Preconditioning techniques for highly indefinite systems

Yousef Saad

Department of Computer Science and Engineering

Workshop on Solution Methodologies for Scattering Problems

Pau, Dec. 11-14, 2007

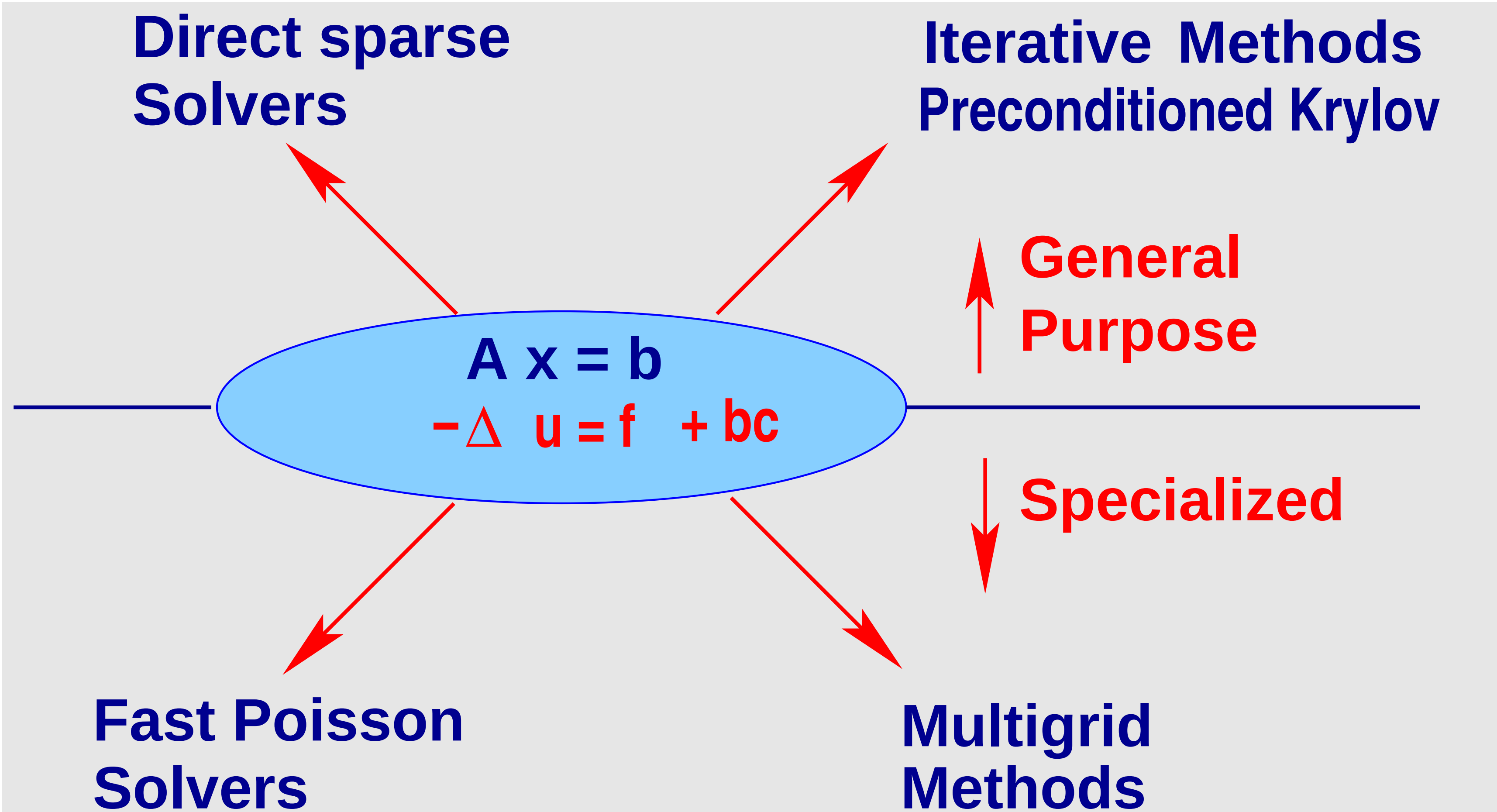
Introduction: Linear System Solvers

- **Problem considered: Linear systems**

$$\boxed{Ax = b}$$

- **Can view the problem from somewhat different angles:**

  - **Discretized problem coming from a PDE**
  - **An algebraic system of equations [ignore origin]**
  - **Sometimes a system of equations where A is not explicitly available**


Direct sparse Solvers
Iterative Methods
Preconditioned Krylov
General Purpose
A x = b
−Δ u = f + bc
Specialized
Fast Poisson Solvers
Multigrid Methods

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

Background on preconditioned Krylov methods

- **Krylov subspace methods: projection methods which extract approximate solutions to $Ax = b$ from the subspace (initial guess $x_0 = 0$)**

$$\text{span}\{b, Ab, \cdots, A^{m-1}b\}$$

- **Essentially: solution is approximated in some optimal way by $p_{m-1}(A)b$, where p_{m-1} = polyn. of degree $m-1$**

- **For $x_0 \neq 0$, approximation is in affine space**

$$x_0 + \{r_0, Ar_0, \cdots, A^{m-1}r_0\}$$

Preconditioning

Use Krylov subspace method on a modified system such as

$$\boxed{M^{-1}Ax = M^{-1}b.}$$

- **Matrix $M^{-1}A$ need not be formed explicitly; only need to solve $Mw = v$ whenever needed.**
- **Requirement: $M^{-1}v$ inexpensive to evaluate $\forall\ v$**

Three different forms:

Left preconditioning	$M^{-1}Ax = M^{-1}b$	
Right preconditioning	$A\,\underbrace{M^{-1}y}_{x} = b$	$x = M^{-1}y$
Split preconditioning	$M_L^{-1}A\,\underbrace{M_R^{-1}y}_{x} = M_L^{-1}b$	$x = M_R^{-1}y$

Standard preconditioners

- **Most common 'general purpose' methods used:**

1. ILU(0) or ILU(k)

2. ILUT

3. New trend: Multilevel ILU, e.g., ARMS .

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 –

ILU(0) and IC(0) preconditioners

- **Notation:** $NZ(X) = \{(i,j) \mid X_{i,j} \neq 0\}$

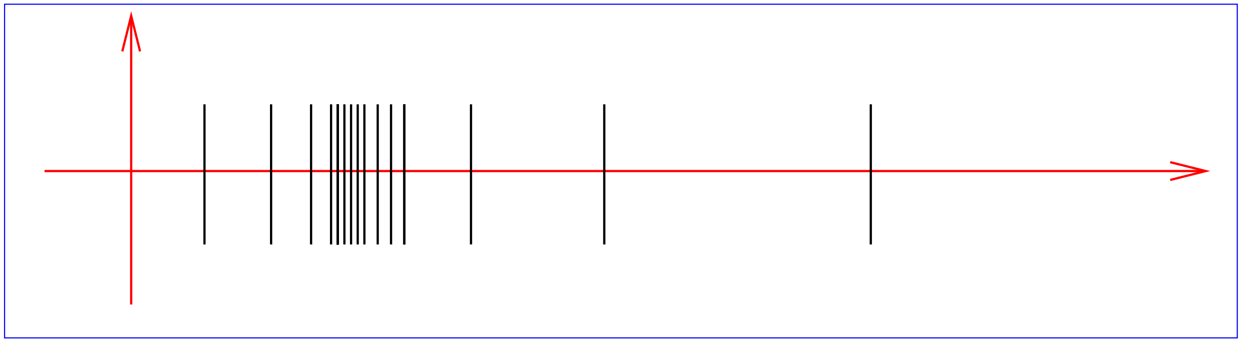
Formal definition of ILU(0)

(Does not define $ILU(0)$ in a unique way)

$$A = LU + R$$
$$NZ(L) \cup NZ(U) = NZ(A)$$
$r_{ij} = 0$ **for** $(i,j) \in NZ(A)$

<u>Constructive definition:</u> Compute the LU factorization of A but drop any fill-in in L and U outside of Struct(A).

- **Typical eigenvalue distribution for preconditoned matrix :**



ILU(p) factorization – level of fill

- **Higher accuracy incomplete Cholesky: for regularly structured problems, IC(p) allows p additional diagonals in L.**

- **Can be generalized to irregular sparse matrices using the notion of level of fill-in [Watts III, 1979]**

- **Initially** $Lev_{ij} = \begin{cases} 0 & \text{for } a_{ij} \neq 0 \\ \infty & \text{for } a_{ij} == 0 \end{cases}$
- **At a given step i of Gaussian elimination:**

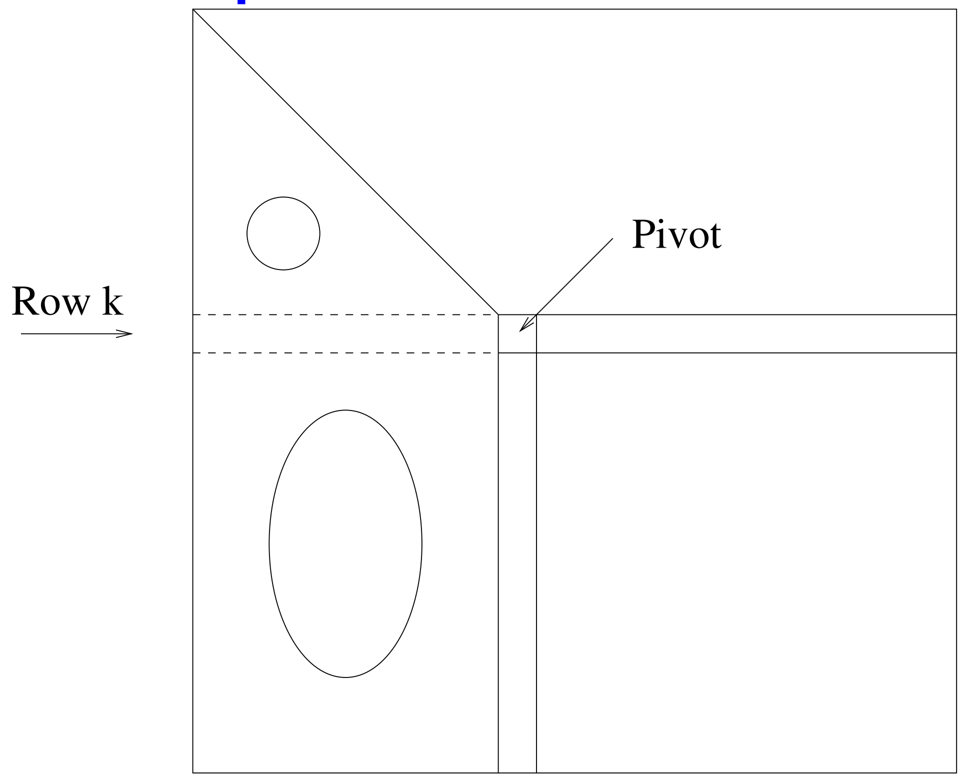
$$\boxed{Lev_{kj} = \min\{Lev_{kj}; Lev_{ki} + Lev_{ij} + 1\}}$$

