Solving the Matrix Eigenvalue Problem: Chasing bulges, cores, or poles

David S. Watkins

Department of Mathematics
Washington State University

October 2019

An Old Topic

An Old Topic

- Matrix eigenvalue problem

An Old Topic

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

An Old Topic

- Matrix eigenvalue problem
- $A \in \mathbb{C}^{n \times n}$
- Find the eigenvalues: $Ax = \lambda x$

An Old Topic

- Matrix eigenvalue problem
- $A \in \mathbb{C}^{n \times n}$
- Find the eigenvalues: $Ax = \lambda x$
- Generalized eigenvalue problem

An Old Topic

- Matrix eigenvalue problem
- $A \in \mathbb{C}^{n \times n}$
- Find the eigenvalues: $Ax = \lambda x$

- Generalized eigenvalue problem
- $A,\ B \in \mathbb{C}^{n \times n}$

An Old Topic

- Matrix eigenvalue problem
- $A \in \mathbb{C}^{n \times n}$
- Find the eigenvalues: $Ax = \lambda x$

- Generalized eigenvalue problem
- $A,\ B \in \mathbb{C}^{n \times n}$
- $Ax = \lambda Bx$

An Old Topic

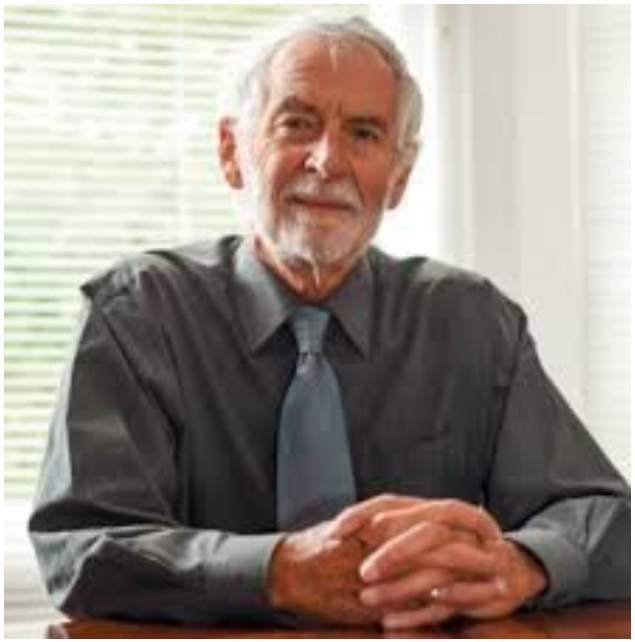
- Matrix eigenvalue problem
- $A \in \mathbb{C}^{n \times n}$
- Find the eigenvalues: $Ax = \lambda x$

- Generalized eigenvalue problem
- $A,\ B \in \mathbb{C}^{n \times n}$
- $Ax = \lambda Bx$

- Haven't these problems been solved?

The Winning Algorithm

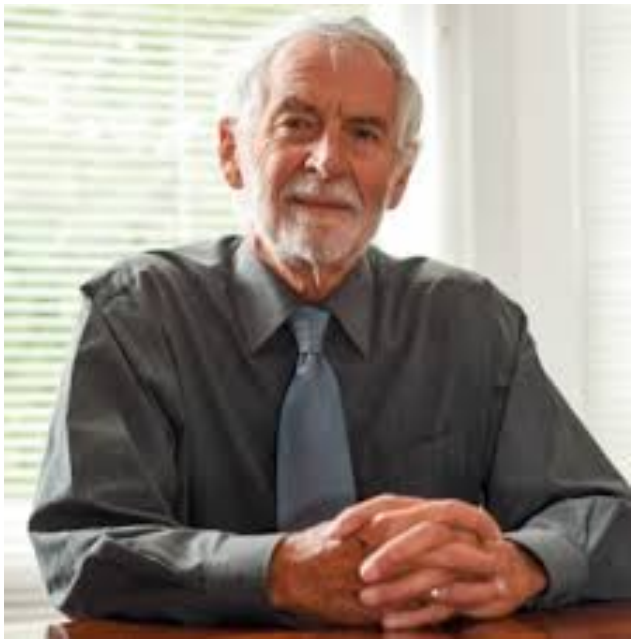
John Francis



- invented the winning algorithm in 1959.

The Winning Algorithm

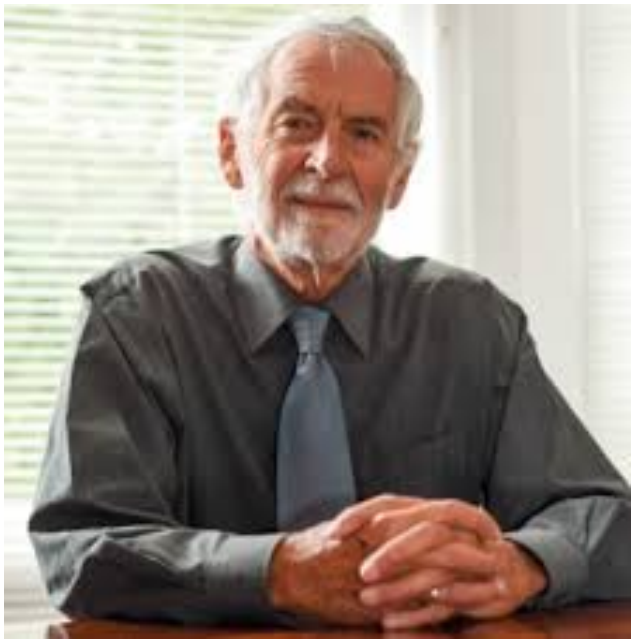
John Francis



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

The Winning Algorithm

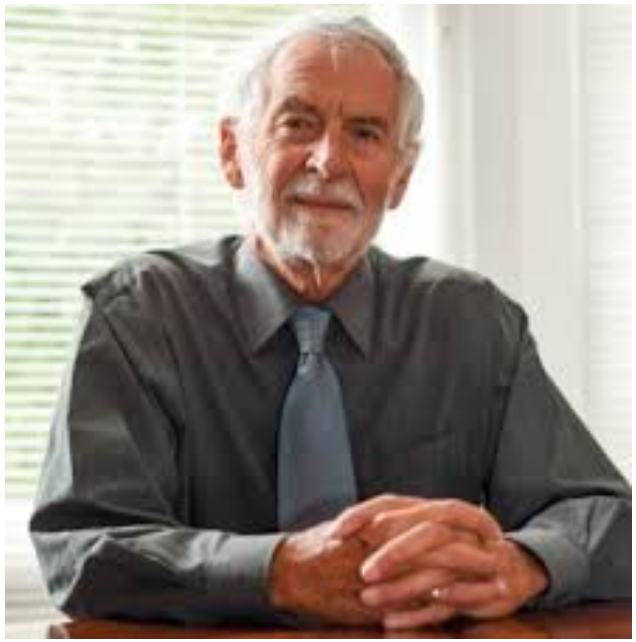
John Francis



- invented the winning algorithm in 1959.
- implicitly-shifted QR algorithm (terrible name)

The Winning Algorithm

John Francis



- invented the winning algorithm in 1959.
- implicitly-shifted QR algorithm (terrible name)
- I call it Francis's algorithm.

The Winning Algorithm

- Moler and Stewart

The Winning Algorithm

- Moler and Stewart
- extended to generalized eigenvalue problem (1973)

The Winning Algorithm

- Moler and Stewart
- extended to generalized eigenvalue problem (1973)
- commonly called the QZ algorithm

The Winning Algorithm

- Moler and Stewart
- extended to generalized eigenvalue problem (1973)
- commonly called the QZ algorithm (not a bad name)

The Winning Algorithm

- Moler and Stewart
- extended to generalized eigenvalue problem (1973)
- commonly called the QZ algorithm (not a bad name)

This is an old topic, but ...

Proliferation of Possibilities

Ways to implement Francis's algorithm:

Proliferation of Possibilities

Ways to implement Francis's algorithm:

- bulge chasing (1961)

Proliferation of Possibilities

Ways to implement Francis's algorithm:

- bulge chasing (1961)
  - my interest (1980–)

Proliferation of Possibilities

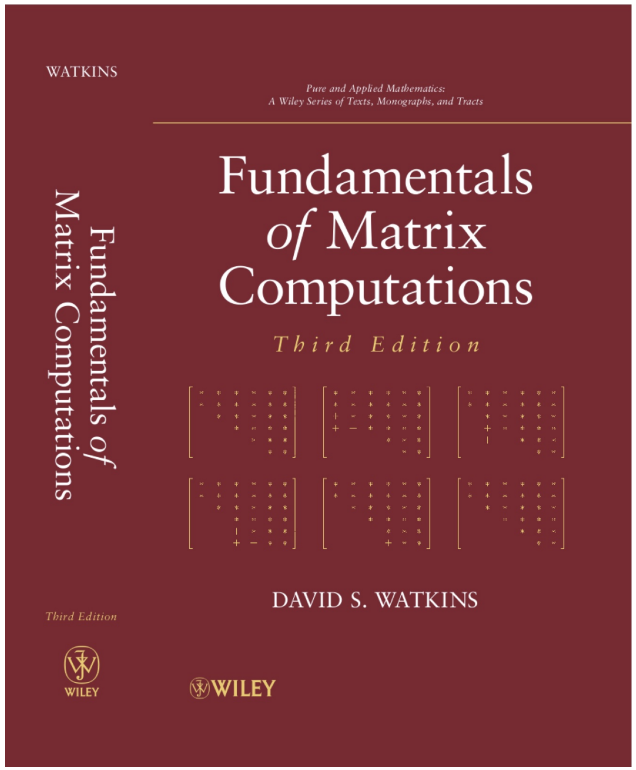
Ways to implement Francis's algorithm:

- bulge chasing (1961)
  - my interest (1980–)
  - *Francis's Algorithm*, Amer. Math. Monthly (2011).

