Fast computation of the zeros of a polynomial

David S. Watkins

Department of Mathematics
Washington State University

Valencia, 2012

Collaborators

This is joint work with

- **Raf Vandebril** (KU Leuven)
- **Jared Aurentz** (WSU)

The Problem

- $p(x) = x^n - a_1 x^{n-1} - a_2 x^{n-2} - \cdots - a_n = 0$
- Why on Earth?
- companion matrix

$$A = \begin{bmatrix} a_1 & a_2 & \cdots & a_n \\ 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \quad 0 \end{bmatrix}$$

- Get the zeros of p by computing the eigenvalues of A.

The Problem

- $p(x) = x^n - a_1 x^{n-1} - a_2 x^{n-2} - \cdots - a_n = 0$
- Why on Earth?
- companion matrix

$$A = \begin{bmatrix} a_1 & a_2 & \cdots & a_n \\ 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \quad 0 \end{bmatrix}$$

- Get the zeros of p by computing the eigenvalues of A.

The Problem

- $p(x) = x^n - a_1 x^{n-1} - a_2 x^{n-2} - \cdots - a_n = 0$
- Why on Earth?
- companion matrix

$$A = \begin{bmatrix} a_1 & a_2 & \cdots & a_n \\ 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \quad 0 \end{bmatrix}$$

- Get the zeros of p by computing the eigenvalues of A.

The Problem

- $p(x) = x^n - a_1 x^{n-1} - a_2 x^{n-2} - \cdots - a_n = 0$
- Why on Earth?
- companion matrix

$$A = \begin{bmatrix} a_1 & a_2 & \cdots & & a_n \\ 1 & & & & \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & 0 \end{bmatrix}$$

- Get the zeros of p by computing the eigenvalues of A.

The Problem

- $p(x) = x^n - a_1 x^{n-1} - a_2 x^{n-2} - \cdots - a_n = 0$
- Why on Earth?
- companion matrix

$$A = \begin{bmatrix} a_1 & a_2 & \cdots & & a_n \\ 1 & & & & \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & 0 \end{bmatrix}$$

- Get the zeros of p by computing the eigenvalues of A.

Fiedler factorization (by Gaussian elimination)

$$\begin{bmatrix} a_1 & a_2 & a_3 & a_4 \\ 1 & & & \\ & 1 & & \\ & & 1 & 0 \end{bmatrix}$$

Fiedler factorization (by Gaussian elimination)

$$\begin{bmatrix} a_1 & a_2 & a_3 & a_4 \\ 1 & & & \\ & 1 & & \\ & & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & & & \\ a_1 & a_2 & a_3 & a_4 \\ & 1 & & \\ & & 1 & 0 \end{bmatrix}$$

Fiedler factorization (by Gaussian elimination)

$$\begin{bmatrix} a_1 & a_2 & a_3 & a_4 \\ 1 & & & \\ & 1 & & \\ & & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & & & \\ 0 & a_2 & a_3 & a_4 \\ & 1 & & \\ & & 1 & 0 \end{bmatrix}$$

Fiedler factorization (by Gaussian elimination)

$$\begin{bmatrix} \\ \\ \\ \end{bmatrix} \begin{bmatrix} a_1 & a_2 & a_3 & a_4 \\ 1 & & & \\ & 1 & & \\ & & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & a_2 & a_3 & a_4 \\ & & 1 & 0 \end{bmatrix}$$

Fiedler factorization (by Gaussian elimination)

$$\begin{bmatrix} \\ \\ \\ \end{bmatrix} \begin{bmatrix} a_1 & a_2 & a_3 & a_4 \\ 1 & & & \\ & 1 & & \\ & & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & 0 & a_3 & a_4 \\ & & 1 & 0 \end{bmatrix}$$

Fiedler factorization (by Gaussian elimination)

$$\begin{bmatrix} a_1 & a_2 & a_3 & a_4 \\ 1 & & & \\ & 1 & & \\ & & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & a_3 & a_4 \end{bmatrix}$$

Fiedler factorization (by Gaussian elimination)

$$\begin{bmatrix} a_1 & a_2 & a_3 & a_4 \\ 1 & & & \\ & 1 & & \\ & & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & 0 & a_4 \end{bmatrix}$$

Fiedler factorization (by Gaussian elimination)

$$\begin{bmatrix} a_1 & a_2 & a_3 & a_4 \\ 1 & & & \\ & 1 & & \\ & & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & 1 \end{bmatrix}$$

Fiedler factorization (by Gaussian elimination)

- Thus

$$\begin{bmatrix} a_1 & a_2 & a_3 & a_4 \\ 1 & & & \\ & 1 & & \\ & & 1 & 0 \end{bmatrix} = \quad \begin{matrix} \uparrow \\ \downarrow \end{matrix} \quad \begin{matrix} \uparrow \\ \downarrow \end{matrix} \quad \begin{matrix} \uparrow \\ \downarrow \end{matrix}$$

Fiedler factorization (by Gaussian elimination)

- Factors are

$$\begin{bmatrix} a_1 & 1 & & \\ 1 & 0 & & \\ & & 1 & \\ & & & 1 \end{bmatrix} \begin{bmatrix} 1 & & & \\ & a_2 & 1 & \\ & 1 & 0 & \\ & & & 1 \end{bmatrix} \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & a_3 & a_4 \\ & & 1 & 0 \end{bmatrix}$$

- An $n \times n$ matrix has $n - 1$ factors.
- We will use this factored form.

Fiedler factorization (by Gaussian elimination)

- Factors are

$$\begin{bmatrix} a_1 & 1 & & \\ 1 & 0 & & \\ & & 1 & \\ & & & 1 \end{bmatrix} \begin{bmatrix} 1 & & & \\ & a_2 & 1 & \\ & 1 & 0 & \\ & & & 1 \end{bmatrix} \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & a_3 & a_4 \\ & & 1 & 0 \end{bmatrix}$$

- An $n \times n$ matrix has $n - 1$ factors.
- We will use this factored form.

Fiedler factorization (by Gaussian elimination)

- Factors are

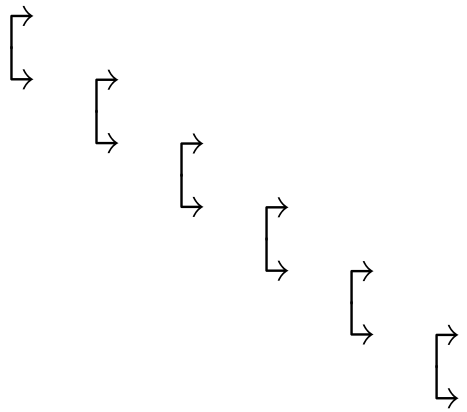
$$\begin{bmatrix} a_1 & 1 & & \\ 1 & 0 & & \\ & & 1 & \\ & & & 1 \end{bmatrix} \begin{bmatrix} 1 & & & \\ & a_2 & 1 & \\ & 1 & 0 & \\ & & & 1 \end{bmatrix} \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & a_3 & a_4 \\ & & 1 & 0 \end{bmatrix}$$

- An $n \times n$ matrix has $n - 1$ factors.
- We will use this factored form.

A variety of orderings

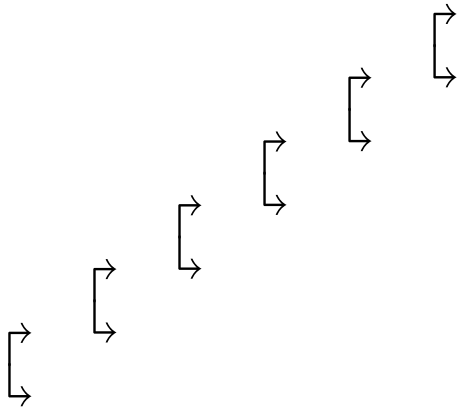
A variety of orderings

Upper Hessenberg



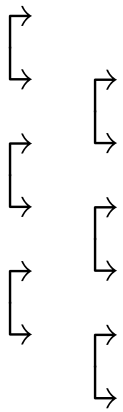
A variety of orderings

Lower Hessenberg



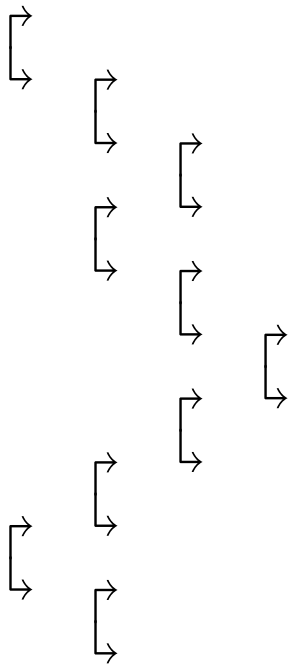
A variety of orderings

so called CMV



A variety of orderings

Random



A variety of orderings

- Our algorithms can be applied to any of these orderings.
- Factors don't have to come from a companion matrix.
- We consider upper Hessenberg case for simplicity.

