

Approximation of Hermitian Matrices by Positive (Semi-)Definite Matrices using Modified LDL^H Decompositions

Joscha Reimer

jor@informatik.uni-kiel.de



March 22, 2018

Decomposition algorithm

```
function DECOMPOSITION( $A$ )  
  for  $i \leftarrow 1, \dots, n$  do  
     $d_i \leftarrow A_{ii} - \sum_{j=1}^{i-1} |L_{ij}|^2 d_j$   
    for  $j \leftarrow i + 1, \dots, n$  do  
      if  $d_i \neq 0$  then  
         $L_{ji} \leftarrow \left( A_{ji} - \sum_{k=1}^{i-1} L_{jk} \bar{L}_{ik} d_k \right) (d_i)^{-1}$   
      else  
         $L_{ji} \leftarrow 0$   
      end if  
    end for  
  end for  
   $L_{ii} \leftarrow 1$  and  $L_{ij} \leftarrow 0$  for all  $i, j \in \{1, \dots, n\}$  with  $j > i$   
  return  $L, d$   
end function
```

Input:

$A \in \mathbb{C}^{n \times n}$
Hermitian

Output:

$L \in \mathbb{C}^{n \times n}$ lower
triangle matrix
with ones on the
diagonal
 $d \in \mathbb{R}^n$

Decomposition algorithm

```
function DECOMPOSITION( $A$ )  
   $\alpha_i \leftarrow 0$  for all  $i \in \{1, \dots, n\}$   
  for  $i \leftarrow 1, \dots, n$  do  
     $d_i \leftarrow A_{ii} - \alpha_i$   
    for  $j \leftarrow i + 1, \dots, n$  do  
      if  $d_i \neq 0$  then  
         $L_{ji} \leftarrow \left( A_{ji} - \sum_{k=1}^{i-1} L_{jk} \bar{L}_{ik} d_k \right) (d_i)^{-1}$   
         $\alpha_j \leftarrow \alpha_j + |L_{ji}|^2 d_i$   
      else  
         $L_{ji} \leftarrow 0$   
      end if  
    end for  
  end for  
   $L_{ii} \leftarrow 1$  and  $L_{ij} \leftarrow 0$  for all  $i, j \in \{1, \dots, n\}$  with  $j > i$   
  return  $L, d$   
end function
```

Input:

$A \in \mathbb{C}^{n \times n}$
Hermitian

Output:

$L \in \mathbb{C}^{n \times n}$ lower
triangle matrix
with ones on the
diagonal
 $d \in \mathbb{R}^n$

Decomposition algorithm

```
function DECOMPOSITION( $A, l, u, \epsilon$ )  
   $\alpha_i \leftarrow 0$  for all  $i \in \{1, \dots, n\}$   
  for  $i \leftarrow 1, \dots, n$  do  
    select  $d_i \in \{d \in \mathbb{R} \mid d \in [l, u], |d| \notin (0, \epsilon)\}$   
    for  $j \leftarrow i + 1, \dots, n$  do  
      if  $d_i \neq 0$  then  
         $L_{ji} \leftarrow \left( A_{ji} - \sum_{k=1}^{i-1} L_{jk} \bar{L}_{ik} d_k \right) (d_i)^{-1}$   
         $\alpha_j \leftarrow \alpha_j + |L_{ji}|^2 d_i$   
      else  
         $L_{ji} \leftarrow 0$   
      end if  
    end for  
  end for  
   $L_{ii} \leftarrow 1$  and  $L_{ij} \leftarrow 0$  for all  $i, j \in \{1, \dots, n\}$  with  $j > i$   
  return  $L, d$   
end function
```

Input:

$A \in \mathbb{C}^{n \times n}$

Hermitian

$l \in \mathbb{R} \cup \{-\infty\}$

$u \in \mathbb{R} \cup \{\infty\}$

$\epsilon \geq 0$

$\max\{l, \epsilon\} \leq u$

Output:

$L \in \mathbb{C}^{n \times n}$ lower
triangle matrix
with ones on the
diagonal

$d \in \mathbb{R}^n$

Decomposition algorithm

```
function DECOMPOSITION( $A, x, y, l, u, \epsilon$ )  
   $\alpha_i \leftarrow 0$  for all  $i \in \{1, \dots, n\}$   
  for  $i \leftarrow 1, \dots, n$  do  
    select  $(\omega_i, d_i) \in \{(\omega, d) \in \mathbb{R}^2 \mid \omega \geq 0, d \in [l, u], |d| \notin$   
     $(0, \epsilon), d + \alpha_i \omega^2 \in [x_i, y_i]\}$   
    for  $j \leftarrow 1, \dots, i - 1$  do  
       $L_{ij} \leftarrow \omega_i L_{ij}$   
    end for  
     $\delta_i \leftarrow d_i + \omega_i^2 \alpha_i - A_{ii}$   
    for  $j \leftarrow i + 1, \dots, n$  do  
      if  $d_i \neq 0$  then  
         $L_{ji} \leftarrow \left( A_{ji} - \sum_{k=1}^{i-1} L_{jk} \bar{L}_{ik} d_k \right) (d_i)^{-1}$   
         $\alpha_j \leftarrow \alpha_j + |L_{ji}|^2 d_i$   
      else  
         $L_{ji} \leftarrow 0$   
      end if  
    end for  
  end for  
   $L_{ii} \leftarrow 1$  and  $L_{ij} \leftarrow 0$  for all  $i, j \in \{1, \dots, n\}$  with  $j > i$   
  return  $L, d, \omega, \delta$   
end function
```

Input:

$A \in \mathbb{C}^{n \times n}$

Hermitian

$x \in (\mathbb{R} \cup \{-\infty\})^n$

$y \in (\mathbb{R} \cup \{\infty\})^n$

$l \in \mathbb{R} \cup \{-\infty\}$

$u \in \mathbb{R} \cup \{\infty\}$

$\epsilon \geq 0$

$\max\{x_i, l, \epsilon\} \leq$
 $\min\{y_i, u\}$ for all
 $i \in \{1, \dots, n\}$

Output:

$L \in \mathbb{C}^{n \times n}$ lower
triangle matrix
with ones on the
diagonal

$d, \omega, \delta \in \mathbb{R}^n$

Decomposition algorithm

```
function DECOMPOSITION( $A, x, y, l, u, \epsilon$ )  
   $\alpha_i \leftarrow 0$ ,  $p_i \leftarrow i$  for all  $i \in \{1, \dots, n\}$   
  for  $i \leftarrow 1, \dots, n$  do  
    select  $j \in \{i, \dots, n\}$  and swap  $p_i$  and  $p_j$ , swap  $L_{ik}$  and  $L_{jk}$  for  
    all  $k \in \{1, \dots, i-1\}$   
    select  $(\omega, d_i) \in \{(\omega, d) \in \mathbb{R}^2 \mid \omega \geq 0, d \in [l, u], |d| \notin$   
     $(0, \epsilon), d + \alpha_{p_i} \omega^2 \in [x_{p_i}, y_{p_i}]\}$   
     $\delta_{p_i} \leftarrow d_i + \omega_{p_i}^2 \alpha_{p_i} - A_{p_i p_i}$   
    for  $j \leftarrow 1, \dots, i-1$  do  
       $L_{ij} \leftarrow \omega_{p_i} L_{ij}$   
    end for  
    for  $j \leftarrow i+1, \dots, n$  do  
      if  $d_i \neq 0$  then  
         $L_{ji} \leftarrow \left( A_{p_j p_i} - \sum_{k=1}^{i-1} L_{jk} \bar{L}_{ik} d_k \right) (d_i)^{-1}$   
         $\alpha_{p_j} \leftarrow \alpha_{p_j} + |L_{ji}|^2 d_i$   
      else  
         $L_{ji} \leftarrow 0$   
      end if  
    end for  
  end for  
   $L_{ii} \leftarrow 1$  and  $L_{ij} \leftarrow 0$  for all  $i, j \in \{1, \dots, n\}$  with  $j > i$   
  return  $L, d, p, \omega, \delta$   
end function
```

Input:

$A \in \mathbb{C}^{n \times n}$

Hermitian

$x \in (\mathbb{R} \cup \{-\infty\})^n$

$y \in (\mathbb{R} \cup \{\infty\})^n$

$l \in \mathbb{R} \cup \{-\infty\}$

$u \in \mathbb{R} \cup \{\infty\}$

$\epsilon \geq 0$

$\max\{x_i, l, \epsilon\} \leq$
 $\min\{y_i, u\}$ for all
 $i \in \{1, \dots, n\}$

Output:

$L \in \mathbb{C}^{n \times n}$ lower
triangle matrix
with ones on the
diagonal

$d, \omega, \delta \in \mathbb{R}^n$

$p \in \{1, \dots, n\}^n$

Decomposition algorithm

```
function DECOMPOSITION( $A, x, y, l, u, \epsilon$ )  
   $\alpha_i \leftarrow 0$ ,  $p_i \leftarrow i$  for all  $i \in \{1, \dots, n\}$   
  for  $i \leftarrow 1, \dots, n$  do  
    select  $j \in \{i, \dots, n\}$  and swap  $p_i$  and  $p_j$ , swap  $L_{ik}$  and  $L_{jk}$  for  
    all  $k \in \{1, \dots, i-1\}$   
    select  $(\omega_{p_i}, d_i) \in \{(\omega, d) \in \mathbb{R}^2 \mid \omega \geq 0, d \in [l, u], |d| \notin$   
     $(0, \epsilon), d + \alpha_{p_i} \omega^2 \in [x_{p_i}, y_{p_i}]\}$   
     $\delta_{p_i} \leftarrow d_i + \omega_{p_i}^2 \alpha_{p_i} - A_{p_i p_i}$   
    for  $j \leftarrow 1, \dots, i-1$  do  
       $L_{ij} \leftarrow \omega_{p_i} L_{ij}$   
    end for  
    for  $j \leftarrow i+1, \dots, n$  do  
      if  $d_i \neq 0$  then  
        
$$L_{ji} \leftarrow \left( A_{p_j p_i} - \sum_{k=1}^{i-1} L_{jk} \bar{L}_{ik} d_k \right) (d_i)^{-1}$$
  
         $\alpha_{p_j} \leftarrow \alpha_{p_j} + |L_{ji}|^2 d_i$   
      else  
         $L_{ji} \leftarrow 0$   
      end if  
    end for  
  end for  
   $L_{ii} \leftarrow 1$  and  $L_{ij} \leftarrow 0$  for all  $i, j \in \{1, \dots, n\}$  with  $j > i$   
  return  $L, d, p, \omega, \delta$   
end function
```

Input:

$A \in \mathbb{C}^{n \times n}$

Hermitian

$x \in (\mathbb{R} \cup \{-\infty\})^n$

$y \in (\mathbb{R} \cup \{\infty\})^n$

$l \in \mathbb{R} \cup \{-\infty\}$

$u \in \mathbb{R} \cup \{\infty\}$

$\epsilon \geq 0$

$\max\{x_i, l, \epsilon\} \leq$
 $\min\{y_i, u\}$ for all
 $i \in \{1, \dots, n\}$

Output:

$L \in \mathbb{C}^{n \times n}$ lower
triangle matrix
with ones on the
diagonal

$d, \omega, \delta \in \mathbb{R}^n$

$p \in \{1, \dots, n\}^n$

Matrix algorithm

Matrix algorithm

Define

$$\text{MATRIX}(A, x, y, l, u, \epsilon) := P^T LDL^H P$$

where

$$L, d, p, \omega, \delta := \text{DECOMPOSITION}(A, x, y, l, u, \epsilon),$$

$$D := \text{diag}(d) \quad \text{and} \quad P_{ij} := \begin{cases} 1 & \text{if } j = p_i \\ 0 & \text{else} \end{cases} \quad \text{for all } i, j \in \{1, \dots, n\}$$

for all valid inputs A, x, y, l, u, ϵ of the algorithm.

Definiteness

Let be

$$B := \text{MATRIX}(A, x, y, l, u, \epsilon)$$

where A, x, y, l, u, ϵ is some valid input for the algorithm.

Definiteness

Let be

$$B := \text{MATRIX}(A, x, y, l, u, \epsilon)$$

where A, x, y, l, u, ϵ is some valid input for the algorithm.

- ▶ *B is positive semidefinite if $l \geq 0$.*

Definiteness

Let be

$$B := \text{MATRIX}(A, x, y, l, u, \epsilon)$$

where A, x, y, l, u, ϵ is some valid input for the algorithm.

- ▶ *B is positive semidefinite if $l \geq 0$.*
- ▶ *B is positive definite if $l > 0$.*

Diagonal values

Let be

$$B := \text{MATRIX}(A, x, y, l, u, \epsilon)$$

where A, x, y, l, u, ϵ is some valid input for the algorithm.

