Core-Chasing Algorithms for the Eigenvalue Problem

David S. Watkins

Department of Mathematics
Washington State University

HHXX, Virginia Tech, June 20, 2017

Our International Research Group

Collaborators:

- **Jared Aurentz**
- **Thomas Mach**
- **Leonardo Robol**
- **Raf Vandebril**

Today's Topic

Today's Topic

- The matrix eigenvalue problem

Today's Topic

- The matrix eigenvalue problem
- $A \in \mathbb{C}^{n \times n}$

Today's Topic

- The matrix eigenvalue problem
- $A \in \mathbb{C}^{n \times n}$
- Find the eigenvalues (. . . vectors, invariant subspaces)

Today's Topic

- The matrix eigenvalue problem
- $A \in \mathbb{C}^{n \times n}$
- Find the eigenvalues (. . . vectors, invariant subspaces)
- Possible structures

Today's Topic

- The matrix eigenvalue problem
- $A \in \mathbb{C}^{n \times n}$
- Find the eigenvalues (. . . vectors, invariant subspaces)
- Possible structures
  - unitary
  - unitary-plus-rank-one (companion matrix)
  - unitary-plus-low-rank

Today's Topic

- The matrix eigenvalue problem
- $A \in \mathbb{C}^{n \times n}$
- Find the eigenvalues (. . . vectors, invariant subspaces)
- Possible structures
  - unitary
  - unitary-plus-rank-one (companion matrix)
  - unitary-plus-low-rank
  - . . . or no special structure

John Francis

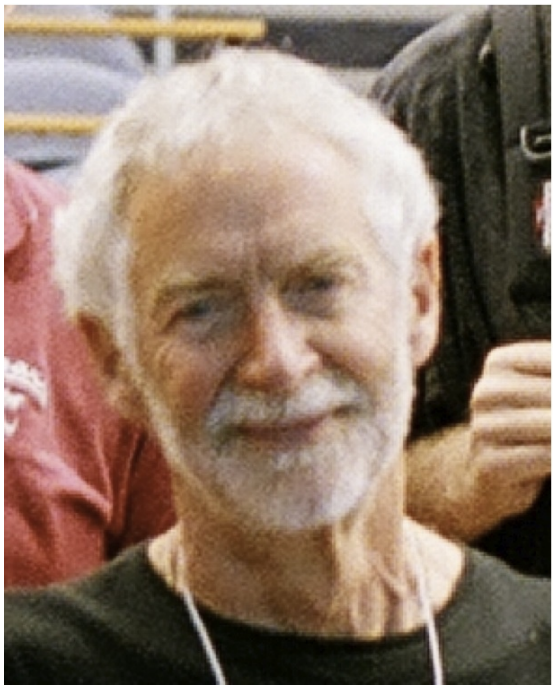


photo: Frank Uhlig, 2009

John Francis

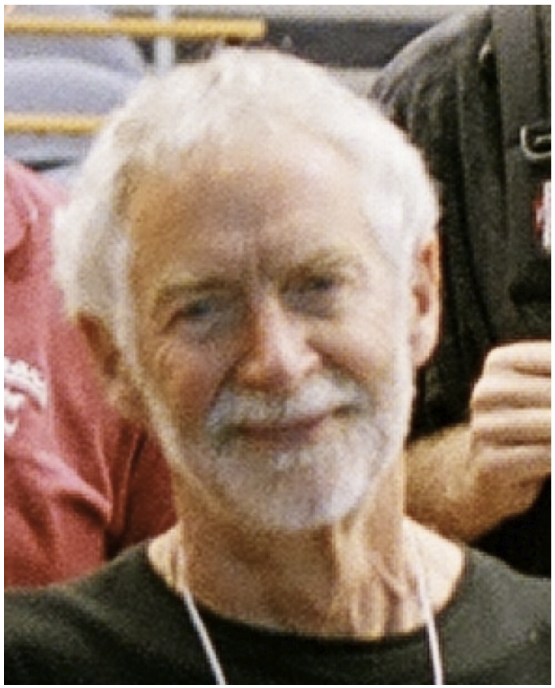


photo: Frank Uhlig, 2009

- invented the winning algorithm in 1959.

John Francis

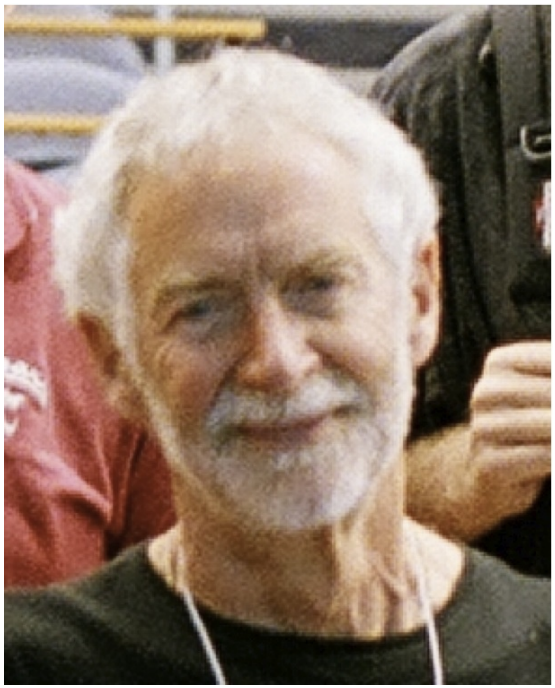


photo: Frank Uhlig, 2009

- invented the winning algorithm in 1959.
- implicitly shifted QR algorithm

John Francis

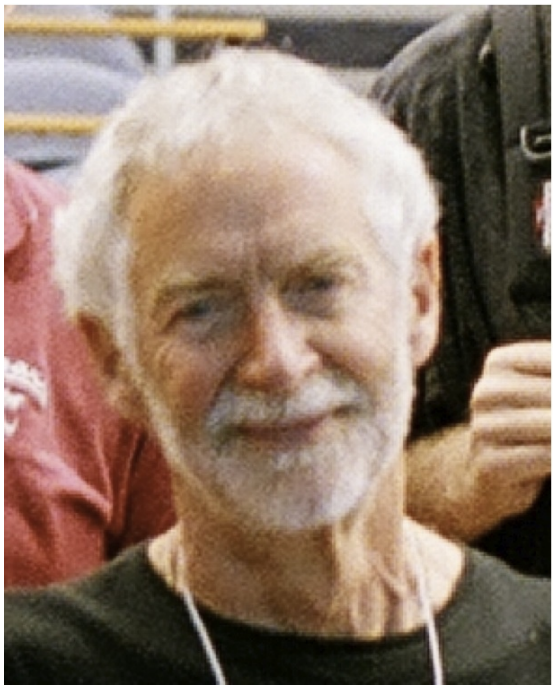


photo: Frank Uhlig, 2009

- invented the winning algorithm in 1959.
- implicitly shifted QR algorithm
- Our algorithms are all variants of this.

- Francis's algorithm . . .

- Francis's algorithm . . .
- . . . is a bulge chasing algorithm.

- Francis's algorithm . . .
- . . . is a bulge chasing algorithm.
- We turn it into a core chasing algorithm.

- Francis's algorithm . . .
- . . . is a bulge chasing algorithm.
- We turn it into a core chasing algorithm.
- Instead of chasing bulges, we chase core transformations.

Core Transformations

- What is a core transformation?

Core Transformations

- What is a core transformation?
- It's a unitary matrix, and

Core Transformations

- What is a core transformation?
- It's a unitary matrix, and
- it's essentially 2×2
- $C_2 = \begin{bmatrix} 1 & & & & \\ & \times & \times & & \\ & \times & \times & & \\ & & & 1 & \\ & & & & 1 \end{bmatrix}$

Core Transformations

- What is a core transformation?
- It's a unitary matrix, and
- it's essentially 2×2
- $C_2 = \begin{bmatrix} 1 & & & & \\ & \times & \times & & \\ & \times & \times & & \\ & & & 1 & \\ & & & & 1 \end{bmatrix}$
- Ex: Givens rotator, reflector, ...

Core Transformations

- What is a core transformation?
- It's a unitary matrix, and
- it's essentially 2×2
- $C_2 = \begin{bmatrix} 1 & & & & \\ & \times & \times & & \\ & \times & \times & & \\ & & & 1 & \\ & & & & 1 \end{bmatrix}$
- Ex: Givens rotator, reflector, ...
- We just wanted a generic term.

Core Transformations

$$\begin{bmatrix} 1 & & \\ & \times & \times \\ & \times & \times \end{bmatrix} \begin{bmatrix} \times & \times & \times \\ & \times & \times \\ & \times & \times \end{bmatrix} = \begin{bmatrix} \times & \times & \times \\ & \times & \times \\ & 0 & \times \end{bmatrix}$$

Core Transformations

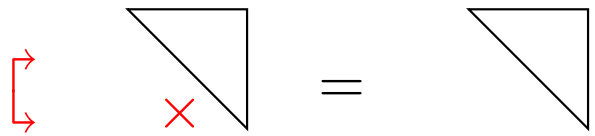
$$\begin{bmatrix} 1 & & \\ & \times & \times \\ & \times & \times \end{bmatrix} \begin{bmatrix} \times & \times & \times \\ & \times & \times \\ & \times & \times \end{bmatrix} = \begin{bmatrix} \times & \times & \times \\ & \times & \times \\ & 0 & \times \end{bmatrix}$$

Abbreviated notation

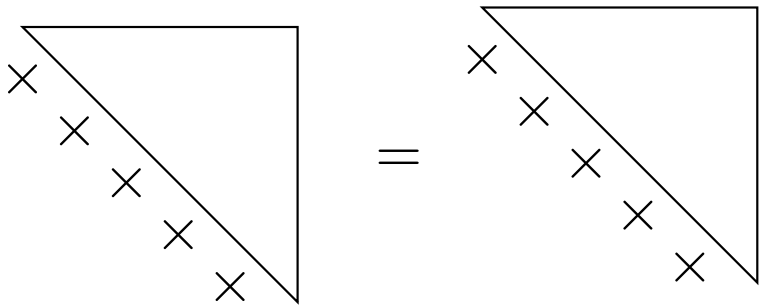
Core Transformations

$$\begin{bmatrix} 1 & & \\ & \times & \times \\ & \times & \times \end{bmatrix} \begin{bmatrix} \times & \times & \times \\ & \times & \times \\ & \times & \times \end{bmatrix} = \begin{bmatrix} \times & \times & \times \\ & \times & \times \\ & 0 & \times \end{bmatrix}$$

