

Using Numerical Values in Sparse Matrix Reorderings for ARMS Preconditioning

Masha Sosonkina

University of Minnesota

Joint work with Yousef Saad

Work supported by NSF/INT grant

OUTLINE

- ▷▷ Introduction: on using matrix values in reorderings for preconditioning.
- ▷▷ Matrix values for reorderings of (almost) structurally symmetric problems.
 - Cost assignment strategies for minimum-cost spanning tree algorithms.
 - ARMS preconditioning.
- ▷▷ Numerical experiments.
- ▷▷ Summary.

INTRODUCTION: REORDERINGS FOR INCOMPLETE LU FACTORIZATIONS

▷▷ Reordering the matrix is an essential part of direct solution methods of sparse linear systems

Goal: Reduce factorization fill-in:

- Reverse Cuthill-McKee (RCM), Nested Dissection (ND), Minimum Degree (MD) ...

▷▷ For **incomplete** factorizations, the goal is amended by *Reduce factorization fill-in while obtaining “good” incomplete factorization.*

- “good” i.e. stable: if $A = LU + R$, then $\|(LU)^{-1}\|$ should be small not to magnify the error.

▷▷ **Method:** Consider reorderings of the matrix graph which take numeric values into account.

▷▷ Especially vital for highly unstructured and indefinite matrices.

- Non-symmetric reordering is employed in this case.

Recent work on nonsymmetric permutations

- Benzi, Haws, Tuma '00 [experiments with nonsymmetric reorderings and scaling for preconditioning].
- Duff, Koster, '99 [nonsymmetric permutation algorithms].
- Olschowska and Neumaier '96 [a permutation strategy for Gaussian elimination; discuss preconditioning].

SCOPE OF THE TALK

▷▷ Consider (almost) symmetrically structured matrices

- symmetric and nonsymmetric in value;
- ill-conditioned, non-diagonally dominant.

Related work: Benzi & Szyld & van Duin '99; Duff & Meurant '89.

▷▷ We consider the following value-based reorderings:

- Approximate Minimum Fill [as implemented by Patrick Amestoy]
- Minimum-Cost Spanning Tree (MST) algorithm.
- Diagonal Dominance Filtering (DDF) to preprocess the matrix for preconditioning

REORDERING OF EQUATIONS IN ARMS PRECONDITIONER

Independent set orderings permute a matrix into the form

$$\begin{pmatrix} B & F \\ E & C \end{pmatrix}$$

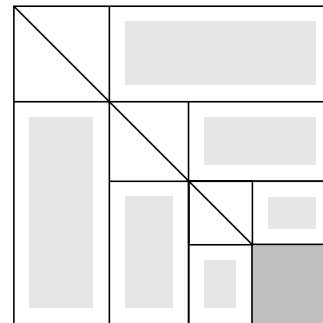
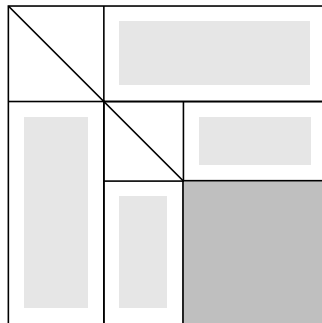
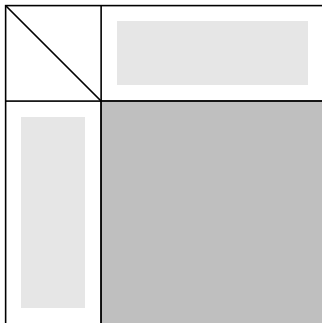
where **B** is diagonal.

▷▷ Unknowns associated with the **B** block form an independent set
(IS)