Diagonal values

Let be

$$B := \text{MATRIX}(A, x, y, l, u, \epsilon)$$

where A, x, y, l, u, ϵ is some valid input for the algorithm. It holds

$$x_i \leq B_{ii} \leq y_i \text{ for all } i \in \{1, \dots, n\}.$$

Condition number

Let be

$$B := \text{MATRIX}(A, x, y, l, u, \epsilon)$$

where A, x, y, l, u, ϵ is some valid input for the algorithm with $l > 0$.

Condition number

Let be

$$B := \text{MATRIX}(A, x, y, l, u, \epsilon)$$

where A, x, y, l, u, ϵ is some valid input for the algorithm with $l > 0$. It holds

$$\kappa_2(B) \leq 4 \frac{a^n b}{l^{n+1}}$$

$$\text{with } a := \frac{1}{n} \sum_{i=1}^n y_i \text{ and } b := \min\{u, \max_{i=1, \dots, n} y_i\}.$$

Relation between A and B

Let be

$$B := \text{MATRIX}(A, x, y, l, u, \epsilon)$$

where A, x, y, l, u, ϵ is some valid input for the algorithm with $l > 0$ and

$$L, d, p, \omega, \delta := \text{DECOMPOSITION}(A, x, y, l, u, \epsilon).$$

Relation between A and B

Let be

$$B := \text{MATRIX}(A, x, y, l, u, \epsilon)$$

where A, x, y, l, u, ϵ is some valid input for the algorithm with $l > 0$ and

$$L, d, p, \omega, \delta := \text{DECOMPOSITION}(A, x, y, l, u, \epsilon).$$

It holds

$$B_{ii} = A_{ii} + \delta_i \text{ for all } i \in \{1, \dots, n\}$$

Relation between A and B

Let be

$$B := \text{MATRIX}(A, x, y, l, u, \epsilon)$$

where A, x, y, l, u, ϵ is some valid input for the algorithm with $l > 0$ and

$$L, d, p, \omega, \delta := \text{DECOMPOSITION}(A, x, y, l, u, \epsilon).$$

It holds

$$B_{ii} = A_{ii} + \delta_i \text{ for all } i \in \{1, \dots, n\}$$

and

$$B_{ij} = A_{ij}\omega_k \text{ for all } i, j \in \{1, \dots, n\} \text{ with } i \neq j \text{ and}$$
$$q_{p_i} := i \text{ for all } i \in \{1, \dots, n\}, k := \begin{cases} i & \text{if } q_i > q_j \\ j & \text{else} \end{cases} .$$

Difference

Let be

$$B := \text{MATRIX}(A, x, y, l, u, \epsilon)$$

where A, x, y, l, u, ϵ is some valid input for the algorithm with $l > 0$ and

$$L, d, p, \omega, \delta := \text{DECOMPOSITION}(A, x, y, l, u, \epsilon).$$

Difference

Let be

$$B := \text{MATRIX}(A, x, y, l, u, \epsilon)$$

where A, x, y, l, u, ϵ is some valid input for the algorithm with $l > 0$ and

$$L, d, p, \omega, \delta := \text{DECOMPOSITION}(A, x, y, l, u, \epsilon).$$

It holds

$$\|B - A\|_F^2 = \sum_{i=1}^n \left((d_i + \omega_{p_i}^2 \alpha_{p_i} - A_{p_i p_i})^2 + 2(\omega_{p_i} - 1)^2 \sum_{j=1}^{i-1} |A_{p_i p_j}|^2 \right)$$

where α is the internal variable of the algorithm
DECOMPOSITION.

Select ω and d

Select ω and d

Select

$$(\omega_{p_i}, d_i) := \arg \min_{(\omega, d) \in \Lambda_i} (d + \omega^2 \alpha_{p_i} - A_{p_i p_i})^2 + 2(\omega - 1)^2 \sum_{j=1}^{i-1} |A_{p_i p_j}|^2$$

Select ω and d

Select

$$(\omega_{p_i}, d_i) := \arg \min_{(\omega, d) \in \Lambda_i} (d + \omega^2 \alpha_{p_i} - A_{p_i p_i})^2 + 2(\omega - 1)^2 \sum_{j=1}^{i-1} |A_{p_i p_j}|^2$$

with

$$\Lambda_i := \{(\omega, d) \in \mathbb{R}^2 \mid \omega \geq 0, d \in [l, u], |d| \notin (0, \epsilon), d + \alpha_{p_i} \omega^2 \in [x_{p_i}, y_{p_i}]\}$$

in each iteration $i \in \{1, \dots, n\}$.

