The importance of Structure in Algebraic Preconditioners (Level-based Algebraic Preconditioning)

Jennifer Scott

Rutherford Appleton Laboratory

Miroslav Tůma

Institute of Computer Science

Academy of Sciences of the Czech Republic

ALA 2010,

In honor of Hans Schneider

Novi Sad, May 24-28, 2010

Outline

1. Introduction: Preconditioned iterative methods
2. Goal of this talk
3. Algebraic preconditioners
4. The importance of having structure
5. Conclusions

Outline

Preconditioned iterative methods

Solving large, sparse SPD systems by iterative methods

$$Ax = b$$

Preconditioned iterative methods

Solving large, sparse SPD systems by iterative methods

$$Ax = b$$

Algebraic preconditioning as a transformation

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

.

Preconditioned iterative methods

Solving large, sparse SPD systems by iterative methods

$$Ax = b$$

Algebraic preconditioning as a transformation

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

In particular: Incomplete decompositions

Preconditioned iterative methods

Solving large, sparse SPD systems by iterative methods

$$Ax = b$$

Algebraic preconditioning as a transformation

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

In particular: Incomplete decompositions

- As usual, should be cheap, fast to compute, implying fast converging preconditioned iterative method
- sparse enough
- providing just sufficient approximation of the algebraic problem if this makes computations faster
- Our target is robustness

Outline

Goal of this talk

Search of more robust algebraic preconditioners

Goal of this talk

Search of more robust algebraic preconditioners

1. Show the importance of structure of the matrix and its decomposition in algebraic preconditioners.

Goal of this talk

Search of more robust algebraic preconditioners

1. Show the importance of structure of the matrix and its decomposition in algebraic preconditioners.
2. Present the effect separately from the other possible improvements (no compensations, no diagonal changes etc.).

Goal of this talk

Search of more robust algebraic preconditioners

1. Show the importance of structure of the matrix and its decomposition in algebraic preconditioners.
2. Present the effect separately from the other possible improvements (no compensations, no diagonal changes etc.).
3. Propose a new way to level-based strategies in incomplete decompositions.

Goal of this talk

Search of more robust algebraic preconditioners

1. Show the importance of structure of the matrix and its decomposition in algebraic preconditioners.
2. Present the effect separately from the other possible improvements (no compensations, no diagonal changes etc.).
3. Propose a new way to level-based strategies in incomplete decompositions.
4. The techniques are a basis of the HSL code MI22 which is being developed.

Outline

Brief notes on development of algebraic preconditioners: I.

- Stencil based advent (Buleev, 1959, 1960; Varga, 1960; etc.):
  stencils $\leftrightarrow$ local interpolation $\leftrightarrow$ elimination

Brief notes on development of algebraic preconditioners: I.

- Stencil based advent (Buleev, 1959, 1960; Varga, 1960; etc.): stencils $\leftrightarrow$ local interpolation $\leftrightarrow$ elimination
- Crucial moment: paper by Meijerink and van der Vorst (1977) recognizing the potential of incomplete decompositions for preconditioning.

Brief notes on development of algebraic preconditioners: I.

- Stencil based advent (Buleev, 1959, 1960; Varga, 1960; etc.): stencils $\leftrightarrow$ local interpolation $\leftrightarrow$ elimination
- Crucial moment: paper by Meijerink and van der Vorst (1977) recognizing the potential of incomplete decompositions for preconditioning.
- Dropping entries with “smaller magnitudes” (absolutely/relatively) (Zlatev et al. (1978), Munksgaard (1980), Axelsson (1972, 1983 et al. etc.)

Brief notes on development of algebraic preconditioners: I.

- Stencil based advent (Buleev, 1959, 1960; Varga, 1960; etc.): stencils $\leftrightarrow$ local interpolation $\leftrightarrow$ elimination
- Crucial moment: paper by Meijerink and van der Vorst (1977) recognizing the potential of incomplete decompositions for preconditioning.
- 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

Brief notes on development of algebraic preconditioners: II.

