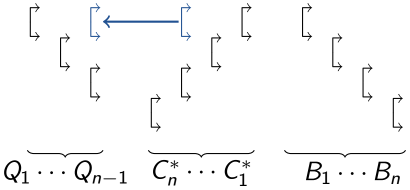

Similarity 1

# The Chase



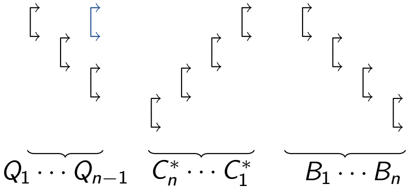

Similarity 1

# The Chase



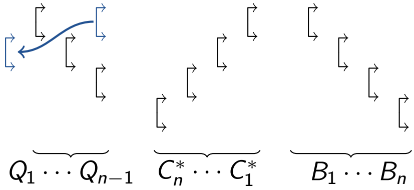

Similarity 1

# The Chase



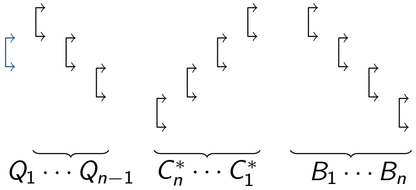

Similarity 1

# The Chase



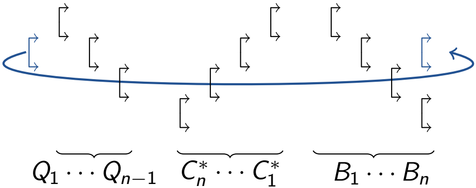
Similarity 1

# The Chase



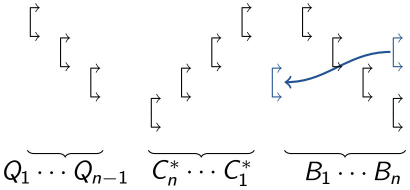
Similarity 1

# The Chase



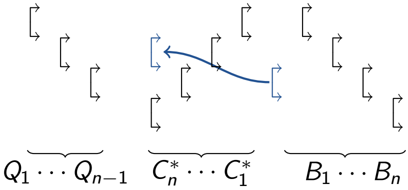

Similarity 2

# The Chase



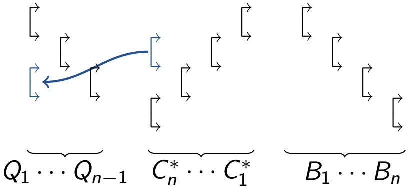

Similarity 2

# The Chase



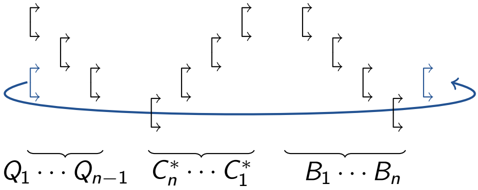

Similarity 2

# The Chase



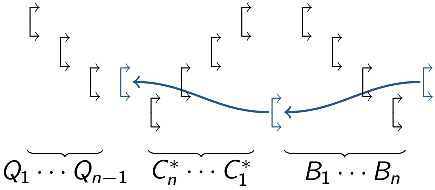

Similarity 2

# The Chase



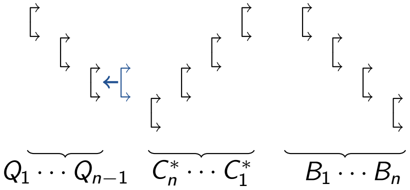

Similarity 3

# The Chase



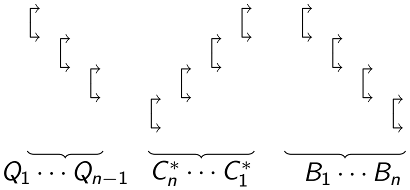

Similarity 3

# The Chase




Similarity 3

# The Chase




Similarity 3
We operate on a $5 \times 5$ matrix ($n = 4$), so it is fine.

- Iteration complete!
- Cost roughly $3n$ turnovers/iteration, so $O(n)$ flops/iteration.
- To the Schur form thus $O(n^2)$ operations.

# Outline

# Backward Stability

- Backward error on the Schur form:

$$Q^*(A + \Delta A)Q = S,$$

where

$$\|\Delta A\|_F \leq \|\text{coefficients of } p(x)\|^2 \ \mathcal{O}(\epsilon_m).$$

# Backward Stability

- Backward error on the Schur form:

$$Q^*(A + \Delta A)Q = S,$$

where

$$\|\Delta A\|_F \leq \|\text{coefficients of } p(x)\|^2 \; \mathcal{O}(\epsilon_m).$$

- Lapack (roots) does better here:

$$\|\Delta A\|_F \leq \|\text{coefficients of } p(x)\|^1 \; \mathcal{O}(\epsilon_m).$$

# Backward Stability (Version 1)

One step further, push the error to the polynomial coefficients:

- Following P. Dewilde and P. Van Dooren we must add another

$$\|\text{coefficients of } p(x)\|.$$

- So we would get:

$$\|\text{error on coefficients of } p(x)\| \quad \leq \quad \|\text{coefficients of } p(x)\|^{3}\, \mathcal{O}(\epsilon_m).$$

- Roots would get:

$$\|\text{error on coefficients of } p(x)\| \quad \leq \quad \|\text{coefficients of } p(x)\|^{2}\, \mathcal{O}(\epsilon_m).$$

# Backward Stability (Version 2 - 2 years later)

- Considering the structure in the perturbation:

$$A + \Delta A = U + \Delta U + uv^T + \Delta(uv^T)$$

we get

  - unitary part only perturbed by $\mathcal{O}(\epsilon_m)$,
  - rank one part (reconstruction) introduces errors of the order

$$\|\text{coefficients of } p(x)\|^2 \mathcal{O}(\epsilon_m)$$

.