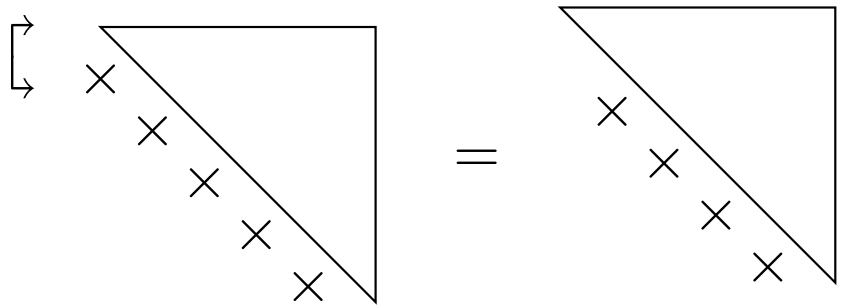
Abbreviated notation



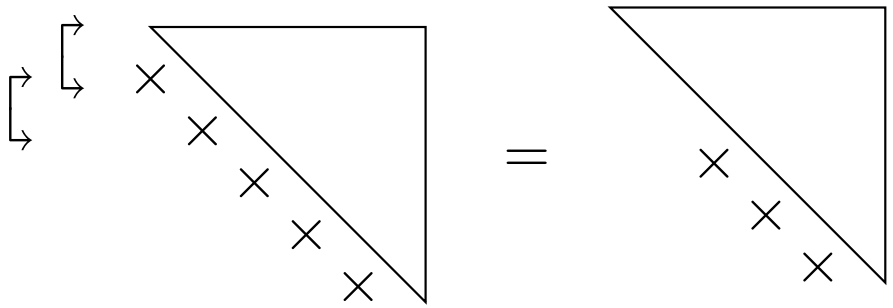
Hessenberg QR decomposition



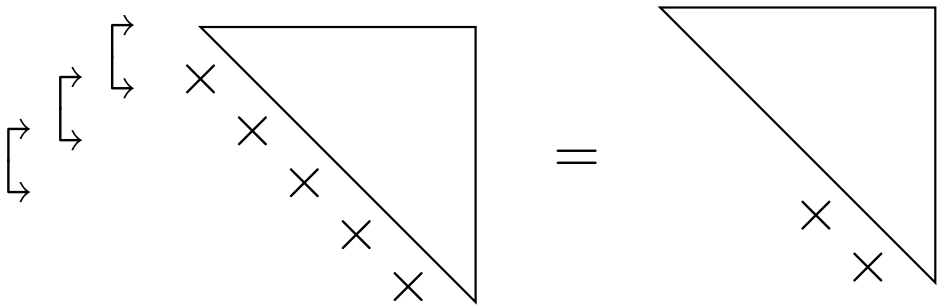
Hessenberg QR decomposition



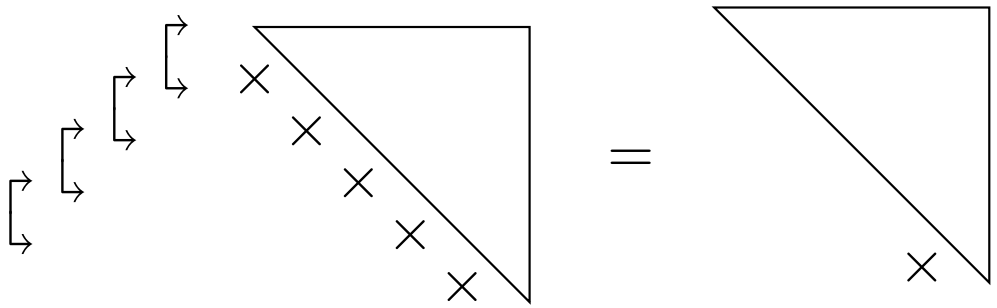
Hessenberg QR decomposition



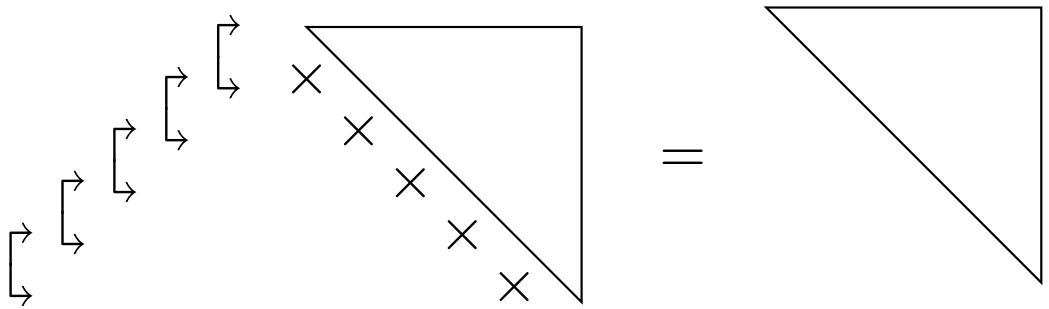
Hessenberg QR decomposition



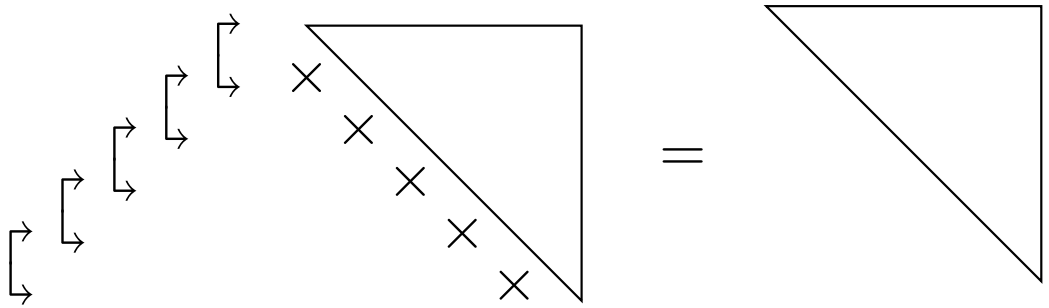
Hessenberg QR decomposition



Hessenberg QR decomposition

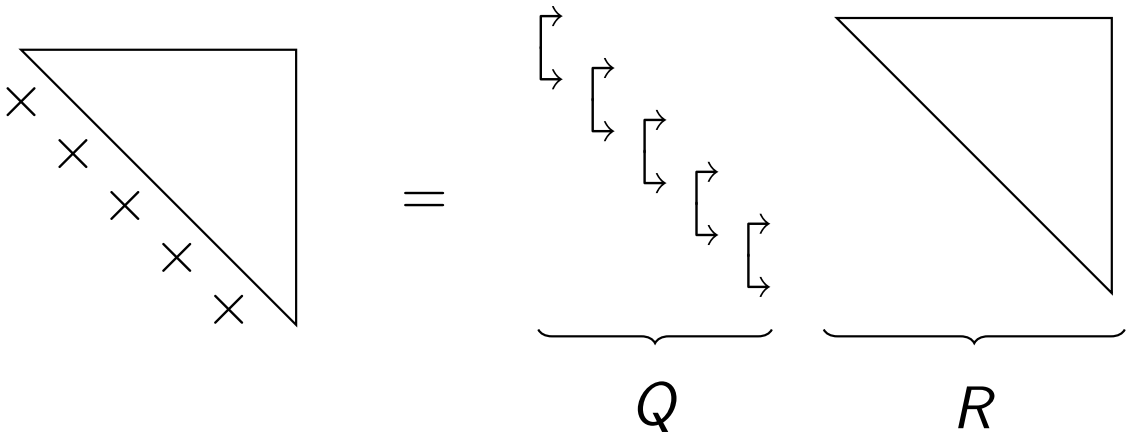


Hessenberg QR decomposition



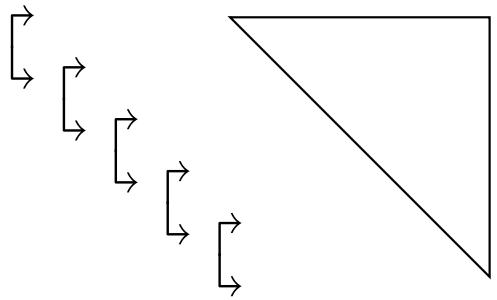
Now invert the core transformations.

Hessenberg QR decomposition



Our algorithms operate on the matrix in QR decomposed form.

$$A = QR =$$

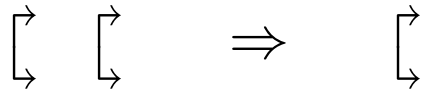


This is not inefficient.

We apply Francis's algorithm to this factored form.