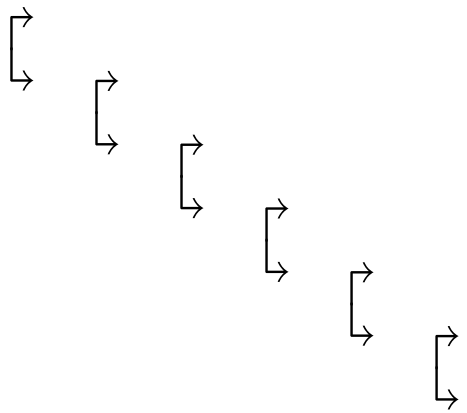
A variety of orderings

- Our algorithms can be applied to any of these orderings.
- Factors don't have to come from a companion matrix.
- We consider upper Hessenberg case for simplicity.

A variety of orderings

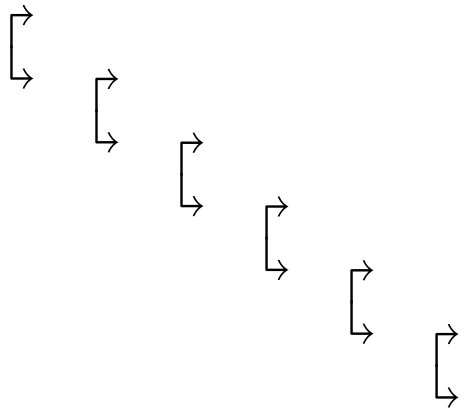
- Our algorithms can be applied to any of these orderings.
- Factors don't have to come from a companion matrix.
- We consider upper Hessenberg case for simplicity.

Upper Hessenberg matrix



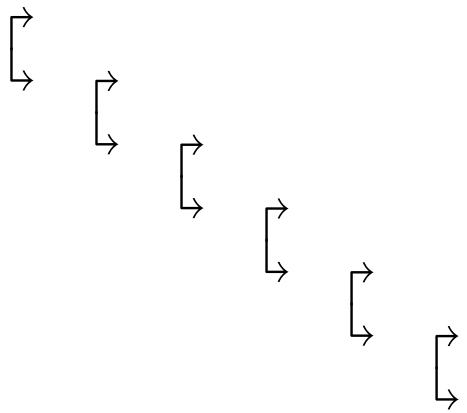
- Matrix is stored in $O(n)$ space.
- Can we find eigenvalues in $O(n^2)$ time?

Upper Hessenberg matrix



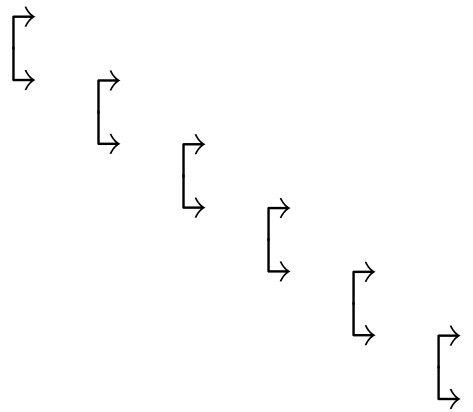
- Matrix is stored in $O(n)$ space.
- Can we find eigenvalues in $O(n^2)$ time?

Upper Hessenberg matrix



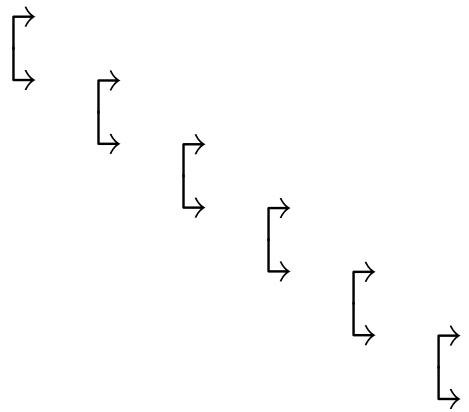
- Matrix is stored in $O(n)$ space.
- Can we find eigenvalues in $O(n^2)$ time?

Inspiration from the unitary eigenvalue problem



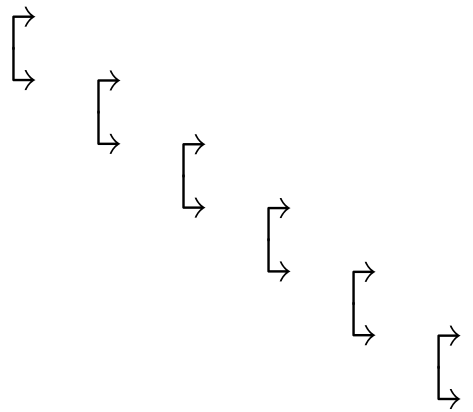
- Every unitary upper Hessenberg matrix . . .
- Factors are unitary
- QR algorithm ($O(n^2)$ time)
- Gragg (1986)
- Ammar, Reichel, M. Stewart, Bunse-Gerstner, Elsner, . . .

Inspiration from the unitary eigenvalue problem



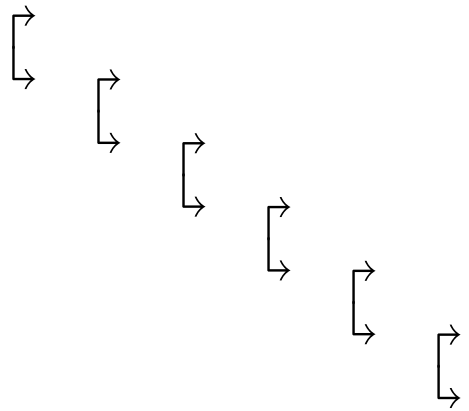
- Every unitary upper Hessenberg matrix . . .
- Factors are unitary
- QR algorithm ($O(n^2)$ time)
- Gragg (1986)
- Ammar, Reichel, M. Stewart, Bunse-Gerstner, Elsner, . . .

Inspiration from the unitary eigenvalue problem



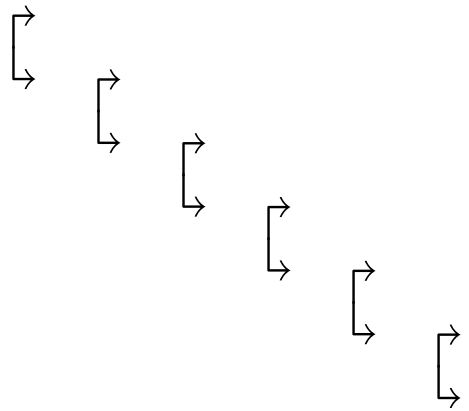
- Every unitary upper Hessenberg matrix . . .
- Factors are unitary
- QR algorithm ($O(n^2)$ time)
- Gragg (1986)
- Ammar, Reichel, M. Stewart, Bunse-Gerstner, Elsner, . . .

Inspiration from the unitary eigenvalue problem



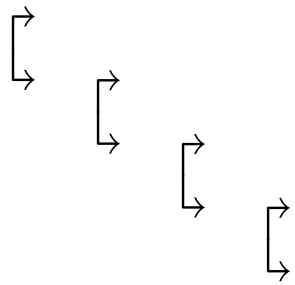
- Every unitary upper Hessenberg matrix . . .
- Factors are unitary
- QR algorithm ($O(n^2)$ time)
- Gragg (1986)
- Ammar, Reichel, M. Stewart, Bunse-Gerstner, Elsner, . . .

Inspiration from the unitary eigenvalue problem



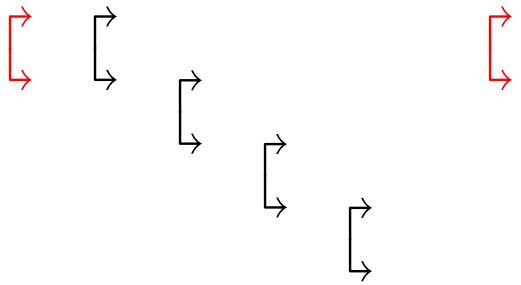
- Every unitary upper Hessenberg matrix . . .
- Factors are unitary
- QR algorithm ($O(n^2)$ time)
- Gragg (1986)
- Ammar, Reichel, M. Stewart, Bunse-Gerstner, Elsner, . . .

