High-performance computation of pseudospectra

Jack Poulson [1,2,3]   Greg Henry [4]

[1]Dept. of Mathematics
Stanford University

[2]Inst. of Comput. and Math. Eng.
Stanford University

[3]School of Comput. Science and Eng.
Georgia Institute of Technology

[4]Intel Corporation

SIAM CSE15
Salt Lake City, Utah
March 14, 2015

Context / Trojan horse for current research

- Have spent the past few months developing distributed sparse-direct Interior Point Methods for QPs,

$$\min_x \frac{1}{2}x^T Q x + c^T x$$
$$\text{s.t. } Ax = b, Gx + s = h, s \geq 0$$

- Ran into fundamental instability of dynamic regularization [Altman/Gondzio-1998,Li/Demmel-1998] closely related to pseudospectra
- There exists a simple, small *symmetric quasi-definite matrix* [Vanderbei-1993] with condition number less than 15 that can cause dynamic regularization to catastrophically fail

Context / Trojan horse for current research

- Have spent the past few months developing distributed sparse-direct Interior Point Methods for QPs,

$$\min_x \frac{1}{2} x^T Q x + c^T x$$
$$\text{s.t. } Ax = b, Gx + s = h, s \geq 0$$

- Ran into fundamental instability of dynamic regularization [Altman/Gondzio-1998,Li/Demmel-1998] closely related to pseudospectra
- There exists a simple, small *symmetric quasi-definite matrix* [Vanderbei-1993] with condition number less than 15 that can cause dynamic regularization to catastrophically fail

Context / Trojan horse for current research

- Have spent the past few months developing distributed sparse-direct Interior Point Methods for QPs,

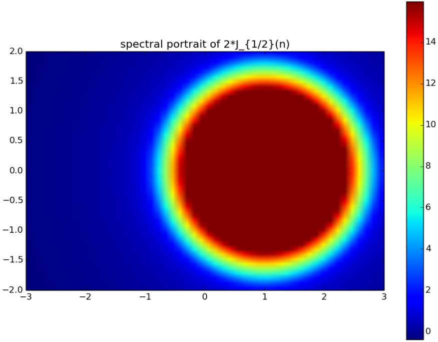
$$\min_x \frac{1}{2}x^T Q x + c^T x$$
$$\text{s.t. } Ax = b, Gx + s = h, s \geq 0$$

- Ran into fundamental instability of dynamic regularization [Altman/Gondzio-1998,Li/Demmel-1998] closely related to pseudospectra
- There exists a simple, small *symmetric quasi-definite matrix* [Vanderbei-1993] with condition number less than 15 that can cause dynamic regularization to catastrophically fail

The irrelevance of pivot magnitudes

$$M_n \equiv \begin{pmatrix} 1 & 2 & & & & \\ 2 & 5 & 2 & & & \\ & & & \ddots & & \\ & 2 & 5 & \ddots & & \\ & & \ddots & \ddots & & \\ & & & & 5 & 2 \\ & & & & 2 & 5 \end{pmatrix} = \begin{pmatrix} 1 & & & & & \\ 2 & 1 & & & & \\ & 2 & 1 & & & \\ & & \ddots & \ddots & & \\ & & & & 1 & \\ & & & & 2 & 1 \end{pmatrix} \begin{pmatrix} 1 & 2 & & & & \\ & 1 & 2 & & & \\ & & & \ddots & & \\ & & 1 & \ddots & & \\ & & & \ddots & & \\ & & & & 1 & 2 \\ & & & & & 1 \end{pmatrix}$$

While all eigenvalues of $L_n \equiv \text{chol}(M_n)$ are 1, $\|(L_n - \xi I)^{-1}\|_2$ is large within a disk containing the origin



The irrelevance of pivot magnitudes

$$M_n \equiv \begin{pmatrix} 1 & 2 & & & & \\ 2 & 5 & 2 & & & \\ & & & \ddots & & \\ & 2 & 5 & \ddots & & \\ & & \ddots & \ddots & & \\ & & & & 5 & 2 \\ & & & & 2 & 5 \end{pmatrix} = \begin{pmatrix} 1 & & & & & \\ 2 & 1 & & & & \\ & 2 & 1 & & & \\ & & \ddots & \ddots & & \\ & & & & 1 & \\ & & & & 2 & 1 \end{pmatrix} \begin{pmatrix} 1 & 2 & & & & \\ & 1 & 2 & & & \\ & & & \ddots & & \\ & & 1 & \ddots & & \\ & & & \ddots & & \\ & & & & 1 & 2 \\ & & & & & 1 \end{pmatrix}$$

But the condition number of

$$K_n \equiv \begin{pmatrix} M_n & I \\ I & -I \end{pmatrix}$$

for $n = 100$ is less than 15 while causing dynamic regularization to fail catastrophically when solving against $L_n = \text{chol}(M_n)$.

Van Loan's algorithm and the Demmel matrix

- Efficient algorithm for evaluating resolvent norm over vertical line in complex plane [Van Loan-1985]
- Proposed as means of computing distance to nearest stable matrix; counter-example given in [Demmel-1987]
- Counter-example involved first (computer) pseudospectral plot

Van Loan's algorithm and the Demmel matrix

- Efficient algorithm for evaluating resolvent norm over vertical line in complex plane [Van Loan-1985]
- Proposed as means of computing distance to nearest stable matrix; counter-example given in [Demmel-1987]
- Counter-example involved first (computer) pseudospectral plot

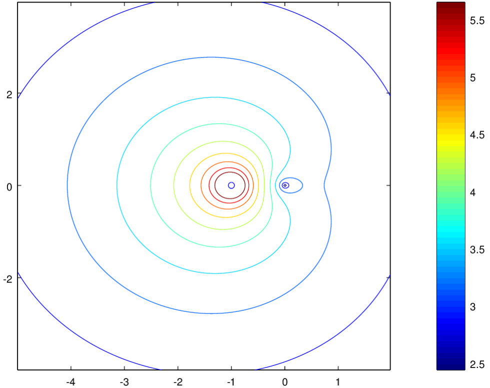
Van Loan's algorithm and the Demmel matrix

- Efficient algorithm for evaluating resolvent norm over vertical line in complex plane [Van Loan-1985]
- Proposed as means of computing distance to nearest stable matrix; counter-example given in [Demmel-1987]
- Counter-example involved first (computer) pseudospectral plot

Van Loan's algorithm and the Demmel matrix

- Efficient algorithm for evaluating resolvent norm over vertical line in complex plane [Van Loan-1985]
- Proposed as means of computing distance to nearest stable matrix; counter-example given in [Demmel-1987]
- Counter-example involved first (computer) pseudospectral plot

$$D(n, \beta) = (\beta J_{-\beta^{-1},n})^{-1}, \quad n = 3,\ \beta = 100$$



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 $\epsilon > 0$:

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

Reinvented many times [Varah-1967,Landau-1975,Godunov et al.-1982,Trefethen-1990,Hinrichsen/Pritchard-1992,...]