Proliferation of Possibilities

Ways to implement Francis's algorithm:

- bulge chasing (1961)
  - my interest (1980–)
  - *Francis's Algorithm*, Amer. Math. Monthly (2011).

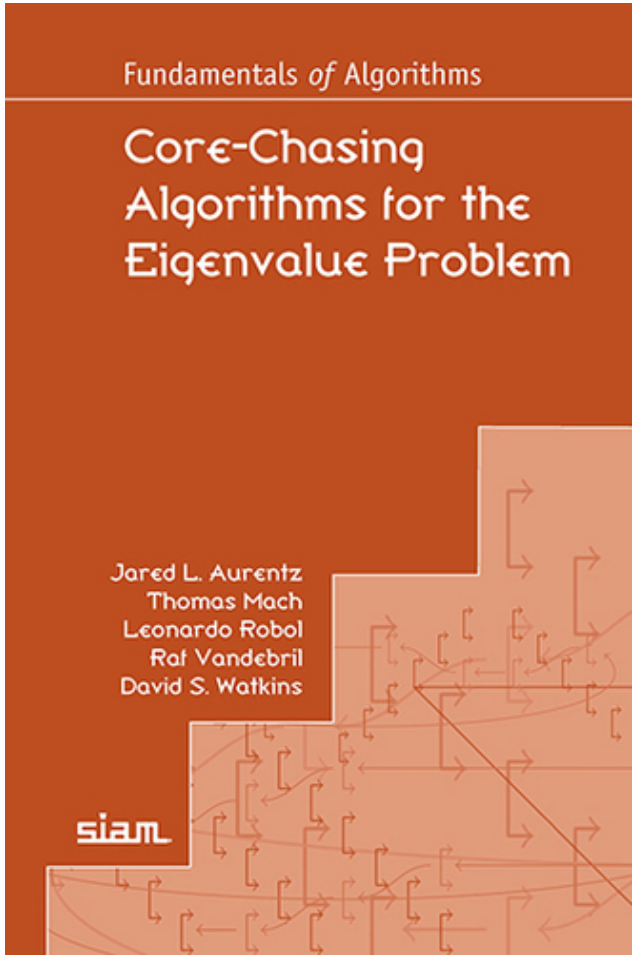


Proliferation of Possibilities

- core chasing (2011–)

Proliferation of Possibilities

- core chasing (2011–)



2018

Proliferation of Possibilities

- pole swapping (2016–)

Proliferation of Possibilities

- pole swapping (2016–)
- generalized eigenvalue problem

Proliferation of Possibilities

- pole swapping (2016–)
- generalized eigenvalue problem

- D. Camps, K. Meerbergen, and R. Vandebril, *A rational QZ method*, SIAM J. Matrix Anal. Appl., 40 (2019) pp. 943–972.
- D. Camps, K. Meerbergen, and R. Vandebril, *A multishift, multipole rational QZ method with aggressive early deflation*, arXiv:1902.10954.
- D. Camps, T. Mach, R. Vandebril, and D. S. Watkins, *On pole-swapping algorithms for the eigenvalue problem*, arXiv:1906.08672.

Proliferation of Possibilities

In summary: There are at least three ways to implement Francis and related algorithms.

- bulge chasing
- core chasing
- pole swapping

Proliferation of Possibilities

In summary: There are at least three ways to implement Francis and related algorithms.

- bulge chasing
- core chasing
- pole swapping

. . . and some of them are brand new!

Francis Bulge-Chasing Algorithm

Francis Bulge-Chasing Algorithm

- single-shift algorithm (not double-shift)

Francis Bulge-Chasing Algorithm

- single-shift algorithm (not double-shift)
- standard eigenvalue problem

Francis Bulge-Chasing Algorithm

- single-shift algorithm (not double-shift)
- standard eigenvalue problem
- upper Hessenberg

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

Francis Bulge-Chasing Algorithm

- single-shift algorithm (not double-shift)
- standard eigenvalue problem
- upper Hessenberg

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

- iterate!

Francis Bulge-Chasing Algorithm

Description of one iteration.

Francis Bulge-Chasing Algorithm

Description of one iteration.

- Pick a shift ρ. (simple example: $\rho = a_{nn}$)

Francis Bulge-Chasing Algorithm

Description of one iteration.

- Pick a shift ρ. (simple example: $\rho = a_{nn}$)
- Compute $x = (A - \rho I)e_1 = \begin{bmatrix} a_{11} - \rho \\ a_{21} \\ 0 \\ \vdots \end{bmatrix} = \begin{bmatrix} \times \\ \times \\ 0 \\ \vdots \end{bmatrix}$.

Francis Bulge-Chasing Algorithm

Description of one iteration.

- Pick a shift ρ. (simple example: $\rho = a_{nn}$)
- Compute $x = (A - \rho I)e_1 = \begin{bmatrix} a_{11} - \rho \\ a_{21} \\ 0 \\ \vdots \end{bmatrix} = \begin{bmatrix} \times \\ \times \\ 0 \\ \vdots \end{bmatrix}$.
- Compute unitary $Q = \begin{bmatrix} \times & \times & \\ \times & \times & \\ & & I \end{bmatrix}$ with first column proportional to x.

Francis Bulge-Chasing Algorithm

Description of one iteration.

- Pick a shift ρ. (simple example: $\rho = a_{nn}$)
- Compute $x = (A - \rho I)e_1 = \begin{bmatrix} a_{11} - \rho \\ a_{21} \\ 0 \\ \vdots \end{bmatrix} = \begin{bmatrix} \times \\ \times \\ 0 \\ \vdots \end{bmatrix}$.
- Compute unitary $Q = \begin{bmatrix} \times & \times & \\ \times & \times & \\ & & I \end{bmatrix}$
  with first column proportional to x. (core transformation)

Francis Bulge-Chasing Algorithm

Description of one iteration.

- Pick a shift ρ. (simple example: $\rho = a_{nn}$)
- Compute $x = (A - \rho I)e_1 = \begin{bmatrix} a_{11} - \rho \\ a_{21} \\ 0 \\ \vdots \end{bmatrix} = \begin{bmatrix} \times \\ \times \\ 0 \\ \vdots \end{bmatrix}$.
- Compute unitary $Q = \begin{bmatrix} \times & \times & \\ \times & \times & \\ & & I \end{bmatrix}$ with first column proportional to x. (core transformation)
- Similarity transformation $A \to Q^* A Q$

Francis Bulge-Chasing Algorithm

Description of one iteration.

- Pick a shift ρ. (simple example: $\rho = a_{nn}$)
- Compute $x = (A - \rho I)e_1 = \begin{bmatrix} a_{11} - \rho \\ a_{21} \\ 0 \\ \vdots \end{bmatrix} = \begin{bmatrix} \times \\ \times \\ 0 \\ \vdots \end{bmatrix}$.
- Compute unitary $Q = \begin{bmatrix} \times & \times & \\ \times & \times & \\ & & I \end{bmatrix}$ with first column proportional to x. (core transformation)
- Similarity transformation $A \to Q^* A Q$ creates a bulge in the Hessenberg form.

Francis Bulge-Chasing Algorithm

Q^*AQ

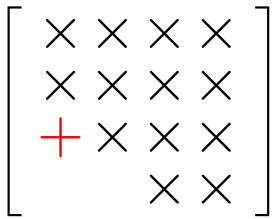
Francis Bulge-Chasing Algorithm

$$Q^*AQ = \begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ & \times & \times & \times \\ & & \times & \times \end{bmatrix}$$

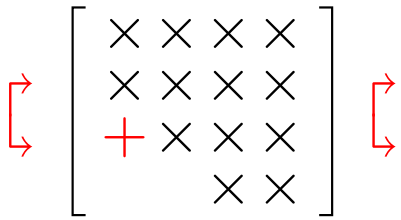
Francis Bulge-Chasing Algorithm

$$Q^*AQ = \hookrightarrow \begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ & \times & \times & \times \\ & & \times & \times \end{bmatrix} \hookrightarrow = \begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ + & \times & \times & \times \\ & & \times & \times \end{bmatrix}$$

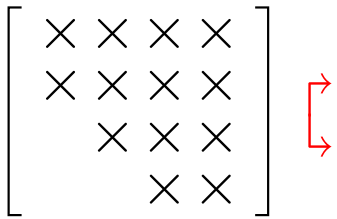
Francis Bulge-Chasing Algorithm



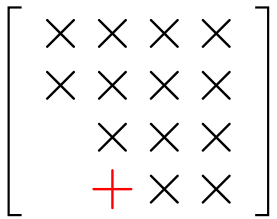
Francis Bulge-Chasing Algorithm



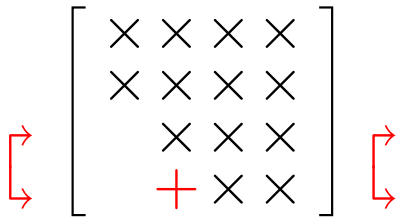
Francis Bulge-Chasing Algorithm



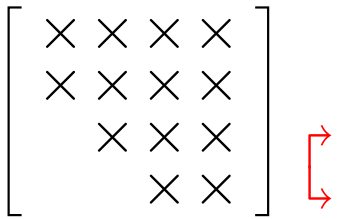
Francis Bulge-Chasing Algorithm



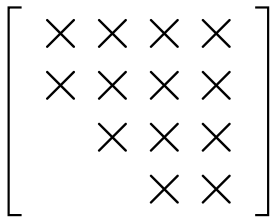
Francis Bulge-Chasing Algorithm



Francis Bulge-Chasing Algorithm



Francis Bulge-Chasing Algorithm



Francis Bulge-Chasing Algorithm

- Iteration is complete.

Francis Bulge-Chasing Algorithm

- Iteration is complete.
- Now repeat.

Francis Bulge-Chasing Algorithm

- Iteration is complete.
- Now repeat.