- Plassman, Jones (1995): no structure, just the memory predictability, see also Freund, Nachtigal, (1990). Similarly Lin, Moré with extended memory. ILUT by Saad, (1994).

Brief notes on development of algebraic preconditioners: II.

- Plassman, Jones (1995): no structure, just the memory predictability, see also Freund, Nachtigal, (1990). Similarly Lin, Moré with extended memory. ILUT by Saad, (1994).
- Allowing fill up to a maximum length ℓ of any fill path (Watts III, (1981)).

Brief notes on development of algebraic preconditioners: II.

- Plassman, Jones (1995): no structure, just the memory predictability, see also Freund, Nachtigal, (1990). Similarly Lin, Moré with extended memory. ILUT by Saad, (1994).
- Allowing fill up to a maximum length ℓ of any fill path (Watts III, (1981)).
- Practically: A fill entry is permitted provided $level(i,j) \le \ell$.

$$level(i,j) = \min_{1 \le l \le \min\{i,j\}} \{level(i,l) + level(l,j) + 1\}$$

(one of more definitions which slightly differ)

Brief notes on development of algebraic preconditioners: II.

- Plassman, Jones (1995): no structure, just the memory predictability, see also Freund, Nachtigal, (1990). Similarly Lin, Moré with extended memory. ILUT by Saad, (1994).
- Allowing fill up to a maximum length ℓ of any fill path (Watts III, (1981)).
- Practically: A fill entry is permitted provided $level(i,j) \le \ell$.

$$level(i,j) = \min_{1 \le l \le \min\{i,j\}} \{level(i,l) + level(l,j) + 1\}$$

(one of more definitions which slightly differ)

- Structure of levels helps but it has its strong drawbacks as well.

Level-based approach: further comments

- Often found that fill in L grows too quickly with ℓ.

Level-based approach: further comments

- Often found that fill in L grows too quickly with ℓ.
- While the error $R = A - LL^T$ inside the prespecified pattern is zero, outside can be large.

Level-based approach: further comments

- Often found that fill in L grows too quickly with ℓ.
- While the error $R = A - LL^T$ inside the prespecified pattern is zero, outside can be large.
- First simple combination of level-based approaches with dropping: D'Azevedo, Forsyth, Tang, (1992a).

Level-based approach: further comments

- Often found that fill in L grows too quickly with ℓ.
- While the error $R = A - LL^T$ inside the prespecified pattern is zero, outside can be large.
- First simple combination of level-based approaches with dropping: D'Azevedo, Forsyth, Tang, (1992a).
- The real breakthrough in level-based approaches: cheap predictions by Hysom, Pothen, (2002)

Level-based approach: further comments

- Often found that fill in L grows too quickly with ℓ.
- While the error $R = A - LL^T$ inside the prespecified pattern is zero, outside can be large.
- First simple combination of level-based approaches with dropping: D'Azevedo, Forsyth, Tang, (1992a).
- The real breakthrough in level-based approaches: cheap predictions by Hysom, Pothen, (2002)
- Our MI22 preconditioner is a new way to use level-based information, memory prediction and dropping at the same time.

Outline

First component of our approach: new setting of levels

Preassign levels to the entries individually

First component of our approach: new setting of levels

Preassign levels to the entries individually

- Computing the absolute values of the smallest and largest entries of A: $msmall$ and $mbig$.

First component of our approach: new setting of levels

Preassign levels to the entries individually

- Computing the absolute values of the smallest and largest entries of A: $msmall$ and $mbig$.
- Distribute nonzero entries uniformly by $\log|a_{ij}|$ into $ngroup0 = [log(mbig) - log(msmall)] + 1$ groups. Shrink zero groups to get $ngroup$ of them.

