The Complex Cholesky-Jacobi Algorithm for PGEP

Vjeran Hari

Faculty of Science, Department of Mathematics, University of Zagreb
hari@math.hr

ICNAAM 2018, Rhodes
September 13–18, 2018, Rhodes, Greece

OUTLINE

OUTLINE

- GEP, PGEP

This work has been fully supported by Croatian Science Foundation under the project IP-09-2014-3670.

OUTLINE

- GEP, PGEP
- Known Real and Complex Diagonalization Methods for PGEP

This work has been fully supported by Croatian Science Foundation under the project IP-09-2014-3670.

OUTLINE

- GEP, PGEP
- Known Real and Complex Diagonalization Methods for PGEP
- Derivation of the Complex Cholesky-Jacobi Algorithm

This work has been fully supported by Croatian Science Foundation under the project IP-09-2014-3670.

hrzz
Hrvatska zaklada za znanost

OUTLINE

- GEP, PGEP
- Known Real and Complex Diagonalization Methods for PGEP
- Derivation of the Complex Cholesky-Jacobi Algorithm
- Properties: Convergence (global and asymptotic)

This work has been fully supported by Croatian Science Foundation under the project IP-09-2014-3670.

hrzz
Hrvatska zaklada za znanost

OUTLINE

- GEP, PGEP
- Known Real and Complex Diagonalization Methods for PGEP
- Derivation of the Complex Cholesky-Jacobi Algorithm
- Properties: Convergence (global and asymptotic)
- Stability and High Relative Accuracy (HRA)

This work has been fully supported by Croatian Science Foundation under the project IP-09-2014-3670.

GEP and PGEP

Let $A = A^*, \quad B = B^*.$

GEP and PGEP

Let $A = A^*, \quad B = B^*$.
We consider the Generalized Eigenvalue Problem (GEP)

$$Ax = \lambda Bx, \quad x \neq 0.$$

GEP and PGEP

Let $A = A^*, \quad B = B^*$.
We consider the Generalized Eigenvalue Problem (GEP)

$$Ax = \lambda Bx, \quad x \neq 0.$$

If $B \succ O$, GEP is called Positive definite GEP (PGEP)

GEP and PGEP

Let $A = A^*, \quad B = B^*$.
We consider the Generalized Eigenvalue Problem (GEP)

$$Ax = \lambda Bx, \quad x \neq 0.$$

If $B \succ O$, GEP is called Positive definite GEP (PGEP)

If $B \succ O$, then the pair (A, B) is called positive definite pair

GEP and PGEP

Let $A = A^*$, $\quad B = B^*$.
We consider the Generalized Eigenvalue Problem (GEP)

$$Ax = \lambda Bx, \quad x \neq 0.$$

If $B \succ O$, GEP is called Positive definite GEP (PGEP)

If $B \succ O$, then the pair (A, B) is called positive definite pair

For each positive definite pair (A, B) there exists a nonsingular matrix F such that

$$F^* AF = \Lambda_A = \mathrm{diag}(\alpha_1, \ldots, \alpha_n), \quad F^* BF = \Lambda_B = \mathrm{diag}(\beta_1, \ldots, \beta_n)$$

GEP and PGEP

Let $A = A^*, \quad B = B^*$.
We consider the Generalized Eigenvalue Problem (GEP)

$$Ax = \lambda Bx, \quad x \neq 0.$$

If $B \succ O$, GEP is called Positive definite GEP (PGEP)

If $B \succ O$, then the pair (A, B) is called positive definite pair

For each positive definite pair (A, B) there exists a nonsingular matrix F such that

$$F^* AF = \Lambda_A = \mathrm{diag}(\alpha_1, \ldots, \alpha_n), \quad F^* BF = \Lambda_B = \mathrm{diag}(\beta_1, \ldots, \beta_n)$$

The eigenpairs of (A, B) are: $(\alpha_i/\beta_i, Fe_i), \ 1 \leq i \leq n$,

here $I_n = [e_1, \ldots, e_n]$

Well-behaved Pairs are Linked to High Relative Accuracy

There is a special class of pairs of Hermitian matrices that we briefly call

well-behaved pairs.

Well-behaved Pairs are Linked to High Relative Accuracy

There is a special class of pairs of Hermitian matrices that we briefly call

well-behaved pairs.

This class consists of pairs of

well-behaved Hermitian positive definite matrices.

Well-behaved Pairs are Linked to High Relative Accuracy

There is a special class of pairs of Hermitian matrices that we briefly call

well-behaved pairs.

This class consists of pairs of

well-behaved Hermitian positive definite matrices.

$B \succ O$ is well-behaved if it can be well-scaled,

Well-behaved Pairs are Linked to High Relative Accuracy

There is a special class of pairs of Hermitian matrices that we briefly call

well-behaved pairs.

This class consists of pairs of

well-behaved Hermitian positive definite matrices.

$B \succ O$ is well-behaved if it can be well-scaled, i.e. if

$$\kappa_2(DBD) = \|DBD\|_2 \|(DBD)^{-1}\|_2$$

is small for some diagonal matrix D.

Well-behaved Pairs are Linked to High Relative Accuracy

There is a special class of pairs of Hermitian matrices that we briefly call

well-behaved pairs.

This class consists of pairs of

well-behaved Hermitian positive definite matrices.

$B \succ O$ is well-behaved if it can be well-scaled, i.e. if

$$\kappa_2(DBD) = \|DBD\|_2 \|(DBD)^{-1}\|_2$$

is small for some diagonal matrix D.

To detect whether B is well-behaved, it is sufficient to check whether

$$\kappa_2(B_S), \quad B_S = [\mathrm{diag}(B)]^{-1/2} B [\mathrm{diag}(B)]^{-1/2} \quad \text{is small.}$$

Why are Element-wise Methods Important

Why are Element-wise Methods Important

On contemporary GPU and CPU parallel computing machines, probably the best methods for solving PGEP are block diagonalization methods.

Why are Element-wise Methods Important

On contemporary GPU and CPU parallel computing machines, probably the best methods for solving PGEP are block diagonalization methods.

They use kernel algorithms to perform an intrinsic job at each step - solving PGEP with much smaller matrices (say, $n = 32 - 512$).

Why are Element-wise Methods Important

On contemporary GPU and CPU parallel computing machines, probably the best methods for solving PGEP are block diagonalization methods.

They use kernel algorithms to perform an intrinsic job at each step - solving PGEP with much smaller matrices (say, $n = 32 - 512$).

The block method will function well only if the kernel algorithm if globally convergent, fast and accurate.

Why are Element-wise Methods Important

On contemporary GPU and CPU parallel computing machines, probably the best methods for solving PGEP are block diagonalization methods.

They use kernel algorithms to perform an intrinsic job at each step - solving PGEP with much smaller matrices (say, $n = 32 - 512$).

The block method will function well only if the kernel algorithm if globally convergent, fast and accurate.

Most of the time, the kernel algorithm will operate on nearly diagonal matrices. On such matrices, the element-wise diagonalization methods are fast and highly accurate.

Why are Element-wise Methods Important

On contemporary GPU and CPU parallel computing machines, probably the best methods for solving PGEP are block diagonalization methods.

They use kernel algorithms to perform an intrinsic job at each step - solving PGEP with much smaller matrices (say, $n = 32 - 512$).

The block method will function well only if the kernel algorithm if globally convergent, fast and accurate.

Most of the time, the kernel algorithm will operate on nearly diagonal matrices. On such matrices, the element-wise diagonalization methods are fast and highly accurate.

Hence, probably the best choice for the kernel algorithm are element-wise diagonalization methods.

Why Element-wise, Two-sided Jacobi Methods

Why Element-wise, Two-sided Jacobi Methods

- they can be used standalone or as kernel algorithms in the block methods

Why Element-wise, Two-sided Jacobi Methods

- they can be used standalone or as kernel algorithms in the block methods
- as basic algorithms they can be "upgraded" to one-sided algorithms

Why Element-wise, Two-sided Jacobi Methods

- they can be used standalone or as kernel algorithms in the block methods

- as basic algorithms they can be "upgraded" to one-sided algorithms

- the theoretical aspects of one-sided methods can be better analysed and understood if they are considered/imagined as two-sided methods

Why Element-wise, Two-sided Jacobi Methods

- they can be used standalone or as kernel algorithms in the block methods

- as basic algorithms they can be "upgraded" to one-sided algorithms

- the theoretical aspects of one-sided methods can be better analysed and understood if they are considered/imagined as two-sided methods

- One-sided methods have problem with terminating the process. Stopping of the process can be costly, especially if the matrix dimension n is large.

