Preconditioning by incomplete factorizations and approximate inverses

Miroslav Tůma

Institute of Computer Science

Academy of Sciences of the Czech Republic

tuma@cs.cas.cz

based on joint work with **Michele Benzi, Rafael Bru, Jurjen Duintjer Tebbens, José Marín, José Mas, Miroslav Rozložník, Jennifer Scott** et al.

Preconditioning 2009,

August 24-26, 2009, Hong Kong

Motivation: I.

Solving large, sparse systems by preconditioned iterative methods

$$Ax = b$$

Motivation: I.

Solving large, sparse systems by preconditioned iterative methods

$$Ax = b$$

Algebraic preconditioning as a transformation

$$M^{-1}Ax = M^{-1}b$$

.

Motivation: I.

Solving large, sparse systems by preconditioned iterative methods

$$Ax = b$$

Algebraic preconditioning as a transformation

$$M^{-1}Ax = M^{-1}b$$

In particular: Incomplete decompositions

Motivation: I.

Solving large, sparse systems by preconditioned iterative methods

$$Ax = b$$

Algebraic preconditioning as a transformation

$$M^{-1}Ax = M^{-1}b$$

In particular: Incomplete decompositions

- As usual, should be cheap to compute, implying fast converging preconditioned iterative method

Motivation: I.

Solving large, sparse systems by preconditioned iterative methods

$$Ax = b$$

Algebraic preconditioning as a transformation

$$M^{-1}Ax = M^{-1}b$$

In particular: Incomplete decompositions

- As usual, should be cheap to compute, implying fast converging preconditioned iterative method
- First point usually satisfied

Motivation: I.

Solving large, sparse systems by preconditioned iterative methods

$$Ax = b$$

Algebraic preconditioning as a transformation

$$M^{-1}Ax = M^{-1}b$$

In particular: Incomplete decompositions

- As usual, should be cheap to compute, implying fast converging preconditioned iterative method
- First point usually satisfied
- Should be sparse enough

Motivation: I.

Solving large, sparse systems by preconditioned iterative methods

$$Ax = b$$

Algebraic preconditioning as a transformation

$$M^{-1}Ax = M^{-1}b$$

In particular: Incomplete decompositions

- As usual, should be cheap to compute, implying fast converging preconditioned iterative method
- First point usually satisfied
- Should be sparse enough
- Our target is robustness.

Motivation: II.

What we are interested in?

- How can be a general direct incomplete decomposition modified such that it serves as a “better” inverse in the form of preconditioner
- We seek for a solution which is purely decomposition-based

Motivation: II.

What we are interested in?

- How can be a general direct incomplete decomposition modified such that it serves as a "better" inverse in the form of preconditioner
- We seek for a solution which is purely decomposition-based

What we do not discuss here?

- modifications of the basic algorithm (basic diagonal modifications, more general diagonal compensations with respect to some matvecs etc.)
- a priori diagonal changes
- matrix pre/post processings
- embedding into a more general scheme (e.g., with more levels)

Motivation: II.

What we are interested in?

- How can be a general direct incomplete decomposition modified such that it serves as a “better” inverse in the form of preconditioner
- We seek for a solution which is purely decomposition-based

What we do not discuss here?

- modifications of the basic algorithm (basic diagonal modifications, more general diagonal compensations with respect to some matvecs etc.)
- a priori diagonal changes
- matrix pre/post processings
- embedding into a more general scheme (e.g., with more levels)

All the above mentioned techniques are very important. But, here we try to analyze, not to defend a synthetic approach.

Motivation: II.

What we are interested in?

- How can be a general direct incomplete decomposition modified such that it serves as a "better" inverse in the form of preconditioner
- We seek for a solution which is purely decomposition-based

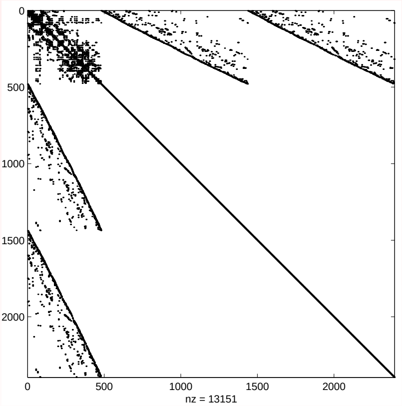
What we do not discuss here?

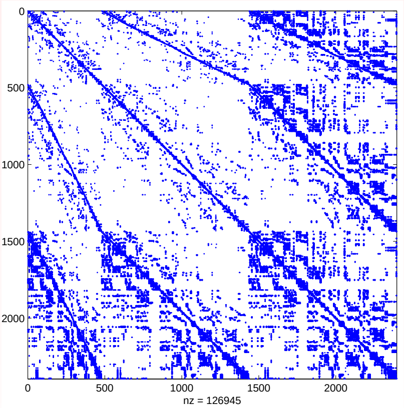
- modifications of the basic algorithm (basic diagonal modifications, more general diagonal compensations with respect to some matvecs etc.)
- a priori diagonal changes
- matrix pre/post processings
- embedding into a more general scheme (e.g., with more levels)

All the above mentioned techniques are very important. But, here we try to analyze, not to defend a synthetic approach.

But, why we are interested in inverses?

Motivation: III.

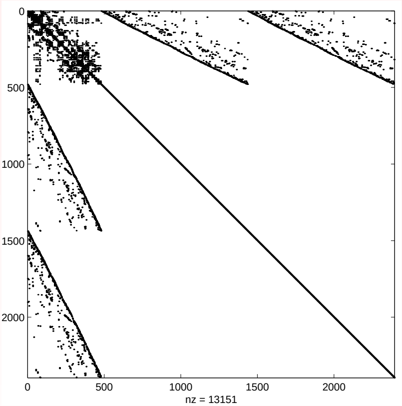




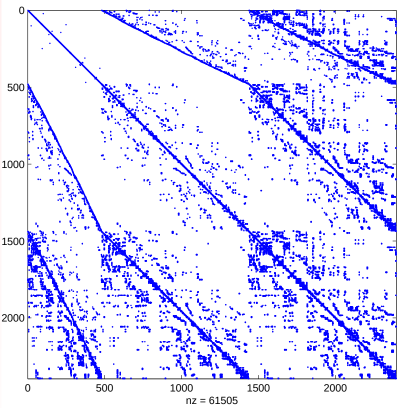
matrix ADD20

rather precise inverse
(2 its BiCGStab)

Motivation: III.

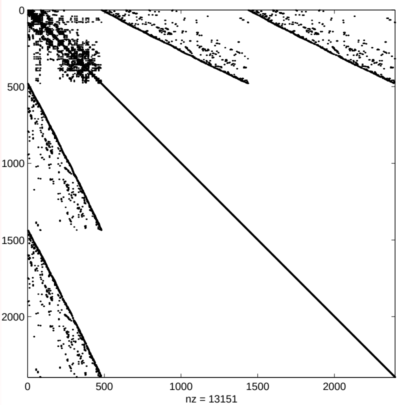


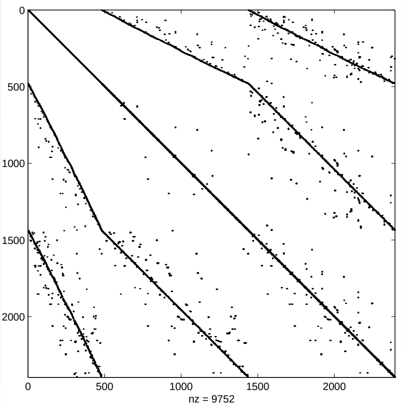
matrix ADD20



less precise inverse

Motivation: III.

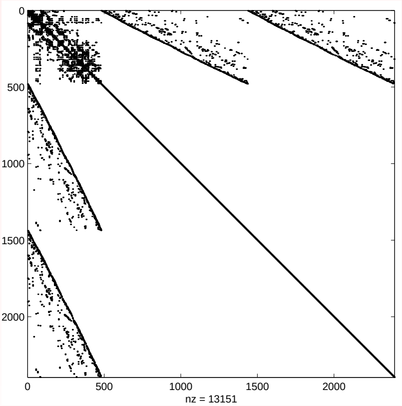


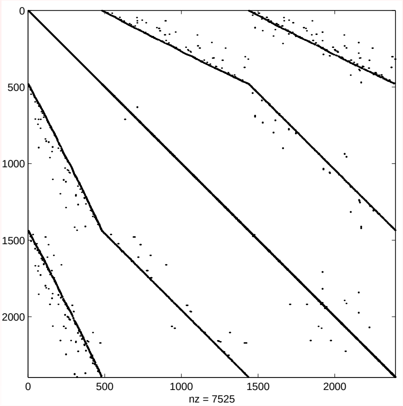


matrix ADD20

even less precise inverse

Motivation: III.