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

Francis Bulge-Chasing Algorithm

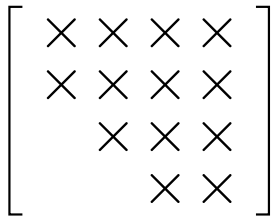
- Iteration is complete.
- Now repeat.

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

- $a_{n,n-1} \to 0$

Francis Bulge-Chasing Algorithm

- Iteration is complete.
- Now repeat.



- $a_{n,n-1} \to 0$ (quadratically)

Francis's Algorithm as a Core-Chasing Algorithm

Reformulation as a core-chasing algorithm

Francis's Algorithm as a Core-Chasing Algorithm

Reformulation as a core-chasing algorithm

Write A in QR decomposed form: $A = QR$

Francis's Algorithm as a Core-Chasing Algorithm

Reformulation as a core-chasing algorithm

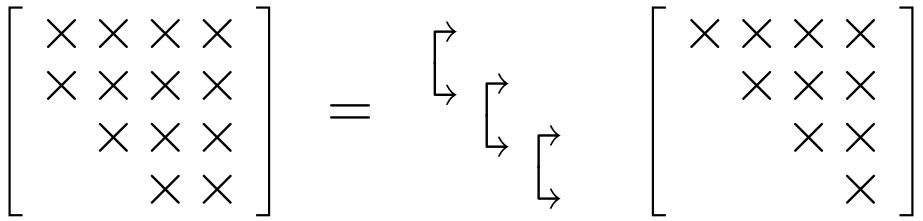
Write A in QR decomposed form: $A = QR$

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

Francis's Algorithm as a Core-Chasing Algorithm

Reformulation as a core-chasing algorithm

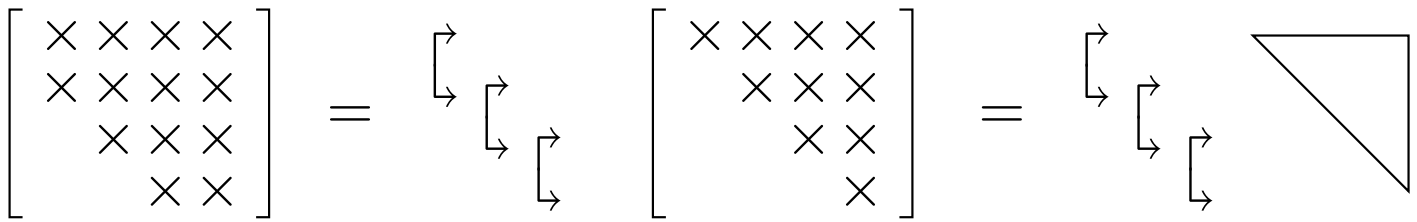
Write A in QR decomposed form: $A = QR$



Francis's Algorithm as a Core-Chasing Algorithm

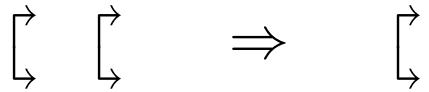
Reformulation as a core-chasing algorithm

Write A in QR decomposed form: $A = QR$



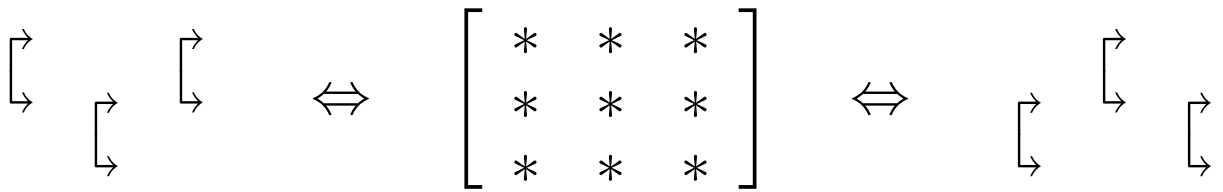
Operating on Core Transformations

Fusion



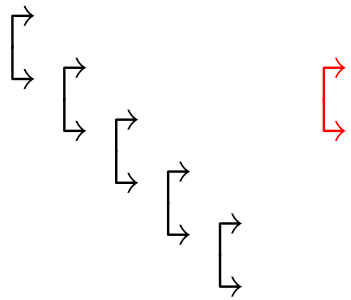
Operating on Core Transformations

Turnover



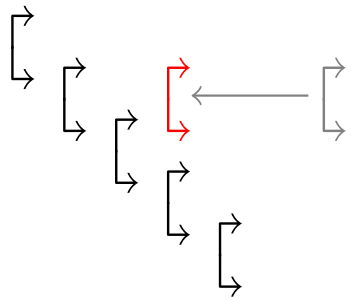
Operating on Core Transformations

Turnover as a shift-through operation



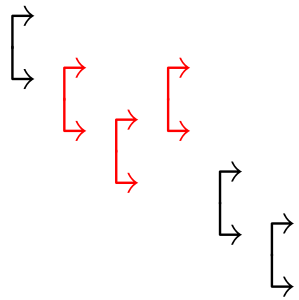
Operating on Core Transformations

Turnover as a shift-through operation



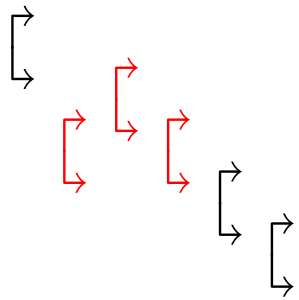
Operating on Core Transformations

Turnover as a shift-through operation



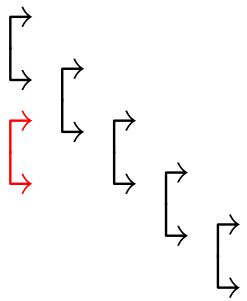
Operating on Core Transformations

Turnover as a shift-through operation



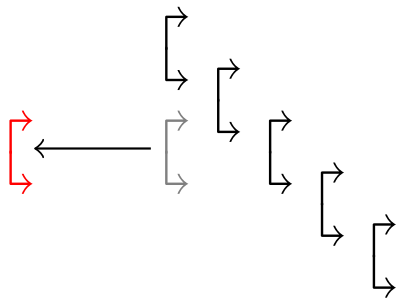
Operating on Core Transformations

Turnover as a shift-through operation



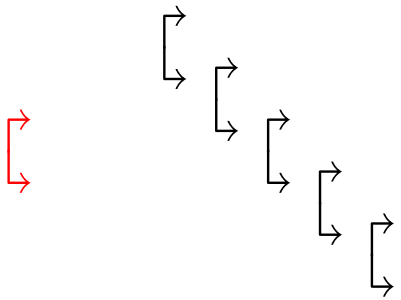
Operating on Core Transformations

Turnover as a shift-through operation



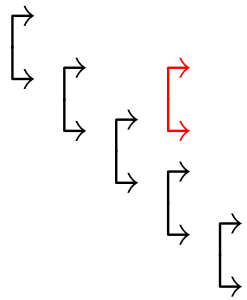
Operating on Core Transformations

Turnover as a shift-through operation



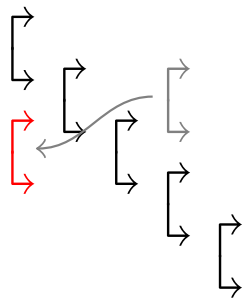
Operating on Core Transformations

Turnover as a shift-through operation : Abbreviated notation



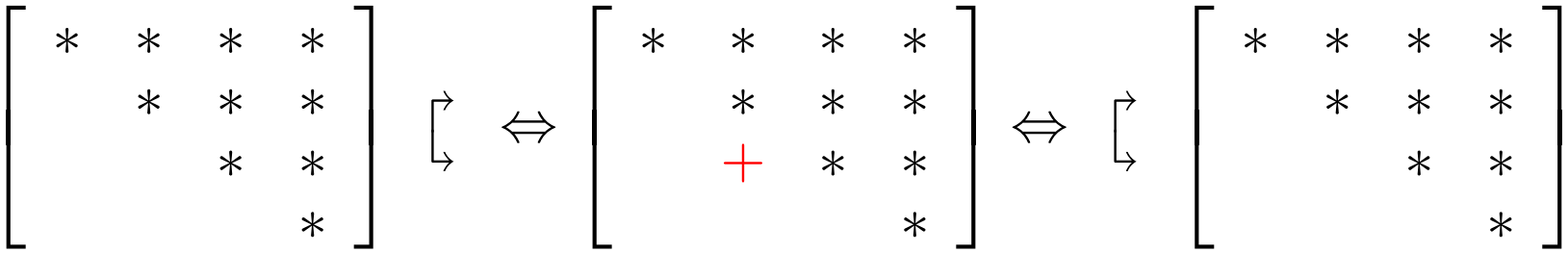
Operating on Core Transformations

Turnover as a shift-through operation : Abbreviated notation



Operating on Core Transformations

Passing a core transformation through a triangular matrix

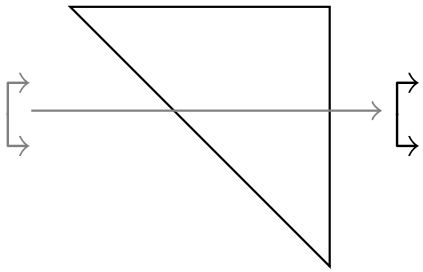


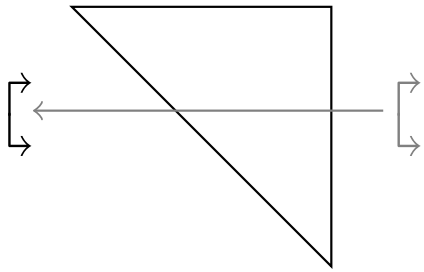
Cost is $O(n)$ flops.