- **ILU(p) Strategy = drop anything with level of fill-in exceeding p.**

LU - standard (KIJ) version

- At step k :

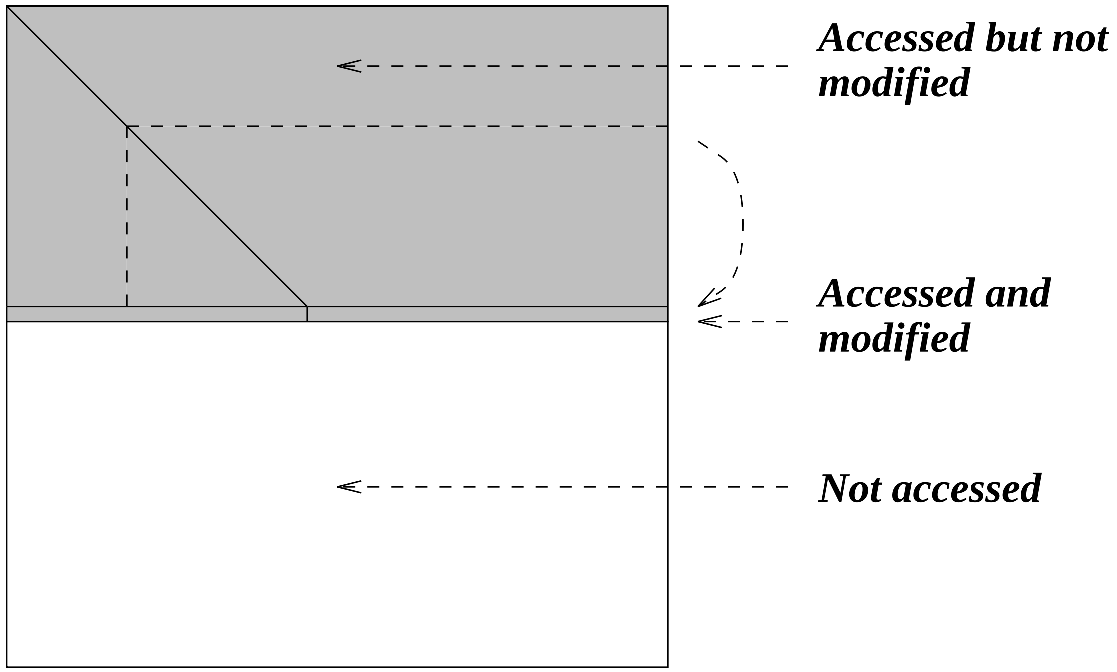
```
for k=1:n-1
    for i = k+1:n
        for j = k+1:n
            a(i,j) = ...
```



ILUT - based on the IKJ version of GE

► **At step i :**

```
for i=1:n-1
   for k = 1:i-1
      for j = i+1:n
         a(i,j) = ...
```



ILU with threshold: ILUT(k, ϵ)

- **Do the i, k, j version of Gaussian Elimination (GE).**
- **Discard any pivot or fill-in whose value is below $\epsilon \|row_i(A)\|$.**
- **Once the i-th row of $L + U$, (L-part + U-part) is computed retain only the k largest elements in both parts.**

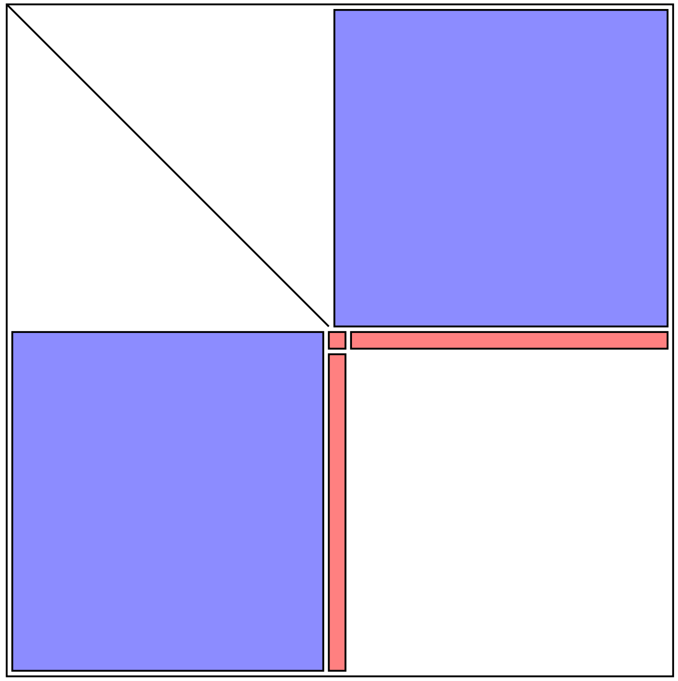
► **Advantages: controlled fill-in. Smaller memory overhead.**

► **Easy to implement – much more so than preconditioners derived from direct solvers.**

► **Can be quite inexpensive for accurate factorizations (high fill) – Solution : Crout versions of ILU**

Crout-based ILUT (ILUTC)

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:

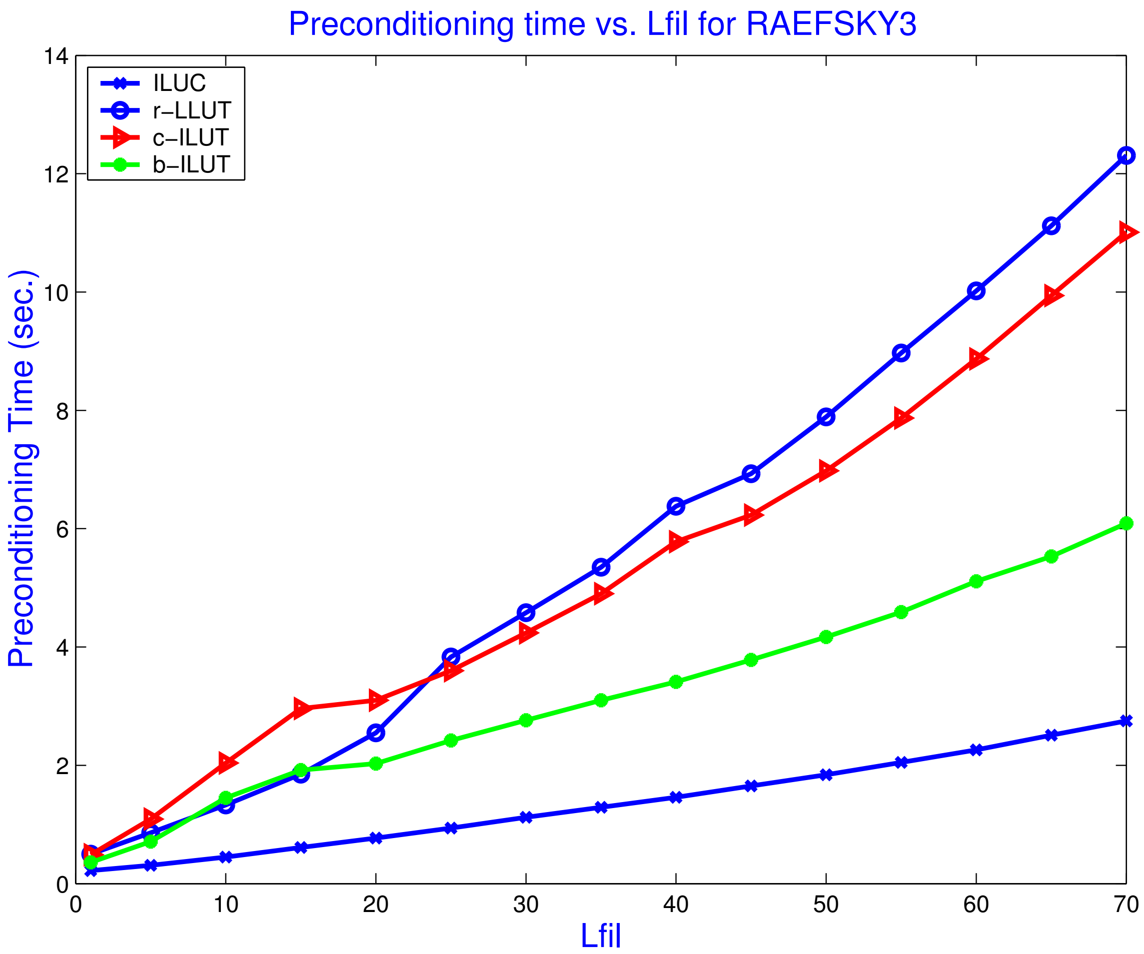
[1] M. Jones and P. Plassman. An improved incomplete Choleski factorization. *ACM Transactions on Mathematical Software*, 21:5–17, 1995.

[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 ILUT

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



NONSYMMETRIC REORDERINGS

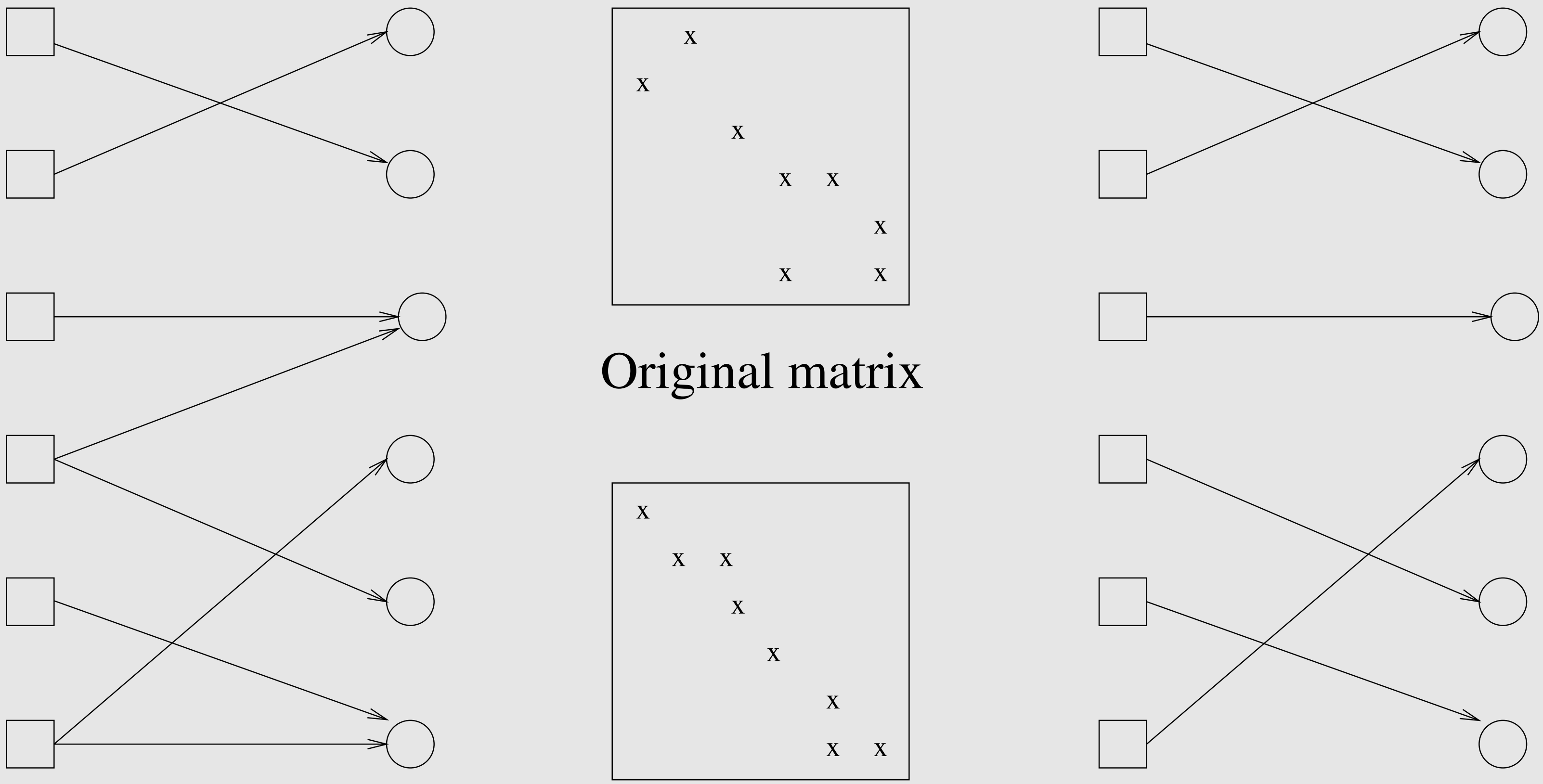
Enhancing robustness: One-sided permutations

