High-performance computation of pseudospectra

Jack Poulson

Department of Mathematics
Stanford University

Innovative Computing Laboratory
The University of Tennessee
October 24, 2014
A note on collaborators

**Multishift Hessenberg solves:** collaboration with Greg Henry
Overview of talk

- Van Loan’s algorithm and the Demmel matrix
  - Brief overview of pseudospectra
  - Previous work
  - High-performance batched analogues
  - A brief example of the python interface
  - Results
  - Conclusions and future work
Overview of talk

- Van Loan’s algorithm and the Demmel matrix
- Brief overview of pseudospectra
- Previous work
- High-performance batched analogues
- A brief example of the python interface
- Results
- Conclusions and future work
Overview of talk

- Van Loan’s algorithm and the Demmel matrix
- Brief overview of pseudospectra
- Previous work
  - High-performance batched analogues
  - A brief example of the python interface
- Results
- Conclusions and future work
Overview of talk

- Van Loan’s algorithm and the Demmel matrix
- Brief overview of pseudospectra
- Previous work
- High-performance batched analogues
- A brief example of the python interface
- Results
- Conclusions and future work
Overview of talk

- Van Loan’s algorithm and the Demmel matrix
- Brief overview of pseudospectra
- Previous work
- High-performance batched analogues
- A brief example of the python interface
- Results
- Conclusions and future work
Overview of talk

- Van Loan’s algorithm and the Demmel matrix
- Brief overview of pseudospectra
- Previous work
- High-performance batched analogues
- A brief example of the python interface
- Results
  - Conclusions and future work
Overview of talk

- Van Loan’s algorithm and the Demmel matrix
- Brief overview of pseudospectra
- Previous work
- High-performance batched analogues
- A brief example of the python interface
- Results
- Conclusions and future work
Van Loan’s algorithm and the Demmel matrix

Brief overview of pseudospectra

Previous work

High-performance batched analogues

A brief example of the python interface

Results

Conclusions and future work
Van Loan’s algorithm and the Demmel matrix

- Efficient algorithm for evaluating resolvent over vertical line in complex plane [Van Loan-1985]
- Relationship to nearest stable matrix based on false conjecture, counter-example given in [Demmel-1987]
- Counter-example was first pseudospectral (computer) plot
Van Loan’s algorithm and the Demmel matrix

- Efficient algorithm for evaluating resolvent over vertical line in complex plane [Van Loan-1985]
- Relationship to nearest stable matrix based on false conjecture, counter-example given in [Demmel-1987]
- Counter-example was first pseudospectral (computer) plot
Van Loan’s algorithm and the Demmel matrix

- Efficient algorithm for evaluating resolvent over vertical line in complex plane [Van Loan-1985]
- Relationship to nearest stable matrix based on false conjecture, counter-example given in [Demmel-1987]
- Counter-example was first pseudospectral (computer) plot
Van Loan’s algorithm and the Demmel matrix

- Efficient algorithm for evaluating resolvent over vertical line in complex plane [Van Loan-1985]
- Relationship to nearest stable matrix based on false conjecture, counter-example given in [Demmel-1987]
- Counter-example was first pseudospectral (computer) plot

\[ D(n, \beta) = (\beta J_{-\beta^{-1}, n})^{-1}, \quad n = 3, \beta = 100 \]
Van Loan’s algorithm and the Demmel matrix

Brief overview of pseudospectra

Previous work

High-performance batched analogues

A brief example of the python interface

Results

Conclusions and future work
Pseudospectra

With convention $\|(\lambda I - A)^{-1}\|_p = \infty$ for $\lambda \in \mathcal{L}(A)$, eigenvalues are singularities of resolvent norm $\|(\xi I - A)^{-1}\|_p$.

Natural generalization of spectrum for each $p$-norm and $\epsilon > 0$:

$$\mathcal{L}_\epsilon^p(A) = \left\{ \xi \in \mathbb{C} : \|(\xi I - A)^{-1}\|_p > \frac{1}{\epsilon} \right\}$$


Extensive review of field in Trefethen and Embree’s book *Spectra and pseudospectra*...

Most common tool is EigTool [Wright et al.-2001], but level 2 BLAS (trsv) and sequential
Pseudospectra

With convention \( \|(\lambda I - A)^{-1}\|_p = \infty \) for \( \lambda \in \mathcal{L}(A) \), eigenvalues are singularities of resolvent norm \( \|(\xi I - A)^{-1}\|_p \).

Natural generalization of spectrum for each \( p \)-norm and \( \epsilon > 0 \):

\[
\mathcal{L}_\epsilon^p(A) = \left\{ \xi \in \mathbb{C} : \|(\xi I - A)^{-1}\|_p > \frac{1}{\epsilon} \right\}
\]


Extensive review of field in Trefethen and Embree’s book *Spectra and pseudospectra*...

Most common tool is EigTool [Wright et al.-2001], but level 2 BLAS (trsv) and sequential
Pseudospectra

With convention $\|(\lambda I - A)^{-1}\|_p = \infty$ for $\lambda \in \mathcal{L}(A)$, eigenvalues are singularities of resolvent norm $\|(\xi I - A)^{-1}\|_p$.