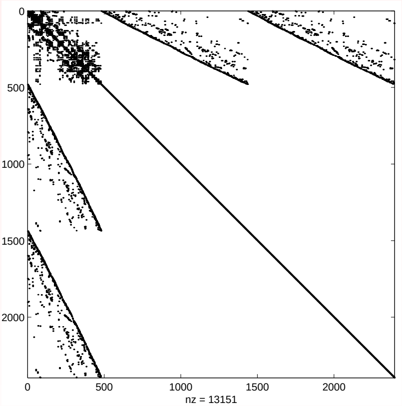




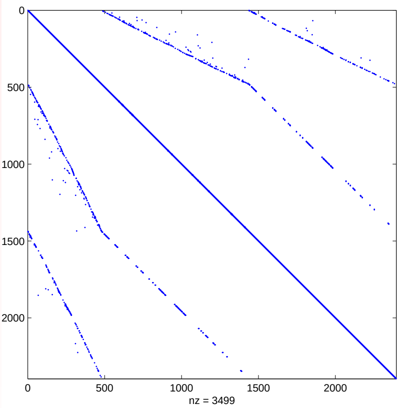
matrix ADD20

rough inverse

Motivation: III.

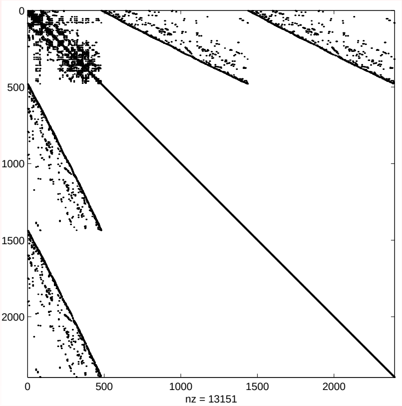


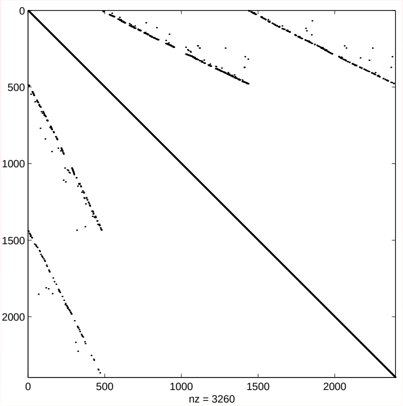
matrix ADD20



very rough inverse

Motivation: III.

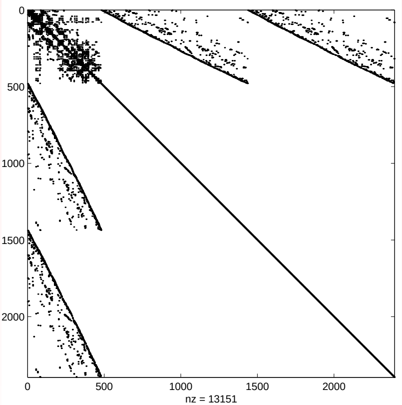


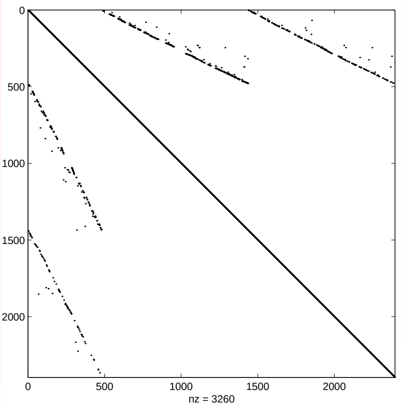


matrix ADD20

ILU decomposition

Motivation: III.





matrix ADD20

inverted ILU decomposition

Motivation: IV.

Concluded motivation

- Consulting / employing matrix inverse can provide useful information
- Two extreme cases of incomplete decompositions
  - approximate inverse decompositions
  - direct incomplete decompositions

- Approximate inverse decompositions (Kolotilina, Yeremin, 1993; Benzi, Meyer, T., 1996; Benzi, T., 1998 etc.)
- Successful use of parts of factorized matrix inverse used in inverse-based incomplete decompositions (Bollhöfer, Saad, 2002; Bollhöfer, 2003)

Motivation: IV.

Concluded motivation

- Consulting / employing matrix inverse can provide useful information
- Two extreme cases of incomplete decompositions
  - approximate inverse decompositions
  - direct incomplete decompositions

- Approximate inverse decompositions (Kolotilina, Yeremin, 1993; Benzi, Meyer, T., 1996; Benzi, T., 1998 etc.)
- Successful use of parts of factorized matrix inverse used in inverse-based incomplete decompositions (Bollhöfer, Saad, 2002; Bollhöfer, 2003)
- Our final goal to be presented here: combined use of direct and inverse incomplete decompositions

Motivation: IV.

Concluded motivation

- Consulting / employing matrix inverse can provide useful information
- Two extreme cases of incomplete decompositions
  - approximate inverse decompositions
  - direct incomplete decompositions

- Approximate inverse decompositions (Kolotilina, Yeremin, 1993; Benzi, Meyer, T., 1996; Benzi, T., 1998 etc.)
- Successful use of parts of factorized matrix inverse used in inverse-based incomplete decompositions (Bollhöfer, Saad, 2002; Bollhöfer, 2003)
- Our final goal to be presented here: combined use of direct and inverse incomplete decompositions
- One of the tools: generalized biconjugation formula

Outline

1. Limits of standard algebraic approaches
2. Standard biconjugation and matrix inverses
3. Fast implementations of more sophisticated incomplete decompositions
4. Direct-inverse decompositions
5. Conclusions

Outline

Standard sparsity pattern limits: I.

- The error $E = A - \bar{L}\bar{L}^T$ inside the pattern is zero

Standard sparsity pattern limits: I.

- The error $E = A - \bar{L}\bar{L}^T$ inside the pattern is zero
- Error outside the pre-specified pattern can be large.

Standard sparsity pattern limits: I.

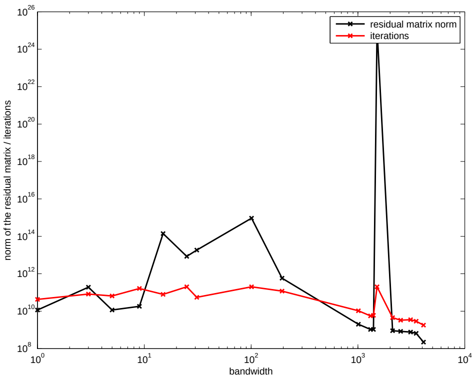
- The error $E = A - \bar{L}\bar{L}^T$ inside the pattern is zero
- Error outside the pre-specified pattern can be large.
- Example: banded pattern: BCSSTK38, $n = 8032$, $nz = 181,746$; airplane engine component.

bandwidth (full)	iterations
1	426
3	821
5	648
9	1638
15	792
1011	105
1311	56
1511	†
3111	35
4111	18

Standard sparsity pattern limits: I.

- The error $E = A - \bar{L}\bar{L}^T$ inside the pattern is zero
- Error outside the pre-specified pattern can be large.
- Example: banded pattern: BCSSTK38, $n = 8032$, $nz = 181,746$; airplane engine component.

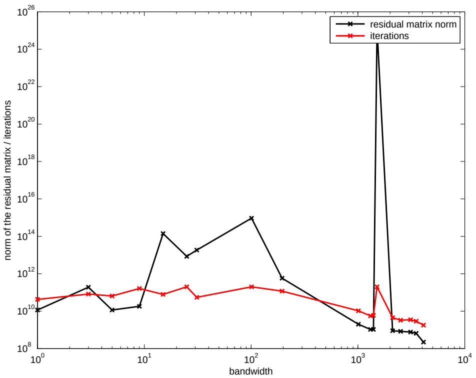
bandwidth (full)	iterations
1	426
3	821
5	648
9	1638
15	792
1011	105
1311	56
1511	†
3111	35
4111	18



Standard sparsity pattern limits: I.