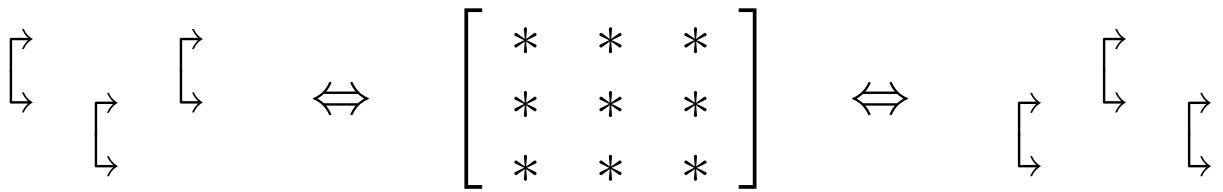
Operating on Core Transformations

Fusion



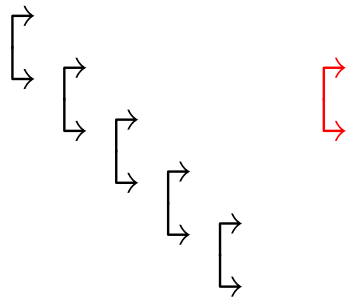
Operating on Core Transformations

Turnover



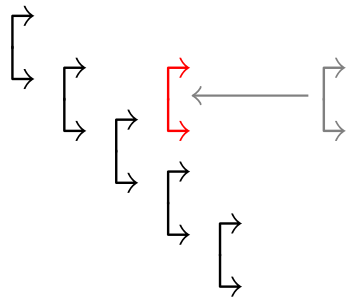
Operating on Core Transformations

Turnover as a shift-through operation



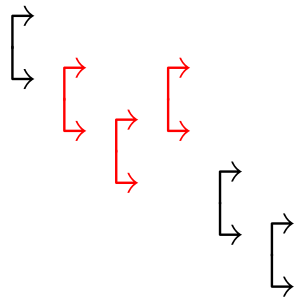
Operating on Core Transformations

Turnover as a shift-through operation



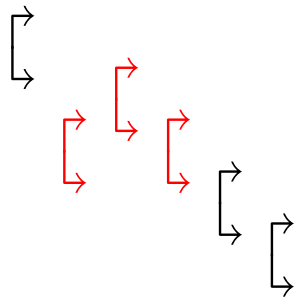
Operating on Core Transformations

Turnover as a shift-through operation



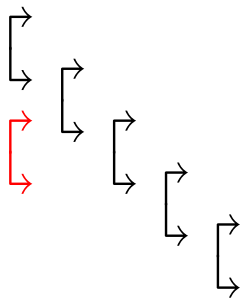
Operating on Core Transformations

Turnover as a shift-through operation



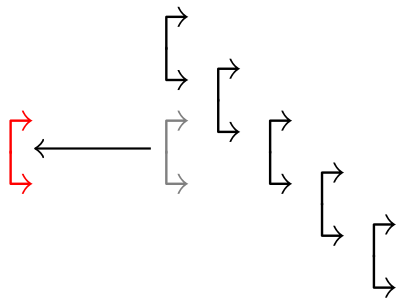
Operating on Core Transformations

Turnover as a shift-through operation



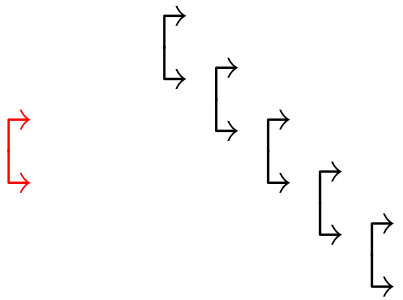
Operating on Core Transformations

Turnover as a shift-through operation



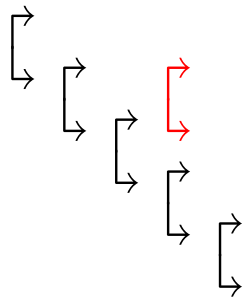
Operating on Core Transformations

Turnover as a shift-through operation



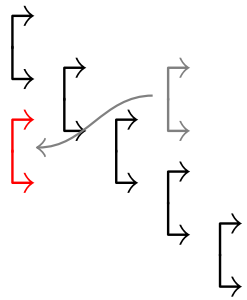
Operating on Core Transformations

Turnover as a shift-through operation : Abbreviated notation



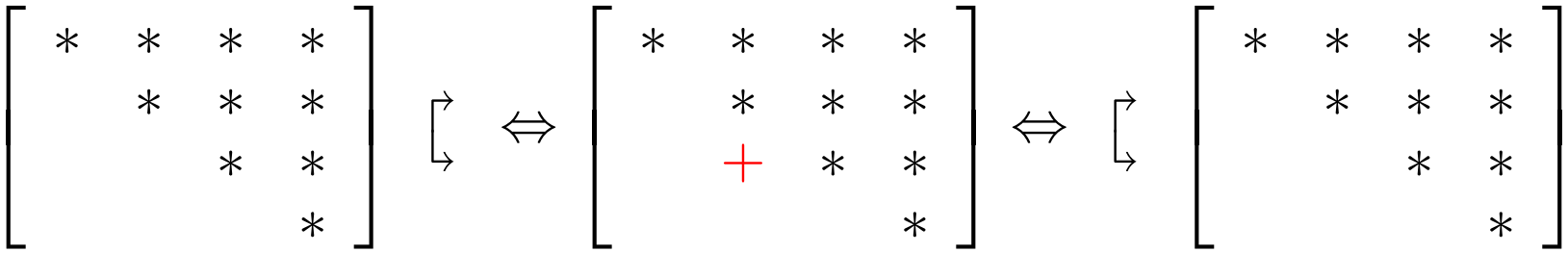
Operating on Core Transformations

Turnover as a shift-through operation : Abbreviated notation



Operating on Core Transformations

Passing a core transformation through a triangular matrix

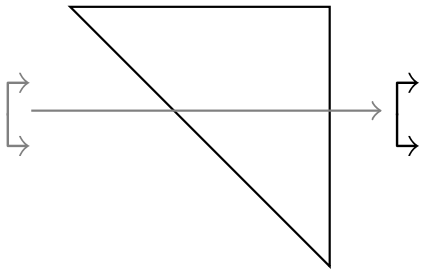


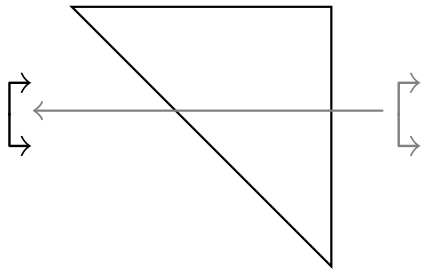
Cost is $O(n)$ flops.

Operating on Core Transformations

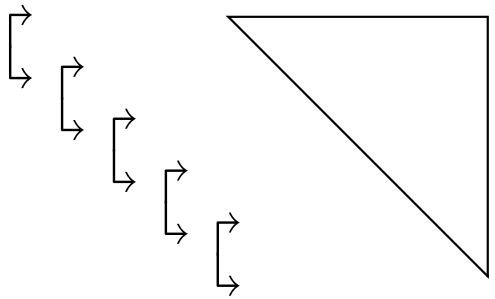
Passing a core transformation through a triangular matrix

Abbreviated Notation:

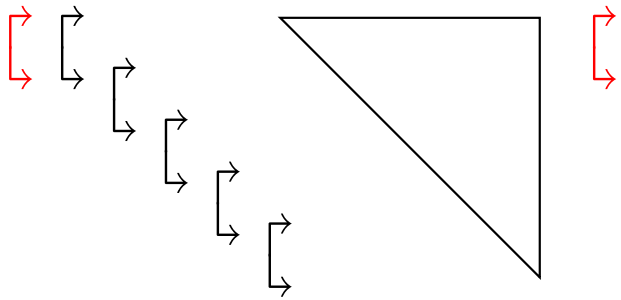




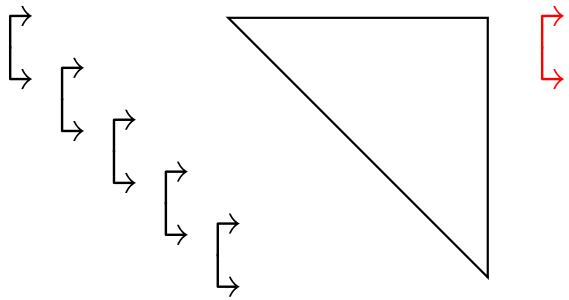
Core Chasing



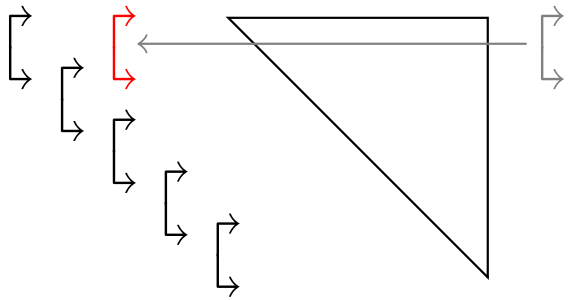
Core Chasing



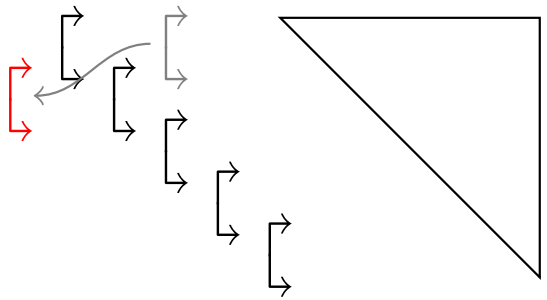
Core Chasing



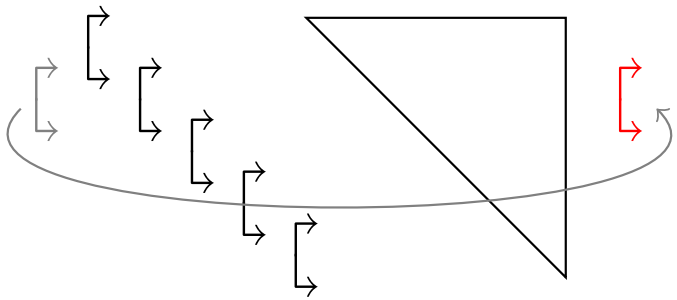
Core Chasing



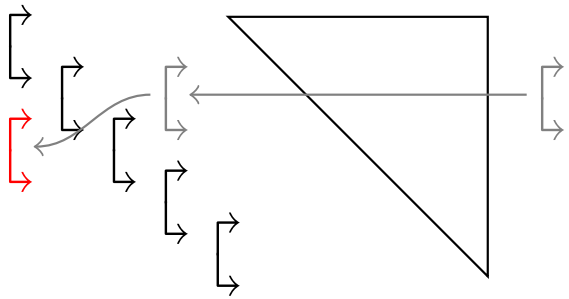
Core Chasing



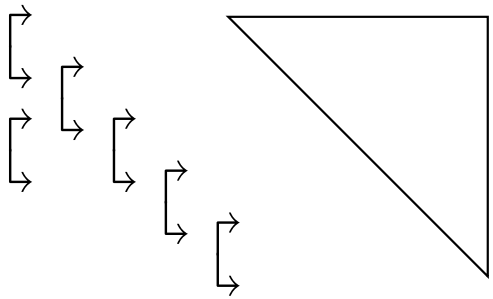
Core Chasing



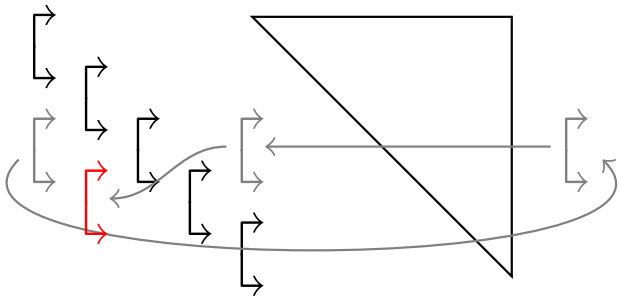
Core Chasing



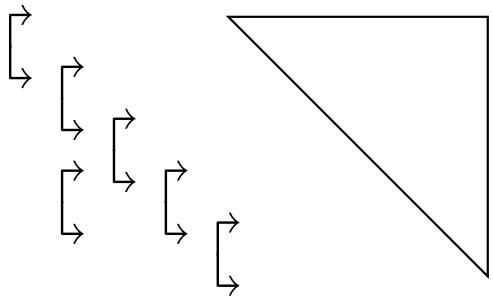
Core Chasing



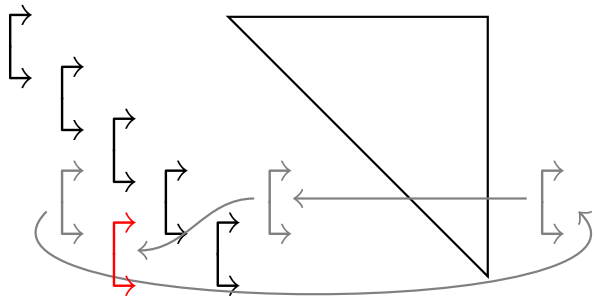
Core Chasing



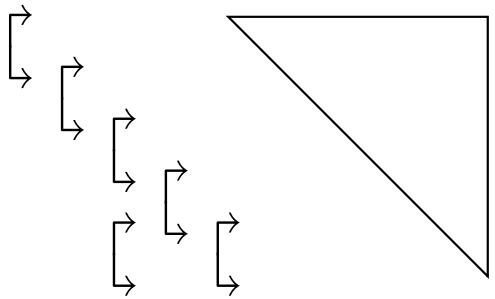
Core Chasing



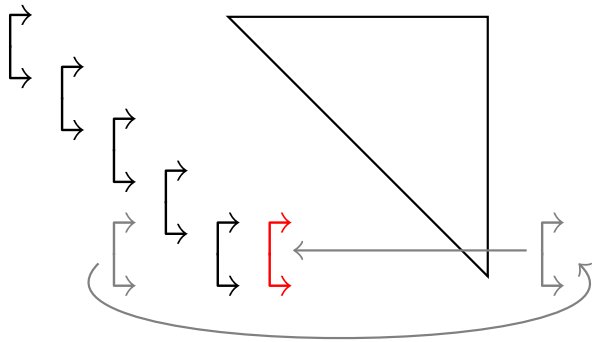
Core Chasing



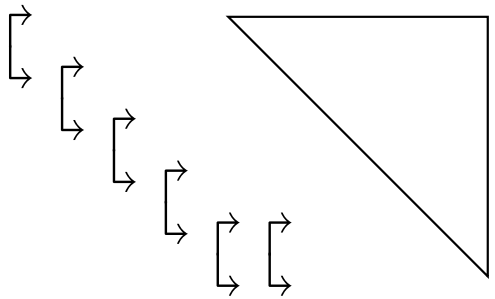
Core Chasing



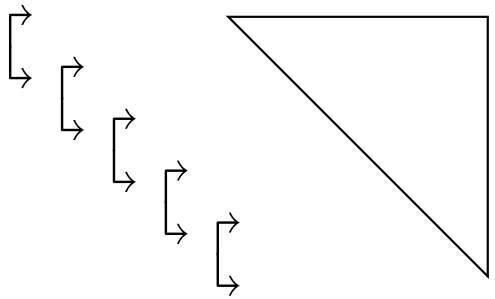
Core Chasing



Core Chasing



Core Chasing



Cost

Flop Count

Cost

- $O(n^3)$ total flops
- $O(n^2)$ storage
- about the same as for standard Francis iteration.

