Fast Stable Computation of the Eigenvalues of Companion Matrices

David S. Watkins

Department of Mathematics
Washington State University

Spa, Belgium, 2014

Collaborators

This is joint work with

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

Collaborators

This is joint work with

- **Jared Aurentz** (Oxford)
- **Thomas Mach** (KU Leuven)
- **Raf Vandebril** (KU Leuven)

The Problem

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

The Problem

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

Cost

- If structure not exploited:
  - $O(n^2)$ storage, $O(n^3)$ flops
  - Francis's implicitly-shifted QR algorithm
- If structure exploited:
  - $O(n)$ storage, $O(n^2)$ flops
  - data-sparse representation + Francis's algorithm
  - Several methods proposed
  - including some by us (lightning fast but not stable)

Cost

- If structure not exploited:
  - $O(n^2)$ storage, $O(n^3)$ flops
  - Francis's implicitly-shifted QR algorithm
- If structure exploited:
  - $O(n)$ storage, $O(n^2)$ flops
  - data-sparse representation + Francis's algorithm
  - Several methods proposed
  - including some by us (lightning fast but not stable)

Cost

- If structure not exploited:
  - $O(n^2)$ storage, $O(n^3)$ flops
  - Francis's implicitly-shifted QR algorithm
- If structure exploited:
  - $O(n)$ storage, $O(n^2)$ flops
  - data-sparse representation + Francis's algorithm
  - Several methods proposed
  - including some by us (lightning fast but not stable)

Cost

- If structure not exploited:
  - $O(n^2)$ storage, $O(n^3)$ flops
  - Francis's implicitly-shifted QR algorithm
- If structure exploited:
  - $O(n)$ storage, $O(n^2)$ flops
  - data-sparse representation + Francis's algorithm
  - Several methods proposed
  - including some by us (lightning fast but not stable)

Cost

- If structure not exploited:
  - $O(n^2)$ storage, $O(n^3)$ flops
  - Francis's implicitly-shifted QR algorithm
- If structure exploited:
  - $O(n)$ storage, $O(n^2)$ flops
  - data-sparse representation + Francis's algorithm
  - Several methods proposed
  - including some by us (lightning fast but not stable)

Some of the Competitors

- Chandrasekaran, Gu, Xia, Zhu (2007)
- Bini, Boito, Eidelman, Gemignani, Gohberg (2010)
- Boito, Eidelman, Gemignani, Gohberg (2012)

- Fortran codes available
- quasiseparable generator representation
- We will do something else.
- evidence of backward stability

Some of the Competitors

- Chandrasekaran, Gu, Xia, Zhu (2007)
- Bini, Boito, Eidelman, Gemignani, Gohberg (2010)
- Boito, Eidelman, Gemignani, Gohberg (2012)

- Fortran codes available
- quasiseparable generator representation
- We will do something else.
- evidence of backward stability

Some of the Competitors

- Chandrasekaran, Gu, Xia, Zhu (2007)
- Bini, Boito, Eidelman, Gemignani, Gohberg (2010)
- Boito, Eidelman, Gemignani, Gohberg (2012)

- Fortran codes available
- quasiseparable generator representation
- We will do something else.
- evidence of backward stability

Some of the Competitors

- Chandrasekaran, Gu, Xia, Zhu (2007)
- Bini, Boito, Eidelman, Gemignani, Gohberg (2010)
- Boito, Eidelman, Gemignani, Gohberg (2012)

- Fortran codes available
- quasiseparable generator representation
- We will do something else.
- evidence of backward stability

Some of the Competitors

- Chandrasekaran, Gu, Xia, Zhu (2007)
- Bini, Boito, Eidelman, Gemignani, Gohberg (2010)
- Boito, Eidelman, Gemignani, Gohberg (2012)

- Fortran codes available
- quasiseparable generator representation
- We will do something else.
- evidence of backward stability

Our Contribution

We present

- Yet another $O(n)$ representation
- Francis algorithm in $O(n)$ flops/iteration
- Fortran codes (we're faster)
- normwise backward stable (We can prove it.)

Our Contribution

We present

- Yet another $O(n)$ representation
- Francis algorithm in $O(n)$ flops/iteration
- Fortran codes (we're faster)
- normwise backward stable (We can prove it.)

Our Contribution

We present

- Yet another $O(n)$ representation
- Francis algorithm in $O(n)$ flops/iteration
- Fortran codes (we're faster)
- normwise backward stable (We can prove it.)

Our Contribution

We present

- Yet another $O(n)$ representation
- Francis algorithm in $O(n)$ flops/iteration
- Fortran codes (we're faster)
- normwise backward stable (We can prove it.)

Structure

- Companion matrix is unitary-plus-rank-one

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

- preserved by unitary similarities
- Companion matrix is also upper Hessenberg.
- preserved by Francis algorithm
- We exploit this structure.

Structure

- Companion matrix is unitary-plus-rank-one

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

- preserved by unitary similarities
- Companion matrix is also upper Hessenberg.
- preserved by Francis algorithm
- We exploit this structure.

Structure

- Companion matrix is unitary-plus-rank-one

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

- preserved by unitary similarities
- Companion matrix is also upper Hessenberg.
- preserved by Francis algorithm
- We exploit this structure.

Structure

- Companion matrix is unitary-plus-rank-one

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

- preserved by unitary similarities
- Companion matrix is also upper Hessenberg.
- preserved by Francis algorithm
- We exploit this structure.

Structure

- Companion matrix is unitary-plus-rank-one

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

- preserved by unitary similarities
- Companion matrix is also upper Hessenberg.
- preserved by Francis algorithm
- We exploit this structure.

Structure

- Chandrasekaran, Gu, Xia, Zhu (2007)
- $A = QR$
- Q is upper Hessenberg and unitary.
- R is upper triangular and unitary-plus-rank-one.
- We do this too.

Structure

- Chandrasekaran, Gu, Xia, Zhu (2007)
- $A = QR$
- Q is upper Hessenberg and unitary.
- R is upper triangular and unitary-plus-rank-one.
- We do this too.

Structure

- Chandrasekaran, Gu, Xia, Zhu (2007)
- $A = QR$
- Q is upper Hessenberg and unitary.
- R is upper triangular and unitary-plus-rank-one.
- We do this too.

The Unitary Part

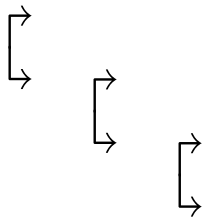
$$\begin{bmatrix} x & x & x & x \\ x & x & x & x \\ & x & x & x \\ & & x & x \end{bmatrix} = \begin{bmatrix} x & x & & \\ x & x & & \\ & & 1 & \\ & & & 1 \end{bmatrix} \begin{bmatrix} 1 & & & \\ & x & x & \\ & x & x & \\ & & & 1 \end{bmatrix} \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & x & x \\ & & x &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

$Q =$

- $O(n)$ storage

The Unitary Part

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

$Q =$ 

- $O(n)$ storage

The Unitary Part

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

$$Q = \quad \begin{matrix} \cdot & & \\ & \cdot & \\ & & \cdot \end{matrix}$$

- $O(n)$ storage

The Upper Triangular Part

- $R = U + xy^T$ unitary-plus-rank-one, so
- R has quasiseparable rank 2.

$$R = \left[\begin{array}{ccc|ccc} x & \cdots & x & x & \cdots & x \\ & \ddots & \vdots & \vdots & & \vdots \\ & & x & x & \cdots & x \\ \hline & & & x & \cdots & x \\ & & & & \ddots & \vdots \\ & & & & & x \end{array}\right]$$

- quasiseparable generator representation ($O(n)$ storage)
- Chandrasekaran et. al. exploit this structure.
- We do it differently.

The Upper Triangular Part

- $R = U + xy^T$ unitary-plus-rank-one, so
- R has quasiseparable rank 2.

$$R = \left[\begin{array}{ccc|ccc} x & \cdots & x & x & \cdots & x \\ & \ddots & \vdots & \vdots & & \vdots \\ & & x & x & \cdots & x \\ \hline & & & x & \cdots & x \\ & & & & \ddots & \vdots \\ & & & & & x \end{array}\right]$$

- quasiseparable generator representation ($O(n)$ storage)
- Chandrasekaran et. al. exploit this structure.
- We do it differently.

