Matthias Bollhöfer
TU Berlin / TU Braunschweig
Dept. of Mathematics

UNIVERSTIY OF MINNESOTA, APRIL 09, 2004

OUTLINE

1. Theoretical background
2. Multilevel algorithms
3. ILUPACK
   (a) Names, formats and data types
   (b) Getting started
   (c) Templates that build up ILUPACK
   (d) Components from other packages
4. Numerical results
5. Conclusions

$$Ax = b$$

INCOMPLETE LU DECOMPOSITION

k steps of an incomplete LU decomposition

$$A = \begin{pmatrix} B & F \\ E & C \end{pmatrix} = \underbrace{\begin{pmatrix} L_B & 0 \\ L_E & I \end{pmatrix}}_{L_k} \underbrace{\begin{pmatrix} D_B & 0 \\ 0 & S_C \end{pmatrix}}_{D_k} \underbrace{\begin{pmatrix} U_B & U_F \\ 0 & I \end{pmatrix}}_{U_k} + E_k,$$

where L_B, U_B are unit diagonal, S_C remaining matrix ("approximate Schur complement").

INVERSE ERROR

For preconditioning we have to apply $L_k^{-1} A U_k^{-1}$.

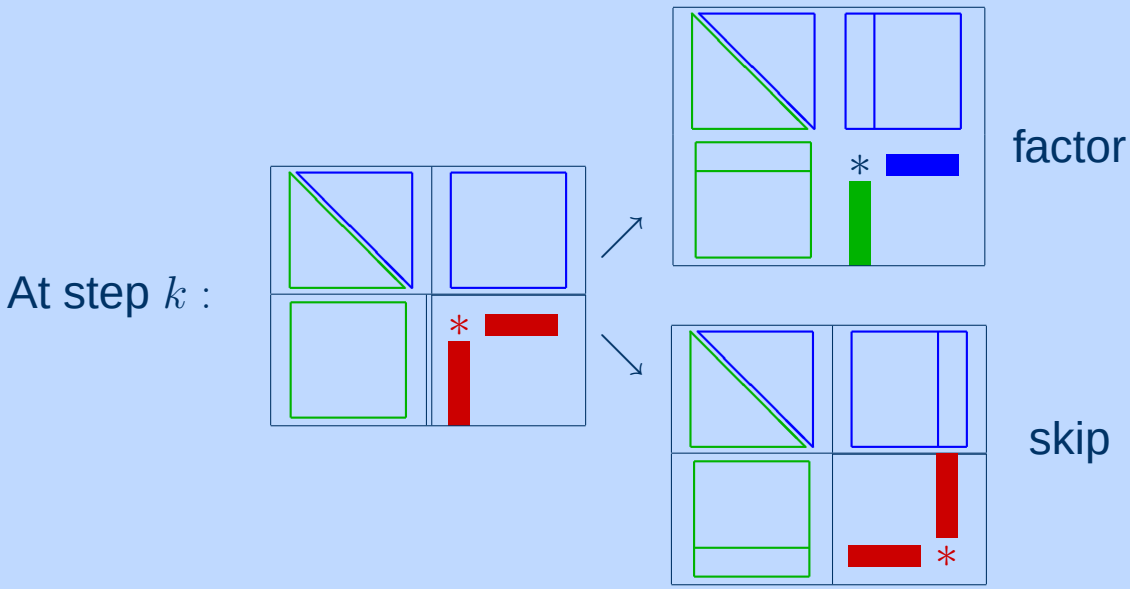
$$\longrightarrow F_k := L_k^{-1} E_k U_k^{-1} \text{ "inverse error"}$$

Problem. $\|L_k^{-1}\|$ and $\|U_k^{-1}\|$ may become very large $\Rightarrow$ pivoting recommended.

Consequences

Prescribe a bound κ and control growth $\|L_k^{-1}\|, \|U_k^{-1}\| \leqslant \kappa$ by diagonal pivoting.

Factor or skip strategy



Rows/columns that exceed the prescribed bound are pushed to the end

SKETCH OF THE TEMPLATES

1. Compute a static partial reordering

$$A \to P^\top A Q = \begin{pmatrix} B & F \\ E & C \end{pmatrix}$$

2. Factor $P^\top A Q$ and control $\|L_k^{-1}\|, \|U_k^{-1}\| \leqslant \kappa$.

$$P^\top A Q \longrightarrow \hat{P}^\top A \hat{Q} = \begin{pmatrix} \hat{B} & \hat{F} \\ \hat{E} & \hat{C} \end{pmatrix} = \begin{pmatrix} L_B & 0 \\ L_E & I \end{pmatrix} \begin{pmatrix} D_B & 0 \\ 0 & S_C \end{pmatrix} \begin{pmatrix} U_B & U_F \\ 0 & I \end{pmatrix}$$

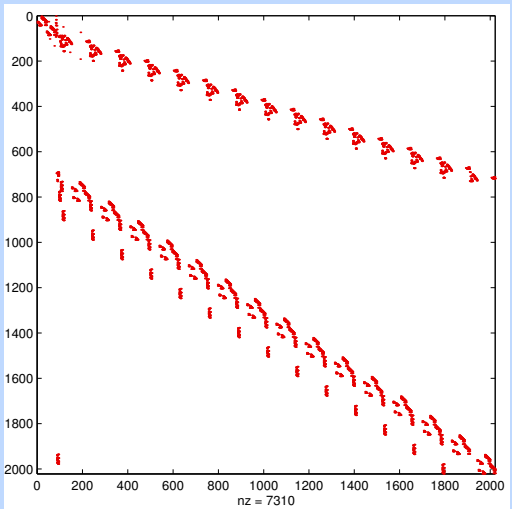
3. Apply strategy recursively to the approximate Schur complement S_C (multilevel scheme)

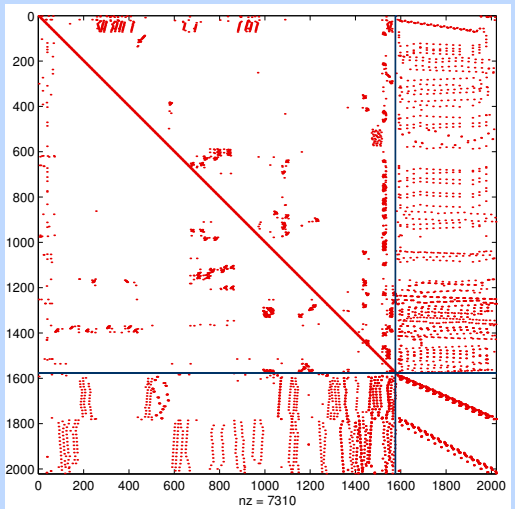
$$A \xrightarrow{1.} P^\top A Q \xrightarrow{2.} \hat{P}^\top A \hat{Q} \xrightarrow{3.} \left(\begin{array}{c|c} L_B \cdot D_B \cdot U_B & F \\ \hline E & S_C \end{array}\right)$$

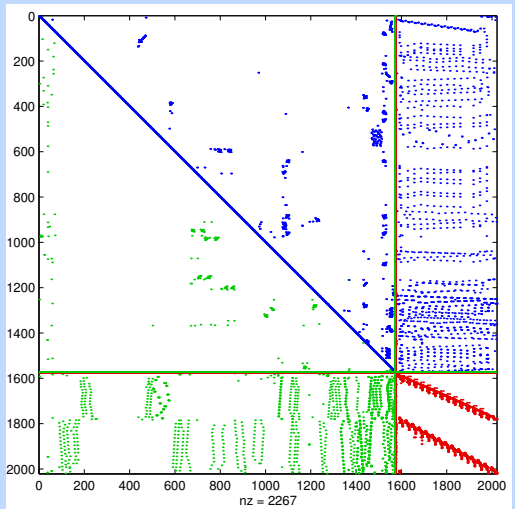
$$S_C \xrightarrow{1.} P_S^\top S_C Q_S \xrightarrow{2.} \hat{P}_S^\top S_C \hat{Q}_S \xrightarrow{3.} \left(\begin{array}{c|c} L_S \cdot D_S \cdot U_S & F_S \\ \hline E_S & T \end{array}\right)$$

$$\ldots$$

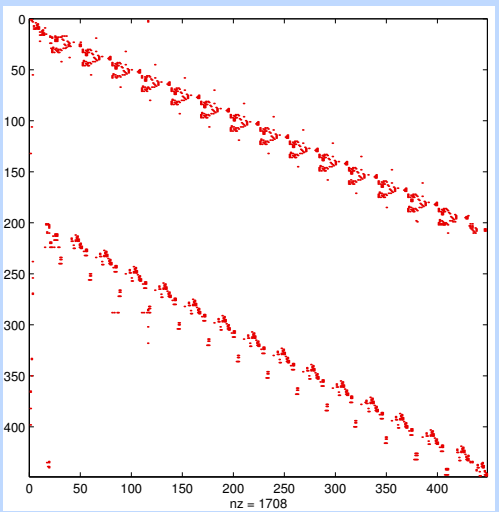
Inverse–Based Multilevel ILUs

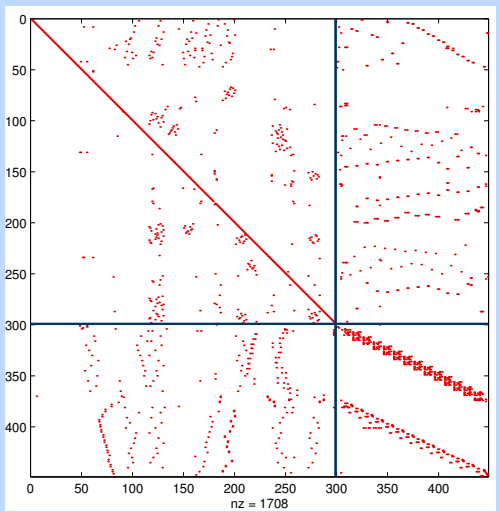


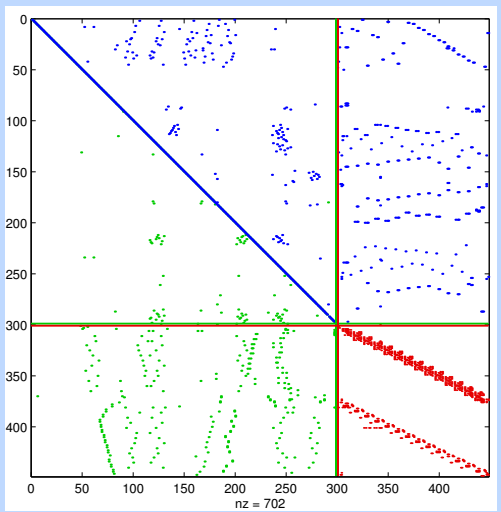




$$A \xrightarrow{1.} P^\top A Q \xrightarrow{2.} \hat{P}^\top A \hat{Q} \xrightarrow{3.} \left(\begin{array}{c|c} L \cdot D \cdot U & F \\ \hline E & S \end{array}\right)$$







Names And Formats

- ILUPACK currently supports real single/double precision and complex single/double precision.
- Apart from the arithmetic ILUPACK distinguishes between two classes of matrices, namely general matrices and symmetric (Hermitian) positive definite matrices.
- Most commands require two prefixes, one character for the arithmetic and an additional three–character prefix for the class.

S	real single precision
D	real double precision
C	complex single precision
Z	complex double precision

GNL	general matrices
SPD	real symmetric positive definite matrices
HPD	complex Hermitian positive definite matrices

- matrices are stored in SPARSKIT format (compressed sparse row format), indices starting with 1, FORTRAN–style.
- Symmetric matrices. Let D be the diagonal part of A and $A = D + U + U^\top$, then $D + U$ are stored.

Examples

- A command that computes the incomplete LU factorization of A is available as

General Case	Symmetric Positive Definite Case
`SGNLAMGfactor(...)`	`SSPDAMGfactor(...)`
`DGNLAMGfactor(...)`	`DSPDAMGfactor(...)`
`CGNLAMGfactor(...)`	`CHPDAMGfactor(...)`
`ZGNLAMGfactor(...)`	`ZHPDAMGfactor(...)`

- Compressed sparse row format.

$$\begin{pmatrix} -.1 & .2 & 0 \\ 0 & 1.1 & 0 \\ 0 & 0 & 2.3 \end{pmatrix} \rightarrow \begin{array}{c|cccc} \texttt{ia} & 1 & 3 & 4 & 5 \\ \texttt{ja} & 1 & 2 & 2 & 3 \\ \texttt{a} & -.1 & .2 & 1.1 & 2.3 \end{array}, \quad \begin{pmatrix} \boxed{3.1} & \boxed{-.2} & 0 \\ -.2 & \boxed{1.1} & \boxed{.4} \\ 0 & .4 & \boxed{2.3} \end{pmatrix} \rightarrow \begin{array}{c|ccccc} \texttt{ia} & 1 & 3 & 5 & 6 & \\ \texttt{ja} & 1 & 2 & 2 & 3 & 3 \\ \texttt{a} & 3.1 & -.2 & 1.1 & .4 & 2.3 \end{array}$$

DATATYPES

- `mat A` (i.e. **S**`mat`, **D**`mat`, **C**`mat` or **Z**`mat`). Sparse matrices:

  `A.nr,A.nc`: number of rows and columns
  `A.ia,A.ja,A.a`: pointer, indices and numerical values in SPARSKIT sparse row format.

- `AMGlevelmat PRE`. Multilevel preconditioner, double linked list, each level:

`PRE.n,PRE.nB:`	total size and size of the leading block
`PRE.LU:`	sparse matrix continaing the ILU.
`PRE.LUperm:`	permutation array (in case of ILUTP)
`PRE.E,PRE.F:`	submatrices needed for multilevel ILU parts E and F.
`PRE.p,PRE.invq:`	row and inverse column permutation
`PRE.rowscal,PRE.colscal:`	numerical values for row and column scaling
`PRE.prev,PRE.next:`	pointers to the previous and next level

- `ILUPACKPARAM param`. Parameters to control miscellaneous ILUPACK routines:

  `param.ipar[40],param.fpar[40]`: integer and double parameters. many,many switches!
  `param.ibuff,param.dbuff`: integer and numerical value buffers.

```
// init param with the default parameters
    AMGinit(A, &param);

// get some of the most common parameters and overwrite them
    AMGgetparams(param, &flags, &fill, droptols, &kappa, &restol,
                 &max_it, &nrestart);

// main drop tolerance and elbow space measured by nnz(LU)/nnz(A)
droptols[1]=5e-3; fill=5.0;

// rewrite the updated parameters
    AMGsetparams(&param, flags, fill, droptols, kappa, restol,
                 max_it, nrestart);

// choose three permutation functions, initial/regular/final
permp=   permnull; permr=   permrcm; permf=   permpq;

// main factorization routine for the multilevel ILU
ierr=    AMGfactor(&A, &PRE, &nlev, &param, permp,permr,permf);

// iterative solver to solve A*sol=rhs
ierr=    AMGsolver(A, PRE, nlev, &param, sol, rhs);
```

TEMPLATE 1: STATIC REORDERINGS AND THEIR USAGE IN ILUPACK

Orderings used to partially define a well–suited leading block B

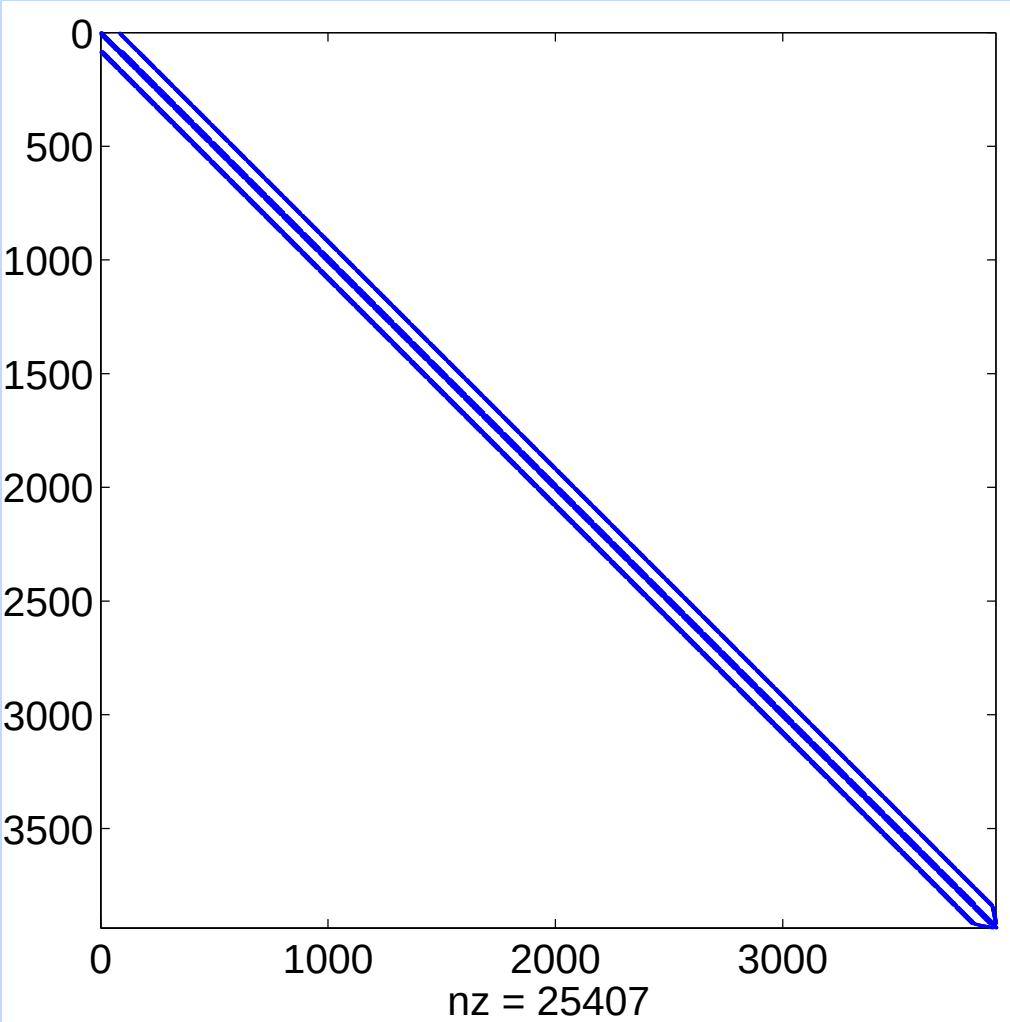
$$P^\top A Q = \begin{pmatrix} B & F \\ E & C \end{pmatrix}$$

Three orderings can be prescribed

1. Initial preprocessing.
   Ordering and scaling ONLY applied to the initial system.
2. Regular reordering.
   Used for any subsequent level, as long as possible.
3. Final pivoting.
   If the regular reordering fails to process a suitable leading block B, switch to the final pivoting strategy.

