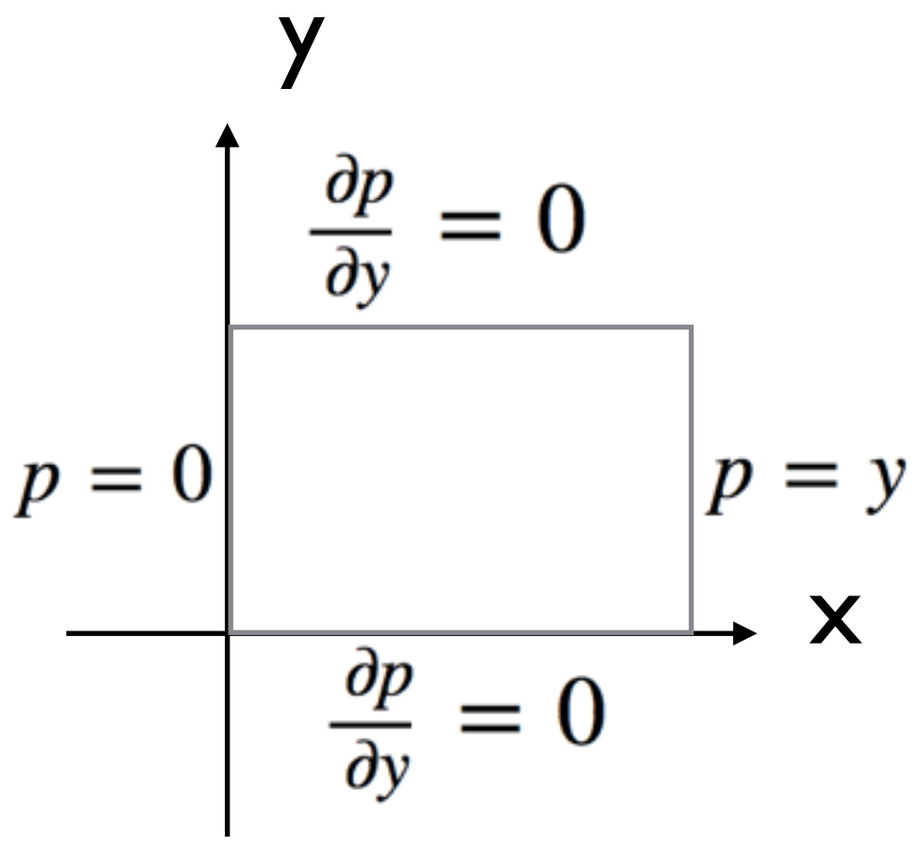
05/09	Class 9	Dense direct solvers	Understand the principle of LU decomposition and the optimization and parallelization techniques that lead to the LINPACK benchmark.
05/12	Class 10	Dense eigensolvers	Determine eigenvalues and eigenvectors and understand the fast algorithms for diagonalization and orthonormalization.
05/16	Class 11	Sparse direct solvers	Understand reordering in AMD and nested dissection, and fast algorithms such as skyline and multifrontal methods.
05/19	Class 12	Sparse iterative solvers	Understand the notion of positive definiteness, condition number, and the difference between Jacobi, CG, and GMRES.
05/23	Class 13	Preconditioners	Understand how preconditioning affects the condition number and spectral radius, and how that affects the CG method.
05/26	Class 14	Multigrid methods	Understand the role of smoothers, restriction, and prolongation in the V-cycle.
05/30	Class 15	Fast multipole methods, H-matrices	Understand the concept of multipole expansion and low-rank approximation, and the role of the tree structure.

2-D Laplace equation

$$\frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2} = 0$$

$$p_{i,j}^n = \frac{\Delta y^2 (p_{i+1,j}^n + p_{i-1,j}^n) + \Delta x^2 (p_{i,j+1}^n + p_{i,j-1}^n)}{2(\Delta x^2 + \Delta y^2)}$$



A=

$$\begin{bmatrix}
4 & -1 & & -1 & & & & & \\
-1 & 4 & -1 & & -1 & & & & \\
 & -1 & 4 & & & -1 & & & \\
-1 & & & 4 & -1 & & -1 & & \\
 & -1 & & -1 & 4 & -1 & & -1 & \\
 & & -1 & & -1 & 4 & & & -1 \\
 & & & -1 & & & 4 & -1 & \\
 & & & & -1 & & -1 & 4 & -1 \\
 & & & & & -1 & & -1 & 4
\end{bmatrix}$$

Ax=b

Sparse linear algebra

Iterative methods ⟶ Easier to parallelize, optimize

- Stationary methods

  Jacobi, Gauss-Seidel, SOR, SSOR, ADI

- Non-stationary methods (Krylov subspace methods)

  CG, BiCG, BiCGSTAB, GMRES, CGR

Direct methods ⟶ Robust

- Approximate Minimum Degree
- Supernodal
- Nested Dissection

Sparse direct solvers

scipy.sparse.linalg.spsolve

scipy.sparse.linalg.spsolve(*A*, *b*, *permc_spec=None*, *use_umfpack=True*) [source]

Solve the sparse linear system Ax=b, where b may be a vector or a matrix.