The Upper Triangular Part

- $R = U + xy^T$ unitary-plus-rank-one, so
- R has quasiseparable rank 2.

$$R = \left[\begin{array}{ccc|ccc} x & \cdots & x & x & \cdots & x \\ & \ddots & \vdots & \vdots & & \vdots \\ & & x & x & \cdots & x \\ \hline & & & x & \cdots & x \\ & & & & \ddots & \vdots \\ & & & & & x \end{array}\right]$$

- quasiseparable generator representation ($O(n)$ storage)
- Chandrasekaran et. al. exploit this structure.
- We do it differently.

The Upper Triangular Part

- $R = U + xy^T$ unitary-plus-rank-one, so
- R has quasiseparable rank 2.

$$R = \left[\begin{array}{ccc|ccc} x & \cdots & x & x & \cdots & x \\ & \ddots & \vdots & \vdots & & \vdots \\ & & x & x & \cdots & x \\ \hline & & & x & \cdots & x \\ & & & & \ddots & \vdots \\ & & & & & x \end{array}\right]$$

- quasiseparable generator representation ($O(n)$ storage)
- Chandrasekaran et. al. exploit this structure.
- We do it differently.

The Upper Triangular Part

- $R = U + xy^T$ unitary-plus-rank-one, so
- R has quasiseparable rank 2.

$$R = \left[\begin{array}{ccc|ccc} x & \cdots & x & x & \cdots & x \\ & \ddots & \vdots & \vdots & & \vdots \\ & & x & x & \cdots & x \\ \hline & & & x & \cdots & x \\ & & & & \ddots & \vdots \\ & & & & & x \end{array}\right]$$

- quasiseparable generator representation ($O(n)$ storage)
- Chandrasekaran et. al. exploit this structure.
- We do it differently.

Our Representation

- Add a row/column for extra wiggle room

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

- Extra zero root can be deflated immediately.
- $A = QR$, where

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

Our Representation

- Add a row/column for extra wiggle room

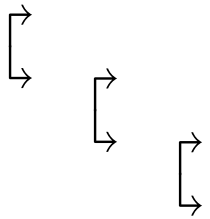
$$A = \left[\begin{array}{ccc|c} 0 & & -a_0 & 1 \\ 1 & & -a_1 & 0 \\ & \ddots & \vdots & \vdots \\ & 1 & -a_{n-1} & 0 \\ \hline & & 0 & 0 \end{array}\right]$$

- Extra zero root can be deflated immediately.
- $A = QR$, where

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

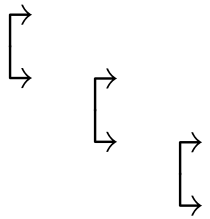
Our Representation

$$Q = \left[\begin{array}{cccc|c} 0 & & & \pm 1 & 0 \\ 1 & & & 0 & 0 \\ & \ddots & & \vdots & \vdots \\ & & 1 & 0 & 0 \\ \hline & & & 0 & 1 \end{array}\right]$$

- Q is stored in factored form
- Q $=$ 
- $Q = Q_1 Q_2 \cdots Q_{n-1}$

Our Representation

$$Q = \left[\begin{array}{cccc|c} 0 & & & \pm 1 & 0 \\ 1 & & & 0 & 0 \\ & \ddots & & \vdots & \vdots \\ & & 1 & 0 & 0 \\ \hline & & & 0 & 1 \end{array}\right]$$

- Q is stored in factored form
- Q = 
- $Q = Q_1 Q_2 \cdots Q_{n-1}$

Our Representation

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

- R is unitary-plus-rank-one:

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

Representation of R

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

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

- Next step: Roll up x.

Representation of R

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

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

- Next step: Roll up x.

Representation of R

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

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

- Next step: Roll up x.

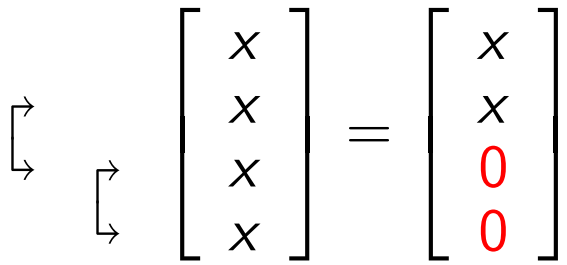
Representation of R

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

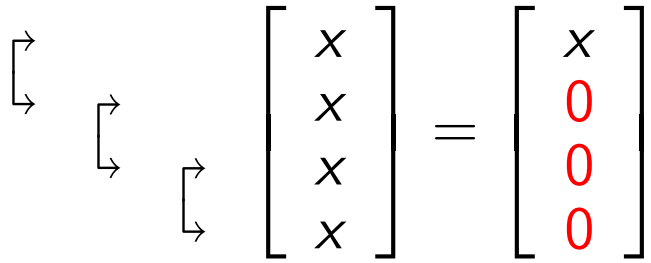
Representation of R

$$\hookrightarrow \begin{bmatrix} x \\ x \\ x \\ x \end{bmatrix} = \begin{bmatrix} x \\ x \\ x \\ 0 \end{bmatrix}$$

Representation of R



Representation of R



Representation of R

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

$$C_1 \cdots C_{n-1} C_n x = \alpha e_1 \qquad \text{(w.l.g. } \alpha = 1\text{)}$$

Representation of R

- $C_1 \cdots C_{n-1} C_n x = e_1$
- $Cx = e_1$
- $C^* e_1 = x$
- $R = U + xy^T = U + C^* e_1 y^T = C^*(CU + e_1 y^T)$
- $R = C^*(B + e_1 y^T)$
- B is upper Hessenberg (and unitary) so $B = B_1 \cdots B_n$.
- $R = C^*(B + e_1 y^T) = C_n^* \cdots C_1^*(B_1 \cdots B_n + e_1 y^T)$
- $O(n)$ storage
- Bonus: Redundancy! No need to keep track of y.

Representation of R

- $C_1 \cdots C_{n-1} C_n x = e_1$
- $Cx = e_1$
- $C^* e_1 = x$
- $R = U + xy^T = U + C^* e_1 y^T = C^*(CU + e_1 y^T)$
- $R = C^*(B + e_1 y^T)$
- B is upper Hessenberg (and unitary) so $B = B_1 \cdots B_n$.
- $R = C^*(B + e_1 y^T) = C_n^* \cdots C_1^*(B_1 \cdots B_n + e_1 y^T)$
- $O(n)$ storage
- Bonus: Redundancy! No need to keep track of y.

Representation of R

- $C_1 \cdots C_{n-1} C_n x = e_1$
- $Cx = e_1$
- $C^* e_1 = x$
- $R = U + xy^T = U + C^* e_1 y^T = C^*(CU + e_1 y^T)$
- $R = C^*(B + e_1 y^T)$
- B is upper Hessenberg (and unitary) so $B = B_1 \cdots B_n$.
- $R = C^*(B + e_1 y^T) = C_n^* \cdots C_1^*(B_1 \cdots B_n + e_1 y^T)$
- $O(n)$ storage
- Bonus: Redundancy! No need to keep track of y.

Representation of R

- $C_1 \cdots C_{n-1} C_n x = e_1$
- $Cx = e_1$
- $C^* e_1 = x$
- $R = U + xy^T = U + C^* e_1 y^T = C^*(CU + e_1 y^T)$
- $R = C^*(B + e_1 y^T)$
- B is upper Hessenberg (and unitary) so $B = B_1 \cdots B_n$.
- $R = C^*(B + e_1 y^T) = C_n^* \cdots C_1^*(B_1 \cdots B_n + e_1 y^T)$
- $O(n)$ storage
- Bonus: Redundancy! No need to keep track of y.

Representation of R

- $C_1 \cdots C_{n-1} C_n x = e_1$
- $Cx = e_1$
- $C^* e_1 = x$
- $R = U + xy^T = U + C^* e_1 y^T = C^*(CU + e_1 y^T)$
- $R = C^*(B + e_1 y^T)$
- B is upper Hessenberg (and unitary) so $B = B_1 \cdots B_n$.
- $R = C^*(B + e_1 y^T) = C_n^* \cdots C_1^*(B_1 \cdots B_n + e_1 y^T)$
- $O(n)$ storage
- Bonus: Redundancy! No need to keep track of y.

Representation of R

- $C_1 \cdots C_{n-1} C_n x = e_1$
- $Cx = e_1$
- $C^* e_1 = x$
- $R = U + xy^T = U + C^* e_1 y^T = C^*(CU + e_1 y^T)$
- $R = C^*(B + e_1 y^T)$
- B is upper Hessenberg (and unitary) so $B = B_1 \cdots B_n$.
- $R = C^*(B + e_1 y^T) = C_n^* \cdots C_1^*(B_1 \cdots B_n + e_1 y^T)$
- $O(n)$ storage
- Bonus: Redundancy! No need to keep track of y.

