Implicit Jacobi Algorithm for the SVD with High Relative Accuracy

Johan A. Ceballos, Froilán M. Dopico and Juan Manuel Molera

Universidad
Carlos III de Madrid

IWASEP 8 Berlin, June 28-July 1

Outline

Outline

Objectives of this work

- Demmel, Gu, Eisenstat, Slapničar, Vesélic and Drmač [1999] presented
  1. Two algorithms that compute SVD with high relative accuracy (hra).

Objectives of this work

- Demmel, Gu, Eisenstat, Slapničar, Vesélic and Drmač [1999] presented
  1. Two algorithms that compute SVD with high relative accuracy (hra).
  2. Both start from a Rank Revealing Decomposition (RRD) $A = XDY^T$.

Objectives of this work

- Demmel, Gu, Eisenstat, Slapničar, Vesélic and Drmač [1999] presented
  1. Two algorithms that compute SVD with high relative accuracy (hra).
  2. Both start from a Rank Revealing Decomposition (RRD) $A = XDY^T$.
  3. Alg. 3.1, QR-decomposition+one application of one-sided Jacobi. The error bound depends of the condition number (usually small) of an intermediate factor: $\kappa(R')$.

Objectives of this work

- Demmel, Gu, Eisenstat, Slapničar, Vesélic and Drmač [1999] presented
  1. Two algorithms that compute SVD with high relative accuracy (hra).
  2. Both start from a Rank Revealing Decomposition (RRD) $A = XDY^T$.
  3. Alg. 3.1, QR-decomposition+one application of one-sided Jacobi. The error bound depends of the condition number (usually small) of an intermediate factor: $\kappa(R')$.
  4. Alg. 3.2, Two applications of one-sided Jacobi. It does not have the previous factor.

Objectives of this work

- Demmel, Gu, Eisenstat, Slapničar, Vesélic and Drmač [1999] presented
  1. Two algorithms that compute SVD with high relative accuracy (hra).
  2. Both start from a Rank Revealing Decomposition (RRD) $A = XDY^T$.
  3. Alg. 3.1, QR-decomposition+one application of one-sided Jacobi. The error bound depends of the condition number (usually small) of an intermediate factor: $\kappa(R')$.
  4. Alg. 3.2, Two applications of one-sided Jacobi. It does not have the previous factor.
- The implicit Jacobi SVD algorithm gets hra, starting from an RRD, with error bounds that reflect the sensitivity of the problem, with only one run of Jacobi.

Objectives of this work

- This work is a natural extension of the implicit Jacobi method for the symmetric eigenvalue problem [Dopico, Koev, M, 2009]

Objectives of this work

- This work is a natural extension of the implicit Jacobi method for the symmetric eigenvalue problem [Dopico, Koev, M, 2009]
- With this algorithm we have now algorithms that useRRD to compute with hra

Objectives of this work

- This work is a natural extension of the implicit Jacobi method for the symmetric eigenvalue problem [Dopico, Koev, M, 2009]
- With this algorithm we have now algorithms that useRRD to compute with hra
  1. SVD [Demmel et al; Ceballos, Dopico, M]

Objectives of this work

- This work is a natural extension of the implicit Jacobi method for the symmetric eigenvalue problem [Dopico, Koev, M, 2009]
- With this algorithm we have now algorithms that useRRD to compute with hra
  1. SVD [Demmel et al; Ceballos, Dopico, M]
  2. Eigenvalues and Eigenvectors [Dopico, M, Moro; Dopico, Koev, M]

Objectives of this work

- This work is a natural extension of the implicit Jacobi method for the symmetric eigenvalue problem [Dopico, Koev, M, 2009]
- With this algorithm we have now algorithms that useRRD to compute with hra
  1. SVD [Demmel et al; Ceballos, Dopico, M]
  2. Eigenvalues and Eigenvectors [Dopico, M, Moro; Dopico, Koev, M]
  3. Solutions of Linear Systems (for almost any rhs) [Dopico, M]

Outline

Jacobi SVD Methods

One-sided Jacobi: Diagonalize implicitly $A^T A$

Run implicitly the eigenvalue Jacobi algorithm on the matrix $A^T A$.

Jacobi SVD Methods

One-sided Jacobi: Diagonalize implicitly $A^T A$

Run implicitly the eigenvalue Jacobi algorithm on the matrix $A^T A$.

`for each` $i > j$

Jacobi SVD Methods

One-sided Jacobi: Diagonalize implicitly $A^T A$

Run implicitly the eigenvalue Jacobi algorithm on the matrix $A^T A$.

`for each` $i > j$

- compute $J(i, j, c, s)$ such that the columns i and j of AJ are orthogonal

Jacobi SVD Methods

One-sided Jacobi: Diagonalize implicitly $A^T A$

Run implicitly the eigenvalue Jacobi algorithm on the matrix $A^T A$.

`for each` $i > j$

- compute $J(i, j, c, s)$ such that the columns i and j of AJ are orthogonal

`repeat` cyclicly until convergence

Jacobi SVD Methods

One-sided Jacobi: Diagonalize implicitly $A^T A$

Run implicitly the eigenvalue Jacobi algorithm on the matrix $A^T A$.

`for each` $i > j$

- compute $J(i, j, c, s)$ such that the columns i and j of AJ are orthogonal

`repeat` cyclicly until convergence

- given convergence, the columns of $AJ_1 J_2 \cdots J_N$ are orthogonal, so

Jacobi SVD Methods

One-sided Jacobi: Diagonalize implicitly $A^T A$

Run implicitly the eigenvalue Jacobi algorithm on the matrix $A^T A$.

`for each` $i > j$

- compute $J(i, j, c, s)$ such that the columns i and j of AJ are orthogonal

`repeat` cyclicly until convergence

- given convergence, the columns of $AJ_1 J_2 \cdots J_N$ are orthogonal, so
- $AV = U\Sigma$ with $V = J_1 J_2 \cdots J_N$ and $U\Sigma$ is the final matrix with orthogonal columns.

Jacobi SVD Methods

Symmetrize A + Classical Jacobi

Jacobi SVD Methods

Symmetrize A + Classical Jacobi

`for each` $i > j$

Jacobi SVD Methods

Symmetrize A + Classical Jacobi

`for each` $i > j$

- compute $J_1(i, j, c_1, s_1)$ such that the 2×2 submatrix $(J_1 A)_{([i,j],[i,j])}$ is symmetric.

Jacobi SVD Methods

Symmetrize A + Classical Jacobi

`for each` $i > j$

- compute $J_1(i, j, c_1, s_1)$ such that the 2×2 submatrix $(J_1 A)_{([i,j],[i,j])}$ is symmetric.
- compute $J_2(i, j, c_2, s_2)$ such that the 2×2 submatrix $(J_2^T (J_1 A) J_2)_{([i,j],[i,j])}$ is diagonal.

Jacobi SVD Methods

Symmetrize A + Classical Jacobi

`for each` $i > j$

- compute $J_1(i, j, c_1, s_1)$ such that the 2×2 submatrix $(J_1 A)_{([i,j],[i,j])}$ is symmetric.
- compute $J_2(i, j, c_2, s_2)$ such that the 2×2 submatrix $(J_2^T (J_1 A) J_2)_{([i,j],[i,j])}$ is diagonal.

`repeat` cyclicly until convergence

Jacobi SVD Methods

Kogbetliantz's Method

Compute the SVD of a triangular matrix.

Jacobi SVD Methods

Kogbetliantz's Method

Compute the SVD of a triangular matrix.

`for each` $i > j$

Jacobi SVD Methods

Kogbetliantz's Method

Compute the SVD of a triangular matrix.