Observation: Reduced system obtained by eliminating the unknowns associated with IS, is still sparse since its coefficient matrix is the Schur Complement

$$S = C - EB^{-1}F$$

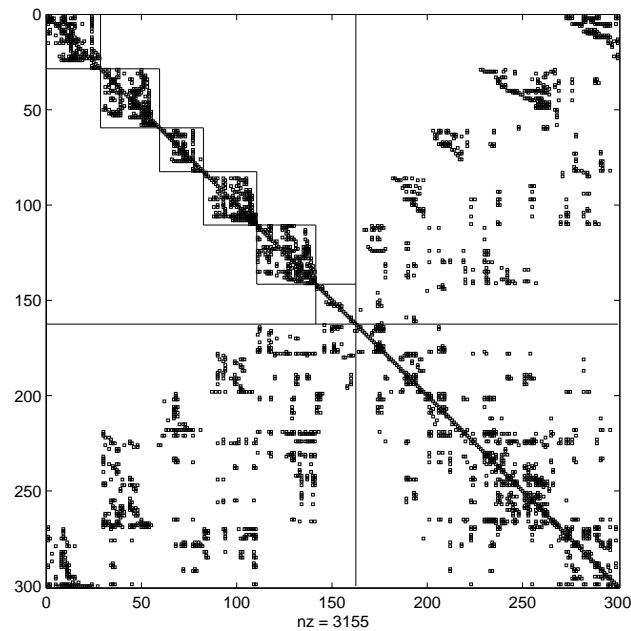
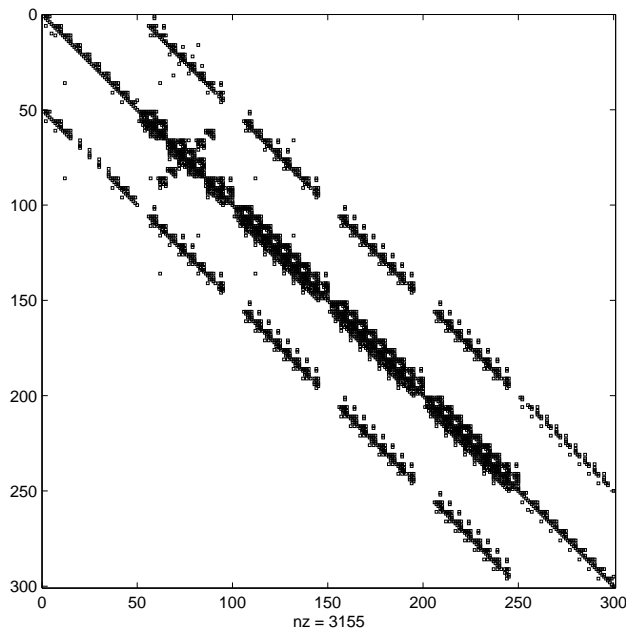
- ▷▷ Apply independent set reduction recursively.
- ▷▷ When reduced system is small enough, solve in a certain way
- ▷▷ Can devise an ILU factorization based on this strategy.



Idea: Use independent sets of “cliques”, or “aggregates”.

▷▷ Reorder equations such that independent sets come first.

Original matrix A and reordered matrix $A_0 = P_0^T A P_0$.



▷▷ Diagonal blocks can be treated as dense (small blocks)

▷▷ Can be treated as sparse blocks (ILU factorizations)

Factorization: The main factorization step is as follows

$$P_l^T A_l P_l = \begin{pmatrix} B_l & F_l \\ E_l & C_l \end{pmatrix} \approx \begin{pmatrix} L_l & 0 \\ G_l & I \end{pmatrix} \times \begin{pmatrix} U_l & W_l \\ 0 & A_{l+1} \end{pmatrix}$$

ALGORITHM : 1 . ARMS(A_{lev}) factorization

1. *If lev = last_lev then*
2. **Compute $A_{lev} \approx L_{lev} U_{lev}$**
3. *Else:*
4. **Find an independent set permutation P_{lev}**
5. **Apply permutation $A_{lev} := P_{lev}^T A_{lev} P$**
6. **Compute factorization**
7. **Call ARMS(A_{lev+1})**
8. *Endlf*

DIAGONAL DOMINANCE FILTRATION (DDF)

Essence: Permute weakly-diagonally non-dominant rows to the end of the matrix (to Schur Complement) given some **relative threshold**.

