Fast and Accurate Implementations of the One–Sided Indefinite Block Jacobi Method

Vjeran Hari\textsuperscript{1} Sanja Singer\textsuperscript{2} Saša Singer\textsuperscript{1}

\textsuperscript{1}Department of Mathematics, University of Zagreb

\textsuperscript{2}Faculty of Mechanical Engineering and Naval Architecture, University of Zagreb

International Workshop on Accurate Solution of Eigenvalue Problems VI
Motivation

Our goals are graded as follows:

▶ create the fastest possible (indefinite) block algorithm,
▶ should remain accurate (as ordinary $J$–Jacobi),
▶ probable side-effect – faster definite algorithm,
▶ catch (if you can) speed of dqds and/or DC.

Our results:

▶ families of indefinite block algorithms are analyzed,
▶ convergence (of most of them) can be proved,
▶ accuracy proved,
▶ title ‘best algorithm’ not yet awarded.
Motivation

Our goals are graded as follows:

- create the fastest possible (indefinite) block algorithm,
- should remain accurate (as ordinary $J$–Jacobi),
- probable side-effect – faster definite algorithm,
- catch (if you can) speed of dqds and/or DC.

Our results:

- families of indefinite block algorithms are analyzed,
- convergence (of most of them) can be proved,
- accuracy proved,
- title ‘best algorithm’ not yet awarded.
Motivation

Our goals are graded as follows:

- create the fastest possible (indefinite) block algorithm,
- should remain accurate (as ordinary $J$–Jacobi),
- probable side-effect – faster definite algorithm,
- catch (if you can) speed of dqds and/or DC.

Our results:

- families of indefinite block algorithms are analyzed,
- convergence (of most of them) can be proved,
- accuracy proved,
- title ‘best algorithm’ not yet awarded.
**Motivation**

Our goals are graded as follows:

- create the fastest possible (indefinite) block algorithm,
- should remain accurate (as ordinary $J$–Jacobi),
- probable side-effect – faster definite algorithm,
- catch (if you can) speed of dqds and/or DC.

Our results:

- families of indefinite block algorithms are analyzed,
- convergence (of most of them) can be proved,
- accuracy proved,
- title ‘best algorithm’ not yet awarded.
Motivation

Our goals are graded as follows:

- create the fastest possible (indefinite) block algorithm,
- should remain accurate (as ordinary $J$–Jacobi),
- probable side-effect – faster definite algorithm,
- catch (if you can) speed of dqds and/or DC.

Our results:

- families of indefinite block algorithms are analyzed,
- convergence (of most of them) can be proved,
- accuracy proved,
- title ‘best algorithm’ not yet awarded.
Motivation

Our goals are graded as follows:

- create the fastest possible (indefinite) block algorithm,
- should remain accurate (as ordinary $J$–Jacobi),
- probable side-effect – faster definite algorithm,
- catch (if you can) speed of dqds and/or DC.

Our results:

- families of indefinite block algorithms are analyzed,
- convergence (of most of them) can be proved,
- accuracy proved,
- title ‘best algorithm’ not yet awarded.
Motivation

Our goals are graded as follows:

- create the fastest possible (indefinite) block algorithm,
- should remain accurate (as ordinary $J$–Jacobi),
- probable side-effect – faster definite algorithm,
- catch (if you can) speed of dqds and/or DC.

Our results:

- families of indefinite block algorithms are analyzed,
- convergence (of most of them) can be proved,
- accuracy proved,
- title ‘best algorithm’ not yet awarded.
Motivation

Our goals are graded as follows:

- create the fastest possible (indefinite) block algorithm,
- should remain accurate (as ordinary $J$–Jacobi),
- probable side-effect – faster definite algorithm,
- catch (if you can) speed of dqds and/or DC.

Our results:

- families of indefinite block algorithms are analyzed,
- convergence (of most of them) can be proved,
- accuracy proved,
- title ‘best algorithm’ not yet awarded.
Motivation

Our goals are graded as follows:

- create the fastest possible (indefinite) block algorithm,
- should remain accurate (as ordinary $J$–Jacobi),
- probable side-effect – faster definite algorithm,
- catch (if you can) speed of dqds and/or DC.

Our results:

- families of indefinite block algorithms are analyzed,
- convergence (of most of them) can be proved,
- accuracy proved,
- title ‘best algorithm’ not yet awarded.
Motivation

Our goals are graded as follows:

- create the fastest possible (indefinite) block algorithm,
- should remain accurate (as ordinary $J$–Jacobi),
- probable side-effect – faster definite algorithm,
- catch (if you can) speed of dqds and/or DC.

Our results:

- families of indefinite block algorithms are analyzed,
- convergence (of most of them) can be proved,
- accuracy proved,
- title ‘best algorithm’ not yet awarded.
If Hermitian, indefinite $H$ is given, the method consists of two steps:

1. factorize $H$ using symmetric indefinite factorization (with pivoting): $H = MDM^*$, $D$ block diagonal. Additional diagonalization of $D$ and scaling of columns of $M$ yields

$$H = GJG^*, \quad J = \text{diag}(j_{11}, \ldots, j_{nn}), \quad j_{ii} \in \{-1, 1\}$$

2. diagonalize the pair $(G^*G, J)$ using congruence by $J$–unitary matrices, either implicitly (one-sided algorithm) or explicitly (two-sided algorithm).
If Hermitian, indefinite $H$ is given, the method consists of two steps:

1. factorize $H$ using symmetric indefinite factorization (with pivoting): $H = MDM^*$, $D$ block diagonal. Additional diagonalization of $D$ and scaling of columns of $M$ yields

$$H = GJG^*, \quad J = \text{diag}(j_{11}, \ldots, j_{nn}), \quad j_{ii} \in \{-1, 1\}$$

2. diagonalize the pair $(G^*G, J)$ using congruence by $J$–unitary matrices, either implicitly (one-sided algorithm) or explicitly (two-sided algorithm).
If Hermitian, indefinite $H$ is given, the method consists of two steps:

1. factorize $H$ using symmetric indefinite factorization (with pivoting): $H = MDM^*$, $D$ block diagonal. Additional diagonalization of $D$ and scaling of columns of $M$ yields

$$H = GJG^*, \quad J = \text{diag}(j_{11}, \ldots, j_{nn}), \quad j_{ii} \in \{-1, 1\}$$

2. diagonalize the pair $(G^*G, J)$ using congruence by $J$–unitary matrices, either implicitly (one-sided algorithm) or explicitly (two-sided algorithm).
One-sided or two-sided that is the question
In practice answer is very simple – one-sided algorithm.

One-sided algorithm is:

- more accurate (for at least one-two decimal digits, even if we carefully compute ‘annihilated’ elements in two-sided algorithm), does not require special care of the hyperbolic tangent,
- more than twice faster if vectorized routines are used (either compiler vectorization or BLAS from Math Kernel Library).
One-sided or two-sided that is the question

In practice answer is very simple – **one-sided** algorithm.

**One-sided algorithm is:**

- **more accurate** (for at least one-two decimal digits, even if we carefully compute ‘annihilated’ elements in two-sided algorithm), does not require special care of the hyperbolic tangent,

- **more than twice faster** if vectorized routines are used (either compiler vectorization or BLAS from Math Kernel Library).
One-sided or two-sided that is the question

In practice answer is very simple – one-sided algorithm.

