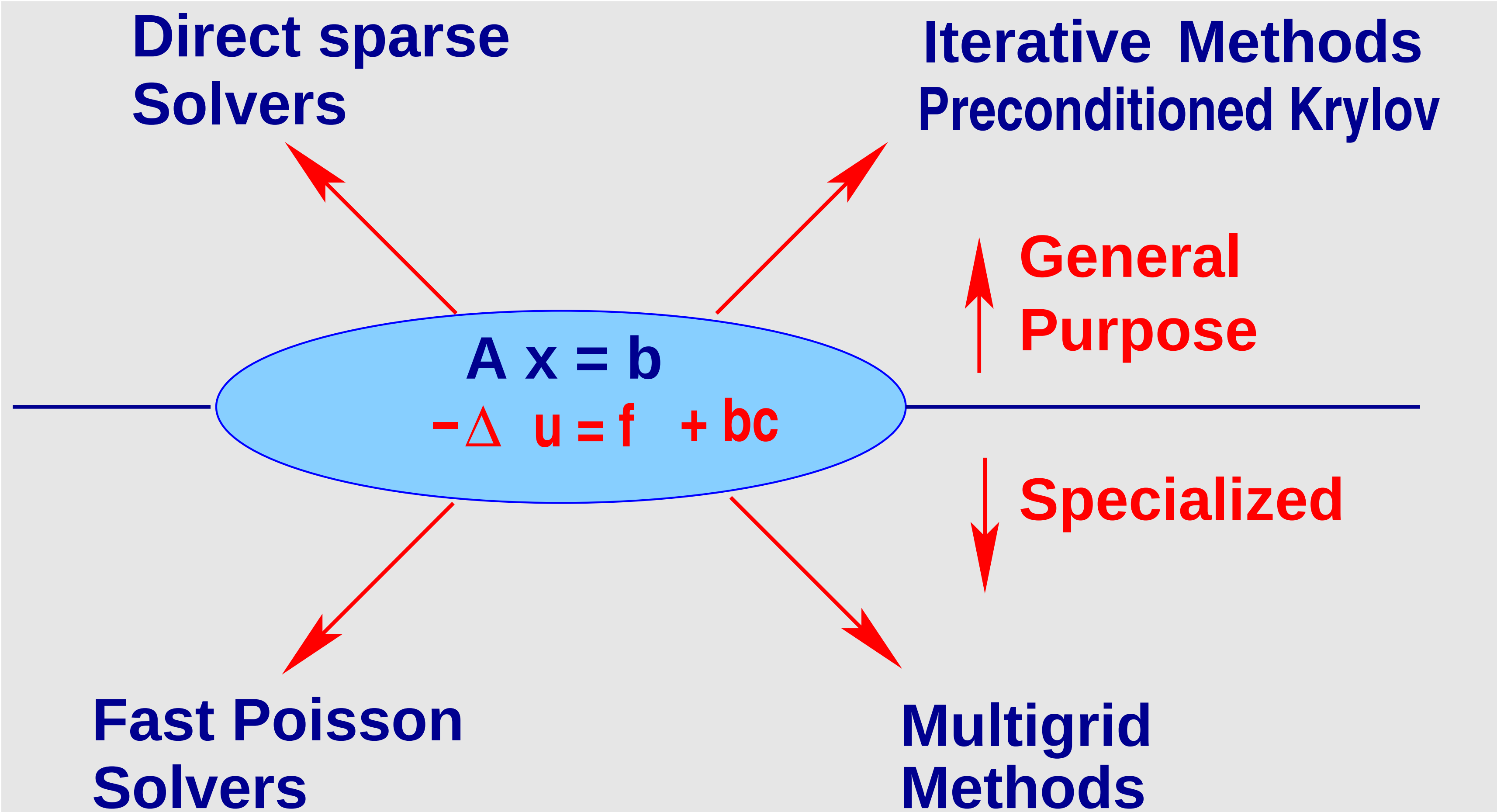
Recent progress in preconditioned Krylov subspace methods

Yousef Saad
University of Minnesota
Dept. of Computer Science and Engineering

QUT – Sep. 14th, 2007

Introduction



Linear system solvers: specialized versus general purpose

Introduction: Linear System Solvers

- **Much of recent work on solvers has focussed on:**

(1) Parallel implementation – scalable performance

(2) Improving robustness, developing more general preconditioners

A few observations

- Problems are getting harder for Sparse Direct methods (more 3-D models, much bigger problems,..)

- Problems are also getting difficult for iterative methods Cause: more complex models - away from Poisson

- Researchers in iterative methods are borrowing techniques from direct methods: $\longrightarrow$ preconditioners

- The inverse is also happening: Direct methods are being adapted for use as preconditioners

Difficult linear systems

- **Traditionally: two areas have been difficult for iterative solvers:**

(a) Problems from circuit simulation

(b) Problems from structures

- **Recently, there has been excellent progress made in developing good preconditioners for both classes of problems..**

One of the main tools: use of nonsymmetric permutations.

An overview of recent progress on ILU

- Bollhöfer defined rigorous dropping strategies [Bollhöfer 2002]
- Approximate inverse methods [limited success]
- Use of different forms of LU factorizations [ILUC, N. Li, YS, Chow]
- Vaidya preconditioners – for problems in structures [very successful in industry]
- Support theory for preconditioners
- Nonsymmetric permutations –

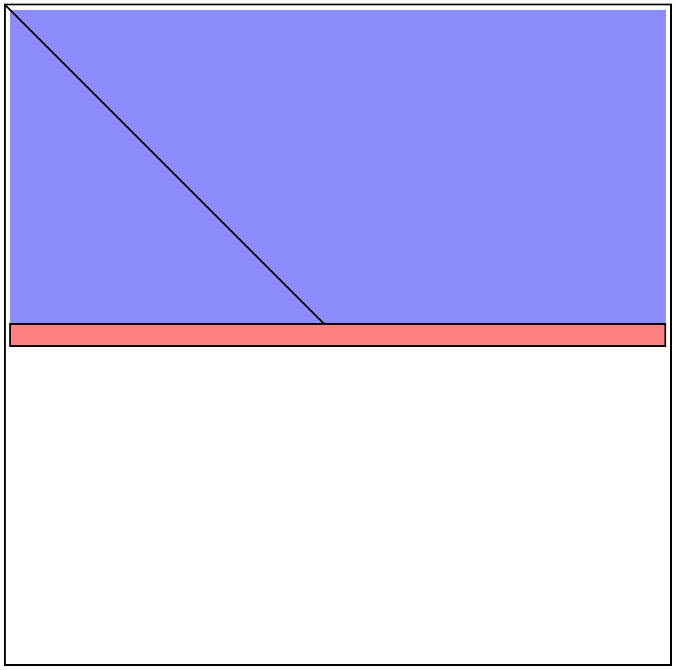
CROUT VERSIONS OF ILUT

Crout-based ILUT (ILUTC)

Background: **ILU codes use so-called ikj- version of Gaussian elimination [equiv. to left looking column LU]**

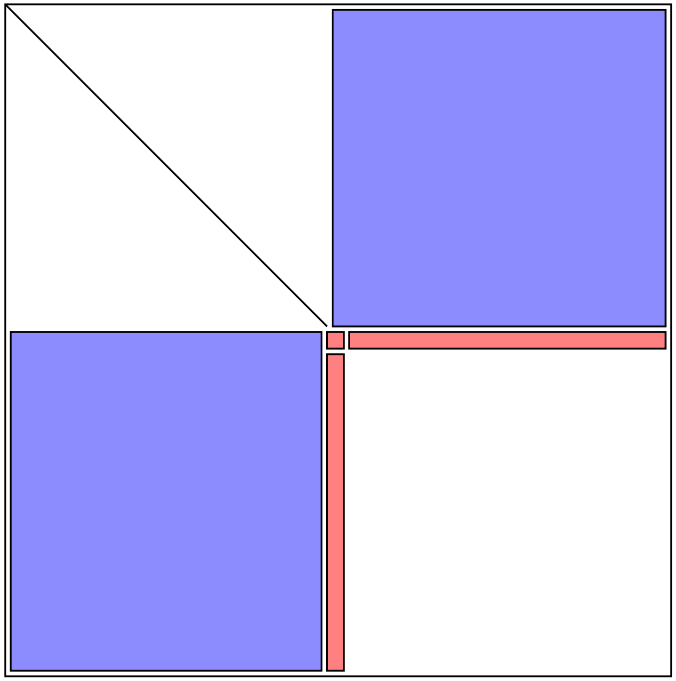
ALGORITHM : 1 . ***GE – IKJ Variant***

1. ***For*** $i = 2, \ldots, n$ ***Do:***
2. ***For*** $k = 1, \ldots, i-1$ ***Do:***
3. $a_{ik} := a_{ik}/a_{kk}$
4. ***For*** $j = k+1, \ldots, n$ ***Do:***
5. $a_{ij} := a_{ij} - a_{ik} * a_{kj}$
6. ***EndDo***
7. ***EndDo***
8. ***EndDo***



Pb: entries in L must be accessed from left to right

Terminology: Crout versions of LU compute the k-th row of U and the k-th column of L at the k-th step.