`for each` $i > j$

- Compute $J_1(i, j, c_1, s_1)$ and $J_2(i, j, c_2, s_2)$ such that the 2×2 submatrix $(J_1 A J_2)_{([i,j],[i,j])}$ is diagonal.

Jacobi SVD Methods

Kogbetliantz's Method

Compute the SVD of a triangular matrix.

`for each` $i > j$

- Compute $J_1(i, j, c_1, s_1)$ and $J_2(i, j, c_2, s_2)$ such that the 2×2 submatrix $(J_1 A J_2)_{([i,j],[i,j])}$ is diagonal.

`repeat` cyclicly until convergence

HRA is not obtained from standard algorithms

Example: 100×100 random Cauchy matrix A

$$a_{ij} = \frac{1}{x_i + y_j}, \quad \text{with random } x, y$$

HRA is not obtained from standard algorithms

Example: 100×100 random Cauchy matrix A

$$a_{ij} = \frac{1}{x_i + y_j}, \quad \text{with random } x, y$$

- $\kappa(A) = 6.24 \cdot 10^{17}$

HRA is not obtained from standard algorithms

Example: 100×100 random Cauchy matrix A

$$a_{ij} = \frac{1}{x_i + y_j}, \quad \text{with random } x, y$$

- $\kappa(A) = 6.24 \cdot 10^{17}$
- Errors in MATLAB's `svd` function: double precision, $u \approx 10^{-16}$, compared to "exact" 200-decimal digits,

$$\max_i \frac{|\widehat{\sigma}_i - \sigma_i|}{|\sigma_i|} = 102.97 \quad \text{and} \quad \max_i \|\widehat{v}_i - v_i\|_2 = 1.15$$

HRA is not obtained from standard algorithms

Example: 100×100 random Cauchy matrix A

$$a_{ij} = \frac{1}{x_i + y_j}, \quad \text{with random } x, y$$

- $\kappa(A) = 6.24 \cdot 10^{17}$
- Errors in MATLAB's `svd` function: double precision, $u \approx 10^{-16}$, compared to "exact" 200-decimal digits,

$$\max_i \frac{|\widehat{\sigma}_i - \sigma_i|}{|\sigma_i|} = 102.97 \quad \text{and} \quad \max_i \|\widehat{v}_i - v_i\|_2 = 1.15$$

- Errors in accurate algorithm (Factorization + Imp. SVD Jacobi)

$$\max_i \frac{|\widehat{\sigma}_i - \sigma_i|}{|\sigma_i|} = 2.9 \cdot 10^{-14} \quad \text{and} \quad \max_i \|\widehat{v}_i - v_i\|_2 = 6.1 \cdot 10^{-13}$$

Outline

Implicit Jacobi SVD

Implicit Jacobi SVD

1. Input: Factors X, D and Y: $A = XDY^T \in \mathbb{R}^{m \times n}$

Implicit Jacobi SVD

1. `Input:` Factors X, D and Y: $A = XDY^T \in \mathbb{R}^{m \times n}$

Accurate Rank Revealing Decomposition

Implicit Jacobi SVD

1. `Input:` Factors X, D and Y: $A = XDY^T \in \mathbb{R}^{m \times n}$

Accurate Rank Revealing Decomposition

- Given $A \in \mathbb{R}^{m \times n}$, with $rank(A) = r$,

$$A = XDY^T,$$

with $X \in \mathbb{R}^{m \times r}$, $Y \in \mathbb{R}^{n \times r}$ and $D \in \mathbb{R}^{r \times r}$ diagonal invertible, it is called a Rank Revealing Decomposition (RRD) of A if

$$\kappa(X), \kappa(Y) \gtrsim 1 \quad \text{and} \quad \kappa(D) \approx \kappa(A).$$

Implicit Jacobi SVD

1. `Input:` Factors X, D and Y: $A = XDY^T \in \mathbb{R}^{m\times n}$

Accurate Rank Revealing Decomposition

- Given $A \in \mathbb{R}^{m\times n}$, with $rank(A) = r$,

$$A = XDY^T,$$

with $X \in \mathbb{R}^{m\times r}$, $Y \in \mathbb{R}^{n\times r}$ and $D \in \mathbb{R}^{r\times r}$ diagonal invertible, it is called a Rank Revealing Decomposition (RRD) of A if

$$\kappa(X), \kappa(Y) \gtrsim 1 \quad \text{and} \quad \kappa(D) \approx \kappa(A).$$

- We say that an RRD, $A = XDY^T$, is accurate if the computed factors (u, the unit roundoff) $\widehat{X}, \widehat{Y}$ and $\widehat{D}$ obey

$$\frac{\|\widehat{X} - X\|}{\|X\|} = O(u), \frac{\|\widehat{Y} - Y\|}{\|Y\|} = O(u) \quad \text{and} \quad \frac{|\widehat{D} - D|}{|D|} = O(u)$$

Implicit Jacobi SVD

1. Input: Factors X, D and Y: $A = XDY^T \in \mathbb{R}^{m \times n}$

Accurate RRDs are only possible for special types of matrices through structured implementations of Gaussian elimination with complete pivoting (**GECP**), or variations of GECP.

- (Scaled-)Cauchy, (Polynomial) Vandermonde (DFT + GECP). [Demmel, Koev]
- Diagonally Dominant M-Matrices. [Demmel and Koev, Peña]
- Polynomial Vandermonde. [Demmel and Koev]
- Well Scaled Positive Definite. [Demmel and Veselić]
- Acyclic Matrices (include bidiagonal). [Demmel and Gragg]
- Diagonally Dominant. [Qiang Ye, Dopico and Koev]
- DSTU. [Demmel]
- Graded matrices, Total signed compound, [Demmel et al.]
- Positive definite [Demmel, Veselić, Mathias]
- Symmetric and diagonally scaled Cauchy, Vandermonde [Dopico, Koev]
- Symmetric DSTU and TSC [Peláez, Moro]

Implicit Jacobi SVD

1. Input: Factors X, D and Y: $A = XDY^T \in \mathbb{R}^{m \times n}$

Implicit Jacobi SVD

1. `Input:` Factors X, D and Y: $A = XDY^T \in \mathbb{R}^{m \times n}$
2. Run QR decompositions: $A = Q_X \begin{bmatrix} X_0 D Y_0^T & 0 \\ 0 & 0 \end{bmatrix} Q_Y^T$

Implicit Jacobi SVD

1. `Input:` Factors X, D and Y: $A = XDY^T \in \mathbb{R}^{m \times n}$
2. Run QR decompositions: $A = Q_X \begin{bmatrix} X_0 D Y_0^T & 0 \\ 0 & 0 \end{bmatrix} Q_Y^T$

$$X = Q_X \begin{bmatrix} X_0 \\ 0 \end{bmatrix} \quad Y = Q_Y \begin{bmatrix} Y_0 \\ 0 \end{bmatrix}$$

$X_0, Y_0^T \in \mathbb{R}^{r \times r}$ upper triangular invertible matrices
$D \in \mathbb{R}^{r \times r}$ diagonal and invertible.

Implicit Jacobi SVD

1. `Input:` Factors X, D and Y: $A = XDY^T \in \mathbb{R}^{m \times n}$
2. Run QR decompositions: $A = Q_X \begin{bmatrix} X_0 D Y_0^T & 0 \\ 0 & 0 \end{bmatrix} Q_Y^T$
3. Run implicit Jacobi SVD Jacobi algorithm on $A_0 = X_0 D Y_0^T \in \mathbb{R}^{r \times r}$ to compute singular values and vectors.