- Already implemented in ARMS
- Better performance than without DDF

▷▷ Define a vector of weights w :

$$w(i) = \hat{w}(i) / \max_{j=1, \dots, n} \hat{w}(j), \quad i = 1, \dots, n,$$

where

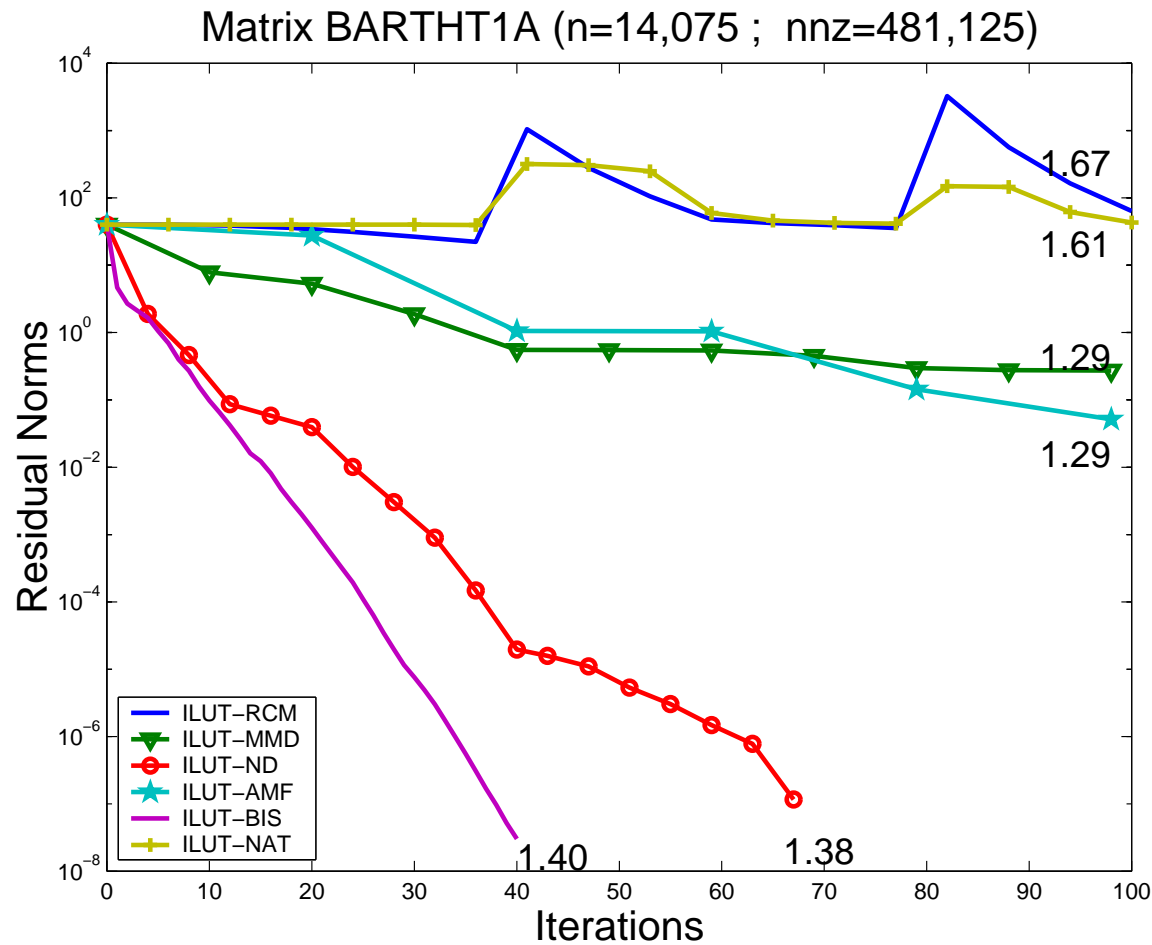
$$\hat{w}(i) = |a_{ii}| / \sum_{j=1}^n |a_{ij}|.$$

▷▷ Other filtration strategies may be used.