- Because of this we get

$$\|\text{error on coefficients of } p(x)\| \leq \|\text{coefficients of } p(x)\|^2 \mathcal{O}(\epsilon_m).$$

- Yeah: we are as good as roots!

# Backward Stability (Version 3 - three years later)

- We were running tests for generalized companion matrices.
- This runs directly on non-monic polynomials and better accuracy expected.
- But experimentally no improvement was observed.

# Backward Stability (Version 3 - three years later)

- We were running tests for generalized companion matrices.
- This runs directly on non-monic polynomials and better accuracy expected.
- But experimentally no improvement was observed.
- We proved

$$\|\text{error on coefficients of } p(x)\| \;\leq\; \|\text{coefficients of } p(x)\|^{1}\,\mathcal{O}(\epsilon_m).$$

- Even when loosening monotonicity, lapack (or roots) gives

$$\|\text{error on coefficients of } p(x)\| \;\leq\; \|\text{coefficients of } p(x)\|^{2}\,\mathcal{O}(\epsilon_m).$$

# Backward Stability (Version 3 - three years later)

- We were running tests for generalized companion matrices.
- This runs directly on non-monic polynomials and better accuracy expected.
- But experimentally no improvement was observed.
- We proved

$$\|\text{error on coefficients of } p(x)\| \;\leq\; \|\text{coefficients of } p(x)\|^{1}\, \mathcal{O}(\epsilon_m).$$

- Even when loosening monotonicity, lapack (or roots) gives

$$\|\text{error on coefficients of } p(x)\| \;\leq\; \|\text{coefficients of } p(x)\|^{2}\, \mathcal{O}(\epsilon_m).$$

- Yeah$^2$.

# Outline

# Speed Comparison, Complex Case

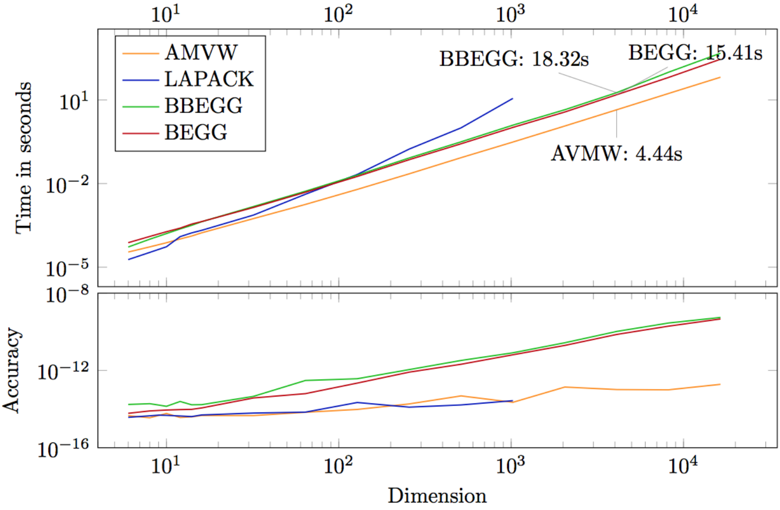
Contestants

- LAPACK code ZHSEQR ($O(n^3)$, unbalanced Hessenberg solver)
- BBEGG (Bini, Boito, Eidelman, Gemignani, and Gohberg 2010)
- BEGG (Boito, Eidelman, Gemignani, and Gohberg 2012)
- CGXZ (Chandrasekaran, Xia, Gu, and Zhu 2007)
- AMVW (Our single-shift or double-shift code)

Relative backward error measure

$$\max_{\lambda} \frac{\|Av - \lambda v\|}{\|A\|_\infty \|v\|_\infty}$$

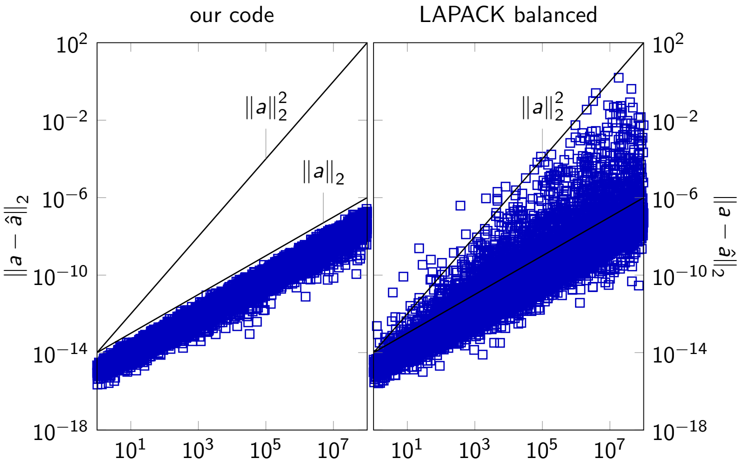
# Comparison, Complex Case




Note: our new implementation is even 25% faster.

# Outline

# Absolute Backward Error on Coefficients



# Conclusions & Comments

- Is this the best method for computing roots?
- Is this the best companion method?
- Better than normwise stability is component wise small error.
- Software part of EisCor (github).

# Conclusions


Thank You!