Select p

Select p

Select

$$(p_i, \omega_{p_i}, d_i) := \arg \min_{(p_i, \omega, d) \in K_i} (d + \omega^2 \alpha_{p_i} - A_{p_i p_i})^2 + 2(\omega - 1)^2 \sum_{j=1}^{i-1} |A_{p_i p_j}|^2$$

Select p

Select

$$(p_i, \omega_{p_i}, d_i) := \arg \min_{(p_i, \omega, d) \in K_i} (d + \omega^2 \alpha_{p_i} - A_{p_i p_i})^2 + 2(\omega - 1)^2 \sum_{j=1}^{i-1} |A_{p_i p_j}|^2$$

with

$$K_i := \{(p_i, \omega, d) \mid p_i \in \{1, \dots, n\} \setminus \{p_j \mid j \in \{1, \dots, i-1\}\}, (\omega, d) \in \Lambda_i\}$$

in each iteration $i \in \{1, \dots, n\}$.

Sparse matrices

Let be

$$B := \text{MATRIX}(A, x, y, l, u, \epsilon)$$

where A, x, y, l, u, ϵ is some valid input for the algorithm with $l > 0$.

Sparse matrices

Let be

$$B := \text{MATRIX}(A, x, y, l, u, \epsilon)$$

where A, x, y, l, u, ϵ is some valid input for the algorithm with $l > 0$. It holds

$B_{ij} = 0$ if $A_{ij} = 0$ for all $i, j \in \{1, \dots, n\}$ with $i \neq j$.

Sparse matrices

Let be

$$B := \text{MATRIX}(A, x, y, l, u, \epsilon)$$

where A, x, y, l, u, ϵ is some valid input for the algorithm with $l > 0$. It holds

$$B_{ij} = 0 \text{ if } A_{ij} = 0 \text{ for all } i, j \in \{1, \dots, n\} \text{ with } i \neq j.$$

Let be

$$L, d, p, \omega, \delta := \text{DECOMPOSITION}(A, x, y, l, u, \epsilon).$$

Sparse matrices

Let be

$$B := \text{MATRIX}(A, x, y, l, u, \epsilon)$$

where A, x, y, l, u, ϵ is some valid input for the algorithm with $l > 0$. It holds

$$B_{ij} = 0 \text{ if } A_{ij} = 0 \text{ for all } i, j \in \{1, \dots, n\} \text{ with } i \neq j.$$

Let be

$$L, d, p, \omega, \delta := \text{DECOMPOSITION}(A, x, y, l, u, \epsilon).$$

- ▶ L might be less sparse than A .

Sparse matrices

Let be

$$B := \text{MATRIX}(A, x, y, l, u, \epsilon)$$

where A, x, y, l, u, ϵ is some valid input for the algorithm with $l > 0$. It holds

$$B_{ij} = 0 \text{ if } A_{ij} = 0 \text{ for all } i, j \in \{1, \dots, n\} \text{ with } i \neq j.$$

Let be

$$L, d, p, \omega, \delta := \text{DECOMPOSITION}(A, x, y, l, u, \epsilon).$$

- ▶ L might be less sparse than A .
- ▶ p can be selected so that the sparsity of L is increased.

Numerical stability

Let be

$$L, d, p, \omega, \delta := \text{DECOMPOSITION}(A, x, y, l, u, \epsilon)$$

where A, x, y, l, u, ϵ is some valid input for the algorithm with $l \geq 0$, $\epsilon > 0$ and $\bar{y} := \max_{i \in \{1, \dots, n\}} y_i$.

Numerical stability

Let be

$$L, d, p, \omega, \delta := \text{DECOMPOSITION}(A, x, y, l, u, \epsilon)$$

where A, x, y, l, u, ϵ is some valid input for the algorithm with $l \geq 0, \epsilon > 0$ and $\bar{y} := \max_{i \in \{1, \dots, n\}} y_i$. Then it holds

$$|d_i| \leq \bar{y} \quad \text{and} \quad |L_{ij}|^2 \leq \frac{\bar{y}}{\epsilon} \quad \text{for all } i, j \in \{1, \dots, n\}.$$

Numerical stability

Let be

$$L, d, p, \omega, \delta := \text{DECOMPOSITION}(A, x, y, l, u, \epsilon)$$

where A, x, y, l, u, ϵ is some valid input for the algorithm with $l \geq 0$, $\epsilon > 0$ and $\bar{y} := \max_{i \in \{1, \dots, n\}} y_i$. Then it holds

$$|d_i| \leq \bar{y} \quad \text{and} \quad |L_{ij}|^2 \leq \frac{\bar{y}}{\epsilon} \quad \text{for all } i, j \in \{1, \dots, n\}.$$

Let be

$$B := \text{MATRIX}(A, x, y, l, u, \epsilon)$$

where A, x, y, l, u, ϵ is some valid input for the algorithm with $l \geq 0$ and $\bar{y} := \max_{i \in \{1, \dots, n\}} y_i$.