Computational pattern

Red = part computed at step k

Blue = part accessed

Main advantages:

1. Less expensive than ILUT (avoids sorting)
2. Allows better techniques for dropping

References:

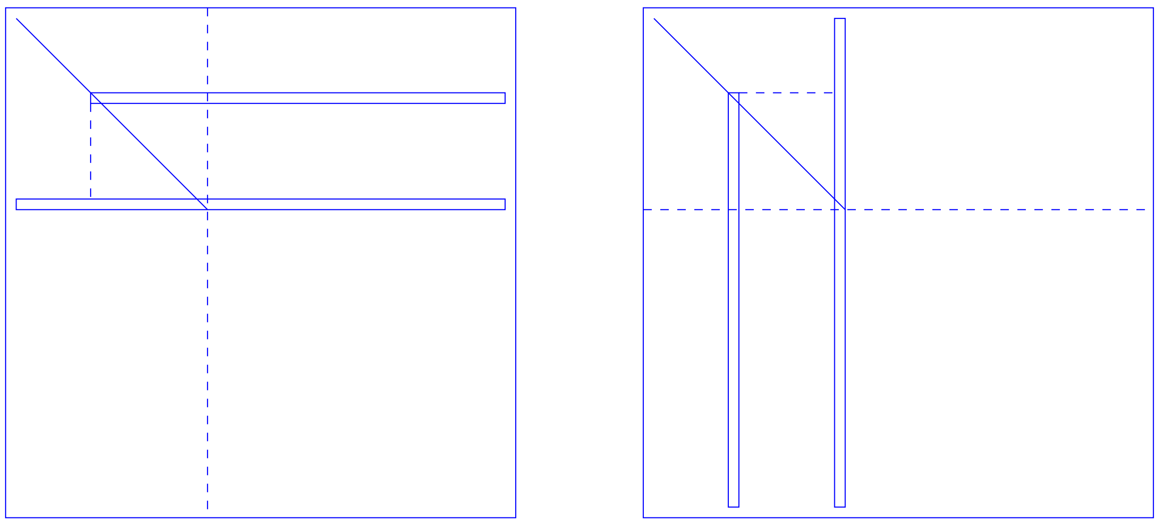
[1] M. Jones and P. Plassman. An improved incomplete Choleski factorization. *ACM Transactions on Mathematical Software*, 21:5–17, 1995.

[2] S. C. Eisenstat, M. H. Schultz, and A. H. Sherman. Algorithms and data structures for sparse symmetric Gaussian elimination. *SIAM Journal on Scientific Computing*, 2:225–237, 1981.

[3] M. Bollhöfer. A robust ILU with pivoting based on monitoring the growth of the inverse factors. *Linear Algebra and its Applications*, 338(1–3):201–218, 2001.

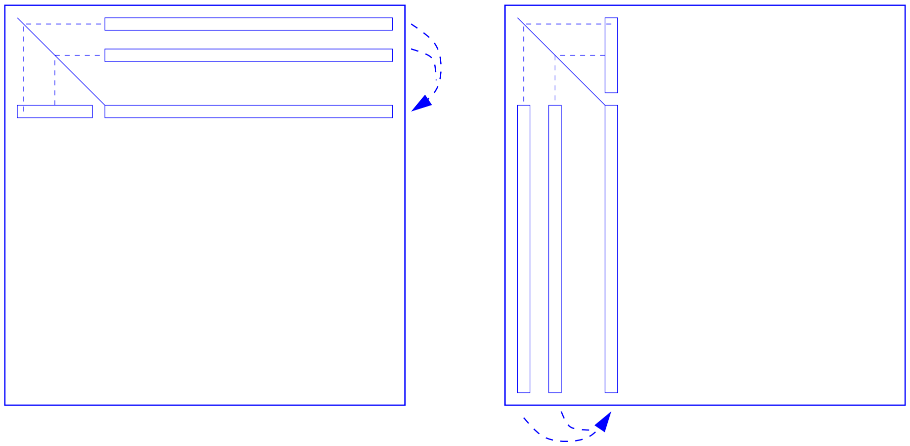
Crout LU (dense case)

- **Go back to delayed update algorithm (IKJ alg.) and observe: we could do both a column and a row version**



- **Left: U computed by rows. Right: L computed by columns**

Note: The entries $1 : k - 1$ in the k-th row in left figure need not be computed. Available from already computed columns of L. Similar observation for L (right).

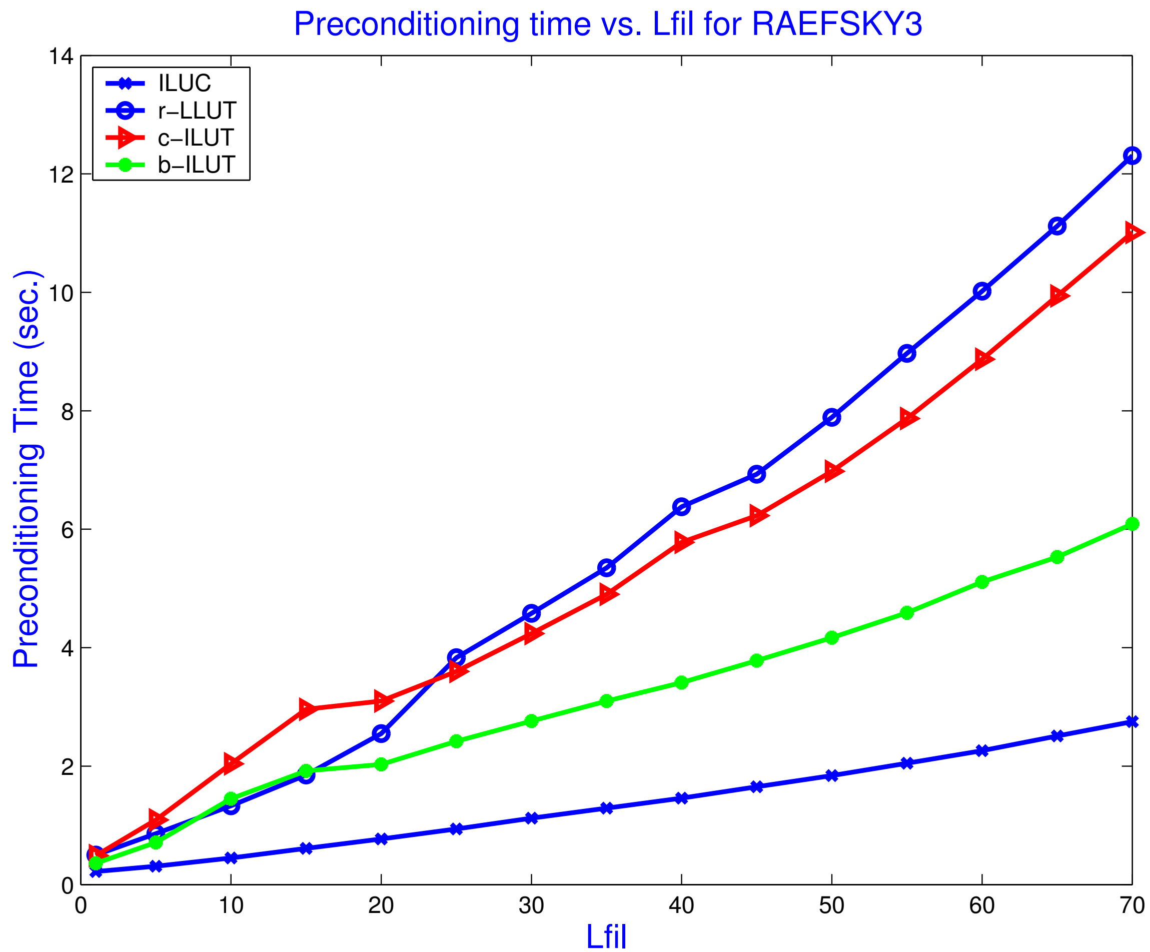


ALGORITHM : 2 . *Crout LU Factorization (dense case)*

1. **For** $k = 1 : n$ **Do :**
2. **For** $i = 1 : k - 1$ **and if** $a_{ki} \neq 0$ **Do :**
3. $a_{k,k:n} = a_{k,k:n} - a_{ki} * a_{i,k:n}$
4. **EndDo**
5. **For** $i = 1 : k - 1$ **and if** $a_{ik} \neq 0$ **Do :**
6. $a_{k+1:n.k} = a_{k+1:n,k} - a_{ik} * a_{k+1:n,i}$
7. **EndDo**
8. $a_{ik} = a_{ik}/a_{kk}$ **for** $i = k + 1, ..., n$
9. **EndDo**