First component of our approach: new setting of levels

Preassign levels to the entries individually

- Computing the absolute values of the smallest and largest entries of A: $msmall$ and $mbig$.
- Distribute nonzero entries uniformly by $\log |a_{ij}|$ into $ngroup0 = [log(mbig) - log(msmall)] + 1$ groups. Shrink zero groups to get $ngroup$ of them.
- Set $level(i, j)$ for individual entries: For $\ell < ngroup$: $level(i, j) = (\ell - 1) * (l/ngroup) + 1$ where l $(1 \leq l \leq ngroup0)$ is the index of the group a_{ij} belongs to, and slightly differently otherwise.

First component of our approach: new setting of levels

Preassign levels to the entries individually

- Computing the absolute values of the smallest and largest entries of A: $msmall$ and $mbig$.
- Distribute nonzero entries uniformly by $\log |a_{ij}|$ into $ngroup0 = [log(mbig) - log(msmall)] + 1$ groups. Shrink zero groups to get $ngroup$ of them.
- Set $level(i,j)$ for individual entries: For $\ell < ngroup$: $level(i,j) = (\ell - 1) * (l/ngroup) + 1$ where l ($1 \leq l \leq ngroup0$) is the index of the group a_{ij} belongs to, and slightly differently otherwise.
- During the $IC(\ell)$ decomposition, entries of the factor L that correspond to nonzero entries of A are assigned the level $level(i,j)$.

First component of our approach: new setting of levels

Preassign levels to the entries individually

- Computing the absolute values of the smallest and largest entries of A: $msmall$ and $mbig$.
- Distribute nonzero entries uniformly by $\log |a_{ij}|$ into $ngroup0 = [log(mbig) - log(msmall)] + 1$ groups. Shrink zero groups to get $ngroup$ of them.
- Set $level(i,j)$ for individual entries: For $\ell < ngroup$: $level(i,j) = (\ell - 1) * (l/ngroup) + 1$ where l $(1 \leq l \leq ngroup0)$ is the index of the group a_{ij} belongs to, and slightly differently otherwise.
- During the $IC(\ell)$ decomposition, entries of the factor L that correspond to nonzero entries of A are assigned the level $level(i,j)$.
- Each potential fill entry l_{ij} is assigned a level

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

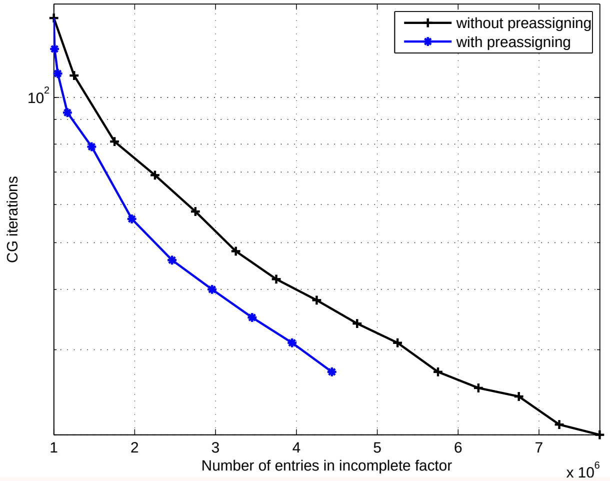
A fill entry is permitted provided $level(i,j) \leq k$.

Effect of individual level preassignments: MI22

First component of our approach: new setting of levels

Experiments: Kohn-Sham equation, n=250500

Effect of individual level preassignments: MI22



First component of our approach: new setting of levels

Experience from the experiments

Notes on the presetting of levels

First component of our approach: new setting of levels

Experience from the experiments

Notes on the presetting of levels

- (+) Settings do not increase timings significantly.

First component of our approach: new setting of levels

Experience from the experiments

Notes on the presetting of levels

- (+) Settings do not increase timings significantly.
- (-)The improvements are often small. We intend to construct a robust strategy which is used as a default value.

First component of our approach: new setting of levels

Experience from the experiments

Notes on the presetting of levels

- (+) Settings do not increase timings significantly.
- (-)The improvements are often small. We intend to construct a robust strategy which is used as a default value.
- Open problem: determine more sophisticated rules to preassign levels.

