dqds with aggressive early deflation

Yuji Nakatsukasa

Department of Mathematics

University of California, Davis

joint with

Kensuke Aishima and Ichitaro Yamazaki

Outline

1. Backgrounds
   - singular value decomposition
   - dqds (with conventional deflation)
   - aggressive early deflation for Hessenberg QR
2. dqds with aggressive early deflation
   - Aggdef(1): direct attempt
   - Aggdef(2): more efficient and stable
   - convergence factor analysis
   - zero-shift $\Rightarrow$ parallel implementation
3. Numerical experiments

Singular Value Decomposition (SVD)

Any matrix $A \in \mathbb{C}^{m \times n}$ $(m \geq n)$ can be decomposed into

$$A = U \begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_n \end{bmatrix} V^* \equiv U \Sigma V^*,$$

where $U^* U = V^* V = I_n$.

- $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$: singular values of A
- $\sigma_i = \sqrt{\lambda_i(A^* A)}$

Applications of the SVD

- Rank of A: number of positive σ_i
- Low-rank approximation: $A \simeq \sum_{i=n-k}^{n} \sigma_i u_i v_i^*$ for $k \ll n$
- Column space, row space, null space, ...
- Condition number: $\kappa_2(A) = \sigma_1/\sigma_n$
- Singular value thresholding
- Inverse $A = V\Sigma^{-1}U^*$, pseudoinverse
- Polar decomposition: $A = U\Sigma V^* = (UV^*) \cdot (V\Sigma V^*) = U_p H$
- Image compression
- ...

Standard SVD algorithm

1. Reduce A to bidiagonal form via Householder reflections H_L, H_R

$$A = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix} \overset{H_L}{\rightarrow} \begin{bmatrix} * & * & * & * \\ & * & * & * \\ & * & * & * \\ & * & * & * \end{bmatrix} \overset{H_R}{\rightarrow} \begin{bmatrix} * & * & & \\ & * & * & * \\ & * & * & * \\ & * & * & * \end{bmatrix}$$

$$\overset{H_L}{\rightarrow} \begin{bmatrix} * & * & & \\ & * & * & * \\ & & * & * \\ & & * & * \end{bmatrix} \overset{H_R}{\rightarrow} \begin{bmatrix} * & * & & \\ & * & * & \\ & & * & * \\ & & * & * \end{bmatrix} \overset{H_L}{\rightarrow} \begin{bmatrix} * & * & & \\ & * & * & \\ & & * & * \\ & & & * \end{bmatrix} \equiv B.$$

–[Golub and Kahan (1965)]

$A = U_A B V_A^*$, where $U_A = (\prod H_L)^*, V_A = \prod H_R$.

2. Compute SVD of $B = U_B \Sigma V_B^*$.
   - Compute singular values Σ via **dqds**.
   - Compute singular vectors U_B, V_B via inverse iteration.
3. SVD: $A = (U_A U_B)\Sigma(V_B^* V_A^*) = U\Sigma V^*$.

Standard SVD algorithm

1. Reduce A to bidiagonal form via Householder reflections H_L, H_R

$$A = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix} \overset{H_L}{\rightarrow} \begin{bmatrix} * & * & * & * \\ & * & * & * \\ & * & * & * \\ & * & * & * \end{bmatrix} \overset{H_R}{\rightarrow} \begin{bmatrix} * & * & & \\ & * & * & * \\ & * & * & * \\ & * & * & * \end{bmatrix}$$

$$\overset{H_L}{\rightarrow} \begin{bmatrix} * & * & & \\ & * & * & * \\ & & * & * \\ & & * & * \end{bmatrix} \overset{H_R}{\rightarrow} \begin{bmatrix} * & * & & \\ & * & * & \\ & & * & * \\ & & * & * \end{bmatrix} \overset{H_L}{\rightarrow} \begin{bmatrix} * & * & & \\ & * & * & \\ & & * & * \\ & & & * \end{bmatrix} \equiv B.$$

–[Golub and Kahan (1965)]

$A = U_A B V_A^*$, where $U_A = (\prod H_L)^*, V_A = \prod H_R$.

2. Compute SVD of $B = U_B \Sigma V_B^*$.
   - Compute singular values Σ via **dqds**.
   - Compute singular vectors U_B, V_B via inverse iteration.
3. SVD: $A = (U_A U_B)\Sigma(V_B^* V_A^*) = U\Sigma V^*$.

Computing bidiagonal singular values: historical aspect

- QR algorithm applied to $B^T B$: yields absolute accuracy

–[Golub and Kahan (1965)]

$$|\sigma_i - \widehat{\sigma}_i| \leq O(n) \cdot \sigma_{\max} \epsilon$$

Computing bidiagonal singular values: historical aspect

- QR algorithm applied to $B^T B$: yields absolute accuracy

  –[Golub and Kahan (1965)]

$$|\sigma_i - \widehat{\sigma}_i| \leq O(n) \cdot \sigma_{\max}\epsilon$$

- Refined QR: attains high relative accuracy

  –[Demmel and Kahan (1990)]

$$|\sigma_i - \widehat{\sigma}_i| \leq 69n^2\sigma_i\epsilon$$

Computing bidiagonal singular values: historical aspect

- QR algorithm applied to $B^T B$: yields absolute accuracy

–[Golub and Kahan (1965)]

$$|\sigma_i - \widehat{\sigma}_i| \leq O(n) \cdot \sigma_{\max}\epsilon$$

- Refined QR: attains high relative accuracy

–[Demmel and Kahan (1990)]

$$|\sigma_i - \widehat{\sigma}_i| \leq 69n^2\sigma_i\epsilon$$

- **dqds**: 4-fold speedup + higher relative accuracy

–[Fernando and Parlett (1994)]

$$|\sigma_i - \widehat{\sigma}_i| \leq 4n\sigma_i\epsilon$$

Computing bidiagonal singular values: historical aspect

- QR algorithm applied to $B^T B$: yields absolute accuracy

–[Golub and Kahan (1965)]

$$|\sigma_i - \widehat{\sigma}_i| \le O(n) \cdot \sigma_{\max}\epsilon$$

- Refined QR: attains high relative accuracy

–[Demmel and Kahan (1990)]

$$|\sigma_i - \widehat{\sigma}_i| \le 69n^2\sigma_i\epsilon$$

- **dqds**: 4-fold speedup + higher relative accuracy

–[Fernando and Parlett (1994)]

$$|\sigma_i - \widehat{\sigma}_i| \le 4n\sigma_i\epsilon$$

	$\frac{\lvert\sigma_{\max}-\widehat{\sigma}_{\max}\rvert}{\sigma_{\max}}$	$\frac{\lvert\sigma_{\min}-\widehat{\sigma}_{\min}\rvert}{\sigma_{\min}}$
QR	10^{-15}	10^{-1}
Refined QR	10^{-15}	10^{-14}
dqds	10^{-15}	10^{-15}

Typical relative accuracy for B with $\sigma_{\max} = 1,\ \sigma_{\min}(B) = 10^{-15}$

dqd and dqds (differential quotient difference with shifts)

B: real upper-bidiagonal

- dqd: Cholesky algorithm, compute $\widehat{B}$ s.t. $\widehat{B}^T\widehat{B} = BB^T$, let $B := \widehat{B}$, repeat.
  - $B \to \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_n)$.
  - bottom off-diagonal convergence factor $\sqrt{\frac{\sigma_n^2}{\sigma_{n-1}^2}}$.

dqd and dqds (differential quotient difference with shifts)

B: real upper-bidiagonal

- dqd: Cholesky algorithm, compute $\widehat{B}$ s.t. $\widehat{B}^T\widehat{B} = BB^T$, let $B := \widehat{B}$, repeat.
  - $B \to \text{diag}(\sigma_1, \sigma_2, \dots, \sigma_n)$.
  - bottom off-diagonal convergence factor $\sqrt{\dfrac{\sigma_n^2}{\sigma_{n-1}^2}}$.
- dqds: Introduce shift $s \geq 0$: $\widehat{B}^T\widehat{B} = BB^T - sI$.
  - bottom off-diagonal convergence factor $\sqrt{\dfrac{\sigma_n^2 - s}{\sigma_{n-1}^2 - s}}$.

    – Need $s \leq \sigma_n^2$ for $\widehat{B}$ to exist.

    – $s \simeq \sigma_n^2$ is optimal, but estimating is nontrivial
    (LAPACK uses a complicated shift strategy).

dqds: pseudocode

Algorithm 1 The dqds algorithm

$q_i = (B_{i,i})^2, \; e_i = (B_{i,i+1})^2$
for $m := 0, 1, \cdots$ **do**
  choose shift $s (\geq 0)$
  $d_1 := q_1 - s$
  for $i := 1, \cdots, n-1$ **do**
    $q_i := d_i + e_i$
    $e_i := e_i q_{i+1} / q_i$
    $d_{i+1} := d_i q_{i+1} / q_i - s$
  end for
  $q_n := d_n$
end for

$$B = \begin{bmatrix} \sqrt{q_1} & \sqrt{e_1} & & & \\ & \sqrt{q_2} & \sqrt{e_2} & & \\ & & \ddots & \ddots & \\ & & & \sqrt{q_{n-1}} & \sqrt{e_{n-1}} \\ & & & & \sqrt{q_n} \end{bmatrix}$$

progress

get shift s $\rightarrow$ dqds $\rightarrow$ get shift s $\rightarrow$ dqds $\rightarrow \cdots$

time $\longrightarrow$

- root-free
- $e_i \rightarrow 0, \; \sqrt{q_i} \rightarrow \sigma_i$ with guaranteed high relative accuracy
- sequential nature, has been difficult to parallelize

dqds with conventional deflation strategy

Typically, running dqds results in

$$B = \begin{bmatrix} \sqrt{q}_1 & \sqrt{e}_1 & & & \\ & \sqrt{q}_2 & \sqrt{e}_2 & & \\ & & \ddots & \ddots & \\ & & & \sqrt{q}_{n-1} & \sqrt{e}_{n-1} \\ & & & & \sqrt{q}_n \end{bmatrix}$$

dqds with conventional deflation strategy

Typically, running dqds results in

$$B = \begin{bmatrix} \sqrt{q}_1 & \sqrt{e}_1 & & & \\ & \sqrt{q}_2 & \sqrt{e}_2 & & \\ & & \ddots & \ddots & \\ & & & \sqrt{q}_{n-1} & \sqrt{e}_{n-1} \\ & & & & \sqrt{q}_n \end{bmatrix}$$