Why Element-wise, Two-sided Jacobi Methods

- they can be used standalone or as kernel algorithms in the block methods
- as basic algorithms they can be "upgraded" to one-sided algorithms
- the theoretical aspects of one-sided methods can be better analysed and understood if they are considered/imagined as two-sided methods
- One-sided methods have problem with terminating the process. Stopping of the process can be costly, especially if the matrix dimension n is large.
- Two sided methods can smoothly, timely and cost effectively stop the process.

What Jacobi Methods for PGEP are Known?

So far we know three "promising" real diagonalization methods:

What Jacobi Methods for PGEP are Known?

So far we know three "promising" real diagonalization methods:

- Falk-Langemeyer method (shorter: FL method)
  (Elektronische Datenverarbeitung, 1960)

What Jacobi Methods for PGEP are Known?

So far we know three "promising" real diagonalization methods:

- Falk-Langemeyer method (shorter: FL method)
  (Elektronische Datenverarbeitung, 1960)
- Hari-Zimmermann method (shorter: HZ method)
  (Numerical Algorithms, to appear)

What Jacobi Methods for PGEP are Known?

So far we know three "promising" real diagonalization methods:

- Falk-Langemeyer method (shorter: FL method)
  (Elektronische Datenverarbeitung, 1960)
- Hari-Zimmermann method (shorter: HZ method)
  (Numerical Algorithms, to appear)
- Cholesky-Jacobi method (shorter: CJ method)
  (Numerical Algorithms, to appear)

What Jacobi Methods for PGEP are Known?

So far we know three "promising" <u>real</u> diagonalization methods:

- Falk-Langemeyer method (shorter: FL method)
  (Elektronische Datenverarbeitung, 1960)
- Hari-Zimmermann method (shorter: HZ method)
  (Numerical Algorithms, to appear)
- Cholesky-Jacobi method (shorter: CJ method)
  (Numerical Algorithms, to appear)

The methods are connected: the FL method can be viewed as the HZ or CJ method with "fast scaled" transformations.

What Jacobi Methods for PGEP are Known?

So far we know three "promising" real diagonalization methods:

- Falk-Langemeyer method (shorter: FL method)
  (Elektronische Datenverarbeitung, 1960)
- Hari-Zimmermann method (shorter: HZ method)
  (Numerical Algorithms, to appear)
- Cholesky-Jacobi method (shorter: CJ method)
  (Numerical Algorithms, to appear)

The methods are connected: the FL method can be viewed as the HZ or CJ method with "fast scaled" transformations.

We have recently derived their "equally promising" complex counterparts.

The Main Characteristics of the FL Methods

The Main Characteristics of the FL Methods

- Very fast (SAXPY BLAS1 operations, Fused multiplyadd)

The Main Characteristics of the FL Methods

- Very fast (SAXPY BLAS1 operations, Fused multiplyadd)
- Very accurate (HRA is indicated on pairs of well-behaved positive definite matrices)

The Main Characteristics of the FL Methods

- Very fast (SAXPY BLAS1 operations, Fused multiplyadd)
- Very accurate (HRA is indicated on pairs of well-behaved positive definite matrices)
- Well defined for a larger class of pairs (they solve definite GEP)

The Main Characteristics of the FL Methods

- Very fast (SAXPY BLAS1 operations, Fused multiplyadd)
- Very accurate (HRA is indicated on pairs of well-behaved positive definite matrices)
- Well defined for a larger class of pairs (they solve definite GEP)
- Problems with renormalizations (every F_k has unit diagonal, hence $$\|A^{(k)}\| \nearrow \infty, \quad \|B^{(k)}\| \nearrow \infty, \quad \|F_1 F_2 \cdots F_k\| \nearrow \infty$$)

The Main Characteristics of the FL Methods

- Very fast (SAXPY BLAS1 operations, Fused multiplyadd)
- Very accurate (HRA is indicated on pairs of well-behaved positive definite matrices)
- Well defined for a larger class of pairs (they solve definite GEP)
- Problems with renormalizations (every F_k has unit diagonal, hence
$$\|A^{(k)}\| \nearrow \infty, \quad \|B^{(k)}\| \nearrow \infty, \quad \|F_1 F_2 \cdots F_k\| \nearrow \infty$$
)
- Difficult and challenging for making a good numerical code (to many freedoms, all we have $\alpha A + \beta B \succ O$, when to stop the iterations?)

The Main Characteristics of the FL Methods

- Very fast (SAXPY BLAS1 operations, Fused multiplyadd)
- Very accurate (HRA is indicated on pairs of well-behaved positive definite matrices)
- Well defined for a larger class of pairs (they solve definite GEP)
- Problems with renormalizations (every F_k has unit diagonal, hence
  $$\|A^{(k)}\| \nearrow \infty, \quad \|B^{(k)}\| \nearrow \infty, \quad \|F_1 F_2 \cdots F_k\| \nearrow \infty$$
)
- Difficult and challenging for making a good numerical code (to many freedoms, all we have $\alpha A + \beta B \succ O$, when to stop the iterations?)
- Theoretical results are lacking (all we have is the quadratic asymptotic convergence result)

The Main Characteristics of the HZ Methods

The Main Characteristics of the HZ Methods

- Fast (the quadratic asymptotic convergence has been proved)

The Main Characteristics of the HZ Methods

- Fast (the quadratic asymptotic convergence has been proved)
- Very accurate (HRA indicated on well-behaved pairs (A, B))

The Main Characteristics of the HZ Methods

- Fast (the quadratic asymptotic convergence has been proved)
- Very accurate (HRA indicated on well-behaved pairs (A, B))
- No Problem with renormalizations, easy to code

The Main Characteristics of the HZ Methods

- Fast (the quadratic asymptotic convergence has been proved)
- Very accurate (HRA indicated on well-behaved pairs (A, B))
- No Problem with renormalizations, easy to code
- Unit diagonal in B simplifies the algorithm and has a stabilizing effect on the iterative process, because it almost optimally reduces the condition of B and all $B^{(k)}$, $k \geq 1$. Van der Sluis, A. Numer. Math. 14 (1969)

The Main Characteristics of the HZ Methods

- Fast (the quadratic asymptotic convergence has been proved)
- Very accurate (HRA indicated on well-behaved pairs (A, B))
- No Problem with renormalizations, easy to code
- Unit diagonal in B simplifies the algorithm and has a stabilizing effect on the iterative process, because it almost optimally reduces the condition of B and all $B^{(k)}$, $k \geq 1$. Van der Sluis, A. Numer. Math. 14 (1969)
- Theoretical results exist (global and asymptotic convergence is proved, much is known on the relative accuracy of the computed eigenvalues)

The Main Characteristics of the HZ Methods

- Fast (the quadratic asymptotic convergence has been proved)
- Very accurate (HRA indicated on well-behaved pairs (A, B))
- No Problem with renormalizations, easy to code
- Unit diagonal in B simplifies the algorithm and has a stabilizing effect on the iterative process, because it almost optimally reduces the condition of B and all $B^{(k)}$, $k \geq 1$. Van der Sluis, A. Numer. Math. 14 (1969)
- Theoretical results exist (global and asymptotic convergence is proved, much is known on the relative accuracy of the computed eigenvalues)
- It requires B to be positive definite (it solves PGEP)

What is Known for the Real CJ Method

What is Known for the Real CJ Method

- Theoretical results exist (the global convergence is proved)

What is Known for the Real CJ Method

- Theoretical results exist (the global convergence is proved)
- Fast (numerical tests indicate quadratic asymptotic convergence)

What is Known for the Real CJ Method

- Theoretical results exist (the global convergence is proved)
- Fast (numerical tests indicate quadratic asymptotic convergence)
- Very accurate (numerical tests indicate HRA on pairs of well-behaved positive definite matrices)

What is Known for the Real *CJ* Method

- Theoretical results exist (the global convergence is proved)
- Fast (numerical tests indicate quadratic asymptotic convergence)
- Very accurate (numerical tests indicate HRA on pairs of well-behaved positive definite matrices)
- No Problem with renormalizations, easy to code

What is Known for the Real CJ Method

- Theoretical results exist (the global convergence is proved)
- Fast (numerical tests indicate quadratic asymptotic convergence)
- Very accurate (numerical tests indicate HRA on pairs of well-behaved positive definite matrices)
- No Problem with renormalizations, easy to code
- Unit diagonal in B has a stabilizing effect

What is Known for the Real CJ Method

- Theoretical results exist (the global convergence is proved)
- Fast (numerical tests indicate quadratic asymptotic convergence)
- Very accurate (numerical tests indicate HRA on pairs of well-behaved positive definite matrices)
- No Problem with renormalizations, easy to code
- Unit diagonal in B has a stabilizing effect
- It requires B to be positive definite (it solves PGEP)