Implicit Jacobi SVD

1. `Input:` Factors X, D and Y: $A = XDY^T \in \mathbb{R}^{m \times n}$
2. Run QR decompositions: $A = Q_X \begin{bmatrix} X_0 D Y_0^T & 0 \\ 0 & 0 \end{bmatrix} Q_Y^T$
3. Run implicit Jacobi SVD Jacobi algorithm on $A_0 = X_0 D Y_0^T \in \mathbb{R}^{r \times r}$ to compute singular values and vectors.

- Compute

$$A_{(i,j)} = \sum_{k=1}^{r} (X_0)_{(i,k)} D_{(k,k)} (Y_0)_{(j,k)}$$

- Apply the rotations only on

$$X_0 \rightarrow J X_0, \quad \text{and} \quad Y_0^T \rightarrow Y_0^T \bar{J}$$

Implicit Jacobi SVD

1. `Input:` Factors X, D and Y: $A = XDY^T \in \mathbb{R}^{m \times n}$
2. Run QR decompositions: $A = Q_X \begin{bmatrix} X_0 D Y_0^T & 0 \\ 0 & 0 \end{bmatrix} Q_Y^T$
3. Run implicit Jacobi SVD Jacobi algorithm on $A_0 = X_0 D Y_0^T \in \mathbb{R}^{r \times r}$ to compute singular values and vectors.
4. Until convergence: small off-diagonal elements

Implicit Jacobi SVD

1. `Input:` Factors X, D and Y: $A = XDY^T \in \mathbb{R}^{m \times n}$
2. Run QR decompositions: $A = Q_X \begin{bmatrix} X_0 D Y_0^T & 0 \\ 0 & 0 \end{bmatrix} Q_Y^T$
3. Run implicit Jacobi SVD Jacobi algorithm on $A_0 = X_0 D Y_0^T \in \mathbb{R}^{r \times r}$ to compute singular values and vectors.
4. Until convergence: small off-diagonal elements

$$\sigma_i = \sum_{k=1}^{r} (X_f)_{(i,k)} D_{(k,k)} (Y_f)_{(i,k)}$$

$$(J_M \dots J_2 J_1 X_0)\ D\ \left(Y_0^T \bar{J}_1 \bar{J}_2 \dots \bar{J}_N\right) = \Sigma$$

and

$$U_0 = J_1^T J_2^T \dots J_M^T \quad V_0 = \bar{J}_1 \bar{J}_2 \dots \bar{J}_N$$

Implicit Jacobi SVD

1. `Input:` Factors X, D and Y: $A = XDY^T \in \mathbb{R}^{m \times n}$
2. Run QR decompositions: $A = Q_X \begin{bmatrix} X_0 D Y_0^T & 0 \\ 0 & 0 \end{bmatrix} Q_Y^T$
3. Run implicit Jacobi SVD Jacobi algorithm on $A_0 = X_0 D Y_0^T \in \mathbb{R}^{r \times r}$ to compute singular values and vectors.
4. Until convergence: small off-diagonal elements

Implicit Jacobi SVD

1. `Input:` Factors X, D and Y: $A = XDY^T \in \mathbb{R}^{m \times n}$
2. Run QR decompositions: $A = Q_X \begin{bmatrix} X_0 D Y_0^T & 0 \\ 0 & 0 \end{bmatrix} Q_Y^T$
3. Run implicit Jacobi SVD Jacobi algorithm on $A_0 = X_0 D Y_0^T \in \mathbb{R}^{r \times r}$ to compute singular values and vectors.
4. Until convergence: small off-diagonal elements
5. `Output`: $A = U \Sigma V^T$

Implicit Jacobi SVD

1. `Input:` Factors X, D and Y: $A = XDY^T \in \mathbb{R}^{m \times n}$
2. Run QR decompositions: $A = Q_X \begin{bmatrix} X_0 D Y_0^T & 0 \\ 0 & 0 \end{bmatrix} Q_Y^T$
3. Run implicit Jacobi SVD Jacobi algorithm on $A_0 = X_0 D Y_0^T \in \mathbb{R}^{r \times r}$ to compute singular values and vectors.
4. Until convergence: small off-diagonal elements
5. `Output`: $A = U \Sigma V^T$

$$\Sigma = diag[\sigma_1, \ldots, \sigma_r]$$

$$U = Q_X \begin{bmatrix} U_0 & 0 \\ 0 & \mathbb{I}_{m-r} \end{bmatrix}, \quad V = Q_Y \begin{bmatrix} V_0 & 0 \\ 0 & \mathbb{I}_{n-r} \end{bmatrix}$$

Implicit Jacobi SVD

1. `Input:` Factors X, D and Y: $A = XDY^T \in \mathbb{R}^{m \times n}$
2. Run QR decompositions: $A = Q_X \begin{bmatrix} X_0 D Y_0^T & 0 \\ 0 & 0 \end{bmatrix} Q_Y^T$
3. Run implicit Jacobi SVD Jacobi algorithm on $A_0 = X_0 D Y_0^T \in \mathbb{R}^{r \times r}$ to compute singular values and vectors.
4. Until convergence: small off-diagonal elements
5. `Output`: $A = U \Sigma V^T$

Implicit Jacobi SVD

1. Input: Factors X, D and Y: $A = XDY^T \in \mathbb{R}^{m \times n}$
2. Run QR decompositions: $A = Q_X \begin{bmatrix} X_0 D Y_0^T & 0 \\ 0 & 0 \end{bmatrix} Q_Y^T$
3. Run implicit Jacobi SVD Jacobi algorithm on $A_0 = X_0 D Y_0^T \in \mathbb{R}^{r \times r}$ to compute singular values and vectors.
4. Until convergence: small off-diagonal elements
5. Output: $A = U \Sigma V^T$

The errors in the computed singular values and singular vectors are,

$$\frac{|\widehat{\sigma}_i - \sigma_i|}{|\sigma_i|} \leq O(u)\,\bar{\kappa} \quad \text{and} \quad \theta(v_i, \hat{v}_i) \leq \frac{O(u)\,\bar{\kappa}}{\min\limits_{j \neq i} \left| \frac{\sigma_i - \sigma_j}{\sigma_i} \right|} \quad \text{for all } i,$$

Implicit Jacobi SVD

1. `Input:` Factors X, D and Y: $A = XDY^T \in \mathbb{R}^{m \times n}$
2. Run QR decompositions: $A = Q_X \begin{bmatrix} X_0 D Y_0^T & 0 \\ 0 & 0 \end{bmatrix} Q_Y^T$
3. Run implicit Jacobi SVD Jacobi algorithm on $A_0 = X_0 D Y_0^T \in \mathbb{R}^{r \times r}$ to compute singular values and vectors.
4. Until convergence: small off-diagonal elements
5. `Output`: $A = U \Sigma V^T$

The errors in the computed singular values and singular vectors are,

$$\frac{|\widehat{\sigma}_i - \sigma_i|}{|\sigma_i|} \leq O(u)\,\bar{\kappa} \quad \text{and} \quad \theta(u_i, \hat{u}_i) \leq \frac{O(u)\,\bar{\kappa}}{\min\limits_{j \neq i} \left| \frac{\sigma_i - \sigma_j}{\sigma_i} \right|} \quad \text{for all } i,$$

Implicit Jacobi SVD

1. `Input:` Factors X, D and Y: $A = XDY^T \in \mathbb{R}^{m \times n}$
2. Run QR decompositions: $A = Q_X \begin{bmatrix} X_0 D Y_0^T & 0 \\ 0 & 0 \end{bmatrix} Q_Y^T$
3. Run implicit Jacobi SVD Jacobi algorithm on $A_0 = X_0 D Y_0^T \in \mathbb{R}^{r \times r}$ to compute singular values and vectors.
4. Until convergence: small off-diagonal elements
5. `Output`: $A = U \Sigma V^T$

The errors in the computed singular values and singular vectors are,

$$\frac{|\widehat{\sigma}_i - \sigma_i|}{|\sigma_i|} \leq O(u)\,\bar{\kappa} \quad \text{and} \quad \theta(u_i, \hat{u}_i) \leq \frac{O(u)\,\bar{\kappa}}{\min\limits_{j \neq i} \left| \frac{\sigma_i - \sigma_j}{\sigma_i} \right|} \quad \text{for all } i,$$

Depend on $\bar{\kappa} = \max\{\kappa(X), \kappa(Y)\}$, independent of $\kappa(A), \kappa(D)$.