Template 1: Example

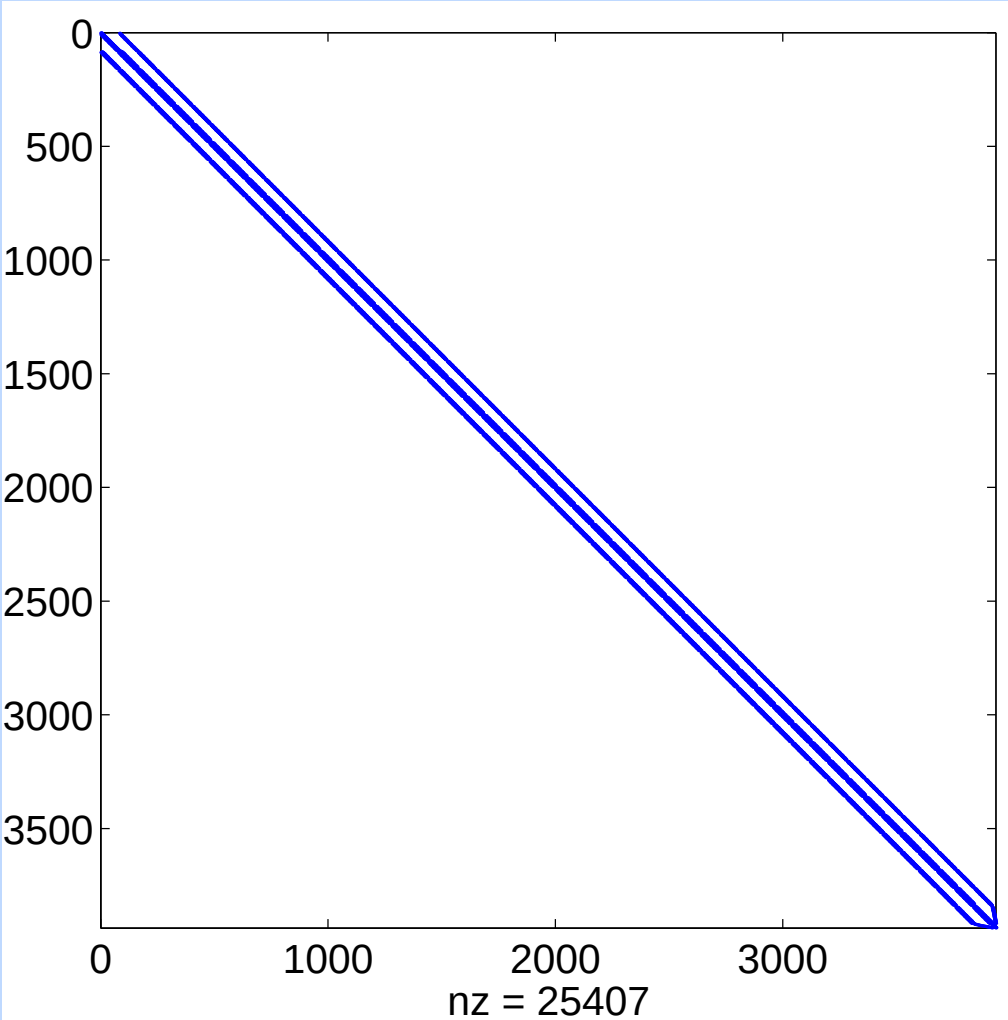
Initial System



Template 1: Example

Initial System, reordered

here: no permutation

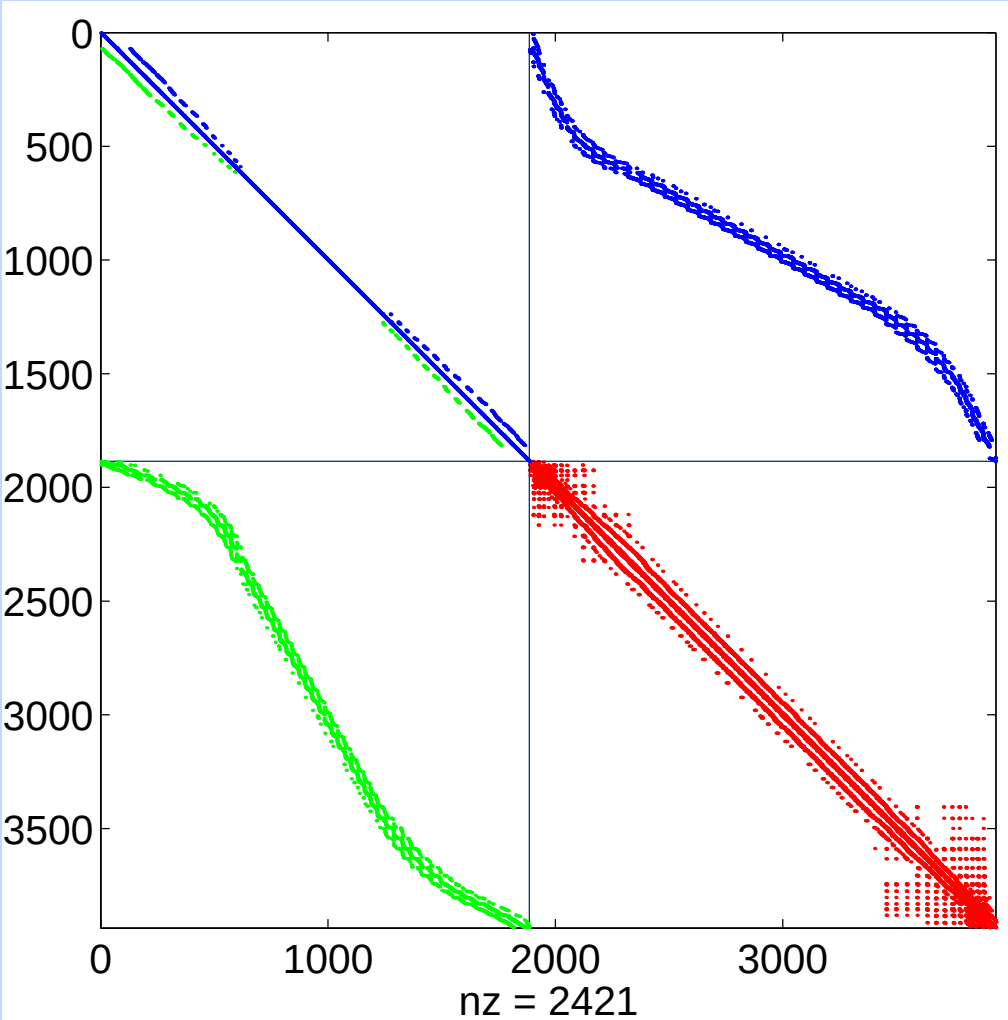


Template 1: Example

Initial Level, reorderd

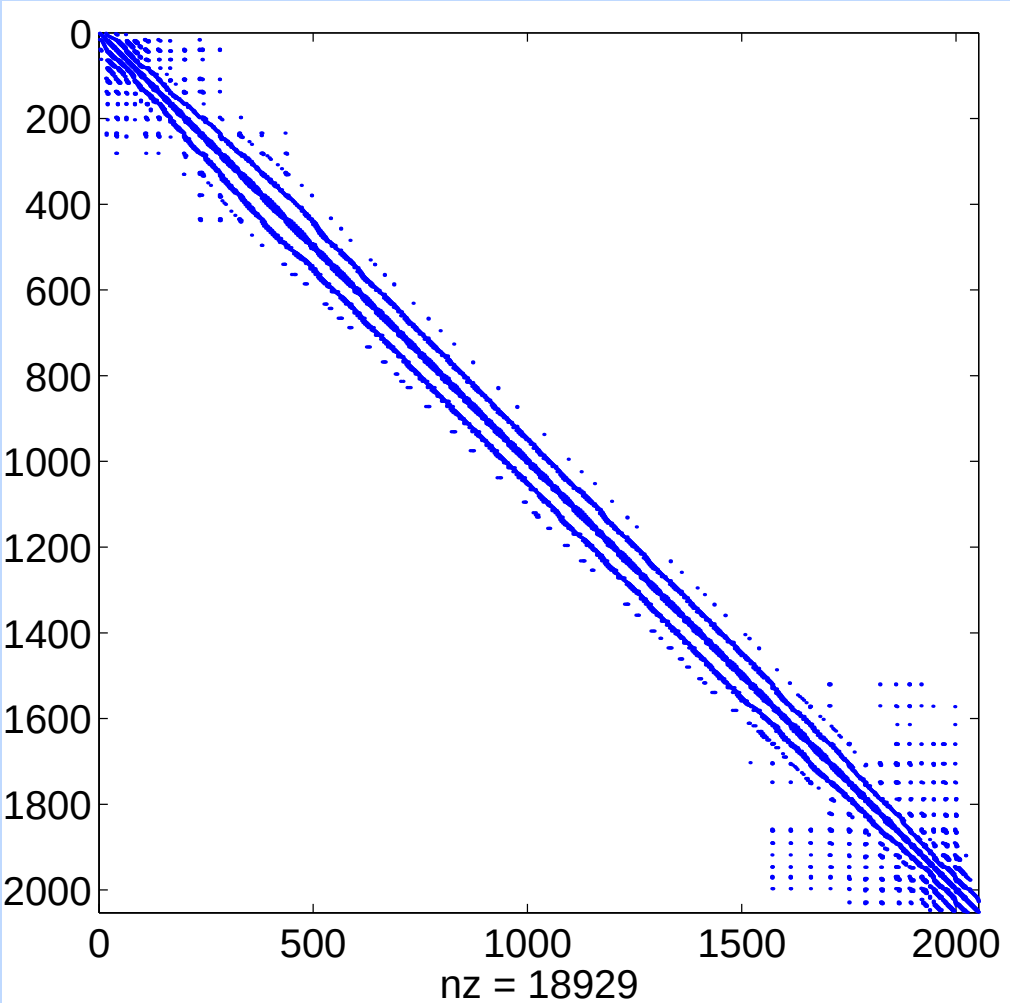
after application of ILU

such that $\|L^{-1}\|, \|U^{-1}\| \leqslant \kappa$



Template 1: Example

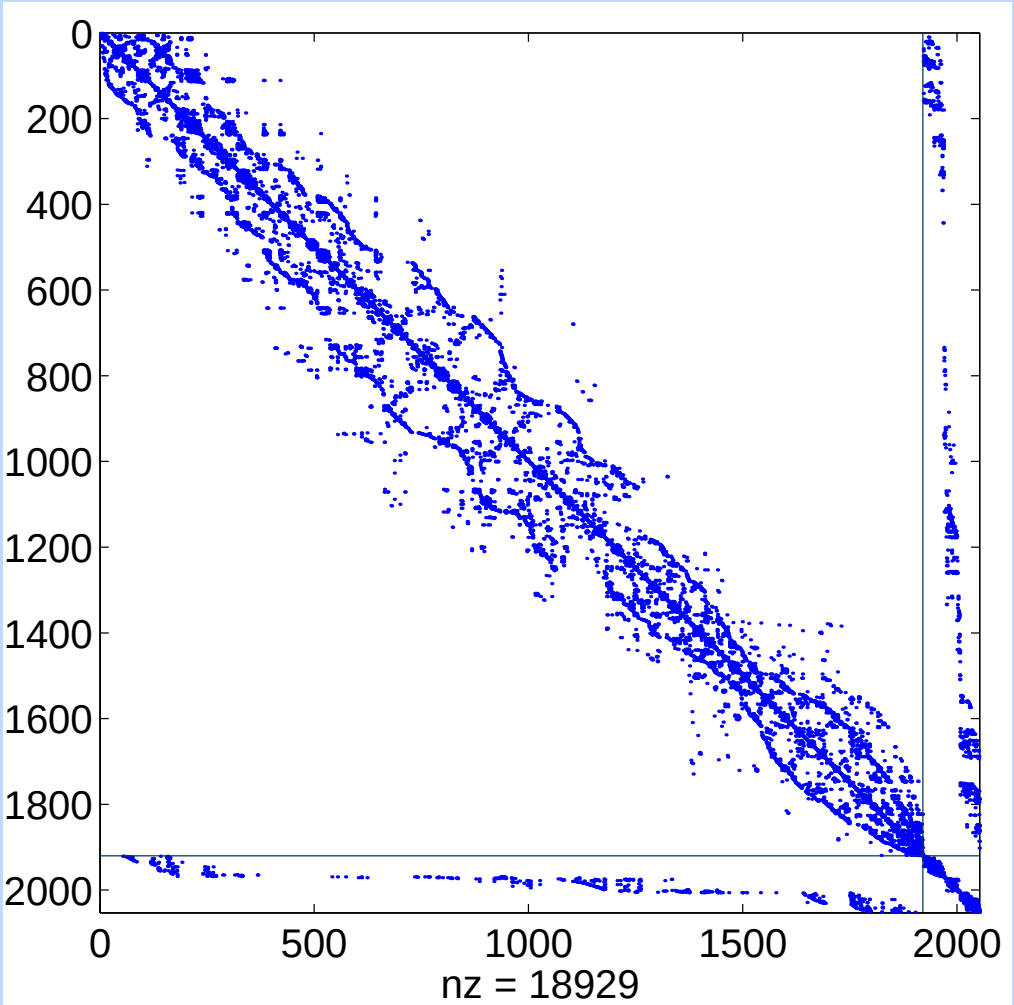
Level 2, initial



Template 1: Example

Level 2, reordered

here Reverse Cuthill-McKee

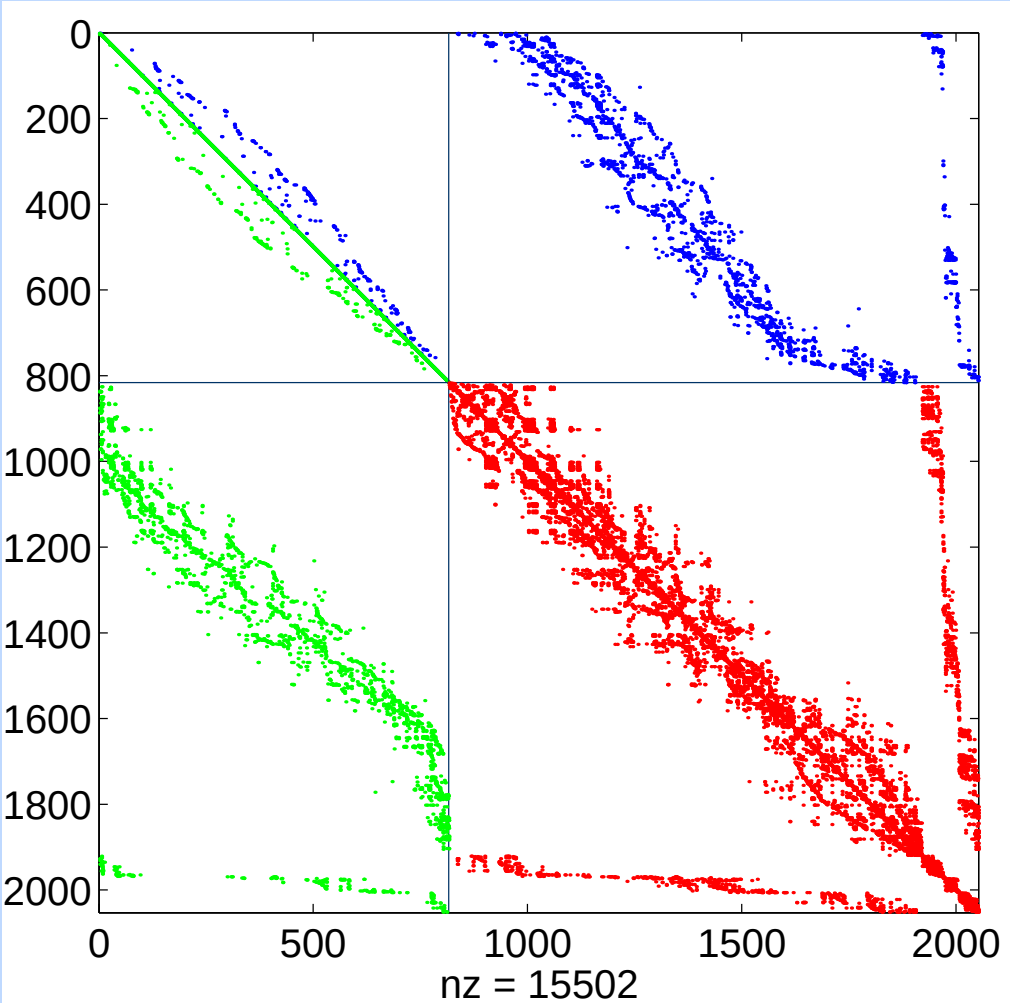


Template 1: Example

Level 2, reordered

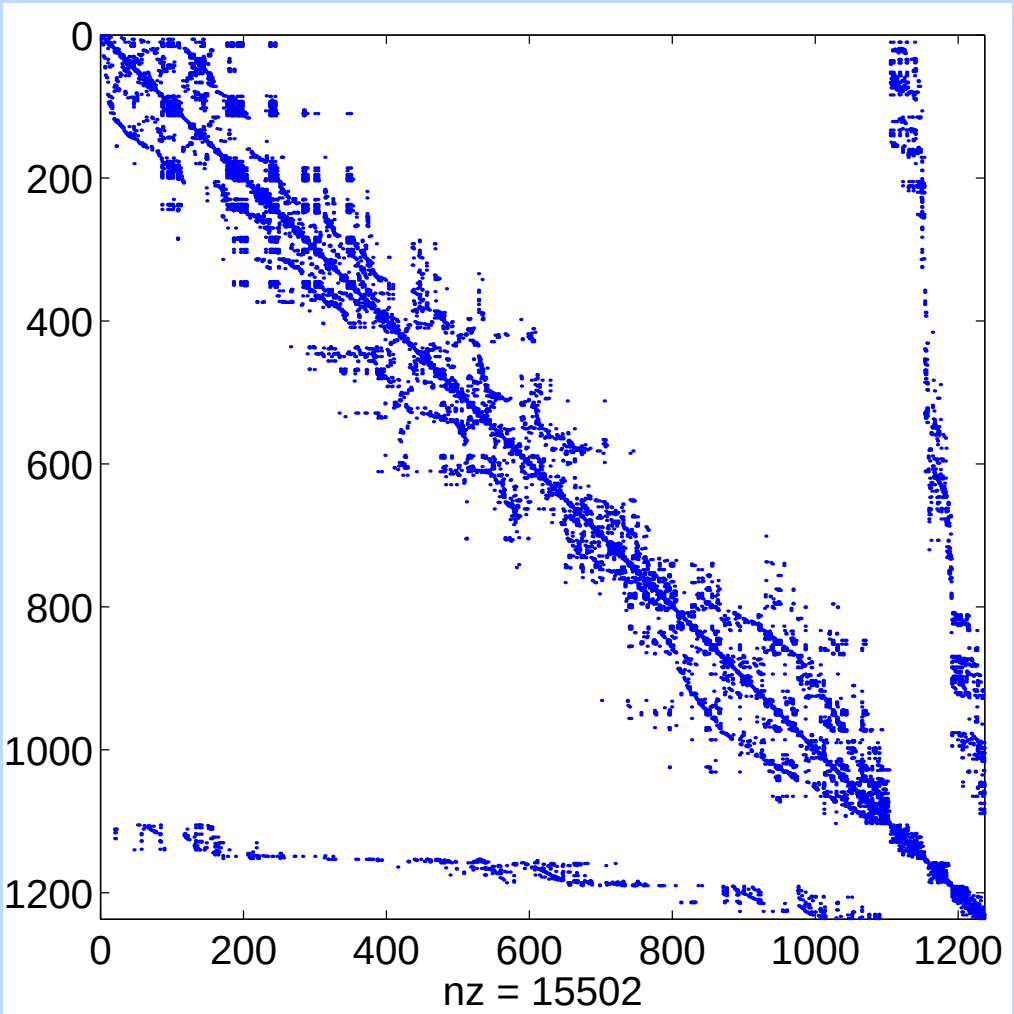
after application of ILU

such that $\|L^{-1}\|, \|U^{-1}\| \leqslant \kappa$