Crout ILUT

- Can derive incomplete versions – by adding dropping.
- Data structure from [Jones-Platzman] - clever implementation



Inverse-based dropping strategies

- Method developed mainly by Matthias Bollhöffer

Observation: norm of inverses of the factors is more important than the errors in the factors themselves: If $A = \tilde{L}\tilde{U} + E$ then

$$\tilde{L}^{-1}A\tilde{U}^{-1} = I + \tilde{L}^{-1}E\tilde{U}^{-1}$$

- In many cases $\|\tilde{L}^{-1}\|$ and $\|\tilde{U}^{-1}\|$ are *very* large $\rightarrow$ **Bad.**

- In contrast assume $A = LU$ = exact LU factorization and

$$\tilde{L}^{-1} = L^{-1} + X \qquad \tilde{U}^{-1} = U^{-1} + Y, \quad \text{Then:}$$

$$\tilde{L}^{-1}A\tilde{U}^{-1} = (L^{-1} + X)A(U^{-1} + Y) = I + AY + XA + XY.$$

- X, Y small $\longrightarrow$ preconditioned matrix close to identity

- Let L_k = matrix of the first k rows of L and the last $n-k$ rows of the identity matrix.

- Consider a term l_{jk} with $j > k$ that is dropped at step k. Perturbed matrix $\tilde{L}_k$ differs from L_k by $l_{jk}e_je_k^T$. Note: $L_ke_j = e_j$ so

$$\tilde{L}_k = L_k - l_{jk}e_je_k^T = L_k(I - l_{jk}e_je_k^T) \quad \rightarrow$$

$$\tilde{L}_k^{-1} = (I - l_{jk}e_je_k^T)^{-1}L_k^{-1} = L_k^{-1} + l_{jk}e_je_k^T L_k^{-1}.$$

- j-th row of inverse of L_k perturbed by $\boxed{l_{jk} \text{ times } k\text{-th row of } L_k^{-1}.}$

- Need to limit the norm of this perturbing row, i.e.,

$$|l_{jk}| \; \|e_k^T L_k^{-1}\|_\infty \text{ should be small}$$

- L^{-1} is not available. Bollhöfer's idea: use techniques for estimating condition numbers

ALGORITHM : 3 . *Estimating the norms* $\|e_k^T L^{-1}\|_\infty$

1. *Set* $\xi_1 = 1$, $\nu_i = 0, i = 1, \ldots, n$
2. *For* $k = 2, \ldots, n$ *do*
3. $\quad \xi_+ = 1 - \nu_k$; $\xi_- = -1 - \nu_k$;
4. $\quad$ *if* $|\xi_+| > |\xi_-|$ *then* $\xi_k = \xi_+$ *else* $\xi_k = \xi_-$
5. $\quad$ *For* $j = k + 1 : n$ *and for* $l_{jk} \neq 0$ *Do*
6. $\qquad \nu_j = \nu_j + \xi_k l_{jk}$
7. $\quad$ *EndDo*
8. *EndDo*

► **Idea fits very well with Crout ILU [Na Li, YS, E. Chow, 2004]**

APPROXIMATE INVERSES

Approximate Inverse preconditioners

Motivation:

- **L - U solves in ILU may be ‘unstable’**
- **Parallelism in L-U solves limited**

Idea: **Approximate the inverse of A directly $M \approx A^{-1}$**

Different forms:

- **Right preconditioning:** **Find M such that**

$$AM \approx I$$

- **Left preconditioning:** **Find M such that**

$$MA \approx I$$

- **Factored approximate inverse:** **Find L and U s.t.**

$$LAU \approx D$$

Some references

- Benson and Frederickson ('82): approximate inverse using stencils
- Grote and Simon ('93): Choose M to be banded
- Cosgrove, Díaz and Griewank ('91) : Procedure to add fill-ins to M
- Kolotilina and Yeremin ('93) : Factorized symmetric preconditionings $M = G_L^T G_L$
- Huckle and Grote ('95) : Procedure to find good pattern for M
- Chow and Saad ('95): Find pattern dynamically by using dropping.
- M. Benzi & Tuma ('96, '97,..): Factored app. inv.

One (of many) options: try to find M which minimizes

$$\boxed{\|I - AM\|_F}$$

Note: Minimization problem to find M decouples

- Problem decouples into n independent least-squares systems
- In each of these systems the matrix and RHS are sparse

Two paths:

1. Can find a good sparsity pattern for M first then compute M using this patters.
2. Can find the pattern dynamically [similar to ILUT]

Approximate inverses for block-partitioned matrices

Motivation. **Domain Decomposition**

$$\begin{pmatrix} B_1 & & & & F_1 \\ & B_2 & & & F_2 \\ & & \ddots & & \vdots \\ & & & B_n & F_n \\ E_1 & E_2 & \cdots & E_n & C \end{pmatrix} \equiv \begin{pmatrix} B & F \\ E & D \end{pmatrix}$$

Note: $$\begin{pmatrix} B & F \\ E & C \end{pmatrix} = \begin{pmatrix} B & 0 \\ E & S \end{pmatrix} \begin{pmatrix} I & B^{-1}F \\ 0 & I \end{pmatrix}$$

in which S is the Schur complement,

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

One idea: Compute $M = LU$ in which

$$L = \begin{pmatrix} B & 0 \\ E & M_S \end{pmatrix} \quad \text{and} \quad U = \begin{pmatrix} I & B^{-1}F \\ 0 & I \end{pmatrix}$$

- M_S = some preconditioner to S.

One option: $M_S = \tilde{S}$ = sparse approximation to S

$$\tilde{S} = C - EY \quad \text{where} \quad Y \approx B^{-1}F$$

- Need to find a sparse matrix Y such that

$$\boxed{BY \approx F}$$

where F and B are sparse.

NONSYMMETRIC REORDERINGS

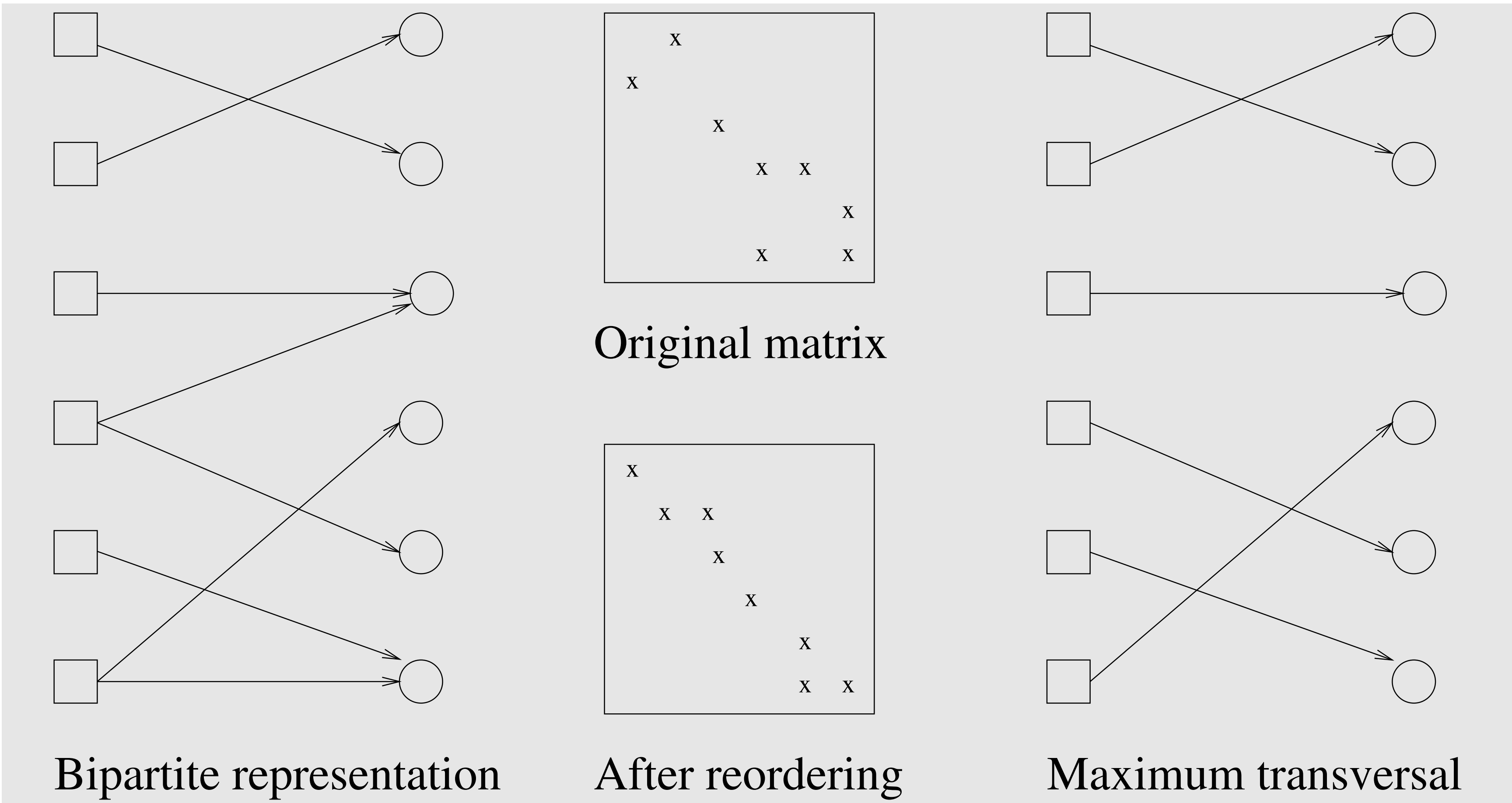
Enhancing robustness: One-sided permutations