One-sided algorithm is:

- **more accurate** (for at least one-two decimal digits, even if we carefully compute ‘annihilated’ elements in two-sided algorithm), does not require special care of the hyperbolic tangent,

- **more than twice faster** if vectorized routines are used (either compiler vectorization or BLAS from Math Kernel Library).
One-sided or two-sided that is the question

In practice answer is very simple – one-sided algorithm.

One-sided algorithm is:

- more accurate (for at least one-two decimal digits, even if we carefully compute ‘annihilated’ elements in two-sided algorithm), does not require special care of the hyperbolic tangent,
- more than twice faster if vectorized routines are used (either compiler vectorization or BLAS from Math Kernel Library).
One-sided or two-sided that is the question

In practice answer is very simple – one-sided algorithm.

One-sided algorithm is:

- more accurate (for at least one-two decimal digits, even if we carefully compute ‘annihilated’ elements in two-sided algorithm), does not require special care of the hyperbolic tangent,
- more than twice faster if vectorized routines are used (either compiler vectorization or BLAS from Math Kernel Library).
One-sided or two-sided that is the question

In practice answer is very simple – one-sided algorithm.

One-sided algorithm is:

- **more accurate** (for at least one-two decimal digits, even if we carefully compute ‘annihilated’ elements in two-sided algorithm), does not require special care of the hyperbolic tangent,
- **more than twice faster** if vectorized routines are used (either compiler vectorization or BLAS from Math Kernel Library).
$J$ should be $I ⊕ (−I)$

Using permutational congruence on $J$ and $G^*G$ we can obtain:

$$J = I ⊕ (−I).$$

**Advantages:**

▶ more than 10% faster convergence,
▶ sorting columns of $G$ (respecting the signs in $J$, and column norms of $G$) – additional 5% in speedup is obtained.
$J$ should be $I \oplus (-I)$

Using permutational congruence on $J$ and $G^*G$ we can obtain:

$$J = I \oplus (-I).$$

**Advantages:**

- more than 10% faster convergence,
- sorting columns of $G$ (respecting the signs in $J$, and column norms of $G$) – additional 5% in speedup is obtained.
\[ J \text{ should be } I \oplus (-I) \]

Using permutational congruence on \( J \) and \( G^*G \) we can obtain:

\[ J = I \oplus (-I). \]

**Advantages:**

- more than 10\% faster convergence,
- sorting columns of \( G \) (respecting the signs in \( J \), and column norms of \( G \)) – additional 5\% in speedup is obtained.
QR factorization?!

In the definite case we can significantly diagonalize $H$ by applying two QR factorizations (inner and outer) on $G$. What if $H$ is indefinite?

### Advantages/shortcomings:

- outer factorization should be indefinite,
- block triangular structure of $G$ is not really used,
- norm-wise pivoting (de Rijk) of columns of $G$ cannot be used in full potential, otherwise quadratic convergence can fail,
- good pivoting strategy is difficult to find – work in progress.
In the definite case we can significantly diagonalize $H$ by applying two QR factorizations (inner and outer) on $G$. What if $H$ is indefinite?

### Advantages/shortcomings:

- **outer factorization should be indefinite,**
- block triangular structure of $G$ is not really used,
- norm-wise pivoting (de Rijk) of columns of $G$ cannot be used in full potential, otherwise quadratic convergence can fail,
- good **pivoting** strategy is difficult to find – work in progress.
QR factorization?!

In the definite case we can significantly diagonalize $H$ by applying two QR factorizations (inner and outer) on $G$. What if $H$ is indefinite?

<table>
<thead>
<tr>
<th>Advantages/shortcomings:</th>
</tr>
</thead>
<tbody>
<tr>
<td>▶ outer factorization should be indefinite,</td>
</tr>
<tr>
<td>▶ block triangular structure of $G$ is not really used,</td>
</tr>
<tr>
<td>▶ norm-wise pivoting (de Rijk) of columns of $G$ cannot be used in full potential, otherwise quadratic convergence can fail,</td>
</tr>
<tr>
<td>▶ good <strong>pivoting</strong> strategy is difficult to find – work in progress.</td>
</tr>
</tbody>
</table>
QR factorization?!

In the definite case we can significantly diagonalize $H$ by applying two QR factorizations (inner and outer) on $G$. What if $H$ is indefinite?

**Advantages/shortcomings:**

- outer factorization should be indefinite,
- block triangular structure of $G$ is not really used,
- norm-wise pivoting (de Rijk) of columns of $G$ cannot be used in full potential, otherwise quadratic convergence can fail,
- good pivoting strategy is difficult to find – work in progress.
QR factorization?!

In the definite case we can significantly diagonalize $H$ by applying two QR factorizations (inner and outer) on $G$. What if $H$ is indefinite?

Advantages/shortcomings:

- outer factorization should be indefinite,
- block triangular structure of $G$ is not really used,
- norm-wise pivoting (de Rijk) of columns of $G$ cannot be used in full potential, otherwise quadratic convergence can fail,
- good pivoting strategy is difficult to find – work in progress.
Block partitions and subdivisions

Note:
It is advantageous to describe algorithms as two-sided, and in implementation use one-sided.

Block algorithms operate on pair \((A, J) := (G^*G, J)\),

- \(A\) is divided in 4 blocks – following natural partition of \(J\):

\[
A = \begin{bmatrix}
A_{11} & A_{12} \\
A^*_{12} & A_{22}
\end{bmatrix}, \quad J = \text{diag}(I_m, -I_{n-m}).
\]

- natural assumption: \(a_{11} \geq a_{22} \geq \cdots \geq a_{mm},\)
  \(a_{m+1,m+1} \leq a_{m+2,m+2} \leq \cdots \leq a_{nn}.\)
Block partitions and subdivisions

Note:
It is advantageous to describe algorithms as two-sided, and in implementation use one-sided.

Block algorithms operate on pair \((A, J) := (G^*G, J)\),

- A is divided in 4 blocks – following natural partition of J:

\[
A = \begin{bmatrix}
A_{11} & A_{12} \\
A_{12}^* & A_{22}
\end{bmatrix}, \quad J = \text{diag}(I_m, -I_{n-m}).
\]

- natural assumption: \(a_{11} \geq a_{22} \geq \cdots \geq a_{mm}, \quad a_{m+1,m+1} \leq a_{m+2,m+2} \leq \cdots \leq a_{nn}\).
Block partitions and subdivisions

Note:

It is advantageous to describe algorithms as two-sided, and in implementation use one-sided.

Block algorithms operate on pair \((A, J) := (G^*G, J)\),

- \(A\) is divided in 4 blocks – following natural partition of \(J\):

\[
A = A_{11} \ A_{12} \\
A_{12}^* \ A_{22}
\]

\(J = \text{diag}(I_m, -I_{n-m})\).