► 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.



Bipartite representation   After reordering   Maximum transversal

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

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

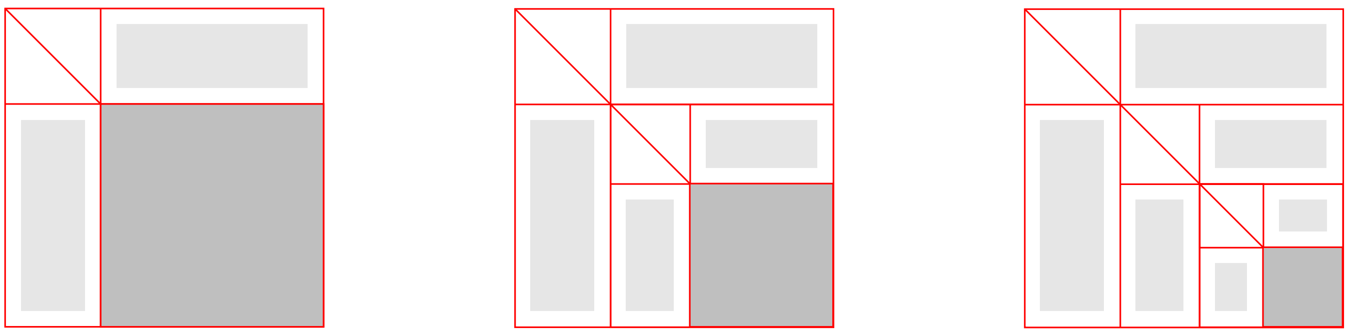
where B is a diagonal matrix.

- 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.

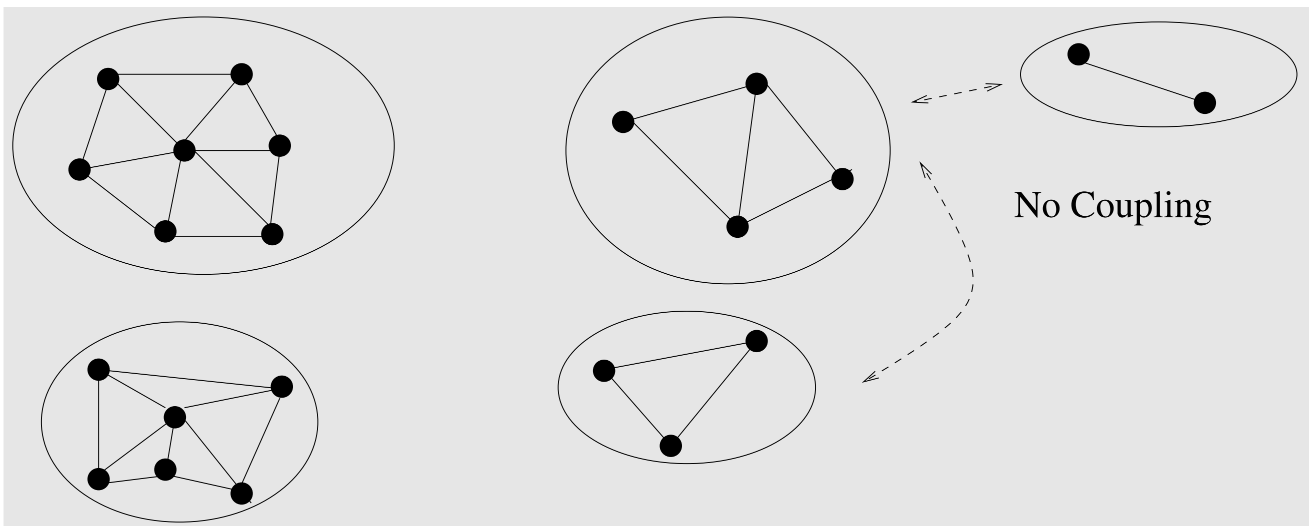


- See work by [Botta-Wubbs '96, '97, YS'94, '96, Leuze '89,..]

Group Independent Sets / Aggregates

Main goal: **generalize independent sets to improve robustness**

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

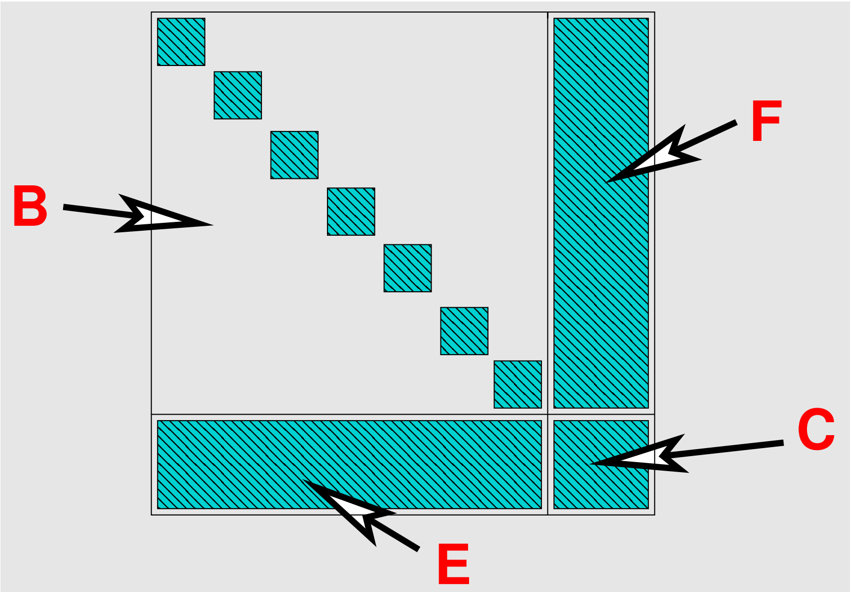


► **Label nodes of independent sets first**

Algebraic Recursive Multilevel Solver (ARMS)

- **Typical shape of reordered matrix:**

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



- **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**

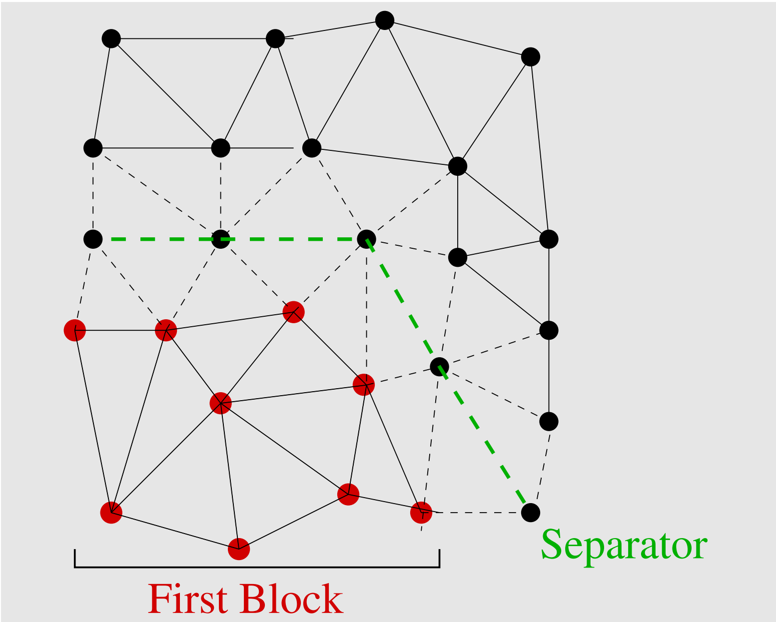
Algebraic Recursive Multilevel Solver (ARMS)

Level l Factorization:

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

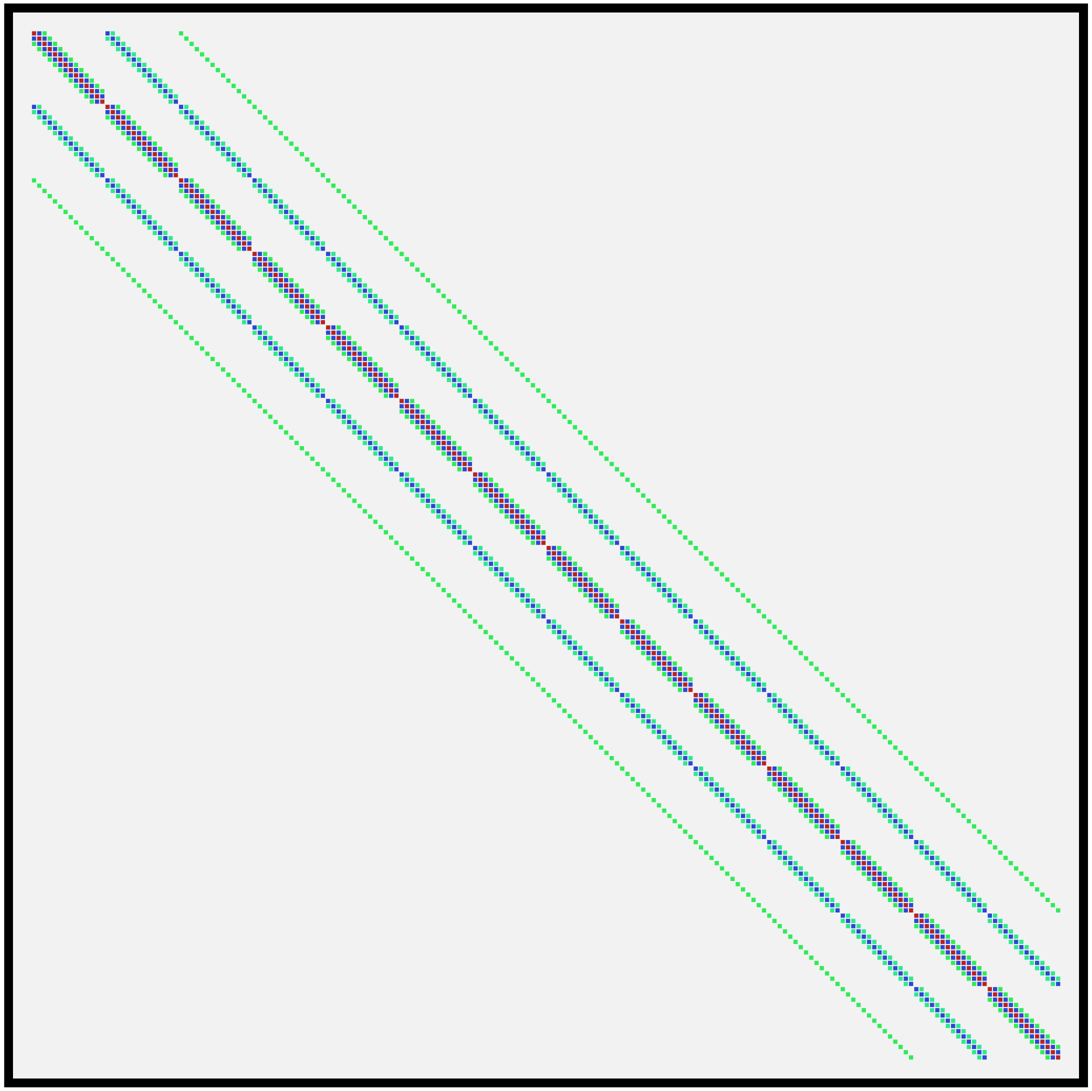
- **L-solve $\sim$ restriction; U-solve $\sim$ prolongation.**
- **Perform above block factorization recursively on A_{l+1}**
- **Blocks in B_l treated as sparse. Can be large or small.**
- **Algorithm is fully recursive**
- **Stability criterion in block independent sets algorithm**

Group Independent Set reordering

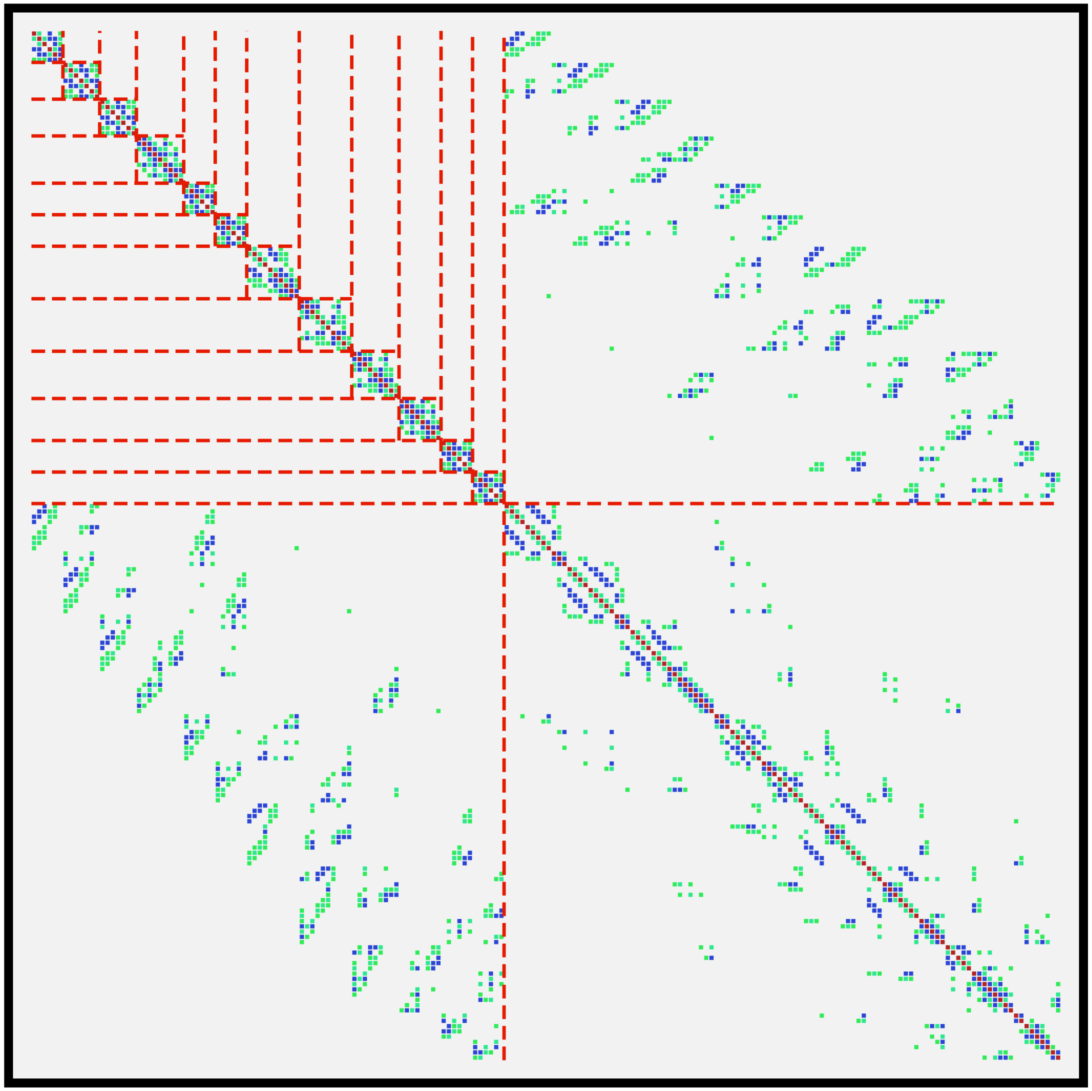


<u>Simple strategy:</u> **Level taversal until there are enough points to form a block. Reverse ordering. Start new block from non-visited node. Continue until all points are visited. Add criterion for rejecting "not sufficiently diagonally dominant rows."**

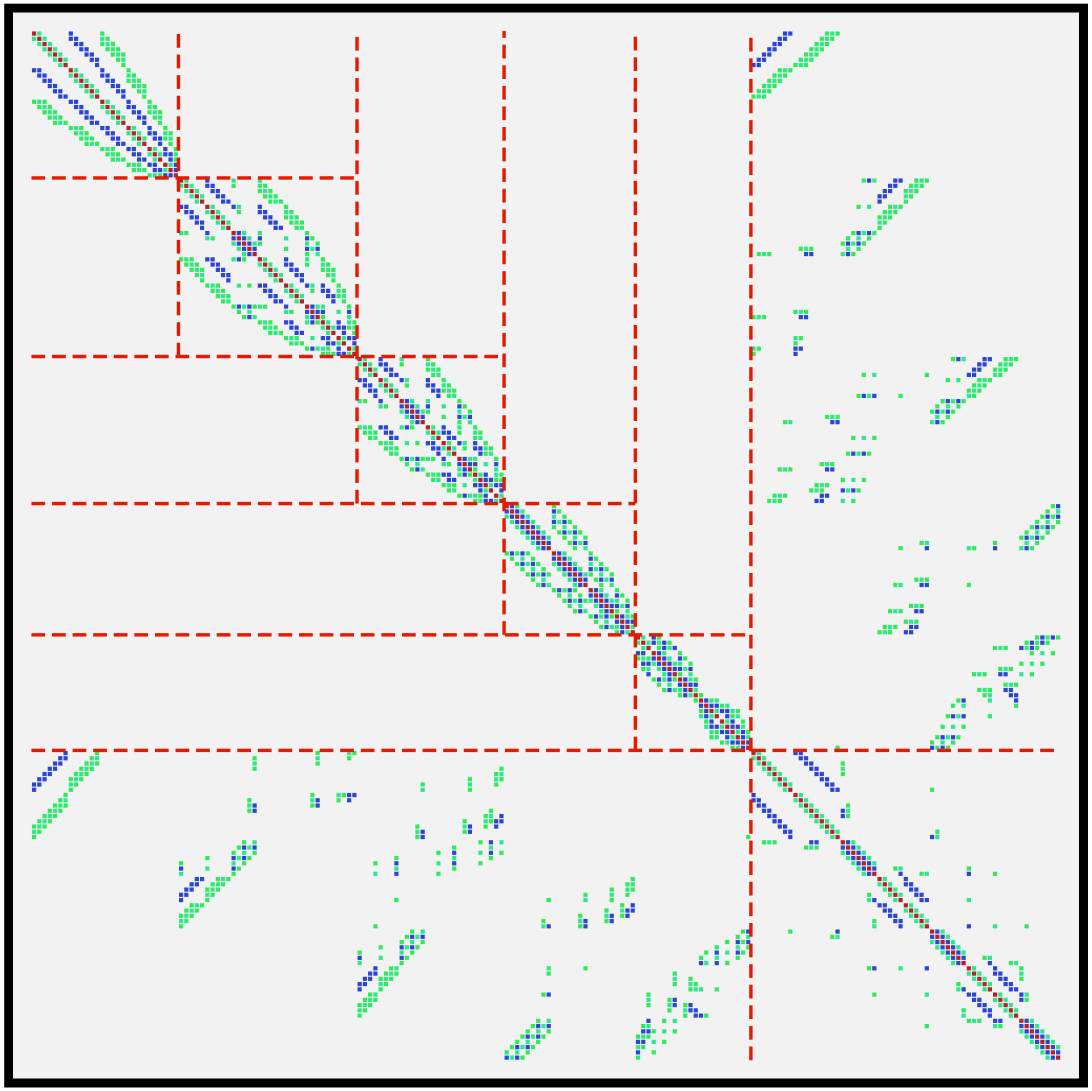
Original matrix



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 $\rightarrow$ 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 $n_M = n \rightarrow$ (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.

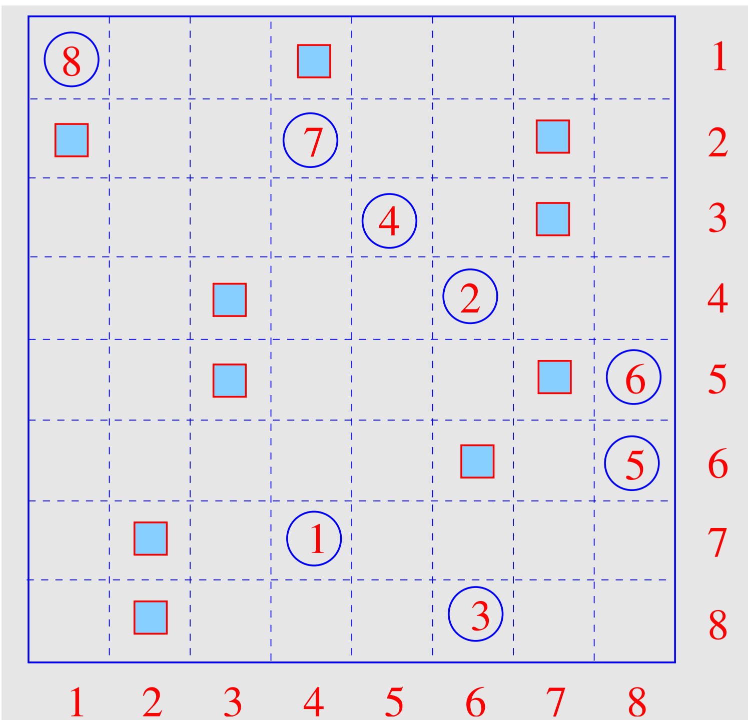
(2) Matching: scan candidate entries in order given by preselection and accept them into the $\mathcal{M}$ set, or reject them.