Numerical stability

Let be

$$L, d, p, \omega, \delta := \text{DECOMPOSITION}(A, x, y, l, u, \epsilon)$$

where A, x, y, l, u, ϵ is some valid input for the algorithm with $l \geq 0$, $\epsilon > 0$ and $\bar{y} := \max_{i \in \{1, \dots, n\}} y_i$. Then it holds

$$|d_i| \leq \bar{y} \quad \text{and} \quad |L_{ij}|^2 \leq \frac{\bar{y}}{\epsilon} \quad \text{for all } i, j \in \{1, \dots, n\}.$$

Let be

$$B := \text{MATRIX}(A, x, y, l, u, \epsilon)$$

where A, x, y, l, u, ϵ is some valid input for the algorithm with $l \geq 0$ and $\bar{y} := \max_{i \in \{1, \dots, n\}} y_i$. Then it holds

$$|B_{ij}| \leq \bar{y} \quad \text{for all } i, j \in \{1, \dots, n\}.$$

Invariance

Let be

$$B := \text{MATRIX}(A, x, y, l, u, \epsilon)$$

where A, x, y, l, u, ϵ is some valid input for the algorithm and

$$L, d, p, \omega, \delta := \text{DECOMPOSITION}(A, x, y, l, u, \epsilon).$$

Invariance

Let be

$$B := \text{MATRIX}(A, x, y, l, u, \epsilon)$$

where A, x, y, l, u, ϵ is some valid input for the algorithm and

$$L, d, p, \omega, \delta := \text{DECOMPOSITION}(A, x, y, l, u, \epsilon).$$

It holds $B = A$ if

$$x_i \leq A_{ii} \leq y_i \text{ for all } i \in \{1, \dots, n\}$$

holds and PAP^T has a LDL^H decomposition with

$$l \leq D_{ii} \leq u \text{ and } |D_{ii}| \notin (0, \epsilon) \text{ for all } i \in \{1, \dots, n\}$$

$$\text{and } P_{ij} = \begin{cases} 1 & \text{if } j = p_i \\ 0 & \text{else} \end{cases} \quad \text{for all } i, j \in \{1, \dots, n\}.$$

Complexity

Complexity

Let $T_U(n)$ be the worst case number of needed basic operations to calculate an unmodified LDL^H decomposition for all $A \in \mathbb{C}^{n \times n}$ which have a LDL^H decomposition

Complexity

Let $T_U(n)$ be the worst case number of needed basic operations to calculate an unmodified LDL^H decomposition for all $A \in \mathbb{C}^{n \times n}$ which have a LDL^H decomposition

$$T_U(n) = \mathcal{O}(n^3)$$

Complexity

Let $T_U(n)$ be the worst case number of needed basic operations to calculate an unmodified LDL^H decomposition for all $A \in \mathbb{C}^{n \times n}$ which have a LDL^H decomposition and $S_U(n)$ the associated worst case number of needed memory cells.

$$T_U(n) = \mathcal{O}(n^3)$$

Complexity

Let $T_U(n)$ be the worst case number of needed basic operations to calculate an unmodified LDL^H decomposition for all $A \in \mathbb{C}^{n \times n}$ which have a LDL^H decomposition and $S_U(n)$ the associated worst case number of needed memory cells.

$$T_U(n) = \mathcal{O}(n^3) \quad S_U(n) = \mathcal{O}(n^2)$$

Complexity

Let $T_U(n)$ be the worst case number of needed basic operations to calculate an unmodified LDL^H decomposition for all $A \in \mathbb{C}^{n \times n}$ which have a LDL^H decomposition and $S_U(n)$ the associated worst case number of needed memory cells.