$e_{n-1} \to 0$ with convergence factor $\dfrac{\sigma_n^2 - s}{\sigma_{n-1}^2 - s} < 1$.

dqds with conventional deflation strategy

Typically, running dqds results in

$$B = \begin{bmatrix} \sqrt{q_1} & \sqrt{e_1} & & & \\ & \sqrt{q_2} & \sqrt{e_2} & & \\ & & \ddots & \ddots & \\ & & & \sqrt{q_{n-1}} & \sqrt{e_{n-1}} \\ & & & & \sqrt{q_n} \end{bmatrix}$$

$e_{n-1} \to 0$ with convergence factor $\dfrac{\sigma_n^2 - s}{\sigma_{n-1}^2 - s} < 1$.

dqds with conventional deflation strategy

Typically, running dqds results in

$$B = \begin{bmatrix} \sqrt{q}_1 & \sqrt{e}_1 & & & \\ & \sqrt{q}_2 & \sqrt{e}_2 & & \\ & & \ddots & \ddots & \\ & & & \sqrt{q}_{n-1} & \sqrt{e}_{n-1} \\ & & & & \sqrt{q}_n \end{bmatrix}$$

$e_{n-1} \to 0$ with convergence factor $\dfrac{\sigma_n^2 - s}{\sigma_{n-1}^2 - s} < 1$.

dqds with conventional deflation strategy

Typically, running dqds results in

$$B = \begin{bmatrix} \sqrt{q}_1 & \sqrt{e}_1 & & & \\ & \sqrt{q}_2 & \sqrt{e}_2 & & \\ & & \ddots & \ddots & \\ & & & \sqrt{q}_{n-1} & \sqrt{e}_{n-1} \\ & & & & \sqrt{q}_n \end{bmatrix}$$

$e_{n-1} \to 0$ with convergence factor $\dfrac{\sigma_n^2 - s}{\sigma_{n-1}^2 - s} < 1$.

dqds with conventional deflation strategy

Typically, running dqds results in

$$B = \begin{bmatrix} \sqrt{q}_1 & \sqrt{e}_1 & & & \\ & \sqrt{q}_2 & \sqrt{e}_2 & & \\ & & \ddots & \ddots & \\ & & & \sqrt{q}_{n-1} & \sqrt{e}_{n-1} \\ & & & & \sqrt{q}_n \end{bmatrix}$$

$e_{n-1} \to 0$ with convergence factor $\dfrac{\sigma_n^2 - s}{\sigma_{n-1}^2 - s} < 1$.

dqds with conventional deflation strategy

Typically, running dqds results in

$$B = \begin{bmatrix} \sqrt{q}_1 & \sqrt{e}_1 & & & \\ & \sqrt{q}_2 & \sqrt{e}_2 & & \\ & & \ddots & \ddots & \\ & & & \sqrt{q}_{n-1} & \scriptstyle\sqrt{e}_{n-1} \\ & & & & \sqrt{q}_n \end{bmatrix}$$

$e_{n-1} \to 0$ with convergence factor $\dfrac{\sigma_n^2 - s}{\sigma_{n-1}^2 - s} < 1.$

- when e_{n-1} is negligibly small, set it to 0.

dqds with conventional deflation strategy

Typically, running dqds results in

$$B = \begin{bmatrix} \sqrt{q_1} & \sqrt{e_1} & & & \\ & \sqrt{q_2} & \sqrt{e_2} & & \\ & & \ddots & \ddots & \\ & & & \sqrt{q_{n-1}} & 0 \\ & & & & \sqrt{q_n} \end{bmatrix}$$

$e_{n-1} \to 0$ with convergence factor $\dfrac{\sigma_n^2 - s}{\sigma_{n-1}^2 - s} < 1$.

- when e_{n-1} is negligibly small, set it to 0.
  $\to$ $\sqrt{q_n}$ is isolated: converged singular value.
  $\to$ remove last row and column (deflation), repeat.

dqds with conventional deflation strategy

Typically, running dqds results in

$$B = \begin{bmatrix} \sqrt{q}_1 & \sqrt{e}_1 & & & \\ & \sqrt{q}_2 & \sqrt{e}_2 & & \\ & & \ddots & \ddots & \\ & & & \sqrt{q}_{n-1} & {\color{red}0} \\ & & & & \sqrt{q}_n \end{bmatrix} \to \begin{bmatrix} \sqrt{q}_1 & \sqrt{e}_1 & & \\ & \sqrt{q}_2 & \ddots & \\ & & \ddots & \sqrt{e}_{n-2} \\ & & & \sqrt{q}_{n-1} \end{bmatrix}$$

$e_{n-1} \to 0$ with convergence factor $\dfrac{\sigma_n^2 - s}{\sigma_{n-1}^2 - s} < 1$.

- when e_{n-1} is negligibly small, set it to 0.
  $\to$ $\sqrt{q}_n$ is isolated: converged singular value.
  $\to$ remove last row and column (deflation), repeat.

Aggressive deflation for nonHermitian eigenproblems

–[Braman, Byers, Mathias (2003)]

$$H = \begin{array}{c} \\ n-k-1 \\ 1 \\ k \end{array} \begin{array}{c} \begin{array}{ccc} n-k-1 & 1 & k \end{array} \\ \begin{bmatrix} H_{11} & H_{12} & H_{13} \\ H_{21} & H_{22} & H_{23} \\ 0 & H_{32} & H_{33} \end{bmatrix} \end{array} \qquad k : \text{window size}$$

- Compute Schur decomposition $H_{33} = VTV^*$ (T is triangular)

$$\begin{bmatrix} I & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & V \end{bmatrix}^* \begin{bmatrix} H_{11} & H_{12} & H_{13} \\ H_{21} & H_{22} & H_{23} \\ 0 & H_{32} & H_{33} \end{bmatrix} \begin{bmatrix} I & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & V \end{bmatrix} = \begin{bmatrix} H_{11} & H_{12} & H_{13}V \\ H_{21} & H_{22} & H_{23}V \\ 0 & t & T \end{bmatrix}.$$

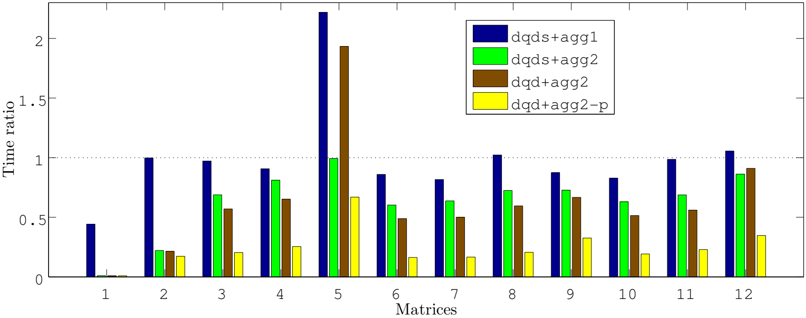
- Find negligible elements in $t = \begin{bmatrix} * \\ * \\ \vdots \\ * \end{bmatrix}$ and deflate.

$\Rightarrow$ Results in significant speed-up.

Contributions

- Incorporate aggressive early deflation into dqds to speed it up
  - direct version: Aggdef(1)
  - refined version: Aggdef(2), efficient and stable
- Convergence analysis

  ⇒ leads to a parallelizable algorithm



Aggressive deflation for dqds -version 1: Aggdef(1)

1. Compute the "small" SVD of k-by-k $B_2 = U\Sigma V^T$ in

$$B = \begin{bmatrix} B_1 & \sqrt{e_{n-k}} \\ & B_2 \end{bmatrix}.$$

2. Compute $\begin{bmatrix} I_{n-k} & \\ & U^T \end{bmatrix} B \begin{bmatrix} I_{n-k} & \\ & V \end{bmatrix}$:

$$\begin{bmatrix} I_{n-k} & \\ & U^T \end{bmatrix} \begin{bmatrix} * & * & & & & & \\ & \ddots & \ddots & & & & \\ & & * & * & & & \\ & & & * & * & & \\ & & & & * & * & \\ & & & & & * & * \\ & & & & & & * \end{bmatrix} \begin{bmatrix} I_{n-k} & \\ & V \end{bmatrix} = \begin{bmatrix} \ddots & \ddots & & & & & \\ & * & * & & & & \\ & & * & {\color{red}*} & {\color{red}*} & {\color{red}*} & {\color{red}*} \\ & & & * & & & \\ & & & & * & & \\ & & & & & * & \\ & & & & & & * \end{bmatrix}.$$

3. Find negligible elements in ${\color{red}*}$, remove corresponding rows and columns.
4. Reduce matrix to bidiagonal form, resume dqds.