Derivation of the Complex CJ Method

Starting with a positive definite pair (A, B), CJ first makes unit diagonal in B:

$$(A^{(0)}, B^{(0)}) = (DAD, DBD), \qquad D = [\mathrm{diag}(B)]^{-1/2}.$$

Derivation of the Complex CJ Method

Starting with a positive definite pair (A, B), CJ first makes unit diagonal in B:

$$(A^{(0)}, B^{(0)}) = (DAD, DBD), \qquad D = [\mathrm{diag}(B)]^{-1/2}.$$

Then it generates a sequence of "congruent" matrix pairs

$$(A^{(0)}, B^{(0)}), \;\; (A^{(1)}, B^{(1)}), \;\; (A^{(2)}, B^{(2)}), \;\; \ldots$$

by the rule

Derivation of the Complex CJ Method

Starting with a positive definite pair (A, B), CJ first makes unit diagonal in B:

$$(A^{(0)}, B^{(0)}) = (DAD, DBD), \qquad D = [\mathrm{diag}(B)]^{-1/2}.$$

Then it generates a sequence of "congruent" matrix pairs

$$(A^{(0)}, B^{(0)}), \;\; (A^{(1)}, B^{(1)}), \;\; (A^{(2)}, B^{(2)}), \;\; \ldots$$

by the rule

$$A^{(k+1)} = F_k^* A^{(k)} F_k \;, \qquad B^{(k+1)} = F_k^* B^{(k)} F_k \;, \quad k \geq 0.$$

Derivation of the Complex CJ Method

Starting with a positive definite pair (A, B), CJ first makes unit diagonal in B:

$$(A^{(0)}, B^{(0)}) = (DAD, DBD), \qquad D = [\mathrm{diag}(B)]^{-1/2}.$$

Then it generates a sequence of "congruent" matrix pairs

$$(A^{(0)}, B^{(0)}),\ \ (A^{(1)}, B^{(1)}),\ \ (A^{(2)}, B^{(2)}),\ \ \ldots$$

by the rule

$$A^{(k+1)} = F_k^* A^{(k)} F_k\ , \qquad B^{(k+1)} = F_k^* B^{(k)} F_k\ , \quad k \geq 0.$$

Here each F_k is elementary plane matrix defined by the *pivot pair* $(i(k), j(k))$ and the *pivot submatrix* $\hat{F}_k$

$$F_k = \begin{array}{cl} \left[\begin{array}{ccccc} I & & & & \\ & * & & * & \\ & & I & & \\ & * & & * & \\ & & & & I \end{array}\right] & \begin{array}{l} \\ i(k) \\ \\ j(k) \\ \\ \end{array} \end{array}, \qquad \hat{F}_k = \begin{bmatrix} * & * \\ * & * \end{bmatrix}$$

Derivation of the Complex CJ Algorithm

Let us fix k, $k \geq 1$, and consider one step of the method.

Derivation of the Complex CJ Algorithm

Let us fix k, $k \geq 1$, and consider one step of the method.

By algorithm we mean one step of the method.

Derivation of the Complex CJ Algorithm

Let us fix k, $k \geq 1$, and consider one step of the method.

By algorithm we mean one step of the method.

We simplify notation:

$$A = A^{(k)}, \quad A' = A^{(k+1)}, \quad F_k = F, \quad (i,j) = (i(k), j(k)).$$

Derivation of the Complex CJ Algorithm

Let us fix k, $k \geq 1$, and consider one step of the method.

By algorithm we mean one step of the method.

We simplify notation:

$$A = A^{(k)}, \quad A' = A^{(k+1)}, \quad F_k = F, \quad (i,j) = (i(k), j(k)).$$

Then we have

$$A' = F^* A F, \quad B' = F^* B F \qquad \left(\hat{A}' = \hat{F}^* \hat{A} \hat{F}, \quad \hat{B}' = \hat{F}^* \hat{B} \hat{F}\right).$$

Derivation of the Complex CJ Algorithm

Let us fix k, $k \geq 1$, and consider one step of the method.

By algorithm we mean one step of the method.

We simplify notation:

$$A = A^{(k)}, \quad A' = A^{(k+1)}, \quad F_k = F, \quad (i,j) = (i(k), j(k)).$$

Then we have

$$A' = F^* A F, \quad B' = F^* B F \qquad \left(\hat{A}' = \hat{F}^* \hat{A} \hat{F}, \quad \hat{B}' = \hat{F}^* \hat{B} \hat{F}\right).$$

The pivot submatrices $\hat{A}$, $\hat{B}$, $\hat{F}$ of A, B, F, resp. are 2×2 principal submatrices obtained on the intersection of pivot rows and columns i, j.

Derivation of the Complex CJ Algorithm

Let us fix k, $k \geq 1$, and consider one step of the method.

By algorithm we mean one step of the method.

We simplify notation:

$$A = A^{(k)}, \quad A' = A^{(k+1)}, \quad F_k = F, \quad (i,j) = (i(k), j(k)).$$

Then we have

$$A' = F^* A F, \quad B' = F^* B F \qquad \left(\hat{A}' = \hat{F}^* \hat{A} \hat{F}, \quad \hat{B}' = \hat{F}^* \hat{B} \hat{F} \right).$$

The pivot submatrices $\hat{A}$, $\hat{B}$, $\hat{F}$ of A, B, F, resp. are 2×2 principal submatrices obtained on the intersection of pivot rows and columns i, j.

The goal is to compute $\hat{F}$ that diagonalizes $\hat{A}$ and reduces $\hat{B}$ to I_2.

The Derivation of the Complex CJ Method

The complex CJ method is a

hybrid method.

The Derivation of the Complex CJ Method

The complex CJ method is a

hybrid method.

At each step it uses

either LL^*J or RR^*J algorithm.

The Derivation of the Complex CJ Method

The complex CJ method is a

hybrid method.

At each step it uses

either LL^*J or RR^*J algorithm.

It chooses the algorithm which is for the given data (that is $(\hat{A}, \hat{B})$)

more accurate.

The Derivation of the LL^*J Algorithm

Consider the Cholesky foctorization of $\hat{B}$: $\hat{B} = \hat{L}\hat{L}^*$,

$$\begin{bmatrix} 1 & b_{ij} \\ \bar{b}_{ij} & 1 \end{bmatrix} = \hat{B} = \hat{L}\hat{L}^* = \begin{bmatrix} 1 & 0 \\ \bar{a} & \bar{c} \end{bmatrix} \begin{bmatrix} 1 & a \\ 0 & c \end{bmatrix} = \begin{bmatrix} 1 & a \\ \bar{a} & |a|^2 + |c|^2 \end{bmatrix}$$

The Derivation of the LL^*J Algorithm

Consider the Cholesky foctorization of $\hat{B}$: $\hat{B} = \hat{L}\hat{L}^*$,

$$\begin{bmatrix} 1 & b_{ij} \\ \bar{b}_{ij} & 1 \end{bmatrix} = \hat{B} = \hat{L}\hat{L}^* = \begin{bmatrix} 1 & 0 \\ \bar{a} & \bar{c} \end{bmatrix} \begin{bmatrix} 1 & a \\ 0 & c \end{bmatrix} = \begin{bmatrix} 1 & a \\ \bar{a} & |a|^2 + |c|^2 \end{bmatrix}$$

Assuming $c > 0$, one obtains $\quad a = b_{ij}, \;\; c = \tau \equiv \sqrt{1 - |b_{ij}|^2}.$

The Derivation of the LL^*J Algorithm

Consider the Cholesky foctorization of $\hat{B}$: $\hat{B} = \hat{L}\hat{L}^*$,

$$\begin{bmatrix} 1 & b_{ij} \\ \bar{b}_{ij} & 1 \end{bmatrix} = \hat{B} = \hat{L}\hat{L}^* = \begin{bmatrix} 1 & 0 \\ \bar{a} & \bar{c} \end{bmatrix} \begin{bmatrix} 1 & a \\ 0 & c \end{bmatrix} = \begin{bmatrix} 1 & a \\ \bar{a} & |a|^2 + |c|^2 \end{bmatrix}$$

Assuming $c > 0$, one obtains $\quad a = b_{ij}, \;\; c = \tau \equiv \sqrt{1 - |b_{ij}|^2}$.