► Very useful techniques for matrices with extremely poor structure. Not as helpful in other cases.

Previous work:

- Benzi, Haws, Tuma '99 [compare various permutation algorithms in context of ILU]
- Duff, Koster, '99 [propose various permutation algorithms. Also discuss preconditioners]
- Duff '81 [Propose max. transversal algorithms. Basis of many other methods. Also Hopcroft & Karp '73, Duff '88]

Transversals - bipartite matching: Find (maximal) set of ordered pairs (i, j) s.t. $a_{ij} \neq 0$ and i and j each appear only once (one diagonal element per row/column). Basis of many algorithms.



Criterion: Find a (column) permutation π such that

$$\prod_{i=1}^{n} |a_{i,\pi(i)}| = \max$$

Olchowsky and Neumaier '96 translate this into

$$\min_{\pi} \sum_{i=1}^{n} c_{i,\pi(i)} \quad \text{with } c_{ij} = \begin{cases} \log \left[\frac{\|a_{:,j}\|_\infty}{|a_{ij}|} \right] & \text{if } a_{ij} \neq 0 \\ +\infty & \text{else} \end{cases}$$

- **Dual problem is solved:**

$$\max_{u_i, u_j} \{ \sum_{i=1}^{n} u_i + \sum_{j=1}^{n} u_j \} \quad \text{subject to:} \quad c_{ij} - u_i - u_j \geq 0$$

- **Algorithms utilize depth-first-search to find max transversals.**
- **Many variants. Best known code: Duff & Koster's MC64**

NONSYMMETRIC REORDERINGS: MULTILEVEL FRAMEWORK

Background: Independent sets, ILUM, ARMS

Independent set orderings permute a matrix into the form

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

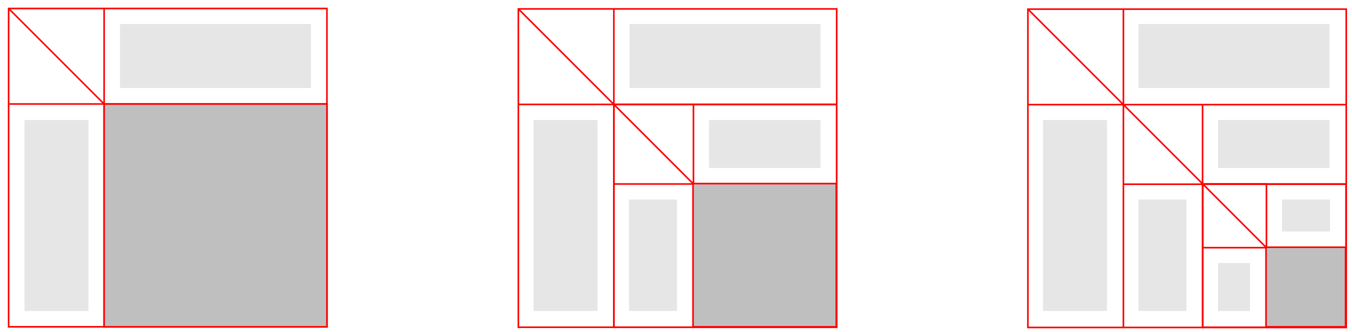
where B is a diagonal matrix.

- **Unknowns associated with the B block form an independent set (IS).**
- **IS is maximal if it cannot be augmented by other nodes to form another IS.**
- **Finding a maximal independentg set is inexpensive**

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

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

- Idea: apply IS set reduction recursively.
- When reduced system small enough solve by any method
- ILUM: ILU factorization based on this strategy. YS '92-94.

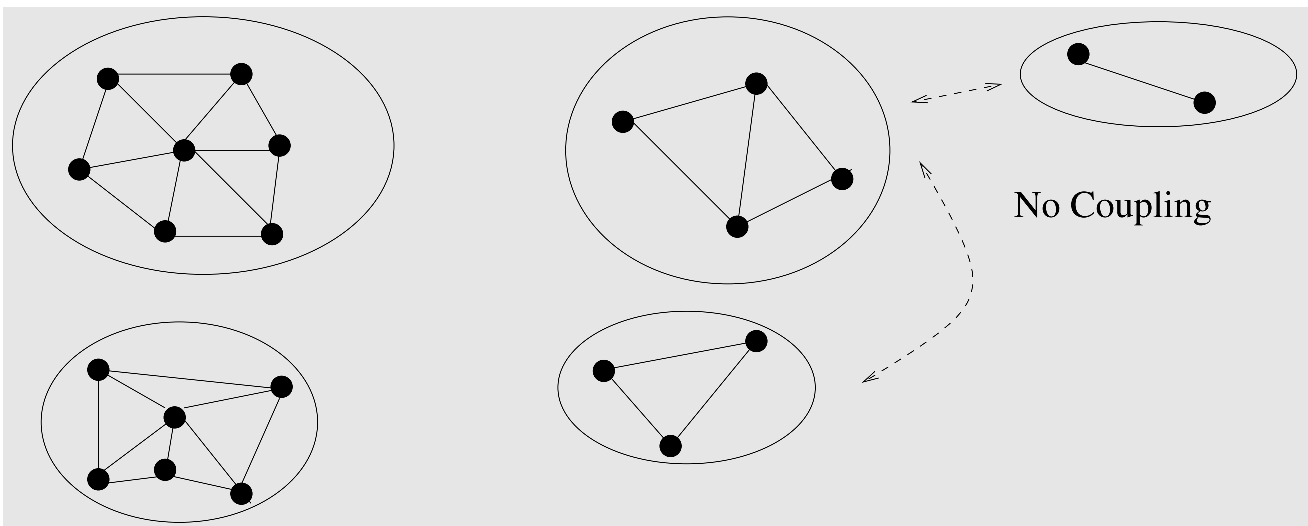


- See work by [Botta-Wubbs '96, '97, YS'94, '96, Leuze '89,..]

Group Independent Sets / Aggregates

Main goal: generalize independent sets to improve robustness

Main idea: use "cliques", or "aggregates". No coupling between the aggregates.

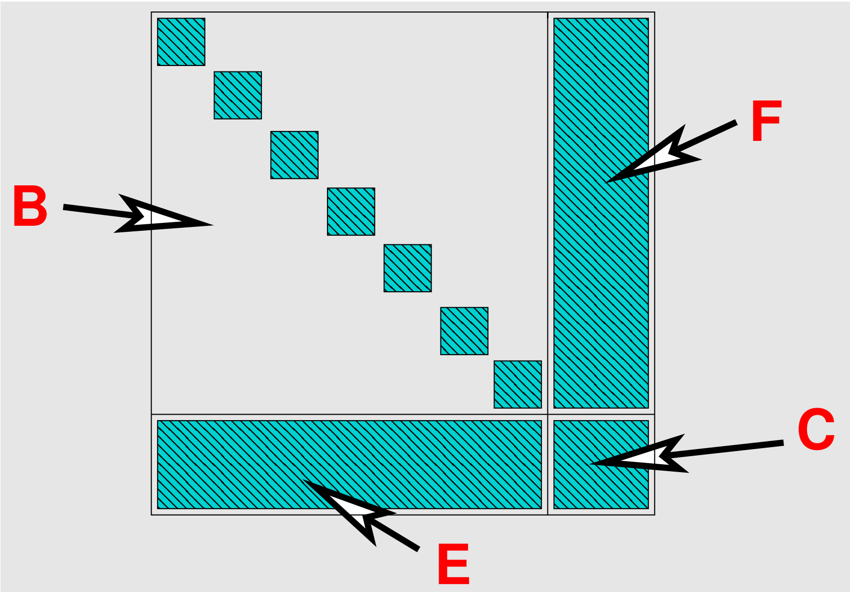


- Label nodes of independent sets first

Algebraic Recursive Multilevel Solver (ARMS)