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

Most common tool is EigTool [Wright-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 $\epsilon > 0$:

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

Reinvented many times [Varah-1967,Landau-1975,Godunov et al.-1982,Trefethen-1990,Hinrichsen/Pritchard-1992,...]

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

Most common tool is EigTool [Wright-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 $\epsilon > 0$:

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

Reinvented many times [Varah-1967,Landau-1975,Godunov et al.-1982,Trefethen-1990,Hinrichsen/Pritchard-1992,...]

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

Most common tool is EigTool [Wright-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 $\epsilon > 0$:

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

Reinvented many times [Varah-1967,Landau-1975,Godunov et al.-1982,Trefethen-1990,Hinrichsen/Pritchard-1992,...]

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

Most common tool is EigTool [Wright-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 $\epsilon > 0$:

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

Reinvented many times [Varah-1967,Landau-1975,Godunov et al.-1982,Trefethen-1990,Hinrichsen/Pritchard-1992,...]

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

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

(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 $p = 2$,

$$\mathcal{L}_\epsilon^2(A) = \{\xi \in \mathbb{C} : \sigma_{\min}(\xi I - G) < \epsilon\}.$$

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 $p = 2$,

$$\mathcal{L}_\epsilon^2(A) = \{\xi \in \mathbb{C} : \sigma_{\min}(\xi I - G) < \epsilon\}.$$

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 $p = 2$,

$$\mathcal{L}_\epsilon^2(A) = \{\xi \in \mathbb{C} : \sigma_{\min}(\xi I - G) < \epsilon\} .$$

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 $\mathcal{L}^2_\epsilon(A)$

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: $\{\phi(\xi)\}_{\xi\in\Omega} \approx \{\|(\xi I - A)^{-1}\|_2\}_{\xi\in\Omega}$
$G :=$ Schur(A), RealSchur(A), or 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 $j = 0, ..., k-1$ **do**
            `//` $(\xi I - G)^{-H}(\xi I - G)^{-1} V_j = V_j H_j + v_j(\beta_j e_j)^H$
            $x_j := (\xi I - G)^{-H}(\xi I - G)^{-1} v_j$
            Expand Arnoldi decomposition using x_j
        $[\lambda, v_0] :=$ MaxEig(H_k)
        $\phi(\xi) :=$ RealPart(λ)

[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: $\{\phi(\xi)\}_{\xi\in\Omega} \approx \{\|(A-\xi I)^{-1}\|_1\}_{\xi\in\Omega}$
$[Q, G] :=$ Schur(A), RealSchur(A), 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
    $x := u$
    $y := Q(G-\xi I)^{-1}Q^H x$
    $w := Q(G-\xi I)^{-H}Q^H \mathrm{sign}(y)$
    $u := e_j$ where $|w(j)| = \|w\|_\infty$
    $k := k+1$
  until $(k \geq 2$ and $\|w\|_\infty \leq w^H x)$ or $k \geq 5$
  $\phi(\xi) := \|y\|_1$
  // Take the maximum between the current estimate and a heuristic
  $b := \frac{2}{3n}[(-1)^j(n+j-1)]_{j=0:n-1}$
  $x := Q(G-\xi I)^{-1}Q^H b$
  $\phi(\xi) := \max(\phi(\xi), \|x\|_1)$

Block algorithm [Higham/Tisseur-2000] should be used in practice

[Hager-1984,Higham-1988] approach for $\mathcal{L}^1_\epsilon(A)$

Algorithm: One-norm pseudospectra via Hager-Higham algorithm

Input: $A \in \mathbb{F}^{n\times n}$, $\Omega \subset \mathbb{C}$
Output: $\{\phi(\xi)\}_{\xi\in\Omega} \approx \{\|(A-\xi I)^{-1}\|_1\}_{\xi\in\Omega}$
$[Q, G] :=$ Schur(A), RealSchur(A), 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
    $x := u$
    $y := Q(G-\xi I)^{-1}Q^H x$
    $w := Q(G-\xi I)^{-H}Q^H \mathrm{sign}(y)$
    $u := e_j$ where $|w(j)| = \|w\|_\infty$
    $k := k+1$
  until $(k \geq 2$ *and* $\|w\|_\infty \leq w^H x)$ *or* $k \geq 5$
  $\phi(\xi) := \|y\|_1$
  // Take the maximum between the current estimate and a heuristic
  $b := \frac{2}{3n}[(-1)^j(n+j-1)]_{j=0:n-1}$
  $x := Q(G-\xi I)^{-1}Q^H b$
  $\phi(\xi) := \max(\phi(\xi), \|x\|_1)$

Block algorithm [Higham/Tisseur-2000] should be used in practice

(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 2×2
- Generalized multishift (quasi-)triangular solves can be handled in an analogous manner (ask about appendix if interested)
- Multishift Hessenberg solves heavily investigated for control theory [Datta et al-1994,Henry-1994]
- 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 2×2
- Generalized multishift (quasi-)triangular solves can be handled in an analogous manner (ask about appendix if interested)
- Multishift Hessenberg solves heavily investigated for control theory [Datta et al-1994,Henry-1994]
- 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 2×2
- Generalized multishift (quasi-)triangular solves can be handled in an analogous manner (ask about appendix if interested)
- Multishift Hessenberg solves heavily investigated for control theory [Datta et al-1994,Henry-1994]
- 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 2×2
- Generalized multishift (quasi-)triangular solves can be handled in an analogous manner (ask about appendix if interested)
- Multishift Hessenberg solves heavily investigated for control theory [Datta et al-1994,Henry-1994]
- 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 2×2
- Generalized multishift (quasi-)triangular solves can be handled in an analogous manner (ask about appendix if interested)
- Multishift Hessenberg solves heavily investigated for control theory [Datta et al-1994,Henry-1994]
- 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 2×2
- Generalized multishift (quasi-)triangular solves can be handled in an analogous manner (ask about appendix if interested)
- Multishift Hessenberg solves heavily investigated for control theory [Datta et al-1994,Henry-1994]
- 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: $f \approx \left[\|(z(s)I - A)^{-1}\|_2\right]_{s=0:m-1}$
$G :=$ Schur(A), RealSchur(A), or Hessenberg(A)
Initialize each column of $W_0 \in \mathbb{C}^{n\times m}$ to have unit two-norm
$\mathcal{I} := (0, ..., m-1)$
while $\mathcal{I} \neq \emptyset$ **do**
  for $j = 0, ..., k-1$ **do**
    // $(z(s)I - G)^{-H}(z(s)I - G)^{-1}\mathcal{W}_j(t) = \mathcal{W}_j(t)\mathcal{H}_j(t) + W_j(:,t)(b_j(t)e_j)^H, \ \forall s = \mathcal{I}(t)$
    $X :=$ MultishiftSolve$(G, z(\mathcal{I}), W_j)$
    $X :=$ MultishiftSolve$(G^H, \bar{z}(\mathcal{I}), X)$
    Expand Arnoldi decompositions
  foreach $s = \mathcal{I}(t)$ **do**
    $[\lambda, W_0(:,t)] :=$ MaxEig$(\mathcal{H}_k(t))$
    $f(s) :=$ RealPart(λ)
    if *converged* **then**
      Delete $\mathcal{I}(t)$ and $W_0(:,t)$