Template 1: Example

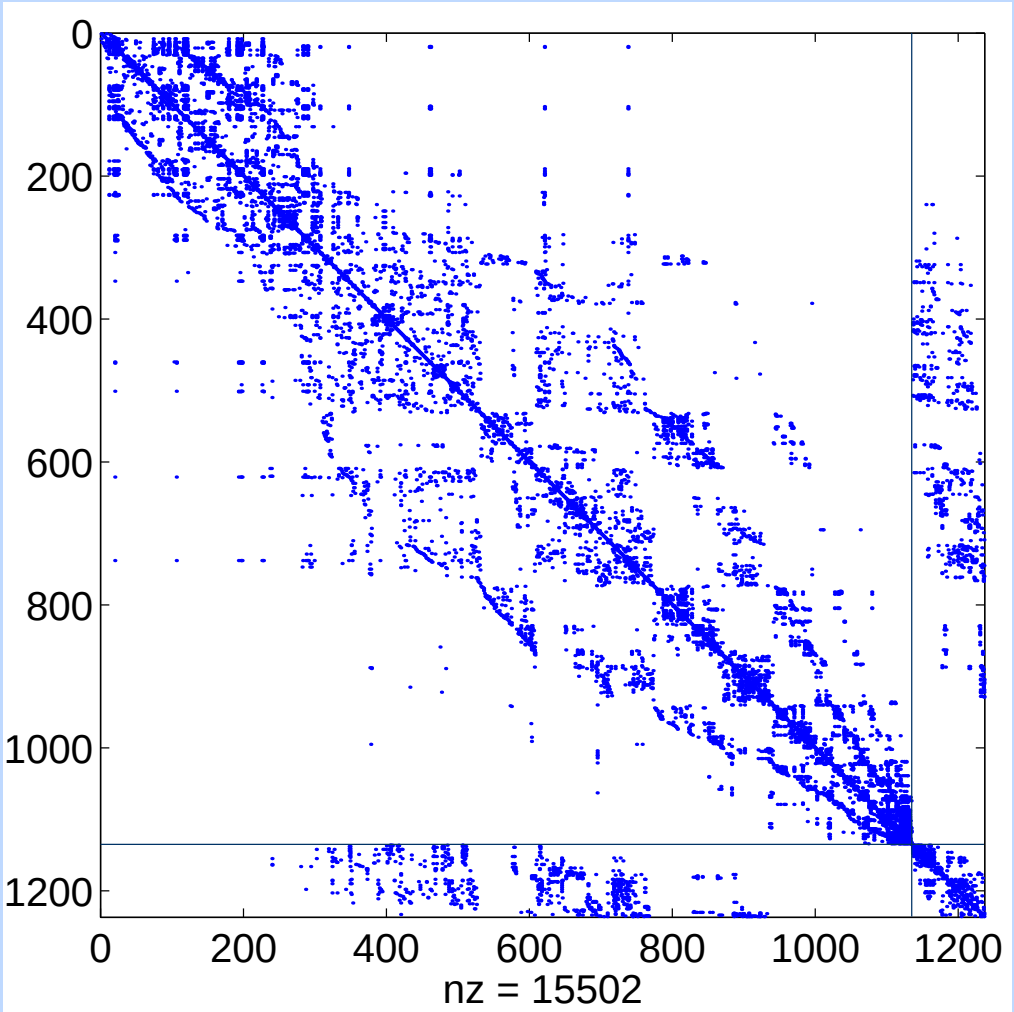
Level 3, initial



Template 1: Example

Level 3, reordered

here Reverse Cuthill-McKee

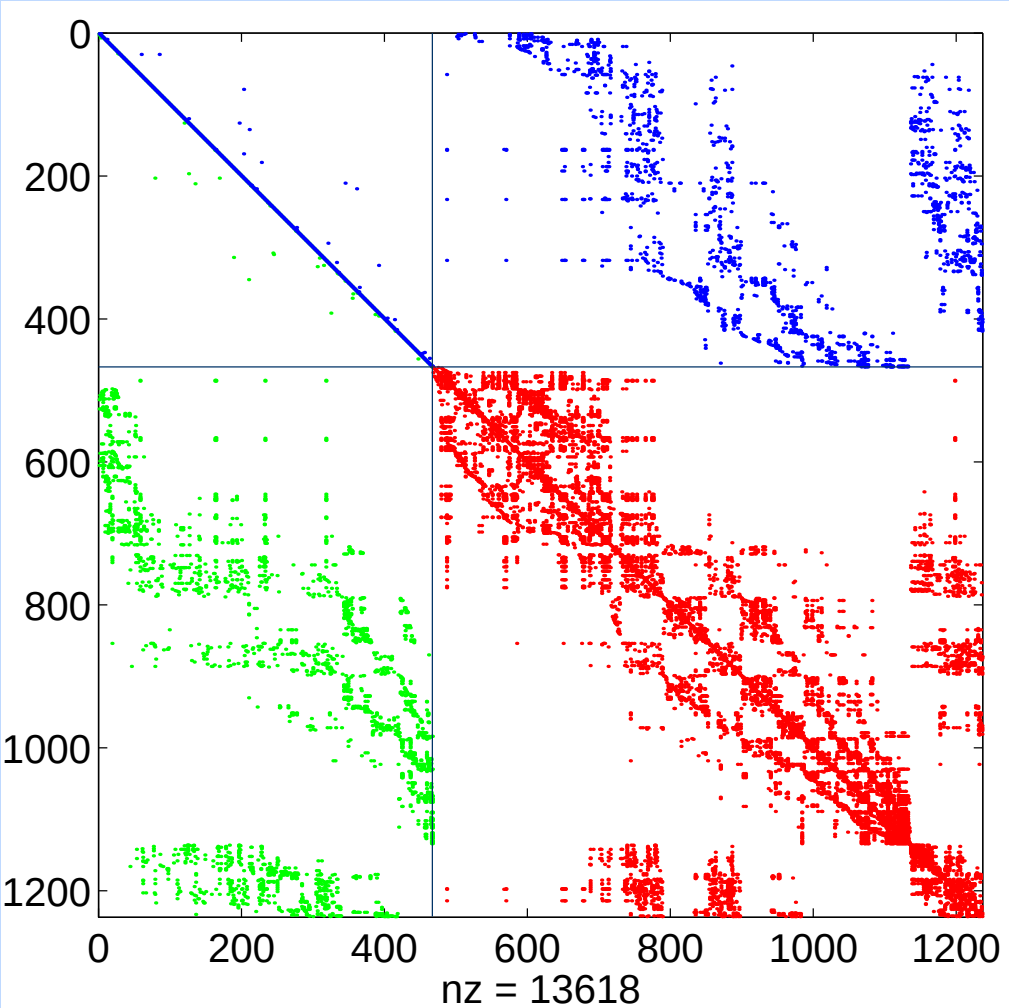


Template 1: Example

Level 3, reordered

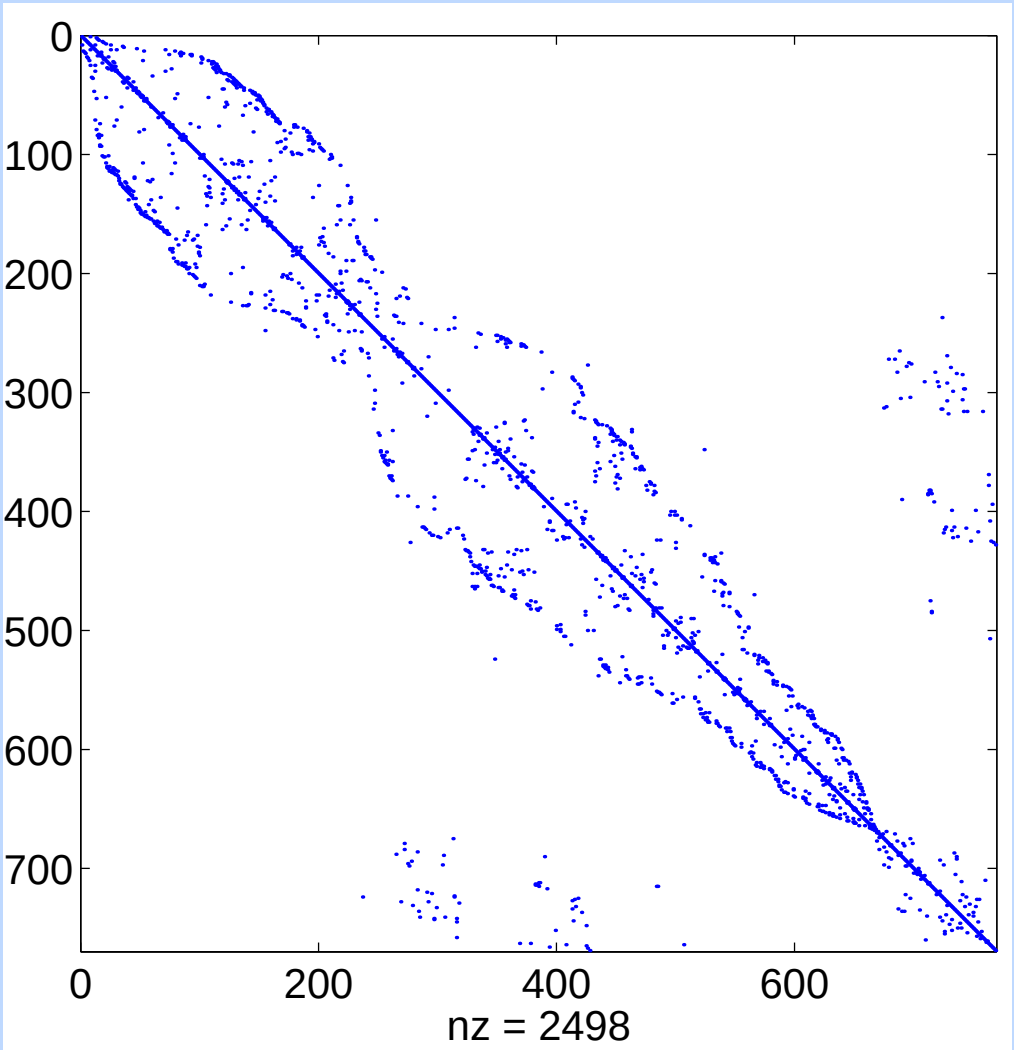
after application of ILU

such that $\|L^{-1}\|, \|U^{-1}\| \leqslant \kappa$



Template 1: Example

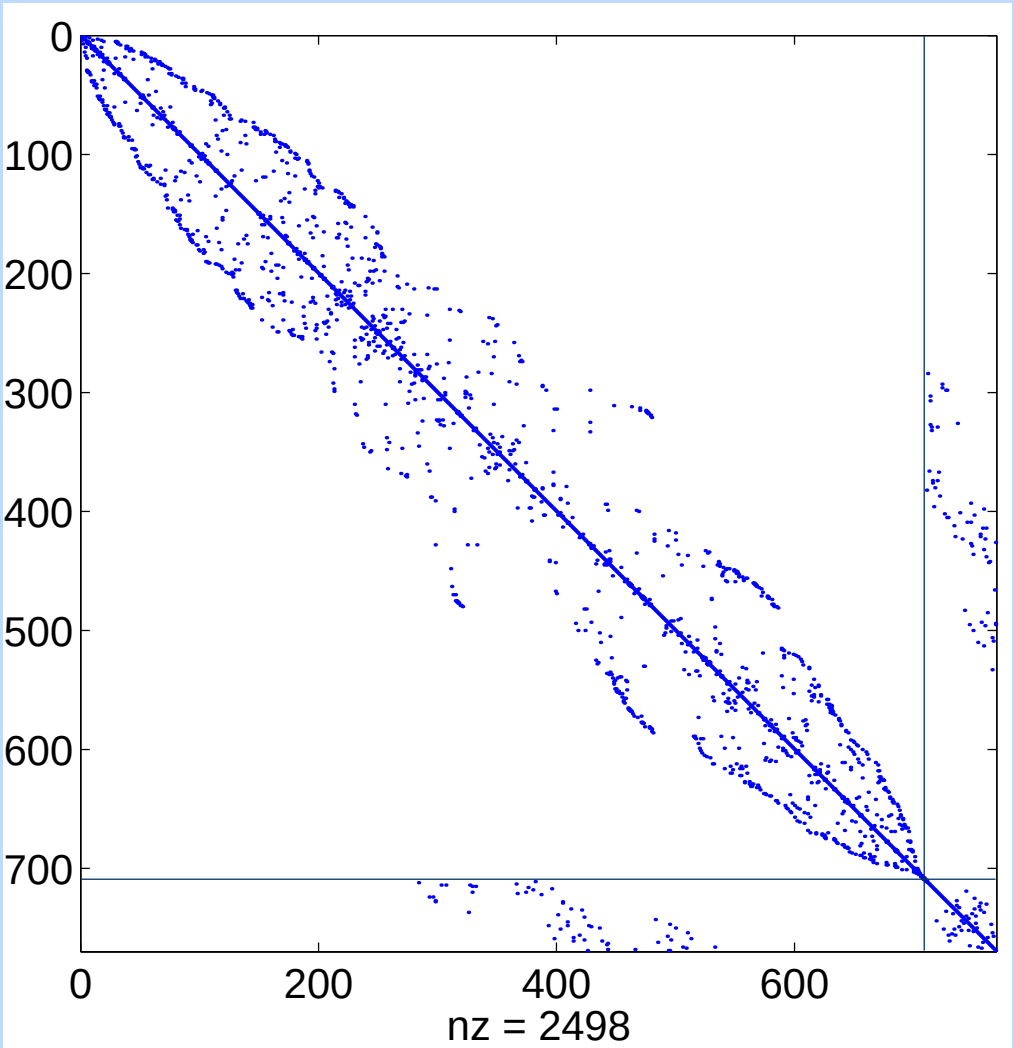
Level 4, initial



Template 1: Example

Level 4, reordered

here reverse Cuthill-McKee

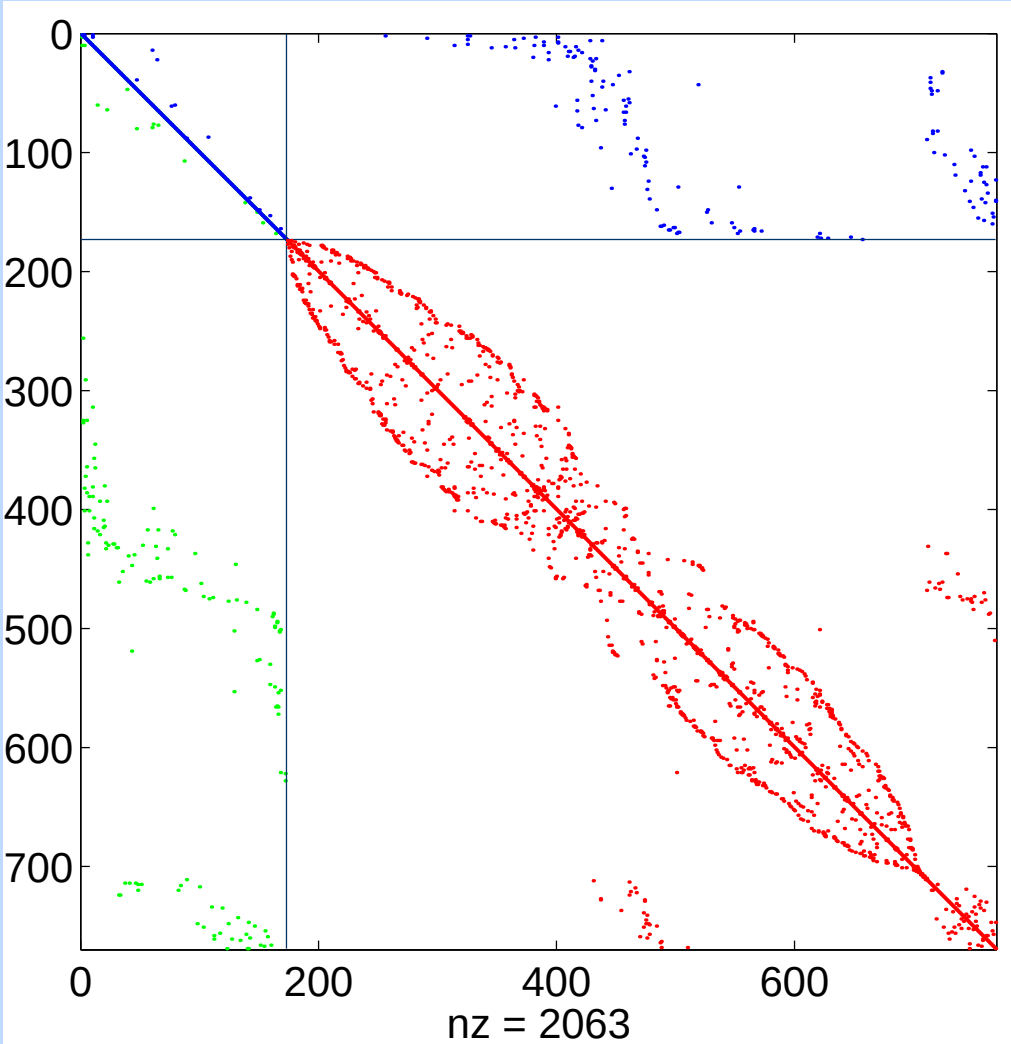


TEMPLATE 1: EXAMPLE

Level 4, reordered

after application of ILU

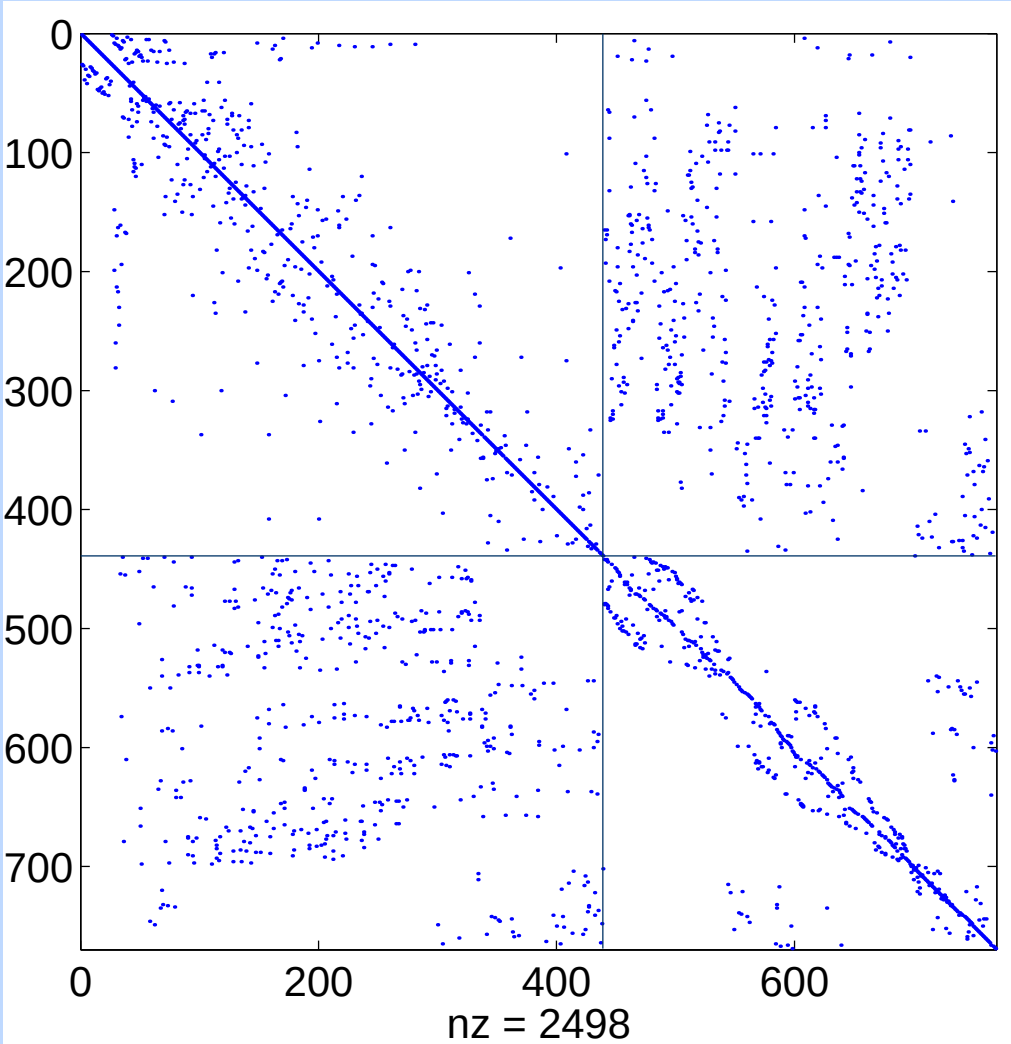
such that $\|L^{-1}\|, \|U^{-1}\| \leqslant \kappa$