- Typical shape of reordered matrix:

$$PAP^T = \begin{pmatrix} B & F \\ E & C \end{pmatrix} =$$



- Block factorize: $\begin{pmatrix} B & F \\ E & C \end{pmatrix} = \begin{pmatrix} L & 0 \\ EU^{-1} & I \end{pmatrix} \begin{pmatrix} U & L^{-1}F \\ 0 & S \end{pmatrix}$

- $S = C - EB^{-1}F$ = Schur complement + dropping to reduce fill

- Next step: treat the Schur complement recursively

Algebraic Recursive Multilevel Solver (ARMS)

Level l Factorization:

$$\begin{pmatrix} B_l & F_l \\ E_l & C_l \end{pmatrix} \approx \begin{pmatrix} L_l & 0 \\ E_l U_l^{-1} & I \end{pmatrix} \begin{pmatrix} I & 0 \\ 0 & A_{l+1} \end{pmatrix} \begin{pmatrix} U_l & L_l^{-1} F_l \\ 0 & I \end{pmatrix}$$

- **L-solve $\sim$ restriction; U-solve $\sim$ prolongation.**

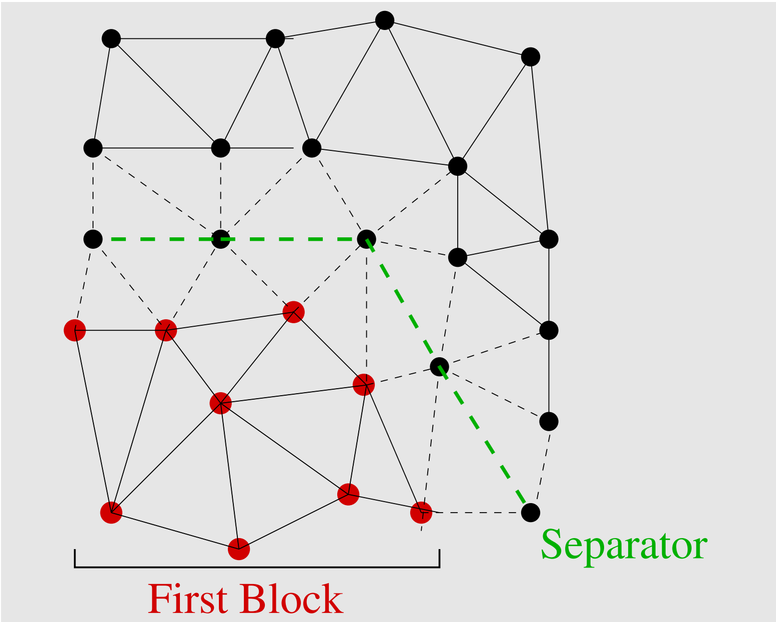
- **Perform above block factorization recursively on A_{l+1}**

- **Blocks in B_l treated as sparse. Can be large or small.**

- **Algorithm is fully recursive**

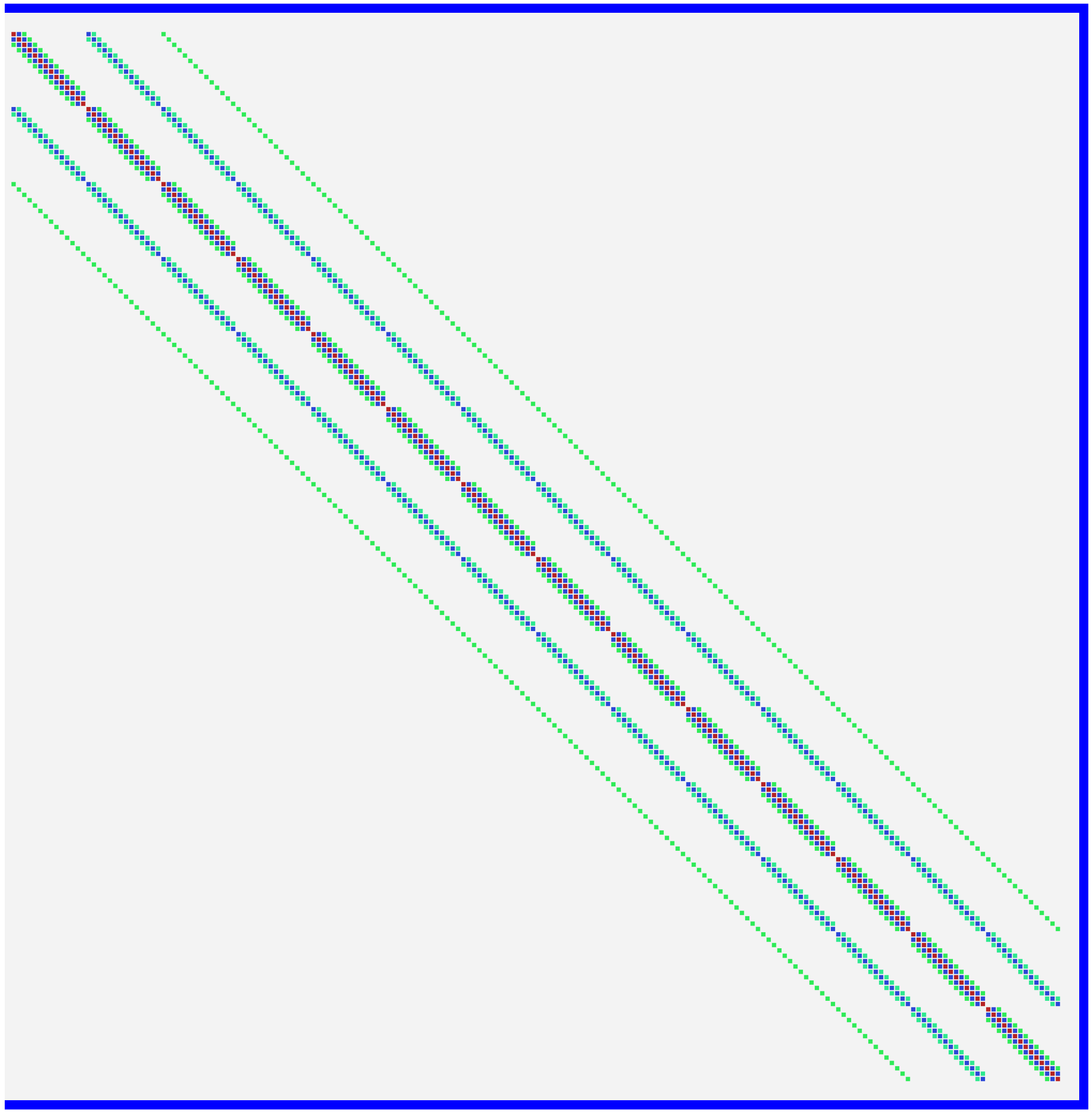
- **Stability criterion in block independent sets algorithm**