Operating on Core Transformations

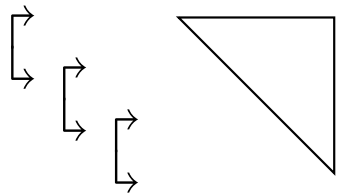
Passing a core transformation through a triangular matrix

Abbreviated Notation:

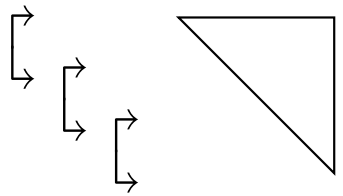




Francis's Algorithm as a Core-Chasing Algorithm

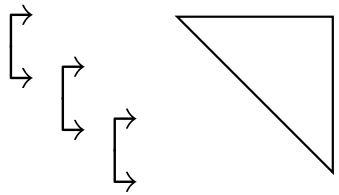
$A = QR =$ 

Francis's Algorithm as a Core-Chasing Algorithm

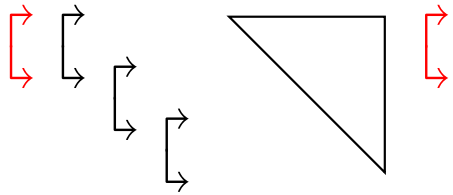
$A = QR =$ 

We will demonstrate one iteration.

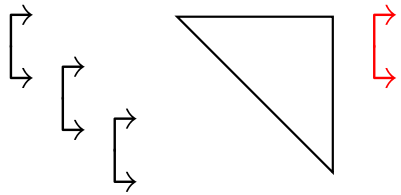
Francis's Algorithm as a Core-Chasing Algorithm



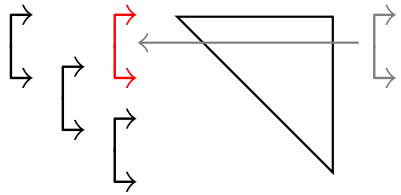
Francis's Algorithm as a Core-Chasing Algorithm



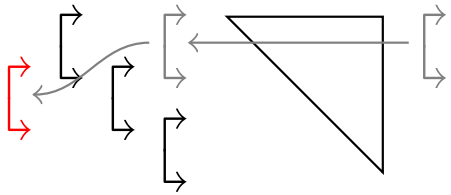
Francis's Algorithm as a Core-Chasing Algorithm



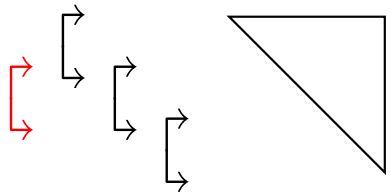
Francis's Algorithm as a Core-Chasing Algorithm



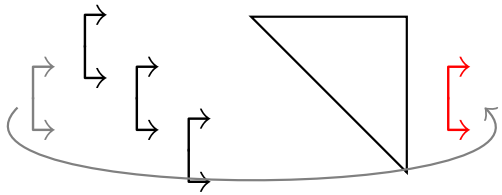
Francis's Algorithm as a Core-Chasing Algorithm



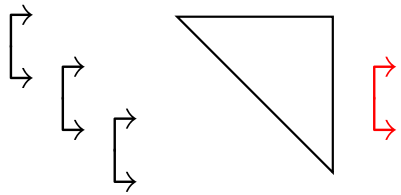
Francis's Algorithm as a Core-Chasing Algorithm



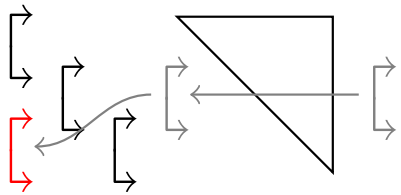
Francis's Algorithm as a Core-Chasing Algorithm



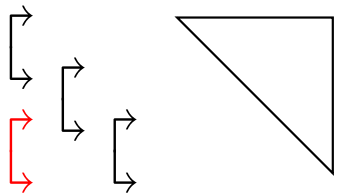
Francis's Algorithm as a Core-Chasing Algorithm



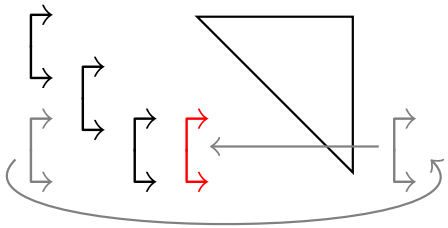
Francis's Algorithm as a Core-Chasing Algorithm



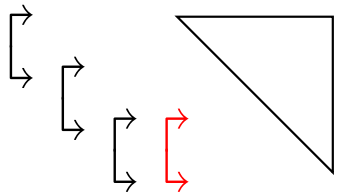
Francis's Algorithm as a Core-Chasing Algorithm



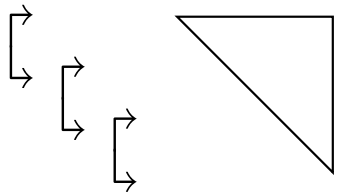
Francis's Algorithm as a Core-Chasing Algorithm



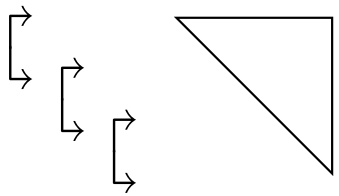
Francis's Algorithm as a Core-Chasing Algorithm



Francis's Algorithm as a Core-Chasing Algorithm



Francis's Algorithm as a Core-Chasing Algorithm



Done!

Advantages of Core Chasing

Are there any advantages?

Advantages of Core Chasing

Are there any advantages?

- Yes,

Advantages of Core Chasing

Are there any advantages?

- Yes, even for unstructured problems,

Advantages of Core Chasing

Are there any advantages?

- Yes, even for unstructured problems,
- but consider certain structured cases:

Advantages of Core Chasing

Are there any advantages?

- Yes, even for unstructured problems,
- but consider certain structured cases:
  - unitary

Advantages of Core Chasing

Are there any advantages?

- Yes, even for unstructured problems,
- but consider certain structured cases:
  - unitary
  - unitary-plus-rank-one (including companion matrices)

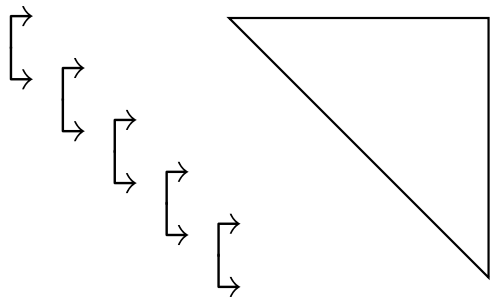
Advantages of Core Chasing

Are there any advantages?

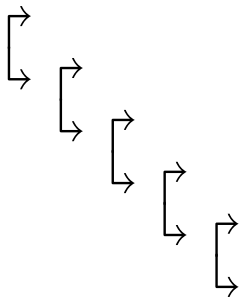
- Yes, even for unstructured problems,
- but consider certain structured cases:
  - unitary
  - unitary-plus-rank-one (including companion matrices)
  - unitary-plus-low-rank

Unitary Case

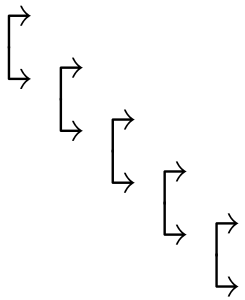
Unitary Case

$A = QR =$ 

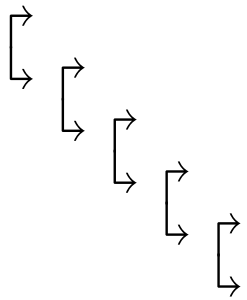
Unitary Case

$A = QR =$ 

Unitary Case

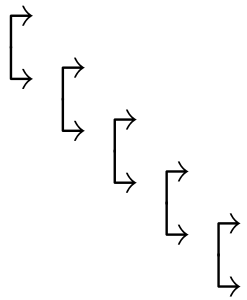
$A = QR =$ 

Unitary Case

$A = QR =$ 

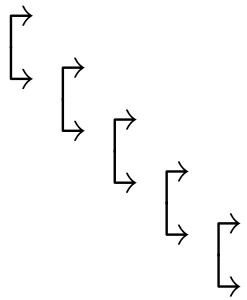
- The expensive part is gone!

Unitary Case

$A = QR =$ 

- The expensive part is gone!
- This is a fast algorithm.

Unitary Case

$A = QR =$ 

- The expensive part is gone!
- This is a fast algorithm.
- Gragg (1986)

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

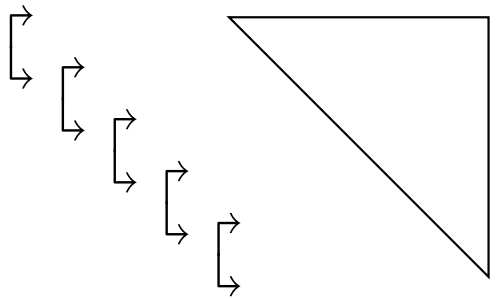
Companion Case

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

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

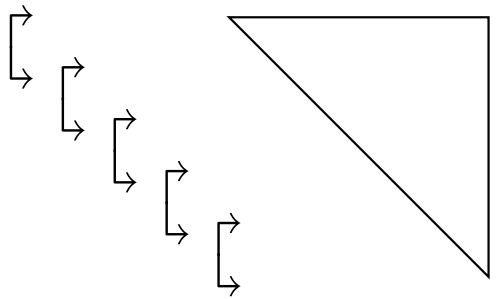
- . . . get the zeros of p by computing the eigenvalues.
- MATLAB's `roots` command
- Companion matrix is unitary-plus-rank-one.