Advantages

Are there any advantages?

Advantages

Are there any advantages?

- superior deflation procedure

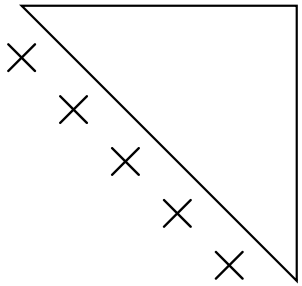
Advantages

Are there any advantages?

- superior deflation procedure
- some structured cases

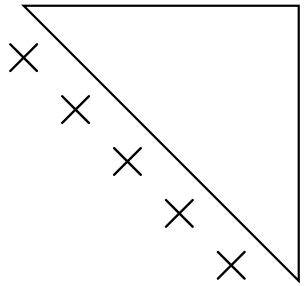
Deflation

Standard deflation criterion:



Deflation

Standard deflation criterion:



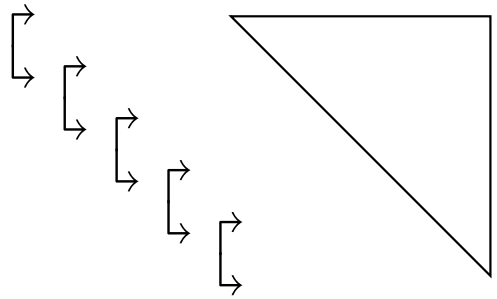
Set $a_{j+1,j}$ to zero if

$$|a_{j+1,j}| < u\,(|a_{j,j}| + |a_{j+1,j+1}|).$$

(u is *unit roundoff.*)

Deflation

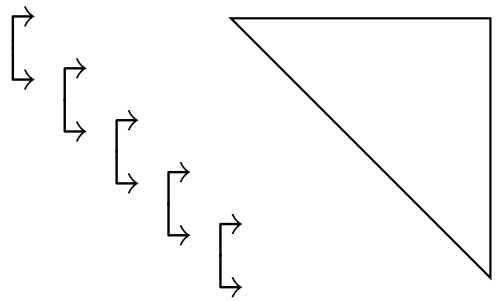
Our deflation criterion:



$$Q_j = \begin{bmatrix} I & & & \\ & c_j & -s_j & \\ & s_j & \overline{c}_j & \\ & & & I \end{bmatrix}$$

Deflation

Our deflation criterion:



$$Q_j = \begin{bmatrix} I & & & \\ & c_j & -s_j & \\ & s_j & \overline{c}_j & \\ & & & I \end{bmatrix}$$

Set s_j to zero if $\quad |s_j| < u.$

(u is *unit roundoff.*)

Deflation

- Both criteria are normwise backward stable.

Deflation

- Both criteria are normwise backward stable.
- How does this affect eigenvalues?

Deflation

- Both criteria are normwise backward stable.
- How does this affect eigenvalues?
- Change in λ depends on *condition number* $\kappa(\lambda)$.

Deflation

- Both criteria are normwise backward stable.
- How does this affect eigenvalues?
- Change in λ depends on *condition number* $\kappa(\lambda)$.
- Standard result: λ is perturbed to μ, where

$$|\lambda - \mu| \leq u\,\kappa(\lambda)\,\|A\| + O(u^2).$$

Deflation

- Both criteria are normwise backward stable.
- How does this affect eigenvalues?
- Change in λ depends on *condition number* $\kappa(\lambda)$.
- Standard result: λ is perturbed to μ, where

$$|\lambda - \mu| \leq u\,\kappa(\lambda)\,\|A\| + O(u^2).$$

- This holds for both deflation criteria.

Deflation

But our criterion does better:

But our criterion does better:

Theorem (Mach and Vandebril (2014))

$$|\lambda - \mu| \leq u\,\kappa(\lambda)\,|\lambda| + O(u^2).$$

Deflation

But our criterion does better:

Theorem (Mach and Vandebril (2014))

$$|\lambda - \mu| \leq u\,\kappa(\lambda)\,|\lambda| + O(u^2).$$

- **Relative** perturbation in each λ is tiny.

Deflation

But our criterion does better:

Theorem (Mach and Vandebril (2014))

$$|\lambda - \mu| \leq u\,\kappa(\lambda)\,|\lambda| + O(u^2).$$

- **Relative** perturbation in each λ is tiny.
- This *does not* hold for standard deflation criterion.

Deflation

Fun Example:

Deflation

Fun Example:

$$A = \begin{bmatrix} 1 & 2 \\ \epsilon & \epsilon \end{bmatrix} \qquad (0 < \epsilon < u)$$

Deflation

Fun Example:

$$A = \begin{bmatrix} 1 & 2 \\ \epsilon & \epsilon \end{bmatrix} \qquad (0 < \epsilon < u)$$

$$\lambda_1 = 1 + 2\epsilon + O(\epsilon^2) \qquad \lambda_2 = -\epsilon + O(\epsilon^2)$$

Deflation

Fun Example:

$$A = \begin{bmatrix} 1 & 2 \\ \epsilon & \epsilon \end{bmatrix} \qquad (0 < \epsilon < u)$$

$$\lambda_1 = 1 + 2\epsilon + O(\epsilon^2) \qquad \lambda_2 = -\epsilon + O(\epsilon^2)$$

These eigenvalues are well conditioned.

Deflation

Fun Example:

$$A = \begin{bmatrix} 1 & 2 \\ \epsilon & \epsilon \end{bmatrix} \qquad (0 < \epsilon < u)$$

$$\lambda_1 = 1 + 2\epsilon + O(\epsilon^2) \qquad \lambda_2 = -\epsilon + O(\epsilon^2)$$

These eigenvalues are well conditioned.

Standard criterion deflates to

$$\begin{bmatrix} 1 & 2 \\ 0 & \epsilon \end{bmatrix}.$$

Deflation

Fun Example:

$$A = \begin{bmatrix} 1 & 2 \\ \epsilon & \epsilon \end{bmatrix} \qquad (0 < \epsilon < u)$$

$$\lambda_1 = 1 + 2\epsilon + O(\epsilon^2) \qquad \lambda_2 = -\epsilon + O(\epsilon^2)$$

These eigenvalues are well conditioned.

Standard criterion deflates to

$$\begin{bmatrix} 1 & 2 \\ 0 & \epsilon \end{bmatrix}.$$

- Eigenvalues are $\mu_1 = 1$ and $\mu_2 = \epsilon$.

Deflation

Fun Example:

$$A = \begin{bmatrix} 1 & 2 \\ \epsilon & \epsilon \end{bmatrix} \qquad (0 < \epsilon < u)$$

$$\lambda_1 = 1 + 2\epsilon + O(\epsilon^2) \qquad \lambda_2 = -\epsilon + O(\epsilon^2)$$

These eigenvalues are well conditioned.

Standard criterion deflates to

$$\begin{bmatrix} 1 & 2 \\ 0 & \epsilon \end{bmatrix}.$$

- Eigenvalues are $\mu_1 = 1$ and $\mu_2 = \epsilon$.
- Small eigenvalue is off by 200%.

Deflation

Example, continued:

Deflation

Example, continued:

$$A = \begin{bmatrix} 1 & 2 \\ \epsilon & \epsilon \end{bmatrix} \qquad (0 < \epsilon < u)$$

$$\lambda_1 = 1 + 2\epsilon + O(\epsilon^2) \qquad \lambda_2 = -\epsilon + O(\epsilon^2)$$

Deflation

Example, continued:

$$A = \begin{bmatrix} 1 & 2 \\ \epsilon & \epsilon \end{bmatrix} \qquad (0 < \epsilon < u)$$

$$\lambda_1 = 1 + 2\epsilon + O(\epsilon^2) \qquad \lambda_2 = -\epsilon + O(\epsilon^2)$$

Our criterion:

$$A = QR \approx \begin{bmatrix} 1 & -\epsilon \\ \epsilon & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 0 & -\epsilon \end{bmatrix}.$$

Deflation

Example, continued:

$$A = \begin{bmatrix} 1 & 2 \\ \epsilon & \epsilon \end{bmatrix} \qquad (0 < \epsilon < u)$$

$$\lambda_1 = 1 + 2\epsilon + O(\epsilon^2) \qquad \lambda_2 = -\epsilon + O(\epsilon^2)$$

Our criterion:

$$A = QR \approx \begin{bmatrix} 1 & -\epsilon \\ \epsilon & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 0 & -\epsilon \end{bmatrix}.$$

Deflates to

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 0 & -\epsilon \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ 0 & -\epsilon \end{bmatrix}.$$

Deflation

Example, continued:

$$A = \begin{bmatrix} 1 & 2 \\ \epsilon & \epsilon \end{bmatrix} \qquad (0 < \epsilon < u)$$

$$\lambda_1 = 1 + 2\epsilon + O(\epsilon^2) \qquad \lambda_2 = -\epsilon + O(\epsilon^2)$$

Our criterion:

$$A = QR \approx \begin{bmatrix} 1 & -\epsilon \\ \epsilon & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 0 & -\epsilon \end{bmatrix}.$$

Deflates to

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 0 & -\epsilon \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ 0 & -\epsilon \end{bmatrix}.$$

- Eigenvalues are $\mu_1 = 1$ and $\mu_2 = -\epsilon$.

Deflation

Example, continued:

$$A = \begin{bmatrix} 1 & 2 \\ \epsilon & \epsilon \end{bmatrix} \qquad (0 < \epsilon < u)$$

$$\lambda_1 = 1 + 2\epsilon + O(\epsilon^2) \qquad \lambda_2 = -\epsilon + O(\epsilon^2)$$

Our criterion:

$$A = QR \approx \begin{bmatrix} 1 & -\epsilon \\ \epsilon & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 0 & -\epsilon \end{bmatrix}.$$

Deflates to

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 0 & -\epsilon \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ 0 & -\epsilon \end{bmatrix}.$$

- Eigenvalues are $\mu_1 = 1$ and $\mu_2 = -\epsilon$.
- Both eigenvalues are accurate.