Group Independent Set reordering

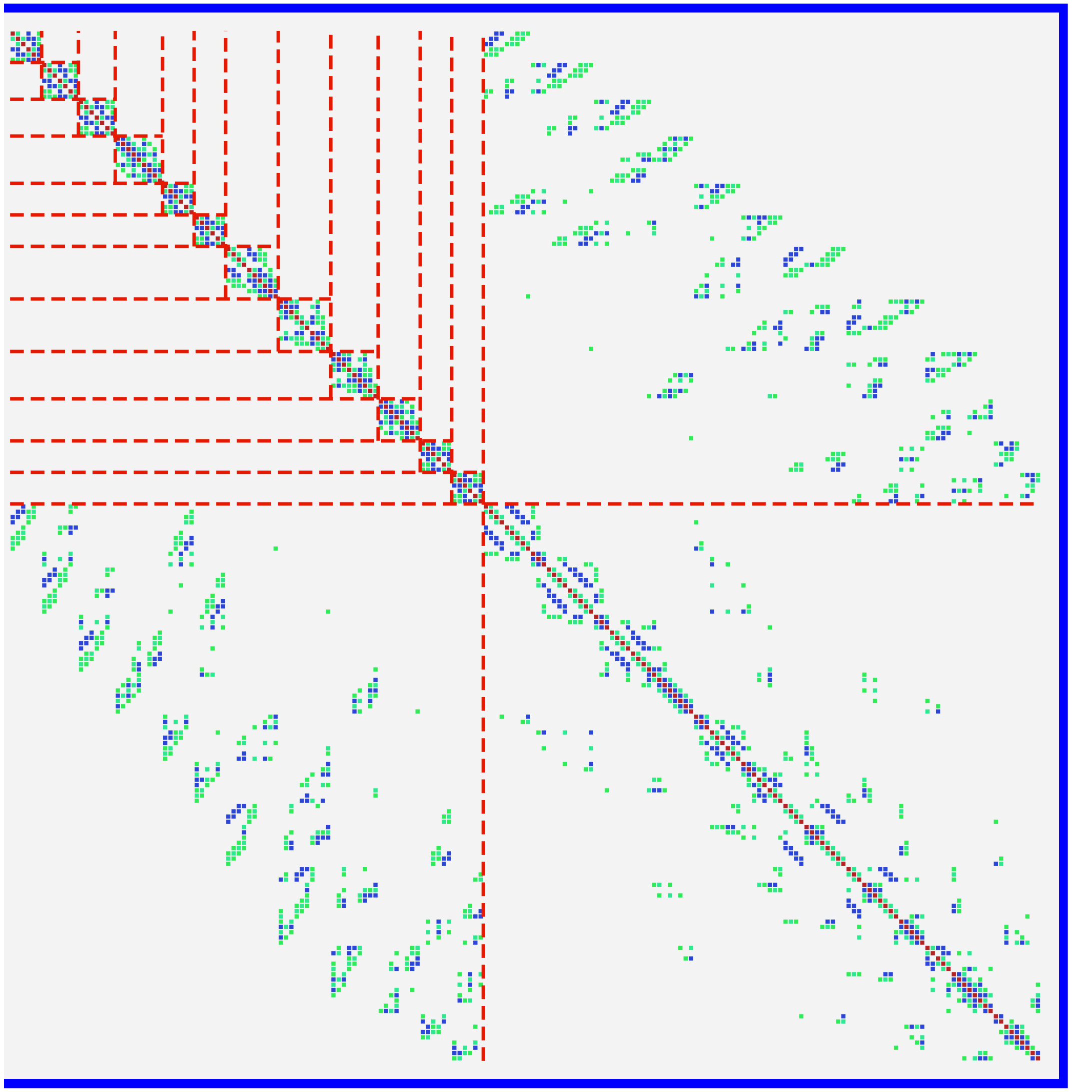


Simple strategy: Level taversal until there are enough points to form a block. Reverse ordering. Start new block from non-visited node. Continue until all points are visited. Add criterion for rejecting "not sufficiently diagonally dominant rows."

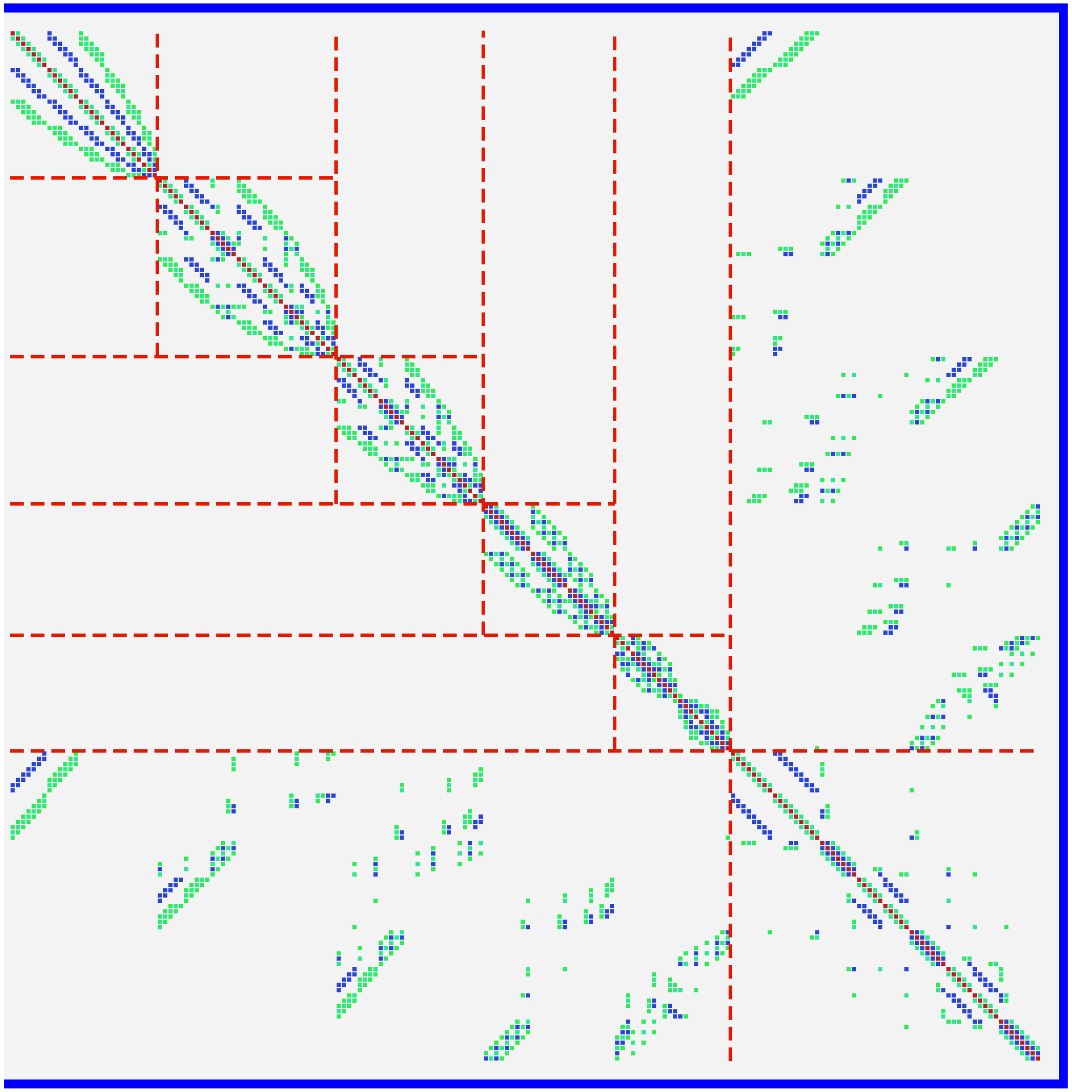
Original matrix



Block size of 6



Block size of 20



Two-sided permutations with diag. dominance

Idea: ARMS + exploit nonsymmetric permutations

- No particular structure or assumptions for B block

- Permute rows * and * columns of A. Use two permutations P (rows) and Q (columns) to transform A into

$$PAQ^T = \begin{pmatrix} B & F \\ E & C \end{pmatrix}$$

P, Q is a pair of permutations (rows, columns) selected so that the B block has the 'most diagonally dominant' rows (after nonsym perm) and few nonzero elements (to reduce fill-in).

Multilevel framework

- At the l-th level reorder matrix as shown above and then carry out the block factorization 'approximately'

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

where

$$B_l \approx L_l U_l$$

$$A_{l+1} \approx C_l - (E_l U_l^{-1})(L_l^{-1} F_l) \, .$$

- As before the matrices $E_l U_l^{-1}, L_l^{-1} F_l$ or their approximations

$$G_l \approx E_l U_l^{-1} \, , \qquad W_l \approx L_l^{-1} F_l$$

need not be saved.

Interpretation in terms of complete pivoting

Rationale: Critical to have an accurate and well-conditioned B block [Bollhöfer, Bollhöfer-YS'04]

- Case when B is of dimension 1 $\longrightarrow$ a form of complete pivoting ILU. Procedure $\sim$ block complete pivoting ILU

Matching sets: define B block. $\mathcal{M}$ is a set of n_M pairs (p_i, q_i) where $n_M \leq n$ with $1 \leq p_i, q_i \leq n$ for $i = 1, \ldots, n_M$ and

$$p_i \neq p_j, \text{ for } i \neq j \qquad q_i \neq q_j, \text{ for } i \neq j$$

- When $n_M = n \longrightarrow$ (full) permutation pair (P, Q). A partial matching set can be easily completed into a full pair (P, Q) by a greedy approach.

Matching - preselection

Algorithm to find permutation consists of 3 phases.

(1) **Preselection:** to filter out poor rows (dd. criterion) and sort the selected rows.