Unitary-plus-low-rank Case

$A = QR =$ 

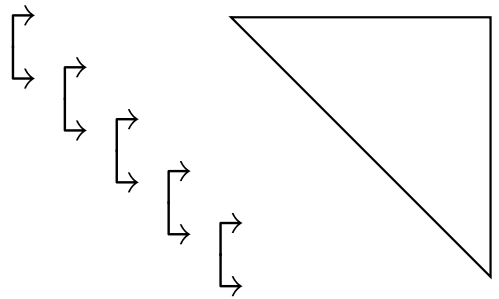
- Suppose A is unitary-plus-rank-k. $(k \ll n)$

Unitary-plus-low-rank Case

$A = QR =$ 

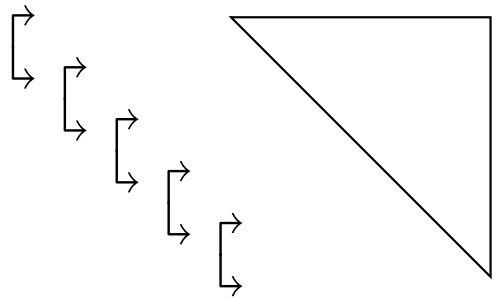
- Suppose A is unitary-plus-rank-k. $(k \ll n)$
- Then R is also unitary-plus-rank-k.

Unitary-plus-low-rank Case

$A = QR =$ 

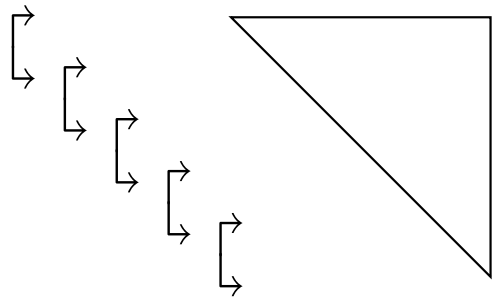
- Suppose A is unitary-plus-rank-k. $(k \ll n)$
- Then R is also unitary-plus-rank-k.
- This property is preserved by iterations of Francis's algorithm.

Unitary-plus-low-rank Case

$A = QR =$ 

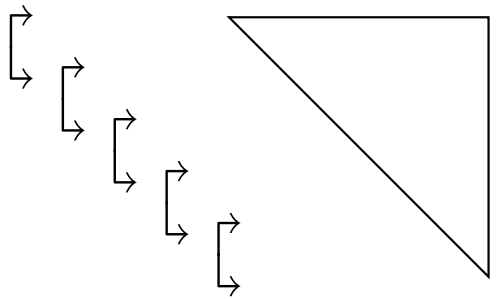
- Suppose A is unitary-plus-rank-k. $(k \ll n)$
- Then R is also unitary-plus-rank-k.
- This property is preserved by iterations of Francis's algorithm.
- Task:

Unitary-plus-low-rank Case

$A = QR =$ 

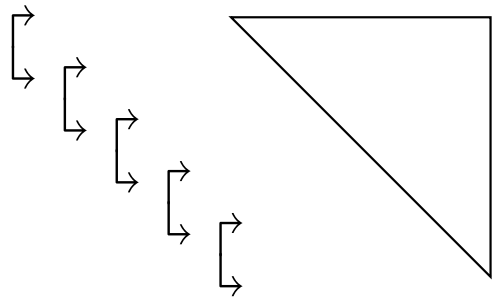
- Suppose A is unitary-plus-rank-k. $(k \ll n)$
- Then R is also unitary-plus-rank-k.
- This property is preserved by iterations of Francis's algorithm.
- Task: Store R in a way that exploits this structure. $(O(n))$

Unitary-plus-low-rank Case

$A = QR =$ 

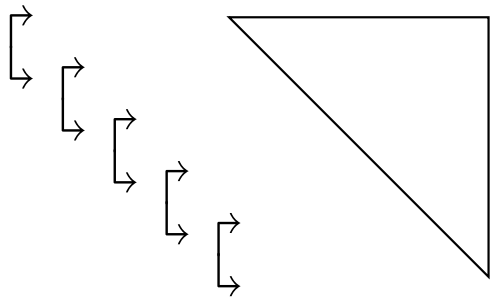
- Suppose A is unitary-plus-rank-k. $(k \ll n)$
- Then R is also unitary-plus-rank-k.
- This property is preserved by iterations of Francis's algorithm.
- Task: Store R in a way that exploits this structure. $(O(n))$
- This leads automatically to a fast algorithm.

Unitary-plus-low-rank Case

$A = QR =$ 

- Suppose A is unitary-plus-rank-k. $(k \ll n)$
- Then R is also unitary-plus-rank-k.
- This property is preserved by iterations of Francis's algorithm.
- Task: Store R in a way that exploits this structure. $(O(n))$
- This leads automatically to a fast algorithm.
- We did this,

Unitary-plus-low-rank Case

$A = QR =$ 

- Suppose A is unitary-plus-rank-k. $(k \ll n)$
- Then R is also unitary-plus-rank-k.
- This property is preserved by iterations of Francis's algorithm.
- Task: Store R in a way that exploits this structure. $(O(n))$
- This leads automatically to a fast algorithm.
- We did this, but it took us a while.

Unitary-plus-low-rank Case

After two years and some pushing we managed to publish

Unitary-plus-low-rank Case

After two years and some pushing we managed to publish

- 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.

Unitary-plus-low-rank Case

After two years and some pushing we managed to publish

- 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.

We thought it was pretty good, and so did some other people.

Unitary-plus-low-rank Case

After two years and some pushing we managed to publish

- 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.

We thought it was pretty good, and so did some other people.

SIAM Outstanding Paper Prize (2017)

Unitary-plus-low-rank Case

After two more years we published

Unitary-plus-low-rank Case

After two more years we published

- 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), pp. 1245–1269.

Unitary-plus-low-rank Case

After two more years we published

- 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), pp. 1245–1269.
- Jared L. Aurentz, Thomas Mach, Leonardo Robol, Raf Vandebril, and David S. Watkins, *Fast and backward stable computation of the eigenvalues and eigenvectors of matrix polynomials,* Math. Comp., 88 (2019), pp. 313–347.

Unitary-plus-low-rank Case

After two more years we published

- 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), pp. 1245–1269.
- Jared L. Aurentz, Thomas Mach, Leonardo Robol, Raf Vandebril, and David S. Watkins, *Fast and backward stable computation of the eigenvalues and eigenvectors of matrix polynomials,* Math. Comp., 88 (2019), pp. 313–347.

Read these papers, or better yet

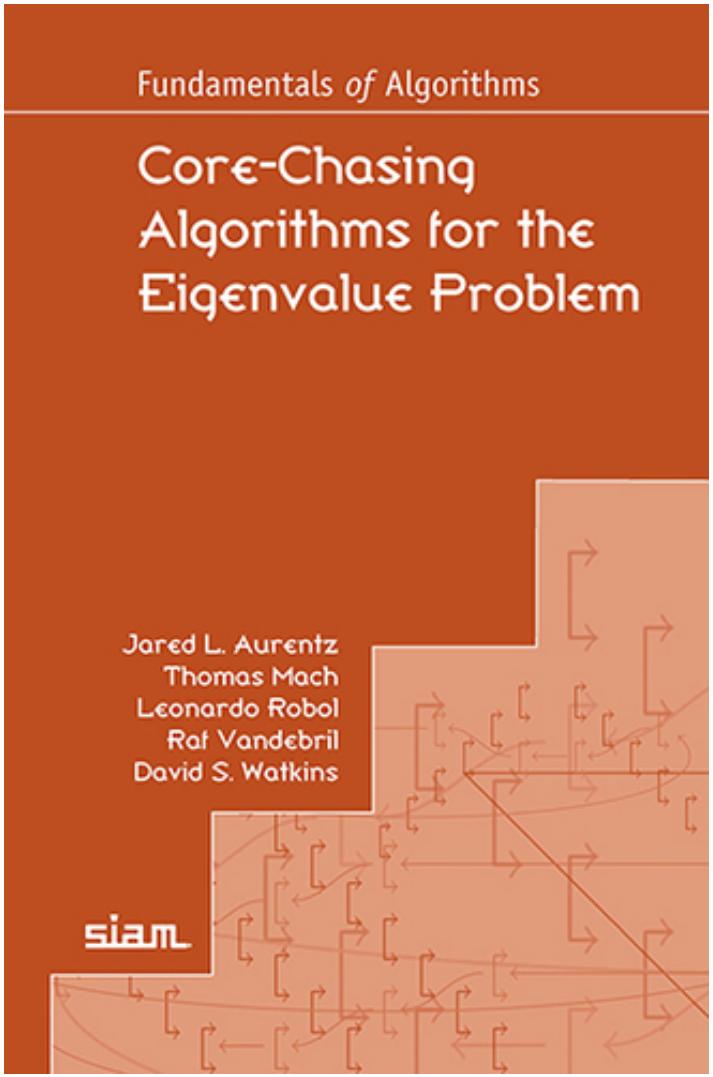
Unitary-plus-low-rank Case

After two more years we published

- 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), pp. 1245–1269.
- Jared L. Aurentz, Thomas Mach, Leonardo Robol, Raf Vandebril, and David S. Watkins, *Fast and backward stable computation of the eigenvalues and eigenvectors of matrix polynomials,* Math. Comp., 88 (2019), pp. 313–347.

Read these papers, or better yet read the book!

Core-Chasing Algorithms . . . (2018)



and now for Pole Swapping

So far we've looked at

and now for Pole Swapping

So far we've looked at

- bulge chasing,

and now for Pole Swapping

So far we've looked at

- bulge chasing,
- core chasing.

and now for Pole Swapping

So far we've looked at

- bulge chasing,
- core chasing.

Now it's time for

and now for Pole Swapping

So far we've looked at

- bulge chasing,
- core chasing.

Now it's time for

- pole swapping.

and now for Pole Swapping

So far we've looked at

- bulge chasing,
- core chasing.

Now it's time for

- pole swapping.

This is best introduced via the generalized eigenvalue problem.