$$T_U(n) = \mathcal{O}(n^3) \quad S_U(n) = \mathcal{O}(n^2)$$

Let $T_D(n)$ be the worst case number of basic operations to calculate $\text{DECOMPOSITION}(A, x, y, l, u, \epsilon)$ for all valid inputs A, x, y, l, u, ϵ with $A \in \mathbb{C}^{n \times n}$

Complexity

Let $T_U(n)$ be the worst case number of needed basic operations to calculate an unmodified LDL^H decomposition for all $A \in \mathbb{C}^{n \times n}$ which have a LDL^H decomposition and $S_U(n)$ the associated worst case number of needed memory cells.

$$T_U(n) = \mathcal{O}(n^3) \quad S_U(n) = \mathcal{O}(n^2)$$

Let $T_D(n)$ be the worst case number of basic operations to calculate $\text{DECOMPOSITION}(A, x, y, l, u, \epsilon)$ for all valid inputs A, x, y, l, u, ϵ with $A \in \mathbb{C}^{n \times n}$

$$T_D(n) = T_U(n) + \mathcal{O}(n^2)$$

Complexity

Let $T_U(n)$ be the worst case number of needed basic operations to calculate an unmodified LDL^H decomposition for all $A \in \mathbb{C}^{n \times n}$ which have a LDL^H decomposition and $S_U(n)$ the associated worst case number of needed memory cells.

$$T_U(n) = \mathcal{O}(n^3) \quad S_U(n) = \mathcal{O}(n^2)$$

Let $T_D(n)$ be the worst case number of basic operations to calculate $\text{DECOMPOSITION}(A, x, y, l, u, \epsilon)$ for all valid inputs A, x, y, l, u, ϵ with $A \in \mathbb{C}^{n \times n}$ and $S_D(n)$ the associated worst case number of needed memory cells.

$$T_D(n) = T_U(n) + \mathcal{O}(n^2)$$

Complexity

Let $T_U(n)$ be the worst case number of needed basic operations to calculate an unmodified LDL^H decomposition for all $A \in \mathbb{C}^{n \times n}$ which have a LDL^H decomposition and $S_U(n)$ the associated worst case number of needed memory cells.

$$T_U(n) = \mathcal{O}(n^3) \quad S_U(n) = \mathcal{O}(n^2)$$

Let $T_D(n)$ be the worst case number of basic operations to calculate $\text{DECOMPOSITION}(A, x, y, l, u, \epsilon)$ for all valid inputs A, x, y, l, u, ϵ with $A \in \mathbb{C}^{n \times n}$ and $S_D(n)$ the associated worst case number of needed memory cells.

$$T_D(n) = T_U(n) + \mathcal{O}(n^2) \quad S_D(n) = S_U(n) + \mathcal{O}(n)$$

Complexity

Let $T_U(n)$ be the worst case number of needed basic operations to calculate an unmodified LDL^H decomposition for all $A \in \mathbb{C}^{n \times n}$ which have a LDL^H decomposition and $S_U(n)$ the associated worst case number of needed memory cells.

$$T_U(n) = \mathcal{O}(n^3) \quad S_U(n) = \mathcal{O}(n^2)$$

Let $T_D(n)$ be the worst case number of basic operations to calculate $\text{DECOMPOSITION}(A, x, y, l, u, \epsilon)$ for all valid inputs A, x, y, l, u, ϵ with $A \in \mathbb{C}^{n \times n}$ and $S_D(n)$ the associated worst case number of needed memory cells.

$$T_D(n) = T_U(n) + \mathcal{O}(n^2) \quad S_D(n) = S_U(n) + \mathcal{O}(n)$$

Let $T_M(n)$ be the worst case number of basic operations to calculate $\text{MATRIX}(A, x, y, l, u, \epsilon)$ for all valid inputs A, x, y, l, u, ϵ with $A \in \mathbb{C}^{n \times n}$

Complexity

Let $T_U(n)$ be the worst case number of needed basic operations to calculate an unmodified LDL^H decomposition for all $A \in \mathbb{C}^{n \times n}$ which have a LDL^H decomposition and $S_U(n)$ the associated worst case number of needed memory cells.

$$T_U(n) = \mathcal{O}(n^3) \quad S_U(n) = \mathcal{O}(n^2)$$

Let $T_D(n)$ be the worst case number of basic operations to calculate $\text{DECOMPOSITION}(A, x, y, l, u, \epsilon)$ for all valid inputs A, x, y, l, u, ϵ with $A \in \mathbb{C}^{n \times n}$ and $S_D(n)$ the associated worst case number of needed memory cells.