- The error $E = A - \bar{L}\bar{L}^T$ inside the pattern is zero
- Error outside the pre-specified pattern can be large.
- Example: banded pattern: BCSSTK38, $n = 8032$, $nz = 181,746$; airplane engine component.

bandwidth (full)	iterations
1	426
3	821
5	648
9	1638
15	792
1011	105
1311	56
1511	†
3111	35
4111	18



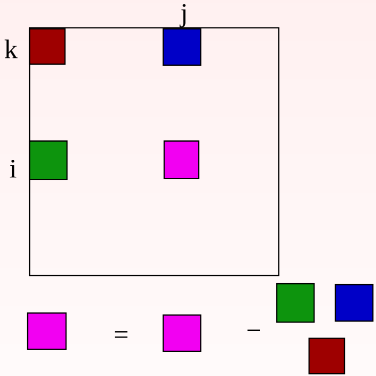
Decay may help

Standard sparsity pattern limits: II.

- More sophisticated approach. Where does the fill appear?

Standard sparsity pattern limits: II.

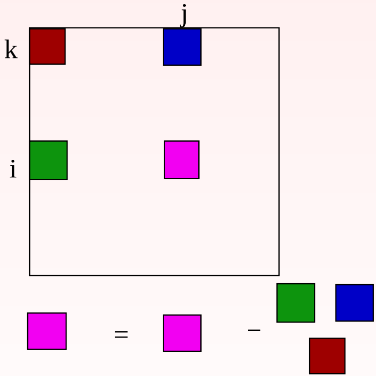
- More sophisticated approach. Where does the fill appear?



$$k < \{i, j\}$$

Standard sparsity pattern limits: II.

- More sophisticated approach. Where does the fill appear?

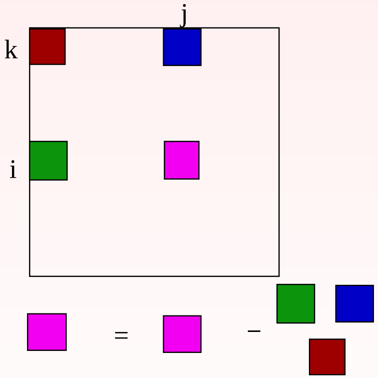


$$k < \{i, j\}$$

- Fill-path is a path in the **matrix adjacency graph** G joining nodes i and i via nodes with labels lower than both i and j.

Standard sparsity pattern limits: II.

- More sophisticated approach. Where does the fill appear?



$$k < \{i, j\}$$

- Fill-path is a path in the **matrix adjacency graph** G joining nodes i and i via nodes with labels lower than both i and j.
- Entries of the Cholesky factor $l_{ij}, i > j$ are nonzero if and only if there is a fill path joining i and j in G.

Standard sparsity pattern limits: II.

- Proposal: Allowing fill up to a maximum length ℓ of any fill path (Watts III, (1981)).

Standard sparsity pattern limits: II.

- Proposal: Allowing fill up to a maximum length ℓ of any fill path (Watts III, (1981)).
- Practically: A fill entry is permitted provided $level(i,j) \le \ell$.

$$level(i,j) = \min_{1 \le l \le \min\{i,j\}} \{level(i,l) + level(l,j) + 1\} \quad \text{(e.g.)}$$

Standard sparsity pattern limits: II.

- Proposal: Allowing fill up to a maximum length ℓ of any fill path (Watts III, (1981)).
- Practically: A fill entry is permitted provided $level(i,j) \le \ell$.

$$level(i,j) = \min_{1 \le l \le \min\{i,j\}} \{level(i,l) + level(l,j) + 1\} \quad \text{(e.g.)}$$

- Example: Matrix ENGINE, $n = 143571$, $nz = 2424822$.

Standard sparsity pattern limits: II.

- Proposal: Allowing fill up to a maximum length ℓ of any fill path (Watts III, (1981)).
- Practically: A fill entry is permitted provided $level(i,j) \le \ell$.

$$level(i,j) = \min_{1 \le l \le \min\{i,j\}} \{level(i,l) + level(l,j) + 1\} \quad \text{(e.g.)}$$

- Example: Matrix ENGINE, $n = 143571$, $nz = 2424822$.

levels	size prec	iterations
0	2424822	523
1	4458588	300
2	7595466	199
3	12128289	115
4	18078603	87
5	25474380	54
6	34153746	45
7	43861328	46
8	54276063	36

Standard sparsity pattern limits: II.

- Proposal: Allowing fill up to a maximum length ℓ of any fill path (Watts III, (1981)).
- Practically: A fill entry is permitted provided $level(i,j) \leq \ell$.

$$level(i,j) = \min_{1 \leq l \leq \min\{i,j\}} \{level(i,l) + level(l,j) + 1\} \quad \text{(e.g.)}$$

- Example: Matrix ENGINE, $n = 143571$, $nz = 2424822$.

levels	size prec	iterations
0	2424822	523
1	4458588	300
2	7595466	199
3	12128289	115
4	18078603	87
5	25474380	54
6	34153746	45
7	43861328	46
8	54276063	36

Often the fill in L grows too fast with ℓ.

Other classical possibilities and their limits: just a few notes

- Dropping entries with "smaller magnitudes" (absolutely/relatively) (Zlatev et al. (1978), Munksgaard (1980), Axelsson (1972, 1983 et al. etc.)

Other classical possibilities and their limits: just a few notes

- Dropping entries with "smaller magnitudes" (absolutely/relatively) (Zlatev et al. (1978), Munksgaard (1980), Axelsson (1972, 1983 et al. etc.)
- But: if only magnitudes of entries are used - structural information may be lost

Other classical possibilities and their limits: just a few notes

- Dropping entries with "smaller magnitudes" (absolutely/relatively) (Zlatev et al. (1978), Munksgaard (1980), Axelsson (1972, 1983 et al. etc.)
- But: if only magnitudes of entries are used - structural information may be lost
- More complicated schemes may strongly influence implementations (e.g., if both row and column access for intermediate quantities is needed)

Other classical possibilities and their limits: just a few notes

- Dropping entries with "smaller magnitudes" (absolutely/relatively) (Zlatev et al. (1978), Munksgaard (1980), Axelsson (1972, 1983 et al. etc.)
- But: if only magnitudes of entries are used - structural information may be lost
- More complicated schemes may strongly influence implementations (e.g., if both row and column access for intermediate quantities is needed)
- Plassman, Jones (1995): no structure, just the memory predictability, see also Freund, Nachtigal, (1990). Similarly Lin, Moré, (1990) with extended memory. ILUT by Saad, (1994).

Other classical possibilities and their limits: just a few notes

- Dropping entries with “smaller magnitudes” (absolutely/relatively) (Zlatev et al. (1978), Munksgaard (1980), Axelsson (1972, 1983 et al. etc.)
- But: if only magnitudes of entries are used - structural information may be lost
- More complicated schemes may strongly influence implementations (e.g., if both row and column access for intermediate quantities is needed)
- Plassman, Jones (1995): no structure, just the memory predictability, see also Freund, Nachtigal, (1990). Similarly Lin, Moré, (1990) with extended memory. ILUT by Saad, (1994).
- The importance of error matrix $E = A - LL^T$ understood (Duff, Meurant, (1989)) and exploited (D'Azevedo, Forsyth, Tang, 1992)

Other classical possibilities and their limits: just a few notes

- Dropping entries with "smaller magnitudes" (absolutely/relatively) (Zlatev et al. (1978), Munksgaard (1980), Axelsson (1972, 1983 et al. etc.)
- But: if only magnitudes of entries are used - structural information may be lost
- More complicated schemes may strongly influence implementations (e.g., if both row and column access for intermediate quantities is needed)
- Plassman, Jones (1995): no structure, just the memory predictability, see also Freund, Nachtigal, (1990). Similarly Lin, Moré, (1990) with extended memory. ILUT by Saad, (1994).
- The importance of error matrix $E = A - LL^T$ understood (Duff, Meurant, (1989)) and exploited (D'Azevedo, Forsyth, Tang, 1992)
- More sophisticated level settings and pattern/values combination (Scott, T., 2009); see the talk of Jennifer Scott

Outline

Generalized Gram-Schmidt

- Orthogonalize columns of I using the inner product $\langle\ ,\ \rangle_A$

Generalized Gram-Schmidt

- Orthogonalize columns of I using the inner product $\langle \, , \, \rangle_A$
- We get (instead of $A = QR$):

$$I = ZU$$

Generalized Gram-Schmidt

- Orthogonalize columns of I using the inner product $\langle \ , \ \rangle_A$
- We get (instead of $A = QR$):

$$I = ZU$$

  - U is upper triangular, as usual.