Natural generalization of spectrum for each $p$-norm and $\epsilon > 0$:

$$\mathcal{L}_\epsilon^p(A) = \left\{ \xi \in \mathbb{C} : \|(\xi I - A)^{-1}\|_p > \frac{1}{\epsilon} \right\}$$


Extensive review of field in Trefethen and Embree’s book Spectra and pseudospectra...

Most common tool is EigTool [Wright et al.-2001], but level 2 BLAS (trsv) and sequential
Pseudospectra

With convention \( \|(\lambda I - A)^{-1}\|_p = \infty \) for \( \lambda \in \mathcal{L}(A) \), eigenvalues are singularities of resolvent norm \( \|(\xi I - A)^{-1}\|_p \).

Natural generalization of spectrum for each \( p \)-norm and \( \epsilon > 0 \):

\[
\mathcal{L}_\epsilon^p(A) = \left\{ \xi \in \mathbb{C} : \|(\xi I - A)^{-1}\|_p > \frac{1}{\epsilon} \right\}
\]


Extensive review of field in Trefethen and Embree’s book *Spectra and pseudospectra*...

Most common tool is EigTool [Wright et al.-2001], but level 2 BLAS (trsv) and sequential
Pseudospectra

With convention \( \|(\lambda I - A)^{-1}\|_p = \infty \) for \( \lambda \in \mathcal{L}(A) \), eigenvalues are singularities of resolvent norm \( \|(\xi I - A)^{-1}\|_p \).

Natural generalization of spectrum for each \( p \)-norm and \( \epsilon > 0 \):

\[
\mathcal{L}^p_{\epsilon}(A) = \left\{ \xi \in \mathbb{C} : \|(\xi I - A)^{-1}\|_p > \frac{1}{\epsilon} \right\}
\]


Extensive review of field in Trefethen and Embree’s book *Spectra and pseudospectra*...

Most common tool is EigTool [Wright et al.-2001], but level 2 BLAS (trsv) and sequential
Van Loan’s algorithm and the Demmel matrix

Brief overview of pseudospectra

Previous work

High-performance batched analogues

A brief example of the python interface

Results

Conclusions and future work
“Embarrassingly parallel” Lanczos \cite{Braconnier-1996}
Parallel sparse-direct shift-and-invert \cite{Fraysse et al.-1996}
Parallel path-following methods, e.g., \cite{Mezher-2001,Bekas et al.-2000,2001,...]
Our focus on level 3 (parallel) batch evaluation over point clouds after parallel Schur decomposition
Key idea: simultaneously drive many Lanczos/Arnoldi methods via multishift extensions of standard TRSM algorithm
Software for one and two-norm pseudospectra already released in Elemental \cite{P. et al.-2013}
Relationship to previous work

- “Embarrassingly parallel” Lanczos [Braconnier-1996]
- Parallel sparse-direct shift-and-invert [Fraysse et al.-1996]
- Parallel path-following methods, e.g., [Mezher-2001, Bekas et al.-2000, 2001,...]
- Our focus on level 3 (parallel) batch evaluation over point clouds after parallel Schur decomposition
- Key idea: simultaneously drive many Lanczos/Arnoldi methods via multishift extensions of standard TRSM algorithm
- Software for one and two-norm pseudospectra already released in Elemental [P. et al.-2013]
Relationship to previous work

- “Embarrassingly parallel” Lanczos [Braconnier-1996]
- Parallel sparse-direct shift-and-invert [Fraysse et al.-1996]
- Parallel path-following methods, e.g., [Mezher-2001, Bekas et al.-2000, 2001,...]
- Our focus on level 3 (parallel) batch evaluation over point clouds after parallel Schur decomposition
- Key idea: simultaneously drive many Lanczos/Arnoldi methods via multishift extensions of standard TRSM algorithm
- Software for one and two-norm pseudospectra already released in Elemental [P. et al.-2013]
Relationship to previous work

- “Embarrassingly parallel” Lanczos [Braconnier-1996]
- Parallel sparse-direct shift-and-invert [Fraysse et al.-1996]
- Our focus on level 3 (parallel) batch evaluation over point clouds after parallel Schur decomposition
  - Key idea: simultaneously drive many Lanczos/Arnoldi methods via multishift extensions of standard TRSM algorithm
  - Software for one and two-norm pseudospectra already released in Elemental [P. et al.-2013]
Relationship to previous work

- “Embarrassingly parallel” Lanczos [Braconnier-1996]
- Parallel sparse-direct shift-and-invert [Fraysse et al.-1996]
- Our focus on level 3 (parallel) batch evaluation over point clouds after parallel Schur decomposition
- Key idea: simultaneously drive many Lanczos/Arnoldi methods via multishift extensions of standard TRSM algorithm
- Software for one and two-norm pseudospectra already released in Elemental [P. et al.-2013]
Relationship to previous work