- natural assumption: \(a_{11} \geq a_{22} \geq \cdots \geq a_{mm},\)
\(a_{m+1,m+1} \leq a_{m+2,m+2} \leq \cdots \leq a_{nn}\).
Each $A_{k\ell}$ block is further divided into smaller blocks $A_{ij}$ (typically of order 8–128) which are used to define (block) pivot strategy:

$A_{11} = \begin{bmatrix} A_{11} & \cdots & A_{1p} \\ \vdots & \ddots & \vdots \\ A_{1p}^* & \cdots & A_{pp} \end{bmatrix}$,

$A_{12} = \begin{bmatrix} A_{1,p+1} & \cdots & A_{1,p+q} \\ \vdots & \ddots & \vdots \\ A_{p,p+1} & \cdots & A_{p,p+q} \end{bmatrix}$,

$A_{22} = \begin{bmatrix} A_{p+1,p+1} & \cdots & A_{p+1,p+q} \\ \vdots & \ddots & \vdots \\ A_{p+1,p+q}^* & \cdots & A_{p+q,p+q} \end{bmatrix}$. 
One (block) step

Block algorithms operate on blocks. On level of pivot sub-matrices we have:

\[ A' = \begin{bmatrix} A'_{ii} & A'_{ij} \\ [A'_{ij}]^* & A'_{jj} \end{bmatrix} = \begin{bmatrix} V^*_i & V^*_j \\ V^*_i & V^*_j \end{bmatrix} \begin{bmatrix} A_{ii} & A_{ij} \\ [A_{ij}]^* & A_{jj} \end{bmatrix} \begin{bmatrix} V_i & V_j \\ V_i & V_j \end{bmatrix} = V^*AV. \]

▶ The purpose of one step is to make \( A' \) more diagonal then \( A \).

▶ We distinguish two possibilities:
  ▶ the norm of the off-diagonal block \( A_{ij} \) is only reduced, (this yields block–oriented type methods),
  ▶ the off-diagonal block is annihilated (this yields proper or full block methods).
One (block) step

Block algorithms operate on blocks. On level of pivot sub-matrices we have:

\[
A' = \begin{bmatrix}
A_{ii}' & A_{ij}' \\
[A_{ij}']^* & A_{jj}'
\end{bmatrix} = \begin{bmatrix}
V_{ii}^* & V_{ji}^* \\
V_{ij}^* & V_{jj}^*
\end{bmatrix} \begin{bmatrix}
A_{ii} & A_{ij} \\
A_{ij}^* & A_{jj}
\end{bmatrix} \begin{bmatrix}
V_{ii} & V_{ij} \\
V_{ji} & V_{jj}
\end{bmatrix} = V^*AV.
\]

- The purpose of one step is to make \(A'\) more diagonal than \(A\).
- We distinguish two possibilities:
  - the norm of the off-diagonal block \(A_{ij}\) is only reduced, (this yields block–oriented type methods),
  - the off-diagonal block is annihilated (this yields proper or full block methods).
One (block) step

Block algorithms operate on blocks. On level of pivot sub-matrices we have:

\[
A' = \begin{bmatrix}
A'_{ii} & A'_{ij} \\
[A'_{ij}]^* & A'_{jj}
\end{bmatrix}
= \begin{bmatrix}
V_{ii}^* & V_{ji}^* \\
V_{ij}^* & V_{jj}^*
\end{bmatrix}
\begin{bmatrix}
A_{ii} & A_{ij} \\
[A_{ij}]^* & A_{jj}
\end{bmatrix}
\begin{bmatrix}
V_{ii} & V_{ij} \\
V_{ji} & V_{jj}
\end{bmatrix}
= V^* A V.
\]

The purpose of one step is to make \( A' \) more diagonal then \( A \).

We distinguish two possibilities:

- the norm of the off-diagonal block \( A_{ij} \) is only reduced, (this yields block–oriented type methods),
- the off-diagonal block is annihilated (this yields proper or full block methods).
One (block) step

Block algorithms operate on blocks. On level of pivot sub-matrices we have:

\[
A' = \begin{bmatrix}
A'_{ii} & A'_{ij} \\
[A'_{ij}]^* & A'_{jj}
\end{bmatrix} = \begin{bmatrix}
V_{ii}^* & V_{ji}^* \\
V_{ij}^* & V_{jj}^*
\end{bmatrix} \begin{bmatrix}
A_{ii} & A_{ij} \\
[A_{ij}]^* & A_{jj}
\end{bmatrix} \begin{bmatrix}
V_{ii} & V_{ij} \\
V_{ji} & V_{jj}
\end{bmatrix} = V^* AV.
\]

- The purpose of one step is to make \( A' \) more diagonal then \( A \).
- We distinguish two possibilities:
  - the norm of the off-diagonal block \( A_{ij} \) is only reduced, (this yields block–oriented type methods),
  - the off-diagonal block is annihilated (this yields proper or full block methods).
We have two levels of (pivot) strategies:

- block (macro) level strategies,
- micro level strategies (inside each block).

On macro level each block can be operated just once in block–cyclic strategies, or fixed number of times in block quasi-cyclic strategies.
Block strategies

- We have two levels of (pivot) strategies:
  - block (macro) level strategies,
  - micro level strategies (inside each block).
- On macro level each block can be operated just once in block–cyclic strategies, or fixed number of times in block quasi-cyclic strategies.
Block strategies

- We have two levels of (pivot) strategies:
  - block (macro) level strategies,
  - micro level strategies (inside each block).

- On macro level each block can be operated just once in block–cyclic strategies, or fixed number of times in block quasi-cyclic strategies.
Block strategies

▶ We have two levels of (pivot) strategies:
  ▶ block (macro) level strategies,
  ▶ micro level strategies (inside each block).

▶ On macro level each block can be operated just once in block–cyclic strategies, or fixed number of times in block quasi-cyclic strategies.
Cyclic block strategies

- Wave-front orderings on macro level (equivalent to column-cyclic strategy) and serial on micro level.

Example: block row–row cyclic:
Cyclic block strategies

- **Wave-front** orderings on macro level (equivalent to column-cyclic strategy) and serial on micro level.

Example: block row–row cyclic:
Cyclic block strategies

- Wave-front orderings on macro level (equivalent to column-cyclic strategy) and serial on micro level.

Example: block row–row cyclic:
Cyclic block strategies

- Wave-front orderings on macro level (equivalent to column-cyclic strategy) and serial on micro level.

Example: block row–row cyclic:
Cyclic block strategies

- **Wave-front** orderings on macro level (equivalent to column-cyclic strategy) and serial on micro level.

Example: block row–row cyclic:
Cyclic block strategies

- **Wave-front** orderings on macro level (equivalent to column-cyclic strategy) and serial on micro level.

Example: block row–row cyclic:
Cyclic block strategies

- Wave-front orderings on macro level (equivalent to column-cyclic strategy) and serial on micro level.

Example: block row–row cyclic:
Cyclic block strategies

- Wave-front orderings on macro level (equivalent to column-cyclic strategy) and serial on micro level.

Example: block row–row cyclic:
Cyclic block strategies

- **Wave-front** orderings on macro level (equivalent to column-cyclic strategy) and serial on micro level.

Example: block row–row cyclic:

![Block row–row cyclic diagram]
Cyclic block strategies

- **Wave-front** orderings on macro level (equivalent to column-cyclic strategy) and serial on micro level.