Exploitation of Structure

Structures we can exploit

Exploitation of Structure

Structures we can exploit

- unitary

Exploitation of Structure

Structures we can exploit

- unitary
- companion matrix (unitary-plus-rank-one)

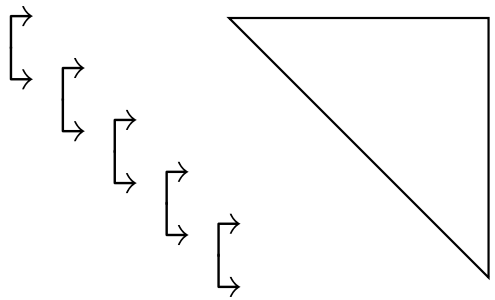
Exploitation of Structure

Structures we can exploit

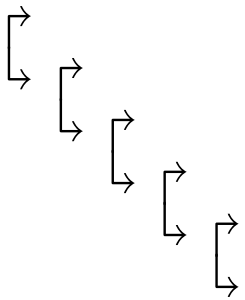
- unitary
- companion matrix (unitary-plus-rank-one)
- unitary-plus-low-rank

Unitary Case

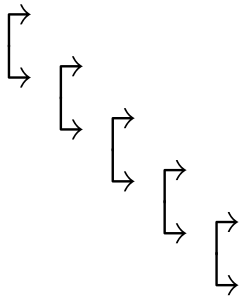
Unitary Case

$A = QR =$ 

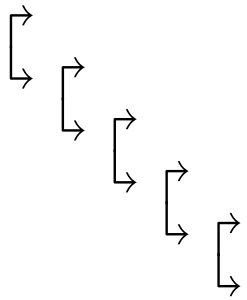
Unitary Case

$A = QR =$ 

Unitary Case

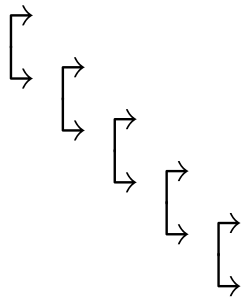
$A = QR =$ 

Unitary Case

$A = QR =$ 

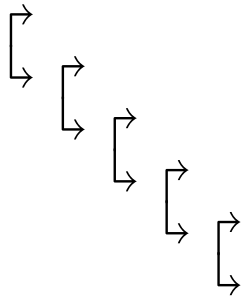
- Cost is $O(n)$ flops per iteration,

Unitary Case

$A = QR =$ 

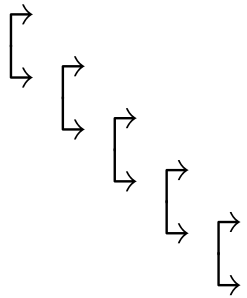
- Cost is $O(n)$ flops per iteration, $O(n^2)$ flops total.

Unitary Case

$A = QR =$ 

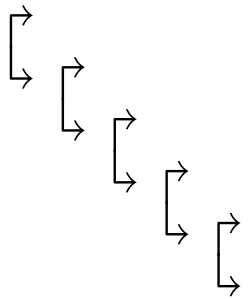
- Cost is $O(n)$ flops per iteration, $O(n^2)$ flops total.
- Storage requirement is $O(n)$.

Unitary Case

$A = QR =$ 

- Cost is $O(n)$ flops per iteration, $O(n^2)$ flops total.
- Storage requirement is $O(n)$.
- Gragg (1986)

Unitary Case

$A = QR =$ 

- Cost is $O(n)$ flops per iteration, $O(n^2)$ flops total.
- Storage requirement is $O(n)$.
- Gragg (1986)
- Ammar, Reichel, M. Stewart, Bunse-Gerstner, Elsner, He, W, ...

Companion Case

Companion Case

- $p(x) = x^n + a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \cdots + a_0 = 0$
- monic polynomial

Companion Case

- $p(x) = x^n + a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \cdots + a_0 = 0$
- monic polynomial
- companion matrix

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

- . . . get the zeros of p by computing the eigenvalues.

Companion Case

- $p(x) = x^n + a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \cdots + a_0 = 0$
- monic polynomial
- companion matrix

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

- . . . get the zeros of p by computing the eigenvalues.
- MATLAB's `roots` command

Companion Case

- $p(x) = x^n + a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \cdots + a_0 = 0$
- monic polynomial
- companion matrix

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

- . . . get the zeros of p by computing the eigenvalues.
- MATLAB's `roots` command
- Companion matrix is unitary-plus-rank-one.

Cost of solving companion eigenvalue problem

Cost of solving companion eigenvalue problem

- If structure not exploited:
  - $O(n^2)$ storage, $O(n^3)$ flops
  - Francis's algorithm

Cost of solving companion eigenvalue problem

- If structure not exploited:
  - $O(n^2)$ storage, $O(n^3)$ flops
  - Francis's algorithm
- If structure exploited:
  - $O(n)$ storage, $O(n^2)$ flops

Cost of solving companion eigenvalue problem

- If structure not exploited:
  - $O(n^2)$ storage, $O(n^3)$ flops
  - Francis's algorithm
- If structure exploited:
  - $O(n)$ storage, $O(n^2)$ flops
  - several methods proposed
  - data-sparse representation + Francis's algorithm

Cost of solving companion eigenvalue problem

- If structure not exploited:
  - $O(n^2)$ storage, $O(n^3)$ flops
  - Francis's algorithm
- If structure exploited:
  - $O(n)$ storage, $O(n^2)$ flops
  - several methods proposed
  - data-sparse representation + Francis's algorithm
  - Ours is fastest . . .

Cost of solving companion eigenvalue problem

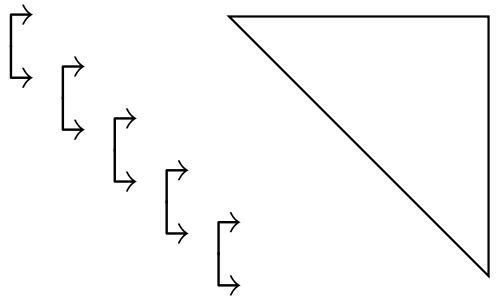
- If structure not exploited:
  - $O(n^2)$ storage, $O(n^3)$ flops
  - Francis's algorithm
- If structure exploited:
  - $O(n)$ storage, $O(n^2)$ flops
  - several methods proposed
  - data-sparse representation + Francis's algorithm
  - Ours is fastest . . .
  - . . . and we can prove backward stability.

Cost of solving companion eigenvalue problem

- If structure not exploited:
  - $O(n^2)$ storage, $O(n^3)$ flops
  - Francis's algorithm
- If structure exploited:
  - $O(n)$ storage, $O(n^2)$ flops
  - several methods proposed
  - data-sparse representation + Francis's algorithm
  - Ours is fastest . . .
  - . . . and we can prove backward stability.
  - I spoke about this at the previous Householder symposium.

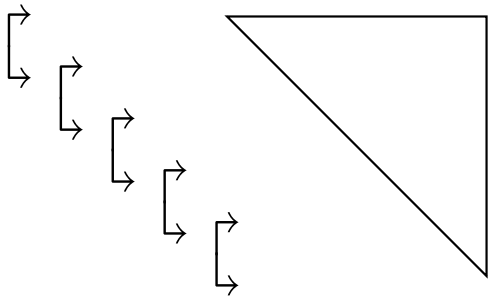
Representation of R

We store the QR decomposed form.

$A = QR =$ 

Representation of R

We store the QR decomposed form.

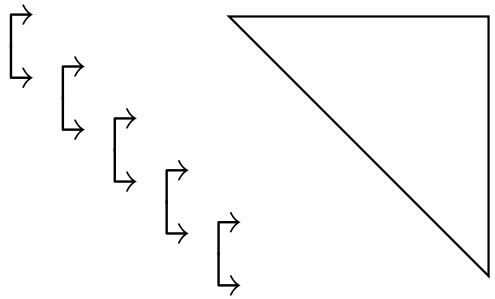
$$A = QR = \quad$$ 

where

$$R = \begin{bmatrix} 1 & 0 & \cdots & -a_1 \\ & 1 & & -a_2 \\ & & \ddots & \vdots \\ & & & -a_0 \end{bmatrix}.$$

Representation of R

We store the QR decomposed form.

$$A = QR = \quad$$ 

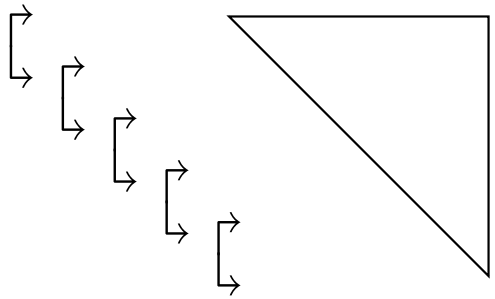
where

$$R = \begin{bmatrix} 1 & 0 & \cdots & -a_1 \\ & 1 & & -a_2 \\ & & \ddots & \vdots \\ & & & -a_0 \end{bmatrix}.$$

- This is unitary-plus-rank-one.

Representation of R

We store the QR decomposed form.

$$A = QR = \quad$$ 

where

$$R = \begin{bmatrix} 1 & 0 & \cdots & -a_1 \\ & 1 & & -a_2 \\ & & \ddots & \vdots \\ & & & -a_0 \end{bmatrix}.$$

- This is unitary-plus-rank-one.
- How do we store it?

Representation of R

- Add a row and column to R.

Representation of R

- Add a row and column to R.

$$\tilde{R} = \left[\begin{array}{cccc|c} 1 & 0 & \cdots & -a_1 & 0 \\ & 1 & & -a_2 & 0 \\ & & \ddots & \vdots & \vdots \\ & & & -a_0 & 1 \\ \hline 0 & 0 & \cdots & 0 & 0 \end{array}\right].$$

Representation of R

- Add a row and column to R.

$$\tilde{R} = \left[\begin{array}{cccc|c} 1 & 0 & \cdots & -a_1 & 0 \\ & 1 & & -a_2 & 0 \\ & & \ddots & \vdots & \vdots \\ & & & -a_0 & 1 \\ \hline 0 & 0 & \cdots & 0 & 0 \end{array}\right].$$

- This is still unitary-plus-rank-one.

Representation of R

$$\tilde{R} = \left[\begin{array}{cccc|c} 1 & 0 & \cdots & 0 & 0 \\ & 1 & & 0 & 0 \\ & & \ddots & \vdots & \vdots \\ & & & 0 & 1 \\ \hline 0 & 0 & \cdots & 1 & 0 \end{array}\right] + \left[\begin{array}{cccc|c} 0 & 0 & \cdots & -a_1 & 0 \\ & 0 & & -a_2 & 0 \\ & & \ddots & \vdots & \vdots \\ & & & -a_0 & 0 \\ \hline 0 & 0 & \cdots & -1 & 0 \end{array}\right].$$