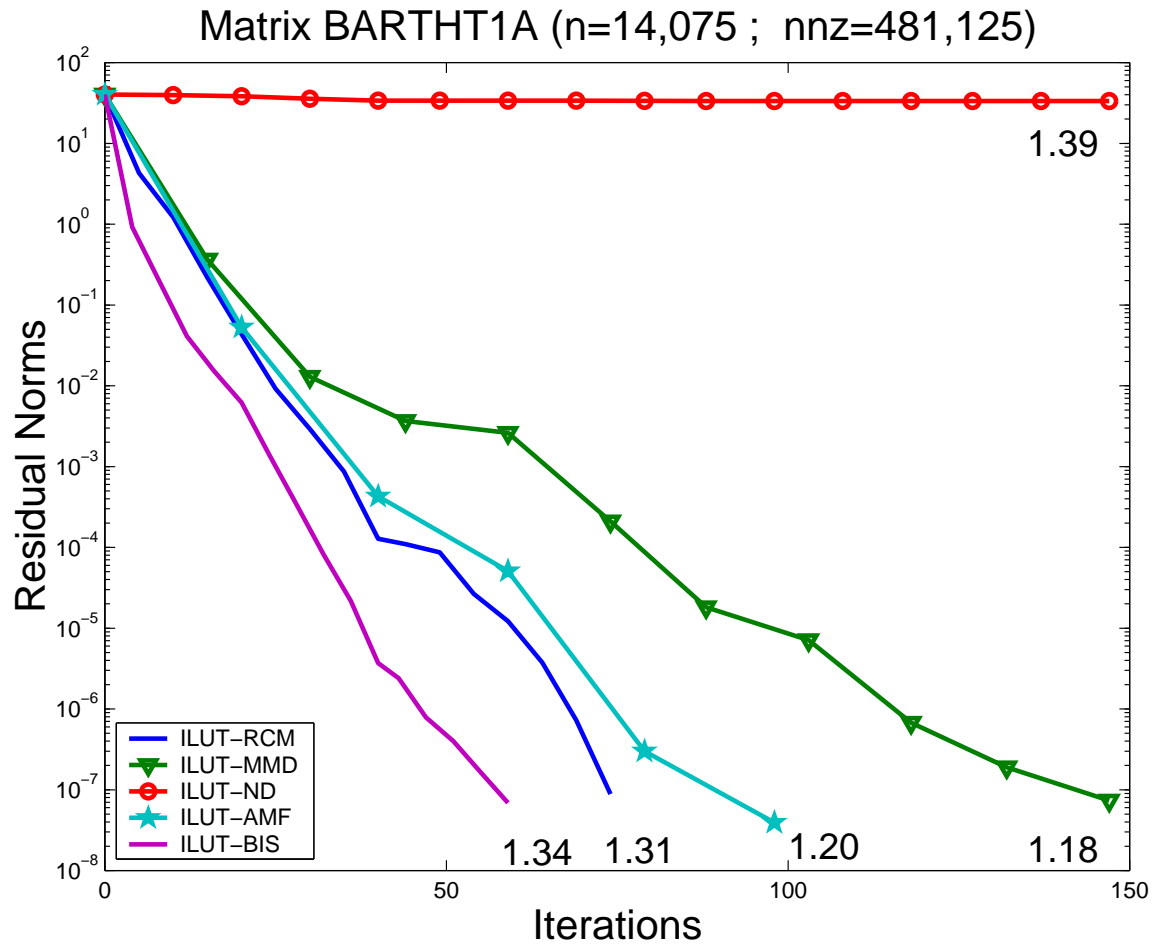
- Permute according to DDF of columns instead of rows.

EXAMPLES WITH AND WITHOUT FILTRATION

No DDF, Reordering + ILUT[30, 0.001] – Iterations



Filter down about 30% of rows



USE MINIMUM COST SPANNING TREE (MST) ALGORITHM WITH EDGE COSTS

Aim: to avoid small diagonal entries as pivots in block B.

Essence: Replace block-independent set reordering by a form of Prim's algorithm (based on Dijkstra's shortest path).

- ▷▷ Different strategies to define weights.
- ▷▷ Start MST from a pseudo-peripheral node as root.
- ▷▷ Record nodes by the distance from Root in reverse order.

▷▷ Strategy E1:

1. For each Row i :
2. If (a_{ij} and a_{ji} exist in A)
3. **Cost**(e_{ij}) = **Cost**(e_{ji}) = $|a_{ij}| \times |a_{ji}| / |a_{ii}|$
4. else
5. **Cost**(e_{ij}) = **Cost**(e_{ji}) = inf.