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

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: $f \approx \left[\|(z(j)I - A)^{-1}\|_1\right]_{j=0:m-1}$
$[Q, G] := \text{Schur}(A)$, $\text{RealSchur}(A)$, or $\text{Hessenberg}(A)$
Initialize each column of $U \in \mathbb{C}^{n\times m}$ to have unit one-norm
$\mathcal{I} := (0, ..., m-1)$
$k := 0$
while $\mathcal{I} \neq \emptyset$ **do**
    $X := U$
    $Y := Q^H X$
    $Y := \text{MultishiftSolve}(G, z(\mathcal{I}), Y)$
    $Y := QY$
    $W := Q^H \text{sign}(Y)$
    $W := \text{MultishiftSolve}(G^H, \bar{z}(\mathcal{I}), W)$
    $W := QW$
    $k := k + 1$
    foreach $s = \mathcal{I}(t)$ **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 $\mathcal{I}(t)$ and $U(:, t)$
        else $U(:, t) := e_j$ where $|w(j)| = \|w\|_\infty$

...

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

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`

$X :=$ 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 $\mathcal{L}^1_\epsilon(A)$

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`

$X :=$ 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...

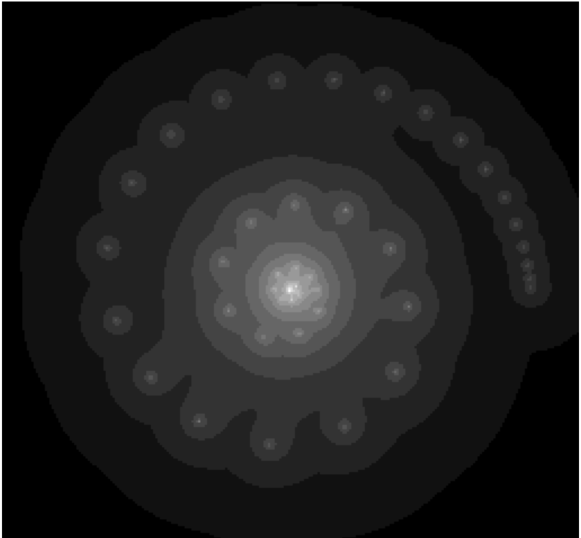
Batching and deflation

- 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...

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

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

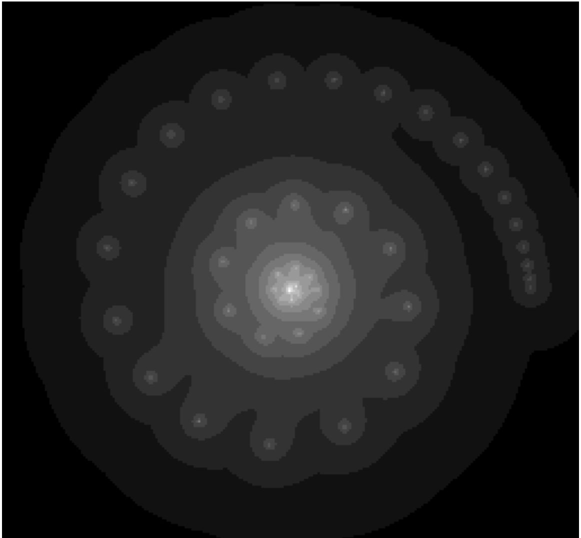
30 sec/iter on 256 cores of Stampede (256 r.h.s./core, >4 TFlops)



FoxLi(15k), $\Omega = (-1.2, 1.2)^2$, 256^2 pixels, 20 its

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

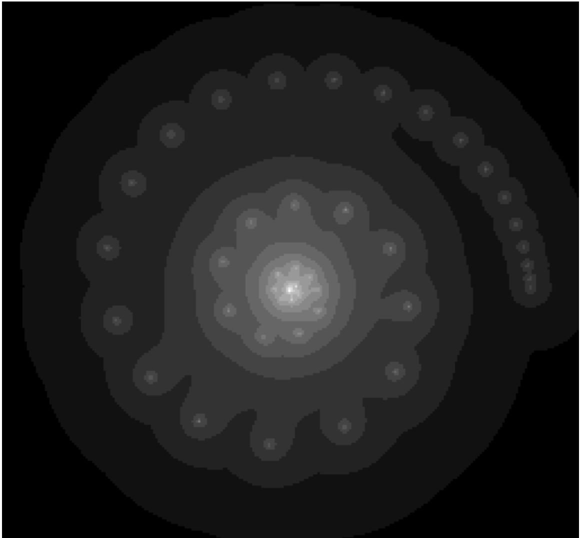
30 sec/iter on 256 cores of Stampede (256 r.h.s./core, >4 TFlops)



FoxLi(15k), $\Omega = (-1.2, 1.2)^2$, 256^2 pixels, 30 its

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

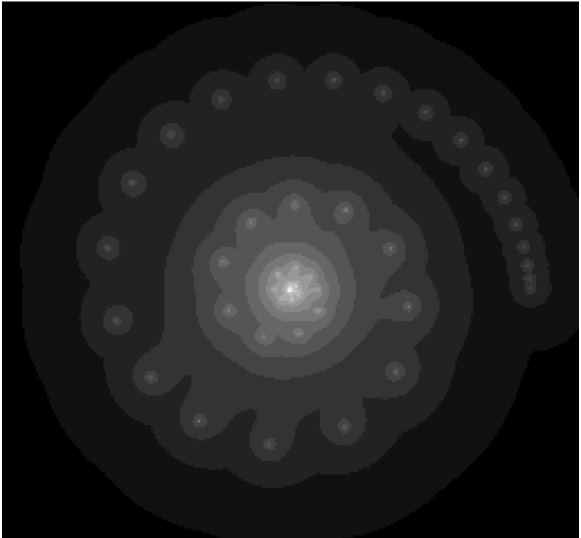
30 sec/iter on 256 cores of Stampede (256 r.h.s./core, >4 TFlops)