Template 1: Example

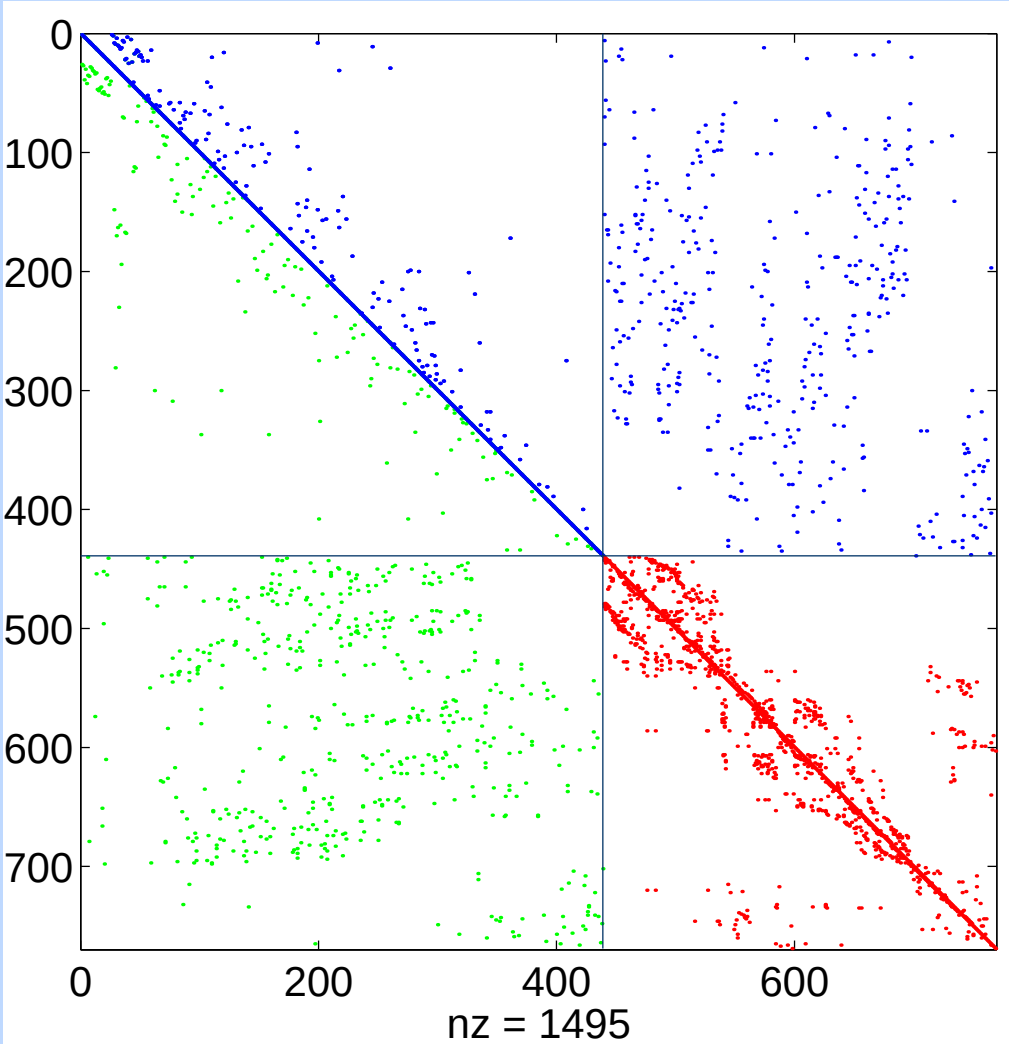
Level 4, reordered again

here PQ (switched to final pivoting)



Template 1: Example

Level 4, reordered again after application of ILU such that $\|L^{-1}\|, \|U^{-1}\| \leqslant \kappa$



TEMPLATE 1. STATIC REORDERINGS

Supported GLOBAL orderings.

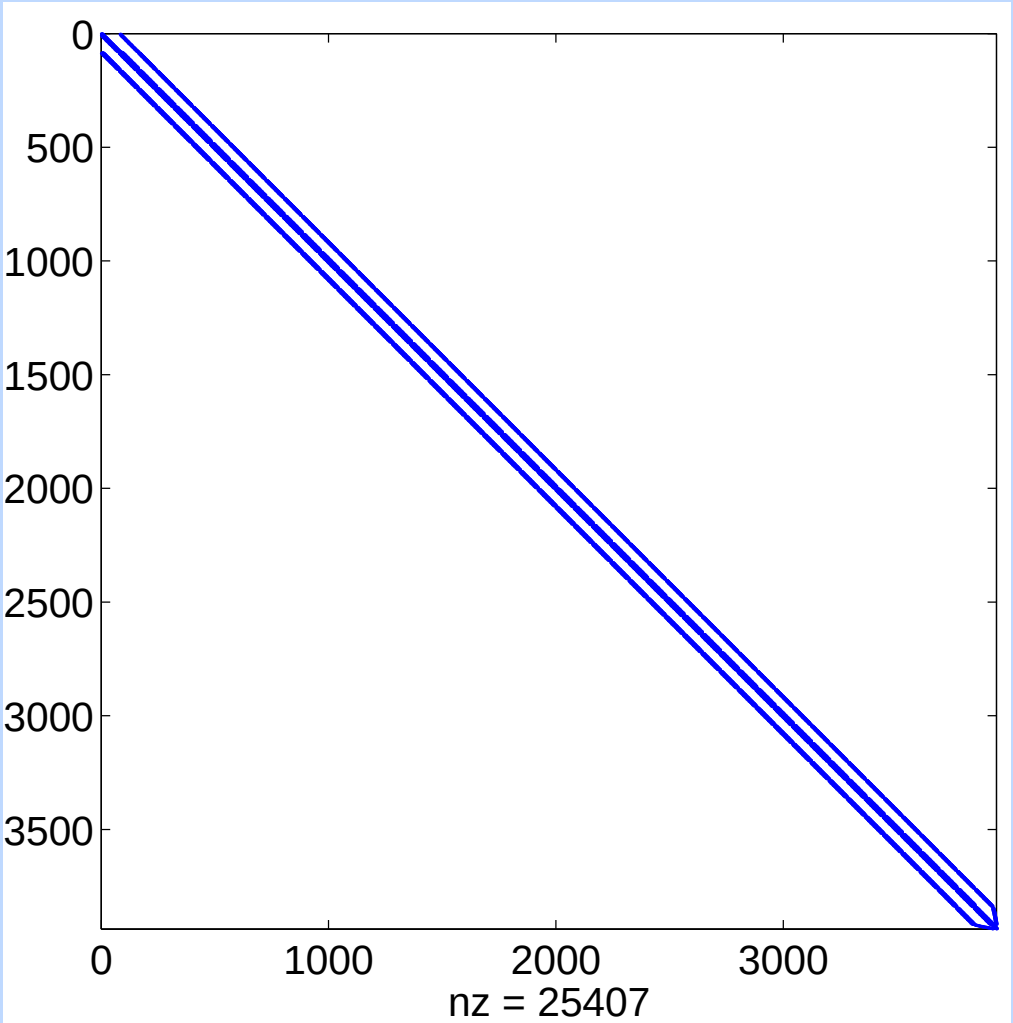
- RCM (Reverse Cuthill–McKee): banded systems, PDE-type problems.
- MMD (Multiple Minimum Degree) or AMF (Approximate Minimum Fill): almost symmetrically structured problems, not recommended for ILUs.
- ND (Nested Dissection).
- MC64 (from HSL): unstructured problems, improves diagonal dominance.

Included PARTIAL orderings.

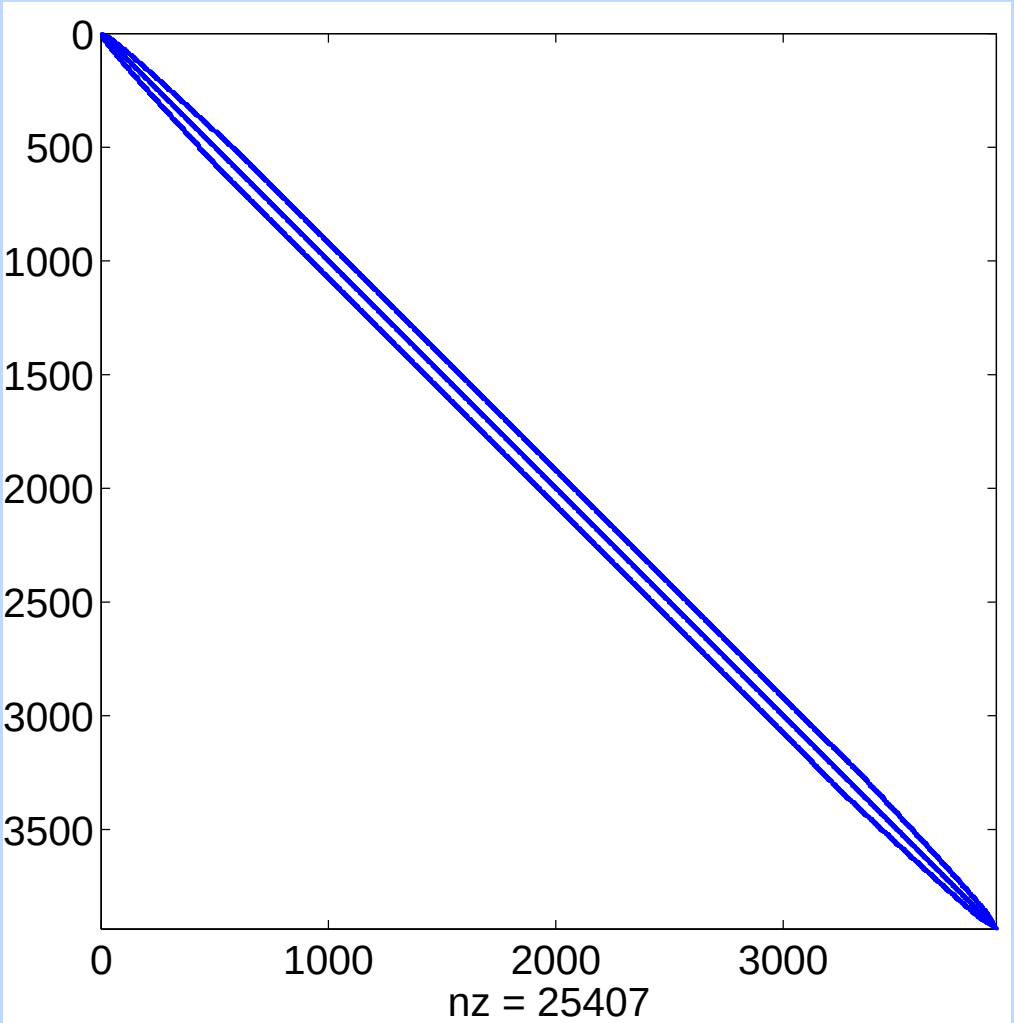
- ddPQ [Saad '03]. Permutations P, Q as a compromise between diagonal dominance and fill. Applicable to all problems, in particular unstructured systems.
- INDSET (independent set, ARMS), symmetrically structured problems, PDE–type problems.
- FC (fine grid/coarse grid partitioning similar to Ruge/Stüben AMG), symmetrically structured problems, PDE–type problems.

Any ordering offers or can be combined with column and/or row scaling

TEMPLATE 1. STATIC GLOBAL REORDERINGS

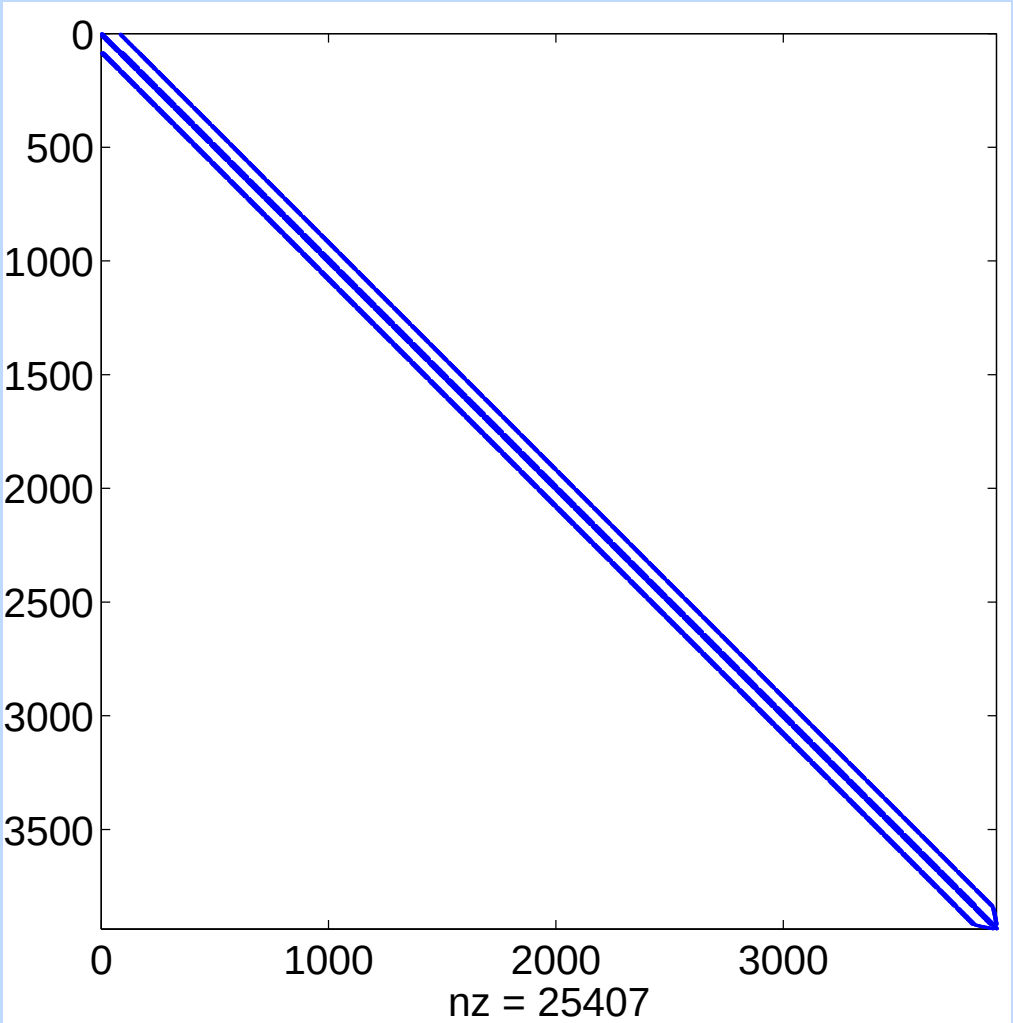


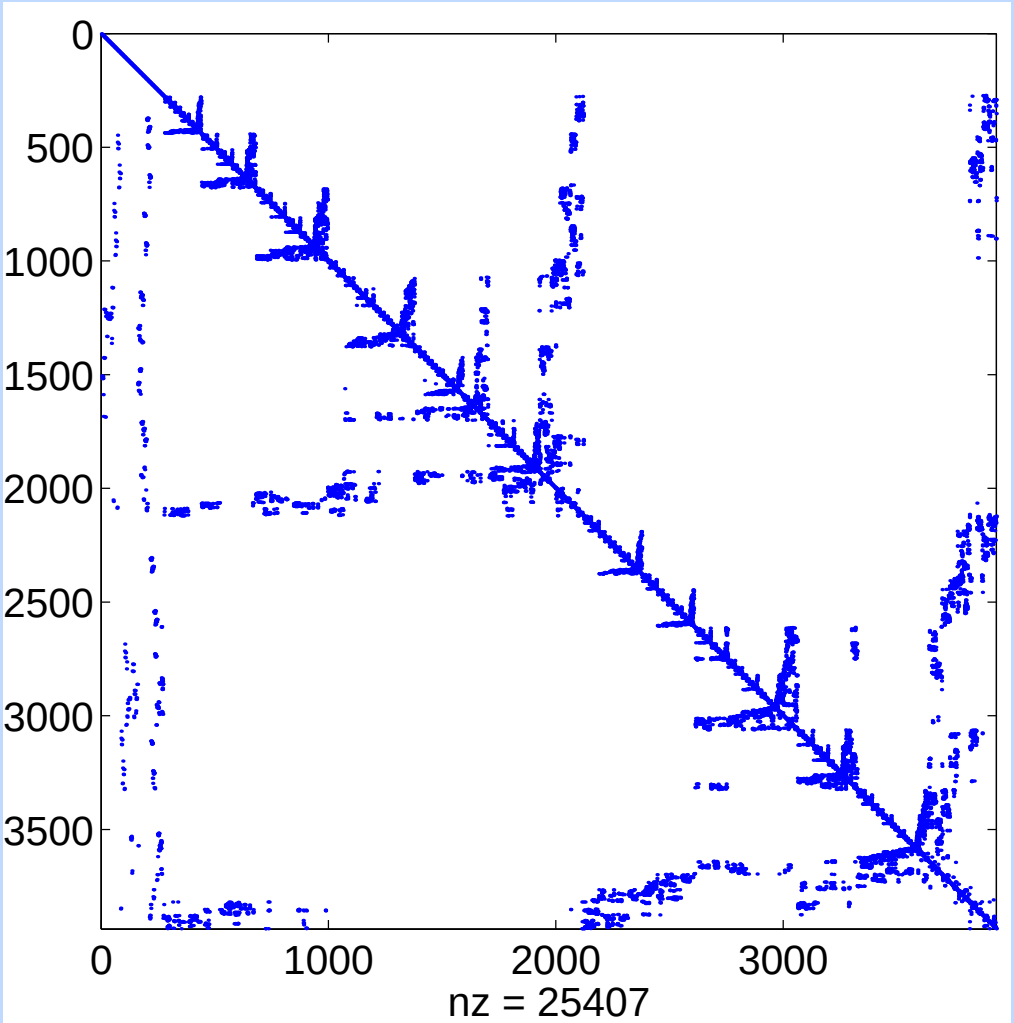
Initial System



Reordered by RCM (reverse Cuthill-McKee)