Example: block row–row cyclic:
Quasi-cyclic strategies

The most appropriate:

- Mascarenhas ordering:

ordering introduced by Drmač and Veselić:
Quasi-cyclic strategies

The most appropriate:

▶ Mascarenhas ordering:

ordering introduced by Drmač and Veselić:
Quasi-cyclic strategies

The most appropriate:

- Mascarenhas ordering:

- ordering introduced by Drmač and Veselić:
Quasi-cyclic strategies

The most appropriate:

- Mascarenhas ordering:

- ordering introduced by Drmač and Veselić:
Quasi-cyclic strategies

The most appropriate:

- Mascarenhas ordering:

- ordering introduced by Drmač and Veselić:
Quasi-cyclic strategies

The most appropriate:

- Mascarenhas ordering:

- ordering introduced by Drmač and Veselić:
Quasi-cyclic strategies

The most appropriate:

- Mascarenhas ordering:

- ordering introduced by Drmač and Veselić:
Quasi-cyclic strategies

The most appropriate:

▶ Mascarenhas ordering:

ordering introduced by Drmač and Veselić:
Quasi-cyclic strategies

The most appropriate:

- **Mascarenhas ordering:**

- **ordering introduced by Drmač and Veselić:**
Quasi-cyclic strategies

The most appropriate:

- Mascarenhas ordering:

ordering introduced by Drmač and Veselić:
Quasi-cyclic strategies

The most appropriate:

- **Mascarenhas ordering:**

- **ordering introduced by Drmač and Veselić:**
Quasi-cyclic strategies

The most appropriate:

- Mascarenhas ordering:

- ordering introduced by Drmač and Veselić:
Quasi-cyclic strategies

The most appropriate:

- Mascarenhas ordering:

- ordering introduced by Drmač and Veselić:
Quasi-cyclic strategies

The most appropriate:

- Mascarenhas ordering:

- ordering introduced by Drmač and Veselić:
Quasi-cyclic strategies

The most appropriate:

- Mascarenhas ordering:

- ordering introduced by Drmač and Veselić:
Quasi-cyclic strategies

The most appropriate:

- Mascarenhas ordering:

- ordering introduced by Drmač and Veselić:
Quasi-cyclic strategies

The most appropriate:

- Mascarenhas ordering:

- ordering introduced by Drmač and Veselić:
Quasi-cyclic strategies

The most appropriate:

- Mascarenhas ordering:

- ordering introduced by Drmač and Veselić:
Quasi-cyclic strategies

The most appropriate:

- Mascarenhas ordering:

- ordering introduced by Drmač and Veselić:
Quasi-cyclic strategies

The most appropriate:

- Mascarenhas ordering:

- ordering introduced by Drmač and Veselić:
Quasi-cyclic strategies

The most appropriate:

- Mascarenhas ordering:

- ordering introduced by Drmač and Veselić:
Quasi-cyclic strategies

The most appropriate:

- Mascarenhas ordering:

- ordering introduced by Drmač and Veselić:
Quasi-cyclic strategies

The most appropriate:

- Mascarenhas ordering:

![Mascarenhas ordering diagram]

- ordering introduced by Drmač and Veselić:

![Drmač and Veselić ordering diagram]
Quasi-cyclic strategies

The most appropriate:

- Mascarenhas ordering:

- ordering introduced by Drmač and Veselić:
Quasi-cyclic strategies

The most appropriate:

- Mascarenhas ordering:

- ordering introduced by Drmač and Veselić:
Accuracy and convergence

Accuracy and convergence of block–oriented strategies:

- **wavefront strategies:**
  - global convergence can easily be proved,
  - accuracy – obvious from non-blocked algorithm,
  - speed – very good for $A \in \mathbb{R}^{n \times n}$, $n \leq 3500$ (compared to full block algorithm),

- **Mascarenhas strategy:**
  - more difficult to prove global convergence,
  - accuracy results – as above (Slapničar, LAA, 2002),
  - cubically (per quasi sweep) convergent strategy,

- **Drmač–Veselić strategy:**
  - global convergence proved (Hari) – similar to definite method,
  - accuracy as above,
  - similar speed as the cyclic strategy.
Accuracy and convergence

Accuracy and convergence of block–oriented strategies:

- **wavefront strategies:**
  - global convergence can easily be proved,
  - accuracy – obvious from non-blocked algorithm,
  - speed – very good for $A \in \mathbb{R}^{n \times n}$, $n \leq 3500$ (compared to full block algorithm),

- **Mascarenhas strategy:**
  - more difficult to prove global convergence,
  - accuracy results – as above (Slapničar, LAA, 2002),
  - cubically (per quasi sweep) convergent strategy,

- **Drmač–Veselić strategy:**
  - global convergence proved (Hari) – similar to definite method,
  - accuracy as above,
  - similar speed as the cyclic strategy.
Accuracy and convergence

Accuracy and convergence of block–oriented strategies:

▶ wavefront strategies:
  ▶ global convergence can easily be proved,
  ▶ accuracy – obvious from non-blocked algorithm,
  ▶ speed – very good for $A \in \mathbb{R}^{n \times n}$, $n \leq 3500$ (compared to full block algorithm),

▶ Mascarenhas strategy:
  ▶ more difficult to prove global convergence,
  ▶ accuracy results – as above (Slapničar, LAA, 2002),
  ▶ cubically (per quasi sweep) convergent strategy,

▶ Drmač–Veselić strategy:
  ▶ global convergence proved (Hari) – similar to definite method,
  ▶ accuracy as above,
  ▶ similar speed as the cyclic strategy.
Accuracy and convergence

Accuracy and convergence of block–oriented strategies:

- **wavefront strategies:**
  - global convergence can easily be proved,
  - accuracy – obvious from non-blocked algorithm,
  - speed – very good for $A \in \mathbb{R}^{n \times n}$, $n \leq 3500$ (compared to full block algorithm),

- **Mascarenhas strategy:**
  - more difficult to prove global convergence,
  - accuracy results – as above (Slapničar, LAA, 2002),
  - cubically (per quasi sweep) convergent strategy,

- **Drmač–Veselić strategy:**
  - global convergence proved (Hari) – similar to definite method,
  - accuracy as above,
  - similar speed as the cyclic strategy.
Accuracy and convergence

Accuracy and convergence of block–oriented strategies:

- wavefront strategies:
  - global convergence can easily be proved,
  - accuracy – obvious from non-blocked algorithm,
  - speed – very good for $A \in \mathbb{R}^{n \times n}$, $n \leq 3500$ (compared to full block algorithm),

- Mascarenhas strategy:
  - more difficult to prove global convergence,
  - accuracy results – as above (Slapničar, LAA, 2002),
  - cubically (per quasi sweep) convergent strategy,

- Drmač–Veselić strategy:
  - global convergence proved (Hari) – similar to definite method,
  - accuracy as above,
  - similar speed as the cyclic strategy.
Accuracy and convergence

Accuracy and convergence of block–oriented strategies:

- **wavefront strategies:**
  - global convergence can easily be proved,
  - accuracy – obvious from non-blocked algorithm,
  - speed – very good for $A \in \mathbb{R}^{n \times n}$, $n \leq 3500$ (compared to full block algorithm),

- **Mascarenhas strategy:**
  - more difficult to prove global convergence,
  - accuracy results – as above (Slapničar, LAA, 2002),
  - cubically (per quasi sweep) convergent strategy,

- **Drmač–Veselić strategy:**
  - global convergence proved (Hari) – similar to definite method,
  - accuracy as above,
  - similar speed as the cyclic strategy.
Accuracy and convergence

Accuracy and convergence of block-oriented strategies:

- **wavefront strategies:**
  - global convergence can easily be proved,
  - accuracy – obvious from non-blocked algorithm,
  - speed – very good for $A \in \mathbb{R}^{n \times n}$, $n \leq 3500$ (compared to full block algorithm),

- **Mascarenhas strategy:**
  - more difficult to prove global convergence,
  - accuracy results – as above (Slapničar, LAA, 2002),
  - cubically (per quasi sweep) convergent strategy,

- **Drmač–Veselić strategy:**
  - global convergence proved (Hari) – similar to definite method,
  - accuracy as above,
  - similar speed as the cyclic strategy.
Accuracy and convergence

Accuracy and convergence of block–oriented strategies:

- **Wavefront strategies:**
  - global convergence can easily be proved,
  - accuracy – obvious from non-blocked algorithm,
  - speed – very good for $A \in \mathbb{R}^{n \times n}$, $n \leq 3500$ (compared to full block algorithm),

- **Mascarenhas strategy:**
  - more difficult to prove global convergence,
  - accuracy results – as above (Slapničar, LAA, 2002),
  - cubically (per quasi sweep) convergent strategy,

- **Drmač–Veselić strategy:**
  - global convergence proved (Hari) – similar to definite method,
  - accuracy as above,
  - similar speed as the cyclic strategy.
Accuracy and convergence

Accuracy and convergence of block–oriented strategies:

- **wavefront strategies:**
  - global convergence can easily be proved,
  - accuracy – obvious from non-blocked algorithm,
  - speed – very good for $A \in \mathbb{R}^{n \times n}$, $n \leq 3500$ (compared to full block algorithm),

- **Mascarenhas strategy:**
  - more difficult to prove global convergence,
  - accuracy results – as above (Slapničar, LAA, 2002),
  - cubically (per quasi sweep) convergent strategy,

- **Drmač–Veselić strategy:**
  - global convergence proved (Hari) – similar to definite method,
  - accuracy as above,
  - similar speed as the cyclic strategy.
Accuracy and convergence

Accuracy and convergence of block-oriented strategies:

- **wavefront strategies:**
  - global convergence can easily be proved,
  - accuracy – obvious from non-blocked algorithm,
  - speed – very good for $A \in \mathbb{R}^{n \times n}$, $n \leq 3500$ (compared to full block algorithm),

- **Mascarenhas strategy:**
  - more difficult to prove global convergence,
  - accuracy results – as above (Slapničar, LAA, 2002),
  - cubically (per quasi sweep) convergent strategy,

- **Drmač–Veselić strategy:**
  - global convergence proved (Hari) – similar to definite method,
  - accuracy as above,
  - similar speed as the cyclic strategy.
Accuracy and convergence

Accuracy and convergence of block–oriented strategies:

- **wavefront strategies:**
  - global convergence can easily be proved,
  - accuracy – obvious from non-blocked algorithm,
  - speed – very good for $A \in \mathbb{R}^{n \times n}$, $n \leq 3500$ (compared to full block algorithm),

- **Mascarenhas strategy:**
  - more difficult to prove global convergence,
  - accuracy results – as above (Slapničar, LAA, 2002),
  - cubically (per quasi sweep) convergent strategy,

- **Drmač–Veselić strategy:**
  - global convergence proved (Hari) – similar to definite method,
  - accuracy as above,
  - similar speed as the cyclic strategy.
Accuracy and convergence

Accuracy and convergence of block–oriented strategies:

- **wavefront strategies:**
  - global convergence can easily be proved,
  - accuracy – obvious from non-blocked algorithm,
  - speed – very good for $A \in \mathbb{R}^{n \times n}$, $n \leq 3500$ (compared to full block algorithm),

- **Mascarenhas strategy:**
  - more difficult to prove global convergence,
  - accuracy results – as above (Slapničar, LAA, 2002),
  - cubically (per quasi sweep) convergent strategy,

- **Drmač–Veselić strategy:**
  - global convergence proved (Hari) – similar to definite method,
  - accuracy as above,
  - similar speed as the cyclic strategy.
Accuracy and convergence of block–oriented strategies:

- **wavefront strategies:**
  - global convergence can easily be proved,
  - accuracy – obvious from non-blocked algorithm,
  - speed – very good for $A \in \mathbb{R}^{n \times n}$, $n \leq 3500$ (compared to full block algorithm),

- **Mascarenhas strategy:**
  - more difficult to prove global convergence,
  - accuracy results – as above (Slapničar, LAA, 2002),
  - cubically (per quasi sweep) convergent strategy,

- **Drmač–Veselić strategy:**
  - global convergence proved (Hari) – similar to definite method,
  - accuracy as above,
  - similar speed as the cyclic strategy.
Full block strategies

- Full block strategies should annihilate off-diagonal blocks of $A$.
  - Annihilation of just $A_{ij}$ is linearly slow.
  - Solution: we should diagonalize the whole pivot sub-matrices $A$.
  - After certain number of steps (at worst after the full sweep) all diagonal blocks $A_{ii}$ will be diagonal

$$
\begin{bmatrix}
\Lambda'_{ii} & 0 \\
0 & \Lambda'_{jj}
\end{bmatrix} =
\begin{bmatrix}
V_{ii}^* & V_{ji}^* \\
V_{ij}^* & V_{jj}^*
\end{bmatrix}
\begin{bmatrix}
\Lambda_{ii} & A_{ij} \\
A_{ij}^* & \Lambda_{jj}
\end{bmatrix}
\begin{bmatrix}
V_{ii} & V_{ij} \\
V_{ji} & V_{jj}
\end{bmatrix}.
$$

- This suggests the following preprocessing step: diagonalization of all $A_{ii}$, the diagonal can be stored in a separate vector, and updated after each sweep.
Full block strategies

- Full block strategies should annihilate off-diagonal blocks of $A$.
- Annihilation of just $A_{ij}$ is linearly slow.
- Solution: we should **diagonalize** the whole pivot sub-matrices $A$.
- After certain number of steps (at worst after the full sweep) all diagonal blocks $A_{ii}$ will be diagonal

\[
\begin{bmatrix}
\Lambda'_{ii} & 0 \\
0 & \Lambda'_{jj}
\end{bmatrix}
= \begin{bmatrix}
V_{ii}^* & V_{ji}^* \\
V_{ij}^* & V_{jj}^*
\end{bmatrix}
\begin{bmatrix}
\Lambda_{ii} & A_{ij} \\
A_{ij}^* & \Lambda_{jj}
\end{bmatrix}
\begin{bmatrix}
V_{ii} & V_{ij} \\
V_{ji} & V_{jj}
\end{bmatrix}.
\]

- This suggests the following preprocessing step: diagonalization of all $A_{ii}$, the diagonal can be stored in a separate vector, and updated after each sweep.
Full block strategies

- Full block strategies should annihilate off-diagonal blocks of $A$.
- Annihilation of just $A_{ij}$ is linearly slow.
- Solution: we should **diagonalize** the whole pivot sub-matrices $A$.

  After certain number of steps (at worst after the full sweep) all diagonal blocks $A_{ii}$ will be diagonal

  $$
  \begin{bmatrix}
  \Lambda'_{ii} & 0 \\
  0 & \Lambda'_{jj}
  \end{bmatrix} = \begin{bmatrix}
  V_{ii}^* & V_{ji}^* \\
  V_{ij}^* & V_{jj}^*
  \end{bmatrix} \begin{bmatrix}
  \Lambda_{ii} & A_{ij} \\
  [A_{ij}]^* & \Lambda_{jj}
  \end{bmatrix} \begin{bmatrix}
  V_{ii} & V_{ij} \\
  V_{ji} & V_{jj}
  \end{bmatrix}.
  $$

- This suggests the following preprocessing step: diagonalization of all $A_{ii}$, the diagonal can be stored in a separate vector, and updated after each sweep.
Full block strategies