Outline

Implicit Jacobi SVD for square factors $A = XDY^T$

input: $X, Y \in \mathbb{R}^{n\times n}$ and $D \in \mathbb{R}^{n\times n}$ diag., all nonsingular
output: sing-values, σ_i, and matrices of sing-vectors, U, V

$U = V = I_n$
repeat
  for i, j
    compute $a_{ii}, a_{ij}, a_{ji}, a_{jj}$ of $A = XDY^T$ and $J, \bar{J}$ Jacobi rotations:

$$J \begin{bmatrix} a_{ii} & a_{ij} \\ a_{ji} & a_{jj} \end{bmatrix} \bar{J} = \begin{bmatrix} \mu_1 & 0 \\ 0 & \mu_2 \end{bmatrix}$$

$$X = J\,X, \quad Y^T = Y^T\,\bar{J}$$
$$U = U\,J^T, \quad V = V\,\bar{J}$$

  endfor
until convergence $\frac{|a_{ij}|}{\sqrt{|a_{ii}a_{jj}|}} \leq \text{tol}$ for all i, j $\qquad$ tol $= O(u)$
compute $\sigma_k = a_{kk}$ for $k = 1, 2, \ldots, n$.

Key (well known) Facts

Key (well known) Facts

1. This implicit Jacobi algorithm is mathematically equivalent to the non-implicit one.

Key (well known) Facts

1. This implicit Jacobi algorithm is mathematically equivalent to the non-implicit one.
2. Multiplicative perturbation theory

Key (well known) Facts

1. This implicit Jacobi algorithm is mathematically equivalent to the non-implicit one.
2. Multiplicative perturbation theory

Theorem (Eisenstat-Ipsen 1995, R-C Li 1998)

Let $\widetilde{A}, A \in \mathbb{C}^{n\times n}$ *and*

$$\widetilde{A} = (I+E)A(I+F).$$

Singular values:

$$|\widetilde{\sigma}_i - \sigma_i| \leq (2\,\eta + \eta^2)\,\sigma_i \quad 1 \leq i \leq n$$

$$\eta = \max\{\|E\|_2, \|F\|_2\} < 1,\ \eta' = 2\eta + \eta^2$$

Key (well known) Facts

1. This implicit Jacobi algorithm is mathematically equivalent to the non-implicit one.
2. Multiplicative perturbation theory

Theorem (Eisenstat-Ipsen 1995, R-C Li 1998)

Let $\widetilde{A}, A \in \mathbb{C}^{n\times n}$ *and*

$$\widetilde{A} = (I+E)A(I+F).$$

Singular vectors:

$$\max\{\theta(u_i,\hat{u}_i),\theta(v_i,\hat{v}_i)\} \leq 2\sqrt{n}\left[\eta + \frac{\eta'}{1-\eta}\,\frac{1}{\min\limits_{j\neq i}\left|\frac{\sigma_i-\sigma_j}{\sigma_i}\right|}\right] \qquad 1\leq i\leq n$$

$\eta = \max\{\|E\|_2, \|F\|_2\} < 1,\ \eta' = 2\eta+\eta^2$

Key (well known) Facts

1. This implicit Jacobi algorithm is mathematically equivalent to the non-implicit one.
2. Multiplicative perturbation theory
3. RRD

Key (well known) Facts

1. This implicit Jacobi algorithm is mathematically equivalent to the non-implicit one.
2. Multiplicative perturbation theory
3. RRD

Theorem (Demmel et al. 1999)

Let $A = XDY^T \in \mathbb{R}^{m\times n}$ *be an RRD, and* $\widetilde{A} = \widetilde{X}\widetilde{D}\widetilde{Y}$ *perturbations, that satisfy, with* $\epsilon < 1$,

$$\frac{\|\widetilde{X} - X\|_2}{\|X\|_2}, \frac{\|\widetilde{Y} - Y\|_2}{\|Y\|_2} \leq \epsilon \quad \textit{and} \quad \frac{|\widetilde{d}_i - d_i|}{|d_i|} \leq \epsilon \quad \textit{for } i = 1, \ldots, r.$$

Then

$$\widetilde{X}\widetilde{D}\widetilde{Y}^T = (I + E)A(I + F)^T,$$

with $\|E\|_2 \leq \epsilon\kappa(X)$ *and* $\|F\|_2 \leq (2\epsilon + \epsilon^2)\kappa(Y)$.

Key (well known) Facts

1. This implicit Jacobi algorithm is mathematically equivalent to the non-implicit one.
2. Multiplicative perturbation theory
3. RRD
4. QR decompositions

Key (well known) Facts

1. This implicit Jacobi algorithm is mathematically equivalent to the non-implicit one.
2. Multiplicative perturbation theory
3. RRD
4. QR decompositions

Theorem

Let $\widehat{Q}$ and $\widehat{R}$ be the computed factors (with Householder reflections) of the full QR decomposition of $X = QR \in \mathbb{R}^{m\times r}$, then there exists an exact orthogonal matrix $Q \in \mathbb{R}^{m\times m}$ such that

$$Q\widehat{R} = (I + E)\, X$$

where $\|E\|_F \leq O(mr^{3/2}u)\,\kappa(X)$ and

$$\|\widehat{Q} - Q\|_F \leq O(m^{3/2}ru)$$

Key (well known) Facts

1. This implicit Jacobi algorithm is mathematically equivalent to the non-implicit one.
2. Multiplicative perturbation theory
3. RRD
4. QR decompositions
5. Orthogonal Jacobi rotations

Key (well known) Facts

1. This implicit Jacobi algorithm is mathematically equivalent to the non-implicit one.
2. Multiplicative perturbation theory
3. RRD
4. QR decompositions
5. Orthogonal Jacobi rotations

Lemma (Small multiplicative backward errors of Jacobi rotations)

Let J_i be exact Jacobi rotations and $\widehat{J}_i$ their floating point approximations. Then

$$\widehat{X}_M \equiv \mathtt{fl}(\widehat{J}_M^T \cdots \widehat{J}_1^T X) = (I+F) J_M^T \cdots J_1^T X$$

where $\|F\|_2 = O(M\, u\, \kappa(X))$, and

Key (well known) Facts

1. This implicit Jacobi algorithm is mathematically equivalent to the non-implicit one.
2. Multiplicative perturbation theory
3. RRD
4. QR decompositions
5. Orthogonal Jacobi rotations

Key (new) Facts

1. Stopping criteria

$$\frac{|a_{ij}|}{\sqrt{|a_{ii}a_{jj}|}} \leq \text{tol} \quad \text{for all } i, j$$

Key (new) Facts

1. Stopping criteria

$$\frac{|a_{ij}|}{\sqrt{|a_{ii}a_{jj}|}} \leq \text{tol} \quad \text{for all } i, j$$

2. Final computation of the singular values

$$\sigma_i = \sum_{j=1}^{r} x_{ij} d_j y_{ij} \quad \text{for all } i$$

Key (new) Facts

1. Stopping criteria

$$\frac{|a_{ij}|}{\sqrt{|a_{ii}a_{jj}|}} \leq \text{tol} \quad \text{for all } i, j$$

Diagonally dominant matrices

Theorem

Let $A \in \mathbb{R}^{n\times n}$ $a_{ii} \neq 0$, *with for all i, and*

$$\frac{|a_{ij}|}{\sqrt{|a_{ii}a_{jj}|}} \leq \delta, \quad \text{for all } i \neq j,$$

where $\delta \leq \frac{1}{5n}$, *then*

$$diag(a_{11}, \ldots, a_{nn}) = (I + F_1)A(I + F_2)$$

with $\|F_1\|_F, \|F_2\|_F \leq n\delta + O(\delta^2)$.