FoxLi(15k), $\Omega = (-1.2, 1.2)^2$, 256^2 pixels, 40 its

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

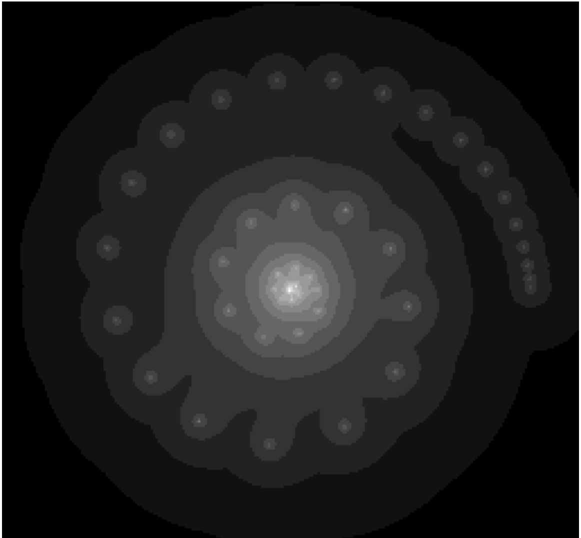
30 sec/iter on 256 cores of Stampede (256 r.h.s./core, >4 TFlops)



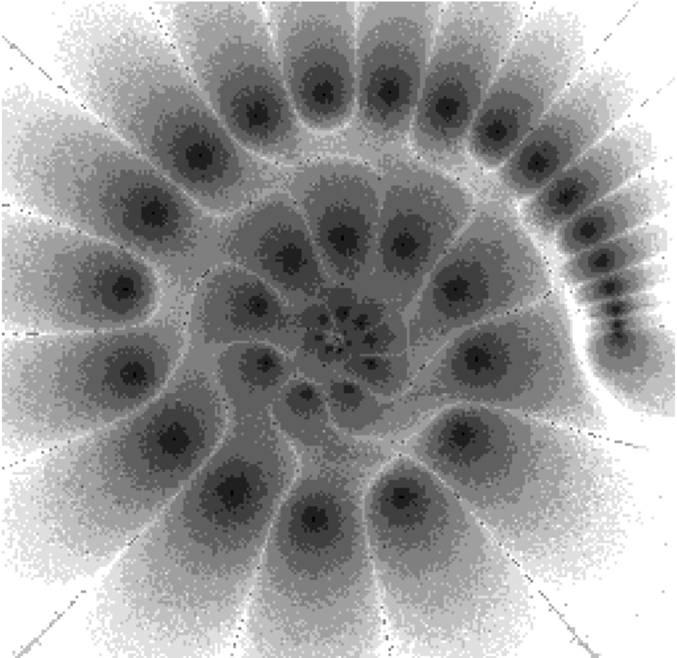
FoxLi(15k), $\Omega = (-1.2, 1.2)^2$, 256^2 pixels, 50 its

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

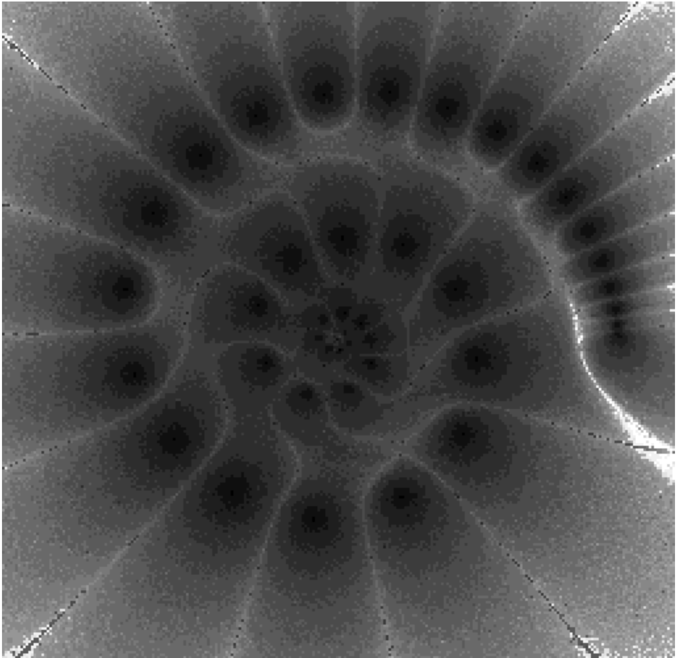
30 sec/iter on 256 cores of Stampede (256 r.h.s./core, >4 TFlops)



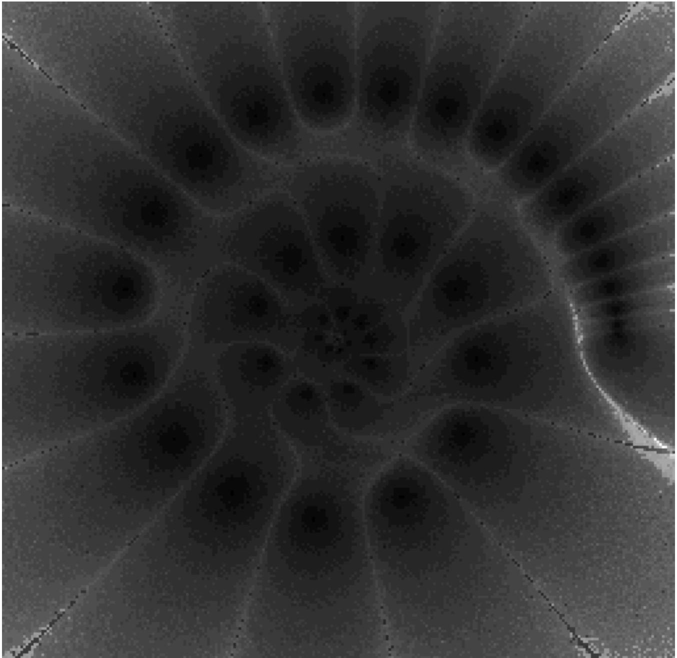
FoxLi(15k) work, $\Omega = (-1.2, 1.2)^2$, 256^2 pix, 10 its