Second component of our approach: keeping structure

Integrate the predefined factor structure with dropping

Second component of our approach: keeping structure

Integrate the predefined factor structure with dropping

- Symbolic part of the modified (MI22) $IC(\ell)$ predefines the structure.

Second component of our approach: keeping structure

Integrate the predefined factor structure with dropping

- Symbolic part of the modified (MI22) $IC(\ell)$ predefines the structure.
- Only very small entries from the structure are not kept. The space is then freed and can be further used.
- The final size parametrized by memory multiplier $0 \leq \theta$.

Second component of our approach: keeping structure

Integrate the predefined factor structure with dropping

- Symbolic part of the modified (MI22) $IC(\ell)$ predefines the structure.
- Only very small entries from the structure are not kept. The space is then freed and can be further used.
- The final size parametrized by memory multiplier $0 \leq \theta$.
- Additional space distributed (1) uniformly or (2) nonuniformly.

Second component of our approach: keeping structure

Integrate the predefined factor structure with dropping

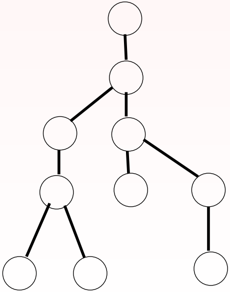
- Symbolic part of the modified (MI22) $IC(\ell)$ predefines the structure.
- Only very small entries from the structure are not kept. The space is then freed and can be further used.
- The final size parametrized by memory multiplier $0 \leq \theta$.
- Additional space distributed (1) uniformly or (2) nonuniformly.

- Nonuniform distribution based on the elimination tree and its row subtrees

Second component of our approach: keeping structure

Integrate the predefined factor structure with dropping

- Symbolic part of the modified (MI22) $IC(\ell)$ predefines the structure.
- Only very small entries from the structure are not kept. The space is then freed and can be further used.
- The final size parametrized by memory multiplier $0 \leq \theta$.
- Additional space distributed (1) uniformly or (2) nonuniformly.



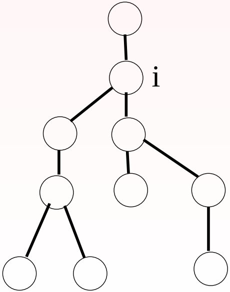
- Nonuniform distribution based on the elimination tree and its row subtrees

Second component of our approach: keeping structure

Integrate the predefined factor structure with dropping

- Symbolic part of the modified (MI22) $IC(\ell)$ predefines the structure.
- Only very small entries from the structure are not kept. The space is then freed and can be further used.
- The final size parametrized by memory multiplier $0 \leq \theta$.
- Additional space distributed (1) uniformly or (2) nonuniformly.

- Nonuniform distribution based on the elimination tree and its row subtrees

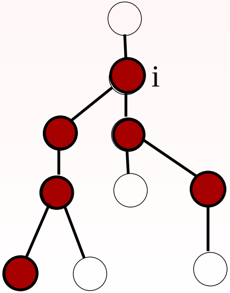


Second component of our approach: keeping structure

Integrate the predefined factor structure with dropping

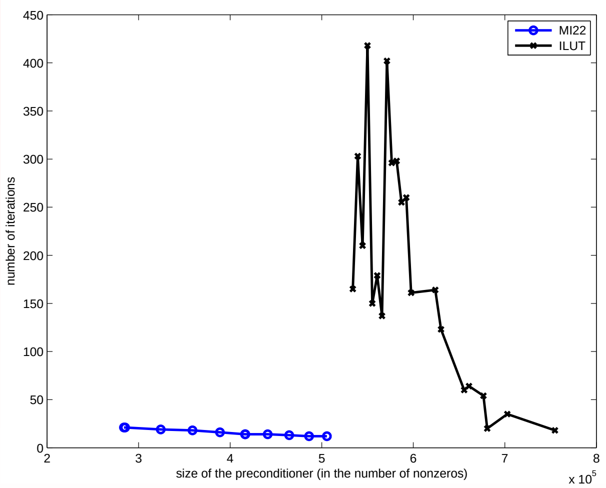
- Symbolic part of the modified (MI22) $IC(\ell)$ predefines the structure.
- Only very small entries from the structure are not kept. The space is then freed and can be further used.
- The final size parametrized by memory multiplier $0 \leq \theta$.
- Additional space distributed (1) uniformly or (2) nonuniformly.