Key Facts (New)

2. Final computation of the singular values

$$\sigma_i = \sum_{j=1}^{r} x_{ij} d_j y_{ij} \quad \text{for all } i$$

Errors in the diagonal entries

$$a_{ii} = \sum_{k=1}^{n} x_{ik} d_k y_{ik}$$

$$\left| \frac{\mathtt{fl}(a_{ii}) - a_{ii}}{a_{ii}} \right| \leq \frac{3(n-1)u}{1-3(n-1)u} \frac{\sum_{k=1}^{n} |x_{ik}||d_k||y_{ik}|}{\left| \sum_{k=1}^{n} x_{ik} d_k y_{ik} \right|}$$

Errors in the diagonal entries of almost diagonal RRDs

Input: $\kappa(X) = 7.21, \quad \kappa(Y) = 8.29$

$$XDY^T = \begin{bmatrix} 1 & 1 & 1 \\ -1 & -1 & 1 \\ 2 & 1 & 1 \end{bmatrix} \begin{bmatrix} 10^{50} & & \\ & 1 & \\ & & -10^{50} \end{bmatrix} \begin{bmatrix} 2 & -3 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & -1 \end{bmatrix}$$

Errors in the diagonal entries of almost diagonal RRDs

Input: $\kappa(X) = 7.21, \quad \kappa(Y) = 8.29$

$$XDY^T = \begin{bmatrix} 1 & 1 & 1 \\ -1 & -1 & 1 \\ 2 & 1 & 1 \end{bmatrix} \begin{bmatrix} 10^{50} & & \\ & 1 & \\ & & -10^{50} \end{bmatrix} \begin{bmatrix} 2 & -3 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & -1 \end{bmatrix}$$

Running implicit Jacobi SVD until convergence

$$\begin{bmatrix} -2.4 & -1.6 & -1.0 \\ -8.4 \cdot 10^{-85} & 0.53 & 5.9 \cdot 10^{-84} \\ 0.31 & 0.42 & -1.4 \end{bmatrix} \begin{bmatrix} 10^{50} & & \\ & 1 & \\ & & -10^{50} \end{bmatrix} \begin{bmatrix} 3.7 & 3.4 \cdot 10^{-49} & -0.63 \\ -0.89 & -0.49 & -0.21 \\ -0.81 & -2.6 \cdot 10^{-49} & -1.5 \end{bmatrix}$$

Errors in the diagonal entries of almost diagonal RRDs

Input: $\kappa(X) = 7.21, \quad \kappa(Y) = 8.29$

$$XDY^T = \begin{bmatrix} 1 & 1 & 1 \\ -1 & -1 & 1 \\ 2 & 1 & 1 \end{bmatrix} \begin{bmatrix} 10^{50} & & \\ & 1 & \\ & & -10^{50} \end{bmatrix} \begin{bmatrix} 2 & -3 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & -1 \end{bmatrix}$$

Running implicit Jacobi SVD until convergence

$$\begin{bmatrix} -2.4 & -1.6 & -1.0 \\ -8.4 \cdot 10^{-85} & 0.53 & 5.9 \cdot 10^{-84} \\ 0.31 & 0.42 & -1.4 \end{bmatrix} \begin{bmatrix} 10^{50} & & \\ & 1 & \\ & & -10^{50} \end{bmatrix} \begin{bmatrix} 3.7 & 3.4 \cdot 10^{-49} & -0.63 \\ -0.89 & -0.49 & -0.21 \\ -0.81 & -2.6 \cdot 10^{-49} & -1.5 \end{bmatrix}$$

$$= \begin{bmatrix} -9.8 \cdot 10^{50} & -110. & -2.1 \cdot 10^{34} \\ -0.45 & -0.26 & -0.11 \\ 0 & -26. & -2.3 \cdot 10^{50} \end{bmatrix}$$

Errors in the diagonal entries of almost diagonal RRDs

Input: $\kappa(X) = 7.21, \quad \kappa(Y) = 8.29$

$$XDY^T = \begin{bmatrix} 1 & 1 & 1 \\ -1 & -1 & 1 \\ 2 & 1 & 1 \end{bmatrix} \begin{bmatrix} 10^{50} & & \\ & 1 & \\ & & -10^{50} \end{bmatrix} \begin{bmatrix} 2 & -3 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & -1 \end{bmatrix}$$

Running implicit Jacobi SVD until convergence

$$\begin{bmatrix} -2.4 & -1.6 & -1.0 \\ -8.4 \cdot 10^{-85} & 0.53 & 5.9 \cdot 10^{-84} \\ 0.31 & 0.42 & -1.4 \end{bmatrix} \begin{bmatrix} 10^{50} & & \\ & 1 & \\ & & -10^{50} \end{bmatrix} \begin{bmatrix} 3.7 & 3.4 \cdot 10^{-49} & -0.63 \\ -0.89 & -0.49 & -0.21 \\ -0.81 & -2.6 \cdot 10^{-49} & -1.5 \end{bmatrix}$$

$$= \begin{bmatrix} -9.8 \cdot 10^{50} & -110. & -2.1 \cdot 10^{34} \\ -0.45 & -0.26 & -0.11 \\ 0 & -26. & -2.3 \cdot 10^{50} \end{bmatrix}$$

Errors in the diagonal entries of almost diagonal RRDs

Input: $\kappa(X) = 7.21, \quad \kappa(Y) = 8.29$

$$XDY^T = \begin{bmatrix} 1 & 1 & 1 \\ -1 & -1 & 1 \\ 2 & 1 & 1 \end{bmatrix} \begin{bmatrix} 10^{50} & & \\ & 1 & \\ & & -10^{50} \end{bmatrix} \begin{bmatrix} 2 & -3 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & -1 \end{bmatrix}$$

Running implicit Jacobi SVD until convergence

$$\begin{bmatrix} -2.4 & -1.6 & -1.0 \\ -8.4 \cdot 10^{-85} & 0.53 & 5.9 \cdot 10^{-84} \\ 0.31 & 0.42 & -1.4 \end{bmatrix} \begin{bmatrix} 10^{50} & & \\ & 1 & \\ & & -10^{50} \end{bmatrix} \begin{bmatrix} 3.7 & 3.4 \cdot 10^{-49} & -0.63 \\ -0.89 & -0.49 & -0.21 \\ -0.81 & -2.6 \cdot 10^{-49} & -1.5 \end{bmatrix}$$

$$= \begin{bmatrix} -9.8 \cdot 10^{50} & -110. & -2.1 \cdot 10^{34} \\ -0.45 & -0.26 & -0.11 \\ 0 & -26. & -2.3 \cdot 10^{50} \end{bmatrix}$$

Errors in the diagonal entries of almost diagonal RRDs

Input: $\kappa(X) = 7.21, \quad \kappa(Y) = 8.29$

$$XDY^T = \begin{bmatrix} 1 & 1 & 1 \\ -1 & -1 & 1 \\ 2 & 1 & 1 \end{bmatrix} \begin{bmatrix} 10^{50} & & \\ & 1 & \\ & & -10^{50} \end{bmatrix} \begin{bmatrix} 2 & -3 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & -1 \end{bmatrix}$$