- “Embarrassingly parallel” Lanczos [Braconnier-1996]
- Parallel sparse-direct shift-and-invert [Fraysse et al.-1996]
- Parallel path-following methods, e.g., [Mezher-2001, Bekas et al.-2000, 2001,...]
- Our focus on level 3 (parallel) batch evaluation over point clouds after parallel Schur decomposition
- Key idea: simultaneously drive many Lanczos/Arnoldi methods via multishift extensions of standard TRSM algorithm
- Software for one and two-norm pseudospectra already released in Elemental [P. et al.-2013]
(Extended) Van Loan algorithm

[Van Loan-1985,Lui-1997]
Given a reduction to condensed form, \( A = QGQ^H \),

\[
\mathcal{L}_\varepsilon^p(A) = \left\{ \xi \in \mathbb{C} : \|Q(\xi I - G)^{-1}Q^H\|_p > \frac{1}{\varepsilon} \right\},
\]

and, when \( p = 2 \),

\[
\mathcal{L}_\varepsilon^2(A) = \left\{ \xi \in \mathbb{C} : \|(\xi I - G)^{-1}\|_2 > \frac{1}{\varepsilon} \right\},
\]

First reduce to (quasi-)triangular/Hessenberg form, then,

- (Restarted) Lanczos/Arnoldi for each \( \|(\xi I - G)^{-1}\|_2 \), or
- Blocked Hager [Higham/Tisseur-2000] for each \( \|Q(\xi I - G)^{-1}Q^H\|_1 \).

Note that this algorithm makes less sense when \((\xi I - A)^{-1}\) can already be applied in quadratic time (e.g., if \( A \) is 3D FEM discretization)
(Extended) Van Loan algorithm

[Van Loan-1985,Lui-1997]
Given a reduction to condensed form, \( A = QGQ^H \),
\[
\mathcal{L}_\varepsilon^p(A) = \left\{ \xi \in \mathbb{C} : \| Q(\xi I - G)^{-1} Q^H \|_p > \frac{1}{\varepsilon} \right\},
\]
and, when \( p = 2 \),
\[
\mathcal{L}_\varepsilon^2(A) = \left\{ \xi \in \mathbb{C} : \| (\xi I - G)^{-1} \|_2 > \frac{1}{\varepsilon} \right\},
\]
First reduce to (quasi-)triangular/Hessenberg form, then,
- (Restarted) Lanczos/Arnoldi for each \( \| (\xi I - G)^{-1} \|_2 \), or
- Blocked Hager [Higham/Tisseur-2000] for each \( \| Q(\xi I - G)^{-1} Q^H \|_1 \).

Note that this algorithm makes less sense when \( (\xi I - A)^{-1} \) can already be applied in quadratic time (e.g., if \( A \) is 3D FEM discretization)
(Extended) Van Loan algorithm

[Van Loan-1985,Lui-1997]
Given a reduction to condensed form, \( A = QGQ^H \),

\[
\mathcal{L}_\epsilon^p(A) = \left\{ \xi \in \mathbb{C} : \| Q(\xi I - G)^{-1} Q^H \|_p > \frac{1}{\epsilon} \right\},
\]

and, when \( p = 2 \),

\[
\mathcal{L}_\epsilon^2(A) = \left\{ \xi \in \mathbb{C} : \| (\xi I - G)^{-1} \|_2 > \frac{1}{\epsilon} \right\},
\]

First reduce to (quasi-)triangular/Hessenberg form, then,

- (Restarted) Lanczos/Arnoldi for each \( \| (\xi I - G)^{-1} \|_2 \), or
- Blocked Hager [Higham/Tisseur-2000] for each \( \| Q(\xi I - G)^{-1} Q^H \|_1 \).

Note that this algorithm makes less sense when \( (\xi I - A)^{-1} \) can already be applied in quadratic time (e.g., if \( A \) is 3D FEM discretization)
(Extended) Van Loan algorithm for $L^2_\epsilon(A)$

Algorithm: Two-norm pseudospectra via extended Van Loan algorithm

Input: $A \in \mathbb{F}^{n \times n}$, shifts $\Omega \subset \mathbb{C}$, restart size $k$

Output: $\{\phi(\xi)\}_{\xi \in \Omega} \approx \{\|(\xi I - A)^{-1}\|_2\}_{\xi \in \Omega}$

$G := \text{Schur}(A)$, $\text{RealSchur}(A)$, or $\text{Hessenberg}(A)$

foreach $\xi \in \Omega$ do

// Estimate $\|(\xi I - G)^{-1}\|_2$ via Restarted Arnoldi
Choose $v_0 \in \mathbb{C}^n$ with $\|v_0\|_2 = 1$

while not converged do

for $j = 0, \ldots, k - 1$ do

// $\"(\xi I - G)^{-H}(\xi I - G)^{-1}v_j = V_jH_j + v_j(\beta_j e_j)^H$ 
$x_j := (\xi I - G)^{-H}(\xi I - G)^{-1}v_j$

Expand Arnoldi decomposition using $x_j$

$[\lambda, v_0] := \text{MaxEig}(H_k)$

$\phi(\xi) := \text{RealPart}(\lambda)$
[Hager-1984, Higham-1988] approach for $\mathcal{L}_1^\epsilon(A)$