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

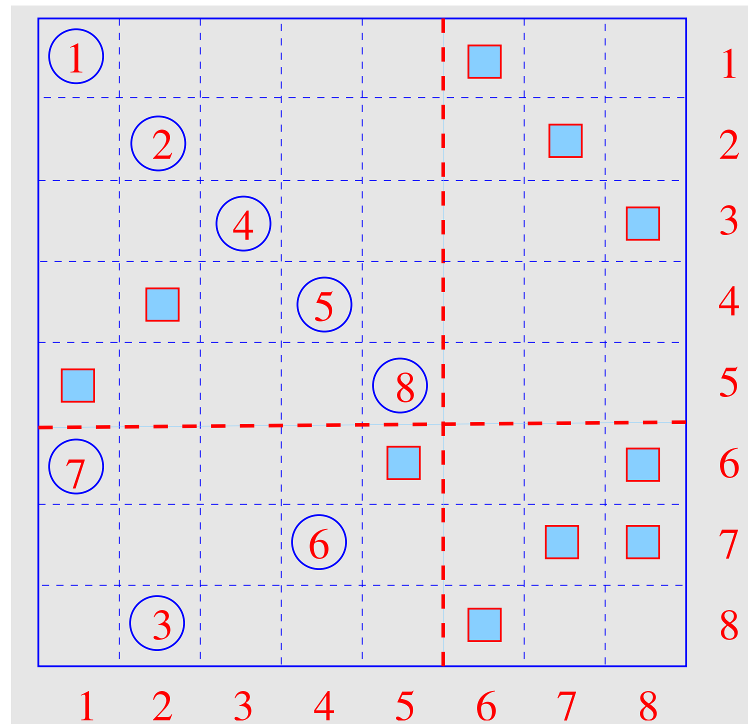
- **Let $j(i) = \text{argmax}_j |a_{ij}|$.**
- **Use the ratio $\gamma_i = \frac{|a_{i,j(i)}|}{\|a_{i,:}\|_1}$ as a measure of diag. domin. of row i**

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) Its.	GMRES(60) Time	GMRES(100) Its.	GMRES(100) 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 `scircuit` – no scaling + two different sets of parameters.

Application to the Helmholtz equation

- **Collaboration with Riyad Kechroud, Azzeddine Soulaimani (ETS, Montreal), and Shiv Gowda: [Math. Comput. Simul., vol. 65., pp 303–321 (2004)]**

- **Problem is set in the open domain Ω_e of $\mathbb{R}^d$**

$$\begin{cases} \Delta u + k^2 u = f & \text{in } \Omega \\ u = -u_{inc} & \text{on } \Gamma \\ or\ \frac{\partial u}{\partial n} = -\frac{\partial u_{inc}}{\partial n} & \text{on } \Gamma \\ \\ lim_{r\to\infty} r^{(d-1)/2} \left(\frac{\partial u}{\partial \vec{n}} - iku\right) = 0 & \text{Sommerfeld condition} \end{cases}$$

where: u the wave diffracted by Γ, f = source function = zero outside domain

- Issue: non-reflective boundary conditions when making the domain finite Γ_{art} = artificial boundary – Many techniques available.
- For high frequencies, linear systems beccome very ‘indefinite’ – [eigenvalues on both sides of the imaginary axis]
- Not very good for iterative methods

Application to the Helmholtz equation

Problem 1:

$$\begin{cases} \Delta u + k^2 u = 0 \text{ in } \Omega_e \\ \frac{\partial u}{\partial \vec{n}} + iku = g \text{ in } \Gamma_{art} \end{cases}$$

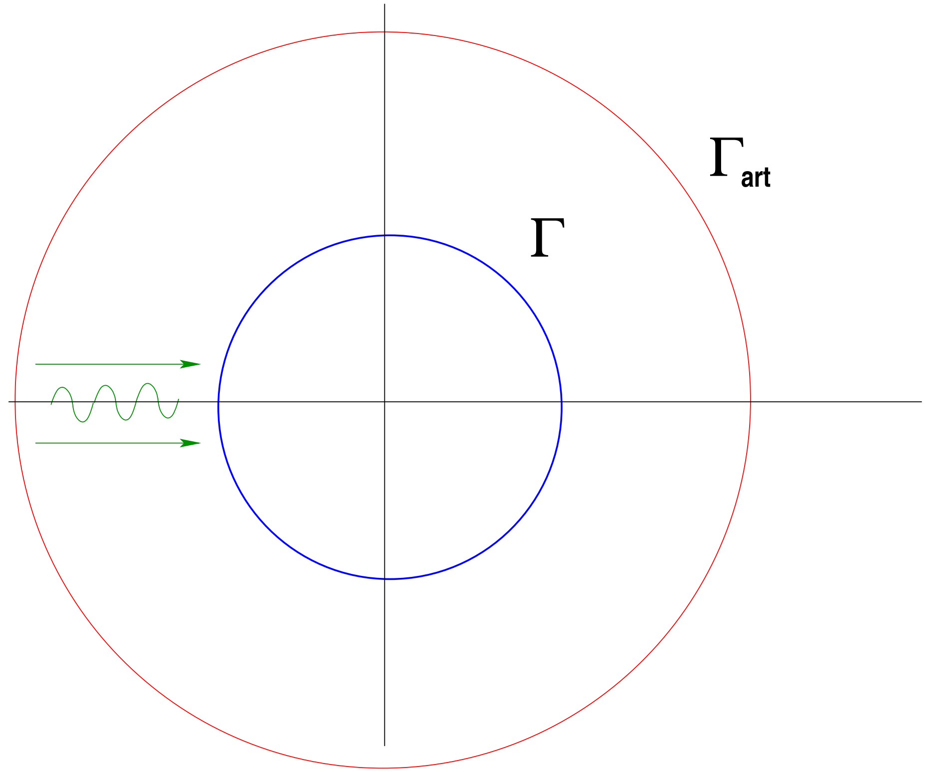
- **Domain:** $\Omega = (0,1) \times (0,1)$

- **Function g selected so that exact solution is** $u(x,y) = \exp[ik\cos(\theta)x + k\sin(\theta)y]$**.**

- **Structured meshes used for the discretization**

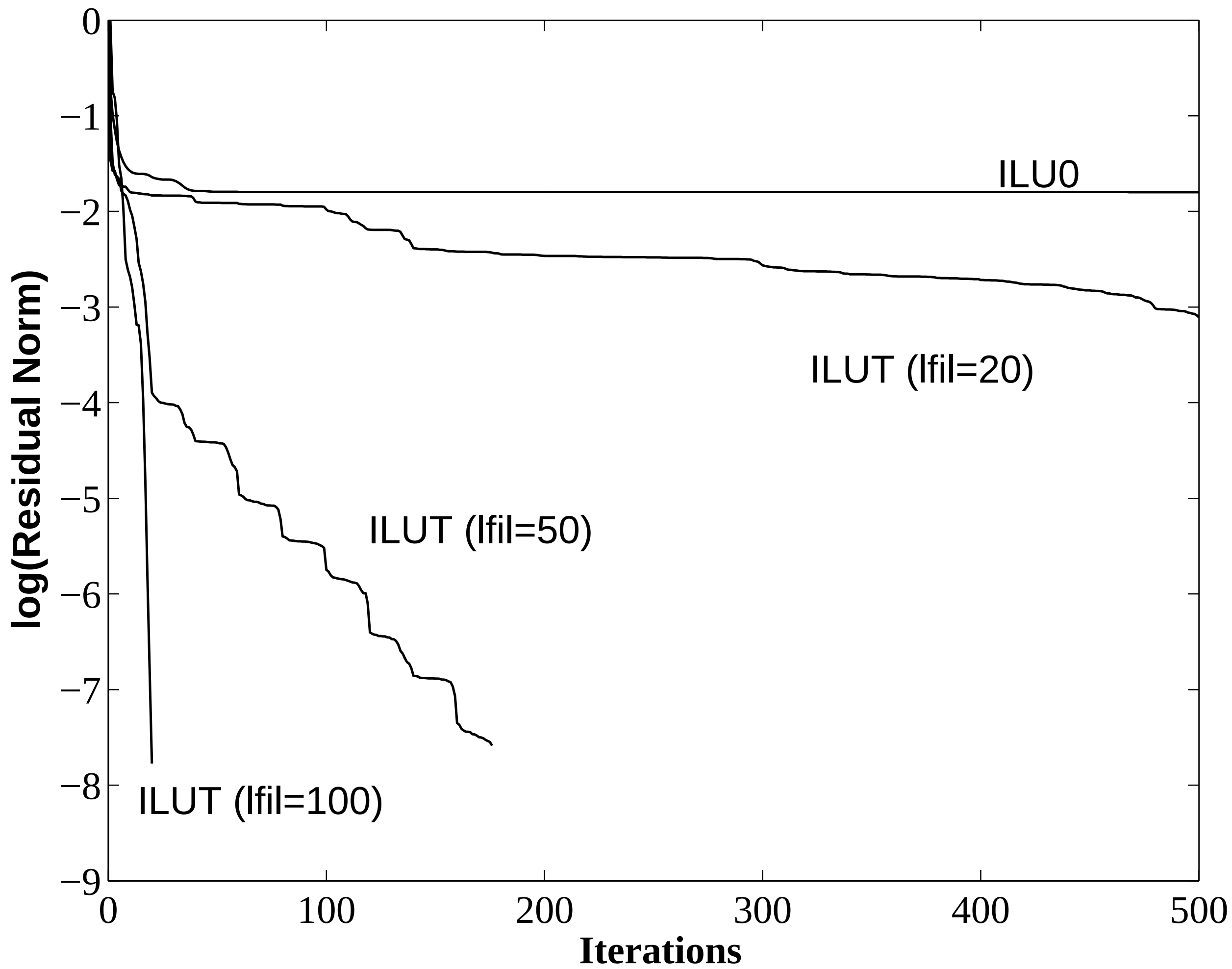
Problem 2. **The soft obstacle == disk of radius $r_0 = 0.5m$. Incident plane wave with a wavelength λ; propagates along the x-axis. 2nd order Bayliss-Turkel boundary conditions used on Γ_{art}, located at a distance $2r_0$ from the obstacle. Discretization uses isoparametric elements with 4 nodes.**