Aggressive deflation for dqds -version 1: Aggdef(1)

1. Compute the "small" SVD of k-by-k $B_2 = U\Sigma V^T$ in

$$B = \begin{bmatrix} B_1 & \sqrt{e_{n-k}} \\ & B_2 \end{bmatrix}.$$

2. Compute $\begin{bmatrix} I_{n-k} & \\ & U^T \end{bmatrix} B \begin{bmatrix} I_{n-k} & \\ & V \end{bmatrix}$:

$$\begin{bmatrix} I_{n-k} & \\ & U^T \end{bmatrix} \begin{bmatrix} * & * & & & & & \\ & \ddots & \ddots & & & & \\ & & * & * & & & \\ & & & * & * & & \\ & & & & * & * & \\ & & & & & * & * \\ & & & & & & * \end{bmatrix} \begin{bmatrix} I_{n-k} & \\ & V \end{bmatrix} = \begin{bmatrix} \ddots & \ddots & & & & & \\ & * & * & & & & \\ & & * & \color{red}{*} & \color{red}{*} & \color{red}{*} & \color{red}{*} \\ & & & * & & & \\ & & & & * & & \\ & & & & & * & \\ & & & & & & * \end{bmatrix}.$$

3. Find negligible elements in $\color{red}{*}$, remove corresponding rows and columns.
4. Reduce matrix to bidiagonal form, resume dqds.

$\Rightarrow$ Problem in **speed + stability**

Efficient and stable Aggressive deflation: Aggdef(2)

1. Compute $\widehat{B_2}$ s.t. $\widehat{B_2^T}\widehat{B_2} = B_2^T B_2 - sI$, where $s = (\sigma_{\min}(B_2))^2$
2. Apply Givens rotations to $\widehat{B_2}$:

$$\begin{bmatrix} * & * & & & \\ & * & * & & \\ & & * & * & \\ & & & * & * \\ & & & & \end{bmatrix} \rightarrow \begin{bmatrix} * & * & & & \\ & * & * & & \\ & & * & * & x \\ & & & * & 0 \\ & & & & \end{bmatrix} \rightarrow \begin{bmatrix} * & * & & & \\ & * & * & & x \\ & & * & * & 0 \\ & & & * & 0 \\ & & & & \end{bmatrix} \rightarrow \begin{bmatrix} * & * & & & x \\ & * & * & & 0 \\ & & * & * & 0 \\ & & & * & 0 \\ & & & & \end{bmatrix}$$

   Set $x \leftarrow 0$ when negligible.

3. Update B_2: $B_2^T B_2 = \widehat{B_2^T}\widehat{B_2} + sI$, deflate, repeat.

Efficient and stable Aggressive deflation: Aggdef(2)

1. Compute $\widehat{B_2}$ s.t. $\widehat{B_2^T}\widehat{B_2} = B_2^T B_2 - sI$, where $s = (\sigma_{\min}(B_2))^2$
2. Apply Givens rotations to $\widehat{B_2}$:

$$\begin{bmatrix} * & * & & & \\ & * & * & & \\ & & * & * & \\ & & & * & * \\ & & & & \end{bmatrix} \rightarrow \begin{bmatrix} * & * & & & \\ & * & * & & \\ & & * & * & x \\ & & & * & 0 \\ & & & & \end{bmatrix} \rightarrow \begin{bmatrix} * & * & & & \\ & * & * & & x \\ & & * & * & 0 \\ & & & * & 0 \\ & & & & \end{bmatrix} \rightarrow \begin{bmatrix} * & * & & & x \\ & * & * & & 0 \\ & & * & * & 0 \\ & & & * & 0 \\ & & & & \end{bmatrix}$$

   Set $x \leftarrow 0$ when negligible.

3. Update B_2: $B_2^T B_2 = \widehat{B_2^T}\widehat{B_2} + sI$, deflate, repeat.

Lemma

Aggdef(1) and Aggdef(2) are mathematically equivalent.

Efficient and stable Aggressive deflation: Aggdef(2)

1. Compute $\widehat{B}_2$ s.t. $\widehat{B}_2^T\widehat{B}_2 = B_2^TB_2 - sI$, where $s = (\sigma_{\min}(B_2))^2$
2. Apply Givens rotations to $\widehat{B}_2$:

$$\begin{bmatrix} * & * & & & \\ & * & * & & \\ & & * & * & \\ & & & * & * \\ & & & & \end{bmatrix} \to \begin{bmatrix} * & * & & & \\ & * & * & & \\ & & * & * & x \\ & & & * & 0 \\ & & & & \end{bmatrix} \to \begin{bmatrix} * & * & & & \\ & * & * & & x \\ & & * & * & 0 \\ & & & * & 0 \\ & & & & \end{bmatrix} \to \begin{bmatrix} * & * & & & x \\ & * & * & & 0 \\ & & * & * & 0 \\ & & & * & 0 \\ & & & & \end{bmatrix}$$

   Set $x \leftarrow 0$ when negligible.

3. Update B_2: $B_2^TB_2 = \widehat{B}_2^T\widehat{B}_2 + sI$, deflate, repeat.

Lemma

Aggdef(1) and Aggdef(2) are mathematically equivalent.

	flops	rel. accuracy
Aggdef(1)	$O(k^2)$	conditional
Aggdef(2)	$O(k\ell)$	guaranteed

k: window size ($\simeq \sqrt{n}$), ℓ: number of singular values deflated by Aggdef

Aggdef(2) preserves high relative accuracy

By a mixed forward-backward relative error analysis, we establish:

Theorem

$$1 - 8n\epsilon \leq \frac{\sigma_i(\widetilde{B})}{\sigma_i(B)} \leq 1 + 8n\epsilon$$

for $i = 1, \ldots, n$.

Aggdef(2) preserves high relative accuracy

By a mixed forward-backward relative error analysis, we establish:

Theorem

$$1 - 8n\epsilon \leq \frac{\sigma_i(\widetilde{B})}{\sigma_i(B)} \leq 1 + 8n\epsilon$$

for $i = 1, \ldots, n$.

- Recall dqds error bound

$$1 - 4n\epsilon \leq \frac{\sigma_i(\widetilde{B})}{\sigma_i(B)} \leq 1 + 4n\epsilon$$

- Calling Aggdef(2) maintains high relative accuracy.

Recap: Aggdef(2)

1. Compute $\widehat{B_2}$ s.t. $\widehat{B_2}^T \widehat{B_2} = B_2^T B_2 - sI$.
2. Apply Givens rotations to $\widehat{B_2}$:

$$\begin{bmatrix} * & * & & & \\ & * & * & & \\ & & * & * & \\ & & & * & * \\ & & & & \end{bmatrix} \rightarrow \begin{bmatrix} * & * & & & \\ & * & * & & \\ & & * & * & x \\ & & & * & 0 \\ & & & & \end{bmatrix} \rightarrow \begin{bmatrix} * & * & & & \\ & * & * & & x \\ & & * & * & 0 \\ & & & * & 0 \\ & & & & \end{bmatrix} \rightarrow \begin{bmatrix} * & * & & & x \\ & * & * & & 0 \\ & & * & * & 0 \\ & & & * & 0 \\ & & & & \end{bmatrix}$$

   Set $x \leftarrow 0$ when negligible.

3. Update B_2: $B_2^T B_2 = \widehat{B_2}^T \widehat{B_2} + sI$

Recap: Aggdef(2)

1. Compute $\widehat{B_2}$ s.t. $\widehat{B_2^T}\widehat{B_2} = B_2^T B_2 - sI$.
2. Apply Givens rotations to $\widehat{B_2}$:

$$\begin{bmatrix} * & * & & & \\ & * & * & & \\ & & * & * & \\ & & & * & * \\ & & & & \end{bmatrix} \rightarrow \begin{bmatrix} * & * & & & \\ & * & * & & \\ & & * & * & x \\ & & & * & 0 \\ & & & & \end{bmatrix} \rightarrow \begin{bmatrix} * & * & & & \\ & * & * & & x \\ & & * & * & 0 \\ & & & * & 0 \\ & & & & \end{bmatrix} \rightarrow \begin{bmatrix} * & * & & & x \\ & * & * & & 0 \\ & & * & * & 0 \\ & & & * & 0 \\ & & & & \end{bmatrix}$$

   Set $x \leftarrow 0$ when negligible.

3. Update B_2: $B_2^T B_2 = \widehat{B_2^T}\widehat{B_2} + sI$

Givens rotations in Aggdef(2)

$$\begin{bmatrix} * & * & & & & & \\ & * & * & & & & \\ & & * & * & & & \\ & & & * & * & & \\ & & & & * & * & \\ & & & & & * & x \\ & & & & & & \end{bmatrix}$$

Each Givens rotation yields

$$x := x \cdot \frac{e_{i-1}}{q_i + x} \leq x \cdot \frac{e_{i-1}}{q_i}.$$

Therefore, x at the top is

$$x \leq e_{n-1} \prod_{i=n-k+2}^{n-1} \frac{e_{i-1}}{q_i}.$$

Givens rotations in Aggdef(2)

$$\begin{bmatrix} * & * & & & & & \\ & * & * & & & & \\ & & * & * & & & \\ & & & * & * & & \\ & & & & * & * & x \\ & & & & & * & \\ & & & & & & \end{bmatrix}$$

Each Givens rotation yields

$$x := x \cdot \frac{e_{i-1}}{q_i + x} \leq x \cdot \frac{e_{i-1}}{q_i}.$$

Therefore, x at the top is

$$x \leq e_{n-1} \prod_{i=n-k+2}^{n-1} \frac{e_{i-1}}{q_i}.$$

Givens rotations in Aggdef(2)

$$\begin{bmatrix} * & * & & & & & \\ & * & * & & & & \\ & & * & * & & & \\ & & & * & * & & x \\ & & & & * & * & \\ & & & & & * & \\ & & & & & & \end{bmatrix}$$

Each Givens rotation yields

$$x := x \cdot \frac{e_{i-1}}{q_i + x} \leq x \cdot \frac{e_{i-1}}{q_i}.$$

Therefore, x at the top is

$$x \leq e_{n-1} \prod_{i=n-k+2}^{n-1} \frac{e_{i-1}}{q_i}.$$

Givens rotations in Aggdef(2)

$$\begin{bmatrix} * & * & & & & & \\ & * & * & & & & \\ & & * & * & & & x \\ & & & * & * & & \\ & & & & * & * & \\ & & & & & * & \\ & & & & & & \end{bmatrix}$$