TEMPLATE 1. STATIC GLOBAL REORDERINGS

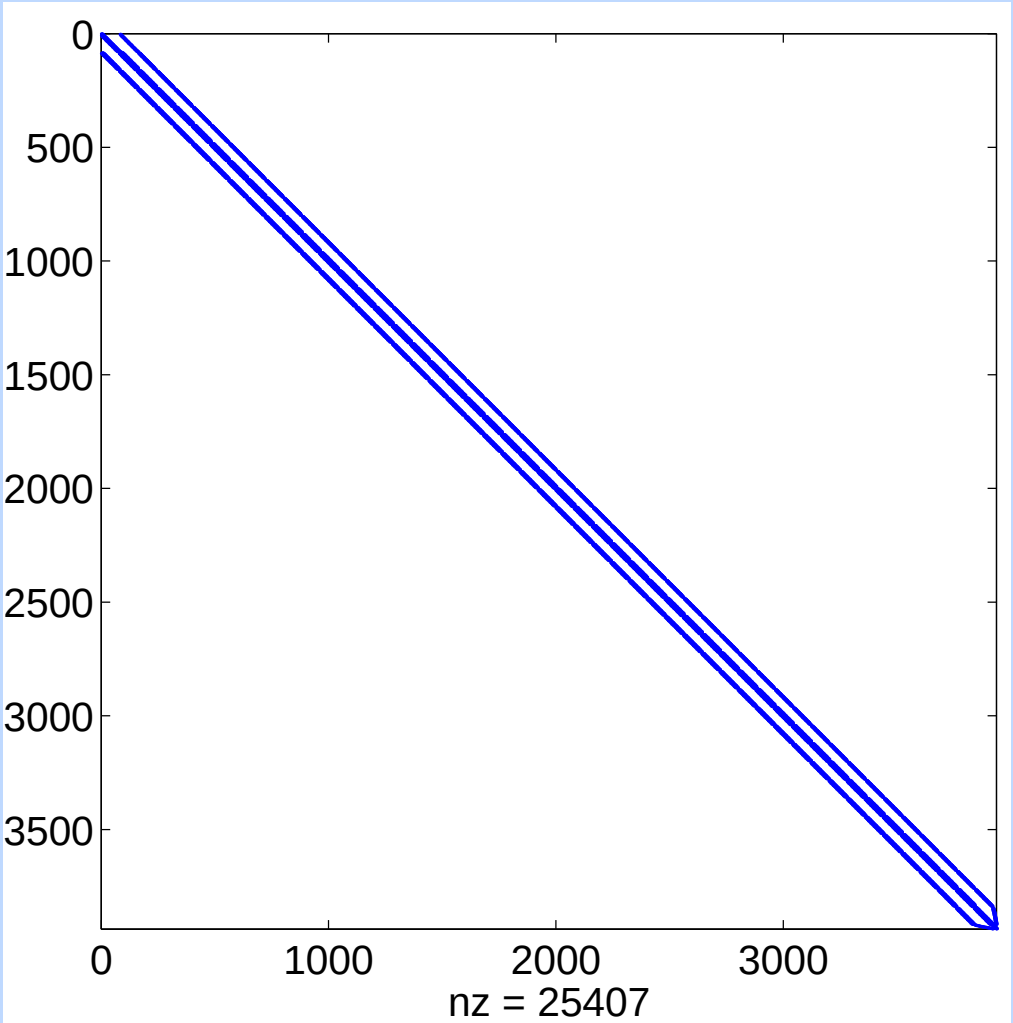




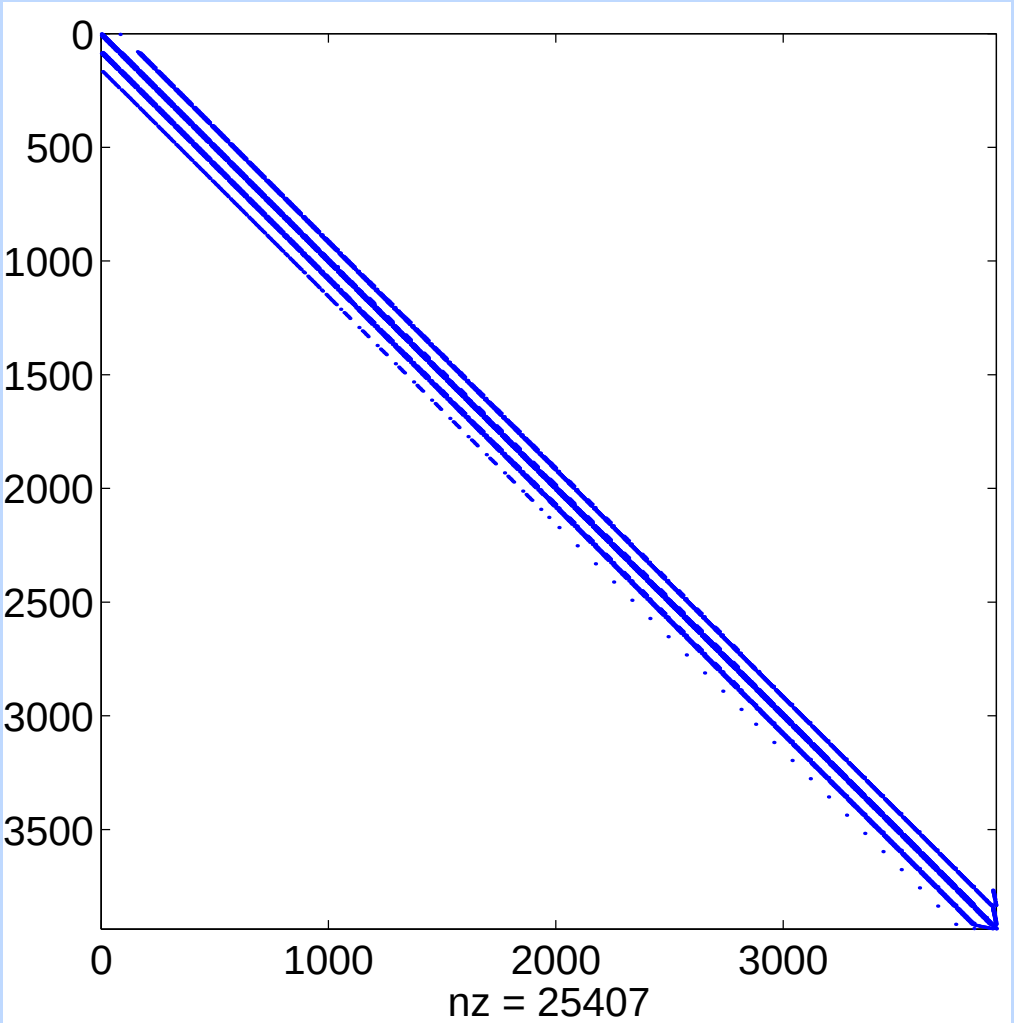
Initial System

Reordered by MMD (multiplie minimum degree)

TEMPLATE 1. STATIC GLOBAL REORDERINGS

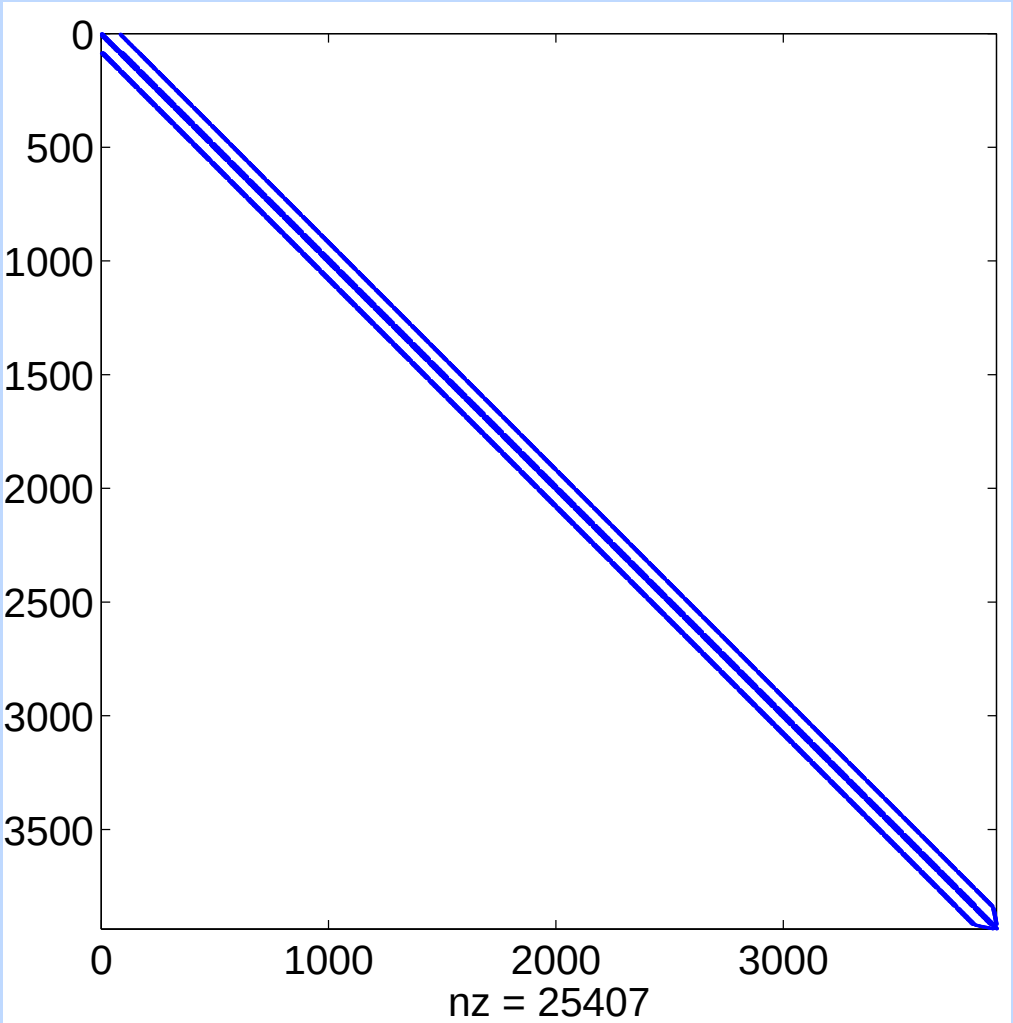


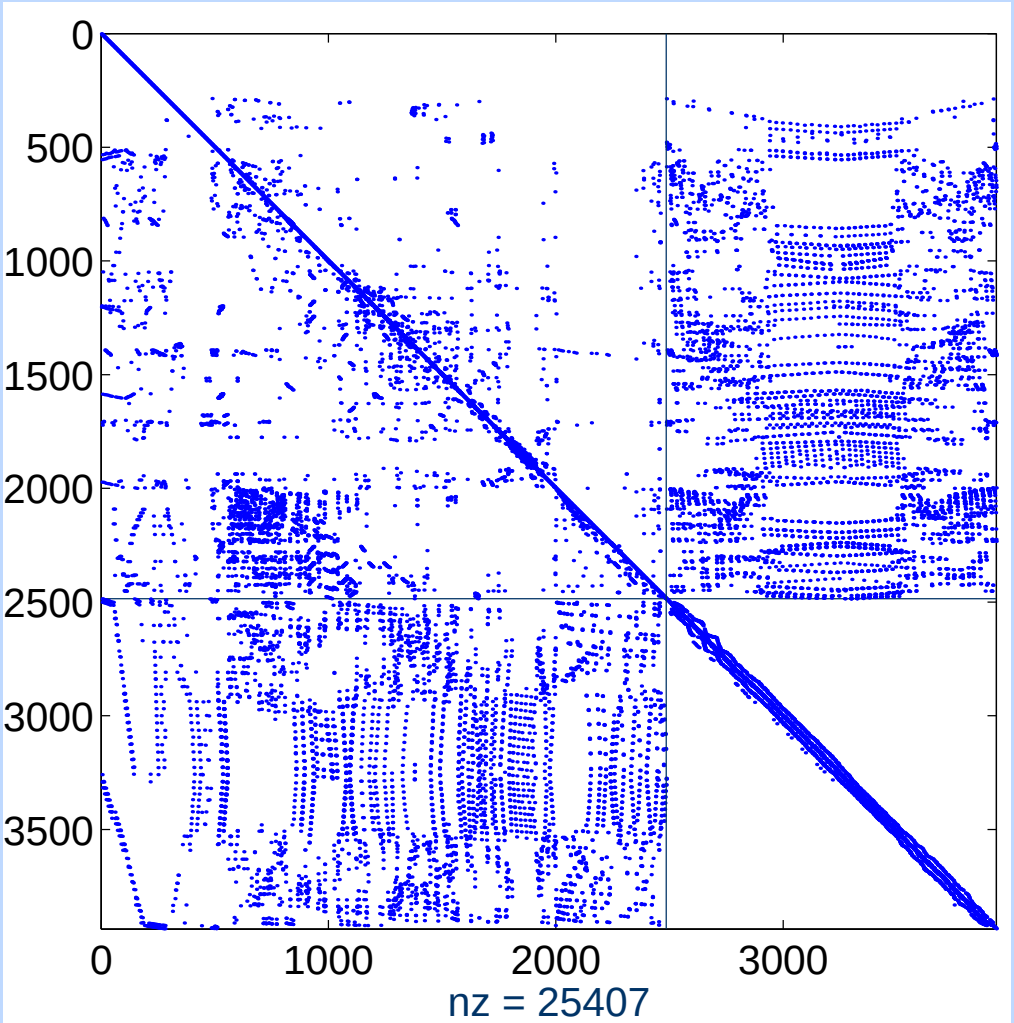
Initial System



Reordered by MC64 (improves diagonal dominance)