Representation of R

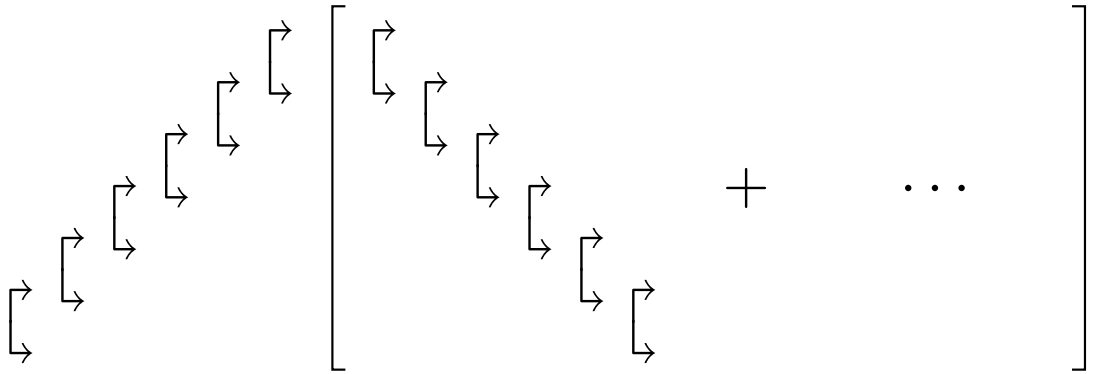
$$\tilde{R} =$$

Representation of R

$$\tilde{R} = C_n^* \cdots C_1^* (B_1 \cdots B_n + e_1 y^T)$$

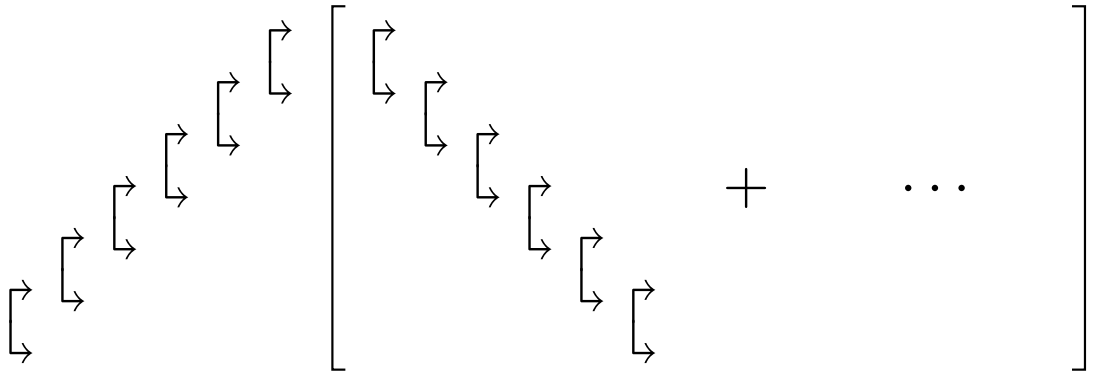
Representation of R

$$\tilde{R} = C_n^* \cdots C_1^* (B_1 \cdots B_n + e_1 y^T)$$



Representation of R

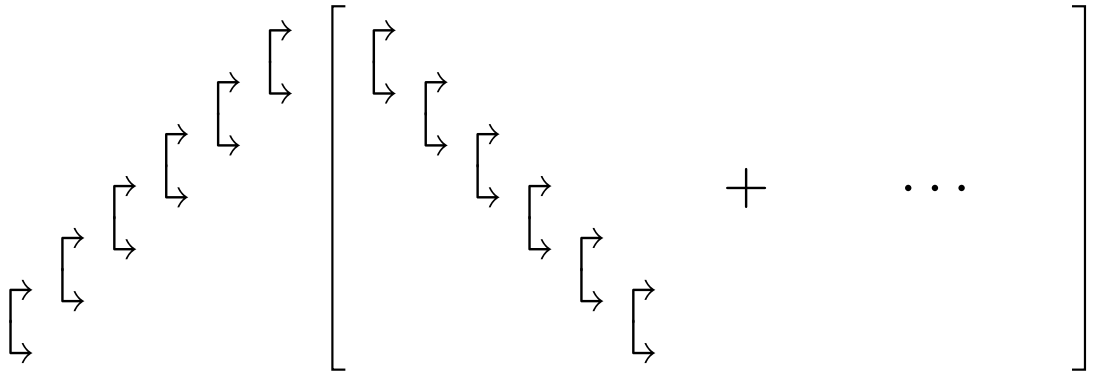
$$\tilde{R} = C_n^* \cdots C_1^* (B_1 \cdots B_n + e_1 y^T)$$



. . . and we don't have to store the rank-one part!

Representation of R

$$\tilde{R} = C_n^* \cdots C_1^* (B_1 \cdots B_n + e_1 y^T)$$

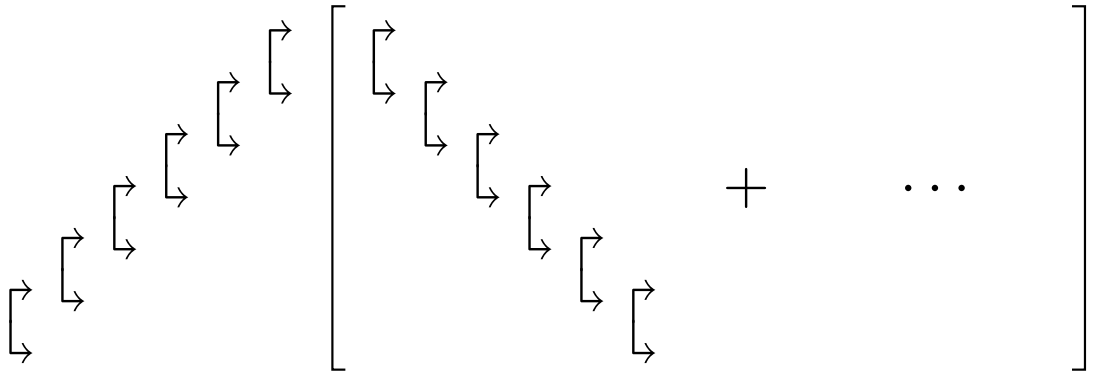


. . . and we don't have to store the rank-one part!

- This helps with backward stability.

Representation of R

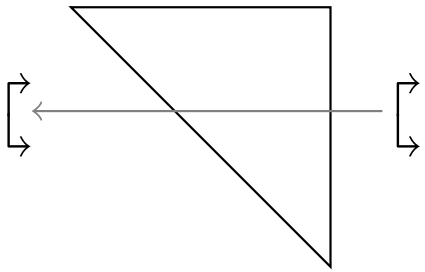
$$\tilde{R} = C_n^* \cdots C_1^* (B_1 \cdots B_n + e_1 y^T)$$



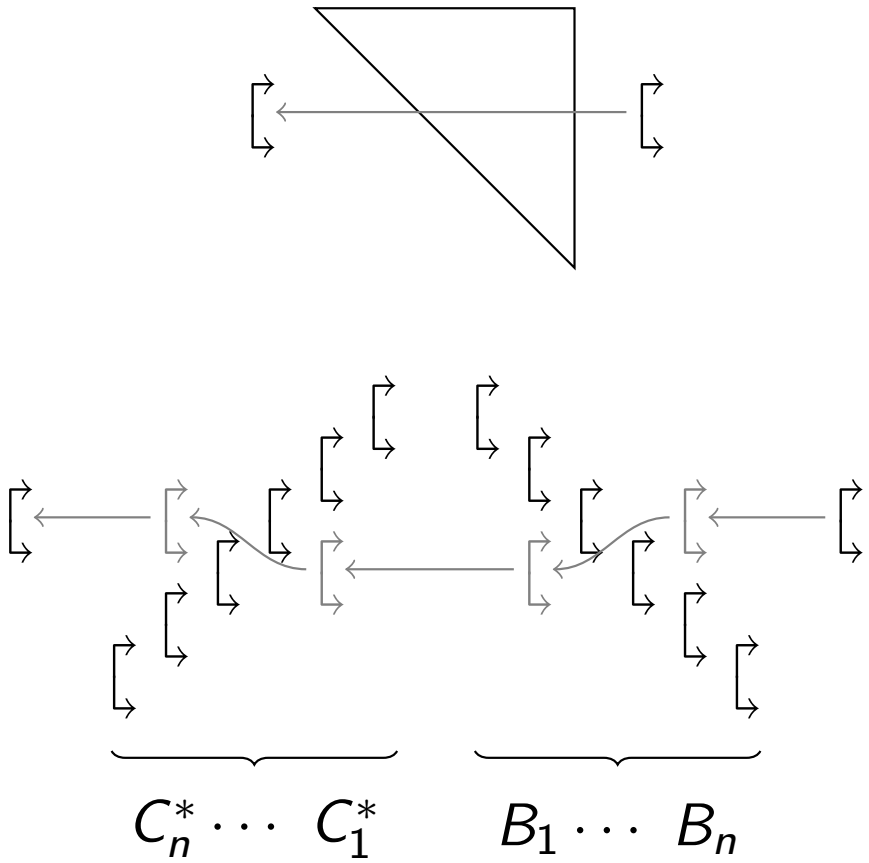
. . . and we don't have to store the rank-one part!

- This helps with backward stability.
- Storage is $O(n)$.

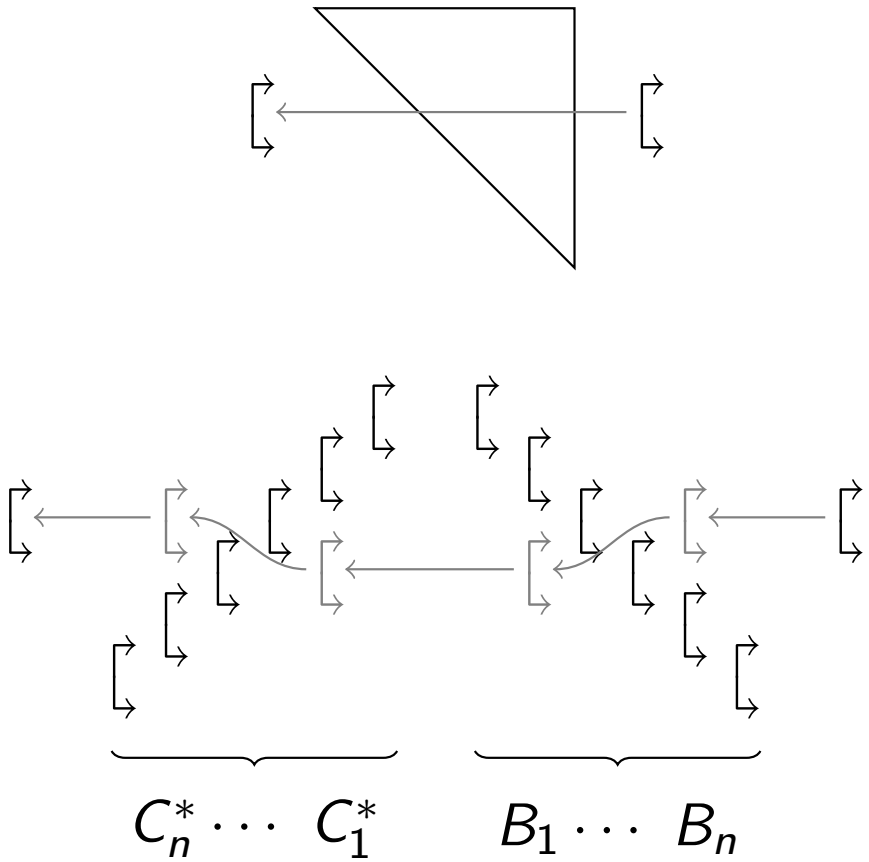
Passing a core transformation through R



Passing a core transformation through R



Passing a core transformation through R



Cost: $O(1)$ flops instead of $O(n)$.

Other things we can do

We can also handle

Other things we can do

We can also handle

- generalized eigenvalue problem
- companion pencil

Other things we can do

We can also handle

- generalized eigenvalue problem
- companion pencil
- matrix polynomial eigenvalue problems
  - L. Robol talk at 2 pm

Other things we can do

We can also handle

- generalized eigenvalue problem
- companion pencil
- matrix polynomial eigenvalue problems
  - L. Robol talk at 2 pm
- generalizations of Hessenberg form

Other things we can do

We can also handle

- generalized eigenvalue problem
- companion pencil
- matrix polynomial eigenvalue problems
  - L. Robol talk at 2 pm
- generalizations of Hessenberg form
- and more.

Other things we can do

We can also handle

- generalized eigenvalue problem
- companion pencil
- matrix polynomial eigenvalue problems
  - L. Robol talk at 2 pm
- generalizations of Hessenberg form
- and more.

Monograph in progress (130+ pp.)

The Companion Pencil

- $p(x) = a_0 + a_1 x + \cdots + a_n x^n$

The Companion Pencil

- $p(x) = a_0 + a_1 x + \cdots + a_n x^n$ (not monic)

The Companion Pencil

- $p(x) = a_0 + a_1 x + \cdots + a_n x^n$ (not monic)
- Divide by a_n,

The Companion Pencil

- $p(x) = a_0 + a_1 x + \cdots + a_n x^n$ (not monic)
- Divide by a_n, or ...