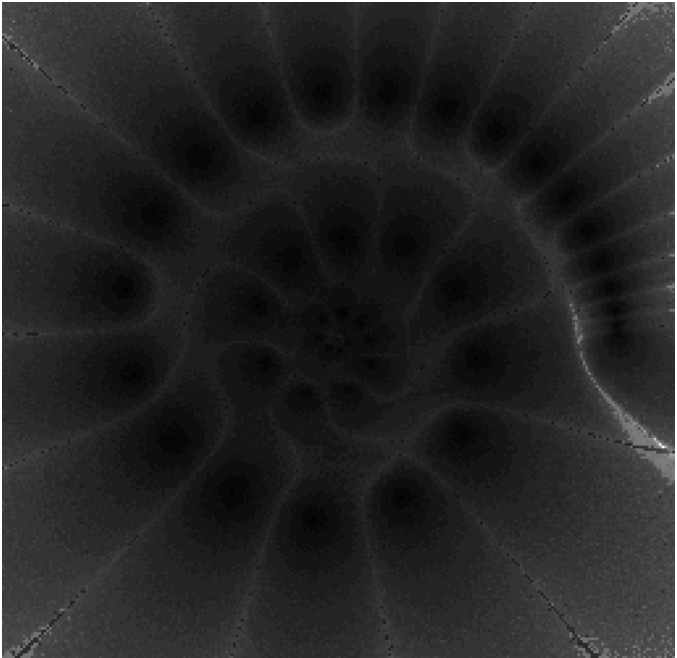
FoxLi(15k) work, $\Omega = (-1.2, 1.2)^2$, 256^2 pix, 20 its



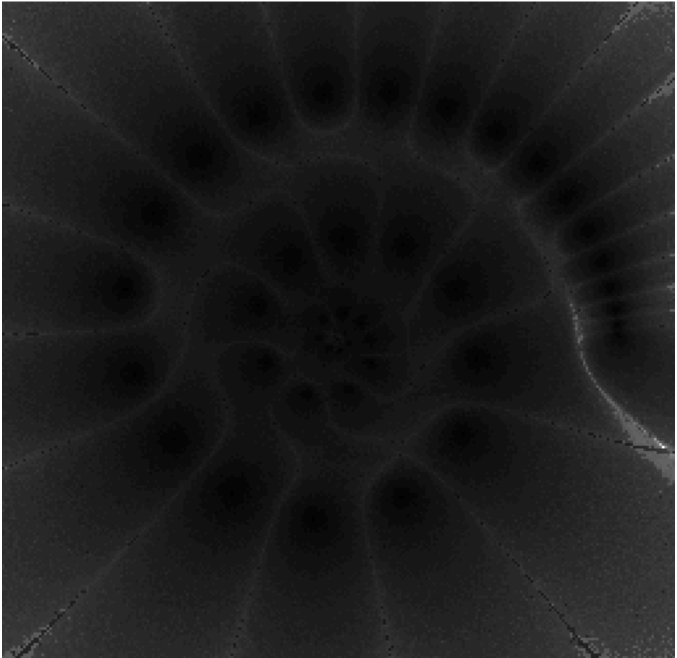
FoxLi(15k) work, $\Omega = (-1.2, 1.2)^2$, 256^2 pix, 30 its



FoxLi(15k) work, $\Omega = (-1.2, 1.2)^2$, 256^2 pix, 40 its

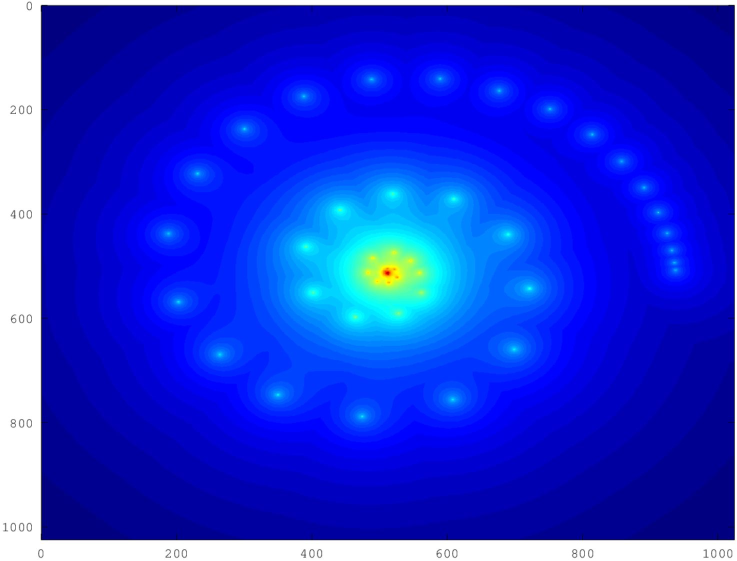


FoxLi(15k) work, $\Omega = (-1.2, 1.2)^2$, 256^2 pix, 50 its



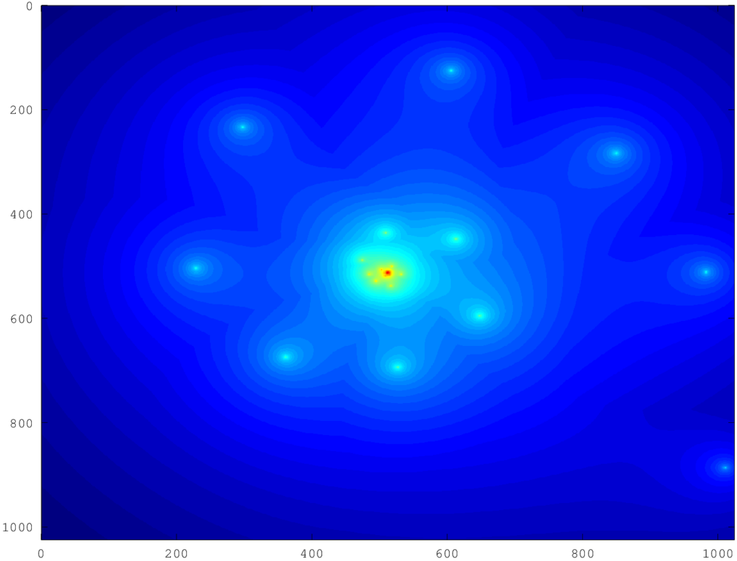
FoxLi(20k), $\Omega = (-1.2, 1.2)^2$, 1024^2 pix