- Full block strategies should annihilate off-diagonal blocks of $A$.
- Annihilation of just $A_{ij}$ is linearly slow.
- Solution: we should diagonalize the whole pivot sub-matrices $A$.
- After certain number of steps (at worst after the full sweep) all diagonal blocks $A_{ii}$ will be diagonal

$$
\begin{bmatrix}
\Lambda'_{ii} & 0 \\
0 & \Lambda'_{jj}
\end{bmatrix} = 
\begin{bmatrix}
V_{ii}^* & V_{ji}^* \\
V_{ij}^* & V_{jj}^*
\end{bmatrix}
\begin{bmatrix}
\Lambda_{ii} & A_{ij} \\
[A_{ij}]^* & \Lambda_{jj}
\end{bmatrix}
\begin{bmatrix}
V_{ii} & V_{ij} \\
V_{ji} & V_{jj}
\end{bmatrix}.
$$

- This suggests the following preprocessing step: diagonalization of all $A_{ii}$, the diagonal can be stored in a separate vector, and updated after each sweep.
Full block strategies

- Full block strategies should annihilate off-diagonal blocks of $A$.
- Annihilation of just $A_{ij}$ is linearly slow.
- Solution: we should diagonalize the whole pivot sub-matrices $A$.
- After certain number of steps (at worst after the full sweep) all diagonal blocks $A_{ii}$ will be diagonal

\[
\begin{bmatrix}
\Lambda'_{ii} & 0 \\
0 & \Lambda'_{jj}
\end{bmatrix} = 
\begin{bmatrix}
V_{ii}^* & V_{ji}^* \\
V_{ij}^* & V_{jj}^*
\end{bmatrix} 
\begin{bmatrix}
\Lambda_{ii} & A_{ij} \\
[A_{ij}]^* & \Lambda_{jj}
\end{bmatrix} 
\begin{bmatrix}
V_{ii} & V_{ij} \\
V_{ji} & V_{jj}
\end{bmatrix}.
\]

- This suggests the following preprocessing step: diagonalization of all $A_{ii}$, the diagonal can be stored in a separate vector, and updated after each sweep.
Useful relations for convergence analysis

We have analyzed behaviour of the off-diagonal norm $\text{Off}(A)$ during one

- unitary step (pivot indices $1 \leq i < j \leq p$ or $p + 1 \leq i < j \leq p + q$),
- $J$-unitary step (pivot indices $1 \leq i \leq p < j \leq p + q$).

We have:

- measure $\Theta^2(A) = 2\|A_{12}\|_F^2 - \text{Off}^2(A_{11}) - \text{Off}^2(A_{22})$
  is decreased (increased) during $J$-unitary (unitary) step by the quantity $2\|A_{ij}\|^2$,
- neither $\Theta(A)$, nor $\text{Off}(A)$ is monotone,
- trace of $A^{(k)}$ is non-increasing, and therefore convergent sequence, i.e.,
  $$\|A^{(k)}\|_2 \leq \text{tr}(A^{(k)}) \leq \text{tr}(A).$$
Useful relations for convergence analysis

We have analyzed behaviour of the off-diagonal norm $\text{Off}(A)$ during one

- unitary step (pivot indices $1 \leq i < j \leq p$ or $p + 1 \leq i < j \leq p + q$),
- $J$-unitary step (pivot indices $1 \leq i \leq p < j \leq p + q$).

We have:

- measure $\Theta^2(A) = 2\|A_{12}\|_F^2 - \text{Off}^2(A_{11}) - \text{Off}^2(A_{22})$ is decreased (increased) during $J$-unitary (unitary) step by the quantity $2\|A_{ij}\|^2$,
- neither $\Theta(A)$, nor $\text{Off}(A)$ is monotone,
- trace of $A^{(k)}$ is non-increasing, and therefore convergent sequence, i.e.,
  $$\|A^{(k)}\|_2 \leq \text{tr}(A^{(k)}) \leq \text{tr}(A).$$
Useful relations for convergence analysis

We have analyzed behaviour of the off-diagonal norm $\text{Off}(A)$ during one

- unitary step (pivot indices $1 \leq i < j \leq p$ or $p + 1 \leq i < j \leq p + q$),
- $J$-unitary step (pivot indices $1 \leq i \leq p < j \leq p + q$).

We have:

- measure $\Theta^2(A) = 2\|A_{12}\|_F^2 - \text{Off}^2(A_{11}) - \text{Off}^2(A_{22})$ is decreased (increased) during $J$-unitary (unitary) step by the quantity $2\|A_{i,j}\|^2$,
- neither $\Theta(A)$, nor $\text{Off}(A)$ is monotone,
- trace of $A^{(k)}$ is non-increasing, and therefore convergent sequence, i.e.,
  \[ \|A^{(k)}\|_2 \leq \text{tr}(A^{(k)}) \leq \text{tr}(A). \]
Accuracy of full block method

We analyze accuracy of one step of the full block method. To keep it simple, we omit indices.

Assumptions:

- the initial factor $G$ is preprocessed by QR factorization so that the obtained $G' (= R)$ has small row-wise (scaled from left) condition;
- denote by
  $$\sigma_1 \geq \cdots \geq \sigma_m, \quad \sigma_{m+1} \leq \cdots \leq \sigma_n,$$

  the hyperbolic singular values of $G$ and $\tilde{G}'$, resp.,
  $$\tilde{G}' = GW(I + EW) = GW + F.$$

- $G$ and $G' = GW$ have the same hyp. singular values.
Accuracy of full block method

We analyze accuracy of one step of the full block method. To keep it simple, we omit indices.

Assumptions:

- the initial factor $G$ is preprocessed by QR factorization so that the obtained $G$ ($= R$) has small row-wise (scaled from left) condition;
- denote by 
  \[ \sigma_1 \geq \cdots \geq \sigma_m, \quad \sigma_{m+1} \leq \cdots \leq \sigma_n, \quad \text{and} \]
  \[ \tilde{\sigma}_1 \geq \cdots \geq \tilde{\sigma}_m, \quad \tilde{\sigma}_{m+1} \leq \cdots \leq \tilde{\sigma}_n. \]
  the hyperbolic singular values of $G$ and $\tilde{G}'$, resp., 
  \[ \tilde{G}' = GW(I + EW) = GW + F. \]
- $G$ and $G' = GW$ have the same hyp. singular values.
Accuracy of full block method

We analyze accuracy of one step of the full block method. To keep it simple, we omit indices.

Assumptions:

- the initial factor $G$ is preprocessed by QR factorization so that the obtained $G$ ($= R$) has small row-wise (scaled from left) condition;

- denote by
  \[ \sigma_1 \geq \cdots \geq \sigma_m, \quad \sigma_{m+1} \leq \cdots \leq \sigma_n, \quad \text{and} \]
  \[ \tilde{\sigma}_1 \geq \cdots \geq \tilde{\sigma}_m, \quad \tilde{\sigma}_{m+1} \leq \cdots \leq \tilde{\sigma}_n. \]

the hyperbolic singular values of $G$ and $\tilde{G}'$, resp.,

\[ \tilde{G}' = GW(I + EW) = GW + F. \]

- $G$ and $G' = GW$ have the same hyp. singular values.
Accuracy of full block method

We analyze accuracy of one step of the full block method. To keep it simple, we omit indices.