- **The analytic solution is known**



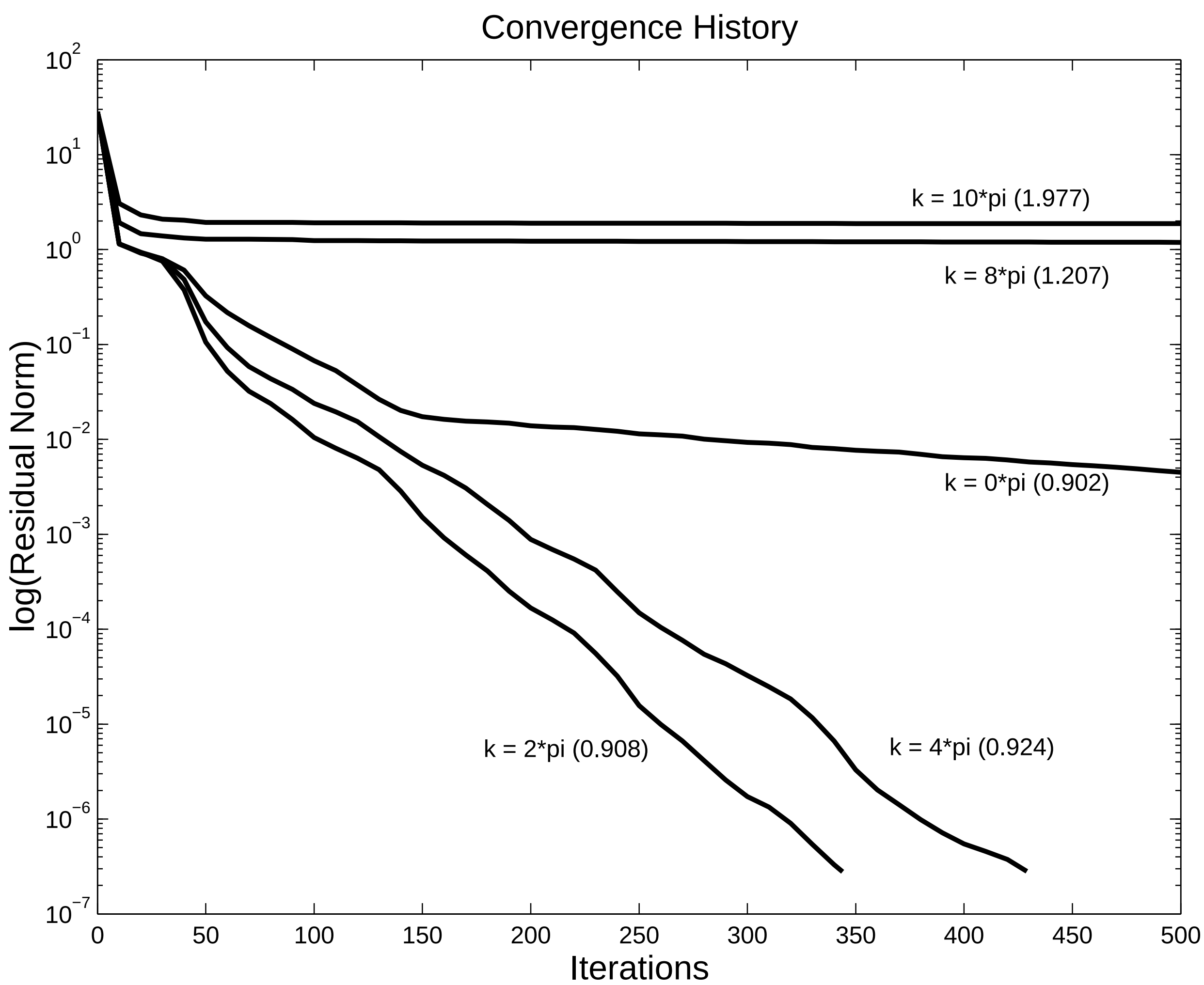
Impact of the dropping strategy in ILUT

Pb 1. Convergence of ILUT-GMRES for different values of $lfil$



Using a preconditioner from a lower wavenumber

- **Good strategy for high frequencies. Test with Problem 2 –**



Solution found – (Thanks: R. Kechroud)

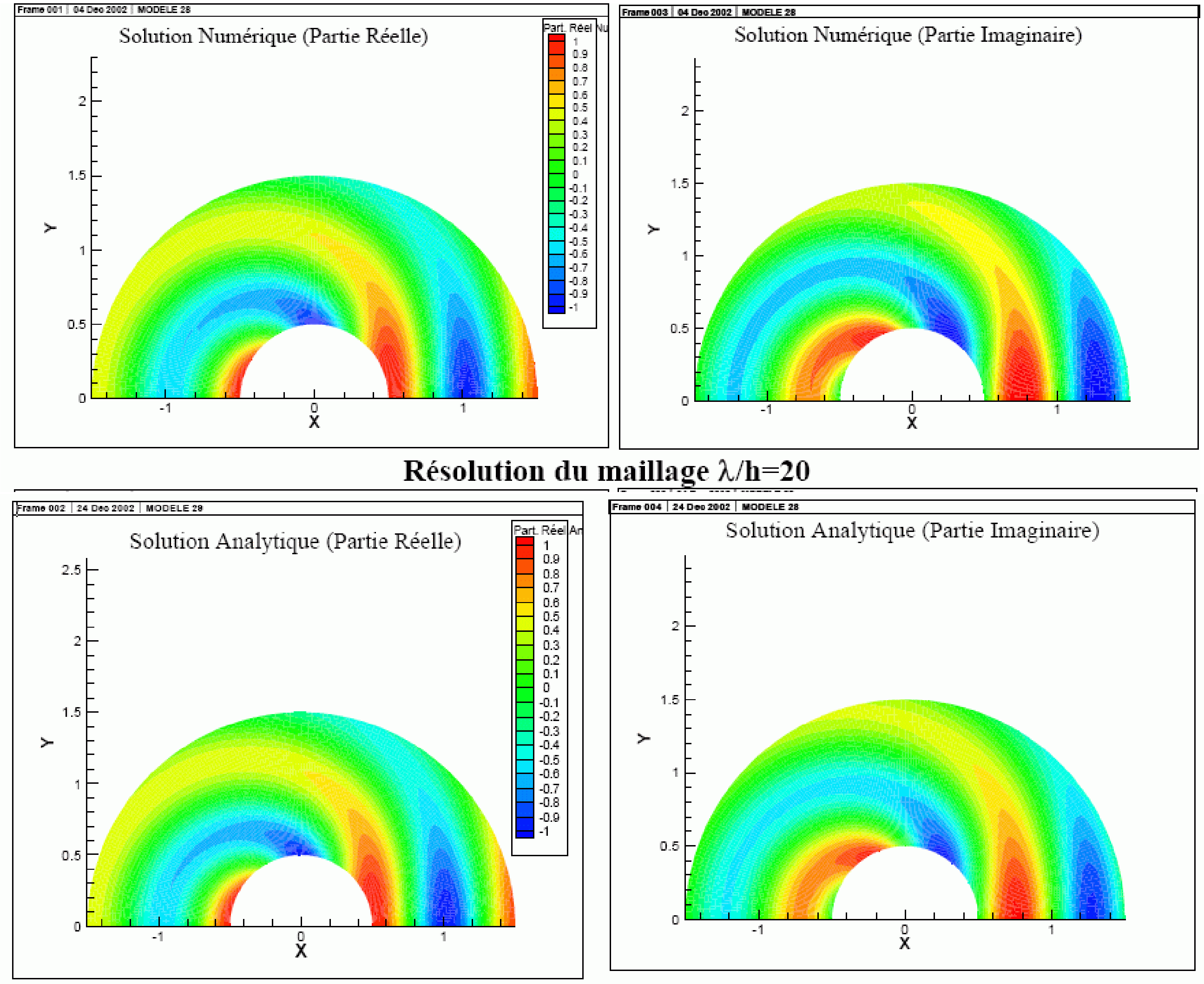


Figure 8 : Lignes de contour (solution analytique)

Recent comparisons

- Thanks: Daniel Osei-Kuffuor
- Setting: Problem 2. Mesh size fixed to $h = 1/80$. Problem size = $n = 29,160$, Number of nonzeroes $nnz = 260,280$
- For each preconditioner $lfil = 5 \times nnz/n$
- Wavenumber varied [until convergence fails]

ILUT with $droptol = 0.001$

k	$\frac{\lambda}{h}$	No. iters	Setup Time (s)	Iter. Time (s)	Fill Factor
2π	80	43	0.38	1.88	4.67
4π	40	87	0.91	4.45	6.97
**	**	**	**	**	**

ILUTP with $droptol = 0.001$

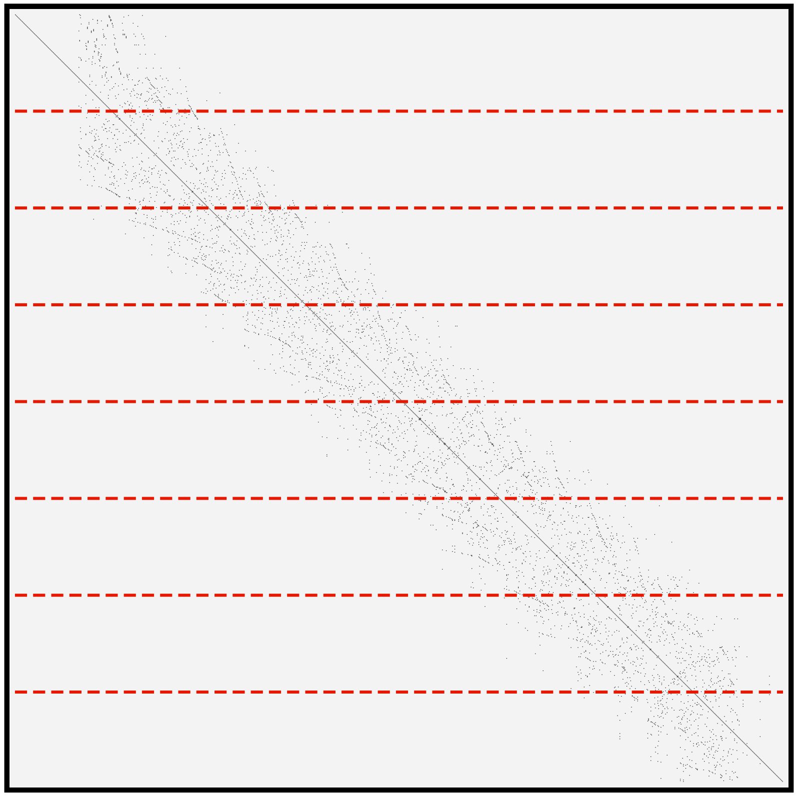
k	$\frac{\lambda}{h}$	No. iters	Setup Time (s)	Iter. Time (s)	Fill Factor
2π	80	41	0.57	1.92	5.73
4π	40	193	1.02	10.4	7.31
**	**	**	**	**	**