Representation of R

- $C_1 \cdots C_{n-1} C_n x = e_1$
- $Cx = e_1$
- $C^* e_1 = x$
- $R = U + xy^T = U + C^* e_1 y^T = C^*(CU + e_1 y^T)$
- $R = C^*(B + e_1 y^T)$
- B is upper Hessenberg (and unitary) so $B = B_1 \cdots B_n$.
- $R = C^*(B + e_1 y^T) = C_n^* \cdots C_1^*(B_1 \cdots B_n + e_1 y^T)$
- $O(n)$ storage
- Bonus: Redundancy! No need to keep track of y.

Representation of R

- $C_1 \cdots C_{n-1} C_n x = e_1$
- $Cx = e_1$
- $C^* e_1 = x$
- $R = U + xy^T = U + C^* e_1 y^T = C^*(CU + e_1 y^T)$
- $R = C^*(B + e_1 y^T)$
- B is upper Hessenberg (and unitary) so $B = B_1 \cdots B_n$.
- $R = C^*(B + e_1 y^T) = C_n^* \cdots C_1^*(B_1 \cdots B_n + e_1 y^T)$
- $O(n)$ storage
- Bonus: Redundancy! No need to keep track of y.

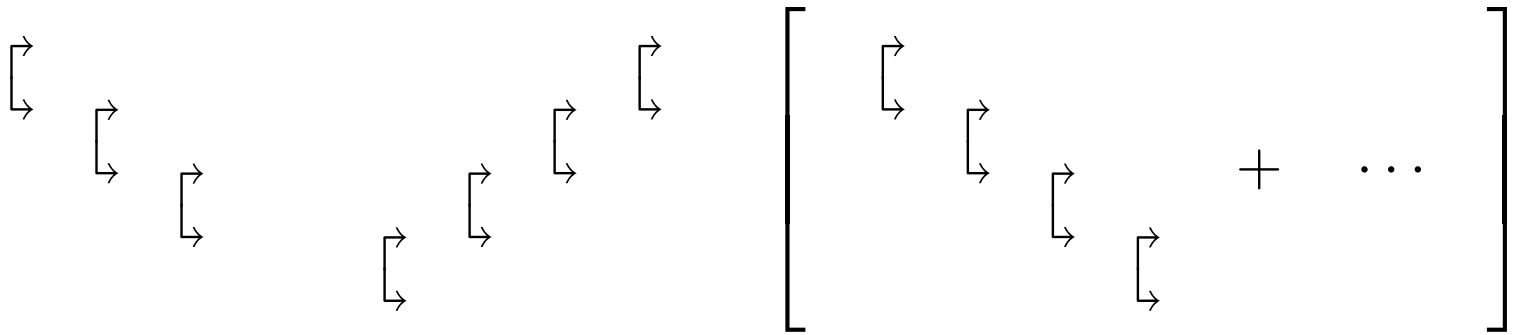
Representation of R

- $C_1 \cdots C_{n-1} C_n x = e_1$
- $Cx = e_1$
- $C^* e_1 = x$
- $R = U + xy^T = U + C^* e_1 y^T = C^*(CU + e_1 y^T)$
- $R = C^*(B + e_1 y^T)$
- B is upper Hessenberg (and unitary) so $B = B_1 \cdots B_n$.
- $R = C^*(B + e_1 y^T) = C_n^* \cdots C_1^*(B_1 \cdots B_n + e_1 y^T)$
- $O(n)$ storage
- Bonus: Redundancy! No need to keep track of y.

Representation of A

Altogether we have

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



Francis Iterations

- We have complex single-shift code . . .
- real double-shift code.
- We describe single-shift case for simplicity.
- ignoring rank-one part . . .

$A =$

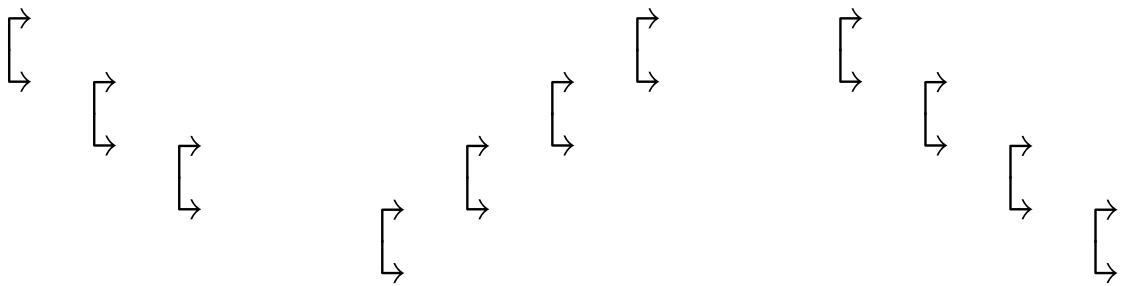
Francis Iterations

- We have complex single-shift code . . .
- real double-shift code.
- We describe single-shift case for simplicity.
- ignoring rank-one part . . .

$A =$

Francis Iterations

- We have complex single-shift code . . .
- real double-shift code.
- We describe single-shift case for simplicity.
- ignoring rank-one part . . .

$A =$ 

Two Basic Operations

Two basic operations:

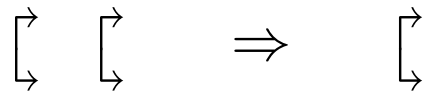
- Fusion

- Turnover (aka shift through, Givens swap, . . .)

Two Basic Operations

Two basic operations:

- Fusion

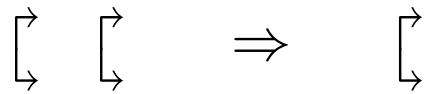


- Turnover (aka shift through, Givens swap, . . .)

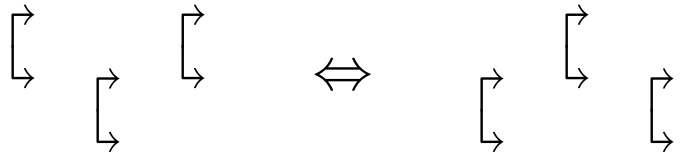
Two Basic Operations

Two basic operations:

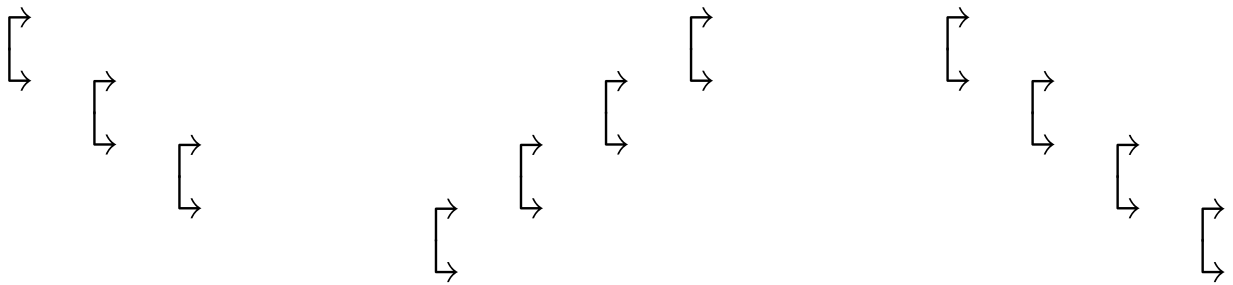
- Fusion



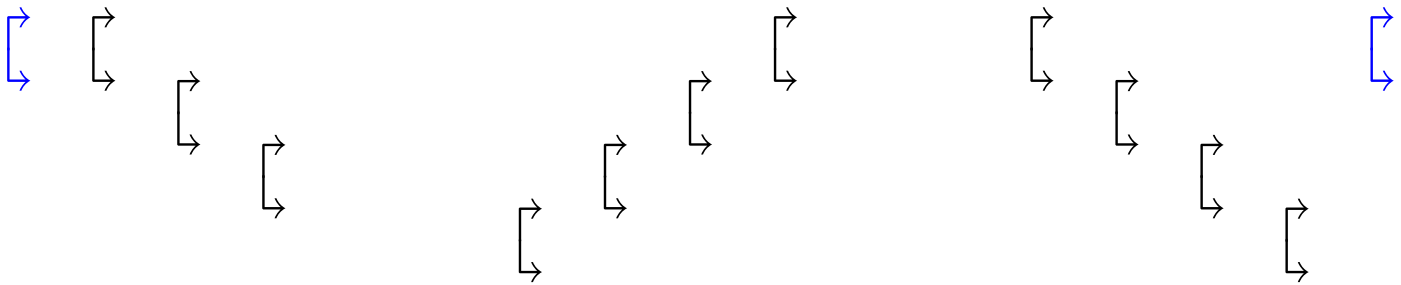
- Turnover (aka shift through, Givens swap, . . .)