Francis algorithm on a unitary Hessenberg matrix



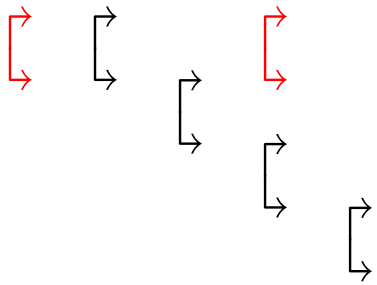
- iteration of degree one for simplicity

Francis algorithm on a unitary Hessenberg matrix



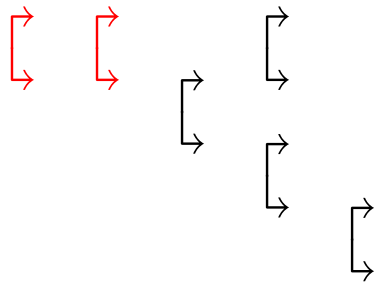
- iteration of degree one for simplicity

Francis algorithm on a unitary Hessenberg matrix



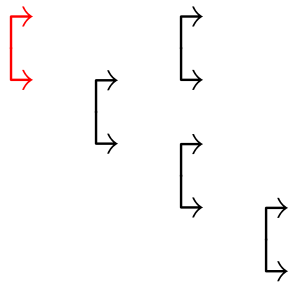
- iteration of degree one for simplicity

Francis algorithm on a unitary Hessenberg matrix



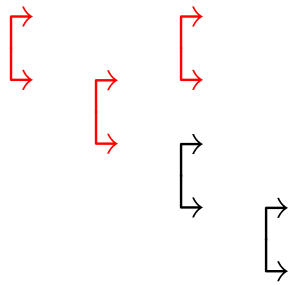
- iteration of degree one for simplicity

Francis algorithm on a unitary Hessenberg matrix



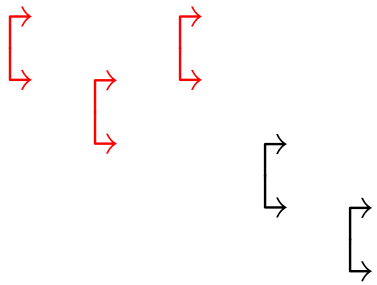
- iteration of degree one for simplicity

Francis algorithm on a unitary Hessenberg matrix



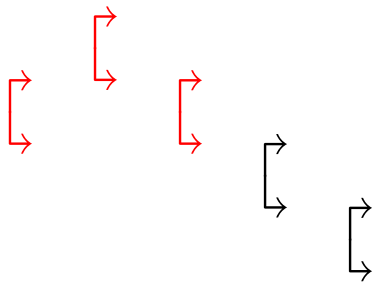
- iteration of degree one for simplicity

Francis algorithm on a unitary Hessenberg matrix



- iteration of degree one for simplicity

Francis algorithm on a unitary Hessenberg matrix



- iteration of degree one for simplicity

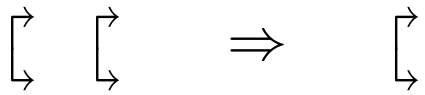
Allowed Operations

- fusion

- turn over

Allowed Operations

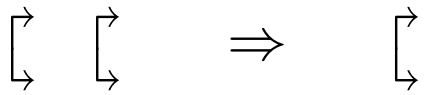
- fusion



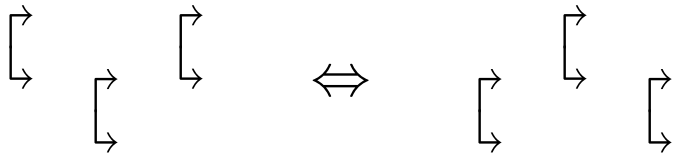
- turn over

Allowed Operations

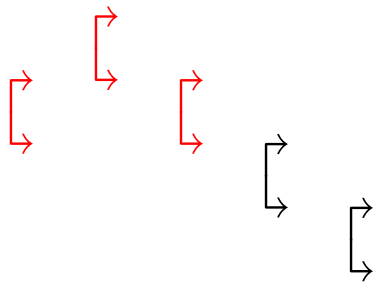
- fusion



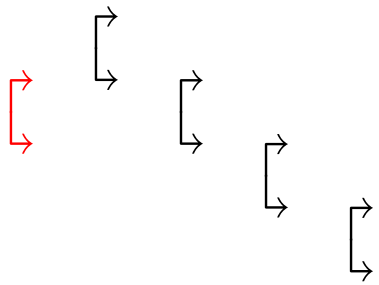
- turn over



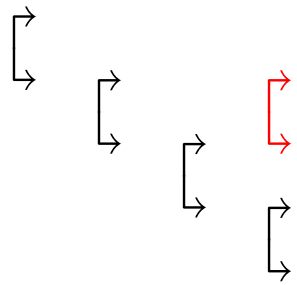
Francis algorithm on a unitary Hessenberg matrix



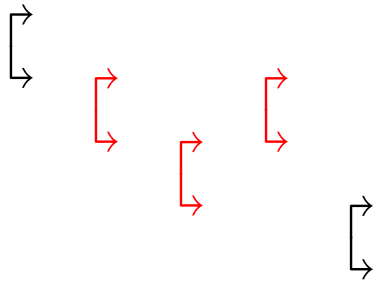
Francis algorithm on a unitary Hessenberg matrix



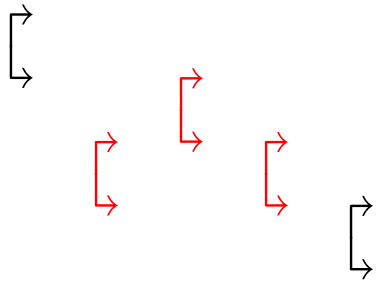
Francis algorithm on a unitary Hessenberg matrix



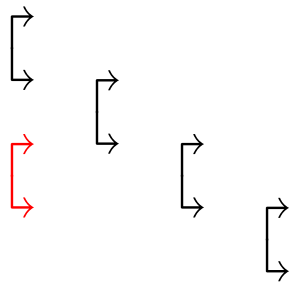
Francis algorithm on a unitary Hessenberg matrix



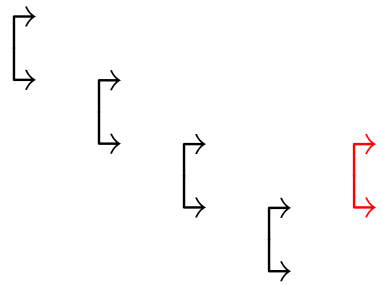
Francis algorithm on a unitary Hessenberg matrix



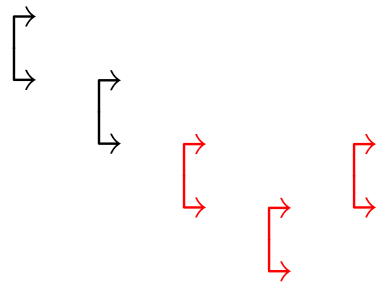
Francis algorithm on a unitary Hessenberg matrix



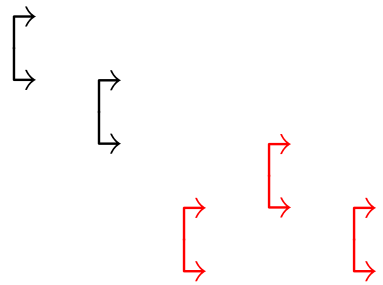
Francis algorithm on a unitary Hessenberg matrix



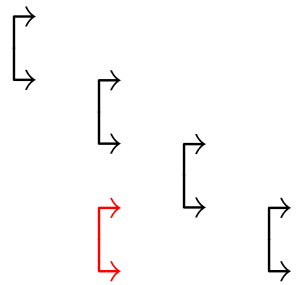
Francis algorithm on a unitary Hessenberg matrix



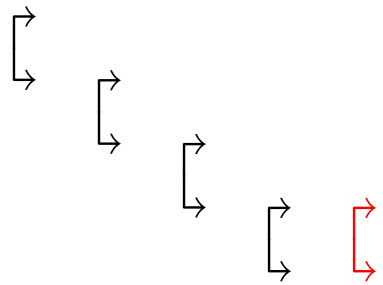
Francis algorithm on a unitary Hessenberg matrix



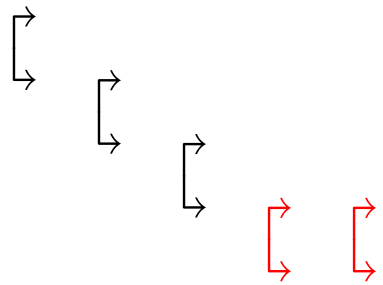
Francis algorithm on a unitary Hessenberg matrix



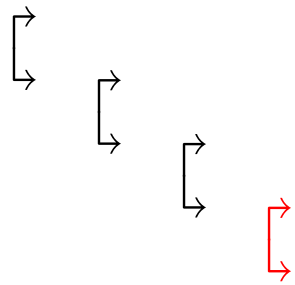
Francis algorithm on a unitary Hessenberg matrix



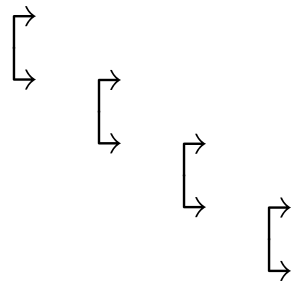
Francis algorithm on a unitary Hessenberg matrix



Francis algorithm on a unitary Hessenberg matrix



Francis algorithm on a unitary Hessenberg matrix



- Done!