ARMS-ddPQ

k	$\frac{\lambda}{h}$	No. iters	Setup Time (s)	Iter. Time (s)	Fill Factor
2π	80	100	0.52	5.07	1.98
4π	40	148	0.57	7.58	2.08
8π	20	998	0.75	52.3	4.6
16π	10	190	4.70	1.41	8.7

PARALLEL KRYLOV METHODS

Distributed Sparse Systems: Simple illustration

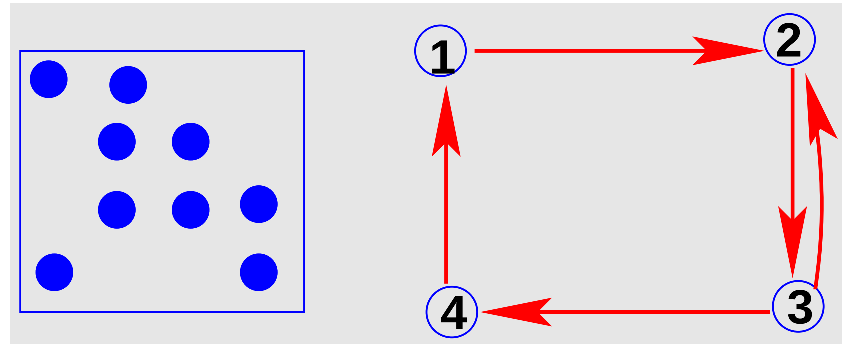


► **Naive partitioning of equations - does not work well in practice (performance)**

Best idea is to use the adjacency graph of A:

Vertices = $\{1, 2, \cdots, n\}$;

Edges: $i \rightarrow j$ iff $a_{ij} \neq 0$



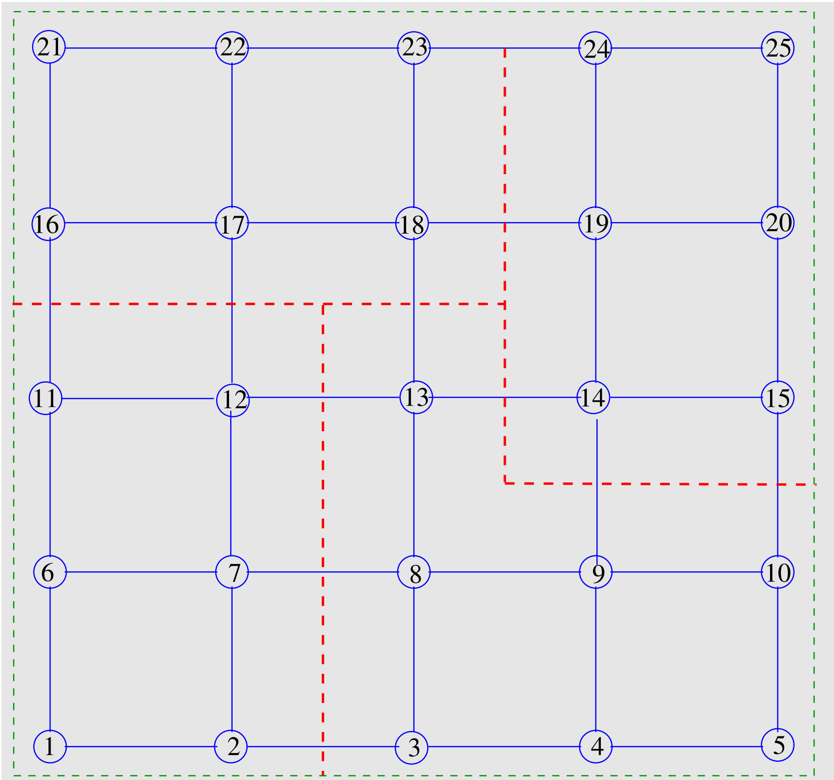
Graph partitioning problem:

- Want a partition of the vertices of the graph so that

(1) partitions have $\sim$ the same sizes

(2) interfaces are small in size

General Partitioning of a sparse linear system



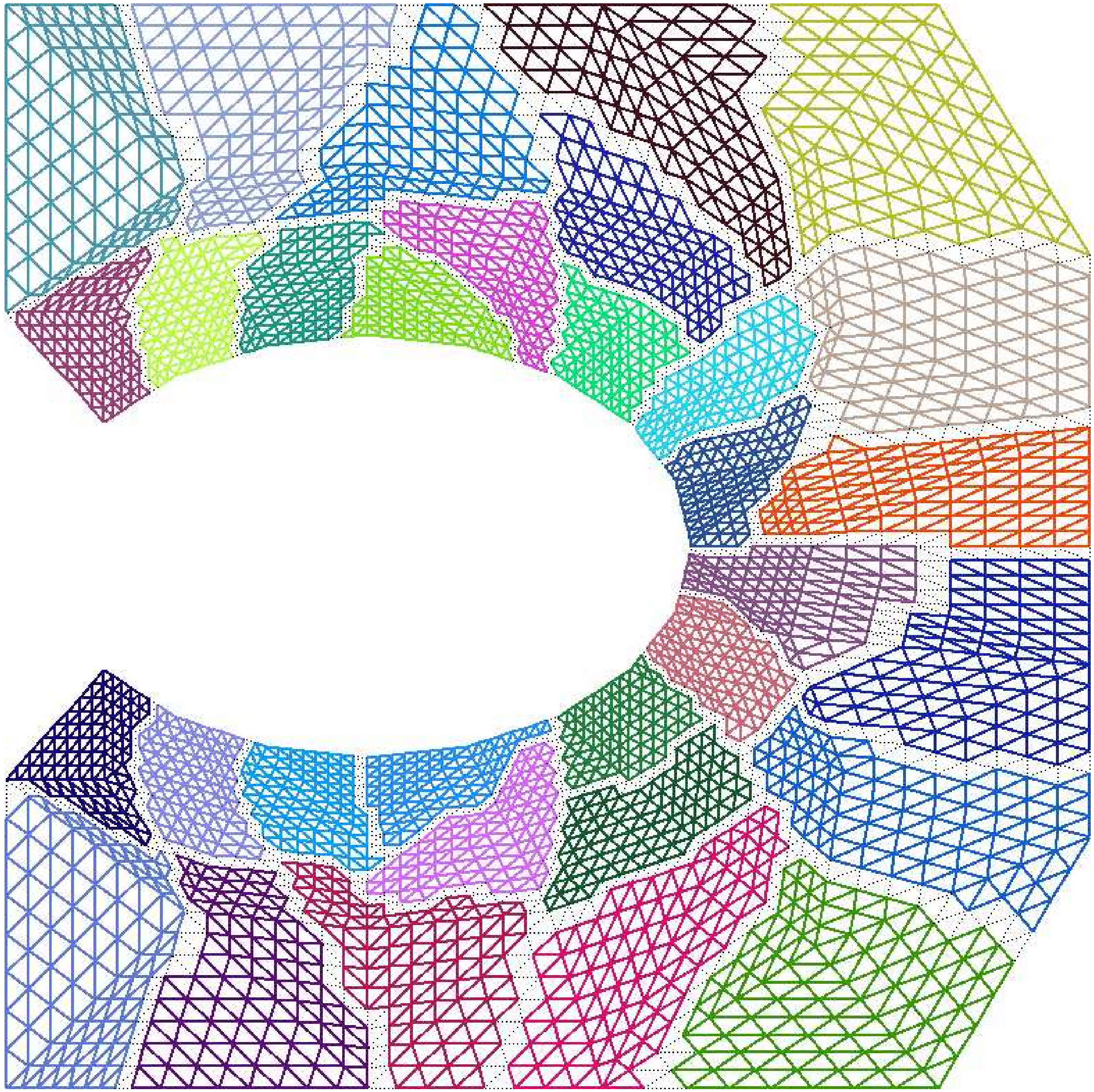
$S_1 = \{1, 2, 6, 7, 11, 12\}$: **This means equations and unknowns 1, 2, 3, 6, 7, 11, 12 are assigned to Domain 1.**

$S_2 = \{3, 4, 5, 8, 9, 10, 13\}$

$S_3 = \{16, 17, 18, 21, 22, 23\}$

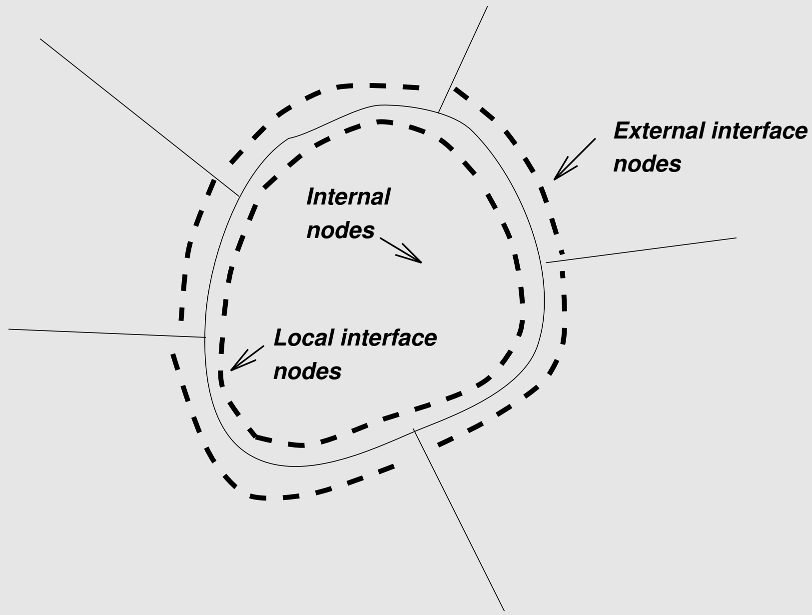
$S_4 = \{14, 15, 19, 20, 24, 25\}$

- **Partitioners : Metis, Chaco, Scotch, ..**
- **More recent: Zoltan, H-Metis, PaToH**

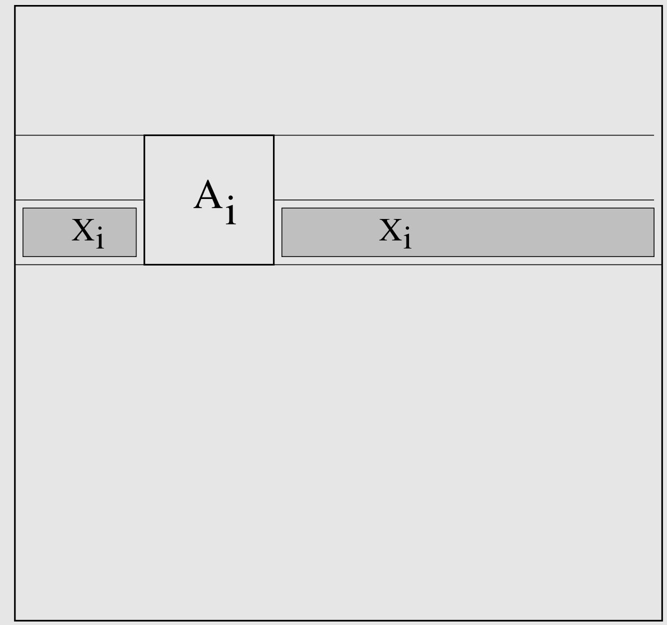


- Standard dual objective: “minimize” communication + “balance” partition sizes
- Recent trend: use of hypergraphs [PaToh, Hmetis,...]
- Hypergraphs are very general.. Ideas borrowed from VLSI work
- Main motivation: to better represent communication volumes when partitioning a graph. Standard models face many limitations
- Hypergraphs can better express complex graph partitioning problems and provide better solutions. Example: completely nonsymmetric patterns.