$$\hat{L} = \begin{bmatrix} 1 & 0 \\ \bar{b}_{ij} & \tau \end{bmatrix}, \qquad \hat{L}^{-1} = \frac{1}{\tau} \begin{bmatrix} \tau & 0 \\ -\bar{b}_{ij} & 1 \end{bmatrix}, \qquad \hat{L}^{-*} = \frac{1}{\tau} \begin{bmatrix} \tau & -b_{ij} \\ 0 & 1 \end{bmatrix}.$$

The Derivation of the LL^*J Algorithm

Consider the Cholesky foctorization of $\hat{B}$: $\hat{B} = \hat{L}\hat{L}^*$,

$$\begin{bmatrix} 1 & b_{ij} \\ \bar{b}_{ij} & 1 \end{bmatrix} = \hat{B} = \hat{L}\hat{L}^* = \begin{bmatrix} 1 & 0 \\ \bar{a} & \bar{c} \end{bmatrix} \begin{bmatrix} 1 & a \\ 0 & c \end{bmatrix} = \begin{bmatrix} 1 & a \\ \bar{a} & |a|^2 + |c|^2 \end{bmatrix}$$

Assuming $c > 0$, one obtains $\quad a = b_{ij}, \;\; c = \tau \equiv \sqrt{1 - |b_{ij}|^2}$.

$$\hat{L} = \begin{bmatrix} 1 & 0 \\ \bar{b}_{ij} & \tau \end{bmatrix}, \qquad \hat{L}^{-1} = \frac{1}{\tau} \begin{bmatrix} \tau & 0 \\ -\bar{b}_{ij} & 1 \end{bmatrix}, \qquad \hat{L}^{-*} = \frac{1}{\tau} \begin{bmatrix} \tau & -b_{ij} \\ 0 & 1 \end{bmatrix}.$$

Let $\hat{F}_1 = \hat{L}^{-*}$. Then $\hat{F}_1^* \hat{B} \hat{F}_1 = I_2$ and

$$\hat{F}_1^* \hat{A} \hat{F}_1 = \begin{bmatrix} a_{ii} & (a_{ij} - b_{ij} a_{ii})/\tau \\ (\bar{a}_{ij} - \bar{b}_{ij} a_{ii})/\tau & a_{jj} - \frac{a_{ij}\bar{b}_{ij} + \bar{a}_{ij} b_{ij} - (a_{ii} + a_{jj})|b_{ij}|^2}{1 - |b_{ij}|^2} \end{bmatrix}.$$

The Derivation of the LL^*J Algorithm

The final $\hat{F}$ is obtained as product $\hat{F} = \hat{F}_1 \hat{R}_1$ where

$\hat{R}_1$ is the complex Jacobi rotation which diagonalizes $\hat{F}_1^* \hat{A} \hat{F}_1$.

Let us assume

$$\hat{R}_1 = \begin{bmatrix} c_{\vartheta_1} & -s_{\vartheta_1}^{+} \\ s_{\vartheta_1}^{-} & c_{\vartheta_1} \end{bmatrix}, \qquad c_{\vartheta_1} = \cos\vartheta_1, \qquad s_{\vartheta_1}^{\pm} = e^{\pm\imath\epsilon_1} \sin\vartheta_1.$$

Then the angles ϑ_1 and ϵ_1 are determined by the formulas

$$\begin{aligned} \epsilon_1 &= \arg(a_{ij} - b_{ij} a_{ii}), \\ \tan(2\vartheta_1) &= \frac{2|a_{ij} - a_{ii} b_{ij}| \sqrt{1 - |b_{ij}|^2}}{a_{ii} - a_{jj} + a_{ij}\bar{b}_{ij} + \bar{a}_{ij} b_{ij} - 2a_{ii}|b_{ij}|^2}, \quad -\frac{\pi}{4} \le \vartheta_1 \le \frac{\pi}{4}. \end{aligned}$$

The Derivation of the LL^*J Algorithm

The transformation formulas for the diagonal elements of A read

$$\begin{aligned}
a'_{ii} &= a_{ii} + \tan\vartheta_1 \cdot \frac{|a_{ij} - a_{ii}b_{ij}|}{\sqrt{1-|b_{ij}|^2}}, \\
a'_{jj} &= a_{jj} - \frac{a_{ij}\bar{b}_{ij} + \bar{a}_{ij}b_{ij} - (a_{ii}+a_{jj})|b_{ij}|^2}{1-|b_{ij}|^2} - \tan\vartheta_1 \cdot \frac{|a_{ij} - a_{ii}b_{ij}|}{\sqrt{1-|b_{ij}|^2}}.
\end{aligned}$$

The Derivation of the LL^*J Algorithm

The transformation formulas for the diagonal elements of A read

$$
\begin{aligned}
a'_{ii} &= a_{ii} + \tan\vartheta_1 \cdot \frac{|a_{ij} - a_{ii} b_{ij}|}{\sqrt{1 - |b_{ij}|^2}}, \\
a'_{jj} &= a_{jj} - \frac{a_{ij}\bar{b}_{ij} + \bar{a}_{ij} b_{ij} - (a_{ii} + a_{jj})|b_{ij}|^2}{1 - |b_{ij}|^2} - \tan\vartheta_1 \cdot \frac{|a_{ij} - a_{ii} b_{ij}|}{\sqrt{1 - |b_{ij}|^2}}.
\end{aligned}
$$

In the case $a_{ii} = a_{jj}$, $a_{ij} = a_{ii} b_{ij}$, $\tan(2\vartheta_1)$ has the form $0/0$.

The Derivation of the LL^*J Algorithm

The transformation formulas for the diagonal elements of A read

$$\begin{aligned} a'_{ii} &= a_{ii} + \tan\vartheta_1 \cdot \frac{|a_{ij} - a_{ii} b_{ij}|}{\sqrt{1 - |b_{ij}|^2}}, \\ a'_{jj} &= a_{jj} - \frac{a_{ij}\bar{b}_{ij} + \bar{a}_{ij} b_{ij} - (a_{ii} + a_{jj})|b_{ij}|^2}{1 - |b_{ij}|^2} - \tan\vartheta_1 \cdot \frac{|a_{ij} - a_{ii} b_{ij}|}{\sqrt{1 - |b_{ij}|^2}}. \end{aligned}$$

In the case $a_{ii} = a_{jj}$, $a_{ij} = a_{ii} b_{ij}$, $\tan(2\vartheta_1)$ has the form $0/0$.

Then we choose $\vartheta_1 = 0$, so that $a'_{ii} = a_{ii}$ and $a'_{jj} = a_{jj}$.

The Derivation of the LL^*J Algorithm

$$
\begin{aligned}
\hat{F} &= \frac{1}{\sqrt{1-|b_{ij}|^2}} \begin{bmatrix} \sqrt{1-|b_{ij}|^2} & -b_{ij} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} c_{\vartheta_1} & -s_{\vartheta_1}^{+} \\ s_{\vartheta_1}^{-} & c_{\vartheta_1} \end{bmatrix} \\
&= \frac{1}{\sqrt{1-|b_{ij}|^2}} \begin{bmatrix} c_{\tilde{\vartheta}_1} & -s_{\tilde{\vartheta}_1} \\ s_{\vartheta_1}^{-} & c_{\vartheta_1} \end{bmatrix} = \begin{bmatrix} c1 & -s1 \\ s2 & c2 \end{bmatrix},
\end{aligned}
$$

The Derivation of the LL^*J Algorithm

$$\begin{aligned} \hat{F} &= \frac{1}{\sqrt{1-|b_{ij}|^2}} \left[\begin{array}{cc} \sqrt{1-|b_{ij}|^2} & -b_{ij} \\ 0 & 1 \end{array} \right] \left[\begin{array}{cc} c_{\vartheta_1} & -s_{\vartheta_1}^{+} \\ s_{\vartheta_1}^{-} & c_{\vartheta_1} \end{array} \right] \\ &= \frac{1}{\sqrt{1-|b_{ij}|^2}} \left[\begin{array}{cc} c_{\tilde{\vartheta}_1} & -s_{\tilde{\vartheta}_1} \\ s_{\vartheta_1}^{-} & c_{\vartheta_1} \end{array} \right] = \left[\begin{array}{cc} c1 & -s1 \\ s2 & c2 \end{array} \right], \end{aligned}$$

$$c_{\tilde{\vartheta}_1} = c_{\vartheta_1}\sqrt{1-|b_{ij}|^2} - s_{\vartheta_1}^{-} b_{ij}, \quad s_{\tilde{\vartheta}_1} = c_{\vartheta_1} b_{ij} + s_{\vartheta_1}^{+}\sqrt{1-|b_{ij}|^2},$$