The Companion Pencil

- $p(x) = a_0 + a_1 x + \cdots + a_n x^n$ (not monic)
- Divide by a_n, or ...
- companion pencil:

$$\lambda \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & \\ & & & & a_n \end{bmatrix} - \begin{bmatrix} 0 & & \cdots & 0 & -a_0 \\ 1 & 0 & \cdots & 0 & -a_1 \\ & 1 & \ddots & \vdots & \vdots \\ & & \ddots & 0 & -a_{n-2} \\ & & & 1 & -a_{n-1} \end{bmatrix}$$

The Companion Pencil

- $p(x) = a_0 + a_1 x + \cdots + a_n x^n$ (not monic)
- Divide by a_n, or ...
- companion pencil:

$$\lambda \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & \\ & & & & a_n \end{bmatrix} - \begin{bmatrix} 0 & & \cdots & 0 & -a_0 \\ 1 & 0 & \cdots & 0 & -a_1 \\ & 1 & \ddots & \vdots & \vdots \\ & & \ddots & 0 & -a_{n-2} \\ & & & 1 & -a_{n-1} \end{bmatrix}$$

- We can handle this too (for a price),

The Companion Pencil

- $p(x) = a_0 + a_1 x + \cdots + a_n x^n$ (not monic)
- Divide by a_n, or ...
- companion pencil:

$$\lambda \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & \\ & & & & a_n \end{bmatrix} - \begin{bmatrix} 0 & & \cdots & 0 & -a_0 \\ 1 & 0 & \cdots & 0 & -a_1 \\ & 1 & \ddots & \vdots & \vdots \\ & & \ddots & 0 & -a_{n-2} \\ & & & 1 & -a_{n-1} \end{bmatrix}$$

- We can handle this too (for a price),
- This should be superior in some situations.
- e.g. if a_n is tiny.

Backward Stability

Backward Stability

- All of these algorithms are normwise backward stable.

Backward Stability

- All of these algorithms are normwise backward stable.
- This is "obvious" because we work only with unitary transformations,

Backward Stability

- All of these algorithms are normwise backward stable.
- This is "obvious" because we work only with unitary transformations,
- but it took us a while to write down a correct proof.

Backward Stability

- All of these algorithms are normwise backward stable.
- This is "obvious" because we work only with unitary transformations,
- but it took us a while to write down a correct proof.
- For details see . . .

Backward Stability

- All of these algorithms are normwise backward stable.
- This is "obvious" because we work only with unitary transformations,
- but it took us a while to write down a correct proof.
- For details see . . .
- 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), pp. 942–973.

Backward Stability

- All of these algorithms are normwise backward stable.
- This is "obvious" because we work only with unitary transformations,
- but it took us a while to write down a correct proof.
- For details see . . .
- 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), pp. 942–973.
- Co-winner of SIAM Best Paper Prize (2017).

Backward Stability

- All of these algorithms are normwise backward stable.
- This is "obvious" because we work only with unitary transformations,
- but it took us a while to write down a correct proof.
- For details see . . .
- 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), pp. 942–973.
- Co-winner of SIAM Best Paper Prize (2017).
- Jared L. Aurentz, Thomas Mach, Leonardo Robol, Raf Vandebril, and David S. Watkins, *Roots of polynomials: on twisted QR methods for companion matrices and pencils,* arXiv:1611.02435,

Backward Stability

- All of these algorithms are normwise backward stable.
- This is "obvious" because we work only with unitary transformations,
- but it took us a while to write down a correct proof.
- For details see . . .
- 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), pp. 942–973.
- Co-winner of SIAM Best Paper Prize (2017).
- Jared L. Aurentz, Thomas Mach, Leonardo Robol, Raf Vandebril, and David S. Watkins, *Roots of polynomials: on twisted QR methods for companion matrices and pencils,* arXiv:1611.02435, currently undergoing a complete rewrite.

Backward Stability Odyssey

Backward Stability Odyssey

- Presentation at Householder 2014

Backward Stability Odyssey

- Presentation at Householder 2014
- First written attempt (horrible)

Backward Stability Odyssey

- Presentation at Householder 2014
- First written attempt (horrible)
- Second attempt was much better (2015 paper) . . .

Backward Stability Odyssey

- Presentation at Householder 2014
- First written attempt (horrible)
- Second attempt was much better (2015 paper) . . .
- . . . but there was one one more thing!

Backward Stability Odyssey

- Presentation at Householder 2014
- First written attempt (horrible)
- Second attempt was much better (2015 paper) . . .
- . . . but there was one one more thing!
- Corrected in companion pencil paper.

Backward Stability Odyssey

- Presentation at Householder 2014
- First written attempt (horrible)
- Second attempt was much better (2015 paper) . . .
- . . . but there was one one more thing!
- Corrected in companion pencil paper. We also exploited the structure of the backward error to get a better result.

Backward Stability Odyssey

- Presentation at Householder 2014
- First written attempt (horrible)
- Second attempt was much better (2015 paper) . . .
- . . . but there was one one more thing!
- Corrected in companion pencil paper. We also exploited the structure of the backward error to get a better result. Rejected!

Backward Stability Odyssey

- Presentation at Householder 2014
- First written attempt (horrible)
- Second attempt was much better (2015 paper) . . .
- . . . but there was one one more thing!
- Corrected in companion pencil paper. We also exploited the structure of the backward error to get a better result. Rejected!
- Search for examples.

Backward Stability Odyssey

- Presentation at Householder 2014
- First written attempt (horrible)
- Second attempt was much better (2015 paper) . . .
- . . . but there was one one more thing!
- Corrected in companion pencil paper. We also exploited the structure of the backward error to get a better result. Rejected!
- Search for examples.
- Take a closer look.

Backward Stability Odyssey

- Presentation at Householder 2014
- First written attempt (horrible)
- Second attempt was much better (2015 paper) . . .
- . . . but there was one one more thing!
- Corrected in companion pencil paper. We also exploited the structure of the backward error to get a better result. Rejected!
- Search for examples.
- Take a closer look.
- backward error on pencil vs. polynomial coefficients

Backward Stability Odyssey

- Presentation at Householder 2014
- First written attempt (horrible)
- Second attempt was much better (2015 paper) . . .
- . . . but there was one one more thing!
- Corrected in companion pencil paper. We also exploited the structure of the backward error to get a better result. Rejected!
- Search for examples.
- Take a closer look.
- backward error on pencil vs. polynomial coefficients
- monic vs. scaled polynomial

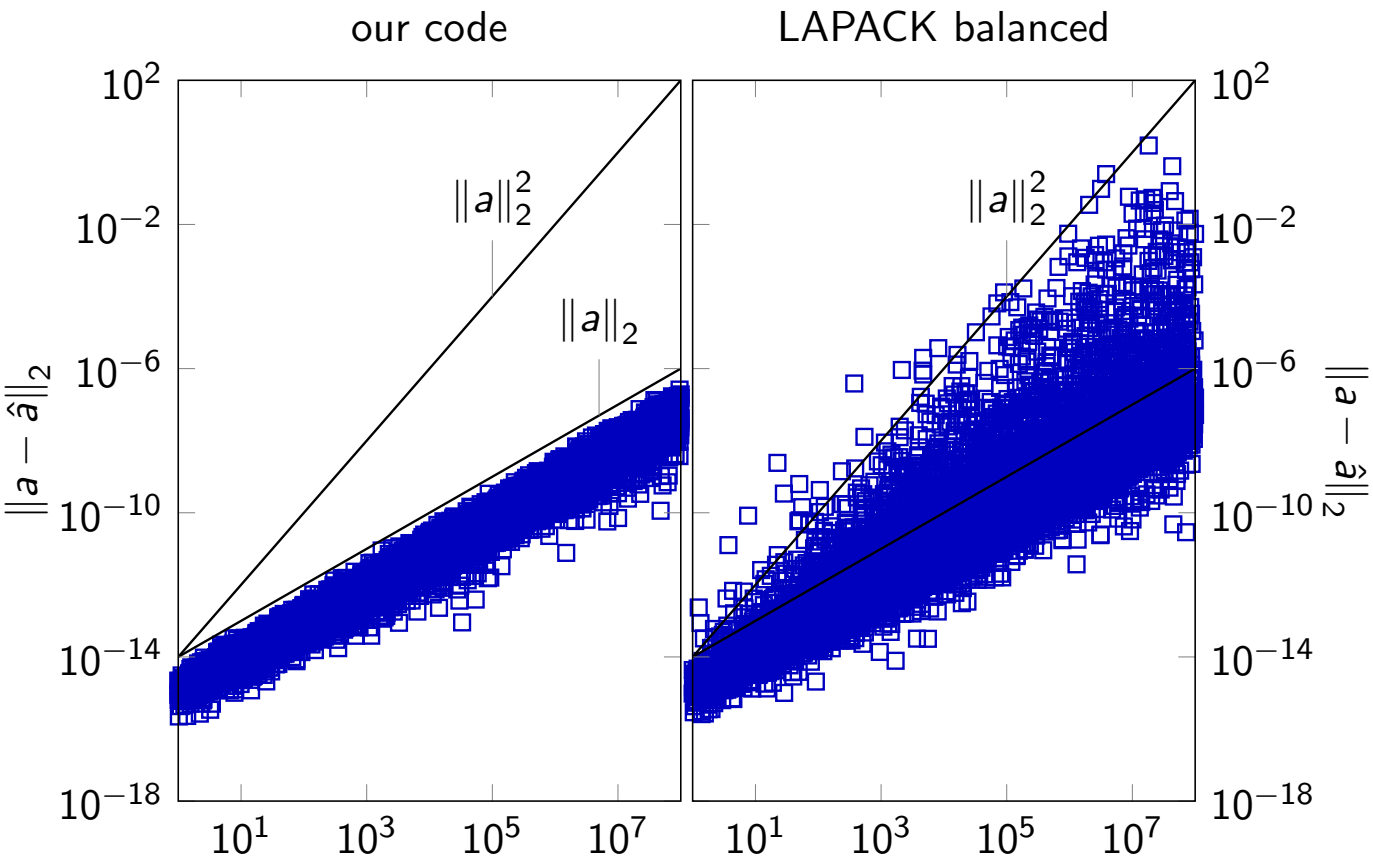
Backward Stability Odyssey

- Presentation at Householder 2014
- First written attempt (horrible)
- Second attempt was much better (2015 paper) . . .
- . . . but there was one one more thing!
- Corrected in companion pencil paper. We also exploited the structure of the backward error to get a better result. Rejected!
- Search for examples.
- Take a closer look.
- backward error on pencil vs. polynomial coefficients
- monic vs. scaled polynomial
- Our analysis keeps getting better.