Algorithm: One-norm pseudospectra via Hager-Higham algorithm

Input: $A \in \mathbb{F}^{n \times n}$, $\Omega \subset \mathbb{C}$
Output: $\{\phi(\xi)\}_{\xi \in \Omega} \approx \{\|(A - \xi I)^{-1}\|_1\}_{\xi \in \Omega}$

$[Q, G] := \text{Schur}(A), \text{RealSchur}(A), \text{or Hessenberg}(A)$

foreach $\xi \in \Omega$ do

// Estimate $\|(A - \xi I)^{-1}\|_1$ via Hager-Higham algorithm

Choose $u \in \mathbb{C}^n$ with $\|u\|_1 = 1$

$k := 0$

repeat

$x := u$

$y := Q(G - \xi I)^{-1}Q^Hx$

$w := Q(G - \xi I)^{-H}Q^H\text{sign}(y)$

$u := e_j$ where $|w(j)| = \|w\|_\infty$

$k := k + 1$

until $(k \geq 2 \text{ and } \|w\|_\infty \leq w^Hx) \text{ or } k \geq 5$

$\phi(\xi) := \|y\|_1$

// Take the maximum between the current estimate and a heuristic

$b := \frac{2}{3n}([-1]^j(n + j - 1)]_{j=0:n-1}$

$x := Q(G - \xi I)^{-1}Q^Hb$

$\phi(\xi) := \max(\phi(\xi), \|x\|_1)$

Block algorithm [Higham/Tisseur-2000] should be used in practice
[Hager-1984,Higham-1988] approach for $\mathcal{L}_1^\varepsilon(A)$

**Algorithm:** One-norm pseudospectra via Hager-Higham algorithm

- **Input:** $A \in \mathbb{F}^{n \times n}$, $\Omega \subset \mathbb{C}$
- **Output:** $\{\phi(\xi)\}_{\xi \in \Omega} \approx \{\|(A - \xi I)^{-1}\|_1\}_{\xi \in \Omega}$

$[Q, G] := \text{Schur}(A)$, $\text{RealSchur}(A)$, or $\text{Hessenberg}(A)$

```plaintext
foreach $\xi \in \Omega$ do
  // Estimate $\|(A - \xi I)^{-1}\|_1$ via Hager-Higham algorithm
  Choose $u \in \mathbb{C}^n$ with $\|u\|_1 = 1$
  $k := 0$
  repeat
    $x := u$
    $y := Q(G - \xi I)^{-1}Q^Hx$
    $w := Q(G - \xi I)^{-H}Q^H\text{sign}(y)$
    $u := e_j$ where $|w(j)| = \|w\|_\infty$
    $k := k + 1$
  until ($k \geq 2$ and $\|w\|_\infty \leq w^Hx$) or $k \geq 5$

  $\phi(\xi) := \|y\|_1$

  // Take the maximum between the current estimate and a heuristic
  $b := \frac{2}{3n}[(n+1)j](n+j-1)]_{j=0:n-1}$
  $x := Q(G - \xi I)^{-1}Q^Hb$
  $\phi(\xi) := \max(\phi(\xi), \|x\|_1)$
```

**Block algorithm** [Higham/Tisseur-2000] should be used in practice
Van Loan’s algorithm and the Demmel matrix

Brief overview of pseudospectra

Previous work

High-performance batched analogues

A brief example of the python interface

Results

Conclusions and future work
(Generalized) multishift solves

Computing $\{(\delta_j F - G)^{-1} y_j\}_j$ equivalent to solving for $X$ in

$$FXD - GX = Y,$$

where $D = \text{diag}((\delta_0, \delta_1, \cdots))$.

- Multishift triangular solves ($G$ triangular, $F = I$) require trivial changes to usual high-performance TRSM algorithms [Henry-1994]
- Quasi-triangular similar; adjust blocksizes to not split $2 \times 2$
- Very recently used by [Gates/Haidar/Dongarra-2014] for eigenvectors of nonsymmetric matrices
- Generalized multishift (quasi-)triangular solves can similarly be made high-performance with a simple trick...
- Large fraction of work in Hessenberg case is level 1...
(Generalized) multishift solves

Computing $\{(\delta_j F - G)^{-1} y_j\}_j$ equivalent to solving for $X$ in

$$FXD - GX = Y,$$

where $D = \text{diag}((\delta_0, \delta_1, \cdots))$.

- Multishift triangular solves ($G$ triangular, $F = I$) require trivial changes to usual high-performance TRSM algorithms [Henry-1994]
- Quasi-triangular similar; adjust blockizes to not split $2 \times 2$
- Very recently used by [Gates/Haidar/Dongarra-2014] for eigenvectors of nonsymmetric matrices
- Generalized multishift (quasi-)triangular solves can similarly be made high-performance with a simple trick...
- Large fraction of work in Hessenberg case is level 1...
(Generalized) multishift solves

Computing \( \{(\delta_j F - G)^{-1} y_j\}_j \) equivalent to solving for \( X \) in

\[
FXD - GX = Y,
\]

where \( D = \text{diag}(\delta_0, \delta_1, \cdots) \).