Generalized Gram-Schmidt

- Orthogonalize columns of I using the inner product $\langle \ , \ \rangle_A$
- We get (instead of $A = QR$):

$$I = ZU$$

  - U is upper triangular, as usual.
  - Z is orthogonal in $\langle \ , \ \rangle_A$:

$$Z^T A Z = I \quad (\text{Biconjugate decomposition})$$

Generalized Gram-Schmidt

- Orthogonalize columns of I using the inner product $\langle \; , \; \rangle_A$
- We get (instead of $A = QR$):

$$I = ZU$$

  - U is upper triangular, as usual.
  - Z is orthogonal in $\langle \; , \; \rangle_A$:

$$Z^T AZ = I \quad (\text{Biconjugate decomposition})$$

  - But: Z is upper triangular as well

Generalized Gram-Schmidt

- Orthogonalize columns of I using the inner product $\langle\ ,\ \rangle_A$
- We get (instead of $A = QR$):

$$I = ZU$$

  - U is upper triangular, as usual.
  - Z is orthogonal in $\langle\ ,\ \rangle_A$:

$$Z^T A Z = I \quad (\text{Biconjugate decomposition})$$

  - But: Z is upper triangular as well
- Easy to interprete the matrix inverse:

$$A^{-1} = ZZ^T (A^{-1} = ZDZ^T)$$

Generalized Gram-Schmidt: the decomposition

$$I = ZU$$

- Z is the inverse (Cholesky) factor of A

Generalized Gram-Schmidt: the decomposition

$$I = ZU$$

- Z is the inverse (Cholesky) factor of A
- U is the direct (Cholesky) factor of A

Generalized Gram-Schmidt: the decomposition

$$I = ZU$$

- Z is the inverse (Cholesky) factor of A
- U is the direct (Cholesky) factor of A
- In the incomplete case:

$$A \approx LL^T,\ U \approx L^T,\ Z \approx L^{-1}$$

Generalized Gram-Schmidt: the decomposition

$$I = ZU$$

- Z is the inverse (Cholesky) factor of A
- U is the direct (Cholesky) factor of A
- In the incomplete case:

$$A \approx LL^T,\ U \approx L^T,\ Z \approx L^{-1}$$

- Computational procedures to compute sparse incomplete factor Z: AINV (Benzi, Meyer, T., 1996; Benzi, T., 1998)
- Computational procedures to compute sparse incomplete U in this way: RIF (Benzi, T., 2003)

Generalized Gram-Schmidt: the decomposition

$$I = ZU$$

- Z is the inverse (Cholesky) factor of A
- U is the direct (Cholesky) factor of A
- In the incomplete case:

$$A \approx LL^T,\ U \approx L^T,\ Z \approx L^{-1}$$

- Computational procedures to compute sparse incomplete factor Z: AINV (Benzi, Meyer, T., 1996; Benzi, T., 1998)
- Computational procedures to compute sparse incomplete U in this way: RIF (Benzi, T., 2003)
- Three related notes: 1) Historical connections, 2) Note on the numerical properties, 3) Are we able to implement such algorithms?

Generalized Gram-Schmidt: historical connections

- Origins of the biconjugation for solving linear systems in more papers in 40's and early 50's (Escalator method by Morris (1946), Vector method by Purcell (1952) etc.)

Generalized Gram-Schmidt: historical connections

- Origins of the biconjugation for solving linear systems in more papers in 40's and early 50's (Escalator method by Morris (1946), Vector method by Purcell (1952) etc.)
- A detailed treatment in the first Wilkinson paper (with Fox and Huskey, 1948)

Generalized Gram-Schmidt: historical connections

- Origins of the biconjugation for solving linear systems in more papers in 40's and early 50's (Escalator method by Morris (1946), Vector method by Purcell (1952) etc.)
- A detailed treatment in the first Wilkinson paper (with Fox and Huskey, 1948)
- First systematic treatment of the technique: Hestenes, Stiefel, 1952 (conjugate direction methods).

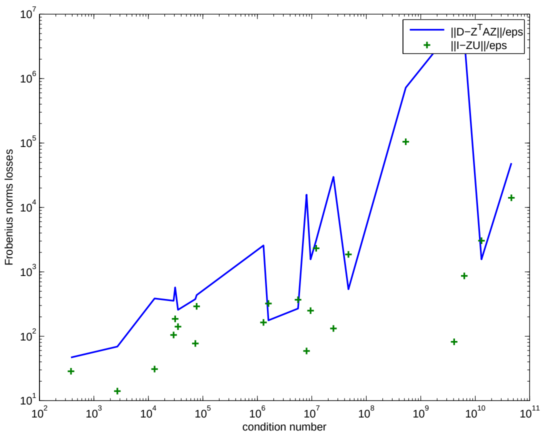
Generalized Gram-Schmidt: historical connections

- Origins of the biconjugation for solving linear systems in more papers in 40's and early 50's (Escalator method by Morris (1946), Vector method by Purcell (1952) etc.)
- A detailed treatment in the first Wilkinson paper (with Fox and Huskey, 1948)
- First systematic treatment of the technique: Hestenes, Stiefel, 1952 (conjugate direction methods).
- Slightly different schemes, papers differently motivated, different breakdown-free properties, different in floating-point arithmetic.

Generalized Gram-Schmidt: historical connections

- Origins of the biconjugation for solving linear systems in more papers in 40's and early 50's (Escalator method by Morris (1946), Vector method by Purcell (1952) etc.)
- A detailed treatment in the first Wilkinson paper (with Fox and Huskey, 1948)
- First systematic treatment of the technique: Hestenes, Stiefel, 1952 (conjugate direction methods).
- Slightly different schemes, papers differently motivated, different breakdown-free properties, different in floating-point arithmetic.
- Remind our goal: improving the algebraic preconditioners from inside the algorithm.
- Projection-based notes on our goal: see the talk T. at SIAM-LA'09 in Monterey.

Note on numerical properties of biconjugation



It can be proved:

- $||I - ZU||$ upper bound proportional to $\kappa^{1/2}(A)$
- $||I - Z^T AZ||$ upper bound proportional to $\kappa(A)$
- (Rozložník, T. et al., 2009)

Outline

General algorithmic scheme of direct and inverse GE-based decompositions

```
for i=1, n
     for j=1, i-1
     ...
     end j
end i
```

General algorithmic scheme of direct and inverse GE-based decompositions

```
for i=1, n
     for j=1, i-1 ←— never
     ...
     end j
end i
```

General algorithmic scheme of direct and inverse GE-based decompositions

```
for i=1, n
     for j=1, i-1 ⟵ never
     ...
     end j
end i
```

- Left-looking direct decompositions know the j indices via the elimination tree.

General algorithmic scheme of direct and inverse GE-based decompositions

```
for i=1, n
     for j=1, i-1 ←— never
     ...
     end j
end i
```

- Left-looking direct decompositions know the j indices via the elimination tree.
- Even simpler in the multifrontal algorithm.

General algorithmic scheme of direct and inverse GE-based decompositions

```
for i=1, n
     for j=1, i-1  ⟵ never
     ...
     end j
end i
```

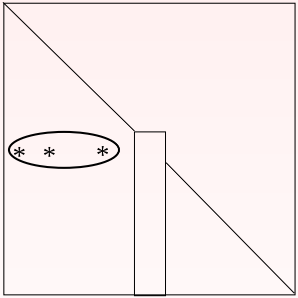
- Left-looking direct decompositions know the j indices via the elimination tree.
- Even simpler in the multifrontal algorithm.
- The real breakthrough useful also for incomplete decompositions came with the Yale sparse package (Eisenstat, Gursky, Schultz, Sherman, 1977, 1982)

General algorithmic scheme of direct and inverse GE-based decompositions

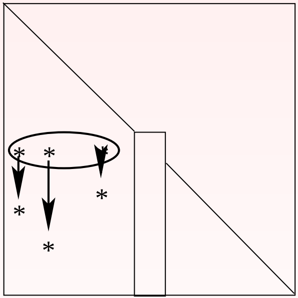
```
for i=1, n
     for j=1, i-1 ←— never
     ...
     end j
end i
```

- Left-looking direct decompositions know the j indices via the elimination tree.
- Even simpler in the multifrontal algorithm.
- The real breakthrough useful also for incomplete decompositions came with the Yale sparse package (Eisenstat, Gursky, Schultz, Sherman, 1977, 1982)
- The idea was rediscovered many times later.