$$|c_{\tilde{\vartheta}_1}|^2 + |s_{\tilde{\vartheta}_1}|^2 = 1,$$

The Derivation of the LL^*J Algorithm

$$
\begin{aligned}
\hat{F} &= \frac{1}{\sqrt{1-|b_{ij}|^2}} \begin{bmatrix} \sqrt{1-|b_{ij}|^2} & -b_{ij} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} c_{\vartheta_1} & -s_{\vartheta_1}^{+} \\ s_{\vartheta_1}^{-} & c_{\vartheta_1} \end{bmatrix} \\
&= \frac{1}{\sqrt{1-|b_{ij}|^2}} \begin{bmatrix} c_{\tilde{\vartheta}_1} & -s_{\tilde{\vartheta}_1} \\ s_{\vartheta_1}^{-} & c_{\vartheta_1} \end{bmatrix} = \begin{bmatrix} c1 & -s1 \\ s2 & c2 \end{bmatrix},
\end{aligned}
$$

$$
c_{\tilde{\vartheta}_1} = c_{\vartheta_1}\sqrt{1-|b_{ij}|^2} - s_{\vartheta_1}^{-} b_{ij}, \quad s_{\tilde{\vartheta}_1} = c_{\vartheta_1} b_{ij} + s_{\vartheta_1}^{+}\sqrt{1-|b_{ij}|^2},
$$

$$
|c_{\tilde{\vartheta}_1}|^2 + |s_{\tilde{\vartheta}_1}|^2 = 1,
$$

$$
\begin{aligned}
c1 &= c_{\vartheta_1} - s_{\vartheta_1}^{-} b_{ij}/\sqrt{1-|b_{ij}|^2}, \qquad & c2 &= c_{\vartheta_1}/\sqrt{1-|b_{ij}|^2}, \\
s1 &= c_{\vartheta_1} b_{ij}/\sqrt{1-|b_{ij}|^2} + s_{\vartheta_1}^{+}, \qquad & s2 &= s_{\vartheta_1}^{-}/\sqrt{1-|b_{ij}|^2}.
\end{aligned}
$$

The Derivation of the LL^*J Algorithm

$$\hat{F} = \begin{bmatrix} c1 & -s1 \\ s2 & c2 \end{bmatrix}, \quad \begin{array}{ll} c1 = c_{\vartheta_1} - s_{\vartheta_1}^{-} b_{ij}/\sqrt{1-|b_{ij}|^2}, & c2 = c_{\vartheta_1}/\sqrt{1-|b_{ij}|^2} \\ s1 = c_{\vartheta_1} b_{ij}/\sqrt{1-|b_{ij}|^2} + s_{\vartheta_1}^{+}, & s2 = s_{\vartheta_1}^{-}/\sqrt{1-|b_{ij}|^2} \end{array}$$

The Derivation of the LL^*J Algorithm

$$\hat{F} = \begin{bmatrix} c1 & -s1 \\ s2 & c2 \end{bmatrix}, \quad \begin{array}{ll} c1 = c_{\vartheta_1} - s_{\vartheta_1}^{-} b_{ij}/\sqrt{1-|b_{ij}|^2}, & c2 = c_{\vartheta_1}/\sqrt{1-|b_{ij}|^2} \\ s1 = c_{\vartheta_1} b_{ij}/\sqrt{1-|b_{ij}|^2} + s_{\vartheta_1}^{+}, & s2 = s_{\vartheta_1}^{-}/\sqrt{1-|b_{ij}|^2} \end{array}$$

This algorithm works well, but we can still reduce the number of floating point operations per iteration step. This is accomplished by transforming the complex element $c1$ into $|c1|$.

The Derivation of the LL^*J Algorithm

$$\hat{F} = \begin{bmatrix} c1 & -s1 \\ s2 & c2 \end{bmatrix}, \quad \begin{array}{ll} c1 = c_{\vartheta_1} - s_{\vartheta_1}^- b_{ij}/\sqrt{1-|b_{ij}|^2}, & c2 = c_{\vartheta_1}/\sqrt{1-|b_{ij}|^2} \\ s1 = c_{\vartheta_1} b_{ij}/\sqrt{1-|b_{ij}|^2} + s_{\vartheta_1}^+, & s2 = s_{\vartheta_1}^-/\sqrt{1-|b_{ij}|^2} \end{array}$$

This algorithm works well, but we can still reduce the number of floating point operations per iteration step. This is accomplished by transforming the complex element $c1$ into $|c1|$.

Formally, we postmultiply $\hat{F}$ by the diagonal matrix $\mathrm{diag}(\bar{c}_{\tilde{\vartheta}_1}/|c_{\tilde{\vartheta}_1}|, 1)$, provided that $c_{\tilde{\vartheta}_1} \neq 0$. That transforms $s2$ into $s2 \cdot \bar{c}_{\tilde{\vartheta}_1}/|c_{\tilde{\vartheta}_1}|$.

$$\hat{F} = \begin{bmatrix} c1 & -s1 \\ s2 & c2 \end{bmatrix}, \quad \begin{array}{ll} c1 = c_{\vartheta_1} - s_{\vartheta_1}^{-} b_{ij}/\sqrt{1-|b_{ij}|^2}, & c2 = c_{\vartheta_1}/\sqrt{1-|b_{ij}|^2} \\ s1 = c_{\vartheta_1} b_{ij}/\sqrt{1-|b_{ij}|^2} + s_{\vartheta_1}^{+}, & s2 = s_{\vartheta_1}^{-}/\sqrt{1-|b_{ij}|^2} \end{array}$$

This algorithm works well, but we can still reduce the number of floating point operations per iteration step. This is accomplished by transforming the complex element $c1$ into $|c1|$.

Formally, we postmultiply $\hat{F}$ by the diagonal matrix $\mathrm{diag}(\bar{c}_{\tilde{\vartheta}_1}/|c_{\tilde{\vartheta}_1}|, 1)$, provided that $c_{\tilde{\vartheta}_1} \neq 0$. That transforms $s2$ into $s2 \cdot \bar{c}_{\tilde{\vartheta}_1}/|c_{\tilde{\vartheta}_1}|$.

The obtained algorithm we call LL^*J algorithm.

The Derivation of the RR^*J Algorithm

Instead of LL^*, one can use RR^* factorization of $\hat{B}$. Then we have

The Derivation of the RR^*J Algorithm

Instead of LL^*, one can use RR^* factorization of $\hat{B}$. Then we have

$$\begin{bmatrix} 1 & b_{ij} \\ \bar{b}_{ij} & 1 \end{bmatrix} = \hat{B} = \hat{R}\hat{R}^* = \begin{bmatrix} c & a \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \bar{c} & 0 \\ \bar{a} & 1 \end{bmatrix} = \begin{bmatrix} |a|^2 + |c|^2 & a \\ \bar{a} & 1 \end{bmatrix}.$$

The Derivation of the RR^*J Algorithm

Instead of LL^*, one can use RR^* factorization of $\hat{B}$. Then we have

$$\begin{bmatrix} 1 & b_{ij} \\ \bar{b}_{ij} & 1 \end{bmatrix} = \hat{B} = \hat{R}\hat{R}^* = \begin{bmatrix} c & a \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \bar{c} & 0 \\ \bar{a} & 1 \end{bmatrix} = \begin{bmatrix} |a|^2 + |c|^2 & a \\ \bar{a} & 1 \end{bmatrix}.$$

Assuming positive c, one obtains $a = b_{ij}$, $c = \sqrt{1 - |b_{ij}|^2} = \tau$. Hence

The Derivation of the RR^*J Algorithm

Instead of LL^*, one can use RR^* factorization of $\hat{B}$. Then we have

$$\begin{bmatrix} 1 & b_{ij} \\ \bar{b}_{ij} & 1 \end{bmatrix} = \hat{B} = \hat{R}\hat{R}^* = \begin{bmatrix} c & a \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \bar{c} & 0 \\ \bar{a} & 1 \end{bmatrix} = \begin{bmatrix} |a|^2 + |c|^2 & a \\ \bar{a} & 1 \end{bmatrix}.$$

Assuming positive c, one obtains $a = b_{ij}$, $c = \sqrt{1 - |b_{ij}|^2} = \tau$. Hence

$$\hat{R} = \begin{bmatrix} \tau & b_{ij} \\ 0 & 1 \end{bmatrix}, \qquad \hat{R}^{-1} = \frac{1}{\tau} \begin{bmatrix} 1 & -b_{ij} \\ 0 & \tau \end{bmatrix}, \qquad \hat{R}^{-*} = \frac{1}{\tau} \begin{bmatrix} 1 & 0 \\ -\bar{b}_{ij} & \tau \end{bmatrix}.$$