- Multishift triangular solves \( G \) triangular, \( F = I \) require trivial changes to usual high-performance TRSM algorithms [Henry-1994]
- Quasi-triangular similar; adjust blockizes to not split \( 2 \times 2 \)
  - Very recently used by [Gates/Haidar/Dongarra-2014] for eigenvectors of nonsymmetric matrices
- Generalized multishift (quasi-)triangular solves can similarly be made high-performance with a simple trick...
- Large fraction of work in Hessenberg case is level 1...
(Generalized) multishift solves

Computing \( \{ (\delta_j F - G)^{-1} y_j \}_j \) equivalent to solving for \( X \) in

\[
FXD - GX = Y,
\]

where \( D = \text{diag}(\delta_0, \delta_1, \cdots) \).

- Multishift triangular solves (\( G \) triangular, \( F = I \)) require
  trivial changes to usual high-performance TRSM
  algorithms [Henry-1994]
- Quasi-triangular similar; adjust block sizes to not split \( 2 \times 2 \)
- Very recently used by [Gates/Haidar/Dongarra-2014] for
eigenvectors of nonsymmetric matrices
- Generalized multishift (quasi-)triangular solves can
  similarly be made high-performance with a simple trick...
- Multishift Hessenberg solves heavily investigated for
- Large fraction of work in Hessenberg case is level 1...
(Generalized) multishift solves

Computing \( \{(\delta_j F - G)^{-1} y_j\}_j \) equivalent to solving for \( X \) in

\[
FXD - GX = Y,
\]

where \( D = \text{diag}((\delta_0, \delta_1, \cdots)) \).

- Multishift triangular solves (\( G \) triangular, \( F = I \)) require trivial changes to usual high-performance TRSM algorithms [Henry-1994]
- Quasi-triangular similar; adjust blocksize to not split \( 2 \times 2 \)
- Very recently used by [Gates/Haidar/Dongarra-2014] for eigenvectors of nonsymmetric matrices
- Generalized multishift (quasi-)triangular solves can similarly be made high-performance with a simple trick...
- Large fraction of work in Hessenberg case is level 1...
(Generalized) multishift solves

Computing \( \{(\delta_j F - G)^{-1} y_j\}_j \) equivalent to solving for \( X \) in

\[
FXD - GX = Y,
\]

where \( D = \text{diag}((\delta_0, \delta_1, \cdots)) \).

- Multishift triangular solves (\( G \) triangular, \( F = I \)) require trivial changes to usual high-performance TRSM algorithms [Henry-1994]
- Quasi-triangular similar; adjust block sizes to not split \( 2 \times 2 \)
- Very recently used by [Gates/Haidar/Dongarra-2014] for eigenvectors of nonsymmetric matrices
- Generalized multishift (quasi-)triangular solves can similarly be made high-performance with a simple trick...
- Large fraction of work in Hessenberg case is level 1...
(Generalized) multishift solves

Computing $\{(\delta_j F - G)^{-1} y_j\}_j$ equivalent to solving for $X$ in

$$FXD - GX = Y,$$

where $D = \text{diag}((\delta_0, \delta_1, \cdots))$.

- Multishift triangular solves ($G$ triangular, $F = I$) require trivial changes to usual high-performance TRSM algorithms [Henry-1994]
- Quasi-triangular similar; adjust blocksizes to not split $2 \times 2$
- Very recently used by [Gates/Haidar/Dongarra-2014] for eigenvectors of nonsymmetric matrices
- Generalized multishift (quasi-)triangular solves can similarly be made high-performance with a simple trick...
- Large fraction of work in Hessenberg case is level 1...
Interleaved Van Loan algorithm for $\mathcal{L}^2_\epsilon(A)$

**Algorithm:** Two-norm pseudospectra via interleaved extended Van Loan algorithm

**Input:** $A \in \mathbb{F}^{n \times n}$, shift vector $z \in \mathbb{C}^m$, restart size $k$

**Output:** $f \approx \left[ \| (z(s)I - A)^{-1} \|_2 \right]_{s=0:m-1}$

$G := \text{Schur}(A)$, $\text{RealSchur}(A)$, or $\text{Hessenberg}(A)$

Initialize each column of $W_0 \in \mathbb{C}^{n \times m}$ to have unit two-norm

$I := (0, ..., m-1)$

while $I \neq \emptyset$ do

for $j = 0, ..., k-1$ do

// $(z(s)I - G)^{-H}(z(s)I - G)^{-1}W_j(t) = W_j(t)H_j(t) + W_j(\cdot, t)(b_j(t)e_j)^H$, $\forall s = I(t)$