TEMPLATE 1. STATIC PARTIAL REORDERINGS

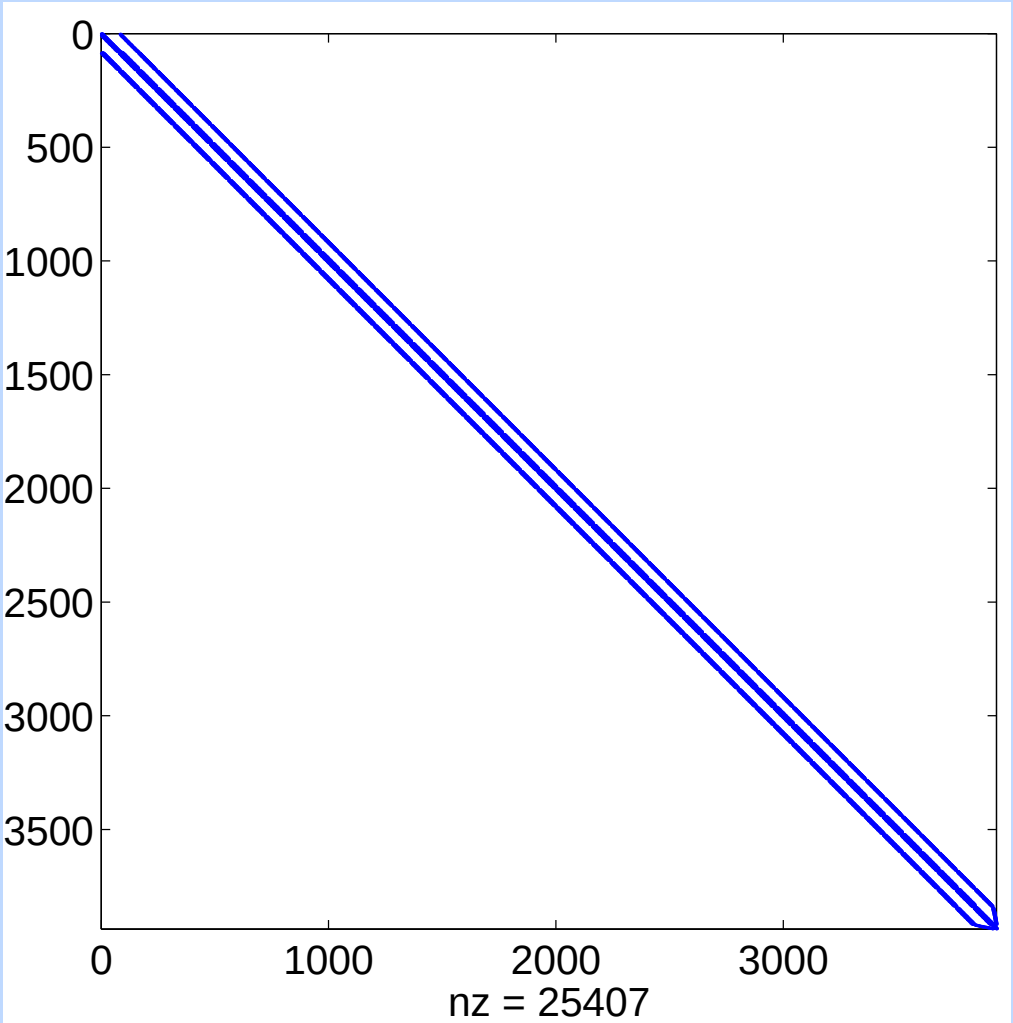


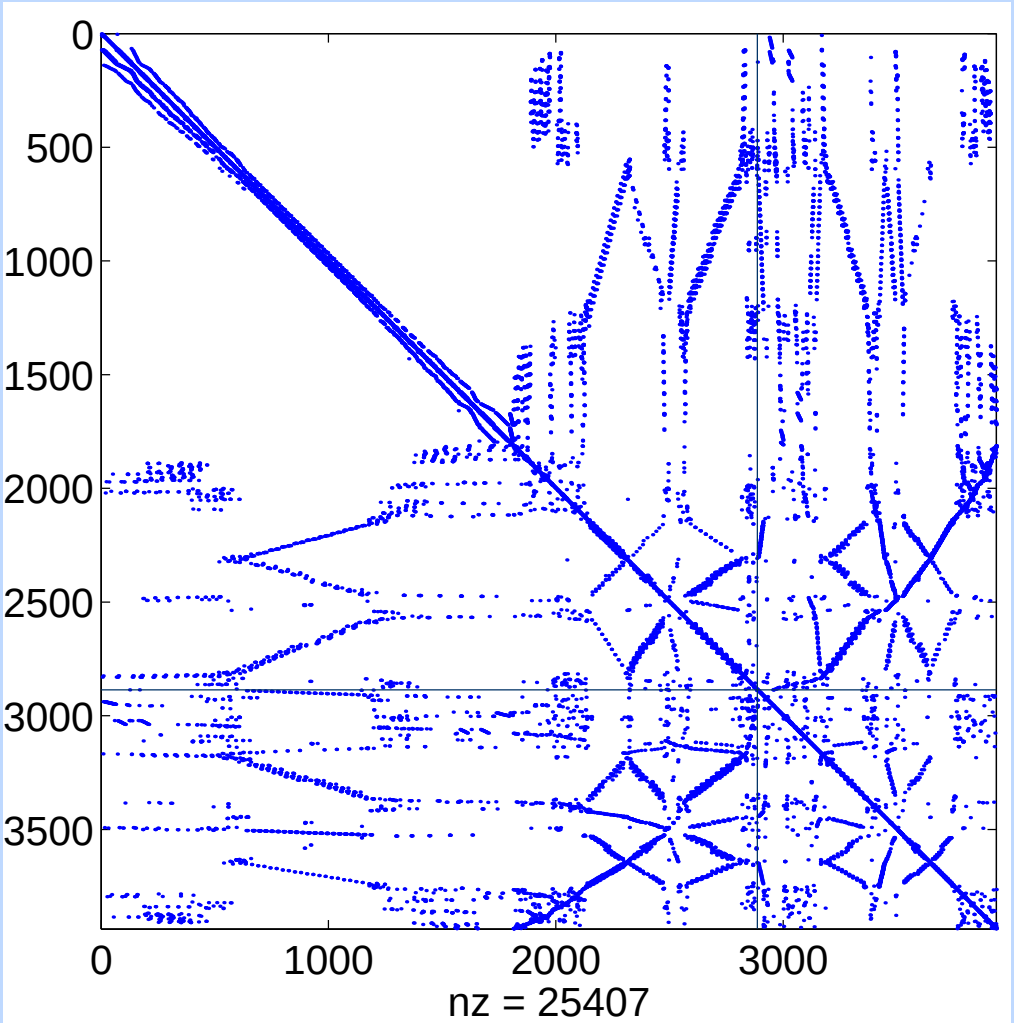


Initial System

Reordered ddPQ

TEMPLATE 1. STATIC PARTIAL REORDERINGS





Initial System

Reordered FC (fine/coarse grid)

Available Interfaces for Reorderings And Scalings

- ??permrcm, ??permmmd, ??permamf, ??permnd, ??permmc64, ??permpq, ??permindset, ??permfc, ??permpp, ??permnull.
  E.g. DGNLpermpq is associated with the ddPQ ordering.
- As nonsymmetric orderings, ??permpq and ??permmc64 are only available in the GNL case.
- Conversely ??permpp is a symmetric variant that is only available in the SPD,HPD cases.
- routines perform scaling and permutation in one step (starting with scaling)
- ILUPACK allows to include custom permutation routines.
- any permutation routines simply needs to follow the fixed parameter list

PROTOTYPES

```
perm...(mat A, FLOAT *prowscale, FLOAT *pcolscale,
        int *p, int *invq, int *nB, ILUPACKPARAM *param)
```

`mat A` is a matrix in sparse row format, A will be rescaled by `*prowscale, *pcolscale`!

FLOAT `*prowscale, *pcolscale` are vectors that return row/column scalings.
FLOAT may either be `float`, `double`, `complex` or `doublecomplex` (defined in `ilupack.h`).

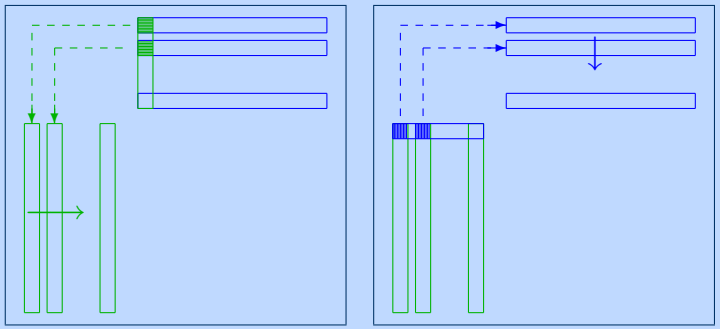
`int *p, *invq` provide row and inverse column permutation vectors (starting with 1 not with 0!!!)

`int *nB` returns the size of leading diagonal block B

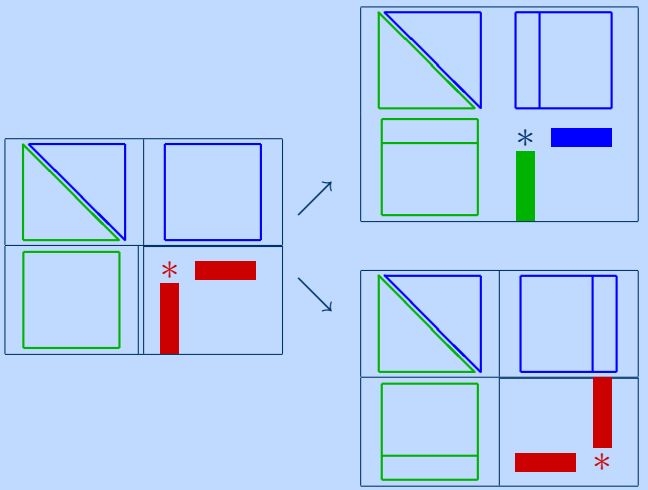
`ILUPACKPARAM *param` is a structure of parameter settings. It covers switches for scalings.

TEMPLATE 2. PILUC — INVERSE-BASED ILU THAT CONTROLS $\|L^{-1}\|, \|U^{-1}\|$

- ILU (Crout–version)



- $\|L_k^{-1}\|, \|U_k^{-1}\| \leqslant \kappa$ controlled by diagonal pivoting



- $\|L_k^{-1}\|$, $\|U_k^{-1}\|$ estimated [Cline, Moler, Stewart, Wilkinson '77], [B. '03]

PILUC AS PART OF ILUPACK

1. FORTRAN77 core routine, offers all–in–one solution, e.g.
   (a) partial incomplete LU factorization $L_B D_B U_B$
   (b) different versions of approximate Schur complement S_C
   (c) diagonal pivoting (factor or skip strategy)
   (d) estimates of the inverse factors
   (e) diagonal compensation

2. MPILUC, variant of PILUC. If

$$\hat{P}^\top A \hat{Q} = \begin{pmatrix} B & F \\ E & C \end{pmatrix} \approx \begin{pmatrix} L_B & 0 \\ L_E & I \end{pmatrix} \begin{pmatrix} D_B & 0 \\ 0 & S_C \end{pmatrix} \begin{pmatrix} U_B & U_F \\ 0 & I \end{pmatrix}$$

has been computed, MPILUC tries to continue with S_C without switching to a new level. Repeated multiple application of MPILUC leads to

$$\tilde{P}^\top A \tilde{Q} = \left(\begin{array}{cc|c} B & F_1 & F_2 \\ E_1 & C_{11} & C_{12} \\ \hline E_2 & C_{21} & C_{22} \end{array}\right) \approx \left(\begin{array}{cc|c} L_B & 0 & 0 \\ L_{E_1} & I & 0 \\ \hline L_{E_2} & L_{C_{21}} & I \end{array}\right) \left(\begin{array}{cc|c} D_B & 0 & 0 \\ 0 & D_{C_{11}} & 0 \\ \hline 0 & 0 & S_{22} \end{array}\right) \left(\begin{array}{cc|c} U_B & U_{F_1} & U_{F_2} \\ 0 & I & U_{C_{12}} \\ 0 & 0 & I \end{array}\right)$$

Template 3. Multilevel scheme

$$(\hat{P}^\top A \hat{Q})^{-1} = \begin{pmatrix} B & F \\ E & C \end{pmatrix}^{-1} \approx \begin{pmatrix} \tilde{B}^{-1} & 0 \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} -\tilde{B}^{-1}F \\ I \end{pmatrix} S_C{}^{-1} \begin{pmatrix} -E\tilde{B}^{-1} & I \end{pmatrix}, \text{ where } \tilde{B} = L_B D_B U_B$$

Recursive application to $S_C \longrightarrow$ AMG, concept of ARMS [Saad, Suchomel'99], i.e. skip L_E, U_F.

Approximate Schur complements

1. $S_C = C - L_E D_B U_F$
   use analogous approximation scheme for $B \approx L_B D_B U_B$ (S–version)

2. $S_C = \begin{bmatrix} -L_E L_B^{-1} & I \end{bmatrix} A \begin{bmatrix} -U_B^{-1} U_F \\ I \end{bmatrix}$
   use analogous approximation scheme for $B \approx L_B D_B U_B$ (T–version)

3. Use S–version for the computation of $B \approx L_B D_B U_B$ (M–version)
   Use T–version for the computation of S_C (default)

ILUPACK MAIN AMG DRIVER

- main factorization routine uses PILUC and MPILUC as templates and builds the multilevel structure around them.
- three permutation/scaling routines are supported (see previous slides)
- Three different versions of approximate Schur complements are supported (S–, M– and T–version). M–version is default.
- After the computation of the approximate Schur complement the parts L_E and U_F are discarded.
- if the fill exceeds a certain amount, the algorithm switches to full-matrix processing.
- if static final pivoting (3-rd permutation) fails, ILUTP with a small drop tolerance is taken (happens rarely in practice).

PROTOTYPES

```
AMGfactor(mat *A, AMGlevelmat *PRE, int *nlev,
          ILUPACKPARAM *param,
           int (*permp)(...), int (*permr)(...), int (*permf)(...))
```

`mat *A` is a source matrix in sparse row format, A will be rescaled!

`AMGlevelmat *PRE` is multilevel preconditioner in a linked list

`int *nlev` total number of levels

`ILUPACKPARAM *param` is a structure of parameter settings. It controls several parts of the preconditioner setup.

`int (*permp)(...), (*permr)(...), (*permf)(...)` three reordering/scaling routines for initial preprocessing, regular reordering and final pivoting.

Template 4. Iterative Solvers

- Multilevel ILU solver
- Iterative solvers GMRES, PCG
- Standard driver routine

Prototypes

```
AMGsol(AMGlevelmat PRE, int *nlev,
       FLOAT *rhs, FLOAT *sol, FLOAT *buff)
```

`AMGlevelmat *PRE` is multilevel preconditioner in a linked list

`int *nlev` total number of levels

`FLOAT *rhs, *sol, buff` right hand side, solution and buffer.

PROTOTYPES

```
    AMGsolver(mat A, AMGlevelmat PRE, int nlev,
              ILUPACKPARAM *param, FLOAT *sol, FLOAT *rhs)
```

`mat A` is a matrix in sparse row format, A already rescaled!

`AMGlevelmat PRE` is multilevel preconditioner in a linked list

`int nlev` total number of levels

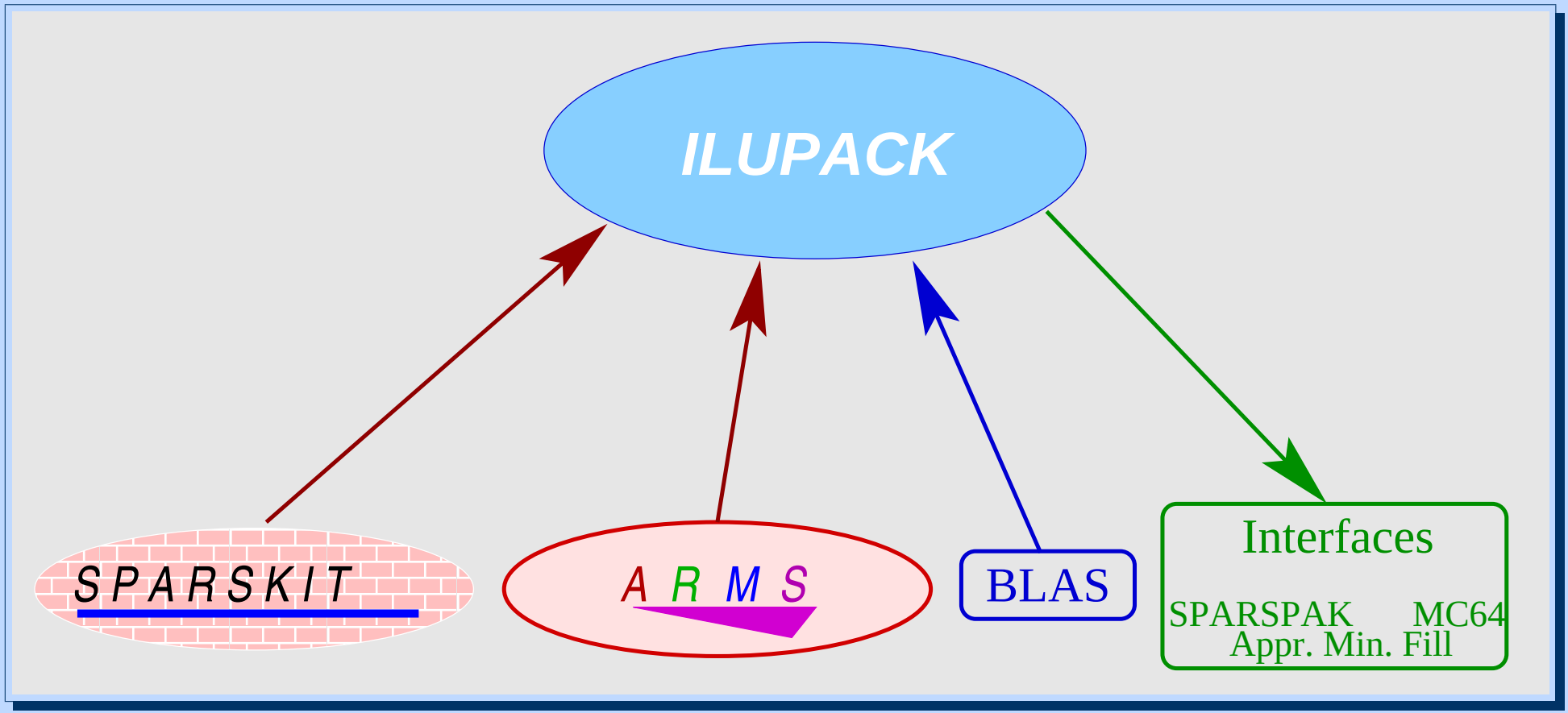
`ILUPACKPARAM *param` is a structure of parameter settings. It covers switches for scalings.

`FLOAT *sol, *rhs` solution and right hand side.

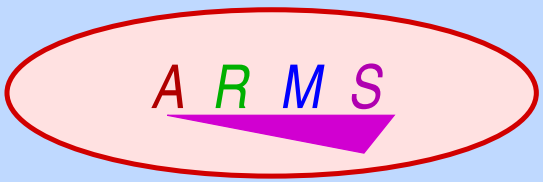
COMMENTS

- Default solvers are GMRES and PCG
- Initialization to standard settings supported by auxilliary routines

Preconditioning software package (C/FORTRAN 77) using inverse–based multilevel ILUs



- Iterative solvers GMRES, FGMRES
- PCG adapted from SPARSKIT–CG
- preconditioners ILUTP and ILUT, now based on a binary search tree
- preconditioners and solvers adapted with respect to complex arithmetic (and BLAS)
- Miscellaneous tools adapted (scaling, readmtc)



- Orderings indset and ddPQ
- Multilevel strategy (L_E, U_F discarded)
- Miscellaneous tools adapted (matvec, spartran)

BLAS

- Iterative solvers GMRES, FGMRES and PCG now based on BLAS
- Miscellaneous tools as well

Interfaces

SPARSPAK MC64
Appr. Min. Fill

- interfaces w.r.t. several reordering strategies provided
- MMD (multiple minimum degree), RCM (reverse Cuthill–McKee), ND (nested dissection) ⟵ SPARSPAK
- AMF (approximate minimum fill) by Patrick Amestoy
- MC64 (HSL, Harwell Subroutine Library)

Iterative Solvers

GMRES(30), PCG. Iteration stopped if $\|b - Ax^{(k)}\| \leqslant \sqrt{\text{eps}}\|b - Ax^{(0)}\|$

Iterative solver treated as failure, if a break down occured or more than 500 steps were needed.