The Derivation of the RR^*J Algorithm

Instead of LL^*, one can use RR^* factorization of $\hat{B}$. Then we have

$$\begin{bmatrix} 1 & b_{ij} \\ \bar{b}_{ij} & 1 \end{bmatrix} = \hat{B} = \hat{R}\hat{R}^* = \begin{bmatrix} c & a \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \bar{c} & 0 \\ \bar{a} & 1 \end{bmatrix} = \begin{bmatrix} |a|^2 + |c|^2 & a \\ \bar{a} & 1 \end{bmatrix}.$$

Assuming positive c, one obtains $a = b_{ij}$, $c = \sqrt{1 - |b_{ij}|^2} = \tau$. Hence

$$\hat{R} = \begin{bmatrix} \tau & b_{ij} \\ 0 & 1 \end{bmatrix}, \qquad \hat{R}^{-1} = \frac{1}{\tau} \begin{bmatrix} 1 & -b_{ij} \\ 0 & \tau \end{bmatrix}, \qquad \hat{R}^{-*} = \frac{1}{\tau} \begin{bmatrix} 1 & 0 \\ -\bar{b}_{ij} & \tau \end{bmatrix}.$$

If we write $\hat{F}_2 = \hat{R}^{-*}$, then $\hat{F}_2^* \hat{B} \hat{F}_2 = \hat{R}^{-1} \hat{B} \hat{R}^{-*} = I_2$ and we have

The Derivation of the RR^*J Algorithm

$$\hat{F}_2^* \hat{A} \hat{F}_2 = \begin{bmatrix} a_{ii} - \frac{a_{ij}\bar{b}_{ij} + \bar{a}_{ij} b_{ij} - (a_{ii}+a_{jj})|b_{ij}|^2}{\tau^2} & (a_{ij} - a_{jj} b_{ij})/\tau \\ (\bar{a}_{ij} - a_{jj}\bar{b}_{ij})/\tau & a_{jj} \end{bmatrix}.$$

The Derivation of the RR^*J Algorithm

$$\hat{F}_2^* \hat{A} \hat{F}_2 = \begin{bmatrix} a_{ii} - \frac{a_{ij}\bar{b}_{ij} + \bar{a}_{ij} b_{ij} - (a_{ii}+a_{jj})|b_{ij}|^2}{\tau^2} & (a_{ij} - a_{jj} b_{ij})/\tau \\ (\bar{a}_{ij} - a_{jj}\bar{b}_{ij})/\tau & a_{jj} \end{bmatrix} .$$

- The final transformation is $\hat{F} = \hat{F}_2 \hat{R}_2$,

The Derivation of the RR^*J Algorithm

$$\hat{F}_2^* \hat{A} \hat{F}_2 = \begin{bmatrix} a_{ii} - \frac{a_{ij}\bar{b}_{ij} + \bar{a}_{ij} b_{ij} - (a_{ii}+a_{jj})|b_{ij}|^2}{\tau^2} & (a_{ij} - a_{jj} b_{ij})/\tau \\ (\bar{a}_{ij} - a_{jj}\bar{b}_{ij})/\tau & a_{jj} \end{bmatrix} .$$

- The final transformation is $\hat{F} = \hat{F}_2 \hat{R}_2$,
- $\hat{R}_2$ is the Jacobi rotation which annihilates $(1,2)$-element of $\hat{F}_2^* \hat{A} \hat{F}_2$

The Derivation of the RR^*J Algorithm

$$\hat{F}_2^* \hat{A} \hat{F}_2 = \begin{bmatrix} a_{ii} - \frac{a_{ij}\bar{b}_{ij} + \bar{a}_{ij} b_{ij} - (a_{ii}+a_{jj})|b_{ij}|^2}{\tau^2} & (a_{ij} - a_{jj} b_{ij})/\tau \\ (\bar{a}_{ij} - a_{jj}\bar{b}_{ij})/\tau & a_{jj} \end{bmatrix} .$$

- The final transformation is $\hat{F} = \hat{F}_2 \hat{R}_2$,
- $\hat{R}_2$ is the Jacobi rotation which annihilates $(1,2)$-element of $\hat{F}_2^* \hat{A} \hat{F}_2$
- Let $(1,2)$-element of $\hat{R}_2$ be $-e^{\imath\epsilon_2} \sin\vartheta_2$

The Derivation of the RR^*J Algorithm

$$\hat{F}_2^* \hat{A} \hat{F}_2 = \begin{bmatrix} a_{ii} - \frac{a_{ij}\bar{b}_{ij} + \bar{a}_{ij} b_{ij} - (a_{ii}+a_{jj})|b_{ij}|^2}{\tau^2} & (a_{ij} - a_{jj} b_{ij})/\tau \\ (\bar{a}_{ij} - a_{jj}\bar{b}_{ij})/\tau & a_{jj} \end{bmatrix} .$$

- The final transformation is $\hat{F} = \hat{F}_2 \hat{R}_2$,
- $\hat{R}_2$ is the Jacobi rotation which annihilates $(1,2)$-element of $\hat{F}_2^* \hat{A} \hat{F}_2$
- Let $(1,2)$-element of $\hat{R}_2$ be $-e^{\imath\epsilon_2} \sin\vartheta_2$

Then the parameters ϵ_2 and ϑ_2 are determined by the formulas

$$\begin{aligned} \epsilon_2 &= \arg(a_{ij} - b_{ij} a_{jj}), \\ \tan(2\vartheta_2) &= \frac{2|a_{ij} - a_{jj} b_{ij}| \sqrt{1 - |b_{ij}|^2}}{a_{ii} - a_{jj} - (a_{ij}\bar{b}_{ij} + \bar{a}_{ij} b_{ij}) + 2a_{jj}|b_{ij}|^2}, \quad -\frac{\pi}{4} \le \vartheta_2 \le \frac{\pi}{4}. \end{aligned}$$

The Derivation of the RR^*J Algorithm

The transformation formulas for the diagonal elements of A:

$$
\begin{aligned}
a'_{ii} &= a_{ii} - \frac{a_{ij}\bar{b}_{ij} + \bar{a}_{ij}b_{ij} - (a_{ii}+a_{jj})|b_{ij}|^2}{1-|b_{ij}|^2} + \tan\vartheta_2 \cdot \frac{|a_{ij} - a_{jj}b_{ij}|}{\sqrt{1-|b_{ij}|^2}}, \\
a'_{jj} &= a_{jj} - \tan\vartheta_2 \cdot \frac{|a_{ij} - a_{jj}b_{ij}|}{\sqrt{1-|b_{ij}|^2}}.
\end{aligned}
$$

The Derivation of the RR^*J Algorithm

The transformation formulas for the diagonal elements of A:

$$
\begin{aligned}
a'_{ii} &= a_{ii} - \frac{a_{ij}\bar{b}_{ij} + \bar{a}_{ij}b_{ij} - (a_{ii}+a_{jj})|b_{ij}|^2}{1-|b_{ij}|^2} + \tan\vartheta_2 \cdot \frac{|a_{ij} - a_{jj}b_{ij}|}{\sqrt{1-|b_{ij}|^2}}, \\
a'_{jj} &= a_{jj} - \tan\vartheta_2 \cdot \frac{|a_{ij} - a_{jj}b_{ij}|}{\sqrt{1-|b_{ij}|^2}}.
\end{aligned}
$$

If $a_{ii} = a_{jj}$, $a_{ij} = a_{jj}b_{ij}$, ϑ_2 is not well defined and we choose $\vartheta_2 = 0$.

The Derivation of the RR^*J Algorithm

The transformation formulas for the diagonal elements of A:

$$
\begin{aligned}
a'_{ii} &= a_{ii} - \frac{a_{ij}\bar{b}_{ij} + \bar{a}_{ij}b_{ij} - (a_{ii}+a_{jj})|b_{ij}|^2}{1-|b_{ij}|^2} + \tan\vartheta_2 \cdot \frac{|a_{ij} - a_{jj}b_{ij}|}{\sqrt{1-|b_{ij}|^2}}, \\
a'_{jj} &= a_{jj} - \tan\vartheta_2 \cdot \frac{|a_{ij} - a_{jj}b_{ij}|}{\sqrt{1-|b_{ij}|^2}}.
\end{aligned}
$$

If $a_{ii} = a_{jj}$, $a_{ij} = a_{jj}b_{ij}$, ϑ_2 is not well defined and we choose $\vartheta_2 = 0$.

In that case a'_{ii} and a'_{jj} reduce to a_{ii} and a_{jj}, respectively.