Each Givens rotation yields

$$x := x \cdot \frac{e_{i-1}}{q_i + x} \leq x \cdot \frac{e_{i-1}}{q_i}.$$

Therefore, x at the top is

$$x \leq e_{n-1} \prod_{i=n-k+2}^{n-1} \frac{e_{i-1}}{q_i}.$$

Givens rotations in Aggdef(2)

$$\begin{bmatrix} * & * & & & & & \\ & * & * & & & & x \\ & & * & * & & & \\ & & & * & * & & \\ & & & & * & * & \\ & & & & & * & \\ & & & & & & \end{bmatrix}$$

Each Givens rotation yields

$$x := x \cdot \frac{e_{i-1}}{q_i + x} \leq x \cdot \frac{e_{i-1}}{q_i}.$$

Therefore, x at the top is

$$x \leq e_{n-1} \prod_{i=n-k+2}^{n-1} \frac{e_{i-1}}{q_i}.$$

Givens rotations in Aggdef(2)

$$\begin{bmatrix} * & * & & & & & x \\ & * & * & & & & \\ & & * & * & & & \\ & & & * & * & & \\ & & & & * & * & \\ & & & & & * & \\ & & & & & & \end{bmatrix}$$

Each Givens rotation yields

$$x := x \cdot \frac{e_{i-1}}{q_i + x} \leq x \cdot \frac{e_{i-1}}{q_i}.$$

Therefore, x at the top is

$$x \leq e_{n-1} \prod_{i=n-k+2}^{n-1} \frac{e_{i-1}}{q_i}.$$

Givens rotations in Aggdef(2)

$$\begin{bmatrix} * & * & & & & & x \\ & * & * & & & & \\ & & * & * & & & \\ & & & * & * & & \\ & & & & * & * & \\ & & & & & * & \\ & & & & & & \end{bmatrix}$$

Each Givens rotation yields

$$x := x \cdot \frac{e_{i-1}}{q_i + x} \leq x \cdot \frac{e_{i-1}}{q_i}.$$

Therefore, x at the top is

$$x \leq e_{n-1} \prod_{i=n-k+2}^{n-1} \frac{e_{i-1}}{q_i}.$$

$\Rightarrow$ x can be negligible even if no e_i is too small!

Aggdef(2) example

$$\begin{bmatrix} 6.00 & 0.10 & & & & & \\ & 5.00 & 0.10 & & & & \\ & & 4.00 & 0.10 & & & \\ & & & 3.00 & 0.10 & & \\ & & & & 2.00 & 0.10 & \\ & & & & & 1.00 & 1 \cdot 10^{-1} \\ & & & & & & \end{bmatrix}$$

Each Givens rotation yields

$$x := x \cdot \frac{e_{i-1}}{q_i + x} \leq x \cdot \frac{e_{i-1}}{q_i}.$$

Therefore, x at the top is

$$x \leq e_{n-1} \prod_{i=n-k+2}^{n-1} \frac{e_{i-1}}{q_i}$$

$\Rightarrow$ x can be negligible even if no e_i is too small!

Aggdef(2) example

$$\begin{bmatrix} 6.00 & 0.10 & & & & & \\ & 5.00 & 0.10 & & & & \\ & & 4.00 & 0.10 & & & \\ & & & 3.00 & 0.10 & & \\ & & & & 2.00 & \color{red}{0.10} & \\ & & & & & \color{red}{1.00} & \color{red}{1 \cdot 10^{-1}} \\ & & & & & & \end{bmatrix}$$

Each Givens rotation yields

$$x := x \cdot \frac{e_{i-1}}{q_i + x} \leq x \cdot \frac{e_{i-1}}{q_i}.$$

Therefore, x at the top is

$$x \leq e_{n-1} \prod_{i=n-k+2}^{n-1} \frac{e_{i-1}}{q_i}$$

$\Rightarrow$ x can be negligible even if no e_i is too small!

Aggdef(2) example

$$\begin{bmatrix} 6.00 & 0.10 & & & & & \\ & 5.00 & 0.10 & & & & \\ & & 4.00 & 0.10 & & & \\ & & & 3.00 & 0.10 & & \\ & & & & 2.00 & 0.09 & 9 \cdot 10^{-3} \\ & & & & & 1.10 & \\ & & & & & & \end{bmatrix}$$

Each Givens rotation yields

$$x := x \cdot \frac{e_{i-1}}{q_i + x} \leq x \cdot \frac{e_{i-1}}{q_i}.$$

Therefore, x at the top is

$$x \leq e_{n-1} \prod_{i=n-k+2}^{n-1} \frac{e_{i-1}}{q_i}$$

$\Rightarrow$ x can be negligible even if no e_i is too small!

Aggdef(2) example

$$\begin{bmatrix} 6.00 & 0.10 & & & & & \\ & 5.00 & 0.10 & & & & \\ & & 4.00 & 0.10 & & & \\ & & & 3.00 & \color{red}{0.10} & & \\ & & & & \color{red}{2.00} & 0.09 & \color{red}{9 \cdot 10^{-3}} \\ & & & & & 1.10 & \\ & & & & & & \end{bmatrix}$$

Each Givens rotation yields

$$x := x \cdot \frac{e_{i-1}}{q_i + x} \leq x \cdot \frac{e_{i-1}}{q_i}.$$

Therefore, x at the top is

$$x \leq e_{n-1} \prod_{i=n-k+2}^{n-1} \frac{e_{i-1}}{q_i}$$

$\Rightarrow$ x can be negligible even if no e_i is too small!

Aggdef(2) example

$$\begin{bmatrix} 6.00 & 0.10 & & & & & \\ & 5.00 & 0.10 & & & & \\ & & 4.00 & 0.10 & & & \\ & & & 3.00 & 0.10 & & 4 \cdot 10^{-4} \\ & & & & 2.01 & 0.09 & \\ & & & & & 1.10 & \\ & & & & & & \end{bmatrix}$$

Each Givens rotation yields

$$x := x \cdot \frac{e_{i-1}}{q_i + x} \leq x \cdot \frac{e_{i-1}}{q_i}.$$

Therefore, x at the top is

$$x \leq e_{n-1} \prod_{i=n-k+2}^{n-1} \frac{e_{i-1}}{q_i}$$

$\Rightarrow$ x can be negligible even if no e_i is too small!

Aggdef(2) example

$$\begin{bmatrix} 6.00 & 0.10 & & & & & \\ & 5.00 & 0.10 & & & & \\ & & 4.00 & {\color{red}0.10} & & & \\ & & & {\color{red}3.00} & 0.10 & & {\color{red}4 \cdot 10^{-4}} \\ & & & & 2.01 & 0.09 & \\ & & & & & 1.10 & \\ & & & & & & \end{bmatrix}$$

Each Givens rotation yields

$$x := x \cdot \frac{e_{i-1}}{q_i + x} \leq x \cdot \frac{e_{i-1}}{q_i}.$$

Therefore, x at the top is

$$x \leq e_{n-1} \prod_{i=n-k+2}^{n-1} \frac{e_{i-1}}{q_i}$$

$\Rightarrow$ x can be negligible even if no e_i is too small!

Aggdef(2) example

$$\begin{bmatrix} 6.00 & 0.10 & & & & & \\ & 5.00 & 0.10 & & & & \\ & & 4.00 & 0.10 & & & 1\cdot 10^{-5} \\ & & & 3.00 & 0.10 & & \\ & & & & 2.01 & 0.09 & \\ & & & & & 1.10 & \\ & & & & & & \end{bmatrix}$$

Each Givens rotation yields

$$x := x \cdot \frac{e_{i-1}}{q_i + x} \leq x \cdot \frac{e_{i-1}}{q_i}.$$

Therefore, x at the top is

$$x \leq e_{n-1} \prod_{i=n-k+2}^{n-1} \frac{e_{i-1}}{q_i}$$

$\Rightarrow$ x can be negligible even if no e_i is too small!

Aggdef(2) example

$$\begin{bmatrix} 6.00 & 0.10 & & & & & \\ & 5.00 & 0.10 & & & & \\ & & 4.00 & 0.10 & & & 1 \cdot 10^{-5} \\ & & & 3.00 & 0.10 & & \\ & & & & 2.01 & 0.09 & \\ & & & & & 1.10 & \\ & & & & & & \end{bmatrix}$$

Each Givens rotation yields

$$x := x \cdot \frac{e_{i-1}}{q_i + x} \leq x \cdot \frac{e_{i-1}}{q_i}.$$

Therefore, x at the top is

$$x \leq e_{n-1} \prod_{i=n-k+2}^{n-1} \frac{e_{i-1}}{q_i}$$

$\Rightarrow$ x can be negligible even if no e_i is too small!

Aggdef(2) example

$$
\begin{bmatrix}
6.00 & 0.10 & & & & & \\
 & 5.00 & \color{blue}{0.10} & & & & \color{blue}{3 \cdot 10^{-7}} \\
 & & \color{blue}{4.00} & 0.10 & & & \\
 & & & 3.00 & 0.10 & & \\
 & & & & 2.01 & 0.09 & \\
 & & & & & 1.10 & \\
 & & & & & &
\end{bmatrix}
$$

Each Givens rotation yields

$$x := x \cdot \frac{e_{i-1}}{q_i + x} \leq x \cdot \frac{e_{i-1}}{q_i}.$$

Therefore, x at the top is

$$x \leq e_{n-1} \prod_{i=n-k+2}^{n-1} \frac{e_{i-1}}{q_i}$$

$\Rightarrow$ x can be negligible even if no e_i is too small!