Francis algorithm on a unitary Hessenberg matrix

- memory requirement: $O(n)$
- work per iteration: $O(n)$
- Total work: $O(n^2)$
- Can we do something like this with the companion matrix?

Francis algorithm on a unitary Hessenberg matrix

- memory requirement: $O(n)$
- work per iteration: $O(n)$
- Total work: $O(n^2)$
- Can we do something like this with the companion matrix?

Francis algorithm on a unitary Hessenberg matrix

- memory requirement: $O(n)$
- work per iteration: $O(n)$
- Total work: $O(n^2)$
- Can we do something like this with the companion matrix?

Francis algorithm on a unitary Hessenberg matrix

- memory requirement: $O(n)$
- work per iteration: $O(n)$
- Total work: $O(n^2)$
- Can we do something like this with the companion matrix?

Francis algorithm on a unitary Hessenberg matrix

- memory requirement: $O(n)$
- work per iteration: $O(n)$
- Total work: $O(n^2)$
- Can we do something like this with the companion matrix?

Companion Matrix Factorization

$$\begin{bmatrix} a_1 & a_2 & a_3 & a_4 & a_5 \\ 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & 1 & 0 \end{bmatrix} = \quad \cdots$$

- . . . but these factors are not unitary.
- Need non-unitary variant of Francis's algorithm.
- Why on Earth? Just for fun!

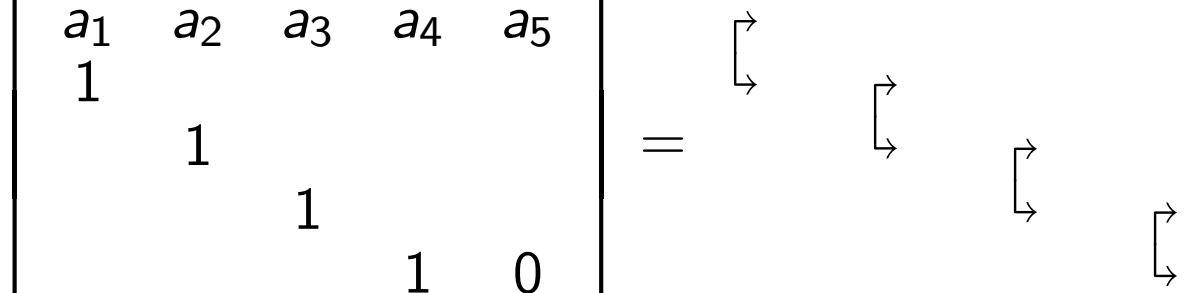
Companion Matrix Factorization

$$\begin{bmatrix} a_1 & a_2 & a_3 & a_4 & a_5 \\ 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & 1 & 0 \end{bmatrix} = \quad \cdots$$

- . . . but these factors are not unitary.
- Need non-unitary variant of Francis's algorithm.
- Why on Earth? Just for fun!

Companion Matrix Factorization

$$\begin{bmatrix} a_1 & a_2 & a_3 & a_4 & a_5 \\ 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & 1 & 0 \end{bmatrix} =$$



- . . . but these factors are not unitary.
- Need non-unitary variant of Francis's algorithm.
- Why on Earth? Just for fun!

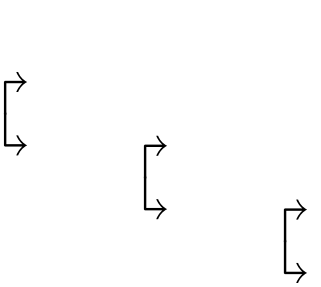
Companion Matrix Factorization

$$\begin{bmatrix} a_1 & a_2 & a_3 & a_4 & a_5 \\ 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & 1 & 0 \end{bmatrix} = \quad \cdots$$

- . . . but these factors are not unitary.
- Need non-unitary variant of Francis's algorithm.
- Why on Earth? Just for fun!

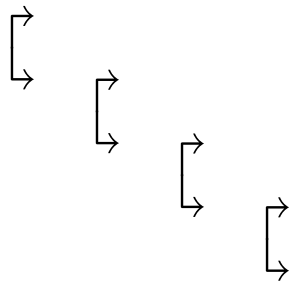
Companion Matrix Factorization

$$\begin{bmatrix} a_1 & a_2 & a_3 & a_4 & a_5 \\ 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & 1 & 0 \end{bmatrix} =$$

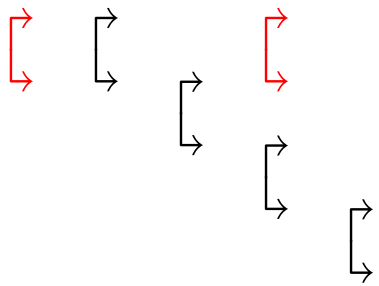


- . . . but these factors are not unitary.
- Need non-unitary variant of Francis's algorithm.
- Why on Earth? Just for fun!

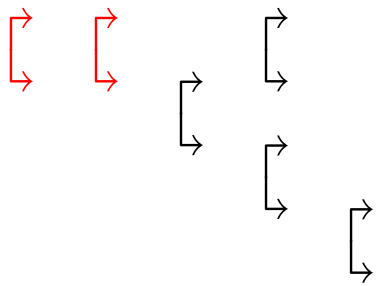
Nonunitary Francis-like algorithm (first try)



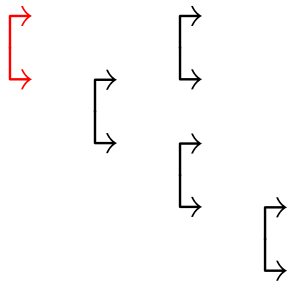
Nonunitary Francis-like algorithm (first try)



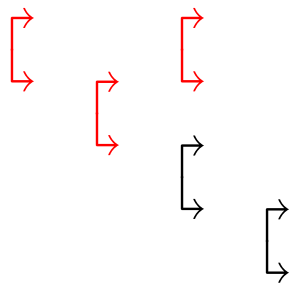
Nonunitary Francis-like algorithm (first try)



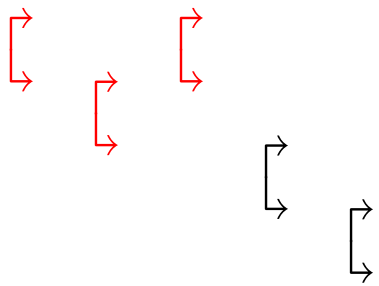
Nonunitary Francis-like algorithm (first try)



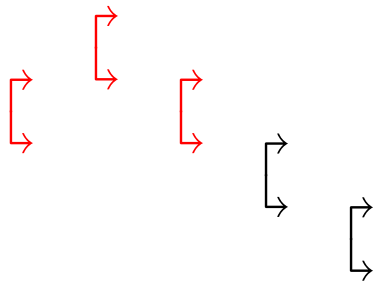
Nonunitary Francis-like algorithm (first try)



Nonunitary Francis-like algorithm (first try)

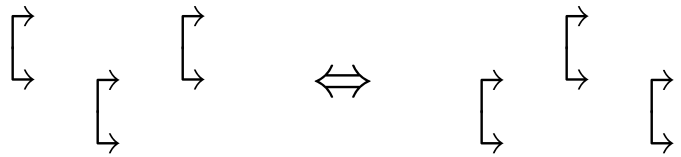


Nonunitary Francis-like algorithm (first try)



Nonunitary Francis-like algorithm (first try)

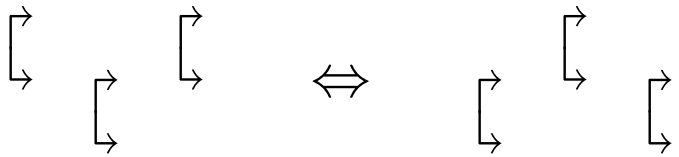
- Can we do this?



- crucial question
- Pessimistic answer: No! (not always)
- Optimistic answer: Yes! (almost always)

Nonunitary Francis-like algorithm (first try)

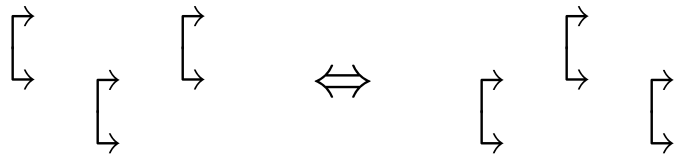
- Can we do this?



- crucial question
- Pessimistic answer: No! (not always)
- Optimistic answer: Yes! (almost always)

Nonunitary Francis-like algorithm (first try)

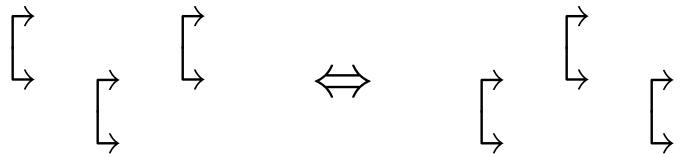
- Can we do this?