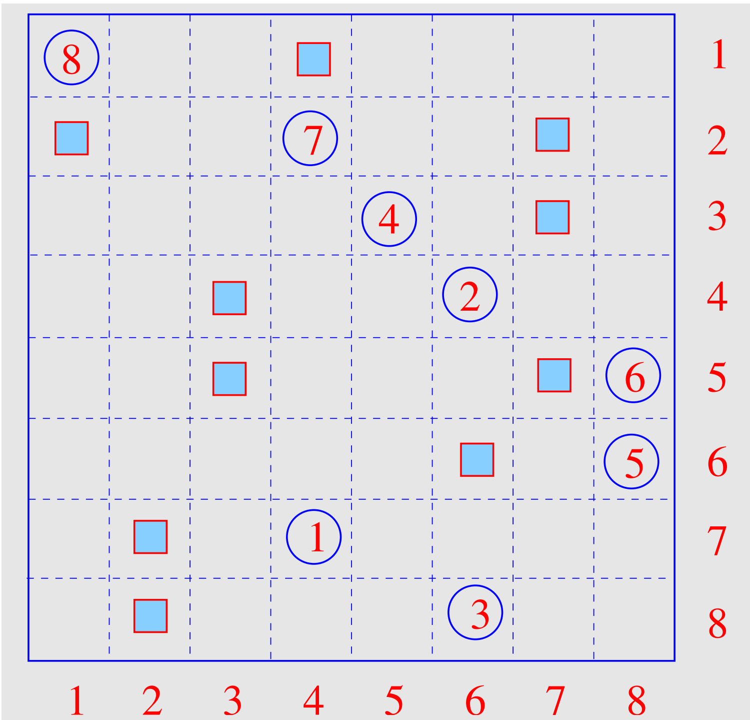
(2) **Matching:** scan candidate entries in order given by preselection and accept them into the $\mathcal{M}$ set, or reject them.

(3) **Complete the matching set:** into a complete pair of permutations (greedy algorithm)

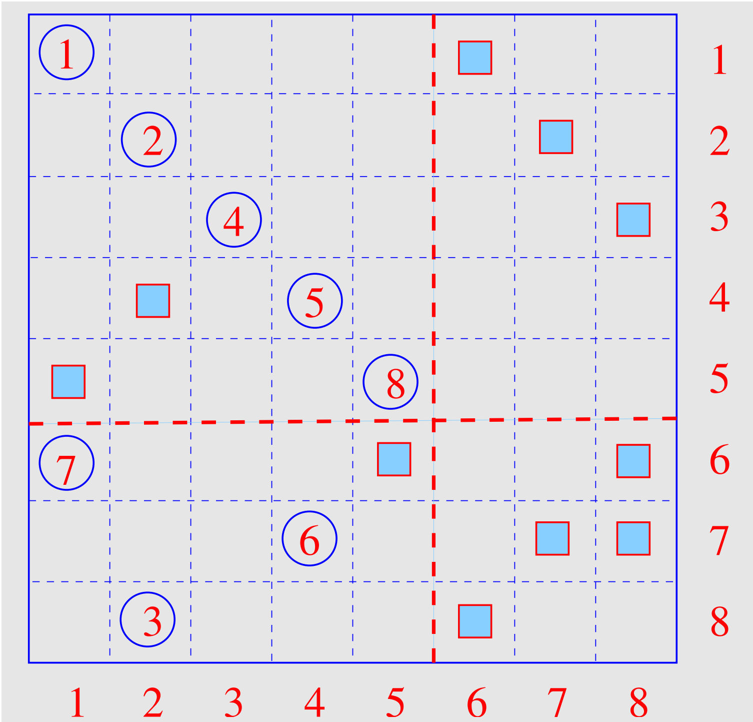
- Let $j(i) = \text{argmax}_j |a_{ij}|$.
- Use the ratio $\gamma_i = \frac{|a_{i,j(i)}|}{\|a_{i,:}\|_1}$ as a measure of diag. domin. of row i

Matching: Greedy algorithm

- Simple algorithm: scan pairs (i_k, j_k) in the given order.
- If i_k and j_k not already assigned, assign them to $\mathcal{M}$.



Matrix after preselection



Matrix after Matching perm.

- **Many heuristics explored – see in particular, recent work with S. MacLachlan '06.**
- **Main advantage over MC64: inexpensive and more dynamic procedure.**

MATLAB DEMO

’REAL’ TESTS

Numerical illustration

Matrix	order	nonzeros	Application (Origin)
barrier2-9	115,625	3,897,557	Device simul. (Schenk)
matrix_9	103,430	2,121,550	Device simul. (Schenk)
mat-n_3*	125,329	2,678,750	Device simul. (Schenk)
ohne2	181,343	11,063,545	Device simul. (Schenk)
para-4	153,226	5,326,228	Device simul. (Schenk)
cir2a	482,969	3,912,413	circuit simul.
scircuit	170998	958936	circuit simul. (Hamm)
circuit_4	80209	307604	Circuit simul. (Bomhof)
wang3.rua	26064	177168	Device simul. (Wang)
wang4.rua	26068	177196	Device simul. (Wang)

* mat-n_3* = matrix-new_3

Parameters

		Drop tolerance				$Fill_{max}$			
$nlev_{max}$	tol_{DD}	LU-B	GW	S	LU-S	LU-B	GW	S	LU-S
40	0.1	0.01	0.01	0.01	1.e-05	3	3	3	20

Matrix	Fill Factor	Set-up Time	GMRES(60)		GMRES(100)	
			Its.	Time	Its.	Time
barr2-9	0.62	4.01e+00	113	3.29e+01	93	3.02e+01
mat-n_3	0.89	7.53e+00	40	1.02e+01	40	1.00e+01
matrix_9	1.77	5.53e+00	160	4.94e+01	82	2.70e+01
ohne2	0.62	4.34e+01	99	6.35e+01	80	5.43e+01
para-4	0.62	5.70e+00	49	1.94e+01	49	1.93e+01
wang3	2.33	8.90e-01	45	2.09e+00	45	1.95e+00
wang4	1.86	5.10e-01	31	1.25e+00	31	1.20e+00
scircuit	0.90	1.86e+00	Fail	7.08e+01	Fail	8.80e+01
circuit_4	0.75	1.60e+00	199	1.69e+01	96	1.07e+01
circ2a	0.76	2.19e+02	18	1.08e+01	18	1.03e+01

Results for the 10 systems - ARMS-ddPQ + GMRES(60) & GMRES(100)

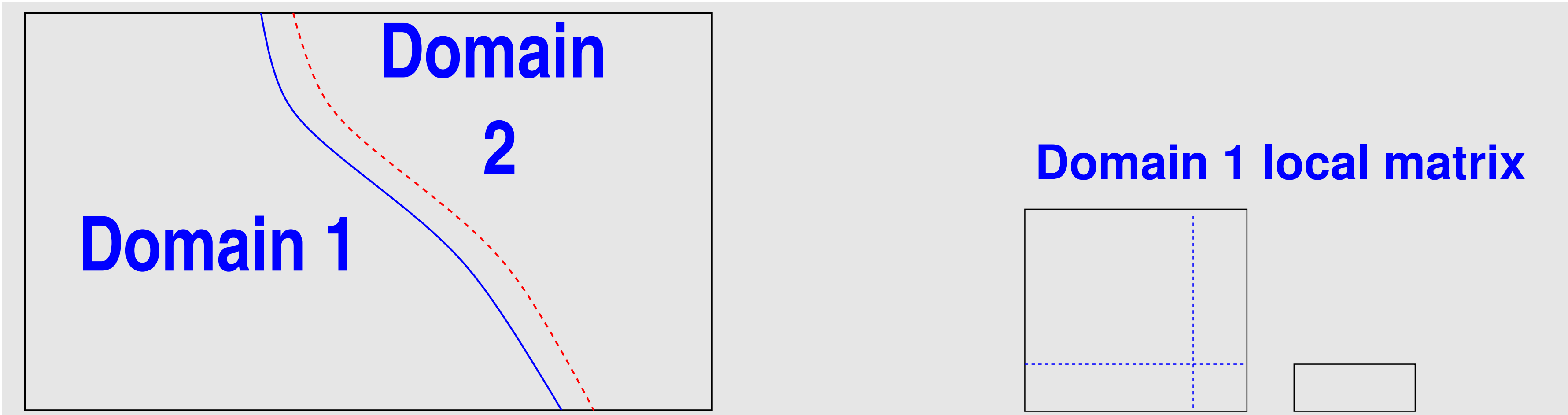
	Fill	Set-up	GMRES(60)		GMRES(100)	
	Factor	Time	Its.	Time	Its.	Time
Same param's	0.89	1.81	400	9.13e+01	297	8.79e+01
Droptol = .001	1.00	1.89	98	2.23e+01	82	2.27e+01