Aggdef(2) example

$$\begin{bmatrix} 6.00 & 0.10 & & & & & \\ & 5.00 & 0.10 & & & & 3\cdot 10^{-7} \\ & & 4.00 & 0.10 & & & \\ & & & 3.00 & 0.10 & & \\ & & & & 2.01 & 0.09 & \\ & & & & & 1.10 & \\ & & & & & & \end{bmatrix}$$

Each Givens rotation yields

$$x := x \cdot \frac{e_{i-1}}{q_i + x} \leq x \cdot \frac{e_{i-1}}{q_i}.$$

Therefore, x at the top is

$$x \leq e_{n-1} \prod_{i=n-k+2}^{n-1} \frac{e_{i-1}}{q_i}$$

$\Rightarrow$ x can be negligible even if no e_i is too small!

Aggdef(2) example

$$\begin{bmatrix} 6.00 & 0.10 & & & & & 7 \cdot 10^{-9} \\ & 5.00 & 0.10 & & & & \\ & & 4.00 & 0.10 & & & \\ & & & 3.00 & 0.10 & & \\ & & & & 2.01 & 0.09 & \\ & & & & & 1.10 & \\ & & & & & & \end{bmatrix}$$

Each Givens rotation yields

$$x := x \cdot \frac{e_{i-1}}{q_i + x} \leq x \cdot \frac{e_{i-1}}{q_i}.$$

Therefore, x at the top is

$$x \leq e_{n-1} \prod_{i=n-k+2}^{n-1} \frac{e_{i-1}}{q_i}$$

$\Rightarrow$ x can be negligible even if no e_i is too small!

Aggdef(2) example

$$\begin{bmatrix} 6.00 & 0.10 & & & & & 7 \cdot 10^{-9} \\ & 5.00 & 0.10 & & & & \\ & & 4.00 & 0.10 & & & \\ & & & 3.00 & 0.10 & & \\ & & & & 2.01 & 0.09 & \\ & & & & & 1.10 & \\ & & & & & & \end{bmatrix}$$

Each Givens rotation yields

$$x := x \cdot \frac{e_{i-1}}{q_i + x} \leq x \cdot \frac{e_{i-1}}{q_i}.$$

Therefore, x at the top is

$$x \leq e_{n-1} \prod_{i=n-k+2}^{n-1} \frac{e_{i-1}}{q_i} \simeq 8 \cdot 10^{-9}.$$

$\Rightarrow$ x can be negligible even if no e_i is too small!

Convergence factor of x with a dqds iteration

$$
\begin{bmatrix} * & * & & & x \\ & * & * & & \\ & & * & * & \\ & & & * & \\ & & & & \end{bmatrix}
\xrightarrow{\text{dqds}}
\begin{bmatrix} \widehat{*} & \widehat{*} & & & \widehat{x} \\ & \widehat{*} & \widehat{*} & & \\ & & \widehat{*} & \widehat{*} & \\ & & & \widehat{*} & \\ & & & & \end{bmatrix}
$$

- $x \simeq e_{n-1} \prod_{i=n-k+2}^{n-1} \frac{e_{i-1}}{q_i}$
- One dqds iteration results in $\widehat{q_i} \simeq q_i, \ \widehat{e_i} \simeq \frac{\sigma_{i+1}^2}{\sigma_i^2} e_i$.

Convergence factor of x with a dqds iteration

$$\begin{bmatrix} * & * & & & x \\ & * & * & & \\ & & * & * & \\ & & & * & \\ & & & & \end{bmatrix} \xrightarrow{\text{dqds}} \begin{bmatrix} \widehat{*} & \widehat{*} & & & \widehat{x} \\ & \widehat{*} & \widehat{*} & & \\ & & \widehat{*} & \widehat{*} & \\ & & & \widehat{*} & \\ & & & & \end{bmatrix}$$

- $x \simeq e_{n-1} \displaystyle\prod_{i=n-k+2}^{n-1} \frac{e_{i-1}}{q_i}$
- One dqds iteration results in $\widehat{q_i} \simeq q_i, \ \widehat{e_i} \simeq \dfrac{\sigma_{i+1}^2}{\sigma_i^2} e_i$.

Hence,

$$\widehat{x} \simeq \widehat{e}_{n-1} \prod_{i=n-k+2}^{n-1} \frac{\widehat{e}_{i-1}}{\widehat{q}_i} = \frac{\sigma_n^2}{\sigma_{n-k+1}^2} x.$$

convergence factor

Convergence factor of x: example

$$\begin{bmatrix} * & * & & & 7\cdot 10^{-9} \\ & * & * & & \\ & & * & * & \\ & & & * & \\ & & & & \end{bmatrix} \xrightarrow{\text{dqds}} \begin{bmatrix} \widehat{*} & \widehat{*} & & & \\ & \widehat{*} & \widehat{*} & & \\ & & \widehat{*} & \widehat{*} & \\ & & & \widehat{*} & \\ & & & & \end{bmatrix}$$

- $x \simeq e_{n-1} \prod_{i=n-k+2}^{n-2} \frac{e_i}{q_i}$
- One dqds iteration results in $\widehat{q}_i \simeq q_i,\ \widehat{e}_i \simeq \frac{\sigma_{i+1}^2}{\sigma_i^2} e_i.$

Hence,

$$\widehat{x} \simeq \widehat{e}_{n-1} \prod_{i=n-k+2}^{n-1} \frac{\widehat{e}_{i-1}}{\widehat{q}_i} = \frac{\sigma_n^2}{\sigma_{n-k+1}^2}\, x.$$

convergence factor

Convergence factor of x: example

$$\begin{bmatrix} * & * & & & 7 \cdot 10^{-9} \\ & * & * & & \\ & & * & * & \\ & & & * & \\ & & & & \end{bmatrix} \xrightarrow{\text{dqds}} \begin{bmatrix} \widehat{*} & \widehat{*} & & & 9 \cdot 10^{-10} \\ & \widehat{*} & \widehat{*} & & \\ & & \widehat{*} & \widehat{*} & \\ & & & \widehat{*} & \\ & & & & \end{bmatrix}$$

- $x \simeq e_{n-1} \prod_{i=n-k+2}^{n-2} \frac{e_i}{q_i}$
- One dqds iteration results in $\widehat{q_i} \simeq q_i, \ \widehat{e_i} \simeq \frac{\sigma_{i+1}^2}{\sigma_i^2} e_i$.

Hence,

$$\widehat{x} \simeq \widehat{e}_{n-1} \prod_{i=n-k+2}^{n-1} \frac{\widehat{e}_{i-1}}{\widehat{q}_i} = \frac{\sigma_n^2}{\sigma_{n-k+1}^2} \, x.$$

convergence factor

Conventional deflation vs. Aggressive deflation

Conventional

$$\begin{bmatrix} * & * & & & & & \\ & \ddots & \ddots & & & & \\ & & * & {\color{blue}*_4} & & & \\ & & & * & {\color{blue}*_3} & & \\ & & & & * & {\color{blue}*_2} & \\ & & & & & * & {\color{blue}*_1} \\ & & & & & & * \end{bmatrix}$$

- looks for negligible values in $*_i$: “local” view
- $*_i = e_{n-i}$
- convergence factor of $*_i$:

$$\frac{\widehat{*_i}}{*_i} \simeq \frac{\sigma^2_{n-i+1}}{\sigma^2_{n-i}}$$

Aggressive

$$\begin{bmatrix} \ddots & \ddots & & & & & \\ & * & * & & & & \\ & & * & {\color{red}*_4} & {\color{red}*_3} & {\color{red}*_2} & {\color{red}*_1} \\ & & & * & & & \\ & & & & * & & \\ & & & & & * & \\ & & & & & & * \end{bmatrix}$$

- looks for negligible values in $*_i$: “global” view
- $*_i \simeq e_{n-i} \prod_{j=n-k+2}^{n-i} \frac{e_j}{q_j}$
- convergence factor of $*_i$:

$$\frac{\widehat{*_i}}{*_i} \simeq \frac{\sigma^2_{n-i+1}}{\sigma^2_{n-k+1}}$$

$\widehat{*_i}$: $*_i$ after one dqd(s) iteration, k: window size ($k = 4$ above)

Conventional deflation vs. Aggressive deflation

Conventional

$$\begin{bmatrix} * & * & & & & & \\ & \ddots & \ddots & & & & \\ & & * & *_4 & & & \\ & & & * & *_3 & & \\ & & & & * & *_2 & \\ & & & & & * & *_1 \\ & & & & & & * \end{bmatrix}$$

- looks for negligible values in $*_i$: “local” view
- $*_i = e_{n-i}$
- convergence factor of $*_i$:

$$\frac{\widehat{*_i}}{*_i} \simeq \frac{\sigma^2_{n-i+1}}{\sigma^2_{n-i}} \quad \to \frac{\sigma^2_{n-i+1} - s}{\sigma^2_{n-i} - s}$$

Aggressive

$$\begin{bmatrix} \ddots & \ddots & & & & & \\ & * & * & & & & \\ & & * & *_4 & *_3 & *_2 & *_1 \\ & & & * & & & \\ & & & & * & & \\ & & & & & * & \\ & & & & & & * \end{bmatrix}$$

- looks for negligible values in $*_i$: “global” view
- $*_i \simeq e_{n-i} \prod_{j=n-k+2}^{n-i} \frac{e_j}{q_j}$
- convergence factor of $*_i$:

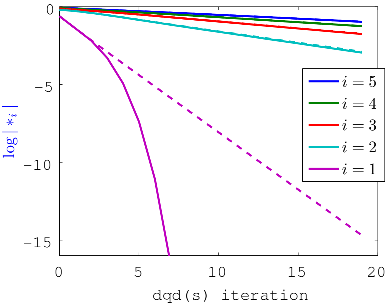
$$\frac{\widehat{*_i}}{*_i} \simeq \frac{\sigma^2_{n-i+1}}{\sigma^2_{n-k+1}} \quad \to \frac{\sigma^2_{n-i+1} - s}{\sigma^2_{n-k+1} - s}$$