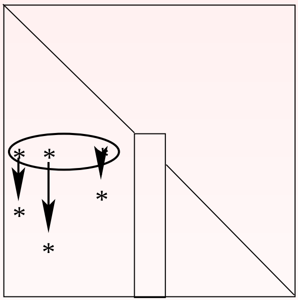
General algorithmic scheme of direct and inverse GE-based decompositions: II.



General algorithmic scheme of direct and inverse GE-based decompositions: II.

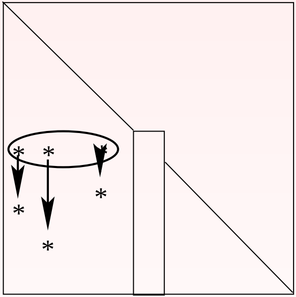


General algorithmic scheme of direct and inverse GE-based decompositions: II.



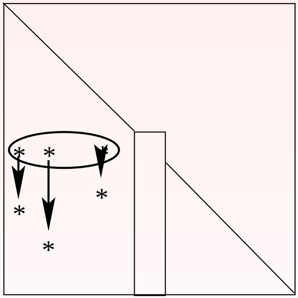
- Linked-list connects the j indices from the algorithm

General algorithmic scheme of direct and inverse GE-based decompositions: II.



- Linked-list connects the j indices from the algorithm
- This linked-list should be updated.

General algorithmic scheme of direct and inverse GE-based decompositions: II.



- Linked-list connects the j indices from the algorithm
- This linked-list should be updated.
- But: Generalized Gram-Schmidt contains the matrix-vector operation in the j loop.

Generalized Gram-Schmidt

$$I = ZU \equiv [z_1, \ldots, z_n]\ (u_{ij})_{i,j}$$

```
for i=1, n
     for j=1, i-1 with nonzero
```
$u_{ij} = \langle e_j^T, z_i^{(j)} \rangle_A$

$$z_i^{(j)} = z_i^{(j-1)} - z_j^{(j-1)} \frac{\langle e_j^T, z_i^{(j-1)} \rangle_A}{\langle e_j^T, z_j^{(j-1)} \rangle_A}$$

```
     end j
end i
```

scale Z by the $\sqrt{diag(d_i)} \equiv \sqrt{diag(\langle e_j^T, z_j^{(j-1)} \rangle_A)}$

Generalized Gram-Schmidt

$$I = ZU \equiv [z_1, \ldots, z_n]\ (u_{ij})_{i,j}$$

for i=1, n

for j=1, i-1 with nonzero $u_{ij} = \langle e_j^T, z_i^{(j)} \rangle_A$

$$z_i^{(j)} = z_i^{(j-1)} - z_j^{(j-1)} \frac{\langle e_j^T, z_i^{(j-1)} \rangle_A}{\langle e_j^T, z_j^{(j-1)} \rangle_A}$$

end j

end i

scale Z by the $\sqrt{diag(d_i)} \equiv \sqrt{diag(\langle e_j^T, z_j^{(j-1)} \rangle_A)}$

- All tests for one outer index i can be obtained in only one search through a few columns of A altogether.

Generalized Gram-Schmidt

$$I = ZU \equiv [z_1, \ldots, z_n]\ (u_{ij})_{i,j}$$

for i=1, n

for j=1, i-1 with nonzero $u_{ij} = \langle e_j^T, z_i^{(j)} \rangle_A$

$$z_i^{(j)} = z_i^{(j-1)} - z_j^{(j-1)} \frac{\langle e_j^T, z_i^{(j-1)} \rangle_A}{\langle e_j^T, z_j^{(j-1)} \rangle_A}$$

end j

end i

scale Z by the $\sqrt{diag(d_i)} \equiv \sqrt{diag(\langle e_j^T, z_j^{(j-1)} \rangle_A)}$

- All tests for one outer index i can be obtained in only one search through a few columns of A altogether.
- Sparse implementation is possible.

Generalized Gram-Schmidt

$$I = ZU \equiv [z_1, \ldots, z_n]\ (u_{ij})_{i,j}$$

```
for i=1, n
     for j=1, i-1 with nonzero
```
$u_{ij} = \langle e_j^T, z_i^{(j)} \rangle_A$

$$z_i^{(j)} = z_i^{(j-1)} - z_j^{(j-1)} \frac{\langle e_j^T, z_i^{(j-1)} \rangle_A}{\langle e_j^T, z_j^{(j-1)} \rangle_A}$$

```
     end j
end i
```
scale Z by the $\sqrt{diag(d_i)} \equiv \sqrt{diag(\langle e_j^T, z_j^{(j-1)} \rangle_A)}$

- All tests for one outer index i can be obtained in only one search through a few columns of A altogether.
- Sparse implementation is possible.
- The same is true for the breakdown-free variant SAINV.

Generalized Gram-Schmidt

$$I = ZU \equiv [z_1, \dots, z_n]\ (u_{ij})_{i,j}$$

for i=1, n

for j=1, i-1 with nonzero $u_{ij} = \langle e_j^T, z_i^{(j)} \rangle_A$

$$z_i^{(j)} = z_i^{(j-1)} - z_j^{(j-1)} \frac{\langle e_j^T, z_i^{(j-1)} \rangle_A}{\langle e_j^T, z_j^{(j-1)} \rangle_A}$$

end j

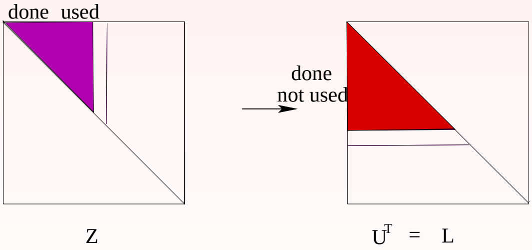
end i

scale Z by the $\sqrt{diag(d_i)} \equiv \sqrt{diag(\langle e_j^T, z_j^{(j-1)} \rangle_A)}$

- All tests for one outer index i can be obtained in only one search through a few columns of A altogether.
- Sparse implementation is possible.
- The same is true for the breakdown-free variant SAINV.
- But: in order to get U we must get Z: direct factor is obtained via the inverse factor

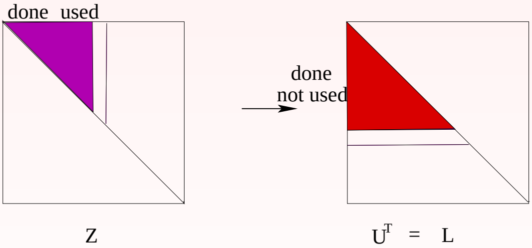
General algorithmic scheme of direct and inverse GE-based decompositions: III.

Let us repeat: Standard biconjugation $Z^T AZ$ via generalized Gram-Schmidt applied to get $I = ZU$:



General algorithmic scheme of direct and inverse GE-based decompositions: III.