$X := \text{MultishiftSolve}(G, z(I), W_j)$

$X := \text{MultishiftSolve}(G^H, \bar{z}(I), X)$

Expand Arnoldi decompositions

foreach $s = I(t)$ do

$[\lambda, W_0(\cdot, t)] := \text{MaxEig}(H_k(t))$

$f(s) := \text{RealPart}(\lambda)$

if converged then

Delete $I(t)$ and $W_0(\cdot, t)$

endif

endfor
Interleaved Hager-Higham algorithm for $\mathcal{L}_{\epsilon}^1(A)$

**Algorithm:** One-norm pseudospectra via interleaved Hager-Higham algorithm

**Input:** $A \in \mathbb{F}^{n \times n}$, shift vector $z \in \mathbb{C}^m$

**Output:** $f \approx \left[ \|(z(j)I - A)^{-1}\|_1 \right]_{j=0 \ldots m-1}$

$[Q, G] := \text{Schur}(A), \text{RealSchur}(A), \text{or} \ D\text{Hessenberg}(A)$

Initialize each column of $U \in \mathbb{C}^{n \times m}$ to have unit one-norm

$I := (0, ..., m - 1)$

$k := 0$

while $I \neq \emptyset$ do

$X := U$

$Y := Q^H X$

$Y := \text{MultishiftSolve}(G, z(I), Y)$

$Y := QY$

$W := Q^H \text{sign}(Y)$

$W := \text{MultishiftSolve}(G^H, \bar{z}(I), W)$

$W := QW$

$k := k + 1$

foreach $s = I(t)$ do

$x := X(:, t)$, $y := Y(:, t)$, $w := W(:, t)$

if $(k \geq 2$ and $\|w\|_\infty \leq w^H x)$ or $k \geq 5$ then

$f(s) = \|y\|_1$

Delete $I(t)$ and $U(:, t)$

else $U(:, t) := e_j$ where $|w(j)| = \|w\|_\infty$


\[ \ldots \]
Interleaved Hager-Higham algorithm for $\mathcal{L}_\epsilon^1(A)$

**Algorithm:** One-norm pseudospectra via interleaved Hager-Higham algorithm

**Input:** $A \in \mathbb{F}^{n \times n}$, shift vector $z \in \mathbb{C}^m$

**Output:** $f \approx \left[ \|(z(j)I - A)^{-1}\|_1 \right]_{j=0:m-1}$

... // Take the maximum between the current estimates and a heuristic

$B := \frac{2}{3n} \left[ ((-1)^j(n + j - 1)) \right]_{j=0:n-1, s=0:m-1}$ // All columns are equal

$X := Q^H B$ // All columns are equal

$X := \text{MultishiftSolve}(G, z, X)$

$X := QX$

foreach $s = 0, ..., m - 1$ do $f(s) := \max(f(s), \|X(:, s)\|_1)$

Again: block algorithm from [Higham/Tisseur-2000] should be used in practice
Interleaved Hager-Higham algorithm for $L^1_\varepsilon(A)$

**Algorithm:** One-norm pseudospectra via interleaved Hager-Higham algorithm

**Input:** $A \in \mathbb{F}^{n \times n}$, shift vector $z \in \mathbb{C}^m$

**Output:** $f \approx \left[ \| (z(j)I - A)^{-1}\|_1 \right]_{j=0:m-1}$

...  

// Take the maximum between the current estimates and a heuristic

$B := \frac{2}{3n} [(-1)^j(n + j - 1)]_{j=0:n-1,s=0:m-1}$  // All columns are equal

$X := Q^H B$  // All columns are equal

$X := \text{MultishiftSolve}(G, z, X)$

$X := QX$

**foreach** $s = 0, ..., m - 1$ **do** $f(s) := \max(f(s), \|X(:, s)\|_1)$

Again: block algorithm from [Higham/Tisseur-2000] should be used in practice
Batching and deflation

- Maximum number of simultaneous shifts constrained by memory
- Could bring in new shift after each deflation, but easier to break into batches
- In practice, only small number of iterations needed per shift...
Batching and deflation

- Maximum number of simultaneous shifts constrained by memory
- Could bring in new shift after each deflation, but easier to break into batches
- In practice, only small number of iterations needed per shift...
Batching and deflation

- Maximum number of simultaneous shifts constrained by memory
- Could bring in new shift after each deflation, but easier to break into batches
- In practice, only small number of iterations needed per shift...
Outline

Van Loan’s algorithm and the Demmel matrix

Brief overview of pseudospectra

Previous work

High-performance batched analogues

A brief example of the python interface

Results

Conclusions and future work
import math, El
n = 100    # matrix size
realRes = imagRes = 100  # grid resolution

# Display an instance of the Fox−Li/Landau matrix
A = El.DistMatrix(El.zTag)
El.FoxLi(A, n)
El.Display(A, "Fox−Li matrix")

# Display its spectral portrait
portrait = El.SpectralPortrait(A, realRes, imagRes)
El.EntrywiseMap(portrait, math.log10)
El.Display(portrait, "Spectral portrait of Fox−Li matrix")

# Display its singular values
s = El.SVD(A)
El.EntrywiseMap(s, math.log10)
El.Display(s, "log10(svd(A))")
Investigating the Fox-Li matrix with python
Investigating the Fox-Li matrix with python
Investigating the Fox-Li matrix with python
Outline