- Nonuniform distribution based on the elimination tree and its row subtrees



Second component of our approach: keeping structure

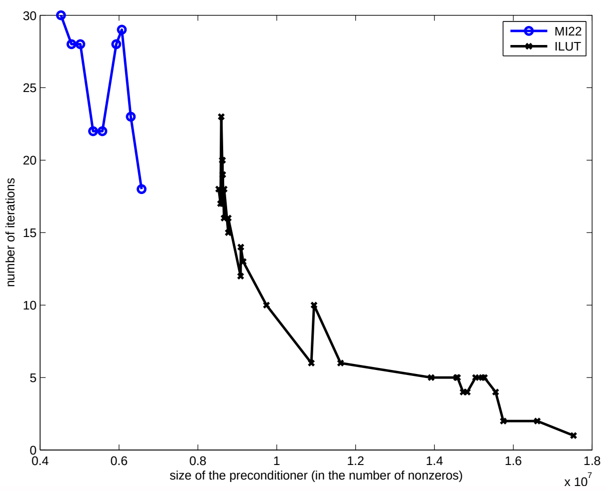
Level-based × value-based: example 1



S1RMT3M1, cylindrical shell problem, n=5489

Second component of our approach: keeping structure

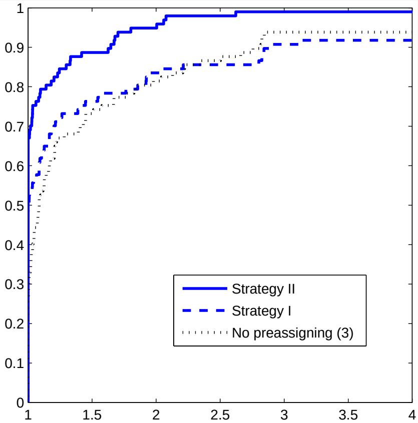
Level-based × value-based: example 2



NASASRB, structural mechanics, n=54870

Second component of our approach: keeping structure

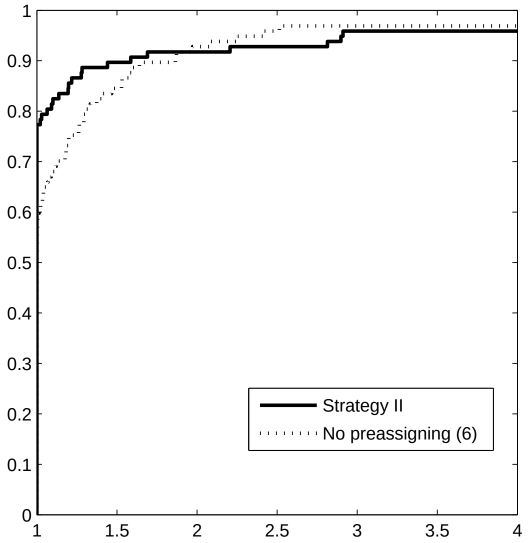
97 problems; efficiency profiles (Dolan, Moré, 2001) for 3 levels
efficiency=size $\times$ iterations; fractions $p(\alpha)$ for which a solver is within a factor of α of the best solver.