Let us repeat: Standard biconjugation $Z^T AZ$ via generalized Gram-Schmidt applied to get $I = ZU$:



One way tranfer of information

Outline

New shifted biconjugation

- Note: we will use here general nonsymmetric formulation

$$A^{-1} = ZZ^T \longleftarrow A^{-1} = ZD^{-1}W^T$$

New shifted biconjugation

- Note: we will use here general nonsymmetric formulation

$$A^{-1} = ZZ^T \longleftarrow A^{-1} = ZD^{-1}W^T$$

Nonsymmetric recursions:

$$z_i^{(j)} = z_i^{(j-1)} - z_j^{(j-1)} \frac{a^j z_i^{(j-1)}}{a^j z_j^{(j-1)}}, \quad w_i^{(j)} = w_i^{(j-1)} - w_j^{(j-1)} \frac{a_j^T w_i^{(j-1)}}{a_j^T w_j^{(j-1)}}$$

New shifted biconjugation

- Note: we will use here general nonsymmetric formulation

$$A^{-1} = ZZ^T \longleftarrow A^{-1} = ZD^{-1}W^T$$

Nonsymmetric recursions:

$$z_i^{(j)} = z_i^{(j-1)} - z_j^{(j-1)} \frac{a^j z_i^{(j-1)}}{a^j z_j^{(j-1)}}, \quad w_i^{(j)} = w_i^{(j-1)} - w_j^{(j-1)} \frac{a_j^T w_i^{(j-1)}}{a_j^T w_j^{(j-1)}}$$

$$\Downarrow$$

$$sI - A^{-1} = ZD^{-1}V^T$$

New shifted biconjugation

- Note: we will use here general nonsymmetric formulation

$$A^{-1} = ZZ^T \longleftarrow A^{-1} = ZD^{-1}W^T$$

Nonsymmetric recursions:

$$z_i^{(j)} = z_i^{(j-1)} - z_j^{(j-1)} \frac{a^j z_i^{(j-1)}}{a^j z_j^{(j-1)}}, \quad w_i^{(j)} = w_i^{(j-1)} - w_j^{(j-1)} \frac{a_j^T w_i^{(j-1)}}{a_j^T w_j^{(j-1)}}$$

$$\Downarrow$$

$$sI - A^{-1} = ZD^{-1}V^T$$

Analogical recursions:

$$z_i = se_i - \sum_{j=1}^{i-1} \frac{v_j^T e_i}{d_j} z_j \quad , \; v_i = (a^i - se^i)^T - \sum_{j=1}^{i-1} \frac{z_j^T (a^i - se^i)}{d_j} v_j,$$

Z and D are the same in both recursions

More on the new biconjugation

- The $(s^{-1}I - A^{-1})^{-1})$ biconjugation introduced by Bru, Cerdán, Marín, Mas, 2003. The incomplete algorithm was proposed as an approximate inverse preconditioner.

More on the new biconjugation

- The $(s^{-1}I - A^{-1})^{-1})$ biconjugation introduced by Bru, Cerdán, Marín, Mas, 2003. The incomplete algorithm was proposed as an approximate inverse preconditioner.
- It was shown that this new biconjugation can be used to get a direct decomposition as well, Bru, Marín, Mas, T., 2008.

$$s^{-1}I - A^{-1} = ZD^{-1}V^T \text{ and } A = LDU \text{ and } Z = U^{-1}$$

$$s^{-1}I - U^{-1}D^{-1}L^{-1} = U^{-1}D^{-1}V^T$$

$$s^{-1}I = U^{-1}D^{-1}(L^{-1} + V^T)$$

More on the new biconjugation

- The $(s^{-1}I - A^{-1})^{-1})$ biconjugation introduced by Bru, Cerdán, Marín, Mas, 2003. The incomplete algorithm was proposed as an approximate inverse preconditioner.
- It was shown that this new biconjugation can be used to get a direct decomposition as well, Bru, Marín, Mas, T., 2008.

$$s^{-1}I - A^{-1} = ZD^{-1}V^T \text{ and } A = LDU \text{ and } Z = U^{-1}$$

$$s^{-1}I - U^{-1}D^{-1}L^{-1} = U^{-1}D^{-1}V^T$$

$$s^{-1}I = U^{-1}D^{-1}(L^{-1} + V^T)$$

upper triangular $\nearrow$

More on the new biconjugation

- The $(s^{-1}I - A^{-1})^{-1})$ biconjugation introduced by Bru, Cerdán, Marín, Mas, 2003. The incomplete algorithm was proposed as an approximate inverse preconditioner.
- It was shown that this new biconjugation can be used to get a direct decomposition as well, Bru, Marín, Mas, T., 2008.

$$s^{-1}I - A^{-1} = ZD^{-1}V^T \text{ and } A = LDU \text{ and } Z = U^{-1}$$

$$s^{-1}I - U^{-1}D^{-1}L^{-1} = U^{-1}D^{-1}V^T$$

$$s^{-1}I = U^{-1}D^{-1}(L^{-1} + V^T)$$

upper triangular ↗ ↖ lower triangular

More on the new biconjugation: II.

Pictorially

$$V = \begin{bmatrix} \ddots & & -sL^{-T} \\ & & \\ & \ddots & \\ & & \\ U^T D & & \ddots \end{bmatrix}, \qquad \mathrm{diag}(V) = D - sI. \tag{1}$$

More on the new biconjugation: II.

Pictorially

$$V = \begin{bmatrix} \ddots & & -sL^{-T} \\ & & \\ & \ddots & \\ & & \\ U^T D & & \ddots \end{bmatrix}, \qquad \mathrm{diag}(V) = D - sI. \qquad (1)$$

- V obtained by a simple recursion for its columns

More on the new biconjugation: II.

Pictorially

$$V = \begin{bmatrix} \ddots & & -sL^{-T} \\ & & \\ & \ddots & \\ & & \\ U^T D & & \ddots \end{bmatrix}, \qquad \mathrm{diag}(V) = D - sI. \quad (1)$$

- V obtained by a simple recursion for its columns
- The new recursions provide scaled U and L^{-1} at the same time!

Different use of the new decomposition

- The way of getting directly column of U and row of L^{-1} can be used for contruction condition estimators. We can profit from using the ideas of Bischof and Vömel, Duff, at the same time.
- New fast block decompositions can be proposed.

New biconjugation in the SPD case

- Note that $s^{-1}I - A^{-1} = ZD^{-1}V^T, V = LD - sL^{-T}, Z = L^{-T}$

$$v_i = (a^i - se^i)^T - \sum_{j=1}^{i-1} \frac{z_j^T(a^i - se^i)}{d_j} v_j,$$

New biconjugation in the SPD case

- Note that $s^{-1}I - A^{-1} = ZD^{-1}V^T, V = LD - sL^{-T}, Z = L^{-T}$

$$v_i = (a^i - se^i)^T - \sum_{j=1}^{i-1} \frac{z_j^T(a^i - se^i)}{d_j} v_j,$$

- We do not need to compute Z at all!

New biconjugation in the SPD case

- Note that $s^{-1}I - A^{-1} = ZD^{-1}V^T, V = LD - sL^{-T}, Z = L^{-T}$

$$v_i = (a^i - se^i)^T - \sum_{j=1}^{i-1} \frac{z_j^T(a^i - se^i)}{d_j} v_j,$$

- We do not need to compute Z at all!
- This is correct strictly mathematically, but computationally?

New biconjugation in the SPD case

- Note that $s^{-1}I - A^{-1} = ZD^{-1}V^T, V = LD - sL^{-T}, Z = L^{-T}$

$$v_i = (a^i - se^i)^T - \sum_{j=1}^{i-1} \frac{z_j^T(a^i - se^i)}{d_j} v_j,$$

- We do not need to compute Z at all!
- This is correct strictly mathematically, but computationally?
- Still the inverse factor influences the direct factor.

$$L^{-1} \longrightarrow L$$

New biconjugation in the SPD case

- Note that $s^{-1}I - A^{-1} = ZD^{-1}V^T, V = LD - sL^{-T}, Z = L^{-T}$

$$v_i = (a^i - se^i)^T - \sum_{j=1}^{i-1} \frac{z_j^T(a^i - se^i)}{d_j} v_j,$$

- We do not need to compute Z at all!
- This is correct strictly mathematically, but computationally?
- Still the inverse factor influences the direct factor.

$$L^{-1} \longrightarrow L$$

- But, dropping can interconnect computation of both L and L^{-1}.

New biconjugation in the SPD case

- Note that $s^{-1}I - A^{-1} = ZD^{-1}V^T, V = LD - sL^{-T}, Z = L^{-T}$

$$v_i = (a^i - se^i)^T - \sum_{j=1}^{i-1} \frac{z_j^T(a^i - se^i)}{d_j} v_j,$$

- We do not need to compute Z at all!
- This is correct strictly mathematically, but computationally?
- Still the inverse factor influences the direct factor.

$$L^{-1} \longrightarrow L$$

- But, dropping can interconnect computation of both L and L^{-1}.
- We drop L using sizes of entries in L^{-1} and vice versa: balanced incomplete factorization, Bru, Mas, Marín, T. 2008.
- Is is the best thing we can do?