Van Loan’s algorithm and the Demmel matrix

Brief overview of pseudospectra

Previous work

High-performance batched analogues

A brief example of the python interface

Results

Conclusions and future work
FoxLi(15k), $\Omega = (-1.2, 1.2)^2$, 256$^2$ pixels, 10 its

$$(Au)(x) = \sqrt{iF/\pi} \int_{-1}^{1} e^{iF(x-y)^2} u(y) \, dy, \quad F = 16\pi$$

30 sec/iter on 256 cores of Stampede (256 r.h.s./core, >4 TFlops)
FoxLi(15k), $\Omega = (-1.2, 1.2)^2$, 256$^2$ pixels, 20 its

$$(Au)(x) = \sqrt{iF/\pi} \int_{-1}^{1} e^{iF(x-y)^2} u(y) \, dy, \quad F = 16\pi$$

30 sec/iter on 256 cores of Stampede (256 r.h.s./core, >4 TFlops)
FoxLi(15k), $\Omega = (-1.2, 1.2)^2$, 256$^2$ pixels, 30 its

$$(\mathcal{A}u)(x) = \sqrt{\frac{iF}{\pi}} \int_{-1}^{1} e^{iF(x-y)^2} u(y) \, dy, \quad F = 16\pi$$

30 sec/iter on 256 cores of Stampede (256 r.h.s./core, >4 TFlops)
FoxLi(15k), $\Omega = (-1.2, 1.2)^2$, 256$^2$ pixels, 40 its

$$(Au)(x) = \sqrt{iF/\pi} \int_{-1}^{1} e^{iF(x-y)^2} u(y) \, dy, \quad F = 16\pi$$

30 sec/iter on 256 cores of Stampede (256 r.h.s./core, >4 TFlops)
FoxLi(15k), $\Omega = (-1.2, 1.2)^2$, 256$^2$ pixels, 50 its

$$(Au)(x) = \sqrt{iF/\pi} \int_{-1}^{1} e^{iF(x-y)^2} u(y) \, dy, \quad F = 16\pi$$

30 sec/iter on 256 cores of Stampede (256 r.h.s./core, >4 TFlops)
FoxLi(15k) work, $\Omega = (-1.2, 1.2)^2$, 256$^2$ pix, 10 its
FoxLi (15k) work, $\Omega = (-1.2, 1.2)^2$, $256^2$ pix, 20 its
FoxLi (15k) work, $\Omega = (-1.2, 1.2)^2$, $256^2$ pix, 30 its
FoxLi (15k) work, $\Omega = (-1.2, 1.2)^2$, 256$^2$ pix, 40 its
FoxLi(15k) work, $\Omega = (-1.2, 1.2)^2$, $256^2$ pix, 50 its
Uniform(20k), $\Omega = (-103, 103)^2$, $1024^2$ pix

4 $256^2$ pieces: 75 sec/iter on 256 nodes of Blue Gene/Q, (64 r.h.s./core, >11 TFlops), 1175 sec for Schur
Uniform(20k), $\Omega = (-10.3, 10.3)^2$, $1024^2$ pix

4 $256^2$ pieces: 75 sec/iter on 256 nodes of Blue Gene/Q, (64 r.h.s./core, >11 TFlops), 1175 sec for Schur
Outline

Van Loan’s algorithm and the Demmel matrix

Brief overview of pseudospectra

Previous work

High-performance batched analogues

A brief example of the python interface

Results

Conclusions and future work
Convergence issues

- Typically small number of iterations required for visual convergence.
- Given current batch iteration, can output image after each fixed number of iterations
- Problematic pixels typically uninteresting (resolvent gradient is small)
Future directions

- Interleaved block one-norm pseudospectral algorithm
- More intelligent interpolation and stopping criteria
- Investigate (preconditioned) 2D/3D wave equations
- Better understanding of practical limits of SDC EVD
- Optional projection onto relevant eigenspaces
- Complex distributed Hessenberg QR/QZ with AED...
- Black-box high-performance generalized pseudospectra
Future directions

▶ Interleaved block one-norm pseudospectral algorithm
▶ More intelligent interpolation and stopping criteria
▶ Investigate (preconditioned) 2D/3D wave equations
▶ Better understanding of practical limits of SDC EVD
▶ Optional projection onto relevant eigenspaces
▶ Complex distributed Hessenberg QR/QZ with AED...
▶ Black-box high-performance generalized pseudospectra
Future directions

- Interleaved block one-norm pseudospectral algorithm
- More intelligent interpolation and stopping criteria
- Investigate (preconditioned) 2D/3D wave equations
- Better understanding of practical limits of SDC EVD
- Optional projection onto relevant eigenspaces
- Complex distributed Hessenberg QR/QZ with AED...
- Black-box high-performance generalized pseudospectra
Future directions

- Interleaved block one-norm pseudospectral algorithm
- More intelligent interpolation and stopping criteria
- Investigate (preconditioned) 2D/3D wave equations
- Better understanding of practical limits of SDC EVD
- Optional projection onto relevant eigenspaces
- Complex distributed Hessenberg QR/QZ with AED...
- Black-box high-performance generalized pseudospectra
Future directions