Assumptions:

- the initial factor $G$ is preprocessed by QR factorization so that the obtained $G (= R)$ has small row-wise (scaled from left) condition;
- denote by
  \[
  \sigma_1 \geq \cdots \geq \sigma_m, \quad \sigma_{m+1} \leq \cdots \leq \sigma_n, \quad \text{and}
  \tilde{\sigma}_1 \geq \cdots \geq \tilde{\sigma}_m, \quad \tilde{\sigma}_{m+1} \leq \cdots \leq \tilde{\sigma}_n.
  \]
- the hyperbolic singular values of $G$ and $\tilde{G}'$, resp.,
  \[
  \tilde{G}' = GW(I + E_W) = GW + F.
  \]
- $G$ and $G' = GW$ have the same hyp. singular values.
Accuracy of full block method

Directly from:

**Theorem (Tm 5, Slapničar and Truhar, LAA, 2002)**

Let \( G' = \Delta B \) be nonsingular, \( \Delta \) diagonal, \( F = \Delta \delta B \), \( \tilde{G}' = G' + F \), and let \( \beta = \| B^{-1} \delta B \| \). If \( 2\beta + \beta^2 < 1 \), then for the hyperbolic singular values of \( G' \) and \( \tilde{G}' \) holds

\[
1 - \gamma \leq \sqrt{1 - \gamma} \leq \frac{\tilde{\sigma}_i}{\sigma_i} \leq \sqrt{1 + \gamma} \leq 1 + \frac{1}{2} \gamma,
\]

\( \gamma = \beta(2 + \beta)\|V\|^2 \), provided that \( \gamma < 1 \). Matrix \( V \) is \( J \)-unitary from the hyperbolic SVD of \( G' \).

we obtain the following result: (\( \xi \) is a slowly growing function of \( n \) times \( \max_i n_i \)):

\[
\max_{1 \leq i \leq n} \frac{\left| \tilde{\sigma}_i - \sigma_i \right|}{\sigma_i} \leq \alpha \xi \kappa(B) \kappa(W) \varepsilon + O(\varepsilon^2), \quad \alpha = 2.5 \kappa(V).
\]
Useful tricks in computation

In each step of the full block algorithm, one has to compute the off-diagonal block $A_{ij}$.

- Denote by $G_i$ and $G_j$ the corresponding pivot block columns of $G$, i.e.,

$$A = \begin{bmatrix} \Lambda_{ii} & A_{ij} \\ [A_{ij}]^* & \Lambda_{jj} \end{bmatrix} = \begin{bmatrix} G_i^* \\ G_j^* \end{bmatrix} \begin{bmatrix} G_i & G_j \end{bmatrix}.$$

- $A_{ij}$ can be computed by xGEMM subroutine.
Useful tricks in computation

In each step of the full block algorithm, one has to compute the off-diagonal block $A_{ij}$.

- Denote by $G_i$ and $G_j$ the corresponding pivot block columns of $G$, i.e.,

$$A = \begin{bmatrix} \Lambda_{ii} & A_{ij} \\ [A_{ij}]^* & \Lambda_{jj} \end{bmatrix} = \begin{bmatrix} G_i^* \\ G_j^* \end{bmatrix} \begin{bmatrix} G_i & G_j \end{bmatrix}.$$

- $A_{ij}$ can be computed by xGEMM subroutine.
Diagonalization

In order to diagonalize $A$, we can choose one procedure from the list:

D1. We can make copy of the block $[G_i, G_j]$, make QR factorization $[G_i, G_j] = Q_{ij}R_{ij}$ and then apply the one-sided (non-blocked) $J$-Hermitian Jacobi to $R_{ij}$.

D2. If $G$ is not catastrophically scaled, we can form the upper triangle of matrix $A$, and then apply the two-sided $J$-Hermitian Jacobi algorithm.

D3. Form $A$, then factorize it by the Cholesky factorization, and then apply the one-sided non-blocked $J$-Hermitian Jacobi to the Cholesky factor.

Our tests show that D3 variant is the fastest one.
Diagonalization

In order to diagonalize $A$, we can choose one procedure from the list:

**D1.** We can make copy of the block $[G_i, G_j]$, make QR factorization $[G_i, G_j] = Q_{ij} R_{ij}$ and then apply the one-sided (non-blocked) $J$-Hermitian Jacobi to $R_{ij}$.

**D2.** If $G$ is not catastrophically scaled, we can form the upper triangle of matrix $A$, and then apply the two-sided $J$-Hermitian Jacobi algorithm.

**D3.** Form $A$, then factorize it by the Cholesky factorization, and then apply the one-sided non-blocked $J$-Hermitian Jacobi to the Cholesky factor.

Our tests show that D3 variant is the fastest one.
Diagonalization

In order to diagonalize $A$, we can choose one procedure from the list:

**D1.** We can make copy of the block $[G_i, G_j]$, make QR factorization $[G_i, G_j] = Q_{ij} R_{ij}$ and then apply the one-sided (non-blocked) $J$-Hermitian Jacobi to $R_{ij}$.

**D2.** If $G$ is not catastrophically scaled, we can form the upper triangle of matrix $A$, and then apply the two-sided $J$-Hermitian Jacobi algorithm.

**D3.** Form $A$, then factorize it by the Cholesky factorization, and then apply the one-sided non-blocked $J$-Hermitian Jacobi to the Cholesky factor.

Our tests show that D3 variant is the fastest one.
Diagonalization

In order to diagonalize $A$, we can choose one procedure from the list:

**D1.** We can make copy of the block $[G_i, G_j]$, make QR factorization $[G_i, G_j] = Q_{ij}R_{ij}$ and then apply the one-sided (non-blocked) $J$-Hermitian Jacobi to $R_{ij}$.

**D2.** If $G$ is not catastrophically scaled, we can form the upper triangle of matrix $A$, and then apply the two-sided $J$-Hermitian Jacobi algorithm.

**D3.** Form $A$, then factorize it by the Cholesky factorization, and then apply the one-sided non-blocked $J$-Hermitian Jacobi to the Cholesky factor.

Our tests show that D3 variant is the fastest one.
Diagonalization

In order to diagonalize $A$, we can choose one procedure from the list:

**D1.** We can make copy of the block $[G_i,G_j]$, make QR factorization $[G_i,G_j] = Q_{ij}R_{ij}$ and then apply the one-sided (non-blocked) $J$-Hermitian Jacobi to $R_{ij}$.

**D2.** If $G$ is not catastrophically scaled, we can form the upper triangle of matrix $A$, and then apply the two-sided $J$-Hermitian Jacobi algorithm.

**D3.** Form $A$, then factorize it by the Cholesky factorization, and then apply the one-sided non-blocked $J$-Hermitian Jacobi to the Cholesky factor.

Our tests show that D3 variant is the fastest one.
Accumulation of transformations

After the diagonalization of $A$ we have at hand $J$-unitary transformation $V_{ij}$. Or, after computing HSVD of the Cholesky factor $R_{ij}$ of $A$, we have

$$R_{ij} = U_{ij} \Sigma_{ij} V_{ij}^{-1},$$

where $U_{ij} \Sigma$ is the final iteration. The matrix $V_{ij}$ is needed to update the pivot block-columns $[G_i, G_j]$. To compute $V_{ij}$, we can:

V1. accumulate used (unitary and hyperbolic) rotations,
V2. solve linear system $R_{ij} V_{ij} = U_{ij} \Sigma_{ij}$ for $V_{ij}$,
V3. use matrix multiplication $R_{ij}^{-1} U_{ij} \Sigma_{ij}$.