Orderings

PQ (PP resp.), RCM, MMD, AMF $+$ row/column scaling

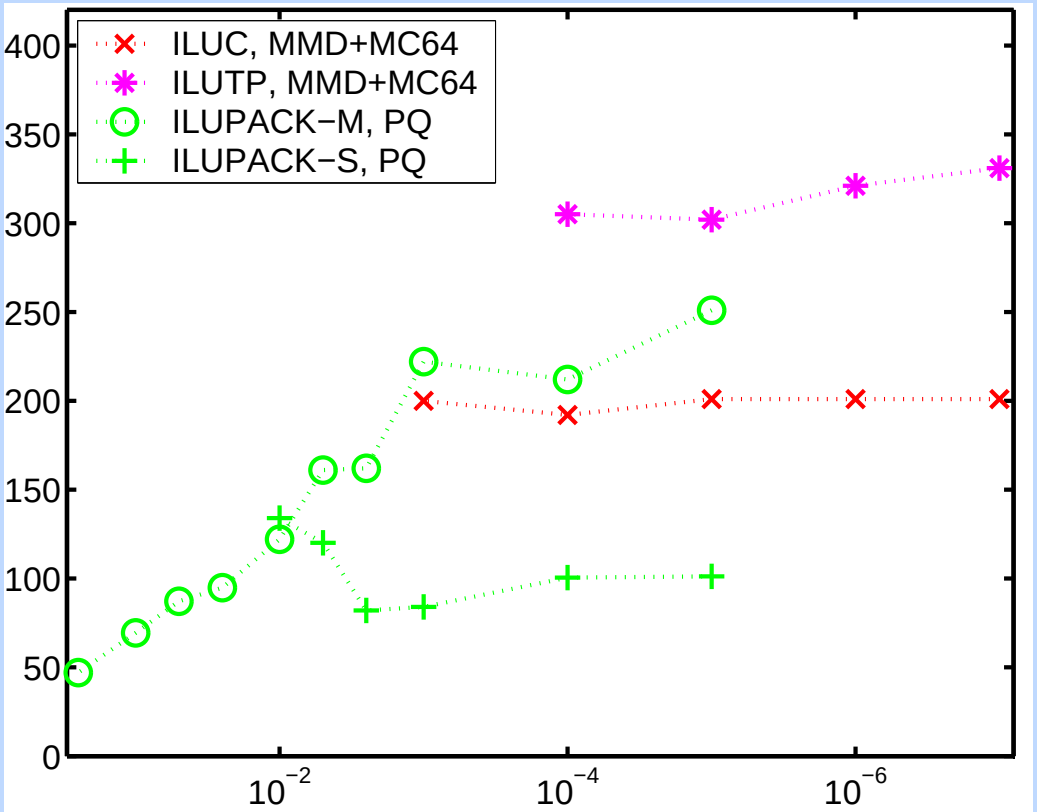
matrices also seperately preprocessed by MC64 and tested

Preconditioners

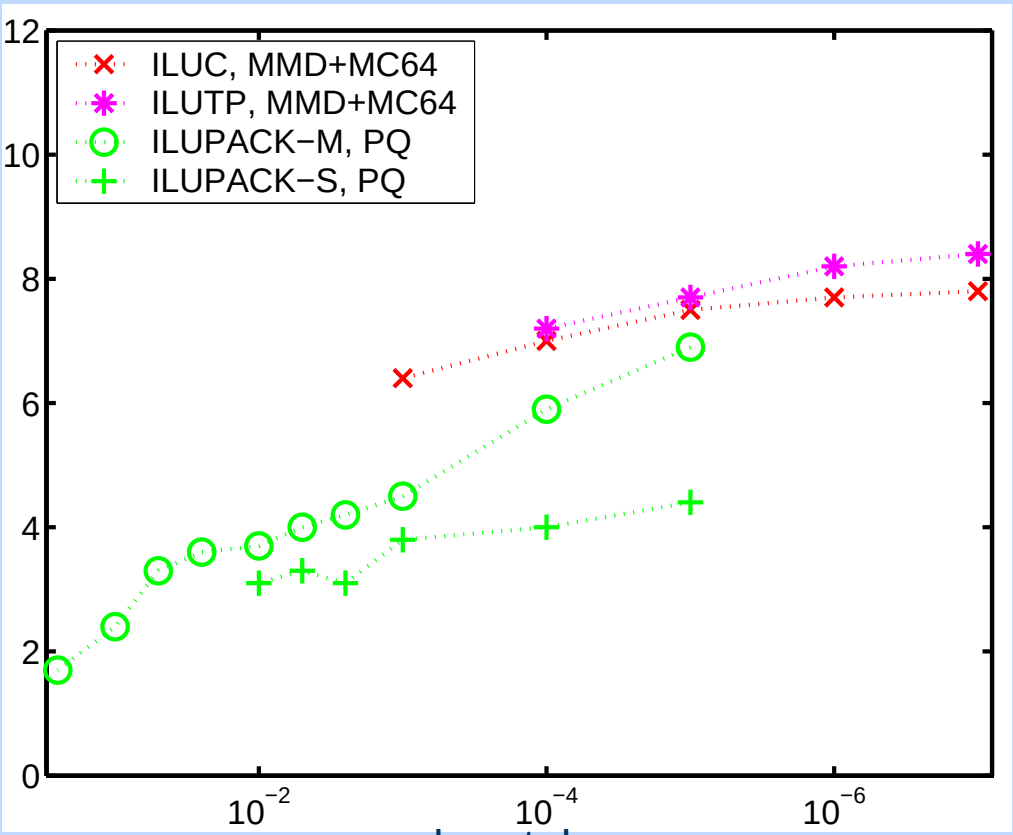
1. inverse–based multilevel ILU (referred as ILUPACK)
2. ILUTP
3. ILUC (single–level, inverse–based)

EXAMPLE. **Matrix ATT/TWOTONE**

Computation time

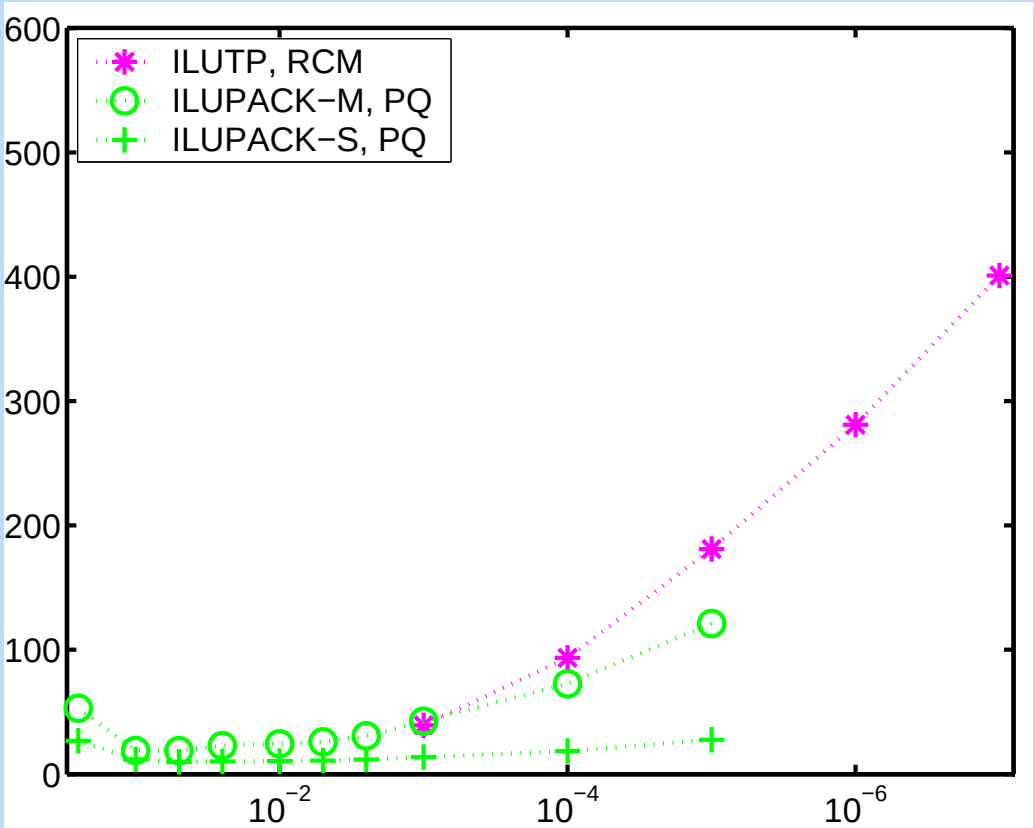


Memory requirement

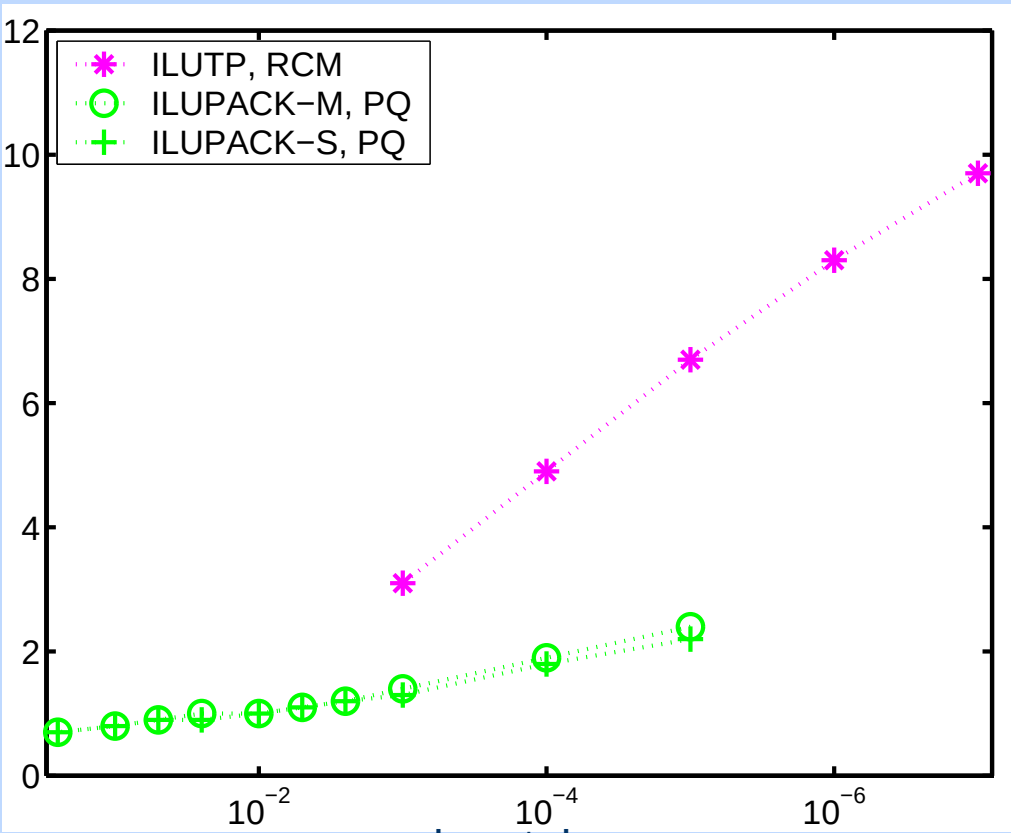


EXAMPLE. **Matrix VAVASIS/AV41092**

Computation time



Memory requirement

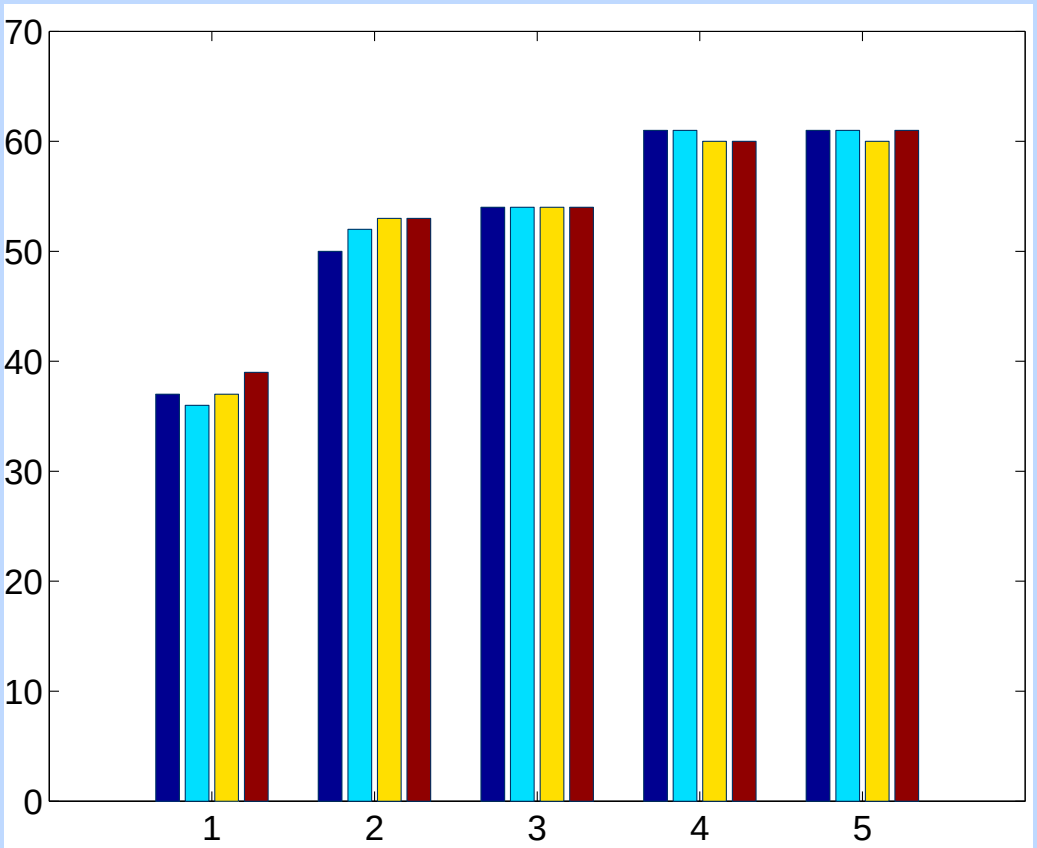


ILUPACK — dependence on inverse bound κ

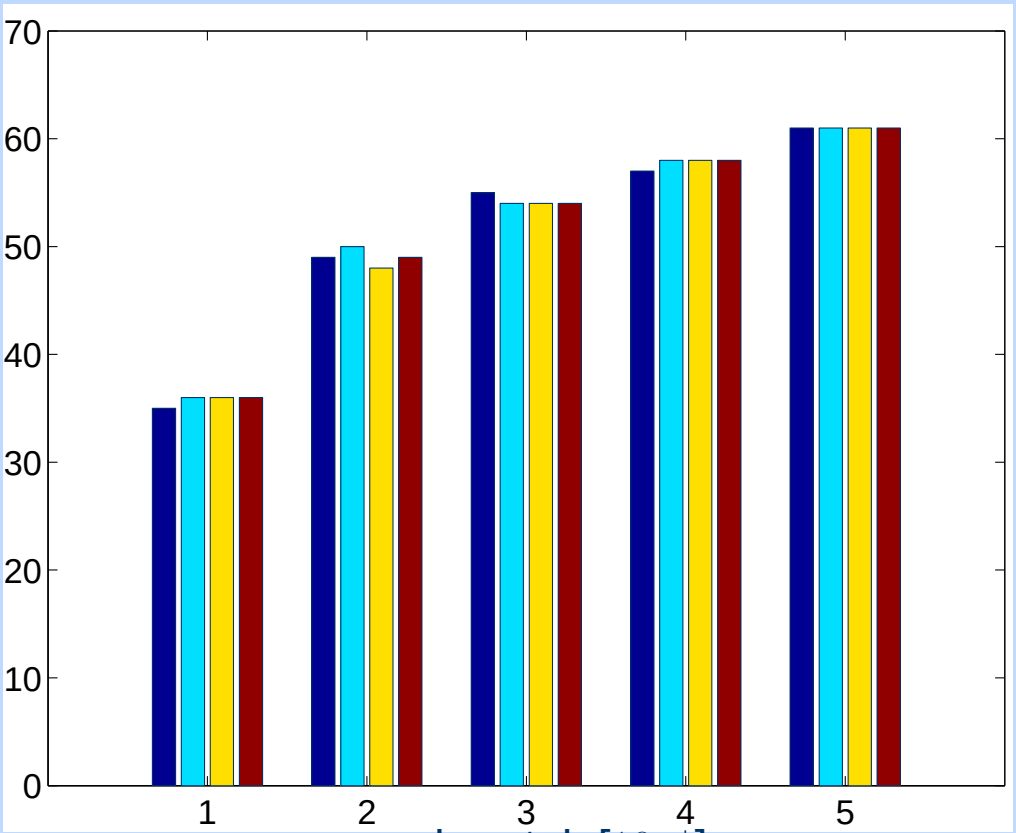
Almost no difference between $\kappa = 10$, $\kappa = 25$, $\kappa = 50$, $\kappa = 100$

64 LARGE TEST MATRICES, NUMBER OF SUCCESSFUL COMPUTATIONS

ILUPACK M–version



ILUPACK S–version

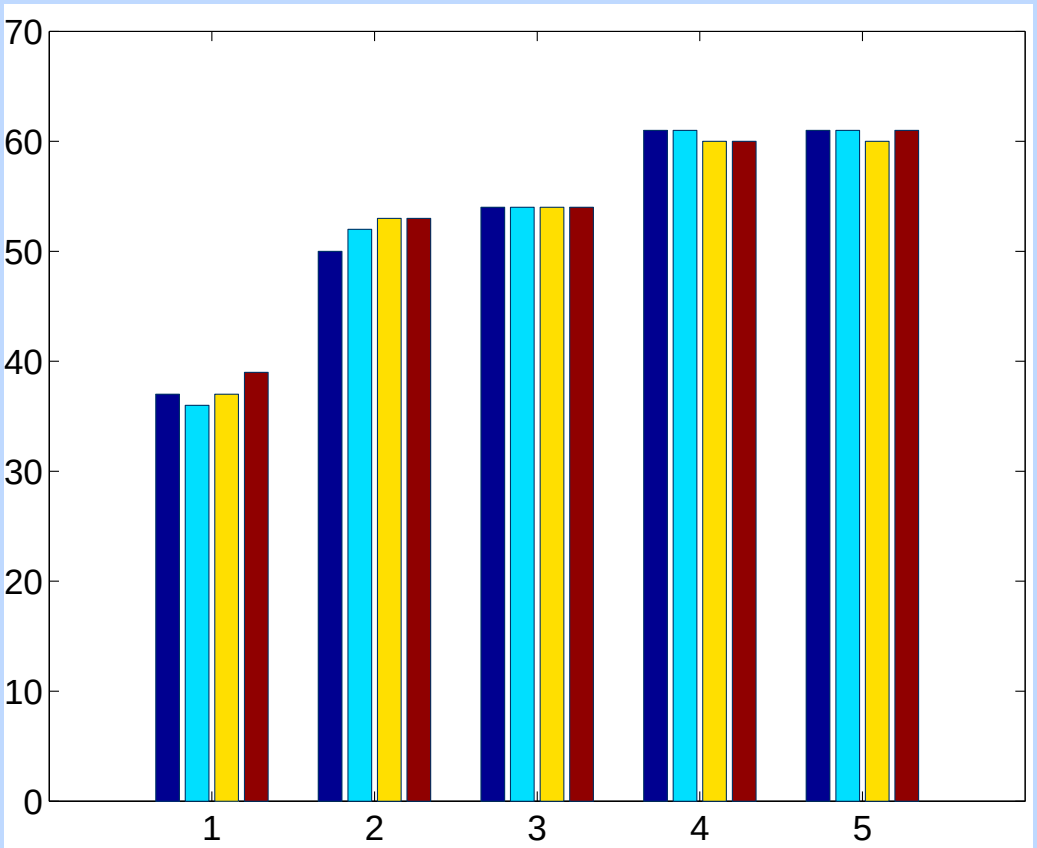


ILUPACK — dependence on inverse bound κ

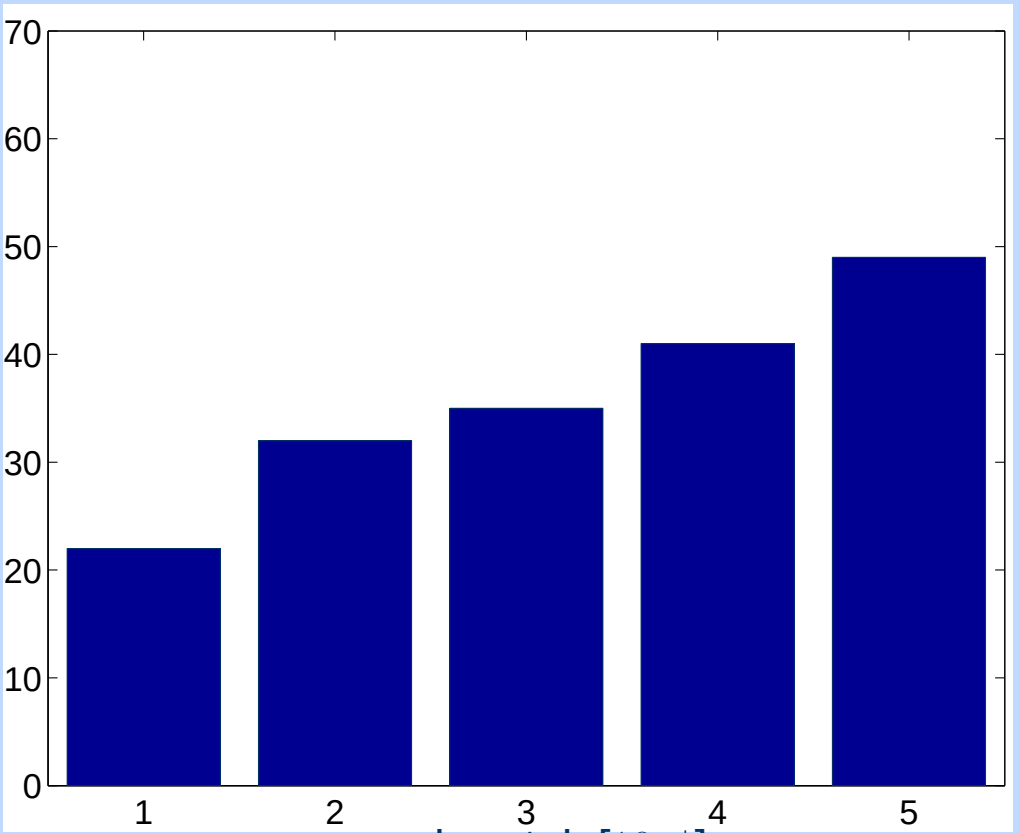
$\kappa = 10$, $\kappa = 25$, $\kappa = 50$, $\kappa = 100$