Running implicit Jacobi SVD until convergence

$$\begin{bmatrix} -2.4 & -1.6 & -1.0 \\ -8.4 \cdot 10^{-85} & 0.53 & 5.9 \cdot 10^{-84} \\ 0.31 & 0.42 & -1.4 \end{bmatrix} \begin{bmatrix} 10^{50} & & \\ & 1 & \\ & & -10^{50} \end{bmatrix} \begin{bmatrix} 3.7 & 3.4 \cdot 10^{-49} & -0.63 \\ -0.89 & -0.49 & -0.21 \\ -0.81 & -2.6 \cdot 10^{-49} & -1.5 \end{bmatrix}$$

$$= \begin{bmatrix} -9.8 \cdot 10^{50} & -110. & -2.1 \cdot 10^{34} \\ -0.45 & -0.26 & -0.11 \\ 0 & -26. & -2.3 \cdot 10^{50} \end{bmatrix}$$

There is no cancellation

$$-0.26 = (-8.4 \cdot 10^{-85})(10^{50})(3.4 \cdot 10^{-49}) + 0.53(-0.49) + (5.9 \cdot 10^{-84})(-10^{50})(-2.6 \cdot 10^{-49})$$

Absence of cancellation in the diagonal

Theorem

Let $X, D, Y \in \mathbb{R}^{n\times n}$ *be nonsingular and* $D = \mathrm{diag}(d_1, \ldots, d_n)$ *be diagonal. If the matrix* $A = XDY^T$ *satisfies* $a_{ii} = \sum_{k=1}^{n} x_{ik} d_k y_{ik} \neq 0$ *for all* i*, and*

$$\frac{|a_{ij}|}{\sqrt{|a_{ii}a_{jj}|}} \leq \delta, \quad \text{for all } i \neq j, \quad \text{where } \delta \leq \tfrac{1}{5n}, \text{ then}$$

Absence of cancellation in the diagonal

Theorem

Let $X, D, Y \in \mathbb{R}^{n\times n}$ *be nonsingular and* $D = \mathrm{diag}(d_1, \ldots, d_n)$ *be diagonal. If the matrix* $A = XDY^T$ *satisfies* $a_{ii} = \sum_{k=1}^{n} x_{ik} d_k y_{ik} \neq 0$ *for all* i*, and*

$$\frac{|a_{ij}|}{\sqrt{|a_{ii}a_{jj}|}} \leq \delta, \quad \text{for all } i \neq j, \quad \text{where } \delta \leq \tfrac{1}{5n}, \text{ then}$$

$$\frac{\sum_{k=1}^{n} |x_{ik}||d_k||y_{ik}|}{|a_{ii}|} \leq \left(1 + O(n^2\delta)\right) \bar{\kappa}, \qquad i = 1, \ldots, n.$$

with $\bar{\kappa} = \max\{\kappa(X), \kappa(Y)\}$

Implicit Jacobi is multiplicative backward stable

Theorem

Let M and N be resp. the number of left and right rotations applied by implicit Jacobi on $A = XDY^T \in \mathbb{R}^{m\times n}$ until convergence, and $\widehat{\Sigma}$ and $\widehat{U}, \widehat{V}$ be the computed matrices of singular values and left/right singular vectors. Then there exists two exact orthogonal matrices $U \in \mathbb{R}^{m\times m}, V \in \mathbb{R}^{n\times n}$ such that

$$U\widehat{\Sigma}V^T = (I+E)\,XDY^T\,(I+F)^T$$

with,

Implicit Jacobi is multiplicative backward stable

Theorem

Let M and N be resp. the number of left and right rotations applied by implicit Jacobi on $A = XDY^T \in \mathbb{R}^{m\times n}$ until convergence, and $\widehat{\Sigma}$ and $\widehat{U}, \widehat{V}$ be the computed matrices of singular values and left/right singular vectors. Then there exists two exact orthogonal matrices $U \in \mathbb{R}^{m\times m}, V \in \mathbb{R}^{n\times n}$ such that

$$U\widehat{\Sigma}V^T = (I+E)\,XDY^T\,(I+F)^T$$

with,

$$\|E\|_F = O(u)\,(M\,\kappa(X) + m^2\,\bar{\kappa}) \quad \textit{and} \quad \|\widehat{U} - U\|_F = O(M\,u).$$

Implicit Jacobi is multiplicative backward stable

Theorem

Let M and N be resp. the number of left and right rotations applied by implicit Jacobi on $A = XDY^T \in \mathbb{R}^{m\times n}$ until convergence, and $\widehat{\Sigma}$ and $\widehat{U}, \widehat{V}$ be the computed matrices of singular values and left/right singular vectors. Then there exists two exact orthogonal matrices $U \in \mathbb{R}^{m\times m}, V \in \mathbb{R}^{n\times n}$ such that

$$U\widehat{\Sigma}V^T = (I+E)\,XDY^T\,(I+F)^T$$

with,

$$\|F\|_F = O(u)\,(N\,\kappa(Y) + n^2\,\bar{\kappa}) \quad \textit{and} \quad \|\widehat{V} - V\|_F = O(N\,u).$$

with $\bar{\kappa} = \max\{\kappa(X), \kappa(Y)\}$

Implicit Jacobi is multiplicative backward stable

Corollary (Forward errors in singular values and vectors)

$$\frac{|\widehat{\sigma}_i - \sigma_i|}{|\sigma_i|} \leq O(u)\bar{\kappa}$$

and

$$\max\{\theta(u_i, \widehat{u}_i), \theta(v_i, \widehat{v}_i)\} \leq \frac{O(u)\bar{\kappa}}{\min\limits_{j \neq i} \left| \frac{\sigma_i - \sigma_j}{\sigma_i} \right|} \quad \textit{for all} \quad i,$$

with $\bar{\kappa} = \max\{\kappa(X), \kappa(Y)\}$

Outline

Numerical Experiments

- We have done numerical experiments to confirm the high relative accuracy that we have rigorously proven.

Numerical Experiments

- We have done numerical experiments to confirm the high relative accuracy that we have rigorously proven.
- We have implemented the Implicit Jacobi algorithm in four versions

Numerical Experiments

- We have done numerical experiments to confirm the high relative accuracy that we have rigorously proven.
- We have implemented the Implicit Jacobi algorithm in four versions
  1. Symmetrize A + Classical Jacobi

Numerical Experiments

- We have done numerical experiments to confirm the high relative accuracy that we have rigorously proven.
- We have implemented the Implicit Jacobi algorithm in four versions
  1. Symmetrize A + Classical Jacobi
  2. Symmetrize A + Classical Jacobi (Preconditioned)

Numerical Experiments

- We have done numerical experiments to confirm the high relative accuracy that we have rigorously proven.
- We have implemented the Implicit Jacobi algorithm in four versions
  1. Symmetrize A + Classical Jacobi
  2. Symmetrize A + Classical Jacobi (Preconditioned)
  3. Triangular Kogbetliantz

Numerical Experiments

- We have done numerical experiments to confirm the high relative accuracy that we have rigorously proven.
- We have implemented the Implicit Jacobi algorithm in four versions
  1. Symmetrize A + Classical Jacobi
  2. Symmetrize A + Classical Jacobi (Preconditioned)
  3. Triangular Kogbetliantz
  4. Aprox. Triangular Kogbetliantz

Numerical Experiments

- We have done numerical experiments to confirm the high relative accuracy that we have rigorously proven.
- We have implemented the Implicit Jacobi algorithm in four versions
  1. Symmetrize A + Classical Jacobi
  2. Symmetrize A + Classical Jacobi (Preconditioned)
  3. Triangular Kogbetliantz
  4. Aprox. Triangular Kogbetliantz
- We have compared it with the two hra algorithms in [Demmel et al, 1999]