▷▷ Strategy E2:

1. For each Row i :
2. If (a_{ij} and a_{ji} exist in A)
3. **Cost(e_{ij}) = Cost(e_{ji}) = $1.0/|a_{ij}| \times |a_{ji}|$**
4. else
5. **Cost(e_{ij}) = Cost(e_{ji}) = inf.**

▷▷ Strategy E3:

1. For each Row i :
2. If (a_{ij} and a_{ji} exist in A , $i < j$)
3. $\text{Cost}(e_{ij}) = \text{Cost}(e_{ji}) =$
 $\frac{\sum_{k=0}^{nnz(i)} |a_{ik}| \sum_{k=0}^{nnz(j)} |a_{ki}|}{|a_{ii}|}$
4. else
5. $\text{Cost}(e_{ij}) = \text{Cost}(e_{ji}) = \text{inf.}$

▷▷ Strategy E4:

1. For each Row i :
2. if (a_{ij} and a_{ji} exist in A , $i < j$)
3. **Cost**(e_{ij}) = **Cost**(e_{ji}) =
 $(\max_k |a_{ik}|)(\max_k |a_{ki}|)/|a_{ii}|$
4. else
5. **Cost**(e_{ij}) = **Cost**(e_{ji}) = inf.

PREPROCESSING WITH ILU(0) FOR MST REORDERINGS

Performed in each preconditioning level (outer G.-E. step):

Before MST reordering

- ▷▷ Perform ILU(0) such that only diagonal entries are updated.
 - Fast and flexible (no sorting of entries in a row).
 - No extra data structures are required.

In MST reordering

- ▷▷ For assignment of edge costs:
 - Use **new** diagonal instead of matrix A diagonal in strategies E1, E3, and E4 (**i0_E1**, **i0_E3**, and **i0_E4**, respectively).

▷▷ For selection of a good block B.

- In DDF, use **new** diagonal instead of matrix A diagonal. (call it **i0_DDF**)
- **i0_DFmax**: Consider only the **new** diagonal $d'_i, \quad i = 1, \dots, n,$

1. Find the maximum element $d^* = \max_{i,i=1,\dots,n} d'_i$.
2. For each d'_i do
3. If the ratio d'_i/d^* is more than threshold,
4. Put row i in block B

TEST PROBLEM DESCRIPTION

Name	Size		S.	Dominance	Origin
s1rmq4m1	5,489	143,300	Y	0.39	CYLSHELL, R/t = 10
s3dkt3m2	90,449	1,921,955	Y	0.064	CYLSHELL, R/t = 1,000
s3rmt3m3	5,357	106,526	Y	0.13	CYLSHELL, R/t = 1,000
COQUE3E3	3,601	36,654	N	0.029	Thin shell
FIDAP006	1,651	49,479	N	0.018	CFD
FIDAP023	1,409	43,481	N	0.02	CFD
FIDAP026	2,163	93,749	N	0.033/ 0.048	CFD
SMALL1	5,076	169,691	N	0.018 / 0.0004	Hydrodynamics

INPUT PARAMETERS

```
inputparams      Tue Jun 11 02:09:58 2002      1
1                | Diag. Scaling before PILUT [D1, D2]    0:no 1:yes
1                | Diag. Scaling in last level [D1, D2] 0:no 1:yes
1                | column perm for last level [0 = ILUT, 1 =ILUTP]
0.201           | tol_ind = tolerance used in independent set -
  2              | nlev = number of levels
10              | bsize = block size for block independent sets
0.0001          | droptol[0-3] L, U, L^{-1} F, E U^{-1} -
0.0010          | droptol[4] for Schur complements at each level
0.0010          | droptol[5] for ILUT in last level Sc. comple
20              | lfil[0] L, U, L^{-1} F, E U^{-1}
20              | lfil[4] Schur
20              | lfil[5] ILUT L, ILUT U in Schur
50              | dimension of Krylov subspace in (outer) FGMRES
300             | maxits in outer fgmres.
0.1             | tolerance for last level iterative solver
0               | Krylov subs dim for last level iterations [0 -> no iter's]
```

COMPARISON OF REORDERING VARIATIONS IN ARMS

COQUE3E3 problem – Preprocessing with DDF

Method	Fill	log ₁₀ ($\tilde{L}\tilde{U}^{-1}$)	Iter				
arms_AMF	2.03	≈ 0	25				
arms_MMD	2.03	≈ 0	25				
arms_BIS	2.55	≈ 0	42				
arms_RCM	2.72	≈ 0	21				
Method	Fill	log ₁₀ ($\tilde{L}\tilde{U}^{-1}$)	Iter	Method	Fill	log ₁₀ ($\tilde{L}\tilde{U}^{-1}$)	Iter
E1	2.62	≈ 0	35	i0_E1	2.63	≈ 0	23
E2	2.96	≈ 0	35				
E3	2.64	≈ 0	20	i0_E3	2.64	≈ 0	19
E4	2.64	≈ 0	23	i0_E4	2.58	≈ 0	21