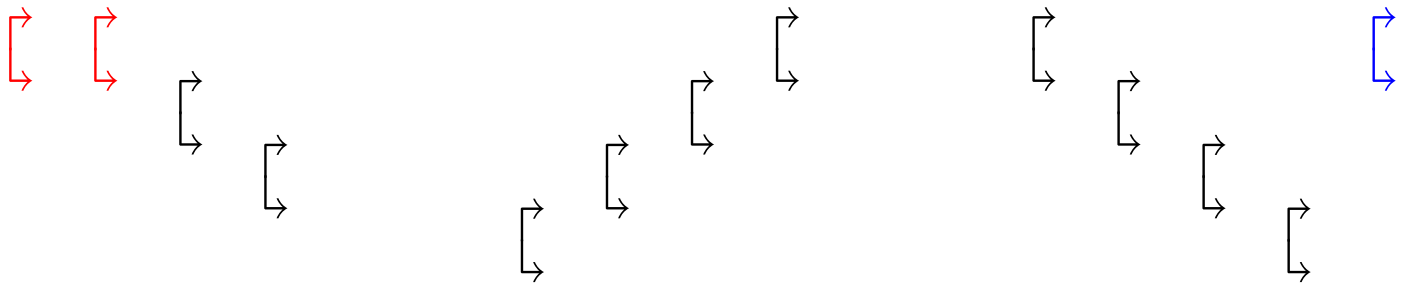
The Bulge Chase



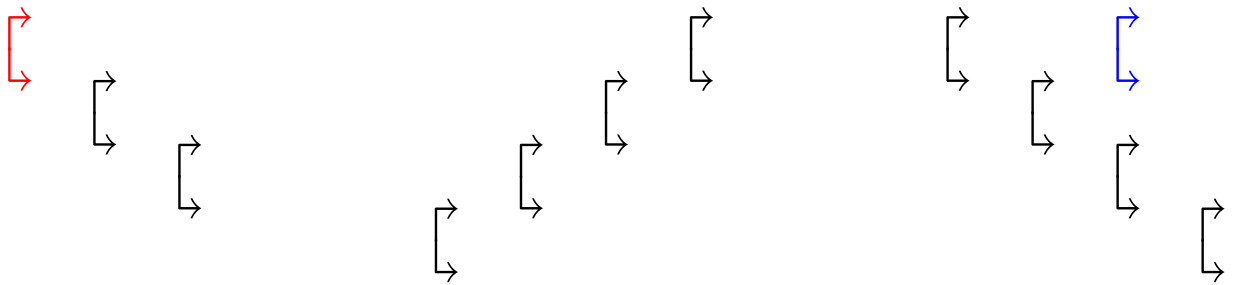
The Bulge Chase



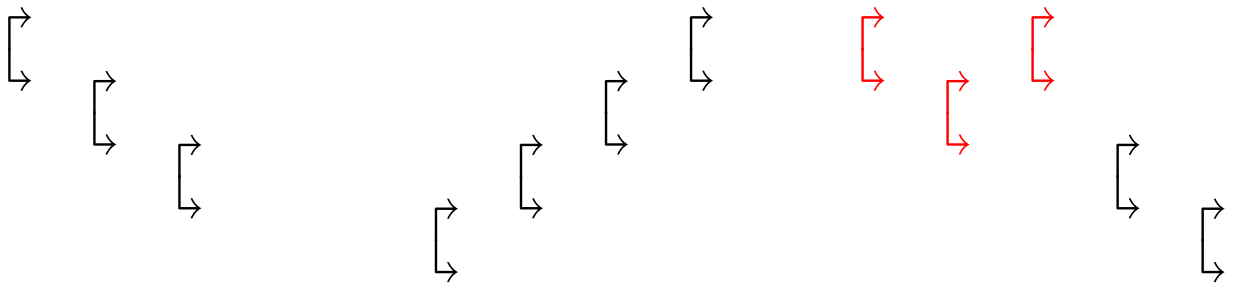
The Bulge Chase



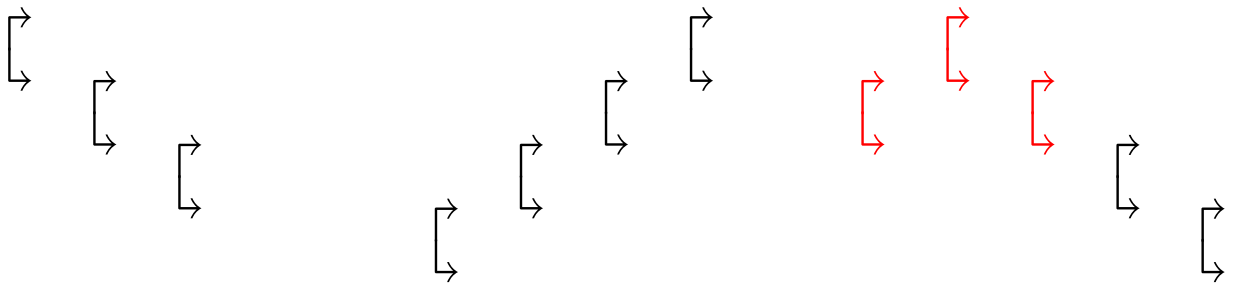
The Bulge Chase



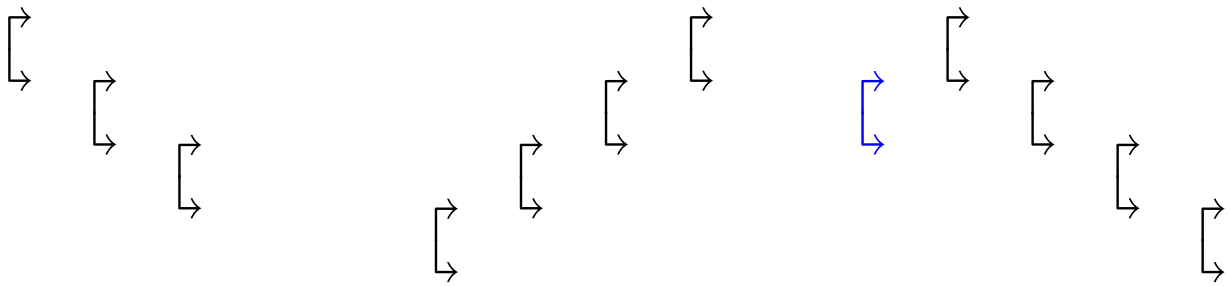
The Bulge Chase



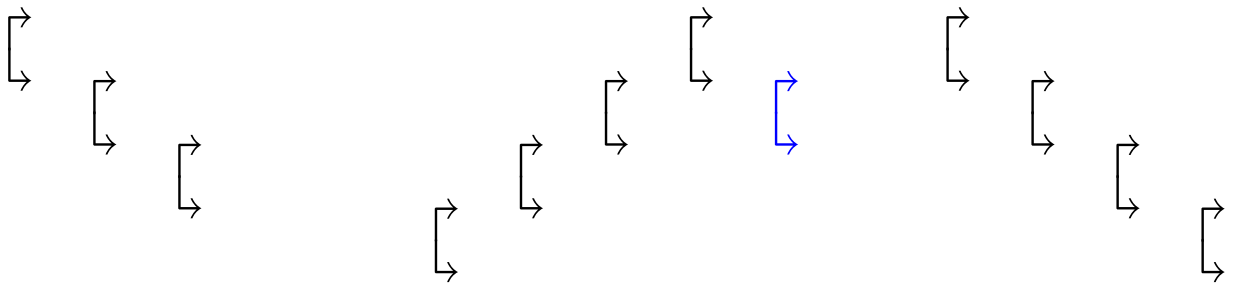
The Bulge Chase



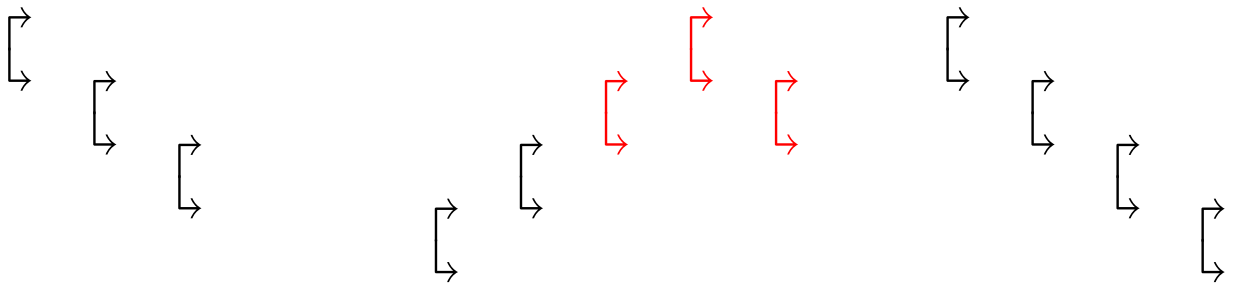
The Bulge Chase