The best way is V1.
Accumulation of transformations

After the diagonalization of $A$ we have at hand $J$-unitary transformation $V_{ij}$. Or, after computing HSVD of the Cholesky factor $R_{ij}$ of $A$, we have

$$R_{ij} = U_{ij} \Sigma_{ij} V_{ij}^{-1},$$

where $U_{ij} \Sigma$ is the final iteration. The matrix $V_{ij}$ is needed to update the pivot block-columns $[G_i, G_j]$. To compute $V_{ij}$, we can:

- **V1.** accumulate used (unitary and hyperbolic) rotations,
- **V2.** solve linear system $R_{ij} V_{ij} = U_{ij} \Sigma_{ij}$ for $V_{ij}$,
- **V3.** use matrix multiplication $R_{ij}^{-1} U_{ij} \Sigma_{ij}$.

The best way is V1.
Accumulation of transformations

After the diagonalization of $A$ we have at hand $J$-unitary transformation $V_{ij}$. Or, after computing HSVD of the Cholesky factor $R_{ij}$ of $A$, we have

$$R_{ij} = U_{ij} \Sigma_{ij} V_{ij}^{-1},$$

where $U_{ij} \Sigma$ is the final iteration. The matrix $V_{ij}$ is needed to update the pivot block-columns $[G_i, G_j]$. To compute $V_{ij}$, we can:

V1. accumulate used (unitary and hyperbolic) rotations,
V2. solve linear system $R_{ij} V_{ij} = U_{ij} \Sigma_{ij}$ for $V_{ij}$,
V3. use matrix multiplication $R_{ij}^{-1} U_{ij} \Sigma_{ij}$.

The best way is V1.
Accumulation of transformations

After the diagonalization of $A$ we have at hand $J$-unitary transformation $V_{ij}$. Or, after computing HSVD of the Cholesky factor $R_{ij}$ of $A$, we have

$$R_{ij} = U_{ij} \Sigma_{ij} V_{ij}^{-1},$$

where $U_{ij} \Sigma$ is the final iteration. The matrix $V_{ij}$ is needed to update the pivot block-columns $[G_i, G_j]$. To compute $V_{ij}$, we can:

**V1.** accumulate used (unitary and hyperbolic) rotations,
**V2.** solve linear system $R_{ij} V_{ij} = U_{ij} \Sigma_{ij}$ for $V_{ij}$,
**V3.** use matrix multiplication $R_{ij}^{-1} U_{ij} \Sigma_{ij}$.

The best way is V1.
Accumulation of transformations

After the diagonalization of $A$ we have at hand $J$-unitary transformation $V_{ij}$. Or, after computing HSVD of the Cholesky factor $R_{ij}$ of $A$, we have

$$R_{ij} = U_{ij} \Sigma_{ij} V_{ij}^{-1},$$

where $U_{ij} \Sigma$ is the final iteration. The matrix $V_{ij}$ is needed to update the pivot block-columns $[G_i, G_j]$. To compute $V_{ij}$, we can:

**V1.** accumulate used (unitary and hyperbolic) rotations,

**V2.** solve linear system $R_{ij} V_{ij} = U_{ij} \Sigma_{ij}$ for $V_{ij}$,

**V3.** use matrix multiplication $R_{ij}^{-1} U_{ij} \Sigma_{ij}$.

The best way is V1.
Multiplication $R_{ij}V_{ij}$

To reduce copy of the pivot block-columns $[G_i, G_j]$ during multiplication:

- we use 2 block-vectors with dimension equal to dimension of max $G_i$ as workspace,
- during process, block-vectors are permuted,
- as a post-processing, block-vectors are re-permuted to their original positions.
Multiplication $R_{ij}V_{ij}$

To reduce copy of the pivot block-columns $[G_i, G_j]$ during multiplication:

- we use 2 block-vectors with dimension equal to dimension of max $G_i$ as workspace,
- during process, block-vectors are permuted,
- as a post-processing, block-vectors are re-permuted to their original positions.
Multiplication $R_{ij}V_{ij}$

To reduce copy of the pivot block-columns $[G_i, G_j]$ during multiplication:

- we use 2 block-vectors with dimension equal to dimension of max $G_i$ as workspace,
- during process, block-vectors are permuted,
- as a post-processing, block-vectors are re-permuted to their original positions.
Test results

\( N = 1000 \)

- blue – block-oriented, \( V \) accumulated,
- grey – full block, \( V \) obtained by \( V3 \),
- red – full block, \( V \) obtained by \( V1 \):

![Graph showing speedup in % vs. block size](image-url)

- Block Jacobi
- V. Hari, 2. (S. Singer)
- Motivation
- J–Jacobi Basics
- Block J–Jacobi algorithm
  - Block-oriented vs. full block strategies
  - Block-oriented block strategies
  - Full block strategies
- Numerical testing
- Work in progress
- Bibliography
Test results

\( N = 1500 \)

- blue – block-oriented, \( V \) accumulated,
- grey – full block, \( V \) obtained by \( V3 \),
- red – full block, \( V \) obtained by \( V1 \):

![Graph showing speedup in percentage against block size for different strategies.](image-url)
Test results

\(N = 2000\)

- blue – block-oriented, \(V\) accumulated,
- grey – full block, \(V\) obtained by \(V3\),
- red – full block, \(V\) obtained by \(V1\):
N = 2500

- blue – block-oriented, \( V \) accumulated,
- grey – full block, \( V \) obtained by \( V_3 \),
- red – full block, \( V \) obtained by \( V_1 \):

![Graph showing speedup in % vs. block size](image)
Test results

\( N = 3000 \)

- blue – block-oriented, \( V \) accumulated,
- grey – full block, \( V \) obtained by \( V3 \),
- red – full block, \( V \) obtained by \( V1 \):
Test results

N = 3500

- blue – block-oriented, V accumulated,
- grey – full block, V obtained by V3,
- red – full block, V obtained by V1:

![Graph showing speedup in % vs. block size]
Test results

\[ N = 4000 \]

- blue – block-oriented, \( V \) accumulated,
- grey – full block, \( V \) obtained by \( V_3 \),
- red – full block, \( V \) obtained by \( V_1 \):

![Speedup vs. Block Size Graph]

- Speedup in %
- Block size
Work in progress

- Efficient **column sorting** in block algorithms.
- Efficient **indefinite QR factorization** with diagonalization effect.
- Usage of the **indefinite CS decomposition** instead of xGEMM.
Work in progress

- Efficient **column sorting** in block algorithms.
- Efficient **indefinite QR factorization** with diagonalization effect.
- Usage of the **indefinite CS decomposition** instead of xGEMM.
Work in progress

- Efficient *column sorting* in block algorithms.
- Efficient *indefinite QR factorization* with diagonalization effect.
- Usage of the *indefinite CS decomposition* instead of xGEMM.
Bibliography

V. Hari,
Convergence of a block-oriented quasi-cyclic Jacobi method,
accepted for publication in SIMAX.

I. Slapničar,
Highly accurate symmetric eigenvalue decomposition and hyperbolic SVD,

I. Slapničar, N. Truhar,
Relative perturbation theory for hyperbolic singular value problem,
Bibliography

V. Hari,
Convergence of a block-oriented quasi-cyclic Jacobi method,
accepted for publication in SIMAX.

I. Slapničar,
Highly accurate symmetric eigenvalue decomposition and hyperbolic SVD,

I. Slapničar, N. Truhar,
Relative perturbation theory for hyperbolic singular value problem,
Bibliography

V. Hari,
Convergence of a block-oriented quasi-cyclic Jacobi method,
accepted for publication in *SIMAX*.

I. Slapničar,
Highly accurate symmetric eigenvalue decomposition and hyperbolic SVD,

I. Slapničar, N. Truhar,
Relative perturbation theory for hyperbolic singular value problem,