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



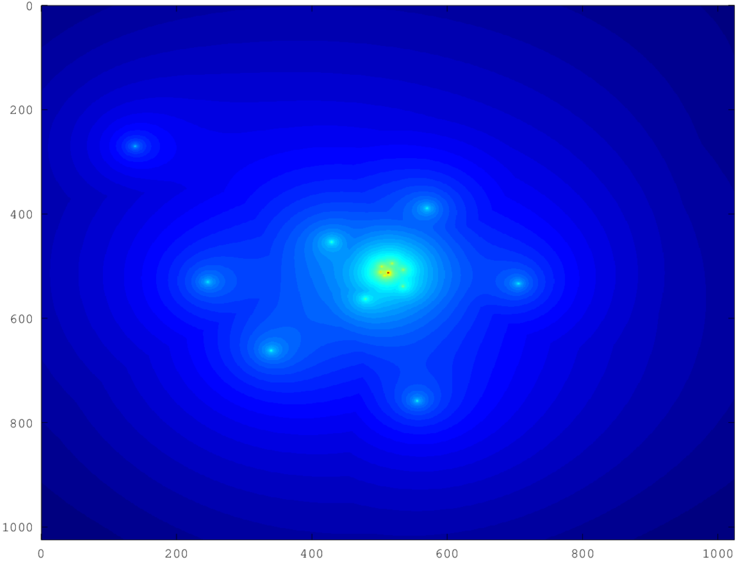
FoxLi(20k), $\Omega = (-0.12, 0.12)^2$, 1024^2 pix

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



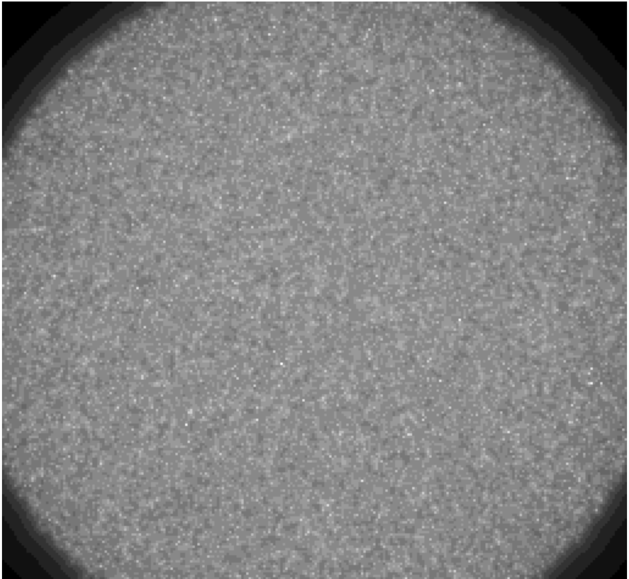
FoxLi(20k), $\Omega = (-0.012, 0.012)^2$, 1024^2 pix

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



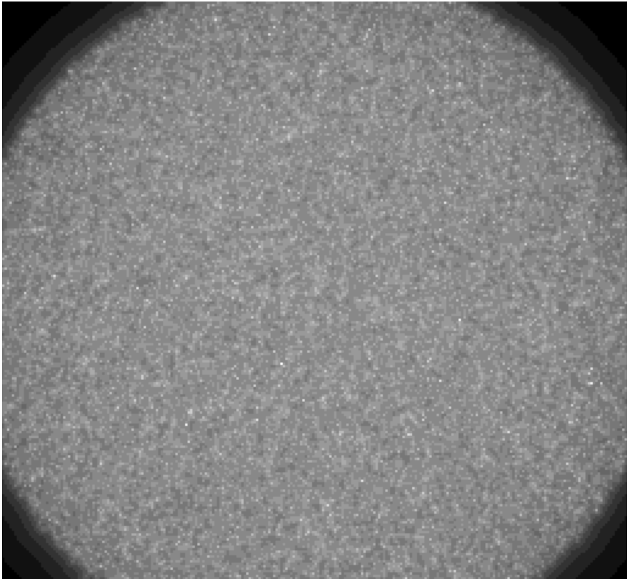
Uniform(15k), $\Omega = (-64, 64)^2$, 256^2 pix, 10 its

30 sec/iter on 256 cores of Stampede
(256 r.h.s./core, >4 TFlops)



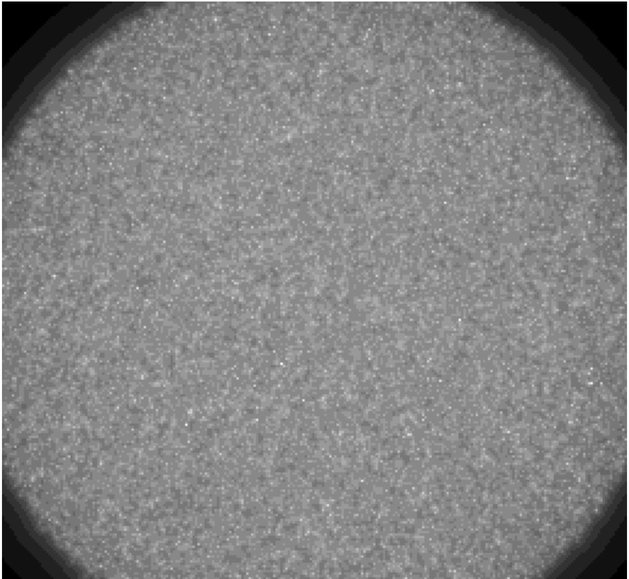
Uniform(15k), $\Omega = (-64, 64)^2$, 256^2 pix, 20 its

30 sec/iter on 256 cores of Stampede
(256 r.h.s./core, >4 TFlops)



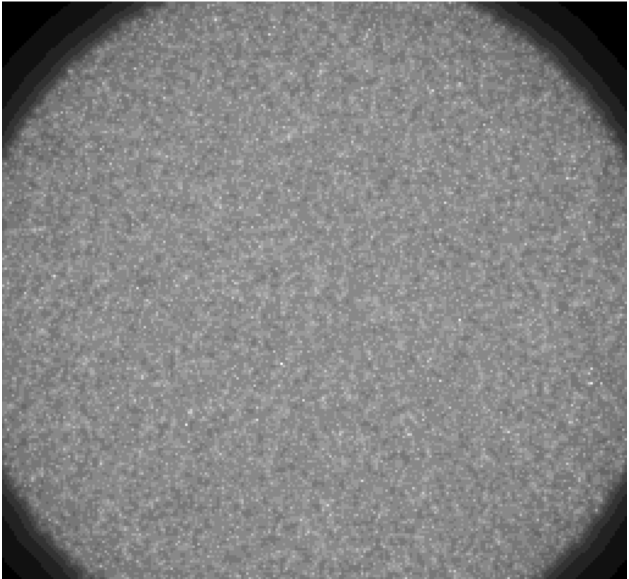
Uniform(15k), $\Omega = (-64, 64)^2$, 256^2 pix, 30 its

30 sec/iter on 256 cores of Stampede
(256 r.h.s./core, >4 TFlops)