Numerical Experiments

- We have done numerical experiments to confirm the high relative accuracy that we have rigorously proven.
- We have implemented the Implicit Jacobi algorithm in four versions
  1. Symmetrize A + Classical Jacobi
  2. Symmetrize A + Classical Jacobi (Preconditioned)
  3. Triangular Kogbetliantz
  4. Aprox. Triangular Kogbetliantz
- We have compared it with the two hra algorithms in [Demmel et al, 1999]
  1. Algorithm 3.1, QR+one one-sided Jacobi, with the factor $\kappa(R')$

Numerical Experiments

- We have done numerical experiments to confirm the high relative accuracy that we have rigorously proven.
- We have implemented the Implicit Jacobi algorithm in four versions
  1. Symmetrize A + Classical Jacobi
  2. Symmetrize A + Classical Jacobi (Preconditioned)
  3. Triangular Kogbetliantz
  4. Aprox. Triangular Kogbetliantz
- We have compared it with the two hra algorithms in [Demmel et al, 1999]
  1. Algorithm 3.1, QR+one one-sided Jacobi, with the factor $\kappa(R')$
  2. Algorithm 3.2, two one-sided Jacobi, without the factor $\kappa(R')$

Numerical Experiments

- We have done numerical experiments to confirm the high relative accuracy that we have rigorously proven.
- We have implemented the Implicit Jacobi algorithm in four versions
  1. Symmetrize A + Classical Jacobi
  2. Symmetrize A + Classical Jacobi (Preconditioned)
  3. Triangular Kogbetliantz
  4. Aprox. Triangular Kogbetliantz
- We have compared it with the two hra algorithms in [Demmel et al, 1999]
  1. Algorithm 3.1, QR+one one-sided Jacobi, with the factor $\kappa(R')$
  2. Algorithm 3.2, two one-sided Jacobi, without the factor $\kappa(R')$
- To test the speed we have compared the number of Jacobi sweeps performed by each algorithm.

Numerical Experiments

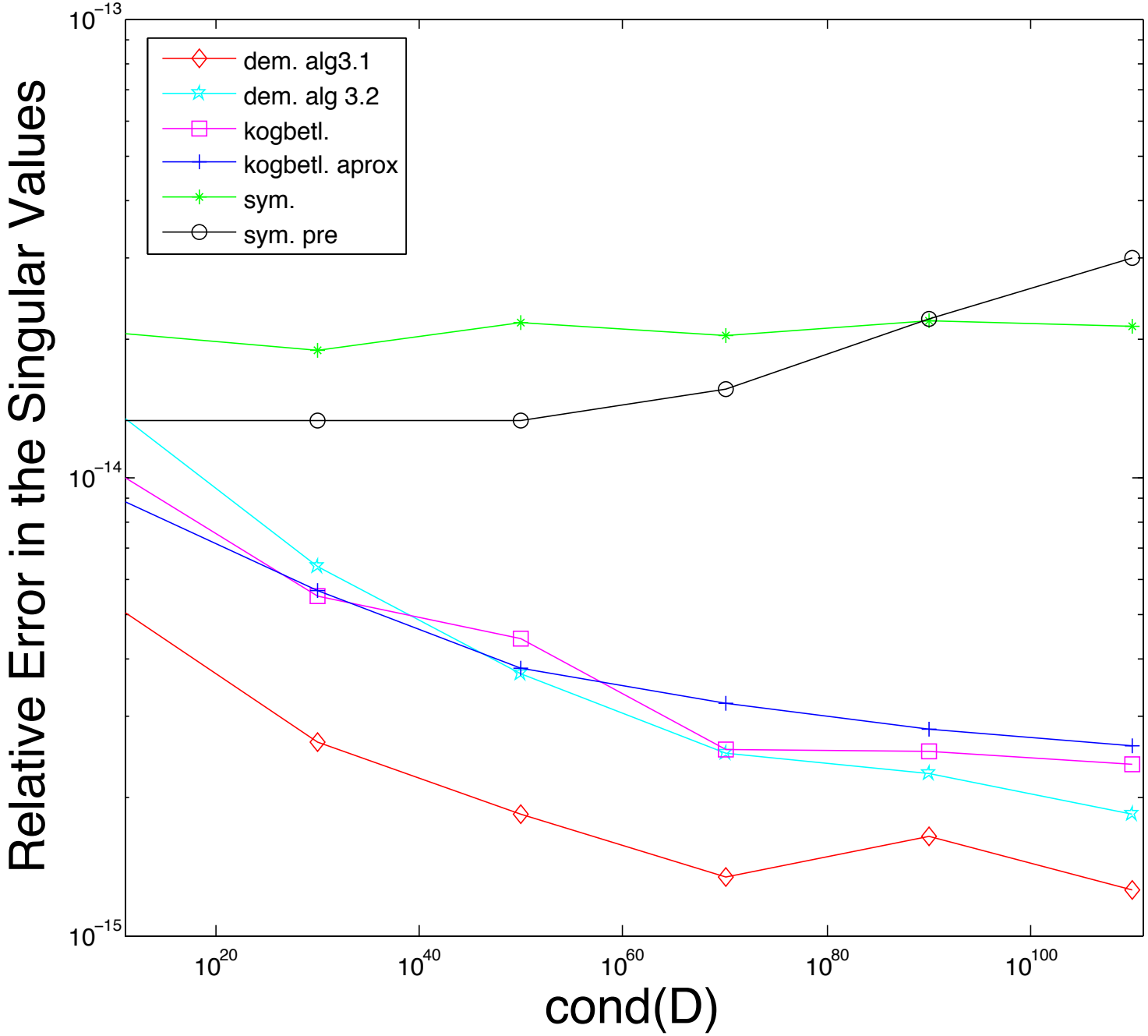
- We have done numerical experiments to confirm the high relative accuracy that we have rigorously proven.
- We have implemented the Implicit Jacobi algorithm in four versions
  1. Symmetrize A + Classical Jacobi
  2. Symmetrize A + Classical Jacobi (Preconditioned)
  3. Triangular Kogbetliantz
  4. Aprox. Triangular Kogbetliantz
- We have compared it with the two hra algorithms in [Demmel et al, 1999]
  1. Algorithm 3.1, QR+one one-sided Jacobi, with the factor $\kappa(R')$
  2. Algorithm 3.2, two one-sided Jacobi, without the factor $\kappa(R')$
- To test the speed we have compared the number of Jacobi sweeps performed by each algorithm.
- We have used `gallery('randsvd',...)` by N. Higham in MATLAB to generate random RRDs with X well-conditioned and D and extremely ill-conditioned.

Number of sweeps: Increasing $\kappa(D)$

In all these tests $\kappa(X) = \kappa(Y) = 30$ and X, D are 100×100.