The Derivation of the RR^*J Algorithm

Let $c_{\vartheta_2} = \cos\vartheta_2$, $s^{\pm}_{\vartheta_2} = e^{\pm\imath\epsilon_2}\sin\vartheta_2$. Then

$$
\begin{aligned}
\hat{F} &= \frac{1}{\sqrt{1-|b_{ij}|^2}} \left[\begin{array}{cc} 1 & 0 \\ -\bar{b}_{ij} & \sqrt{1-|b_{ij}|^2} \end{array}\right] \left[\begin{array}{cc} c_{\vartheta_2} & -s^{+}_{\vartheta_2} \\ s^{-}_{\vartheta_2} & c_{\vartheta_2} \end{array}\right] \\
&= \frac{1}{\sqrt{1-|b_{ij}|^2}} \left[\begin{array}{cc} c_{\vartheta_2} & -s^{+}_{\vartheta_2} \\ s_{\tilde{\vartheta}_2} & c_{\tilde{\vartheta}_2} \end{array}\right] = \left[\begin{array}{cc} c1 & -s1 \\ s2 & c2 \end{array}\right],
\end{aligned}
$$

$$
c_{\tilde{\vartheta}_2} = c_{\vartheta_2}\sqrt{1-|b_{ij}|^2} + s^{+}_{\vartheta_2}\bar{b}_{ij}, \;\; s_{\tilde{\vartheta}_2} = s^{-}_{\vartheta_2}\sqrt{1-|b_{ij}|^2} - c_{\vartheta_2}\bar{b}_{ij}, \;\; |c_{\tilde{\vartheta}}|^2 + |s_{\tilde{\vartheta}}|^2 = 1,
$$

$$
\begin{aligned}
c1 &= c_{\vartheta_2}/\sqrt{1-|b_{ij}|^2}, \qquad c2 = c_{\vartheta_2} + s^{+}_{\vartheta_2}\bar{b}_{ij}/\sqrt{1-|b_{ij}|^2}, \\
s1 &= s^{+}_{\vartheta_2}/\sqrt{1-|b_{ij}|^2}^{\,+}, \qquad s2 = s^{-}_{\vartheta_2} - c_{\vartheta_2}\bar{b}_{ij}/\sqrt{1-|b_{ij}|^2}.
\end{aligned}
$$

We can postmultiply $\hat{F}$ by $\mathrm{diag}(1\,,\, \bar{c}_{\tilde{\vartheta}_2}/|c_{\tilde{\vartheta}_2}|)$ provided that $c_{\tilde{\vartheta}_2} \neq 0$. This ensures that (the updated) $\hat{F}$ has nonnegative diagonal elements.

The Complex Cholesky-Jacobi Method

The CJ method can briefly be defined as follows:

The Complex Cholesky-Jacobi Method

The *CJ* method can briefly be defined as follows:

1. select the pivot pair (i, j)

The Complex Cholesky-Jacobi Method

The CJ method can briefly be defined as follows:

1. select the pivot pair (i, j)
2. if $a_{ii} \leq a_{jj}$ then employ the LL^*J algorithm

   else employ the RR^*J algorithm

The Complex Cholesky-Jacobi Method

The CJ method can briefly be defined as follows:

1. select the pivot pair (i, j)
2. if $a_{ii} \leq a_{jj}$ then employ the LL^*J algorithm

   else employ the RR^*J algorithm

Our numerical tests show that neither LL^*J nor RR^*J is indicated as a HRA algorithm on well-behaved pairs.

The Complex Cholesky-Jacobi Method

The CJ method can briefly be defined as follows:

1. select the pivot pair (i, j)
2. if $a_{ii} \le a_{jj}$ then employ the LL^*J algorithm

   else employ the RR^*J algorithm

Our numerical tests show that neither LL^*J nor RR^*J is indicated as a HRA algorithm on well-behaved pairs.

The same can be said for the hybrid algorithm that selects the LL^*J and RR^*J algorithms in the opposite way, i.e. selects the RR^*J (LL^*J) algorithm when $a_{ii} \le a_{jj}$ ($a_{ii} > a_{jj}$).

The Complex Cholesky-Jacobi Method

The *CJ* method can briefly be defined as follows:

1. select the pivot pair (i, j)
2. if $a_{ii} \leq a_{jj}$ then employ the LL^*J algorithm

   else employ the RR^*J algorithm

Our numerical tests show that neither LL^*J nor RR^*J is indicated as a HRA algorithm on well-behaved pairs.

The same can be said for the hybrid algorithm that selects the LL^*J and RR^*J algorithms in the opposite way, i.e. selects the RR^*J (LL^*J) algorithm when $a_{ii} \leq a_{jj}$ ($a_{ii} > a_{jj}$).

Only the above definition warrants the HRA of the algorithm and it is in complete agreement with the behavior of the real *CJ* method.

The Main Characteristics of the Complex CJ Method

The Main Characteristics of the Complex CJ Method

- Fast (numerical tests indicate the quadratic asymptotic convergence)

The Main Characteristics of the Complex CJ Method

- Fast (numerical tests indicate the quadratic asymptotic convergence)
- Very accurate (numerical tests indicate HRA on well-behaved pairs)

The Main Characteristics of the Complex CJ Method

- Fast (numerical tests indicate the quadratic asymptotic convergence)
- Very accurate (numerical tests indicate HRA on well-behaved pairs)
- Unit diagonal in B simplifies the algorithm and has a stabilizing effect on the iterative process, because it almost optimally reduces the condition of B and all $B^{(k)}$, $k \geq 1$. Van der Sluis, A. Numer. Math. 14 (1969)

The Main Characteristics of the Complex CJ Method

- Fast (numerical tests indicate the quadratic asymptotic convergence)
- Very accurate (numerical tests indicate HRA on well-behaved pairs)
- Unit diagonal in B simplifies the algorithm and has a stabilizing effect on the iterative process, because it almost optimally reduces the condition of B and all $B^{(k)}$, $k \geq 1$. Van der Sluis, A. Numer. Math. 14 (1969)
- No Problem with renormalizations, easy to code

The Main Characteristics of the Complex CJ Method

- Fast (numerical tests indicate the quadratic asymptotic convergence)
- Very accurate (numerical tests indicate HRA on well-behaved pairs)
- Unit diagonal in B simplifies the algorithm and has a stabilizing effect on the iterative process, because it almost optimally reduces the condition of B and all $B^{(k)}$, $k \geq 1$. Van der Sluis, A. Numer. Math. 14 (1969)
- No Problem with renormalizations, easy to code
- Some theoretical results exist (the global convergence has been proved in: E. Begović, V. Hari, Convergence of the Complex Cyclic Jacobi Methods and Applications, preprint 2018)

The Main Characteristics of the Complex CJ Method

- Fast (numerical tests indicate the quadratic asymptotic convergence)
- Very accurate (numerical tests indicate HRA on well-behaved pairs)
- Unit diagonal in B simplifies the algorithm and has a stabilizing effect on the iterative process, because it almost optimally reduces the condition of B and all $B^{(k)}$, $k \geq 1$. Van der Sluis, A. Numer. Math. 14 (1969)
- No Problem with renormalizations, easy to code
- Some theoretical results exist (the global convergence has been proved in: E. Begović, V. Hari, Convergence of the Complex Cyclic Jacobi Methods and Applications, preprint 2018)
- It requires B to be positive definite (it solves PGEP)

Theorem

Let $A = A^* \succ O$, $B = B^* \succ O$ *and* $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$, $\lambda_i \in \sigma(A, B)$.
Let $A_S = D_A^{-1/2} A D_A^{-1/2}$, $B_S = D_B^{-1/2} B D_B^{-1/2}$, $D_A = diag(A)$, $D_B = diag(B)$

Theorem

Let $A = A^* \succ O$, $B = B^* \succ O$ *and* $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$, $\lambda_i \in \sigma(A, B)$.
Let $A_S = D_A^{-1/2} A D_A^{-1/2}$, $B_S = D_B^{-1/2} B D_B^{-1/2}$, $D_A = diag(A)$, $D_B = diag(B)$

Let δA, δB *be Hermitian perturbations and* $\tilde{\lambda}_1 \geq \tilde{\lambda}_2 \geq \cdots \geq \tilde{\lambda}_n$ *the eigenvalues of* $(A + \delta A, B + \delta B)$.

Theorem

Let $A = A^* \succ O$, $B = B^* \succ O$ *and* $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$, $\lambda_i \in \sigma(A, B)$.
Let $A_S = D_A^{-1/2} A D_A^{-1/2}$, $B_S = D_B^{-1/2} B D_B^{-1/2}$, $D_A = diag(A)$, $D_B = diag(B)$

Let δA, δB *be Hermitian perturbations and* $\tilde{\lambda}_1 \geq \tilde{\lambda}_2 \geq \cdots \geq \tilde{\lambda}_n$ *the eigenvalues of* $(A + \delta A, B + \delta B)$.