- crucial question
- Pessimistic answer: No! (not always)
- Optimistic answer: Yes! (almost always)

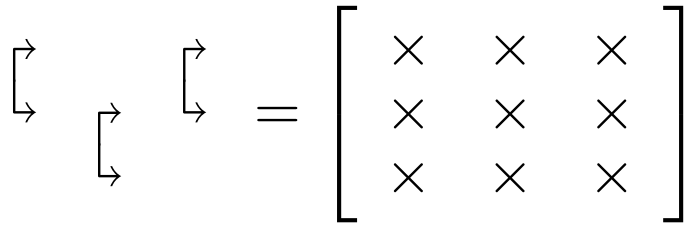
Nonunitary Francis-like algorithm (first try)

- Can we do this?



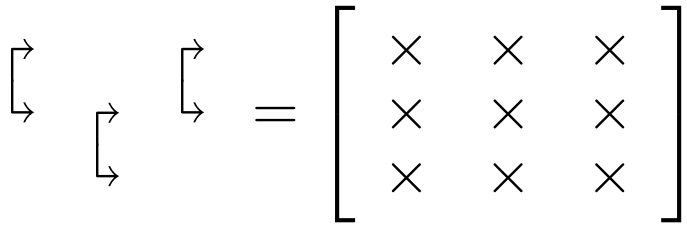
- crucial question
- Pessimistic answer: No! (not always)
- Optimistic answer: Yes! (almost always)

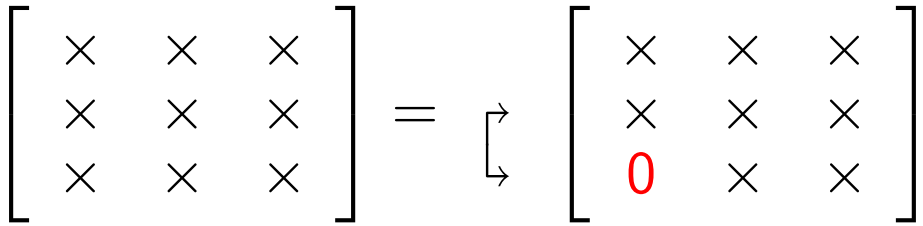
Nonunitary Turnover Operation



Need

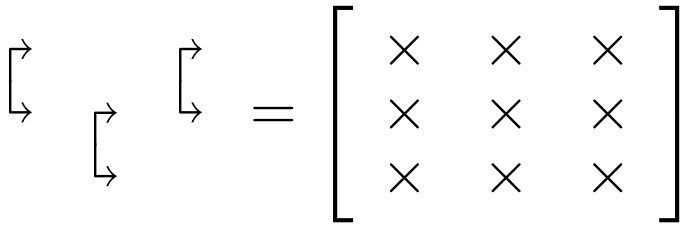
Nonunitary Turnover Operation

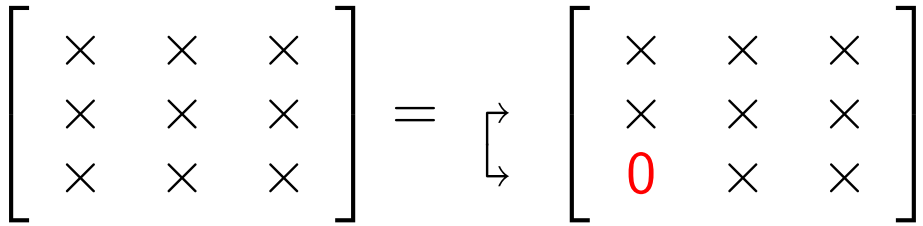




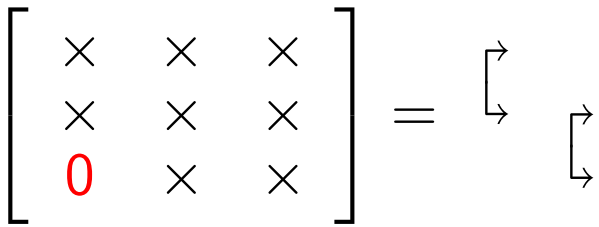
Need

Nonunitary Turnover Operation





Need



Nonunitary Turnover Operation

$$\begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ 0 & \times & \times \end{bmatrix} = \quad \begin{matrix} \hookrightarrow \\ \hookrightarrow \end{matrix}$$

. . . is possibile iff 2×2 submatrix has rank one.

Nonunitary Turnover Operation

$$\begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ 0 & \times & \times \end{bmatrix} = \quad \begin{array}{c} \curvearrowright \\ \curvearrowright \end{array}$$

. . . is possibile iff 2×2 submatrix has rank one.

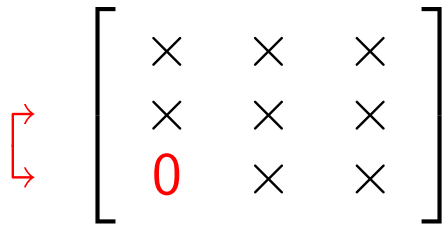
Nonunitary Turnover Operation

- procedure with corrective Gauss transform
- (works almost always)

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

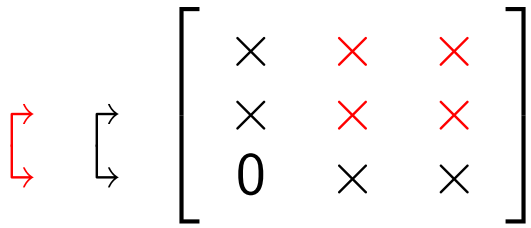
Nonunitary Turnover Operation

- procedure with corrective Gauss transform
- (works almost always)



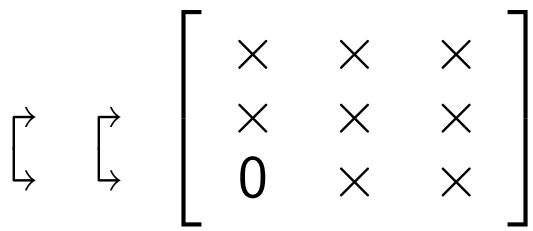
Nonunitary Turnover Operation

- procedure with corrective Gauss transform
- (works almost always)



Nonunitary Turnover Operation

- procedure with corrective Gauss transform
- (works almost always)



Nonunitary Turnover Operation

- procedure with corrective Gauss transform
- (works almost always)

$$\hookrightarrow \begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ 0 & \times & \times \end{bmatrix}$$

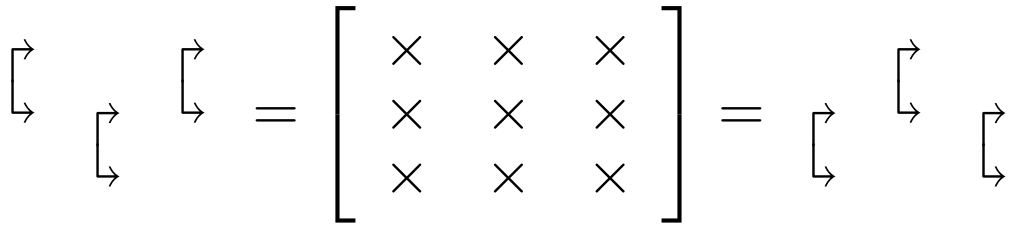
Nonunitary Turnover Operation

- procedure with corrective Gauss transform
- (works almost always)

$$\hookrightarrow \begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ 0 & \times & \times \end{bmatrix}$$

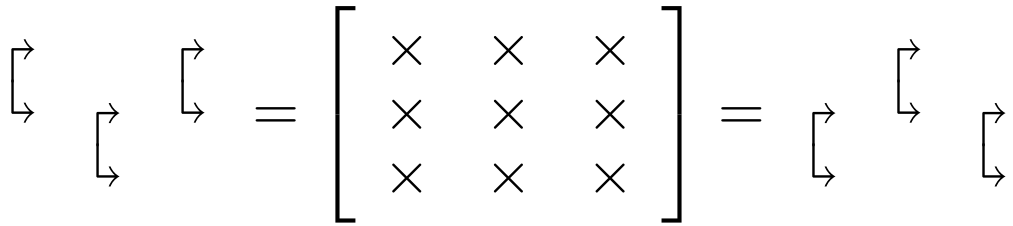
- Option: do both at once.