A distributed sparse system



Graph representation



Matrix representation

- **In each domain [Local interface variables ordered last]:**

$$\underbrace{\begin{pmatrix} B_i & F_i \\ E_i & C_i \end{pmatrix}}_{A_i} \begin{pmatrix} u_i \\ y_i \end{pmatrix} + \underbrace{\begin{pmatrix} 0 \\ \sum_{j \in N_i} E_{ij} y_j \end{pmatrix}}_{y_{ext}} = \begin{pmatrix} f_i \\ g_i \end{pmatrix}$$

- u_i **: Internal variables;** y_i **: Interface variables**

Global viewpoint

Order all interior variables first

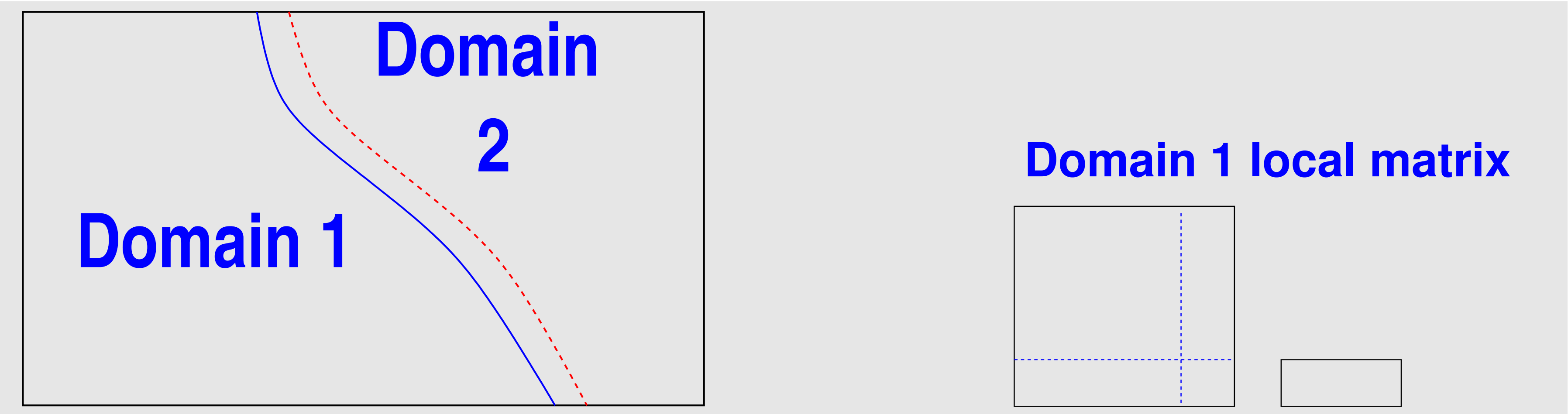
$$
\left(\begin{array}{ccccc|cccc}
B_1 & & & & & F_1 & & & \\
& B_2 & & & & & F_2 & & \\
& & \ddots & & & & & \cdot & \\
& & & \ddots & & & & & \cdot \\
& & & & B_p & & & & F_p \\
\hline
E_1 & & & & & C_1 & E_{12} & \cdots & E_{1p} \\
& E_2 & & & & E_{21} & C_2 & \cdots & E_{2p} \\
& & \ddots & & & \vdots & \vdots & \vdots & \\
& & & & E_p & E_{p1} & E_{p2} & \cdots & C_p
\end{array}\right)
\begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ \vdots \\ u_p \\ y_1 \\ y_2 \\ \vdots \\ y_p \end{pmatrix}
=
\begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ \vdots \\ f_p \\ g_1 \\ g_2 \\ \vdots \\ g_p \end{pmatrix}
$$

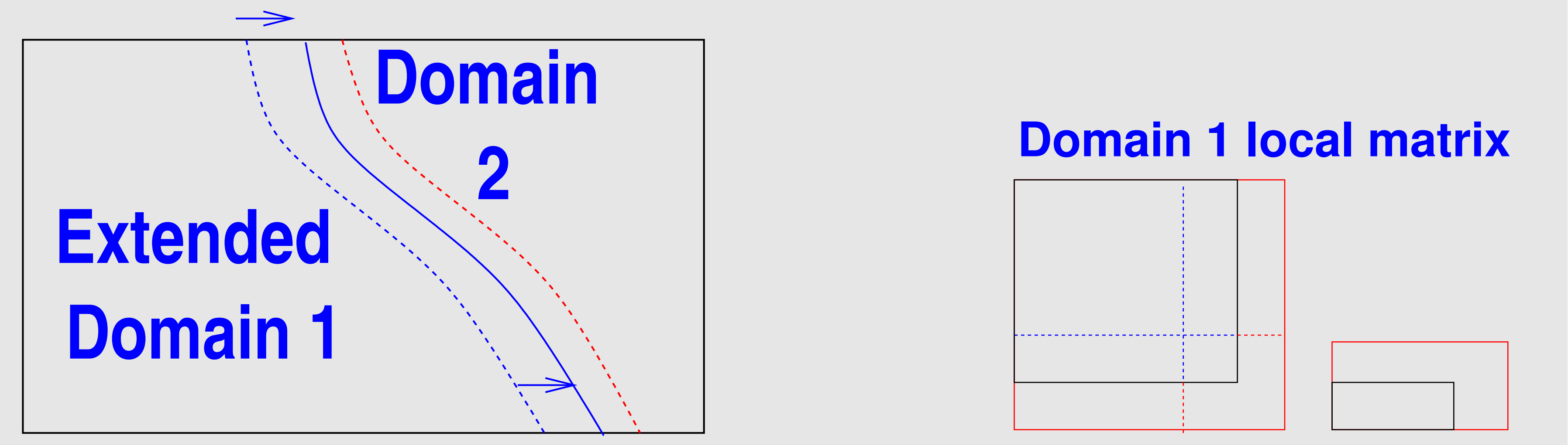
$\longleftarrow$ Interior variables $\longrightarrow$ $\longleftarrow$ Interface variables $\longrightarrow$

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:**

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

- **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

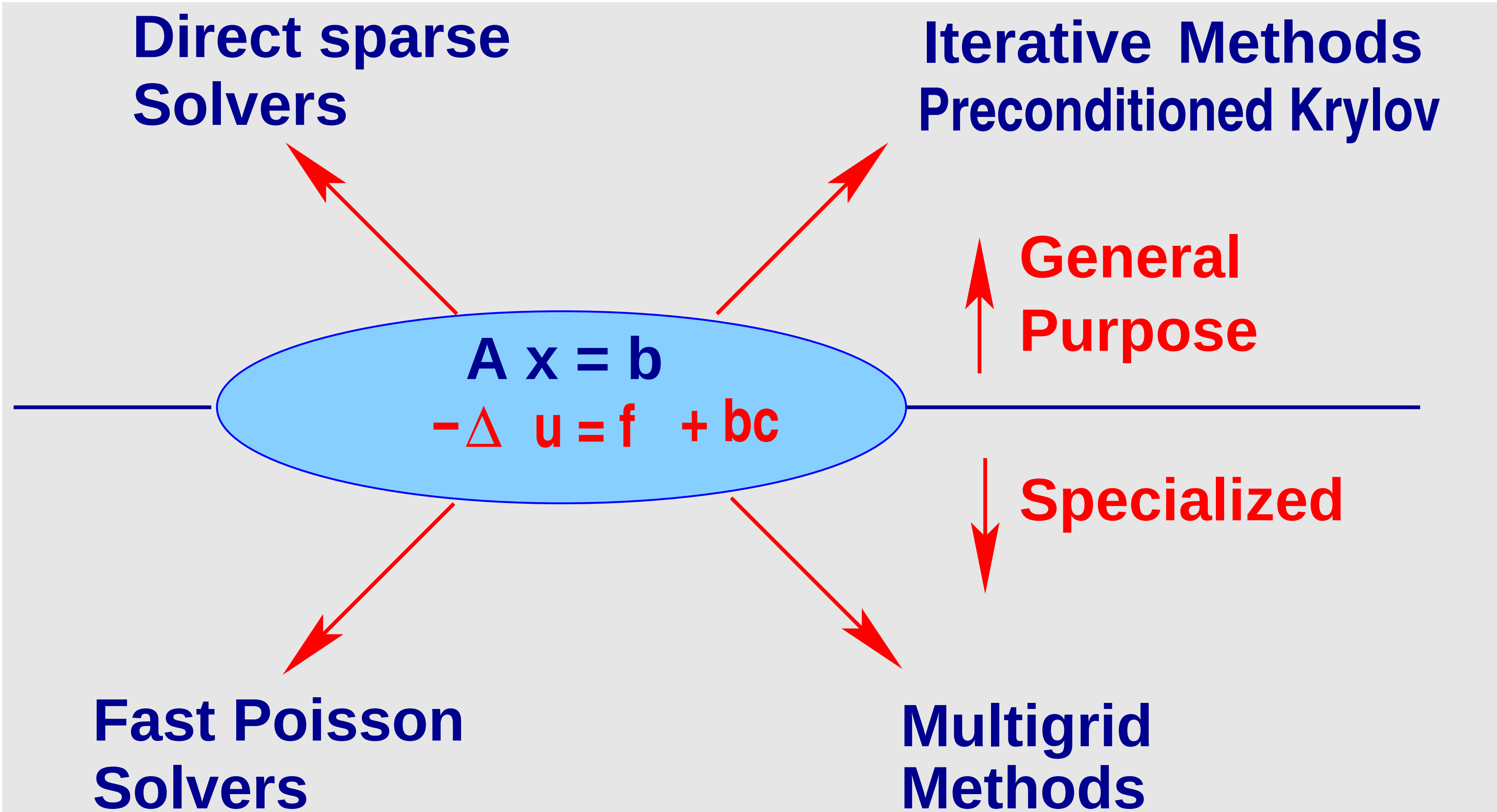
Conclusion

- **ARMS+DDpq works well as a "general-purpose" solver.**

- **Far from being a 100% robust iterative solver ...**

- **Recent work on generalizing nonsymmetric permutations to symmetric matrices [Duff-Pralet, 2006].**

- **As a general rule: ILU-based preconditioners are not meant to replace taylored preconditioners – but they can be used as general purpose tools as parts of other techniqes.**



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/.`