Generalized Eigenvalue Problem

$$Ax = \lambda Bx$$

Generalized Eigenvalue Problem

$$Ax = \lambda Bx$$

$$A = \begin{bmatrix} \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \end{bmatrix} \qquad B = \begin{bmatrix} \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \end{bmatrix}$$

Generalized Eigenvalue Problem

$Ax = \lambda Bx$

$$A = \begin{bmatrix} \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \end{bmatrix} \qquad B = \begin{bmatrix} \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \end{bmatrix}$$

Moler/Stewart:

Generalized Eigenvalue Problem

$Ax = \lambda Bx$

$$A = \begin{bmatrix} \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \end{bmatrix} \qquad B = \begin{bmatrix} \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \end{bmatrix}$$

Moler/Stewart:

$$A = \begin{bmatrix} \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ & \times & \times & \times & \times & \times \\ & & \times & \times & \times & \times \\ & & & \times & \times & \times \\ & & & & \times & \times \end{bmatrix} \qquad B = \begin{bmatrix} \times & \times & \times & \times & \times & \times \\ & \times & \times & \times & \times & \times \\ & & \times & \times & \times & \times \\ & & & \times & \times & \times \\ & & & & \times & \times \\ & & & & & \times \end{bmatrix}$$

Generalized Eigenvalue Problem

$Ax = \lambda Bx$

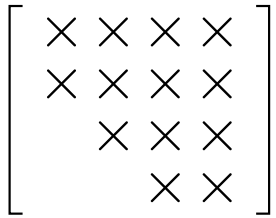
$$A = \begin{bmatrix} \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \end{bmatrix} \qquad B = \begin{bmatrix} \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \end{bmatrix}$$

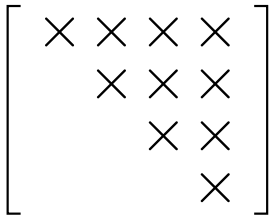
Moler/Stewart:

$$A = \begin{bmatrix} \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ & \times & \times & \times & \times & \times \\ & & \times & \times & \times & \times \\ & & & \times & \times & \times \\ & & & & \times & \times \end{bmatrix} \qquad B = \begin{bmatrix} \times & \times & \times & \times & \times & \times \\ & \times & \times & \times & \times & \times \\ & & \times & \times & \times & \times \\ & & & \times & \times & \times \\ & & & & \times & \times \\ & & & & & \times \end{bmatrix}$$

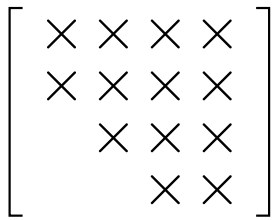
Hessenberg-triangular form

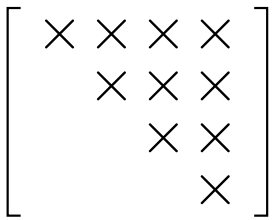
Moler/Stewart Bulge Chase





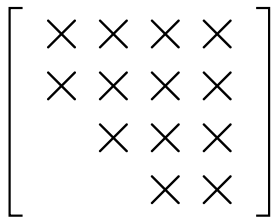
Moler/Stewart Bulge Chase

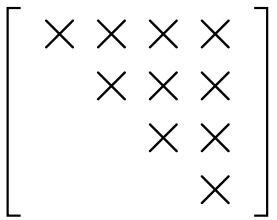




- Create a bulge and chase it.

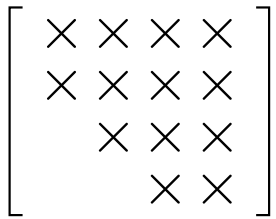
Moler/Stewart Bulge Chase

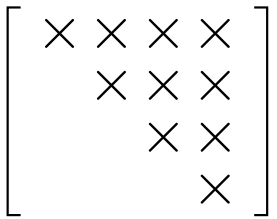




- Create a bulge and chase it.
- single-shift case

Moler/Stewart Bulge Chase

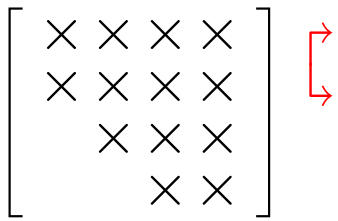


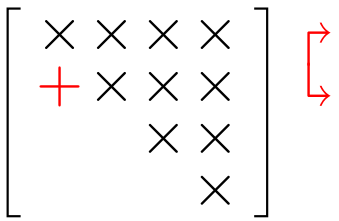


Moler/Stewart Bulge Chase

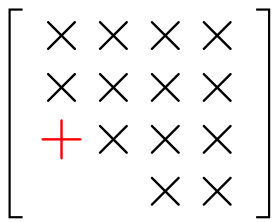
Moler/Stewart Bulge Chase

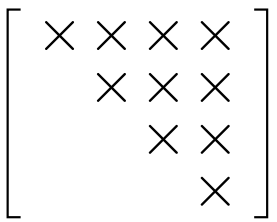
Moler/Stewart Bulge Chase





Moler/Stewart Bulge Chase

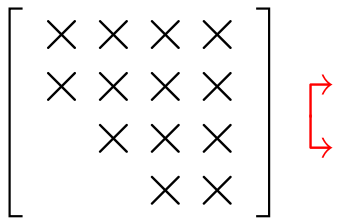


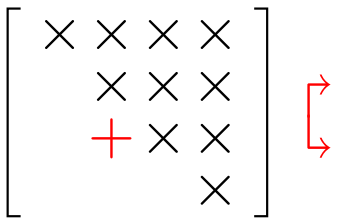


Moler/Stewart Bulge Chase

Moler/Stewart Bulge Chase

Moler/Stewart Bulge Chase

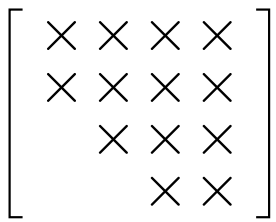


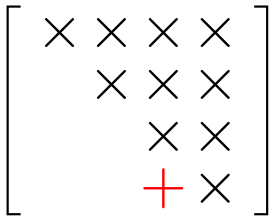


Moler/Stewart Bulge Chase

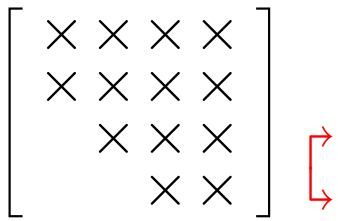
Moler/Stewart Bulge Chase

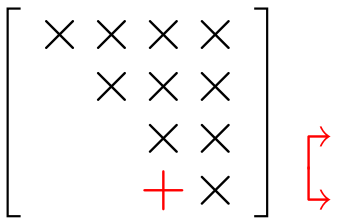
Moler/Stewart Bulge Chase



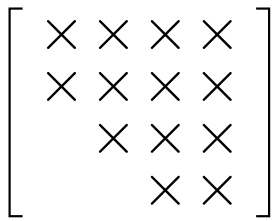


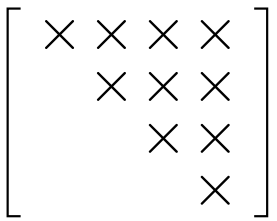
Moler/Stewart Bulge Chase



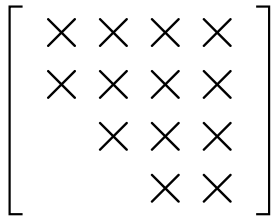


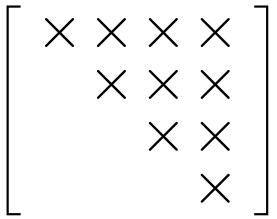
Moler/Stewart Bulge Chase





Moler/Stewart Bulge Chase





- Done!

Moler/Stewart Bulge Chase

- Done!
- Now repeat, again and again.

Moler/Stewart Bulge Chase

- Done!
- Now repeat, again and again.
- $a_{n,n-1} \to 0$ (quadratically)

Moler/Stewart Bulge Chase

- Done!
- Now repeat, again and again.
- $a_{n,n-1} \to 0$ (quadratically)

- This can also be done by core chasing,

Moler/Stewart Bulge Chase

- Done!
- Now repeat, again and again.
- $a_{n,n-1} \to 0$ (quadratically)

- This can also be done by core chasing,
- but let's go straight to pole swapping.

Hessenberg Pairs

Need a condensed form.

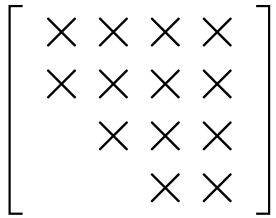
Hessenberg Pairs

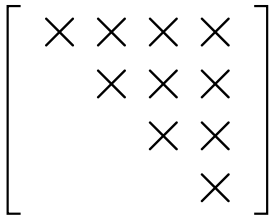
Need a condensed form.

Hessenberg-triangular form can be used, but . . .

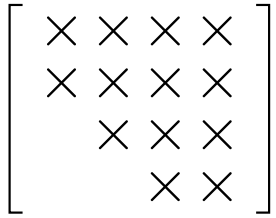
Hessenberg Pairs

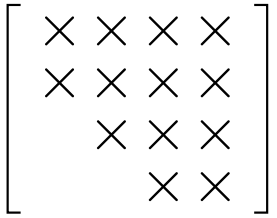
Need a condensed form.





Hessenberg-triangular form can be used, but . . .

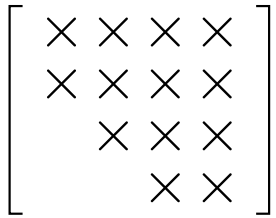


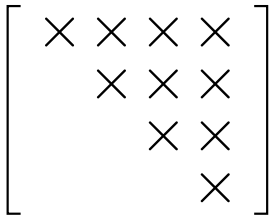


Hessenberg-Hessenberg form is good enough!

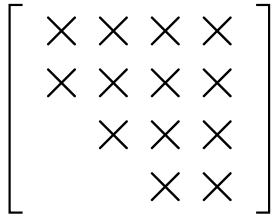
Hessenberg Pairs

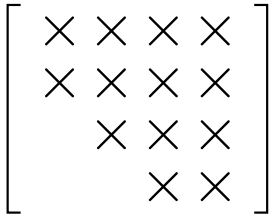
Need a condensed form.





Hessenberg-triangular form can be used, but . . .