D has entries with magnitudes geometrically distributed
$|d_i| = \kappa(D)^{\frac{i-1}{n-1}}, \quad i = 1, \ldots, n$

$\kappa(D)$	sym	sym pre	kogb	kogb aprox	dem-3.1	dem-3.2
10^{10}	16.2	6	6.2	6.2	5.2	10.2
10^{30}	22.8	6.4	5	5	4	8.2
10^{50}	26.6	13.8	6	6	3.6	7.4
10^{70}	29.2	20.4	4	4	3	6
10^{90}	31.6	24.4	4	4	3	6
10^{110}	32	28	4	4	3	6

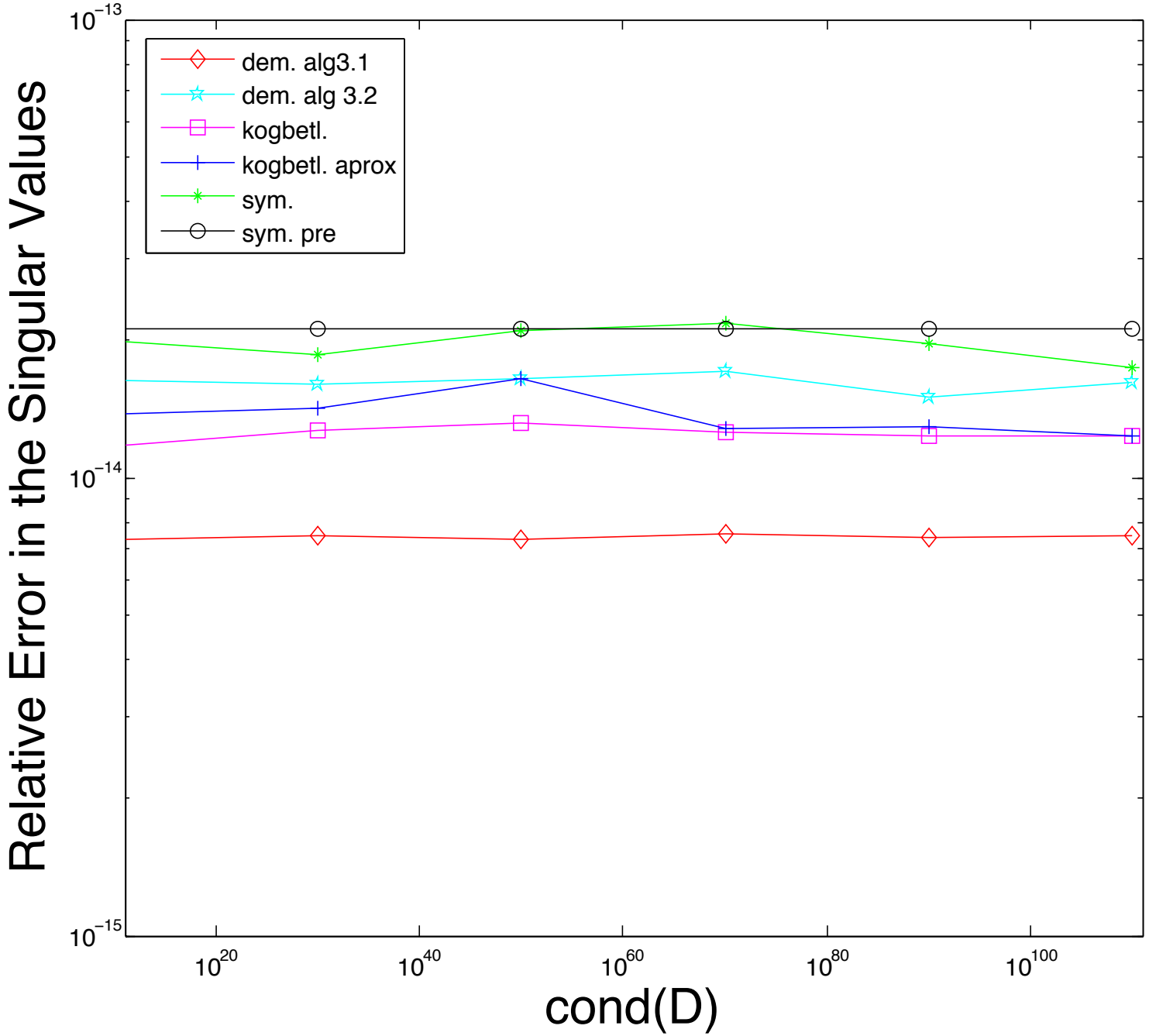

dem. alg3.1
dem. alg 3.2
kogbetl.
kogbetl. aprox
sym.
sym. pre
Relative Error in the Singular Values
cond(D)
10^{-13}
10^{-14}
10^{-15}
10^{20}
10^{40}
10^{60}
10^{80}
10^{100}

Number of sweeps: Increasing $\kappa(D)$

In all of these tests $\kappa(X) = \kappa(Y) = 30$ and X, D are 100×100.

D has one entry with magnitude 1 and the rest $1/\kappa(D)$

$\kappa(D)$	sym	sym pre	kogb	kogb aprox	dem-3.1	dem-3.2
10^{10}	10	9	9	9.2	9.2	18.8
10^{30}	10.2	9	9	9.4	9.2	18.6
10^{50}	11.2	10.8	9.6	9.4	8.8	18.6
10^{70}	11.4	11.2	9.2	9.4	9	18.6
10^{90}	11.4	11.6	9.4	9.2	9.2	18.2
10^{110}	11.8	11.8	10	10	9	18.8


dem. alg3.1
dem. alg 3.2
kogbetl.
kogbetl. aprox
sym.
sym. pre
Relative Error in the Singular Values
10^{-13}
10^{-14}
10^{-15}
10^{20}
10^{40}
10^{60}
10^{80}
10^{100}
cond(D)

Number of sweeps: Increasing the dimension

In all of these tests $\kappa(X) = \kappa(Y) = 100$, $\kappa(D) = 10^{40}$.

D has entries with magnitudes geometrically distributed

$|d_i| = \kappa(D)^{\frac{i-1}{n-1}}, \quad i = 1, \ldots, n$

n	sym	sym pre	kogb	kogb aprox	dem-3.1	dem-3.2
100	34	16	6	6	4	9
500	58	20	7	7	6	12
1000	68	22	7	8	7	13
2000	76	24	9	9	7	14

Number of sweeps: Increasing the dimension

In all of these tests $\kappa(X) = 100$, $\kappa(D) = 10^{40}$.

D has one entry with magnitude 1 and the rest $1/\kappa(D)$

n	sym	sym pre	kogb	kogb aprox	dem-3.1	dem-3.2
100	12	12	10	10	10	21
500	13	12	12	12	14	25
1000	14	13	13	13	15	28
2000	14	13	14	15	17	30

Outline

Conclusions

- The implicit Jacobi SVD algorithm on rank revealing factorizations $A = XDY^T$ is very simple and natural,

Conclusions

- The implicit Jacobi SVD algorithm on rank revealing factorizations $A = XDY^T$ is very simple and natural,
- computes the singular values and vectors of A to high relative accuracy,

Conclusions

- The implicit Jacobi SVD algorithm on rank revealing factorizations $A = XDY^T$ is very simple and natural,
- computes the singular values and vectors of A to high relative accuracy,
- the error bounds are the best possible ones from the sensitivity of the problem,

Conclusions

- The implicit Jacobi SVD algorithm on rank revealing factorizations $A = XDY^T$ is very simple and natural,
- computes the singular values and vectors of A to high relative accuracy,
- the error bounds are the best possible ones from the sensitivity of the problem,
- is backward stable in a strong multiplicative sense.

Conclusions

- The implicit Jacobi SVD algorithm on rank revealing factorizations $A = XDY^T$ is very simple and natural,
- computes the singular values and vectors of A to high relative accuracy,
- the error bounds are the best possible ones from the sensitivity of the problem,
- is backward stable in a strong multiplicative sense.
- the number of Jacobi sweeps is comparable to other algorithms.

Conclusions

- The implicit Jacobi SVD algorithm on rank revealing factorizations $A = XDY^T$ is very simple and natural,
- computes the singular values and vectors of A to high relative accuracy,
- the error bounds are the best possible ones from the sensitivity of the problem,
- is backward stable in a strong multiplicative sense.
- the number of Jacobi sweeps is comparable to other algorithms.
- More research to speed up the algorithm is needed [as is Drmac's works for SVD one-sided Jacobi].