64 LARGE TEST MATRICES, NUMBER OF SUCCESSFUL COMPUTATIONS

ILUPACK M–version



ILUTP + RCM

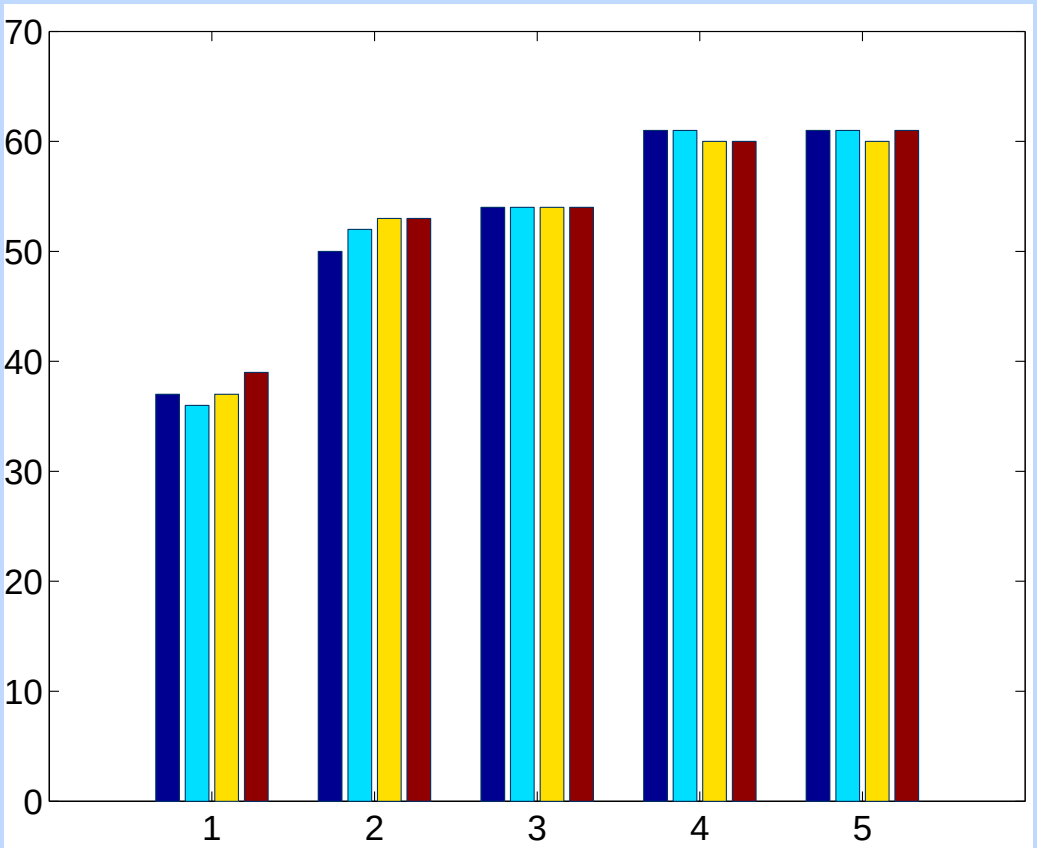


ILUPACK — dependence on inverse bound κ

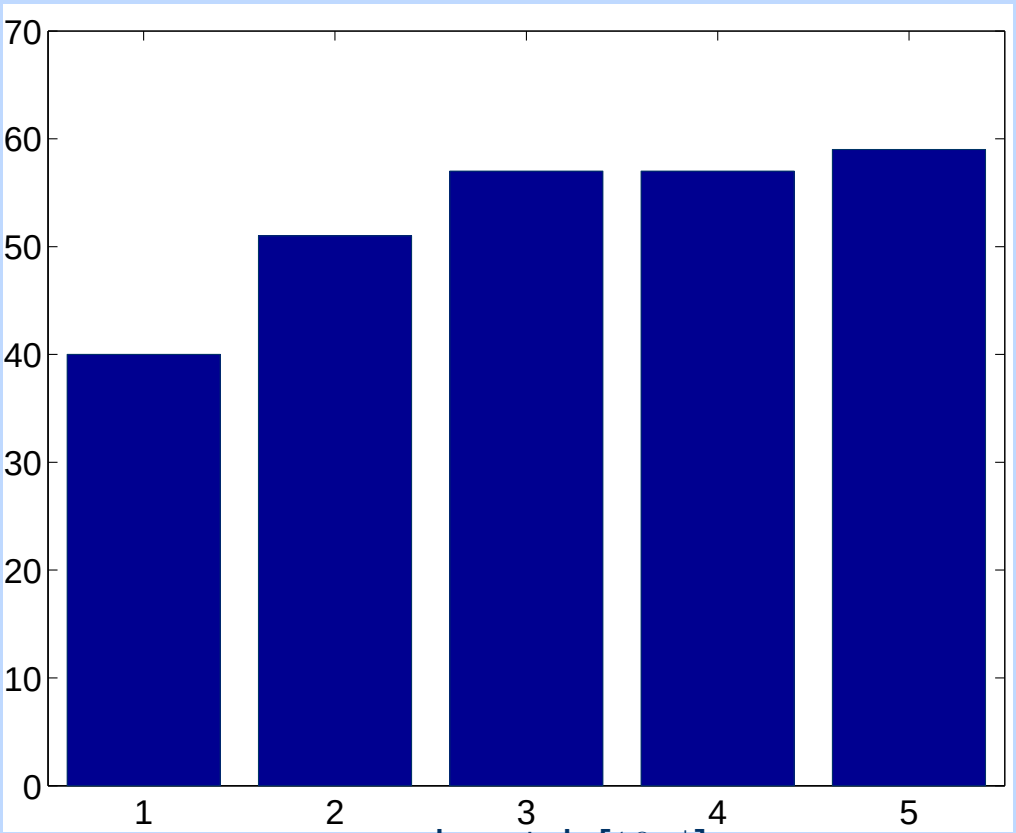
$\kappa = 10, \kappa = 25, \kappa = 50, \kappa = 100$

64 LARGE TEST MATRICES, NUMBER OF SUCCESSFUL COMPUTATIONS

ILUPACK M–version

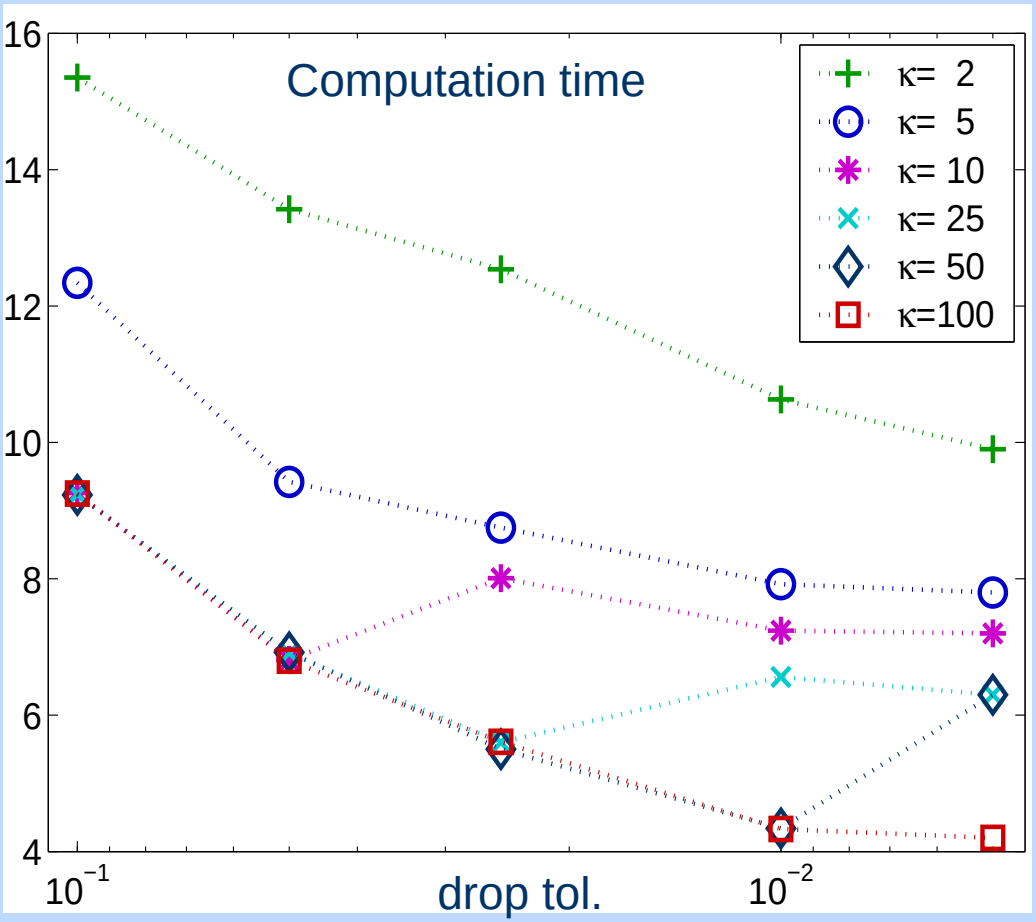


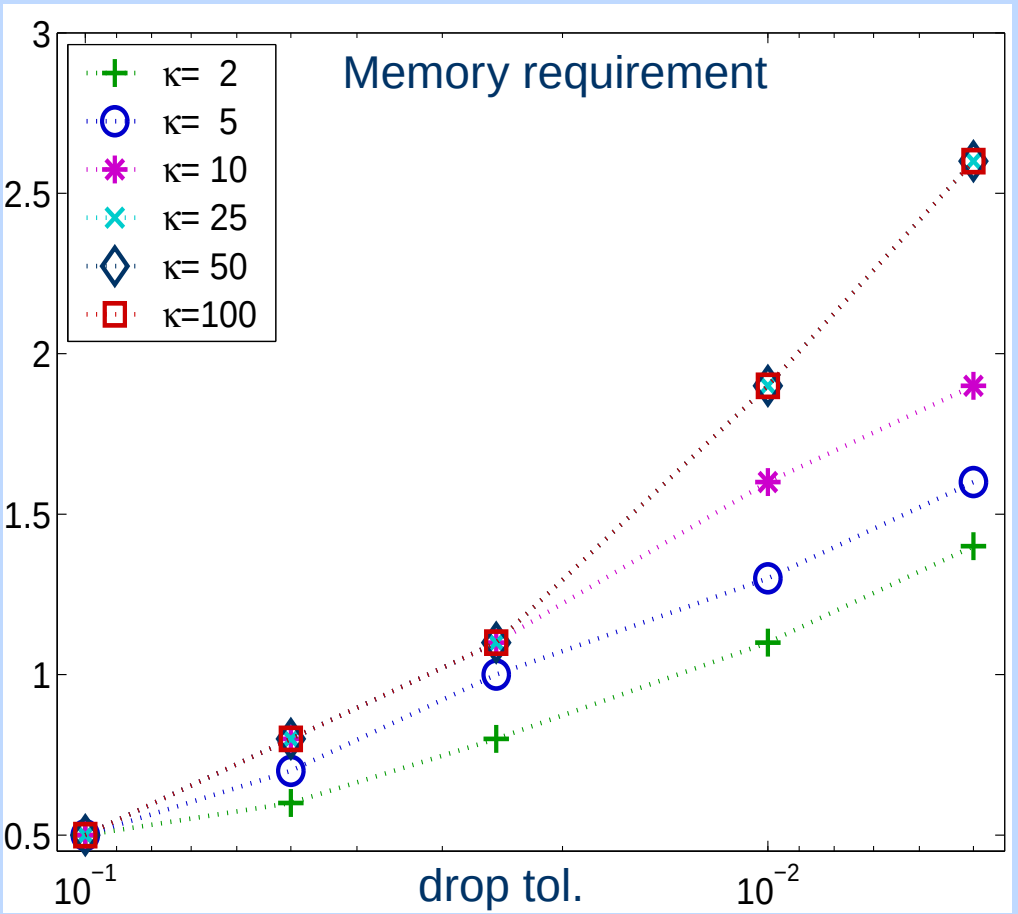
ILUTP + RCM + MC64



EXAMPLE. **cylindric shell problem**

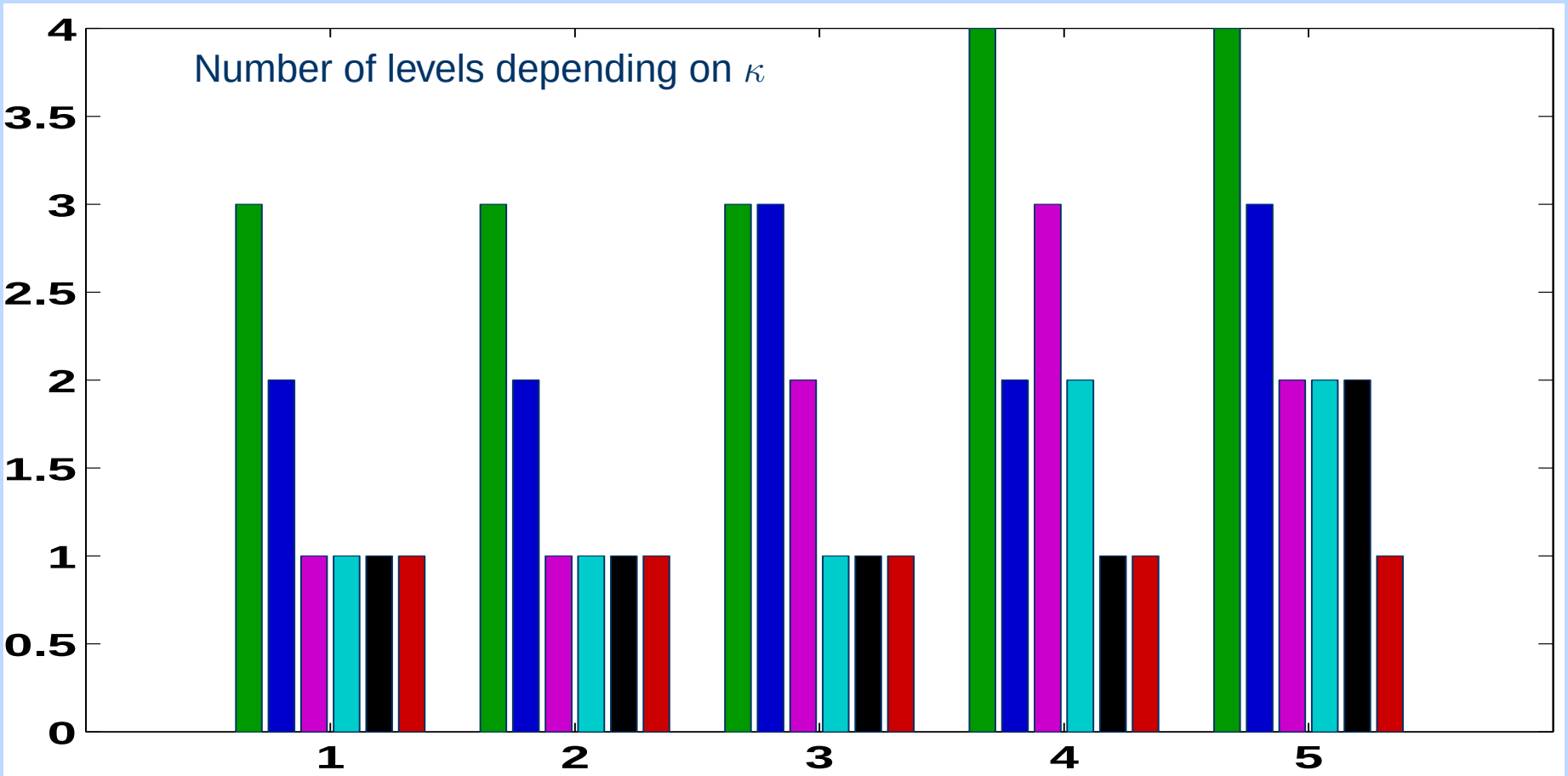
Tight bounds on $\|U^{-1}\| \leqslant \kappa$ implicitly yield FINE GRID/COARSE GRID selection



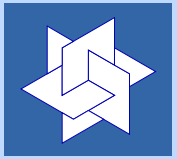


EXAMPLE. cylindric shell problem

Number of levels. $\kappa = 2$, $\kappa = 5$, $\kappa = 10$, $\kappa = 25$, $\kappa = 50$, $\kappa = 100$.



drop tol.	0.1	0.05	0.025	0.01	0.005



Conclusions

1. ILUPACK offers inverse–based multilevel ILU preconditioners for a broad class of matrices

2. Main objective: inverse triangular factors are kept bounded

3. various reorderings are incorporated or can easily be added

4. numerical experiments indicate robustness with respect to parameter tuning
   (time dependent problems, nonlinear equations)

ILUPACK BEING RELEASED BY APRIL/MAY, 2004

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