Let

$$\varepsilon_{A_S} = \|(\delta A)_S\|_2 / \|A_S\|_2, \quad \varepsilon_{B_S} = \|(\delta B)_S\|_2 / \|B_S\|_2$$

where $(\delta A)_S = D_A^{-1/2} \delta A D_A^{-1/2}$, $(\delta B)_S = D_B^{-1/2} \delta B D_B^{-1/2}$.

Theorem

Let $A = A^* \succ O$, $B = B^* \succ O$ *and* $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$, $\lambda_i \in \sigma(A, B)$.
Let $A_S = D_A^{-1/2} A D_A^{-1/2}$, $B_S = D_B^{-1/2} B D_B^{-1/2}$, $D_A = diag(A)$, $D_B = diag(B)$

Let δA, δB *be Hermitian perturbations and* $\tilde{\lambda}_1 \geq \tilde{\lambda}_2 \geq \cdots \geq \tilde{\lambda}_n$ *the eigenvalues of* $(A + \delta A, B + \delta B)$.

Let

$$\varepsilon_{A_S} = \|(\delta A)_S\|_2 / \|A_S\|_2, \quad \varepsilon_{B_S} = \|(\delta B)_S\|_2 / \|B_S\|_2$$

where $(\delta A)_S = D_A^{-1/2} \delta A D_A^{-1/2}$, $(\delta B)_S = D_B^{-1/2} \delta B D_B^{-1/2}$.

If

$$\varepsilon_{A_S} \kappa_2(A_S) < 1 \quad \text{and} \quad \varepsilon_{B_S} \kappa_2(B_S) < 1,$$

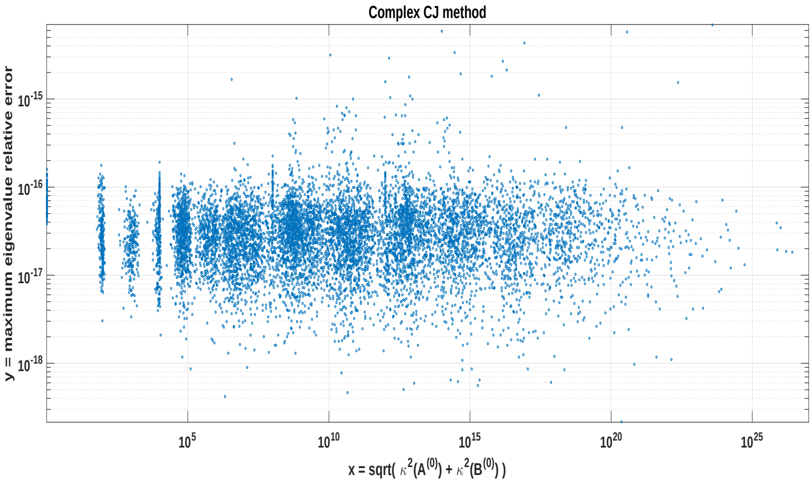
then

$$\max_{1 \leq i \leq n} \frac{|\tilde{\lambda}_i - \lambda_i|}{\lambda_i} \leq \frac{\varepsilon_{A_S} \kappa_2(A_S) + \varepsilon_{B_S} \kappa_2(B_S)}{1 - \varepsilon_{B_S} \kappa_2(B_S)} \leq \frac{\sqrt{\kappa_2(A_S)^2 + \kappa_2(B_S)^2} \sqrt{\varepsilon_{A_S}^2 + \varepsilon_{B_S}^2}}{1 - \varepsilon_{B_S} \kappa_2(B_S)}.$$

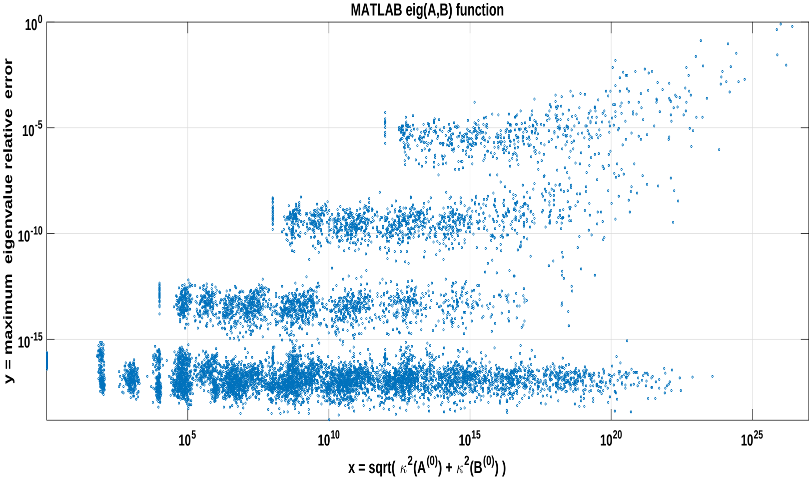
Relative Accuracy

$$
\begin{aligned}
\varrho_{(A,B)} &= \max_{1\le i\le n} \frac{|\tilde{\lambda}_i - \lambda_i|}{\lambda_i} \Big/ \sqrt{\kappa_2^2(A_S) + \kappa_2^2(B_S)} \\
\chi_{(A,B)} &= \sqrt{\kappa_2^2(A^{(0)}) + \kappa_2^2(B^{(0)})} \\
\mathcal{E} &= \{(\chi_{(A,B)}\ ,\ \varrho_{(A,B)}) :\ (A,B) \in \Upsilon\}.
\end{aligned}
$$

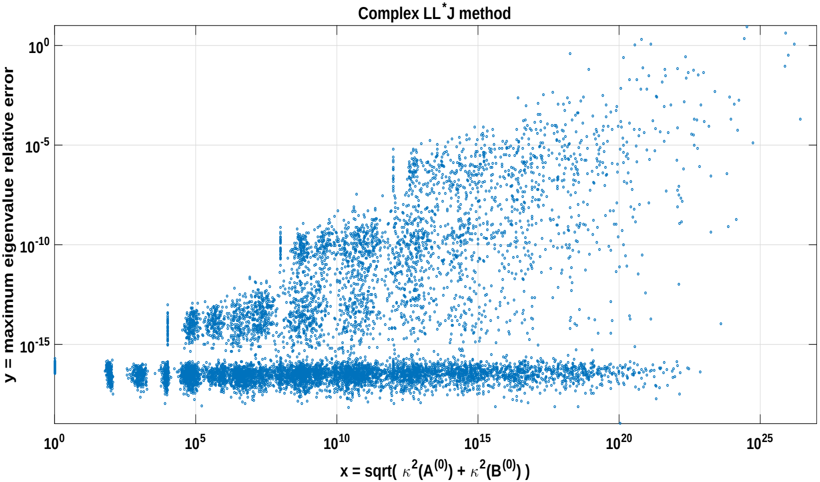
Relative Errors: *CJ* vs. MATLAB eig(A,B)



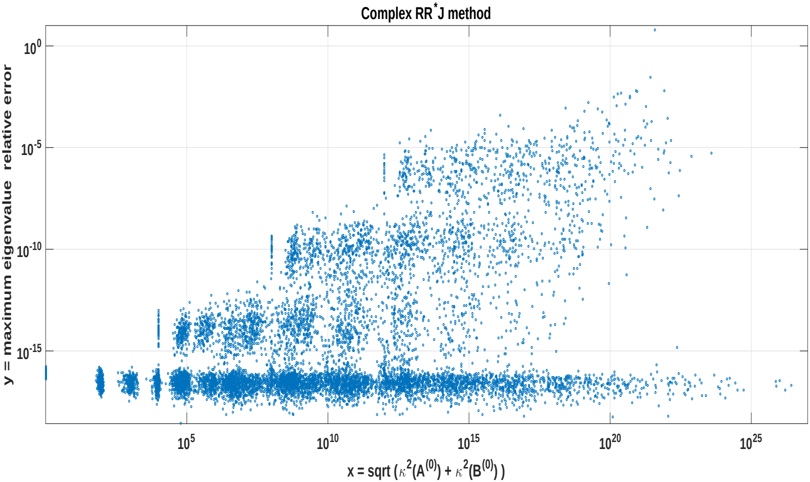
Relative Errors: *CJ* vs. MATLAB eig(A,B)



Relative Errors: $LL * J$



Relative Errors: $RR * J$



Relative Errors: Opposite Choice Than in *CJ*