Nonunitary Turnover Operation



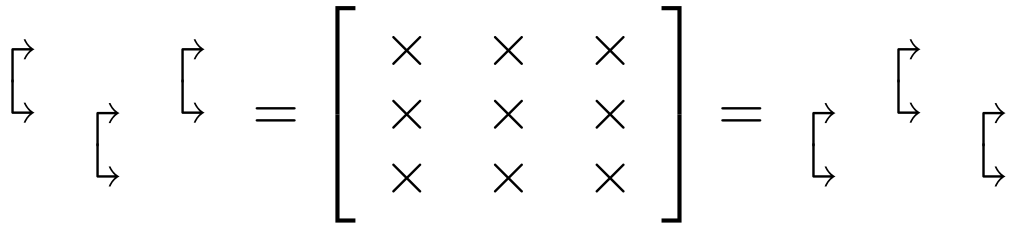
- yields a non-unitary variant of Francis's algorithm
- $O(n)$ space and $O(n^2)$ time
- it's pretty fast
- . . . but we had a better idea.

Nonunitary Turnover Operation



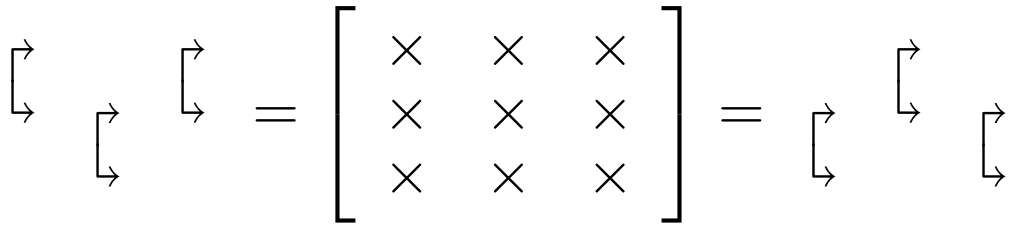
- yields a non-unitary variant of Francis's algorithm
- $O(n)$ space and $O(n^2)$ time
- it's pretty fast
- . . . but we had a better idea.

Nonunitary Turnover Operation



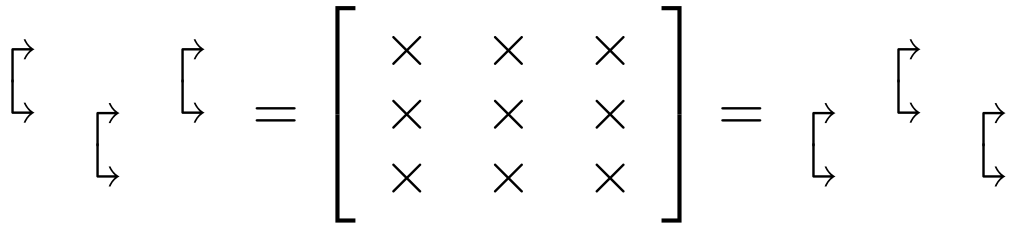
- yields a non-unitary variant of Francis's algorithm
- $O(n)$ space and $O(n^2)$ time
- it's pretty fast
- . . . but we had a better idea.

Nonunitary Turnover Operation



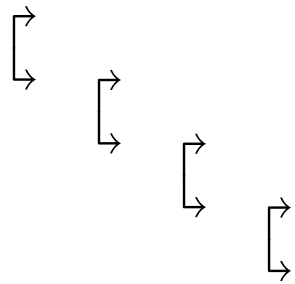
- yields a non-unitary variant of Francis's algorithm
- $O(n)$ space and $O(n^2)$ time
- it's pretty fast
- . . . but we had a better idea.

Nonunitary Turnover Operation

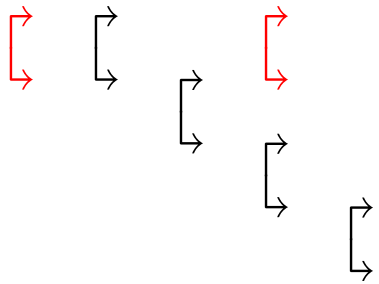


- yields a non-unitary variant of Francis's algorithm
- $O(n)$ space and $O(n^2)$ time
- it's pretty fast
- . . . but we had a better idea.

Nonunitary Francis-like algorithm (second try)

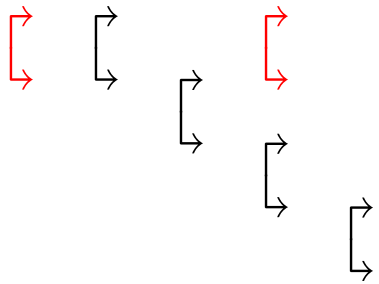


Nonunitary Francis-like algorithm (second try)



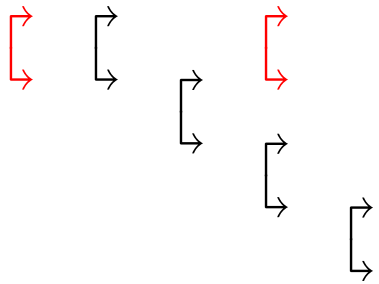
- First transformation doesn't have to be unitary.
- Use a Gauss transformation instead.
- Gauss transform = unit lower triangular
- Thanks, Jared

Nonunitary Francis-like algorithm (second try)



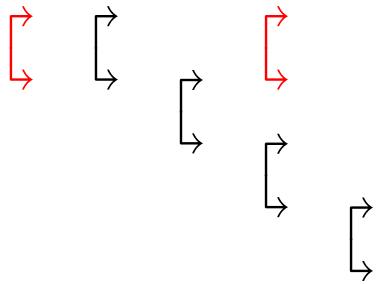
- First transformation doesn't have to be unitary.
- Use a Gauss transformation instead.
- Gauss transform = unit lower triangular
- Thanks, Jared

Nonunitary Francis-like algorithm (second try)



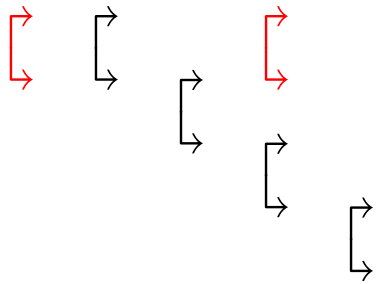
- First transformation doesn't have to be unitary.
- Use a Gauss transformation instead.
- Gauss transform = unit lower triangular
- Thanks, Jared

Nonunitary Francis-like algorithm (second try)



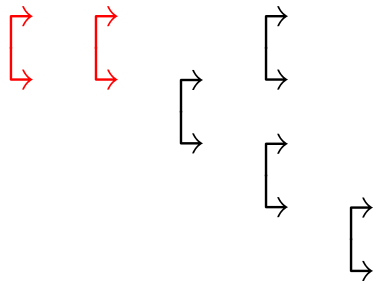
- First transformation doesn't have to be unitary.
- Use a Gauss transformation instead.
- Gauss transform = unit lower triangular
- Thanks, Jared

Nonunitary Francis-like algorithm (second try)

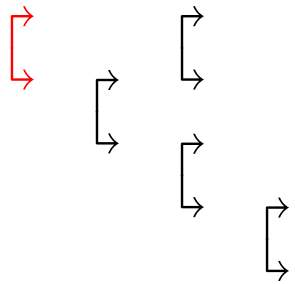


- First transformation doesn't have to be unitary.
- Use a Gauss transformation instead.
- Gauss transform = unit lower triangular
- Thanks, Jared

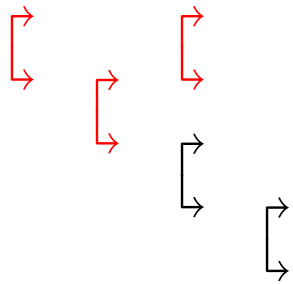
Nonunitary Francis-like algorithm (second try)



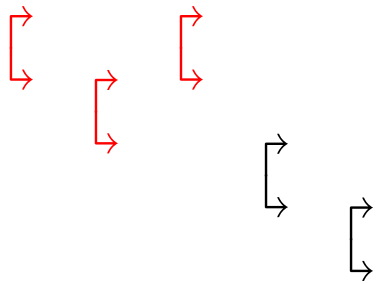
Nonunitary Francis-like algorithm (second try)



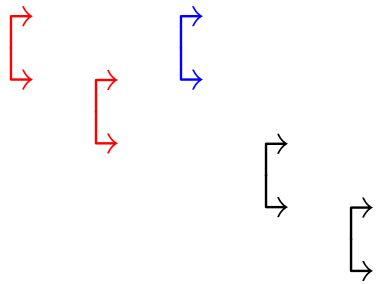
Nonunitary Francis-like algorithm (second try)



Nonunitary Francis-like algorithm (second try)

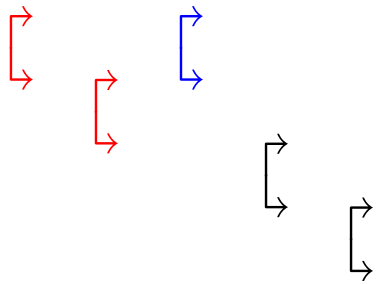


Nonunitary Francis-like algorithm (second try)



- Blue transform is a Gauss transform.

Nonunitary Francis-like algorithm (second try)



- Blue transform is a Gauss transform.
- Can we exploit this?