Hessenberg-Hessenberg form is good enough!

We call this a *Hessenberg pair* or *Hessenberg pencil*.

Moler/Stewart Bulge Chase Revisited

Hessenberg pair

Moler/Stewart Bulge Chase Revisited

not Hessenberg

Moler/Stewart Bulge Chase Revisited

Hessenberg pair

Moler/Stewart Bulge Chase Revisited

not Hessenberg

Moler/Stewart Bulge Chase Revisited

Hessenberg pair

Pole Swapping

Poles of a Hessenberg pair

Pole Swapping

Poles of a Hessenberg pair

$$A = \begin{bmatrix} \times & \times & \times & \times \\ \alpha_1 & \times & \times & \times \\ & \alpha_2 & \times & \times \\ & & \alpha_3 & \times \end{bmatrix} \qquad B = \begin{bmatrix} \times & \times & \times & \times \\ \beta_1 & \times & \times & \times \\ & \beta_2 & \times & \times \\ & & \beta_3 & \times \end{bmatrix}$$

Pole Swapping

Poles of a Hessenberg pair

$$A = \begin{bmatrix} \times & \times & \times & \times \\ \alpha_1 & \times & \times & \times \\ & \alpha_2 & \times & \times \\ & & \alpha_3 & \times \end{bmatrix} \qquad B = \begin{bmatrix} \times & \times & \times & \times \\ \beta_1 & \times & \times & \times \\ & \beta_2 & \times & \times \\ & & \beta_3 & \times \end{bmatrix}$$

Poles: α_1/β_1 α_2/β_2 α_3/β_3

Pole Swapping

Poles of a Hessenberg pair

$$A = \begin{bmatrix} \times & \times & \times & \times \\ \alpha_1 & \times & \times & \times \\ & \alpha_2 & \times & \times \\ & & \alpha_3 & \times \end{bmatrix} \qquad B = \begin{bmatrix} \times & \times & \times & \times \\ \beta_1 & \times & \times & \times \\ & \beta_2 & \times & \times \\ & & \beta_3 & \times \end{bmatrix}$$

$$\textit{Poles:} \quad \alpha_1/\beta_1 \quad \alpha_2/\beta_2 \quad \alpha_3/\beta_3$$

This is related to the rational Arnoldi algorithm.

Pole Swapping

Poles of a Hessenberg pair

$$A = \begin{bmatrix} \times & \times & \times & \times \\ \alpha_1 & \times & \times & \times \\ & \alpha_2 & \times & \times \\ & & \alpha_3 & \times \end{bmatrix} \qquad B = \begin{bmatrix} \times & \times & \times & \times \\ \beta_1 & \times & \times & \times \\ & \beta_2 & \times & \times \\ & & \beta_3 & \times \end{bmatrix}$$

$$\textit{Poles:} \quad \alpha_1/\beta_1 \quad \alpha_2/\beta_2 \quad \alpha_3/\beta_3$$

This is related to the rational Arnoldi algorithm.

Hessenberg-triangular form: special case with all poles ∞.

Pole Swapping

Poles of a Hessenberg pair

$$A = \begin{bmatrix} \times & \times & \times & \times \\ \alpha_1 & \times & \times & \times \\ & \alpha_2 & \times & \times \\ & & \alpha_3 & \times \end{bmatrix} \qquad B = \begin{bmatrix} \times & \times & \times & \times \\ \beta_1 & \times & \times & \times \\ & \beta_2 & \times & \times \\ & & \beta_3 & \times \end{bmatrix}$$

$$\textit{Poles:} \quad \alpha_1/\beta_1 \quad \alpha_2/\beta_2 \quad \alpha_3/\beta_3$$

This is related to the rational Arnoldi algorithm.

Hessenberg-triangular form: special case with all poles ∞.

Pole-swapping algorithms manipulate the poles.

Pole Swapping

Two types of Moves:

Pole Swapping

Two types of Moves:

I: Change the top (or bottom) pole to any desired value.

Pole Swapping

Two types of Moves:

I: Change the top (or bottom) pole to any desired value.

II: Interchange any two adjacent poles.

Pole Swapping

Move of type I:

introducing a new pole at top

Pole Swapping

Move of type I:

introducing a new pole at top

Pole Swapping

Move of type I:

introducing a new pole at top

Pole Swapping

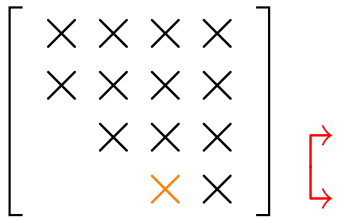
Move of type I:

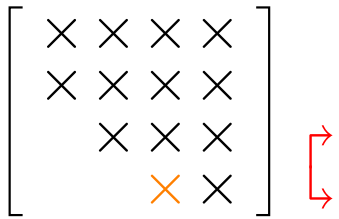
introducing a new pole at bottom

Pole Swapping

Move of type I:

introducing a new pole at bottom





Pole Swapping

Move of type I:

introducing a new pole at bottom

Pole Swapping

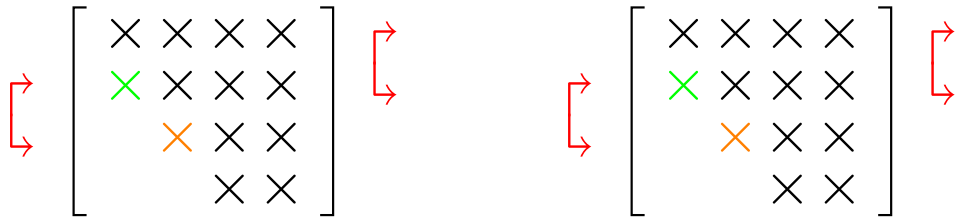
Move of type II:

swapping two adjacent poles

Pole Swapping

Move of type II:

swapping two adjacent poles



Pole Swapping

Move of type II:

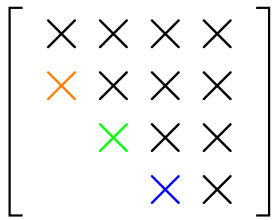
swapping two adjacent poles

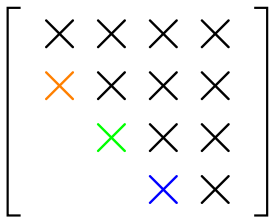
Pole Swapping

A pole swap is an eigenvalue swap!

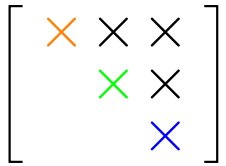
Pole Swapping

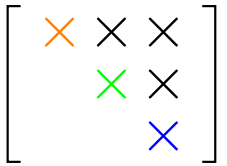
A pole swap is an eigenvalue swap!





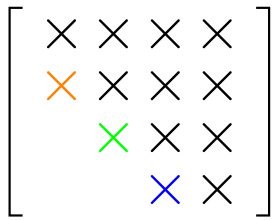
Delete first row and last column to get the *pole pencil*:

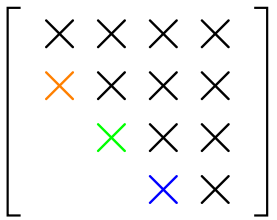




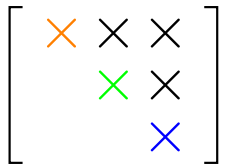
Pole Swapping

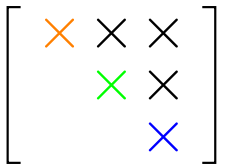
A pole swap is an eigenvalue swap!





Delete first row and last column to get the *pole pencil*:

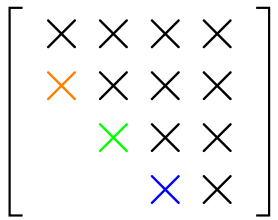


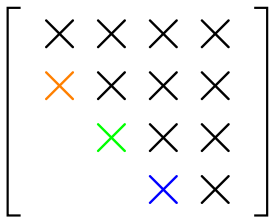


Poles of original pencil are *eigenvalues* of pole pencil.

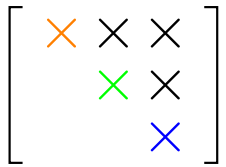
Pole Swapping

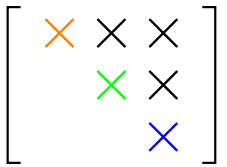
A pole swap is an eigenvalue swap!





Delete first row and last column to get the *pole pencil*:



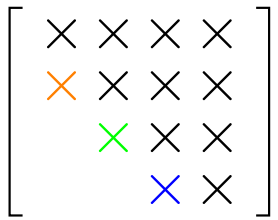


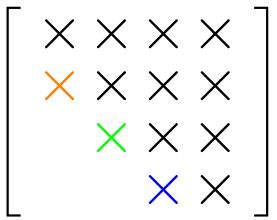
Poles of original pencil are *eigenvalues* of pole pencil.

Swapping poles in original pencil is the same as swapping eigenvalues in pole pencil.

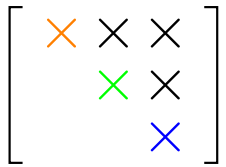
Pole Swapping

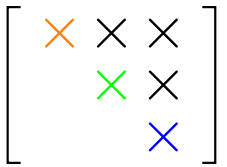
A pole swap is an eigenvalue swap!





Delete first row and last column to get the *pole pencil*:





Poles of original pencil are *eigenvalues* of pole pencil.

Swapping poles in original pencil is the same as swapping eigenvalues in pole pencil. This has been studied.

Pole Swapping

Numerous algorithms can be constructed from these two moves.

Pole Swapping

Numerous algorithms can be constructed from these two moves.

Basic algorithm (rational QZ: Camps, Meerbergen, Vandebril):

Pole Swapping

Numerous algorithms can be constructed from these two moves.

Basic algorithm (rational QZ: Camps, Meerbergen, Vandebril):

- Pick a shift. (as in QZ)

Pole Swapping

Numerous algorithms can be constructed from these two moves.