$\widehat{*_i}$: $*_i$ after one dqd(s) iteration, k: window size ($k = 4$ above)

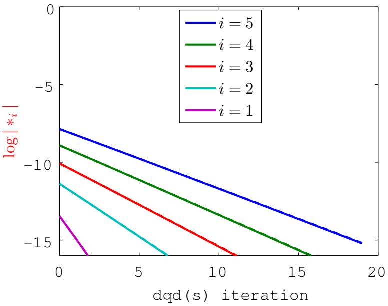
Convergence factors of $*_i$

	Conventional $*_i$	Aggressive $*_i$
$*_i$	e_{n-i}	$e_{n-i}\prod_{j=n-k+2}^{n-i}\frac{e_{j-1}}{q_j}$
$\frac{\widehat{*_i}}{*_i}$ with shift s	$\frac{\sigma^2_{n-i+1}-s}{\sigma^2_{n-i}-s}$	$\frac{\sigma^2_{n-i+1}-s}{\sigma^2_{n-k+1}-s}$

Conventional



Aggressive



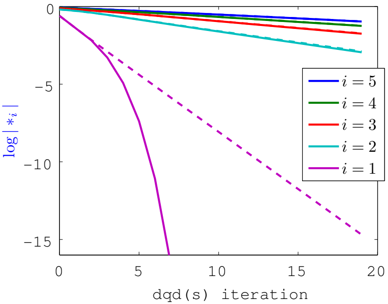
solid: dqds (with shift), dashed: dqd (zero-shift)

- aggressive deflation is much more powerful
- shift seems unnecessary with aggressive deflation

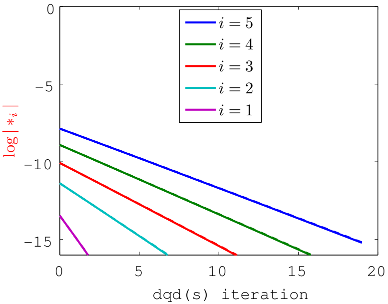
Convergence factors of $*_i$

	Conventional $*_i$	Aggressive $*_i$
$*_i$	e_{n-i}	$e_{n-i} \prod_{j=n-k+2}^{n-i} \frac{e_{j-1}}{q_j}$
$\frac{\widehat{*_i}}{*_i}$ with shift s	$\frac{\sigma_{n-i+1}^2 - s}{\sigma_{n-i}^2 - s}$	$\frac{\sigma_{n-i+1}^2 - s}{\sigma_{n-k+1}^2 - s}$

Conventional



Aggressive



solid: dqds (with shift), dashed: dqd (zero-shift)

- aggressive deflation is much more powerful
- shift seems unnecessary with aggressive deflation
  ⇒ **use dqd (zero-shift)?**

Zero-shift is attractive

Conventional

$$
\begin{bmatrix}
* & * & & & & & \\
& \ddots & \ddots & & & & \\
& & * & \color{blue}{*_4} & & & \\
& & & * & \color{blue}{*_3} & & \\
& & & & * & \color{blue}{*_2} & \\
& & & & & * & \color{blue}{*_1} \\
& & & & & & *
\end{bmatrix}
$$

Aggressive

$$
\begin{bmatrix}
\ddots & \ddots & & & & & \\
& * & * & & & & \\
& & * & \color{red}{*_4} & \color{red}{*_3} & \color{red}{*_2} & \color{red}{*_1} \\
& & & * & & & \\
& & & & * & & \\
& & & & & * & \\
& & & & & & *
\end{bmatrix}
$$

Purpose of shift: speed up “local” convergence of $*_1$

- conventional deflation – “local” view, needs shifts
- aggressive deflation – “global” view, no need for shifts

Benefits of dqd (zero-shift):

- no need to estimate shifts, simpler and cheaper algorithm
- parallel implementation possible (later)

pseudocodes

Inputs: bidiagonal B, Aggdef frequency f $(= 16$ in experiments)

Algorithm 2 dqds+agg: dqds with aggressive early deflation

while size$(B) > 100$ **do**
    run f dqds iterations
    call Aggdef
end while
run dqds to complete

Algorithm 3 dqd+agg: dqd with aggressive early deflation

while size$(B) > 100$ **do**
    run one dqds, then $f - 1$ dqd iterations
    call Aggdef
end while
run dqds to complete

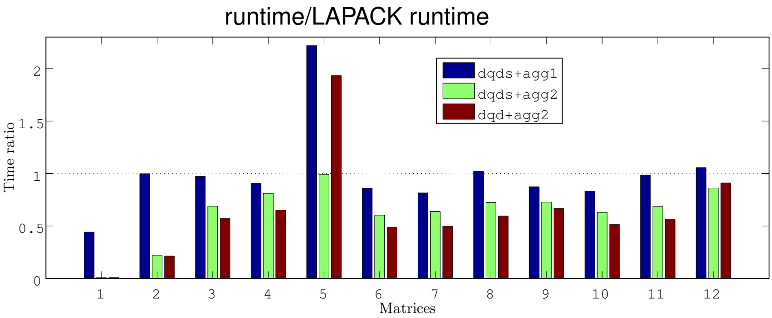
Numerical experiments: specifications

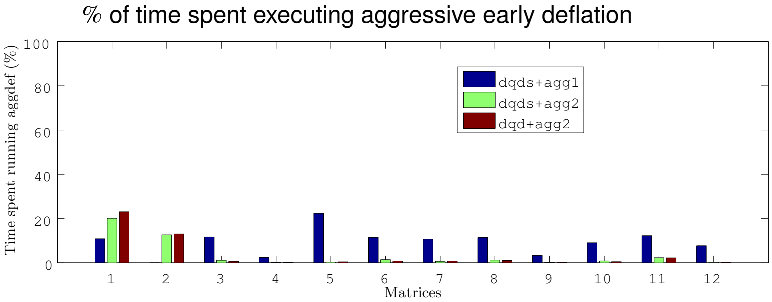
algorithm	deflation strategy	shift
LAPACK	conventional	$s > 0$
dqds+agg1	Aggdef(1)	$s > 0$
dqds+agg2	Aggdef(2)	$s > 0$
dqd+agg2	Aggdef(2)	zero-shift

environment: Intel Core i7 2.67GHz Processor (4 cores, 8 threads), 12GB RAM

	n	Test matrices B: diagonals $\sqrt{q_i}$, off-diagonals $\sqrt{e_i}$
1	30000	$\sqrt{q_i} = n + 1 - i, \quad \sqrt{e_i} = 1$
2	30000	$\sqrt{q_{i-1}} = \beta\sqrt{q_i}, \quad \sqrt{e_i} = \sqrt{q_i}, \beta = 1.01$
3	30000	Toeplitz: $\sqrt{q_i} = 1, \quad \sqrt{e_i} = 2$
4	30000	$\sqrt{q_{2i-1}} = n + 1 - i, \quad \sqrt{q_{2i}} = i, \quad \sqrt{e_i} = (n - i)/5$
5	30000	$\sqrt{q_{i+1}} = \beta\sqrt{q_i}\ (i \geq n/2), \quad \sqrt{q_{n/2}} = 1,$ $\sqrt{q_{i-1}} = \beta\sqrt{q_i}\ (i \leq n/2), \quad \sqrt{e_i} = 1, \beta = 1.01$
6	30000	Cholesky factor of tridiagonal $(1, 2, 1)$ matrix
7	30000	Cholesky factor of Laguerre matrix
8	30000	Cholesky factor of Hermite recurrence matrix
9	30000	Cholesky factor of Wilkinson matrix
10	30000	Cholesky factor of Clement matrix
11	13786	matrix from electronic structure calculations
12	16023	matrix from electronic structure calculations

Numerical experiments





Parallel implementation

Parallel implementation

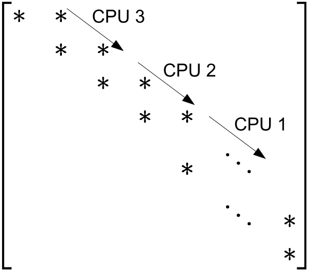
- dqds + conventional

  $\begin{array}{c}\text{get}\\\text{shift s}\end{array} \rightarrow \text{dqds} \rightarrow \begin{array}{c}\text{get}\\\text{shift s}\end{array} \rightarrow \text{dqds} \rightarrow \cdots$

- dqd + aggressive

  $\begin{array}{c}\text{get}\\\text{shift s}\end{array} \rightarrow \text{dqds} \rightarrow \text{dqd} \rightarrow \text{dqd} \rightarrow \cdots \rightarrow \begin{array}{c}\text{get}\\\text{shift s}\end{array} \rightarrow$

Parallel implementation



- dqds + conventional

  get shift s → dqds → get shift s → dqds → ···

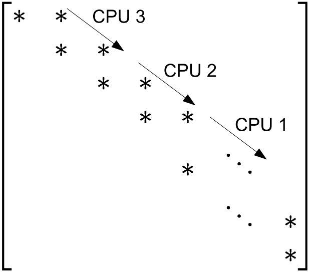
- dqd + aggressive

  get shift s → dqds → dqd → dqd → ··· → get shift s →

- parallel dqd + aggressive

  get shift s → (dqds, dqd, dqd, ⋮) → get shift s → (dqds, dqd, dqd, ⋮) → ···

Parallel implementation



- dqds + conventional

  get shift s → dqds → get shift s → dqds → ⋯

- dqd + aggressive

  get shift s → dqds → dqd → dqd → ⋯ → get shift s →

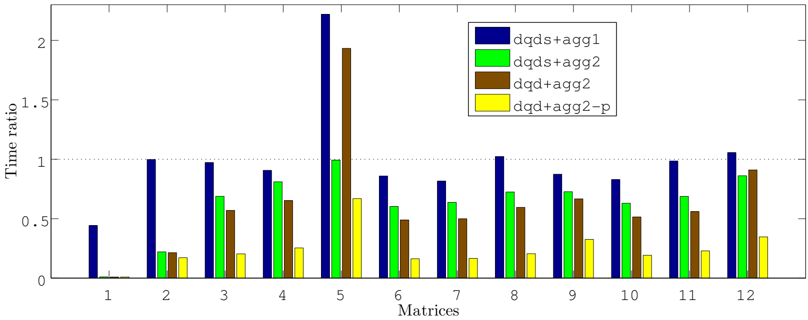
- parallel dqd + aggressive

  get shift s → (dqds, dqd, dqd, ⋮) → get shift s → (dqds, dqd, dqd, ⋮) → ⋯

- Impossible with conventional deflation
  - effective shifts available only after previous dqds is completed
  - with zero-shift, convergence of $*_1$ is extremely slow
- Possible with aggressive deflation + zero shifting
  - shift $s = 0$ is predetermined
  - dqd+agg2 has competitive speed even with sequential run

Numerical experiments: parallel dqd+agg2

OpenMP implementation with 4 CPUs, runtime ratio over LAPACK



- Parallel dqd+agg2 is always the fastest.