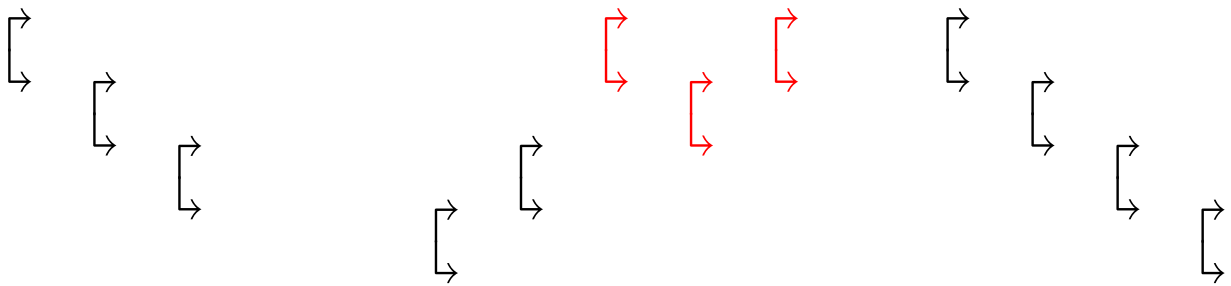
The Bulge Chase



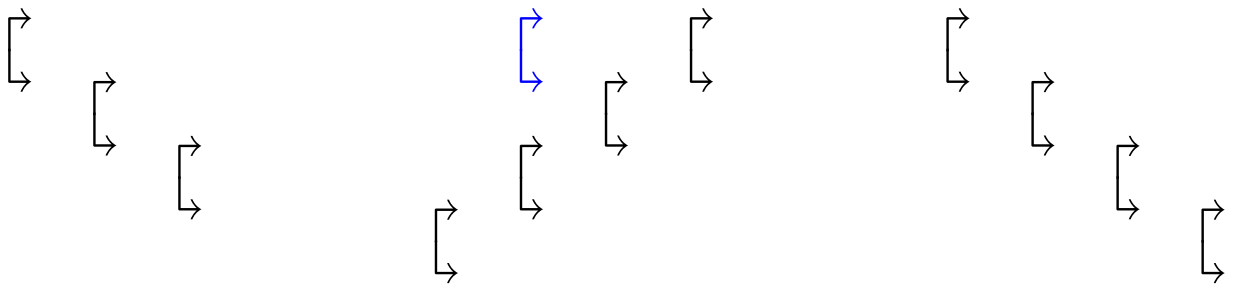
The Bulge Chase



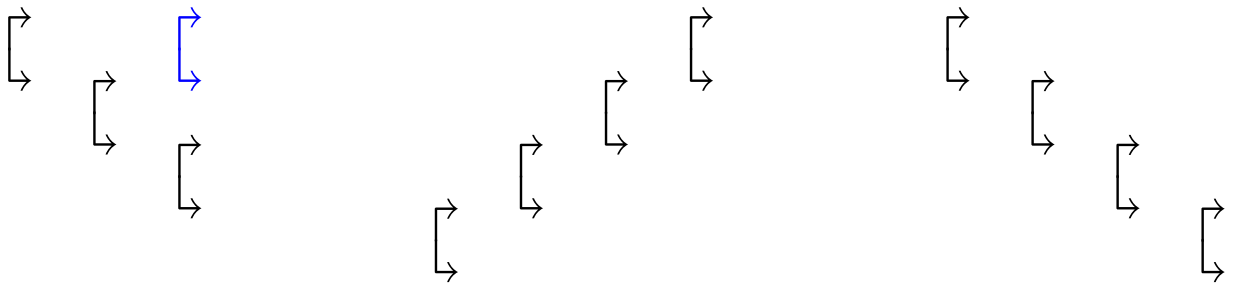
The Bulge Chase



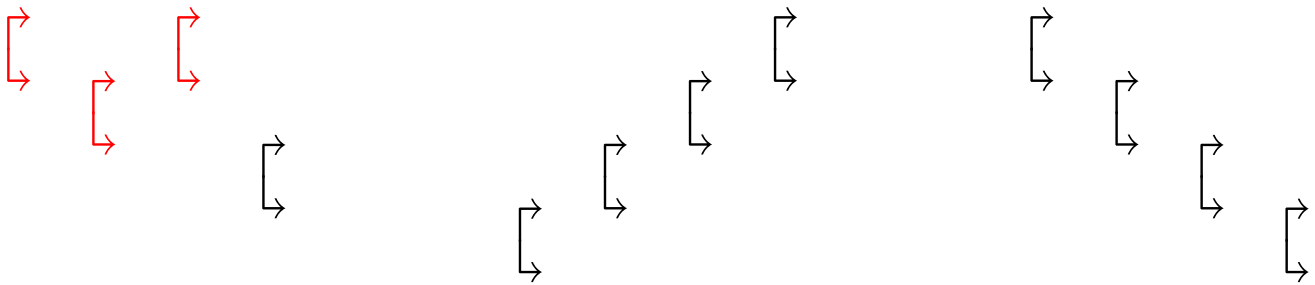
The Bulge Chase



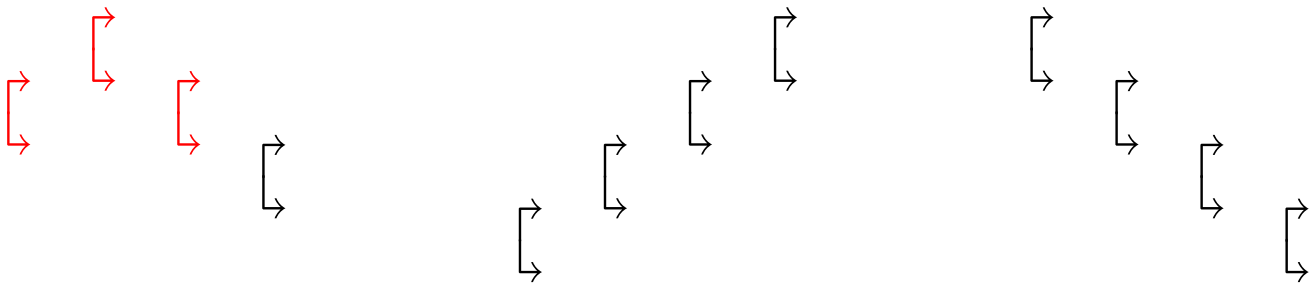
The Bulge Chase



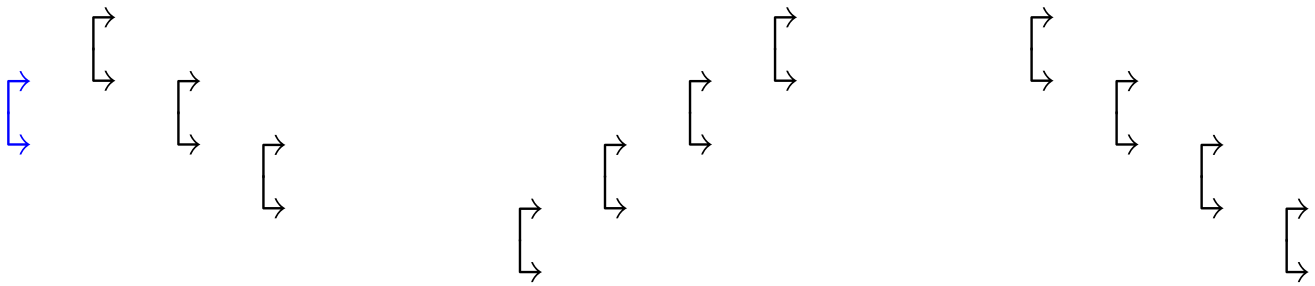
The Bulge Chase



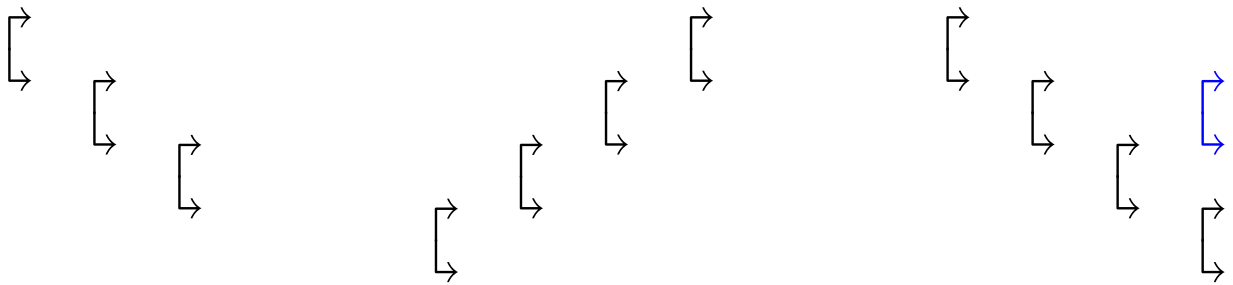
The Bulge Chase