Strategy I.: stress on sparsity; Strategy II.: denser and faster option

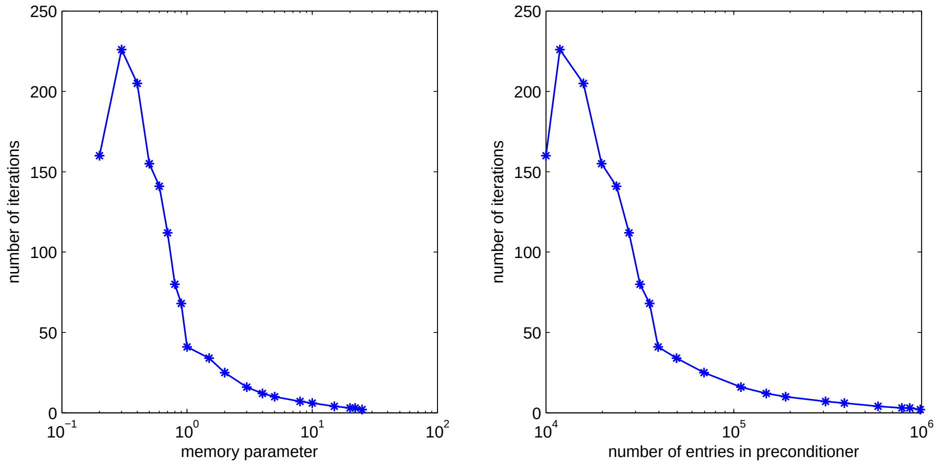
Second component of our approach: keeping structure

Efficiency profiles for 6 levels.



Second component of our approach: keeping structure

MI22: scaling the preconditioner for simple (2D Poisson) problem



MI22 with levels versus $IC(\tau)$(also via MI22)

TUBE1, cylindrical shell, n=21498

struc	drop=0.0		drop=10^{-7}	
level	size	its	size	its
5	1250952	†	1227570	†
6	1660827	429	1618808	423
7	1807337	405	1756733	408
8	2178312	272	2104496	281
9	2368289	260	2280081	267
10	3026431	184	2873613	185
11	3968731	426	3656826	335
12	4874629	†	4398086	†
13	5849563	†	5178688	†
14	6840871	664	5938543	647
15	7838623	262	6680235	215

IC(τ)	size	its
55	280626	†
50	1458024	†
45	2076970	†
40	2252687	†
1e-3	16139618	†
1e-4	9001342	†
5e-5	9649083	471
2e-5	9610841	87
1e-5	10050227	18
5e-6	10741254	6
1e-6	12451396	2
0	21802746	1

MI22 with levels versus $IC(\tau)$(also via MI22)

TUBE1, cylindrical shell, n=21498

struc	drop=0.0		drop=10^{-7}	
level	size	its	size	its
5	1250952	†	1227570	†
6	1660827	429	1618808	423
7	1807337	405	1756733	408
8	2178312	272	2104496	281
9	2368289	260	2280081	267
10	3026431	184	2873613	185
11	3968731	426	3656826	335
12	4874629	†	4398086	†
13	5849563	†	5178688	†
14	6840871	664	5938543	647
15	7838623	262	6680235	215

IC(τ)	size	its
55	280626	†
50	1458024	†
45	2076970	†
40	2252687	†
1e-3	16139618	†
1e-4	9001342	†
5e-5	9649083	471
2e-5	9610841	87
1e-5	10050227	18
5e-6	10741254	6
1e-6	12451396	2
0	21802746	1

But: Reorderings may minimize the effect.

Outline

Conclusions

- Preserving the structure may play significant role in incomplete decompositions.

Conclusions

- Preserving the structure may play significant role in incomplete decompositions.
- Codes may be reasonably fast and robust.

Conclusions

- Preserving the structure may play significant role in incomplete decompositions.
- Codes may be reasonably fast and robust.
- MI22 code of Harwell Subroutine Library offers a way to implement this strategy.

Conclusions

- Preserving the structure may play significant role in incomplete decompositions.
- Codes may be reasonably fast and robust.
- MI22 code of Harwell Subroutine Library offers a way to implement this strategy.

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!