Basic algorithm (rational QZ: Camps, Meerbergen, Vandebril):

- Pick a shift. (as in QZ)
- Insert the shift as a pole at the top. (type I)

Pole Swapping

Numerous algorithms can be constructed from these two moves.

Basic algorithm (rational QZ: Camps, Meerbergen, Vandebril):

- Pick a shift. (as in QZ)
- Insert the shift as a pole at the top. (type I)
- Swap this new pole with the next pole down, (type II)

Pole Swapping

Numerous algorithms can be constructed from these two moves.

Basic algorithm (rational QZ: Camps, Meerbergen, Vandebril):

- Pick a shift. (as in QZ)
- Insert the shift as a pole at the top. (type I)
- Swap this new pole with the next pole down, (type II) and the next pole,

Pole Swapping

Numerous algorithms can be constructed from these two moves.

Basic algorithm (rational QZ: Camps, Meerbergen, Vandebril):

- Pick a shift. (as in QZ)
- Insert the shift as a pole at the top. (type I)
- Swap this new pole with the next pole down, (type II) and the next pole, and so on until the shift is at the bottom.

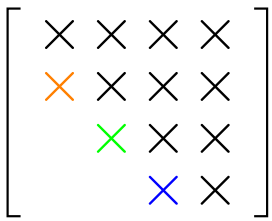
Pole Swapping

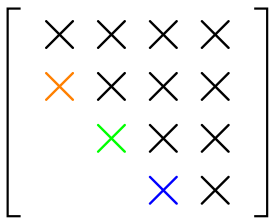
Numerous algorithms can be constructed from these two moves.

Basic algorithm (rational QZ: Camps, Meerbergen, Vandebril):

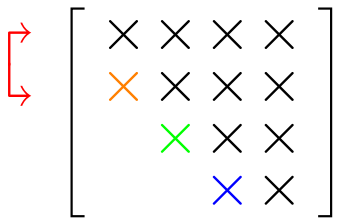
- Pick a shift. (as in QZ)
- Insert the shift as a pole at the top. (type I)
- Swap this new pole with the next pole down, (type II) and the next pole, and so on until the shift is at the bottom.
- Remove the shift from the bottom, inserting any desired pole. (type I)

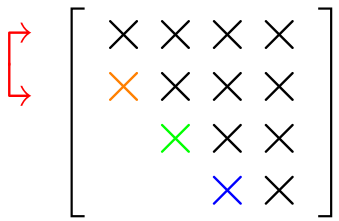
Pole Swapping



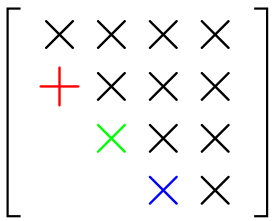


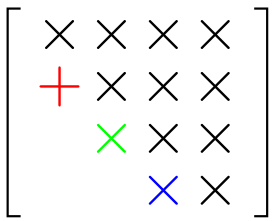
Pole Swapping



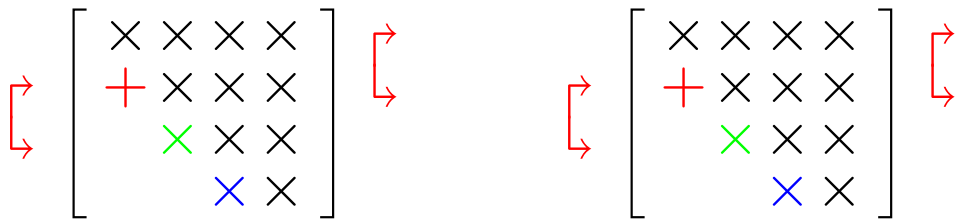


Pole Swapping

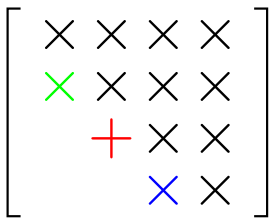


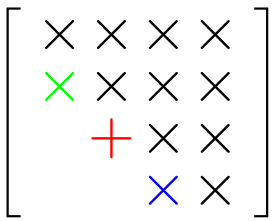


Pole Swapping

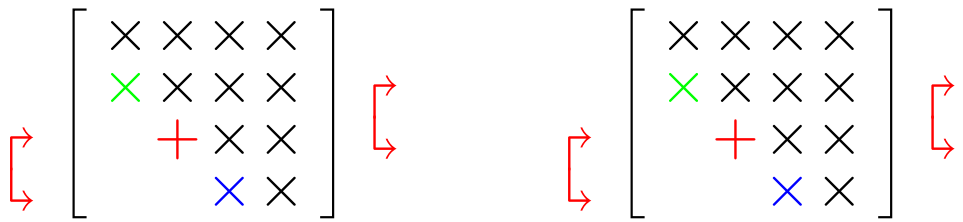


Pole Swapping



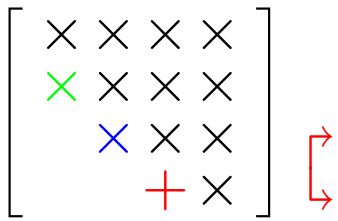


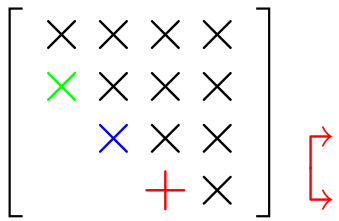
Pole Swapping



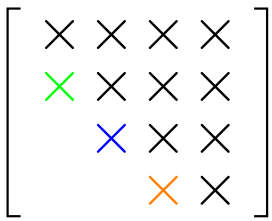
Pole Swapping

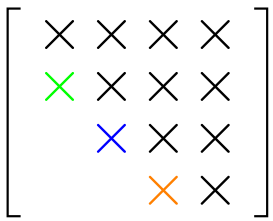
Pole Swapping





Pole Swapping





Pole Swapping

What is this?

Pole Swapping

What is this?

It's a generalization of Moler/Stewart QZ.

Pole Swapping

What is this?

It's a generalization of Moler/Stewart QZ.

QZ is special case where all poles (other than the shift) are ∞.

Pole Swapping

What is this?

It's a generalization of Moler/Stewart QZ.

QZ is special case where all poles (other than the shift) are ∞.

Some variants:

Pole Swapping

What is this?

It's a generalization of Moler/Stewart QZ.

QZ is special case where all poles (other than the shift) are ∞.

Some variants:

- Pick a bunch of shifts and chase them one after the other in tight formation. (level 3 BLAS)

Pole Swapping

What is this?

It's a generalization of Moler/Stewart QZ.

QZ is special case where all poles (other than the shift) are ∞.

Some variants:

- Pick a bunch of shifts and chase them one after the other in tight formation. (level 3 BLAS)
- Chase shifts in both directions at once.

Standard Eigenvalue Problem

We've focused on generalized eigenvalue problem

$$Ax = \lambda Bx,$$

Standard Eigenvalue Problem

We've focused on generalized eigenvalue problem

$$Ax = \lambda Bx,$$

but pole swapping can also be adapted to the standard eigenvalue problem

$$Ax = \lambda Ix.$$

Standard Eigenvalue Problem

We've focused on generalized eigenvalue problem

$$Ax = \lambda Bx,$$

but pole swapping can also be adapted to the standard eigenvalue problem

$$Ax = \lambda Ix.$$

Consider special generalized eigenvalue problem

$$Ax = \lambda Ux \quad \text{with } U \text{ unitary.}$$

Standard Eigenvalue Problem

We've focused on generalized eigenvalue problem

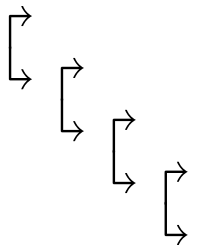
$$Ax = \lambda Bx,$$

but pole swapping can also be adapted to the standard eigenvalue problem

$$Ax = \lambda Ix.$$

Consider special generalized eigenvalue problem

$$Ax = \lambda Ux \quad \text{with } U \text{ unitary.}$$

$U =$ 

Standard Eigenvalue Problem

We've focused on generalized eigenvalue problem

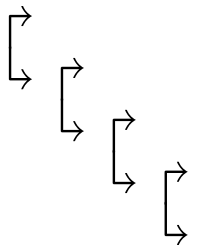
$$Ax = \lambda Bx,$$

but pole swapping can also be adapted to the standard eigenvalue problem

$$Ax = \lambda Ix.$$

Consider special generalized eigenvalue problem

$$Ax = \lambda Ux \quad \text{with } U \text{ unitary.}$$

$U =$ 

The resulting algorithm combines pole swapping with core chasing.

In Conclusion

In Conclusion

Recent discoveries have provided new insights into the Francis and Moler/Stewart algorithms and generalizations.

In Conclusion

Recent discoveries have provided new insights into the Francis and Moler/Stewart algorithms and generalizations.

Methods of implementation include

In Conclusion

Recent discoveries have provided new insights into the Francis and Moler/Stewart algorithms and generalizations.

Methods of implementation include

- bulge chasing (old way)

In Conclusion

Recent discoveries have provided new insights into the Francis and Moler/Stewart algorithms and generalizations.

Methods of implementation include

- bulge chasing (old way)
- core chasing (new way)

In Conclusion

Recent discoveries have provided new insights into the Francis and Moler/Stewart algorithms and generalizations.

Methods of implementation include

- bulge chasing (old way)
- core chasing (new way)
- pole swapping (even newer)

In Conclusion

Recent discoveries have provided new insights into the Francis and Moler/Stewart algorithms and generalizations.

Methods of implementation include

- bulge chasing (old way)
- core chasing (new way)
- pole swapping (even newer)
- or a combination of these.

In Conclusion

Recent discoveries have provided new insights into the Francis and Moler/Stewart algorithms and generalizations.

Methods of implementation include

- bulge chasing (old way)
- core chasing (new way)
- pole swapping (even newer)
- or a combination of these.

Thank you for your attention!