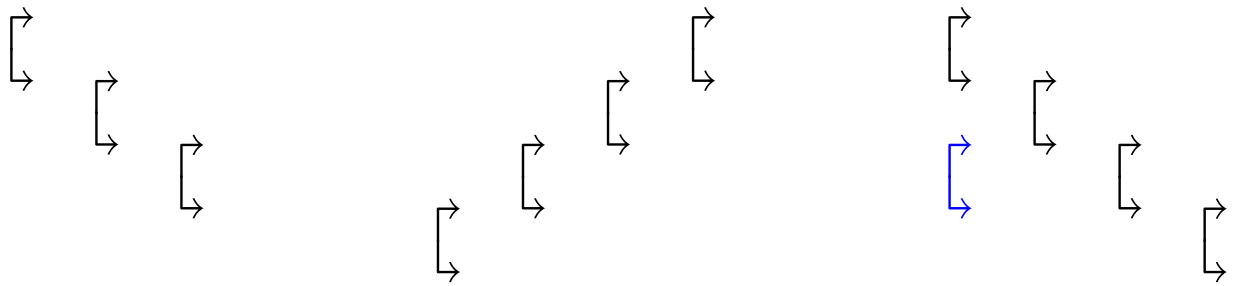
The Bulge Chase



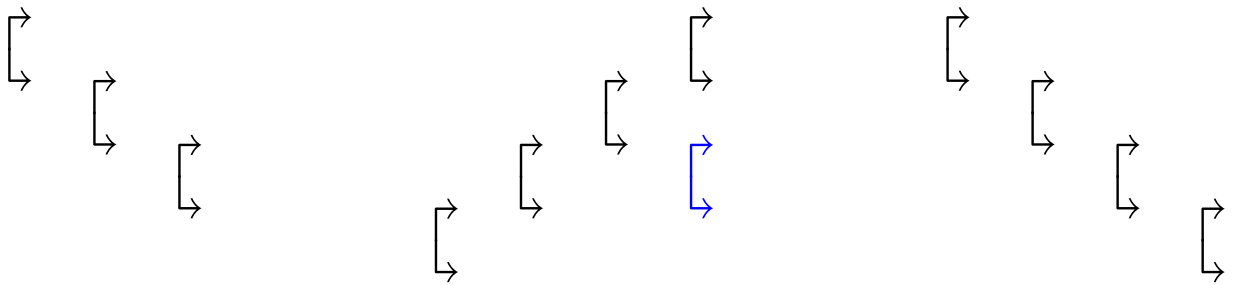
The Bulge Chase



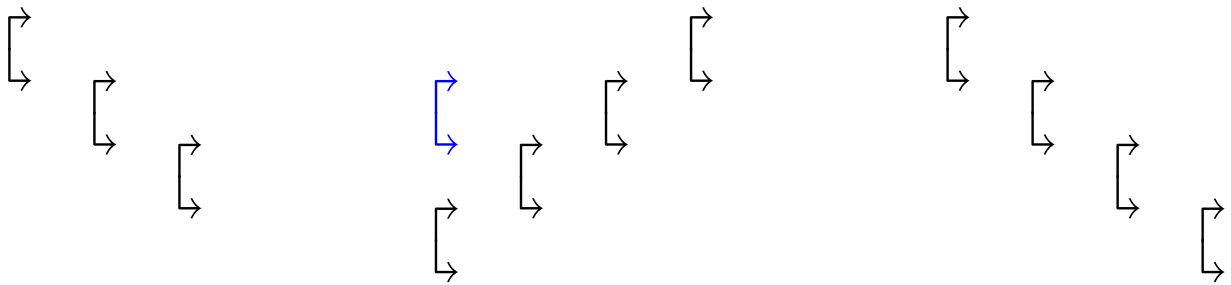
The Bulge Chase



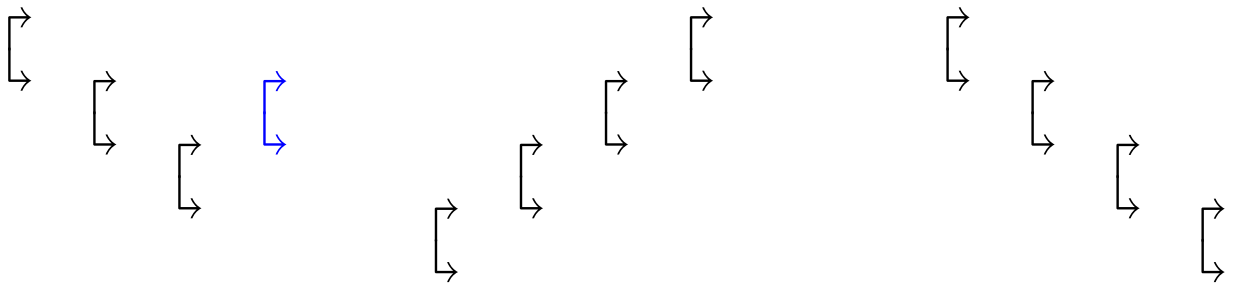
The Bulge Chase



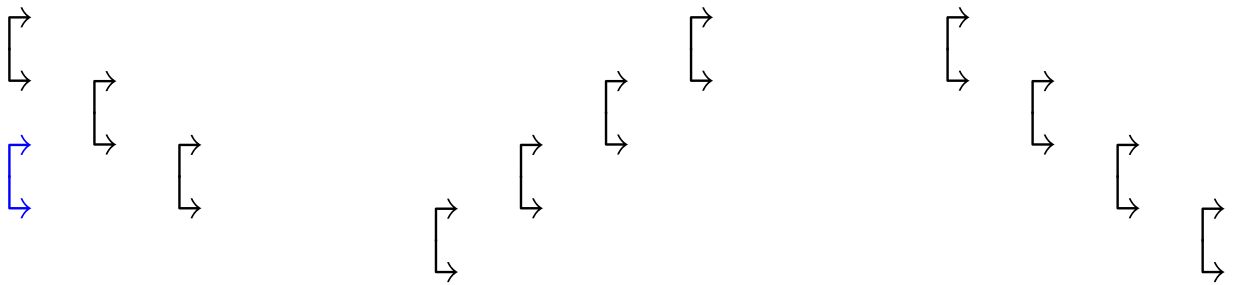
The Bulge Chase



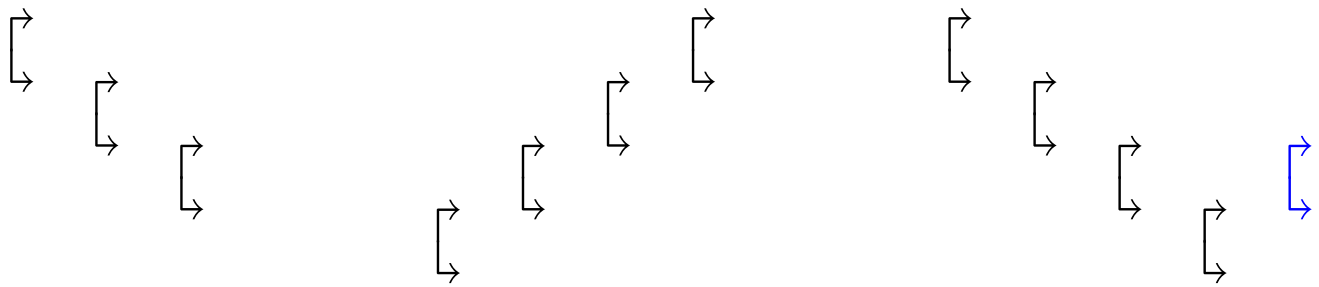
The Bulge Chase



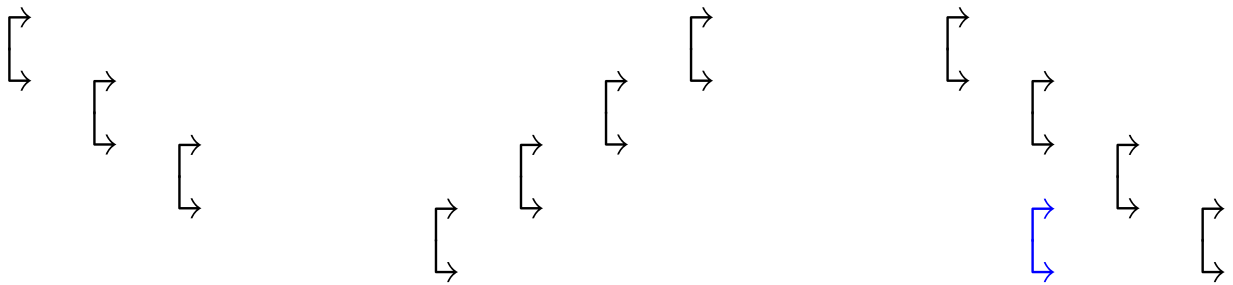