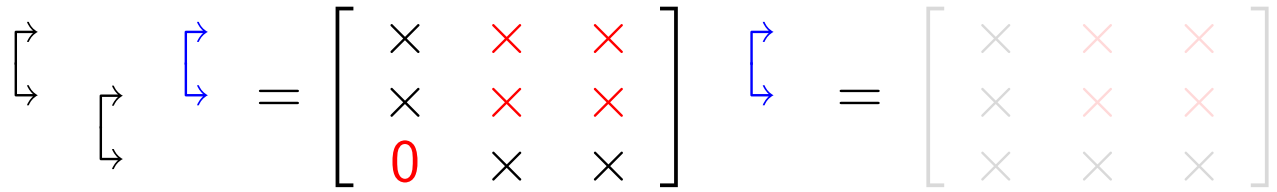
Nonunitary Francis-like algorithm (second try)

- Yes!

- submatrix still has rank one!

Nonunitary Francis-like algorithm (second try)

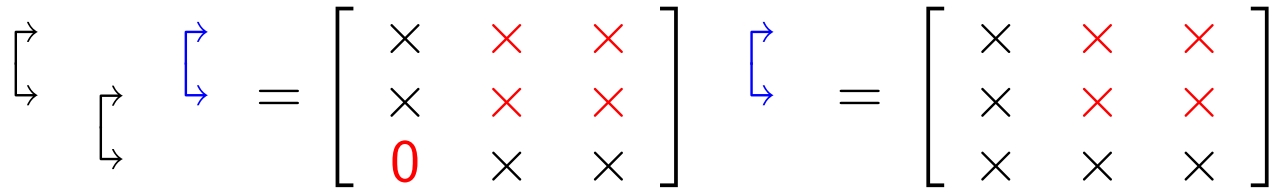
- Yes!



- submatrix still has rank one!

Nonunitary Francis-like algorithm (second try)

- Yes!



- submatrix still has rank one!

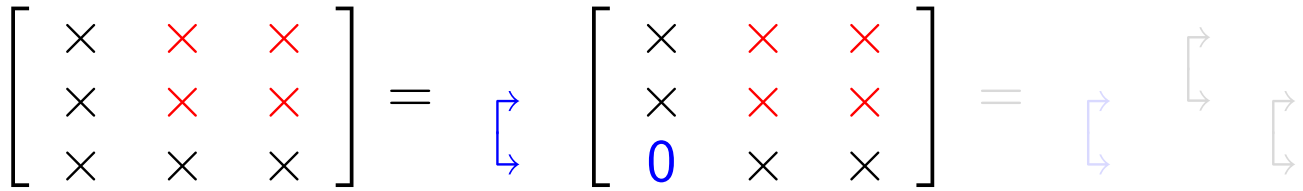
Nonunitary Francis-like algorithm (second try)

- So . . .

- (Blue) Gauss transform does not disturb 2×2 submatrix.

Nonunitary Francis-like algorithm (second try)

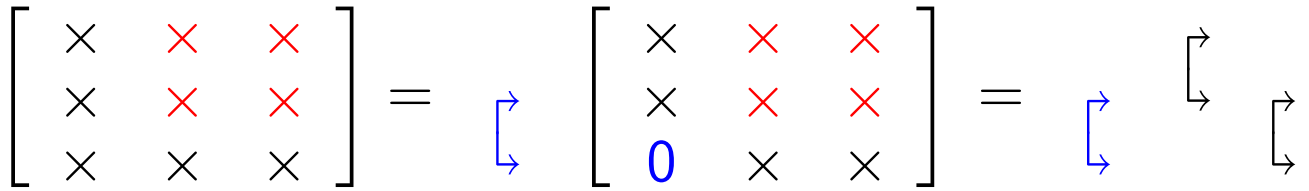
- So ...



- (Blue) Gauss transform does not disturb 2×2 submatrix.

Nonunitary Francis-like algorithm (second try)

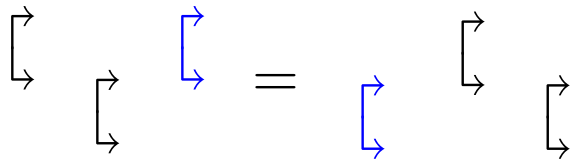
- So ...



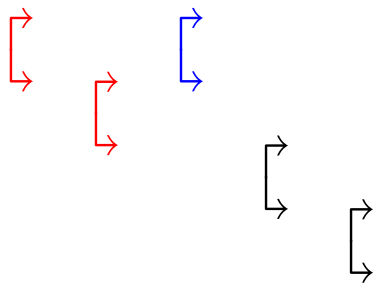
- (Blue) Gauss transform does not disturb 2×2 submatrix.

Nonunitary Francis-like algorithm (second try)

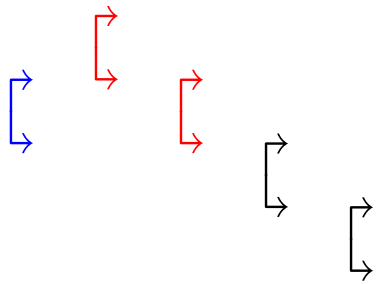
- Complete turnover looks like this:



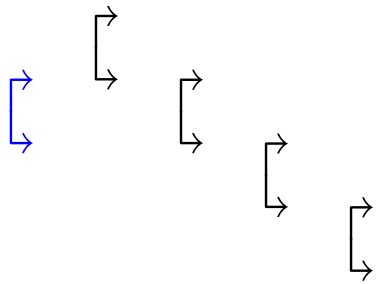
Nonunitary Francis-like algorithm (second try)



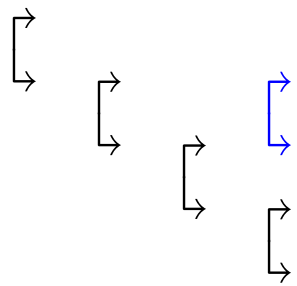
Nonunitary Francis-like algorithm (second try)



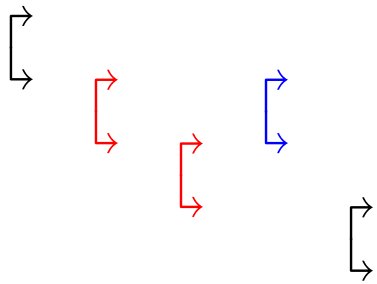
Nonunitary Francis-like algorithm (second try)



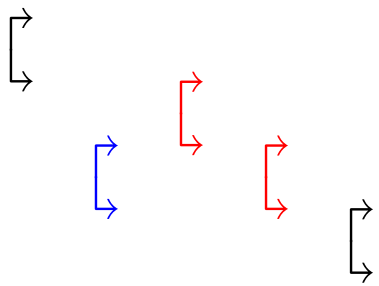
Nonunitary Francis-like algorithm (second try)



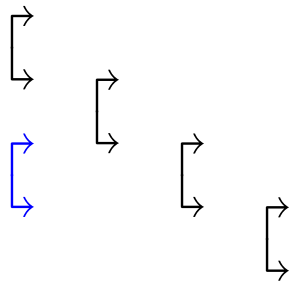
Nonunitary Francis-like algorithm (second try)



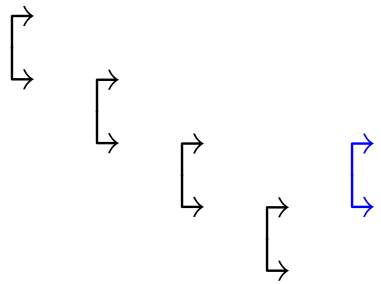
Nonunitary Francis-like algorithm (second try)



Nonunitary Francis-like algorithm (second try)



Nonunitary Francis-like algorithm (second try)



Nonunitary Francis-like algorithm (second try)

- . . . and so on.
- This is much faster!
- It is a structured, implicitly-shifted LR algorithm.

Nonunitary Francis-like algorithm (second try)

- . . . and so on.
- This is much faster!
- It is a structured, implicitly-shifted LR algorithm.