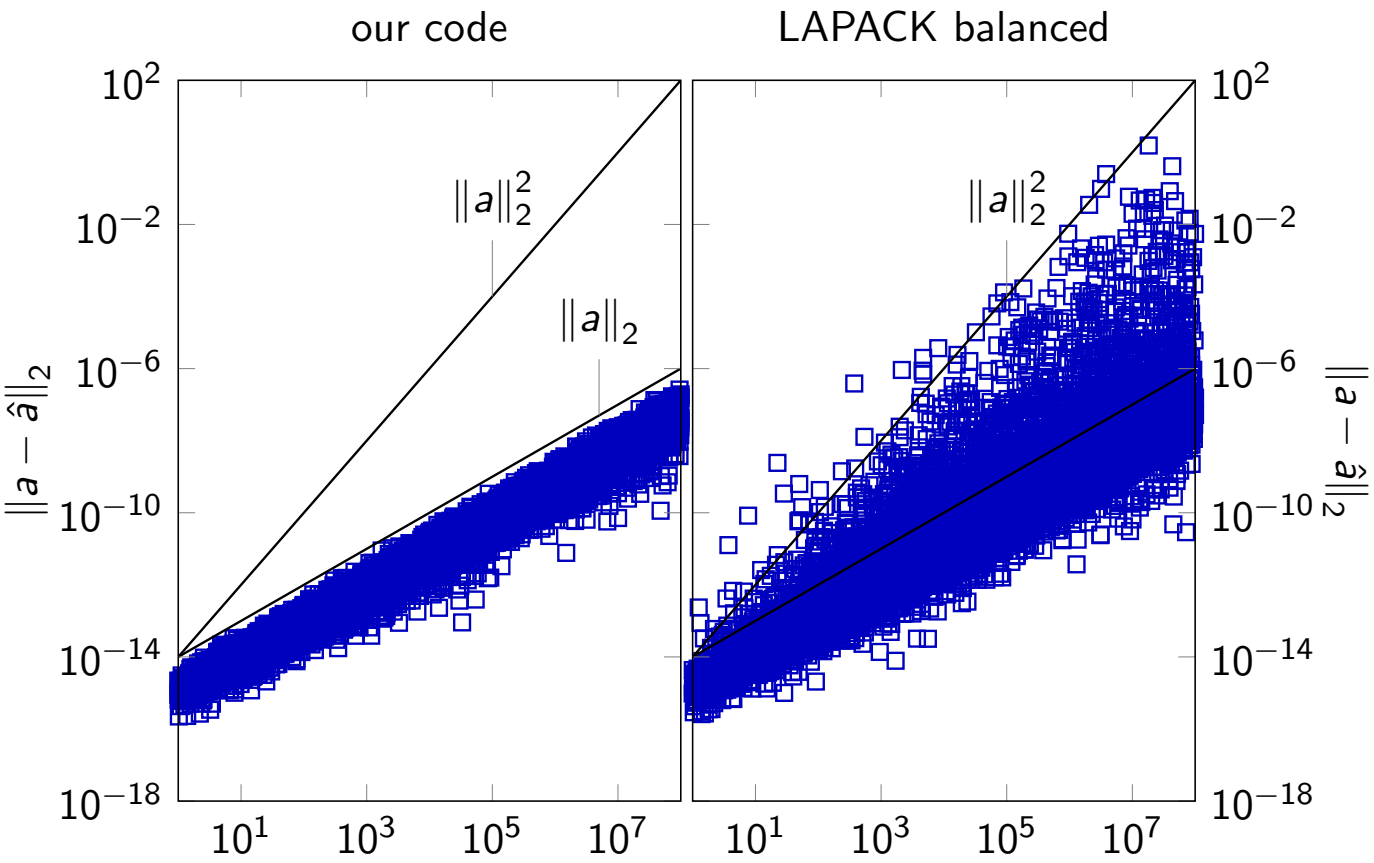
Backward Stability Odyssey

- Presentation at Householder 2014
- First written attempt (horrible)
- Second attempt was much better (2015 paper) . . .
- . . . but there was one one more thing!
- Corrected in companion pencil paper. We also exploited the structure of the backward error to get a better result. Rejected!
- Search for examples.
- Take a closer look.
- backward error on pencil vs. polynomial coefficients
- monic vs. scaled polynomial
- Our analysis keeps getting better.
- Stay tuned for the revised paper.

Nice Picture

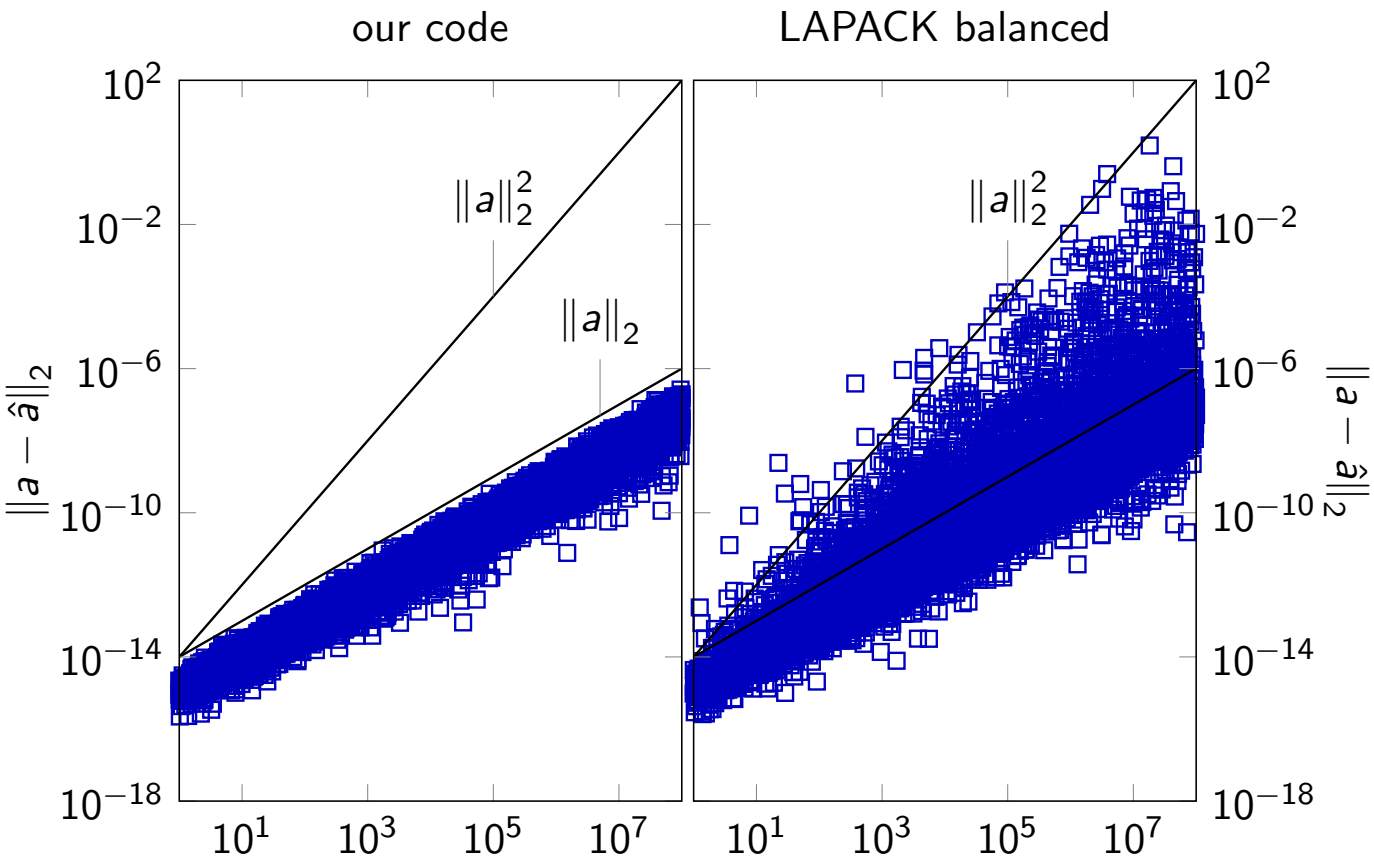


Nice Picture



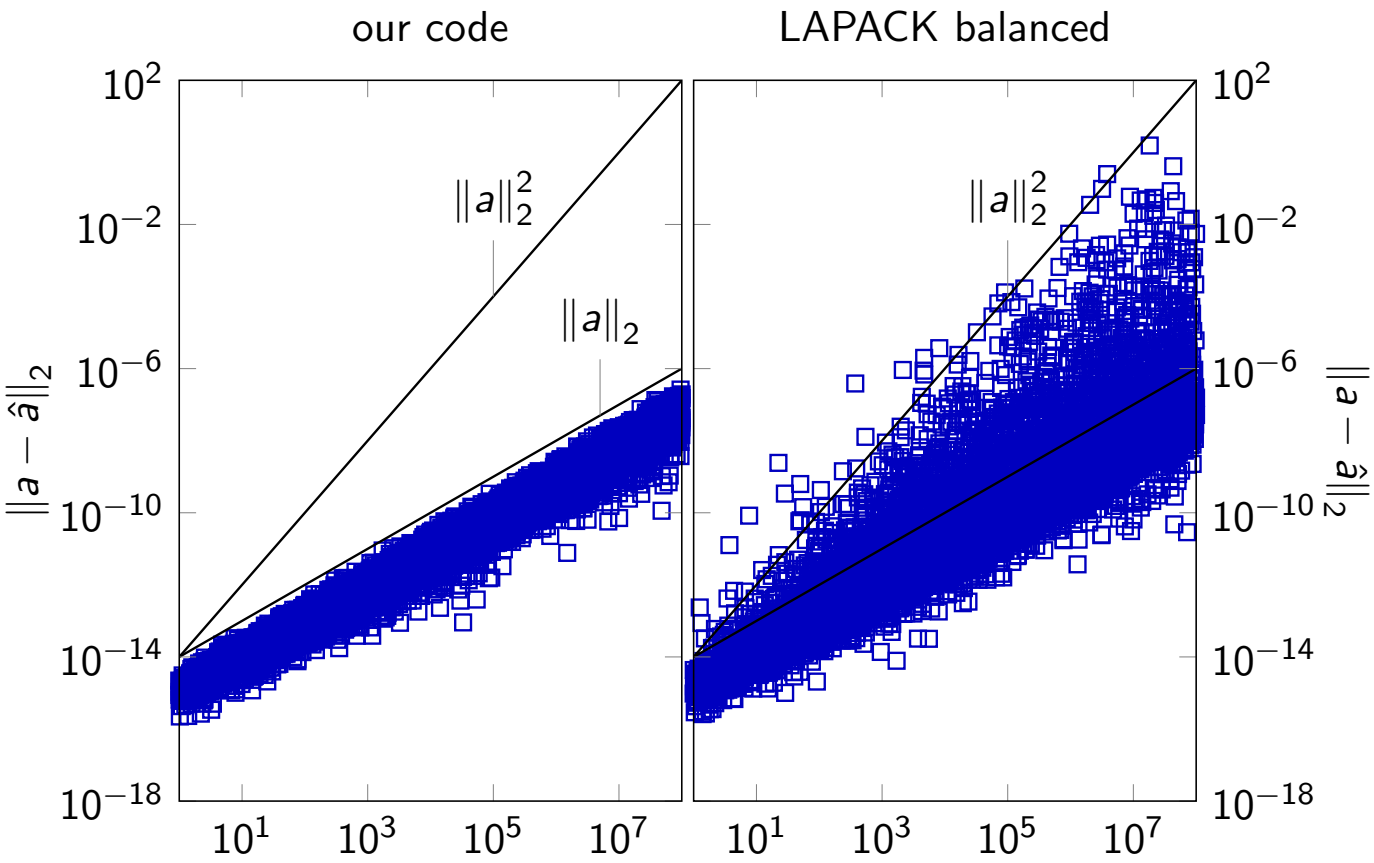
Our code is not just faster,

Nice Picture



Our code is not just faster, it is also more accurate!

Nice Picture



Our code is not just faster, it is also more accurate!

Thank you for your attention.