The Bulge Chase



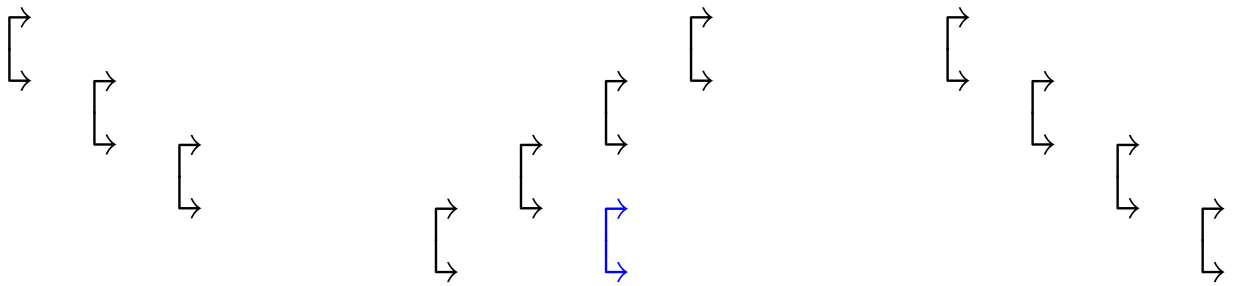
The Bulge Chase



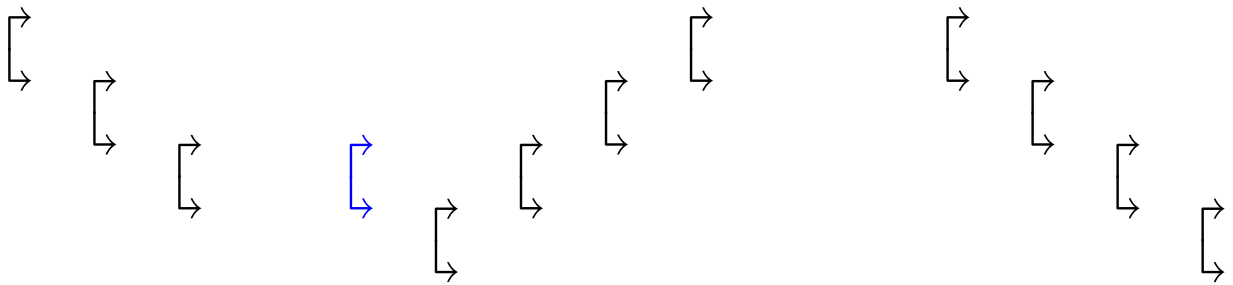
The Bulge Chase



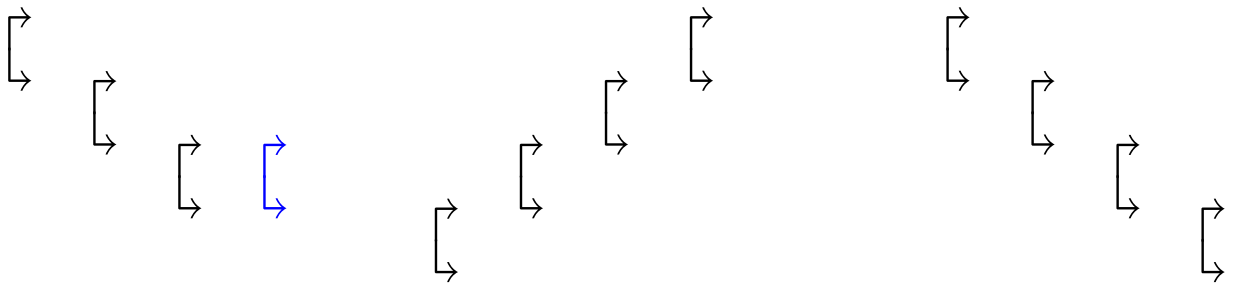
The Bulge Chase



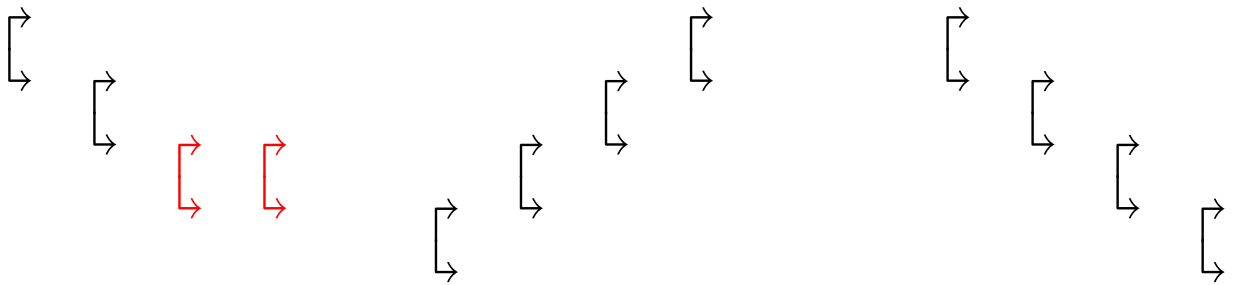
The Bulge Chase



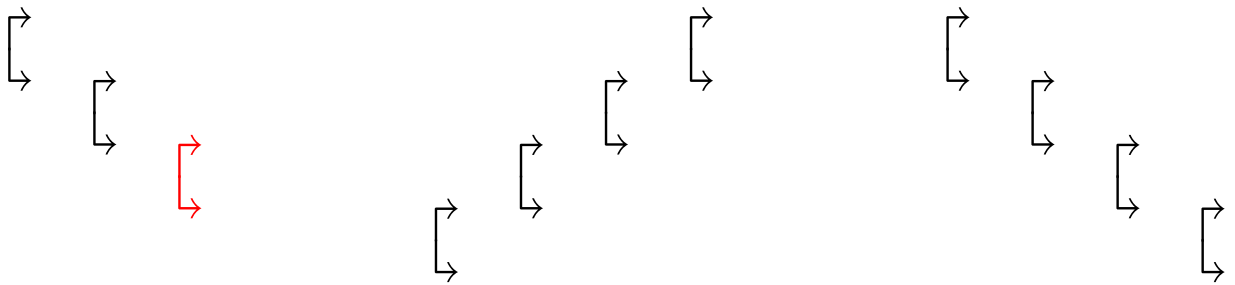
The Bulge Chase



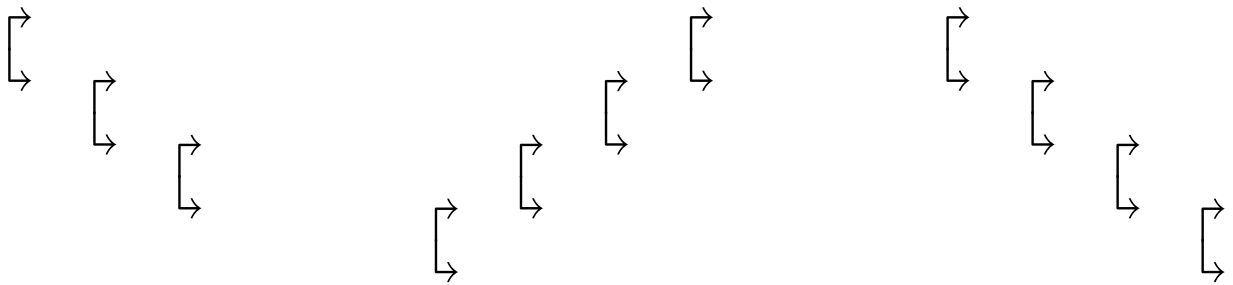
The Bulge Chase



The Bulge Chase



The Bulge Chase



Done!