Summary and Future work

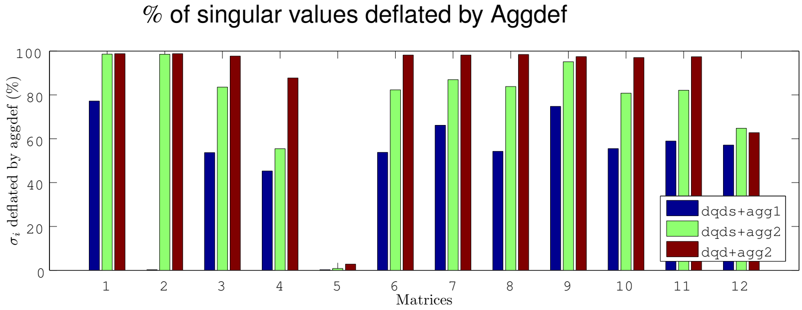
Summary

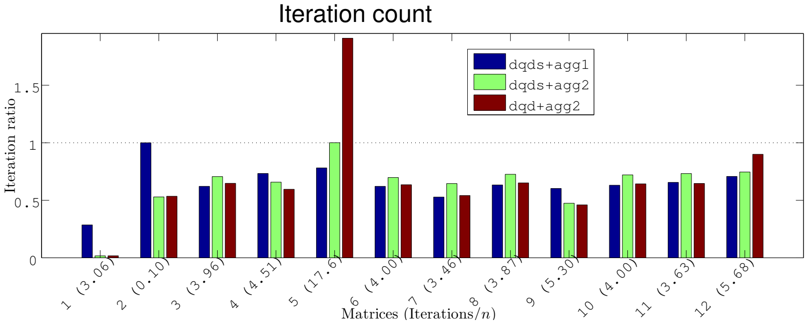
- Combined dqds and aggressive early deflation.
  - “direct” version Aggdef(1) and “efficient” version Aggdef(2).
  - Aggdef(2) is always faster than LAPACK routine, up to $\times 50$.
- Zero-shift becomes viable and attractive
  - fast with a sequential execution.
  - parallel execution is possible.

Future work

- Optimize/implement parallel dqd+Aggdef(2).
  - better shift strategy?
- Release sequential/parallel code as LAPACK routine.

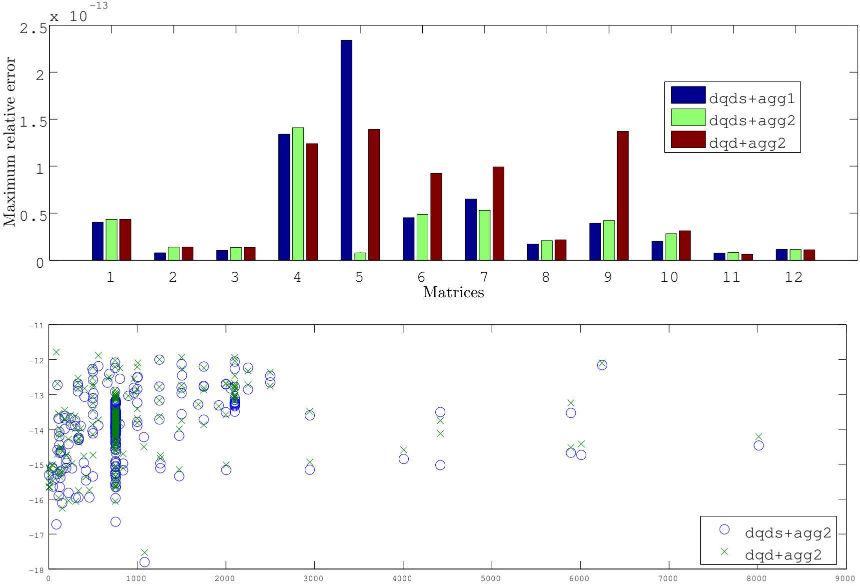
Numerical experiments: more data



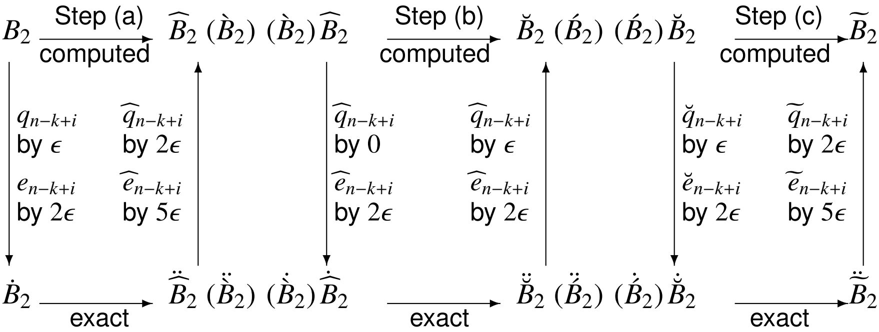


Numerical experiments: relative accuracy

Maximum relative accuracy $\max_i |\sigma_i - \widehat{\sigma}_i|/\sigma_i$



Aggdef(2) preserves high relative accuracy



By a mixed forward-backward relative error analysis, we establish

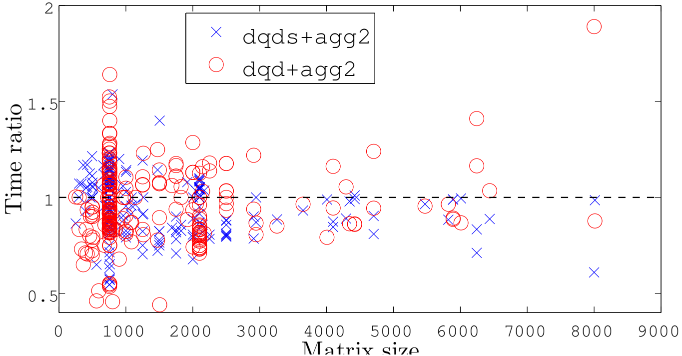
Theorem

$$1 - (7n + 19\sqrt{n} + 2)\epsilon \le \frac{\sigma_i(\widetilde{B})}{\sigma_i(B)} \le 1 + (7n + 19\sqrt{n} + 2)\epsilon$$

for $i = 1, \ldots, n$.

More experiments

500 test matrices from [Marques Voemel, Demmel and Parlett, 2008]



runtime ratio over LAPACK routine DLASQ

- most matrices are too small for Aggdef to make a difference

References

- Z. Bai and J. Demmel. On a block implementation of Hessenberg multishift QR iteration. Int. J. High Speed Comput., (1989)
- K. Braman, R. Byers, and R. Mathias. The multishift QR algorithm. II. Aggressive early deflation. SIAM J. Matrix Anal. Appl., (2002)
- K. V. Fernando and B. N. Parlett. Accurate singular values and differential qd algorithms. Numer. Math., (1994)
- B. N. Parlett and O. A. Marques. An implementation of the dqds algorithm (positive case). Linear Algebra Appl., (2000)

Aggdef(1) and Aggdef(2) are equivalent

Aggdef(1)

$$\begin{bmatrix} \ddots & \ddots & & & & & \\ & * & * & & & & \\ & & * & * & * & * & w \\ & & & * & & & \\ & & & & * & & \\ & & & & & * & \\ & & & & & & * \end{bmatrix}$$

- ignores w when

$$w = v_1 e_{n-k} \leq S\epsilon$$

Aggdef(2)

$$\begin{bmatrix} * & * & & & & & x \\ & * & * & & & & \\ & & * & * & & & \\ & & & * & * & & \\ & & & & * & * & \\ & & & & & * & \\ & & & & & & \end{bmatrix}$$

- ignores x when

$$x = q_{n-k+1} v_1 \leq S\epsilon$$

$(v_1 \; v_2 \; \ldots v_n)$: right singular vector of $\sigma_{\min}(B)$

Aggdef(1) and Aggdef(2) are equivalent

Aggdef(1)

$$\begin{bmatrix} \ddots & \ddots & & & & & \\ & * & * & & & & \\ & & * & * & * & * & w \\ & & & * & & & \\ & & & & * & & \\ & & & & & * & \\ & & & & & & * \end{bmatrix}$$

- ignores w when

$$w = v_1 e_{n-k} \leq S\epsilon$$

Aggdef(2)

$$\begin{bmatrix} * & * & & & & & x \\ & * & * & & & & \\ & & * & * & & & \\ & & & * & * & & \\ & & & & * & * & \\ & & & & & * & \\ & & & & & & \end{bmatrix}$$