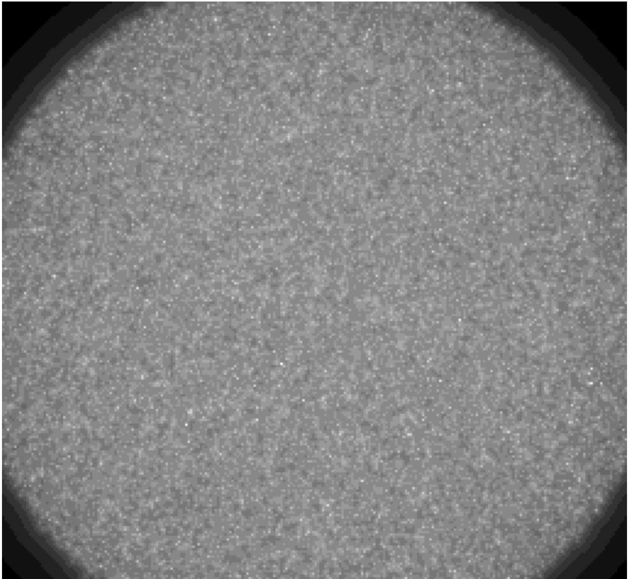
Uniform(15k), $\Omega = (-64, 64)^2$, 256^2 pix, 40 its

30 sec/iter on 256 cores of Stampede
(256 r.h.s./core, >4 TFlops)



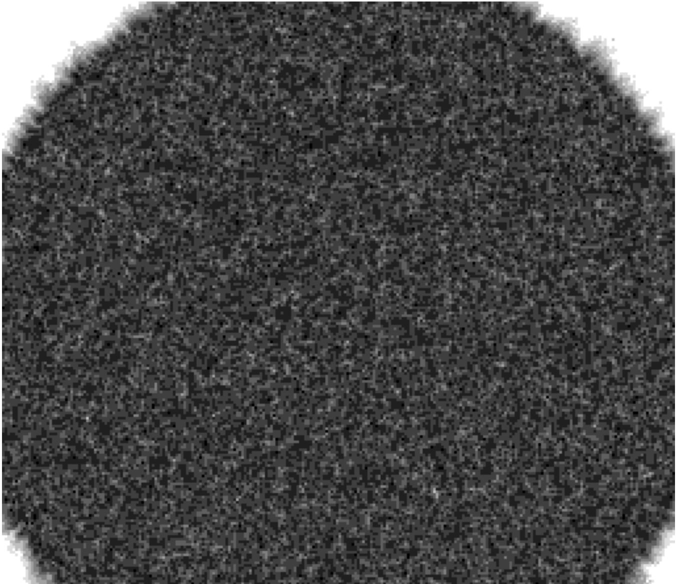
Uniform(15k), $\Omega = (-64, 64)^2$, 256^2 pix, 50 its

30 sec/iter on 256 cores of Stampede
(256 r.h.s./core, >4 TFlops)



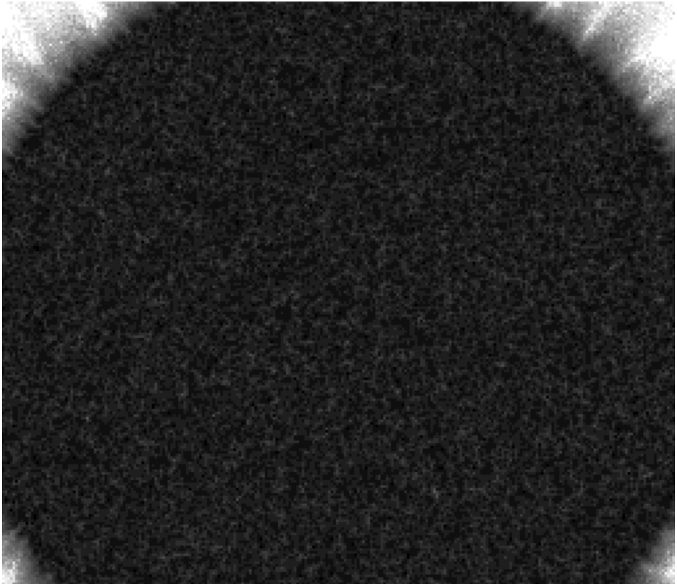
Uniform(15k) work, $\Omega = (-64, 64)^2$, 256^2 pix, 10 its

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



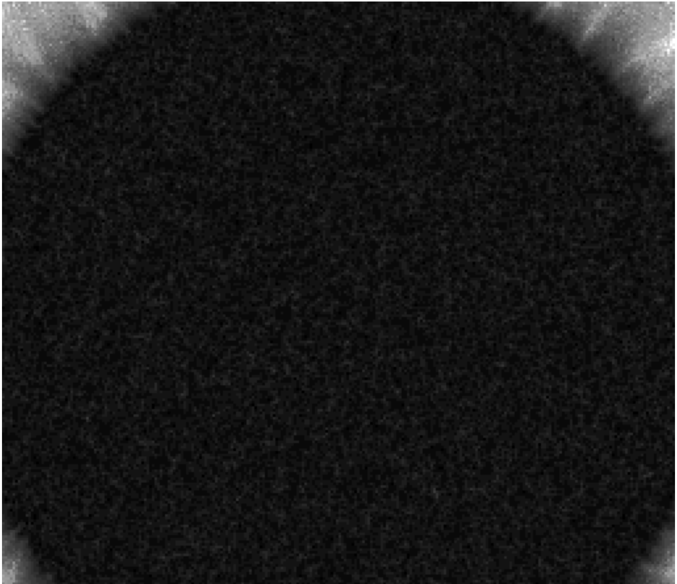
Uniform(15k) work, $\Omega = (-64, 64)^2$, 256^2 pix, 20 its

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



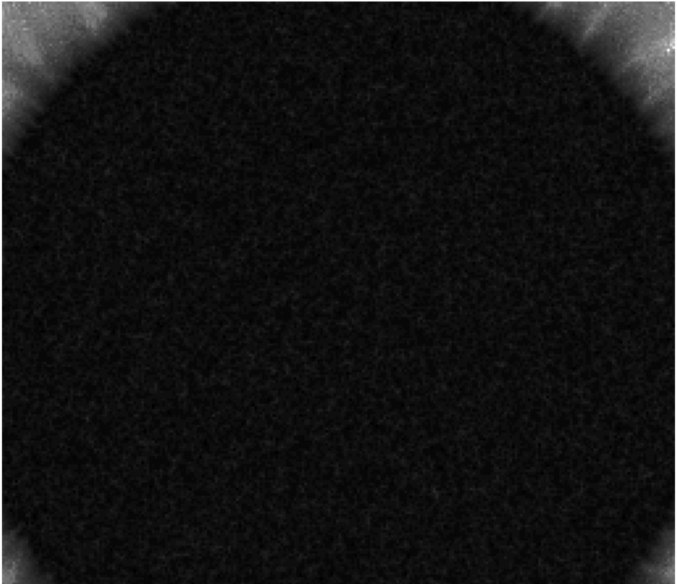
Uniform(15k) work, $\Omega = (-64, 64)^2$, 256^2 pix, 30 its