- iteration complete!
- Cost: $3n$ turnovers/iteration, so $O(n)$ flops/iteration
- Double-shift iteration is similar.
- (Chase two core transformations instead of one.)

Done!

- iteration complete!
- Cost: $3n$ turnovers/iteration, so $O(n)$ flops/iteration
- Double-shift iteration is similar.
- (Chase two core transformations instead of one.)

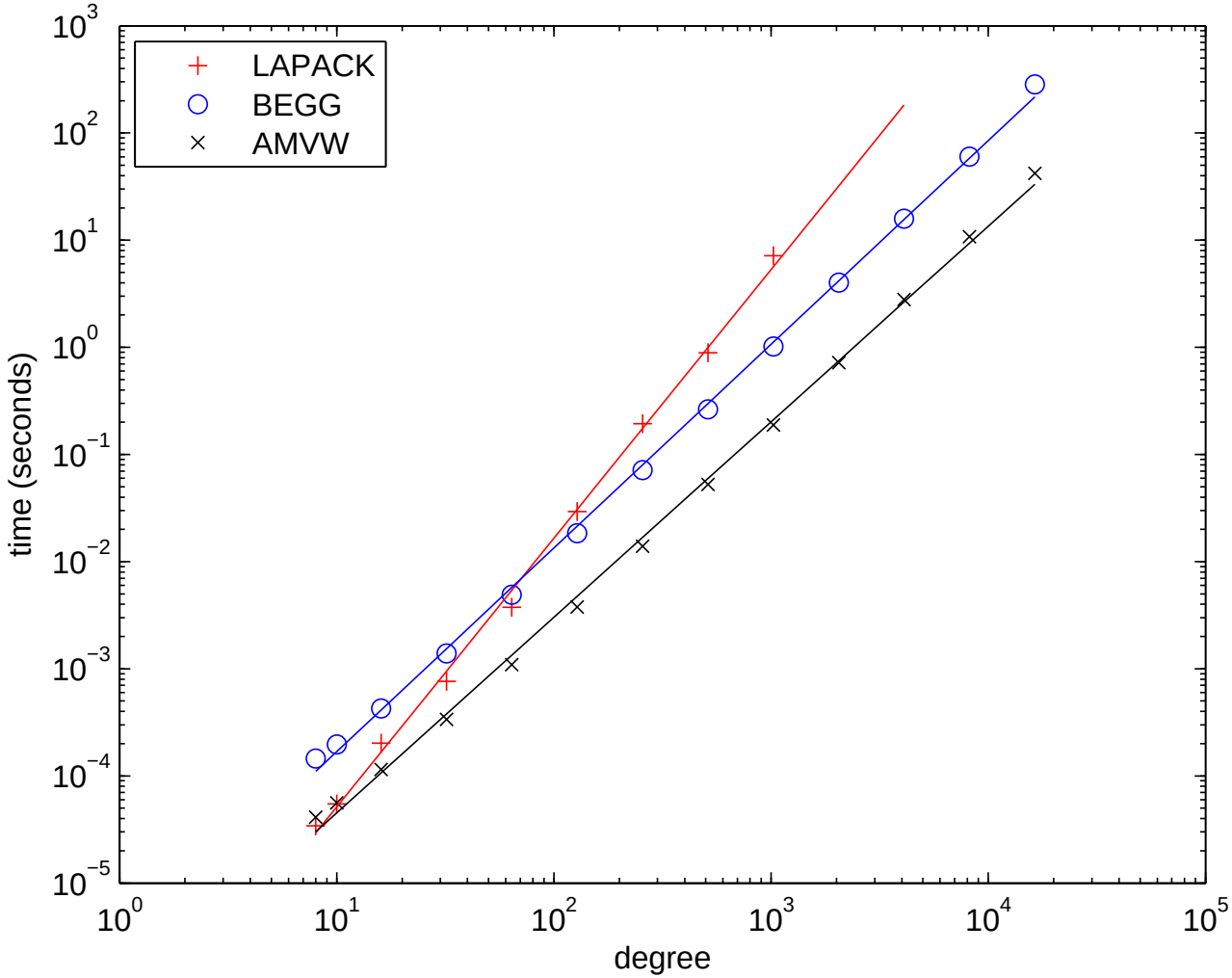
Done!

- iteration complete!
- Cost: $3n$ turnovers/iteration, so $O(n)$ flops/iteration
- Double-shift iteration is similar.
- (Chase two core transformations instead of one.)

Contestants

- LAPACK code ZHSEQR $(O(n^3))$
- BEGG (Boito et. al. 2012)
- AMVW (our single-shift code)

Speed Comparison, Complex Case



Speed Comparison, Complex Case

- At degree 1024:

method	time
LAPACK	7.14
BEGG	1.02
AMVW	0.18

Speed Comparison, Real Case

- Results similar for double-shift code . . .
- . . . but we're only about twice as fast as the competition.
- (compared with Chandrasekaran et. al.)

Accuracy

- On these big (and well conditioned) problems . . .
- we are about as accurate as LAPACK, . . .
- and more accurate than the other fast methods.
- Our codes act backward stable . . .
- because they are backward stable.
- much more in paper (almost done)
- also tried harder problems (we do OK)

Accuracy

- On these big (and well conditioned) problems . . .
- we are about as accurate as LAPACK, . . .
- and more accurate than the other fast methods.
- Our codes act backward stable . . .
- because they are backward stable.
- much more in paper (almost done)
- also tried harder problems (we do OK)

Accuracy

- On these big (and well conditioned) problems . . .
- we are about as accurate as LAPACK, . . .
- and more accurate than the other fast methods.
- Our codes act backward stable . . .
- because they are backward stable.
- much more in paper (almost done)
- also tried harder problems (we do OK)

Accuracy

- On these big (and well conditioned) problems . . .
- we are about as accurate as LAPACK, . . .
- and more accurate than the other fast methods.
- Our codes act backward stable . . .
- because they are backward stable.
- much more in paper (almost done)
- also tried harder problems (we do OK)

Accuracy

- On these big (and well conditioned) problems . . .
- we are about as accurate as LAPACK, . . .
- and more accurate than the other fast methods.
- Our codes act backward stable . . .
- because they are backward stable.
- much more in paper (almost done)
- also tried harder problems (we do OK)

Summary

- We have a new fast method for companion eigenvalue problems
- and unitary-plus-rank-one matrices in general.
- Method is normwise backward stable, accurate,
- and faster than other fast methods.

Summary

- We have a new fast method for companion eigenvalue problems
- and unitary-plus-rank-one matrices in general.
- Method is normwise backward stable, accurate,
- and faster than other fast methods.

Summary

- We have a new fast method for companion eigenvalue problems
- and unitary-plus-rank-one matrices in general.
- Method is normwise backward stable, accurate,
- and faster than other fast methods.

Summary

- We have a new fast method for companion eigenvalue problems
- and unitary-plus-rank-one matrices in general.
- Method is normwise backward stable, accurate,
- and faster than other fast methods.

Summary

- We have a new fast method for companion eigenvalue problems
- and unitary-plus-rank-one matrices in general.
- Method is normwise backward stable, accurate,
- and faster than other fast methods.

Summary

- We have a new fast method for companion eigenvalue problems
- and unitary-plus-rank-one matrices in general.
- Method is normwise backward stable, accurate,
- and faster than other fast methods.

Thank you for your attention.