Balanced incomplete factorization (BIF) experiments

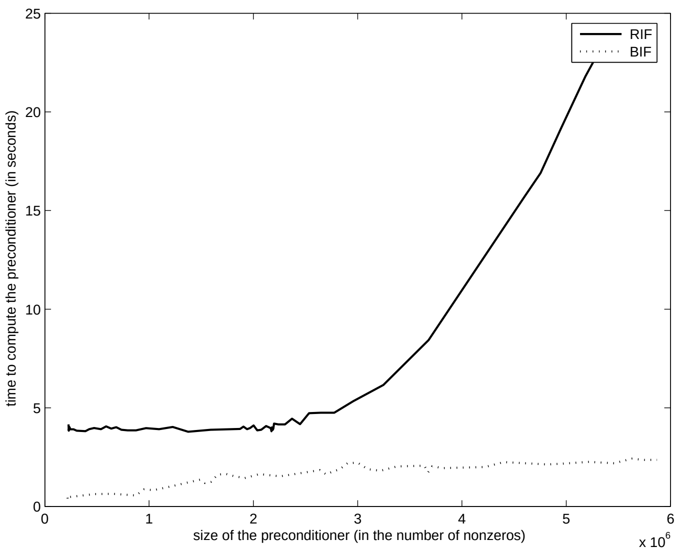
SPD experiments: I.

Example: matrix PWTK, n=217,918, nnz=5,926,171

Balanced incomplete factorization (BIF) experiments

SPD experiments: I.

Example: matrix PWTK, n=217,918, nnz=5,926,171



Balanced incomplete factorization (BIF) experiments: II.

Of course: not only pros; cons as well

- Taking approximate inverses into account, dropping must be always strong. Prefiltration of entries of A is a must.

Balanced incomplete factorization (BIF) experiments: II.

Of course: not only pros; cons as well

- Taking approximate inverses into account, dropping must be always strong. Prefiltration of entries of A is a must.
- We used the inverse-based dropping rules based on Saad, Bollhöfer, 2002, but dropping should be further investigated. It seems that sometimes any rules influence entries of the factors nonuniformly. Also, our dropping often forces skipping a lot of updates in the decomposition. Is this really the right way to go?

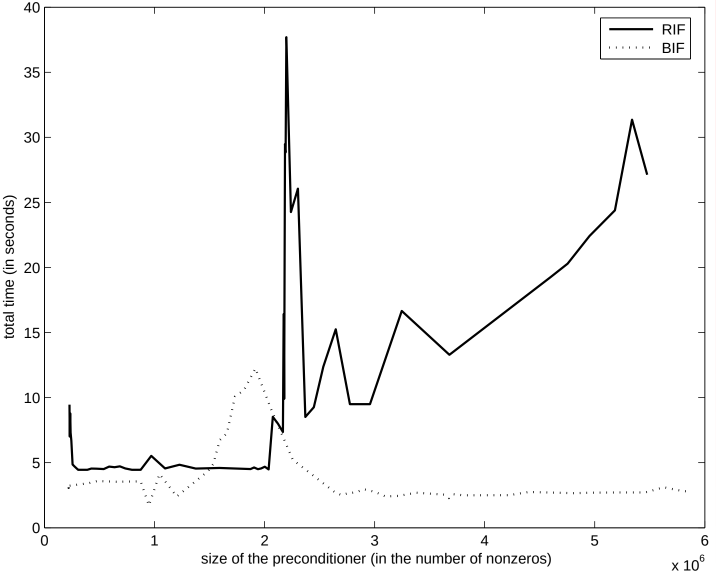
Balanced incomplete factorization (BIF) experiments: II.

Of course: not only pros; cons as well

- Taking approximate inverses into account, dropping must be always strong. Prefiltration of entries of A is a must.
- We used the inverse-based dropping rules based on Saad, Bollhöfer, 2002, but dropping should be further investigated. It seems that sometimes any rules influence entries of the factors nonuniformly. Also, our dropping often forces skipping a lot of updates in the decomposition. Is this really the right way to go?
- The convergence curve is often rather flat if we run many iterations. Is the accuracy sufficient for solving sequences from nonlinear solvers?

Balanced incomplete factorization (BIF) experiments: III.

SPD experiments: II.



Direct-inverse decomposition

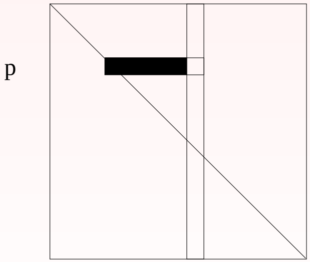
- Vector formulation of the shifted biconjugation can hide important details Bru, Mas, Marín, T. 2009

$$v_i = (a^i - se^i)^T - \sum_{j=1}^{i-1} \frac{z_j^T (a^i - se^i)}{d_j} v_j,$$

Direct-inverse decomposition

- Vector formulation of the shifted biconjugation can hide important details Bru, Mas, Marín, T. 2009

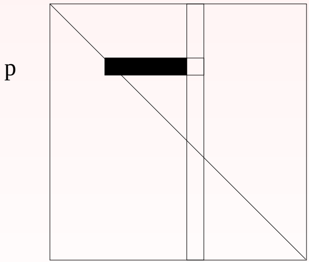
$$v_i = (a^i - se^i)^T - \sum_{j=1}^{i-1} \frac{z_j^T(a^i - se^i)}{d_j} v_j,$$



Direct-inverse decomposition

- Vector formulation of the shifted biconjugation can hide important details Bru, Mas, Marín, T. 2009

$$v_i = (a^i - se^i)^T - \sum_{j=1}^{i-1} \frac{z_j^T(a^i - se^i)}{d_j} v_j,$$

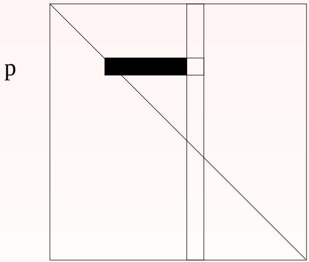


- v_{pi}: just the entries of V with indices $p+1, \ldots, i-1$ are involved

Direct-inverse decomposition

- Vector formulation of the shifted biconjugation can hide important details Bru, Mas, Marín, T. 2009

$$v_i = (a^i - se^i)^T - \sum_{j=1}^{i-1} \frac{z_j^T (a^i - se^i)}{d_j} v_j,$$



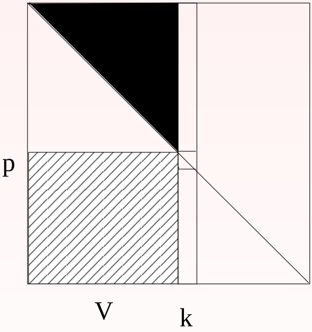
- v_{pi}: just the entries of V with indices $p+1, \ldots, i-1$ are involved
- good, but not enough: the inverse factor still updated only by entries of the inverse factor

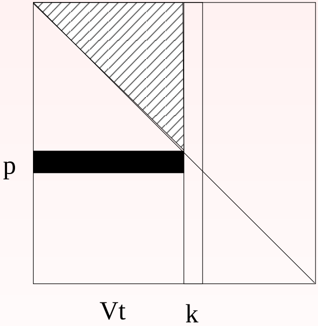
Direct-inverse decomposition: II.

- Even more sophisticated computation possible
- Here we demonstrate the computation in the fully nonsymmetric case

Direct-inverse decomposition: II.

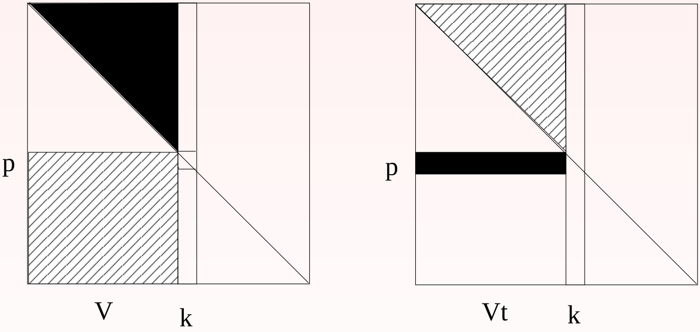
- Even more sophisticated computation possible
- Here we demonstrate the computation in the fully nonsymmetric case





Direct-inverse decomposition: II.

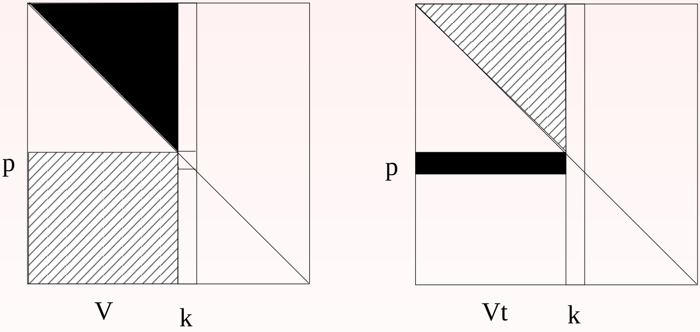
- Even more sophisticated computation possible
- Here we demonstrate the computation in the fully nonsymmetric case



- $v_{1:p-1}$ computed using fully filled areas

Direct-inverse decomposition: II.