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



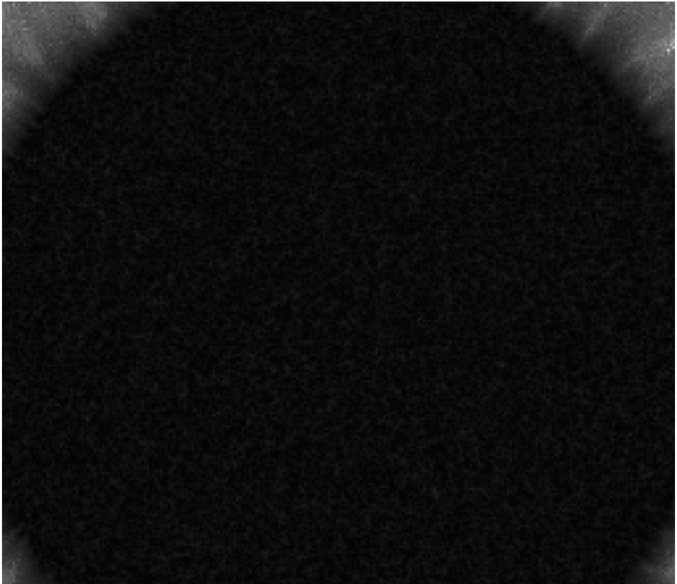
Uniform(15k) work, $\Omega = (-64, 64)^2$, 256^2 pix, 40 its

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



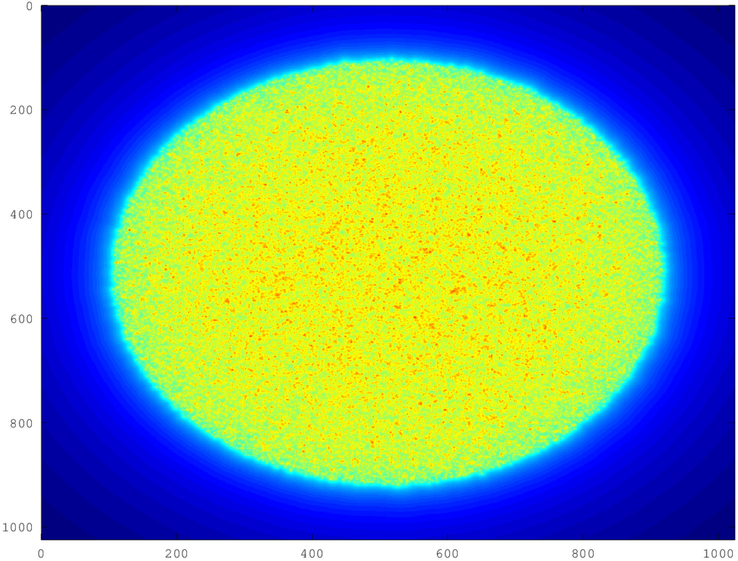
Uniform(15k) work, $\Omega = (-64, 64)^2$, 256^2 pix, 50 its

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



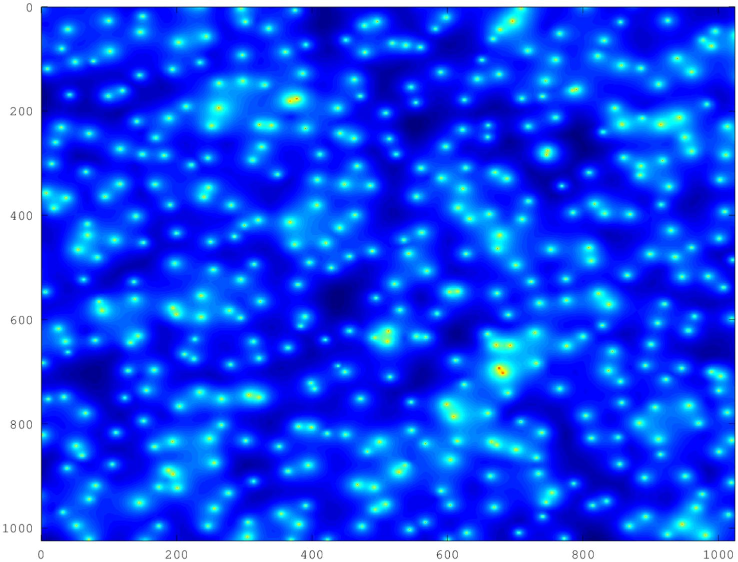
Uniform(20k), $\Omega = (-103, 103)^2$, 1024^2 pix

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



Future work

- Interleaved block one-norm pseudospectral algorithm
- Extend EigTool to support high-performance interleaved algorithms (perhaps with accelerator support for the TRSM-like operation)
- More intelligent interpolation and stopping criteria
- Optional projection onto relevant eigenspaces

Future work

- Interleaved block one-norm pseudospectral algorithm
- Extend EigTool to support high-performance interleaved algorithms (perhaps with accelerator support for the TRSM-like operation)
- More intelligent interpolation and stopping criteria
- Optional projection onto relevant eigenspaces

Future work

- Interleaved block one-norm pseudospectral algorithm
- Extend EigTool to support high-performance interleaved algorithms (perhaps with accelerator support for the TRSM-like operation)
- More intelligent interpolation and stopping criteria
- Optional projection onto relevant eigenspaces

Future work

- Interleaved block one-norm pseudospectral algorithm
- Extend EigTool to support high-performance interleaved algorithms (perhaps with accelerator support for the TRSM-like operation)
- More intelligent interpolation and stopping criteria
- Optional projection onto relevant eigenspaces

Acknowledgments and Availability

Support

Computational resources

Thanks:
Michael Saunders for extended discussions on regularizing QSD systems
Stephen Boyd and AJ Friend for extended discussions on IPMs

Minisymposium organizers:
Piotr Luszczek, Stan Tomov, and Azzam Haidar

Availability
Elemental is available under the New BSD License at `libelemental.org`
(Come see my poster on Elemental in the CSE Software mini on Monday!)

Questions?

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:

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

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:

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

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:

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

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.