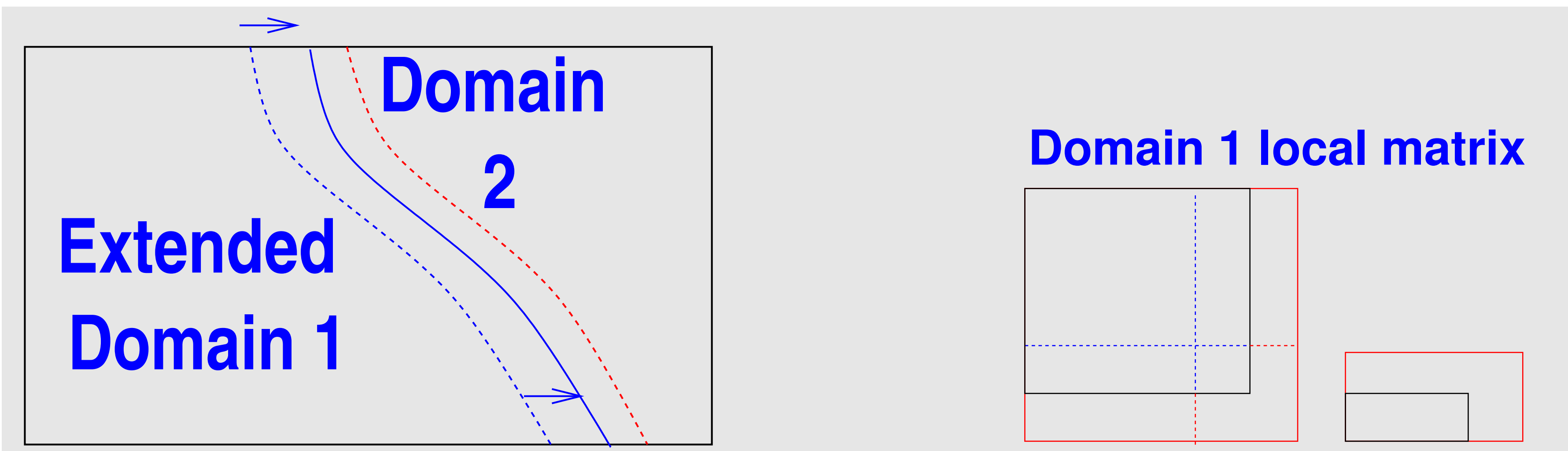
Solution of the system `sciircuit` – no scaling + two different sets of parameters.

Parallel implementation

- Preliminary work – with Zhongze Li
- Ideally would use hypergraph partitioning [in the plans]
- We used only a local version of ddPQ
- Schur complement version not yet available
- In words: Construct the local matrix, extend it with overlapping data and use ddPQ ordering on it.
- Can be used with Standard Schwarz procedures – or with restrictive version [RAS]

Restricted Additive Schwarz Preconditioner(RAS)





- RAS + ddPQ uses arms-ddPQ on extended matrix - for each domain.
- ddPQ Improves robustness enormously in spite of simple (local) implementation.
- Test with problem from MHD problem.

Example: a system from MHD simulation example

- Source of problem: Coupling of Maxwell equations with Navier-Stokes.

- Matrices arises from solving Maxwell's equation:

$$
\begin{aligned}
\frac{\partial \mathrm{B}}{\partial t} - \nabla \times (\mathrm{u} \times \mathrm{B}) - \frac{1}{Re_m} \nabla \times (\nabla \times \mathrm{B}) + \nabla \mathrm{q} &= 0 \\
\nabla \cdot \mathrm{B} &= 0 \, ,
\end{aligned}
$$

- See [Ben-Salah, Soulaimani, Habashi, Fortin, IJNMF 1999]

- Cylindrical domain, tetrahedra used.

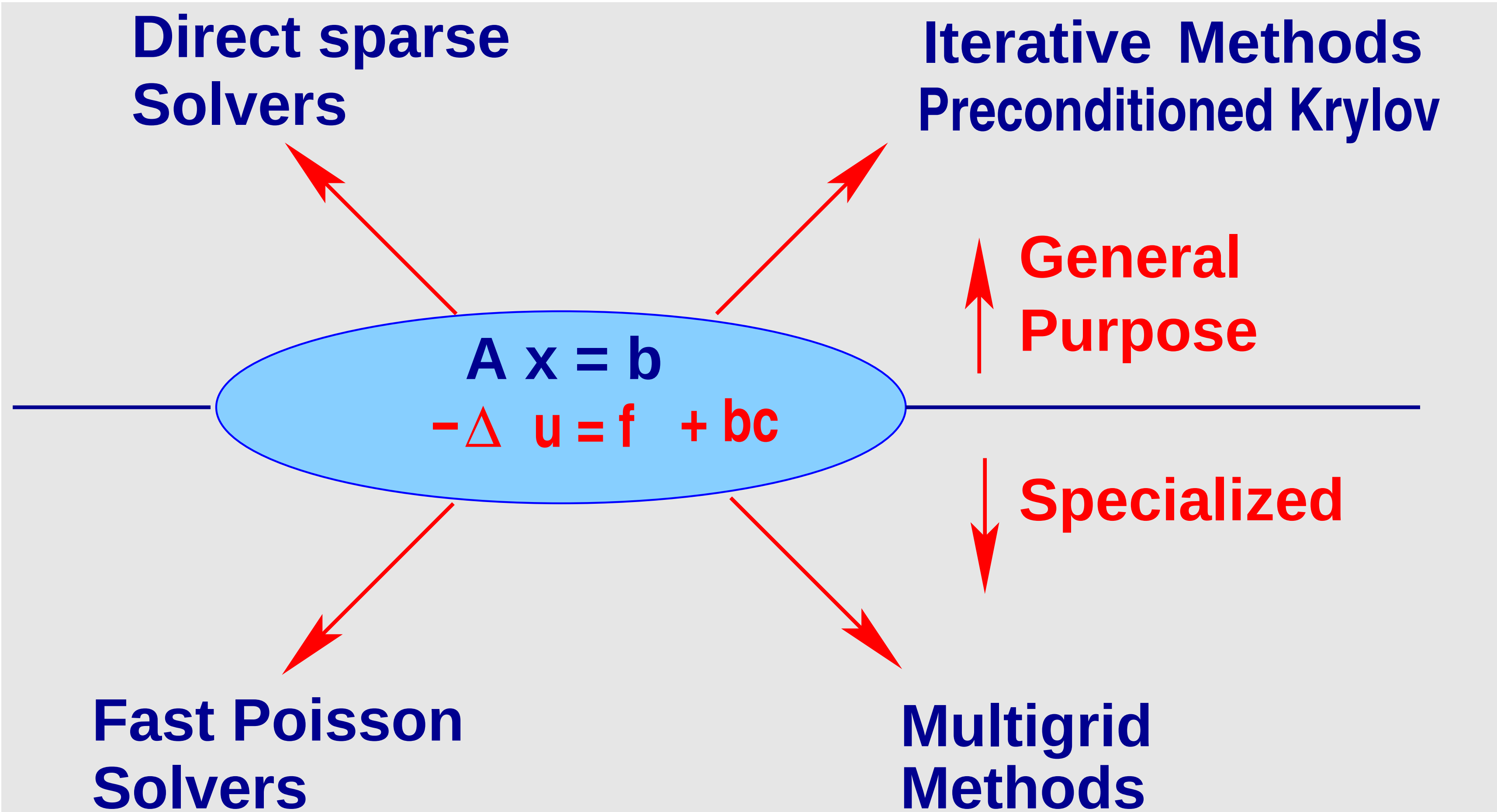
- Not an easy problem for iterative methods.

	RAS+ILUT				RAS+ddPQ		
np	its	t_{set}	t_{it}	np	its	t_{set}	t_{it}
1	107	236.58	320.74	1	60	204.06	187.05
2	118	136.28	232.78	2	104	108.45	162.34
4	354	72.66	326.03	4	109	60.24	86.25
8	2640	40.06	1303.16	8	119	41.56	52.11
16	3994	21.87	1029.88	16	418	22.84	97.88
32	$>$ 10,000	–	–	32	537	12.34	65.77

- Simple Schwarz (RAS) : very poor performance
- severe deterioration of performance with higher np

Conclusion

- ARMS-C works well as a “general-purpose” solver.
- Though far from being a 100% robust iterative solver ...
- .. It is efficient [memory and computatitional costs]
- ... Easier to parallelize than MC64
- Recent work on generalizing nonsymmetric permutations to symmetric matrices [Duff-Pralet, 2006].



What is missing from this picture?

- 1. Intermediate methods which lie in between general purpose and specialized – exploit some information from origin of the problem.

- 2. Considerations related to parallelism. Development of ‘robust’ solvers remains limited to serial algorithms in general.

- Problem: parallel implementations of iterative methods are less effective than their serial counterparts.

Software:

- ARMS-C [C-code] - available from ITSOL package..

    `http://www.cs.umn.edu/~saad/software`

- More comprehensive package: ILUPACK – developed mainly by Matthias Bollhoefer and his team

    `http://www.tu-berlin.de/ilupack/.`