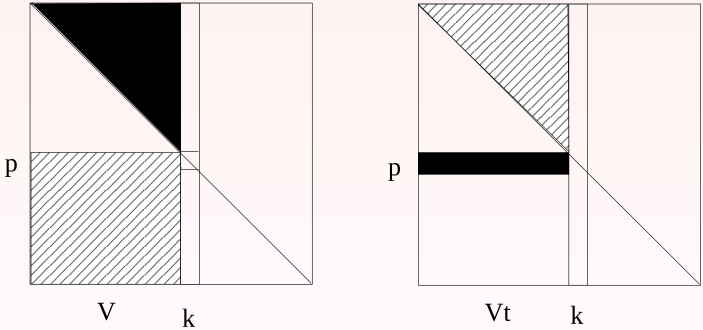
- Even more sophisticated computation possible
- Here we demonstrate the computation in the fully nonsymmetric case



- $v_{1:p-1}$ computed using fully filled areas
- $v_{p+1:n}$ computed using dashed areas

Direct-inverse decomposition: II.

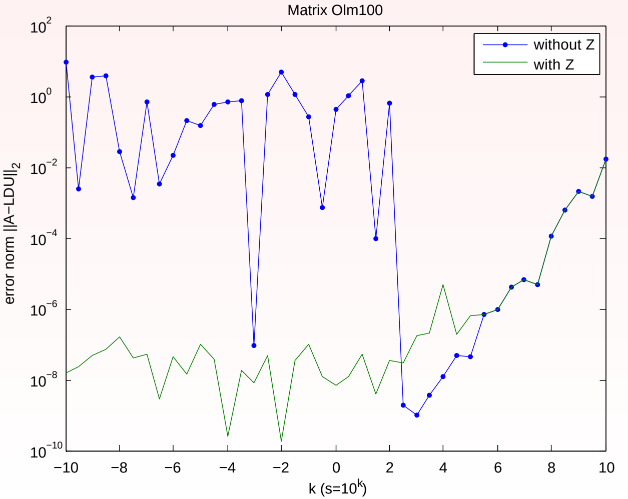
- Even more sophisticated computation possible
- Here we demonstrate the computation in the fully nonsymmetric case



- $v_{1:p-1}$ computed using fully filled areas
- $v_{p+1:n}$ computed using dashed areas
- direct and inverse factors influence each other

Scaling parameter

- Choice of scaling parameter s / computational procedures should be coordinated
- Here we demonstrate the computation in the fully nonsymmetric case



Direct-inverse (NBIF) decomposition: test problems

Matrix	n	nz	Application
CHEM_MASTER1	40,401	201,201	chemical engineering
EPB3	84,617	463,625	thermal problem
POISSON3DB	85,623	2,374,949	CFD
RAJAT20	86,916	604,299	circuit simulation
HCIRCUIT	105,676	513,072	circuit simulation
TRANS4	116,835	749,800	circuit simulation
CAGE12	130,228	2,032,536	directed weighted graph
FEM_3D_THERMAL2	147,900	3,489,300	thermal problem
XENON2	157,464	3,866,668	materials problem
CRASHBASIS	160,000	1,750,416	optimization problem
MAJORBASIS	160,000	1,750,416	optimization problem
STOMACH	213,360	3,021,648	2D/3D problem
TORSO3	256,156	4,429,042	2D/3D problem
ASIC_320KS	321,671	1,316,085	circuit simulation
LANGUAGE	399,130	1,216,334	directed weighted graph
CAGE13	445,315	7,479,343	directed weighted graph
RAJAT30	643,994	6,175,244	circuit simulation
ASIC_680KS	682,862	2,638,997	circuit simulation

Direct-inverse (NBIF) decomposition: experiments

Matrix	NBIF				ILU(τ)			
	$relsize$	t_p	its	t_it	$relsize$	t_p	its	t_it
CHEM_MASTER1	0.53	0.42	169	0.73	0.46	0.02	170	0.75
EPB3	0.93	1.09	76	1.22	1.03	0.03	73	1.14
POISSON3DB	0.11	1.11	126	3.45	0.12	0.11	136	3.92
RAJAT20	0.17	0.70	8	0.09	0.15	0.03	8	0.09
HCIRCUIT	0.39	0.56	182	2.45	0.31	0.03	191	2.45
TRANS4	0.32	0.41	65	1.06	0.22	0.06	66	1.03
CAGE12	0.31	0.94	5	0.13	0.36	0.09	5	0.17
FEM_3D_THERMAL2	0.06	1.45	20	0.63	0.08	0.14	23	0.73
XENON2	0.33	1.58	368	19.3	0.40	0.30	†	†
CRASHBASIS	0.18	0.66	29	0.73	0.18	0.08	25	0.61
MAJORBASIS	0.36	1.08	15	0.42	0.37	0.09	15	0.42
STOMACH	0.07	0.80	20	0.67	0.07	0.09	25	0.86
TORSO3	0.06	1.31	6	0.28	0.06	0.17	3	0.16
ASIC_320KS	0.26	0.55	20	0.88	0.24	0.09	20	0.84
LANGUAGE	0.53	1.72	9	0.53	0.54	0.11	15	0.98
CAGE13	0.06	2.48	5	0.45	0.06	0.30	7	0.64
RAJAT30	0.11	3.53	3	0.34	0.13	0.41	3	0.30
ASIC_680KS	0.26	2.36	5	0.48	0.26	0.13	6	0.55

Direct-inverse (NBIF) decomposition: experiments: II.

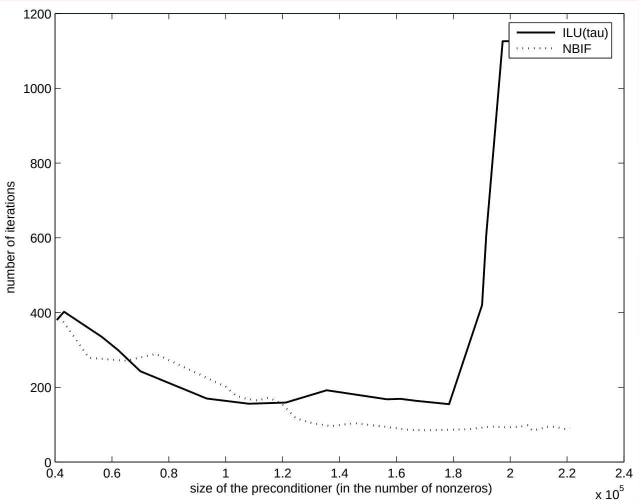


Figure: Sizes of NBIF and ILU(τ) preconditioners versus iteration counts of the preconditioned BiCGStab method for the matrix CHEM_MASTER1.

Outline

Conclusions

- Development of algebraic decompositions not finished.

Conclusions

- Development of algebraic decompositions not finished.
- In particular, a new direct-inverse decomposition BIF/NBIF useful in preconditioning was introduced.

Conclusions

- Development of algebraic decompositions not finished.
- In particular, a new direct-inverse decomposition BIF/NBIF useful in preconditioning was introduced.
- BIF/NBIF may be useful in other applications, e.g. in construction of condition estimators.

Conclusions

- Development of algebraic decompositions not finished.
- In particular, a new direct-inverse decomposition BIF/NBIF useful in preconditioning was introduced.
- BIF/NBIF may be useful in other applications, e.g. in construction of condition estimators.
- Efficiency of the new schemes is strongly related to their implementation.

Conclusions

- Development of algebraic decompositions not finished.
- In particular, a new direct-inverse decomposition BIF/NBIF useful in preconditioning was introduced.
- BIF/NBIF may be useful in other applications, e.g. in construction of condition estimators.
- Efficiency of the new schemes is strongly related to their implementation.
- Further computational aspects are still under investigation.

Thank you for your attention!

Last but not least

Thank you for your attention!

Last but not least

Thank you for your attention!

Last but not least

Thank you for your attention!

Last but not least

Thank you for your attention!