FIDAP023 problem – Preprocessing with DDF

Method	Fill	$\log_{10}(\ \tilde{L}\tilde{U}^{-1}\)$	Iter				
arms_AMF	1.31	6.89	23				
arms_MMD	1.29	6.88	23				
arms_BIS	1.56	9.99	81				
arms_RCM	1.41	6.91	21				
Method	Fill	$\log_{10}(\ \tilde{L}\tilde{U}^{-1}\)$	Iter	Method	Fill	$\log_{10}(\ \tilde{L}\tilde{U}^{-1}\)$	Iter
E1	1.46	6.90	24	i0_E1	1.43	6.90	24
E2	1.50	6.90	24				
E3	1.49	6.89	22	i0_E3	1.49	6.90	21
E4	1.49	6.89	22	i0_E4	1.49	6.93	22

SMALL1 problem – Preprocessing with DDF ¹

Method	Fill	$\log_{10}(\ L\tilde{U}^{-1}\)$	Iter				
arms_AMF	1.42	5.93	186				
arms_MMD	1.40	4.46	148				
arms_BIS	1.43	3.55	218				
arms_RCM	1.54	3.95	153				
Method	Fill	$\log_{10}(\ L\tilde{U}^{-1}\)$	Iter	Method	Fill	$\log_{10}(\ L\tilde{U}^{-1}\)$	Iter
E1	1.54	4.24	177	i0_E1	1.55	4.00	157
E2	1.55	4.00	157				
E3	1.54	5.04	176	i0_E3	1.54	3.65	176
E4	1.53	6.86	> 300	i0_E4	1.54	3.74	177

¹ $tol_{DD} = 0.05$; For $tol_{DD} = 0.2$, no method but ARMS_BIS converged

DEPENDENCE ON TOL_{DD}

FIDAP006 problem for $tol_{DD} = 0.2$ and $tol_{DD} = 0.9^2$

Method	Fill-ratio		Iter		Factorize, sec		Solve, sec	
	0.2	0.9	0.2	0.9	0.2	0.9	0.2	0.9
arms_AMF	2.170	3.525	27	†	1.30	1.25	0.37	†
arms_MMD	2.226	3.526	33	100	1.06	1.28	0.44	1.24
arms_MST	3.154	3.528	29	93	2.52	1.27	0.55	1.18
arms_BIS	2.443	3.523	21	†	1.84	1.27	0.32	†
arms_RCM	2.445	3.527	49	143	1.82	1.27	0.73	1.79

FIDAP026 problem for $tol_{DD} = 0.05$ and $tol_{DD} = 0.2$

Method	Fill-ratio		Iter		Factorize, sec		$\log_{10}(\ L\tilde{U}^{-1}\)$	
	0.05	0.2	0.05	0.2	0.05	0.2	0.05	0.2
arms_AMF	0.85	0.88	34	> 300	1.35	1.38	5.12	7.97
arms_MMD	0.84	0.87	35	> 300	1.33	1.36	4.76	22.0
i0_E4	0.97	0.99	34	> 300	1.41	1.42	4.97	20.4
arms_BIS	1.16	1.15	> 300	> 300	1.56	1.59	7.64	5.59
arms_RCM	0.94	0.98	32	> 300	1.64	1.61	5.78	21.5

CYLSHELL problems – Preprocessing with DDF

	Iterations				
	arms_AMF	arms_MMD	arms_MST	arms_BIS	arms_RCM
S1R	7	7	6	7	5
S3R	5	6	5	5	3
S3D	5	5	4	5	3

SUMMARY – OBSERVATIONS

▷▷ Experiments are in agreement with previous work: for structurally symmetric matrices a fill-reducing preprocessing is beneficial

- for unstructured matrices this is not enough

▷▷ Numerical values may improve convergence when symmetric permutations are used

▷▷ A Diagonal Dominance Filtering strategy is essential

- Drawback: Methods are very sensitive to the filtering tolerance.

▷▷ The reorderings are more efficient when used on larger blocks.

- Trade-off in selecting filtering tolerance.