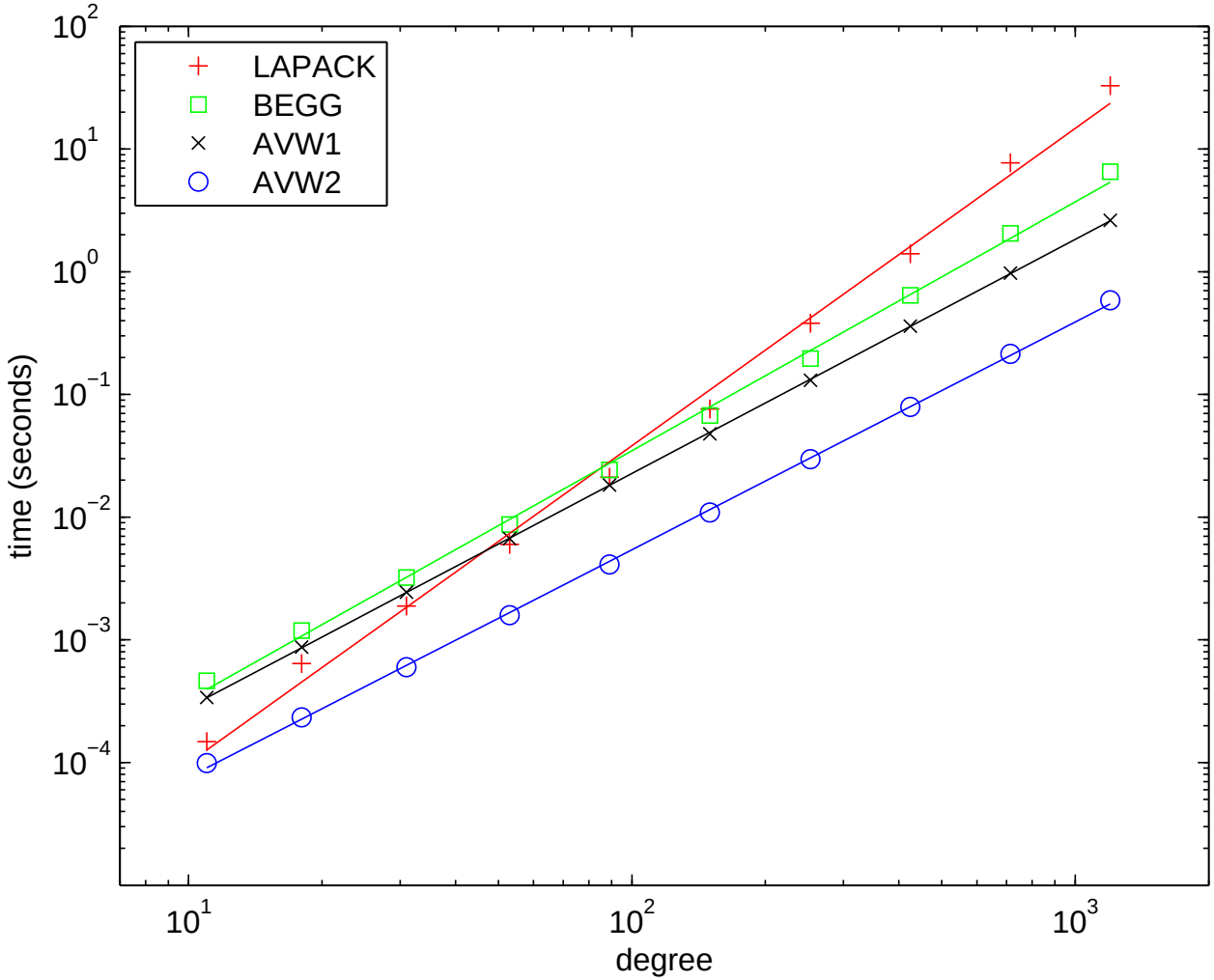
Nonunitary Francis-like algorithm (second try)

- . . . and so on.
- This is much faster!
- It is a structured, implicitly-shifted LR algorithm.

Contestants

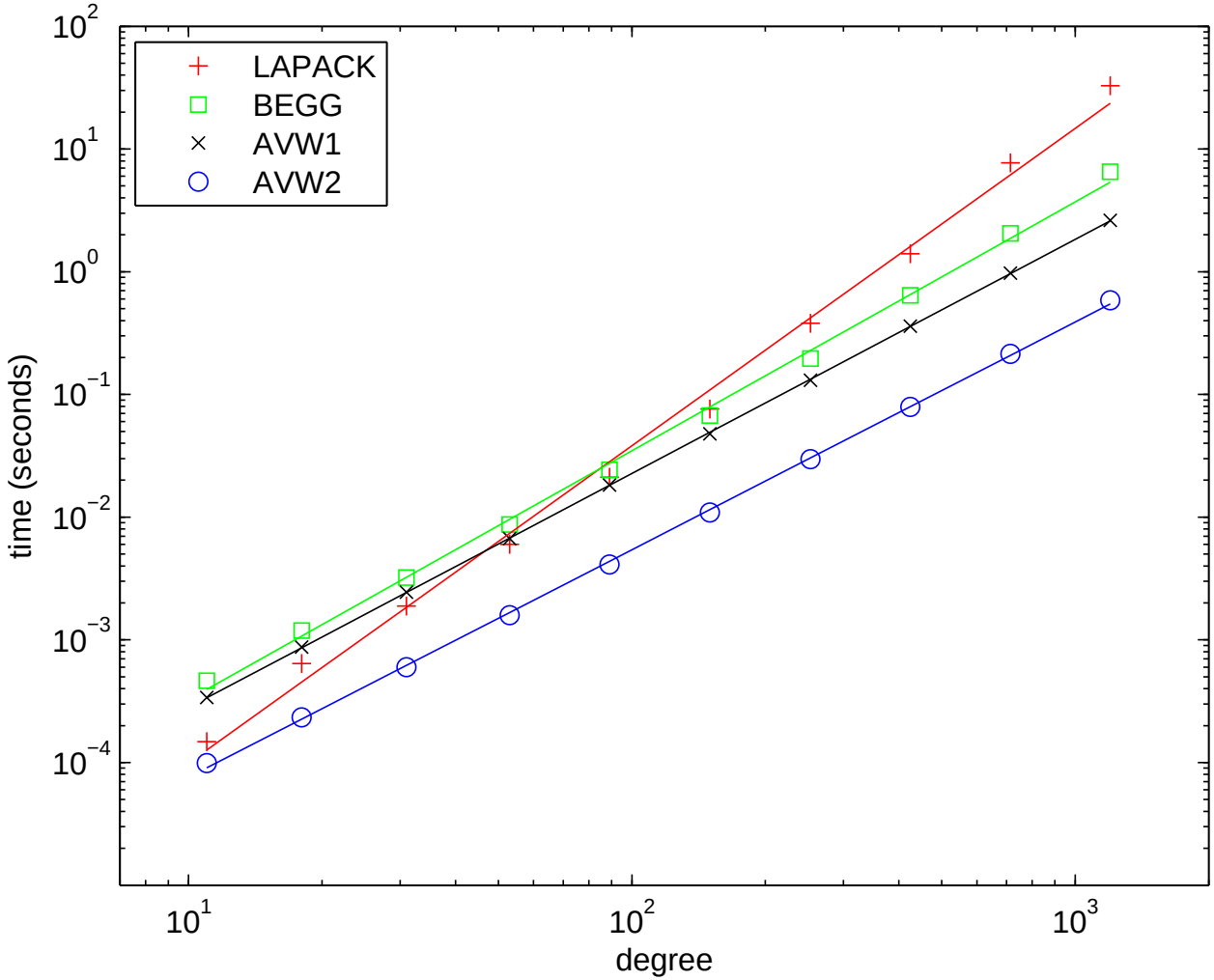
- LAPACK code ZHSEQR ($O(n^3)$)
- BBEGG (fast, unitary, one of the better codes)
- AVW1 (our first attempt)
- AVW2 (our second attempt)

Speed Comparison



No crossover point!

Speed Comparison



No crossover point!

Speed Comparison

- At $n = 1200$:

method	time
LAPACK	32.7
BBEGG	6.5
AVW1	2.6
AVW2	0.6

Final Words

- Our code is lightning fast, but . . .
- how robust is it?
- Answer: Not as robust as we'd like.
- We can't recommend it for general use, but . . .
- it must be good for something.
- (good initial guesses for almost free)

Final Words

- Our code is lightning fast, but . . .
- how robust is it?
- Answer: Not as robust as we'd like.
- We can't recommend it for general use, but . . .
- it must be good for something.
- (good initial guesses for almost free)

Final Words

- Our code is lightning fast, but . . .
- how robust is it?
- Answer: Not as robust as we'd like.
- We can't recommend it for general use, but . . .
- it must be good for something.
- (good initial guesses for almost free)

Final Words

- Our code is lightning fast, but . . .
- how robust is it?
- Answer: Not as robust as we'd like.
- We can't recommend it for general use, but . . .
- it must be good for something.
- (good initial guesses for almost free)

Final Words

- Our code is lightning fast, but . . .
- how robust is it?
- Answer: Not as robust as we'd like.
- We can't recommend it for general use, but . . .
- it must be good for something.
- (good initial guesses for almost free)

Final Words

- Our code is lightning fast, but . . .
- how robust is it?
- Answer: Not as robust as we'd like.
- We can't recommend it for general use, but . . .
- it must be good for something.
- (good initial guesses for almost free)

Final Words

- Our code is lightning fast, but . . .
- how robust is it?
- Answer: Not as robust as we'd like.
- We can't recommend it for general use, but . . .
- it must be good for something.
- (good initial guesses for almost free)

Thank you for your attention.