- Interleaved block one-norm pseudospectral algorithm
- More intelligent interpolation and stopping criteria
- Investigate (preconditioned) 2D/3D wave equations
- Better understanding of practical limits of SDC EVD
- Optional projection onto relevant eigenspaces
- Complex distributed Hessenberg QR/QZ with AED...
- Black-box high-performance generalized pseudospectra
Future directions

- Interleaved block one-norm pseudospectral algorithm
- More intelligent interpolation and stopping criteria
- Investigate (preconditioned) 2D/3D wave equations
- Better understanding of practical limits of SDC EVD
- Optional projection onto relevant eigenspaces
- Complex distributed Hessenberg QR/QZ with AED...
- Black-box high-performance generalized pseudospectra
Future directions

- Interleaved block one-norm pseudospectral algorithm
- More intelligent interpolation and stopping criteria
- Investigate (preconditioned) 2D/3D wave equations
- Better understanding of practical limits of SDC EVD
- Optional projection onto relevant eigenspaces
- Complex distributed Hessenberg QR/QZ with AED...
- Black-box high-performance generalized pseudospectra
Acknowledgments and Availability

Support

Computational resources

My hosts
Jakub Kurzak and Jack Dongarra

Availability
Elemental is available under the New BSD License at libelemental.org

Questions?
Generalized multishift (quasi-)TRSM

\[ FXD - GX = Y \]

As long as a 2x2 block is not split:

\[
\left( (F_{1,1}X_1D - G_{1,1}X_1) + (F_{1,2}X_2D - G_{1,2}X_2) \right) = \begin{pmatrix} Y_1 \\ Y_2 \end{pmatrix}
\]

If \( F_{2,2} \) and \( G_{2,2} \) are small and square, have each process locally solve for a few right-hand sides of \( X_2 \). Then, form

\[
\hat{Y}_1 := Y_1 - (F_{1,2}X_2D - G_{1,2}X_2)
\]

via a parallel GEMM with each column of \( F_{1,2}X_2 \) appropriately scaled afterwards.

All that is left is to recurse on

\[
F_{1,1}X_1D - G_{1,1}X_1 = \hat{Y}_1.
\]

When \( F = I \), algorithm is much simpler.
Generalized multishift (quasi-)TRSM

\[ FXD - GX = Y \]

As long as a 2x2 block is not split:

\[
\begin{pmatrix}
(F_{1,1}X_1 D - G_{1,1} X_1) + (F_{1,2}X_2 D - G_{1,2} X_2) \\
F_{2,2}X_2 D - G_{2,2} X_2
\end{pmatrix} = \begin{pmatrix} Y_1 \\ Y_2 \end{pmatrix}
\]

If \( F_{2,2} \) and \( G_{2,2} \) are small and square, have each process locally solve for a few right-hand sides of \( X_2 \). Then, form

\[ \hat{Y}_1 := Y_1 - (F_{1,2}X_2 D - G_{1,2} X_2) \]

via a parallel GEMM with each column of \( F_{1,2}X_2 \) appropriately scaled afterwards.

All that is left is to recurse on

\[ F_{1,1}X_1 D - G_{1,1} X_1 = \hat{Y}_1. \]

When \( F = I \), algorithm is much simpler.
Generalized multishift (quasi-)TRSM

\[ FXD - GX = Y \]

As long as a 2x2 block is not split:

\[
\begin{pmatrix}
(F_{1,1}X_1D - G_{1,1}X_1) + (F_{1,2}X_2D - G_{1,2}X_2) \\
F_{2,2}X_2D - G_{2,2}X_2
\end{pmatrix} = 
\begin{pmatrix}
Y_1 \\
Y_2
\end{pmatrix}
\]

If \( F_{2,2} \) and \( G_{2,2} \) are small and square, have each process locally solve for a few right-hand sides of \( X_2 \). Then, form

\[ \hat{Y}_1 := Y_1 - (F_{1,2}X_2D - G_{1,2}X_2) \]

via a parallel GEMM with each column of \( F_{1,2}X_2 \) appropriately scaled afterwards.

All that is left is to recurse on

\[ F_{1,1}X_1D - G_{1,1}X_1 = \hat{Y}_1. \]

When \( F = I \), algorithm is much simpler.
Generalized multishift (quasi-)TRSM

\[ FXD - GX = Y \]

As long as a 2x2 block is not split:

\[
\begin{pmatrix}
(F_{1,1}X_1 D - G_{1,1} X_1) + (F_{1,2}X_2 D - G_{1,2} X_2) \\
F_{2,2}X_2 D - G_{2,2} X_2
\end{pmatrix} = \begin{pmatrix} Y_1 \\ Y_2 \end{pmatrix}
\]

If \( F_{2,2} \) and \( G_{2,2} \) are small and square, have each process locally solve for a few right-hand sides of \( X_2 \). Then, form

\[
\hat{Y}_1 := Y_1 - (F_{1,2}X_2 D - G_{1,2} X_2)
\]

via a parallel GEMM with each column of \( F_{1,2}X_2 \) appropriately scaled afterwards.

All that is left is to recurse on

\[ F_{1,1} X_1 D - G_{1,1} X_1 = \hat{Y}_1. \]

When \( F = I \), algorithm is much simpler.