$$T_D(n) = T_U(n) + \mathcal{O}(n^2) \quad S_D(n) = S_U(n) + \mathcal{O}(n)$$

Let $T_M(n)$ be the worst case number of basic operations to calculate $\text{MATRIX}(A, x, y, l, u, \epsilon)$ for all valid inputs A, x, y, l, u, ϵ with $A \in \mathbb{C}^{n \times n}$

$$T_M(n) = T_D(n) + \mathcal{O}(n^2)$$

Complexity

Let $T_U(n)$ be the worst case number of needed basic operations to calculate an unmodified LDL^H decomposition for all $A \in \mathbb{C}^{n \times n}$ which have a LDL^H decomposition and $S_U(n)$ the associated worst case number of needed memory cells.

$$T_U(n) = \mathcal{O}(n^3) \quad S_U(n) = \mathcal{O}(n^2)$$

Let $T_D(n)$ be the worst case number of basic operations to calculate $\text{DECOMPOSITION}(A, x, y, l, u, \epsilon)$ for all valid inputs A, x, y, l, u, ϵ with $A \in \mathbb{C}^{n \times n}$ and $S_D(n)$ the associated worst case number of needed memory cells.

$$T_D(n) = T_U(n) + \mathcal{O}(n^2) \quad S_D(n) = S_U(n) + \mathcal{O}(n)$$

Let $T_M(n)$ be the worst case number of basic operations to calculate $\text{MATRIX}(A, x, y, l, u, \epsilon)$ for all valid inputs A, x, y, l, u, ϵ with $A \in \mathbb{C}^{n \times n}$ and $S_M(n)$ the associated worst case number of needed memory cells.

$$T_M(n) = T_D(n) + \mathcal{O}(n^2)$$

Complexity

Let $T_U(n)$ be the worst case number of needed basic operations to calculate an unmodified LDL^H decomposition for all $A \in \mathbb{C}^{n \times n}$ which have a LDL^H decomposition and $S_U(n)$ the associated worst case number of needed memory cells.

$$T_U(n) = \mathcal{O}(n^3) \quad S_U(n) = \mathcal{O}(n^2)$$

Let $T_D(n)$ be the worst case number of basic operations to calculate $\text{DECOMPOSITION}(A, x, y, l, u, \epsilon)$ for all valid inputs A, x, y, l, u, ϵ with $A \in \mathbb{C}^{n \times n}$ and $S_D(n)$ the associated worst case number of needed memory cells.

$$T_D(n) = T_U(n) + \mathcal{O}(n^2) \quad S_D(n) = S_U(n) + \mathcal{O}(n)$$

Let $T_M(n)$ be the worst case number of basic operations to calculate $\text{MATRIX}(A, x, y, l, u, \epsilon)$ for all valid inputs A, x, y, l, u, ϵ with $A \in \mathbb{C}^{n \times n}$ and $S_M(n)$ the associated worst case number of needed memory cells.

$$T_M(n) = T_D(n) + \mathcal{O}(n^2) \quad S_M(n) = S_D(n) + \mathcal{O}(n)$$

Implementation

Implementation

- ▶ implemented in Python

Implementation

- ▶ implemented in Python
- ▶ `conda install -c jore matrix-decomposition`

Implementation

- ▶ implemented in Python
- ▶ `conda install -c jore matrix-decomposition`
- ▶ `pip install matrix-decomposition`

Implementation

- ▶ implemented in Python
- ▶ `conda install -c jore matrix-decomposition`
- ▶ `pip install matrix-decomposition`
- ▶ `git clone https://github.com/jor-/matrix-decomposition.git`

Summary

The algorithm MATRIX

- ▶ allows to approximate Hermitian matrices $A \in \mathbb{C}^{n \times n}$ by positive (semi-)definite matrices $B \in \mathbb{C}^{n \times n}$.
- ▶ allows to bound B_{ii} by parameters for all $i \in \{1, \dots, n\}$.
- ▶ allows to control $\kappa_2(B)$ by parameters.
- ▶ tries to minimize $\|B - A\|_F$.
- ▶ ensures that $B = A$ if A satisfies the requirements on B .
- ▶ ensures that B is sparse if A is sparse.
- ▶ is numerical stable.
- ▶ has negligible time and space overhead compared to the unmodified LDL^H decomposition algorithm.
- ▶ provides a LDL^H decomposition of B as a by-product.