- ignores x when

$$x = q_{n-k+1} v_1 \leq S\epsilon$$

$(v_1 \ v_2 \ \ldots v_n)$: right singular vector of $\sigma_{\min}(B)$

$$\Downarrow$$

Aggdef(1) and Aggdef(2) are equivalent, modulo a constant difference in neglecting criteria

Aggressive deflation for dqds -version 1: Aggdef(1)

$$\begin{bmatrix} \ddots & \ddots & & & & & \\ & * & * & & & & \\ & & * & \color{red}{*} & \color{red}{*} & \color{red}{*} & \color{red}{*} \\ & & & * & & & \\ & & & & * & & \\ & & & & & * & \\ & & & & & & * \end{bmatrix}$$

1. Find negligible elements among $\color{red}{*}$, remove corresponding rows and columns
2. Reduce “V-matrix” to bidiagonal form, resume dqds

Aggressive deflation for dqds -version 1: Aggdef(1)

$$
\begin{bmatrix}
\ddots & \ddots & & & & & \\
 & * & * & & & & \\
 & & * & \color{red}{*} & \color{red}{*} & \color{red}{*} & \color{red}{*} \\
 & & & * & & & \\
 & & & & * & & \\
 & & & & & * & \\
 & & & & & & *
\end{bmatrix}
$$

1. Find negligible elements among $\color{red}{*}$, remove corresponding rows and columns
2. Reduce "V-matrix" to bidiagonal form, resume dqds

$$
\begin{bmatrix} * & * & * & * \\ & * & & \\ & & * & \\ & & & * \end{bmatrix}
\overset{G_R(3,4)}{\rightarrow}
\begin{bmatrix} * & * & * & 0 \\ & * & & \\ & & * & + \\ & & + & * \end{bmatrix}
\overset{G_L(3,4)}{\rightarrow}
\begin{bmatrix} * & * & * & \\ & * & & \\ & & * & * \\ & & 0 & * \end{bmatrix}
\overset{G_R(2,3)}{\rightarrow}
\begin{bmatrix} * & * & 0 & \\ & * & + & \\ & + & * & * \\ & & & * \end{bmatrix}
$$

$$
\overset{G_L(2,3)}{\rightarrow}
\begin{bmatrix} * & * & & \\ & * & * & + \\ & 0 & * & * \\ & & & * \end{bmatrix}
\overset{G_R(3,4)}{\rightarrow}
\begin{bmatrix} * & * & & \\ & * & * & 0 \\ & & * & * \\ & & + & * \end{bmatrix}
\overset{G_L(3,4)}{\rightarrow}
\begin{bmatrix} * & * & & \\ & * & * & \\ & & * & * \\ & & 0 & * \end{bmatrix}
$$

Choice of parameters

- Window size $k = \min\{\sqrt{n}, p\}$,
  $p = \text{argmax}\{i \mid q_j > e_j \text{ for all } j \geq n - i\}$

  – Let the working matrix tell us a good choice
- Aggdef frequency $f = 16$
  – Rerun Aggdef when more than 3 singular values are deflated

Absolute accuracy and relative accuracy

	$\frac{\lvert\sigma_1(B) - \widehat{\sigma}_1(B)\rvert}{\sigma_1(B)}$	$\frac{\lvert\sigma_{1000}(B) - \widehat{\sigma}_{1000}(B)\rvert}{\sigma_{1000}(B)}$
QR	10^{-15}	10^{-1}
refined QR	10^{-15}	10^{-14}
dqds	10^{-15}	10^{-15}

Typical relative accuracy for $B : \mathbb{R}^{1000\times1000},\ \|B\|_2 = 1,\ \sigma_{1000}(B) = 10^{-14}$

- dqds computes all σ_i to high relative accuracy
  - smallest singular values are often important (distance to a singular matrix, null space, ...)

Aggdef(2): mathematical description

$$\widetilde{B}^T \widetilde{B} = \begin{bmatrix} I_{n-k+1} & \\ & Q^T \end{bmatrix} B^T B \begin{bmatrix} I_{n-k+1} & \\ & Q \end{bmatrix} + \begin{bmatrix} & \\ & E \end{bmatrix},$$

where Q is a product of Givens rotations, and

$$E = \begin{bmatrix} & & & \\ & & & -\sqrt{xq_{n-k+1}} \\ & & & -\sqrt{xe_{n-k+1}} \\ & & & \\ -\sqrt{xq_{n-k+1}} & -\sqrt{xe_{n-k+1}} & & x \end{bmatrix}.$$

Aggdef(2): mathematical description

$$\widetilde{B}^T \widetilde{B} = \begin{bmatrix} I_{n-k+1} & \\ & Q^T \end{bmatrix} B^T B \begin{bmatrix} I_{n-k+1} & \\ & Q \end{bmatrix} + \begin{bmatrix} & \\ & E \end{bmatrix},$$

where Q is a product of Givens rotations, and

$$E = \begin{bmatrix} & & -\sqrt{xq_{n-k+1}} \\ & & -\sqrt{xe_{n-k+1}} \\ & & \\ -\sqrt{xq_{n-k+1}} & -\sqrt{xe_{n-k+1}} & x \end{bmatrix}.$$

- neglect x when $\|E\|_2 < S\,\epsilon$
  $\Rightarrow$ maintain high relative accuracy of singular values

dstqds for computing $\widehat{B_2}$ s.t. $\widehat{B_2}^T\widehat{B_2} = B_2^T B_2 - sI$

Algorithm 4 dstqds: differential stationary qds

$d = -s$

$\widehat{q}_{n-k+1} = q_{n-k+1} + d$

for $i := n-k+1, \cdots, n-1$ **do**

$\quad \widehat{e_i} = q_i e_i / \widehat{q_i}$

$\quad d = d e_i / \widehat{q_i} - s$

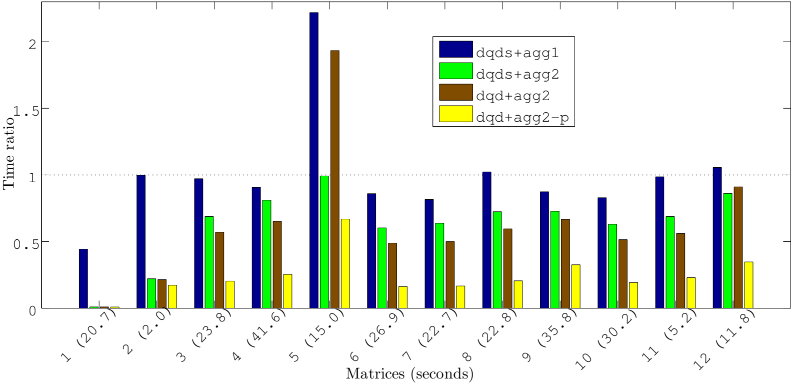
$\quad \widehat{q}_{i+1} = q_{i+1} + d$

end for

- Cost is $O(k)$ flops.
- Backward-forward stable in the relative sense [Dhillon and Parlett (2004)].

Numerical experiments: pipelined dqd+agg2

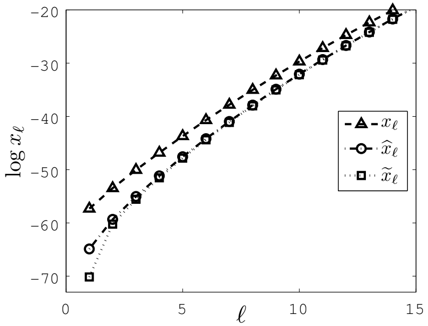
OpenMP implementation with 4 CPUs, runtime ratio over LAPACK



	zero-shift	parallel
conventional	×	×
aggressive	√	√

dqds-dqd comparison: Aggdef(2) chased-up element

$$B = \text{bidiag}\begin{pmatrix} & \sqrt{0.1} & . & . & \sqrt{0.1} & & \sqrt{0.1} & \\ \sqrt{1000} & & . & . & . & \sqrt{2} & & \sqrt{1} \end{pmatrix}$$



ℓ-$\log x_\ell$ plots. $\widehat{x}_\ell$ and $\widetilde{x}_\ell$ are obtained after running 5 dqd and dqds iterations

- dqds and dqd perform the same, except for the smallest singular value $\sigma_{\min}$
- $\sigma_{\min}$ is deflated anyway $\Rightarrow$ shift is not needed

Convergence factors and effect of shift

Conventional

Aggressive

- Convergence factors with one dqd iteration

$$\frac{\widehat{*_i}}{*_i} \rightarrow \frac{\sigma_i}{\sigma_{i+1}}$$

$$\frac{\widehat{*_i}}{*_i} \rightarrow \frac{\sigma_i}{\sigma_{k+1}}$$

Convergence factors and effect of shift

Conventional | Aggressive

- Convergence factors with one dqd iteration

$$\frac{\widehat{*_i}}{*_i} \to \frac{\sigma_i}{\sigma_{i+1}} \qquad\qquad \frac{\widehat{*_i}}{*_i} \to \frac{\sigma_i}{\sigma_{k+1}}$$

- Convergence factors with one dqds iteration (introduce shift s)

Convergence factors and effect of shift

Conventional	Aggressive

- Convergence factors with one dqd iteration

$\frac{\widehat{*_i}}{*_i} \rightarrow \frac{\sigma_i}{\sigma_{i+1}}$	$\frac{\widehat{*_i}}{*_i} \rightarrow \frac{\sigma_i}{\sigma_{k+1}}$

- Convergence factors with one dqds iteration (introduce shift s)

$\frac{\widehat{*_i}}{*_i} \rightarrow \frac{\sigma_i^2 - s}{\sigma_{i+1}^2 - s}$	$\frac{\widehat{*_i}}{*_i} \rightarrow \frac{\sigma_i^2 - s}{\sigma_{k+1}^2 - s}$