Parameters:

A *: ndarray or sparse matrix*
The square matrix A will be converted into CSC or CSR form

b *: ndarray or sparse matrix*
The matrix or vector representing the right hand side of the equation. If a vector, b.size must

permc_spec *: str, optional*
How to permute the columns of the matrix for sparsity preservation. (default: 'COLAMD')

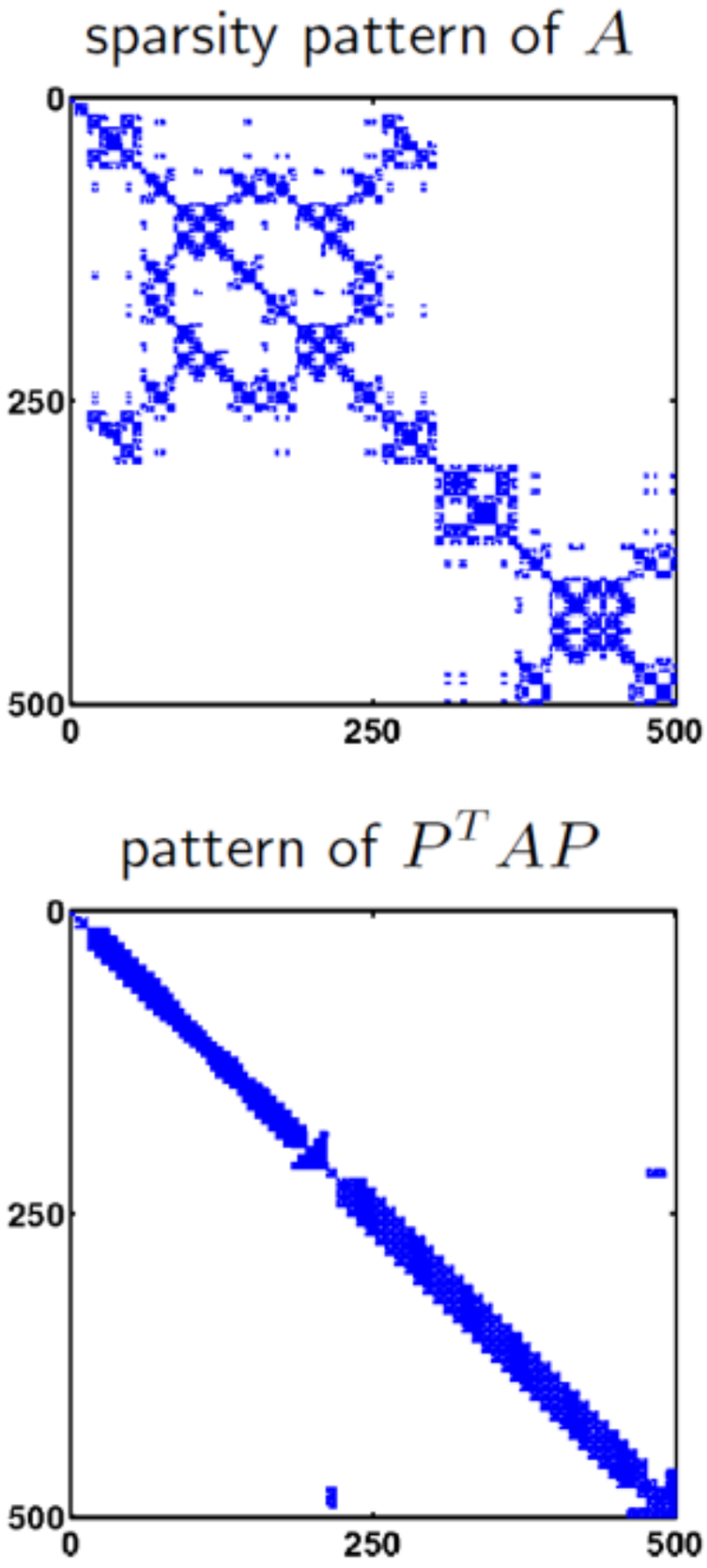
- `NATURAL`: natural ordering.
- `MMD_ATA`: minimum degree ordering on the structure of A^T A.
- `MMD_AT_PLUS_A`: minimum degree ordering on the structure of A^T+A.
- `COLAMD`: approximate minimum degree column ordering

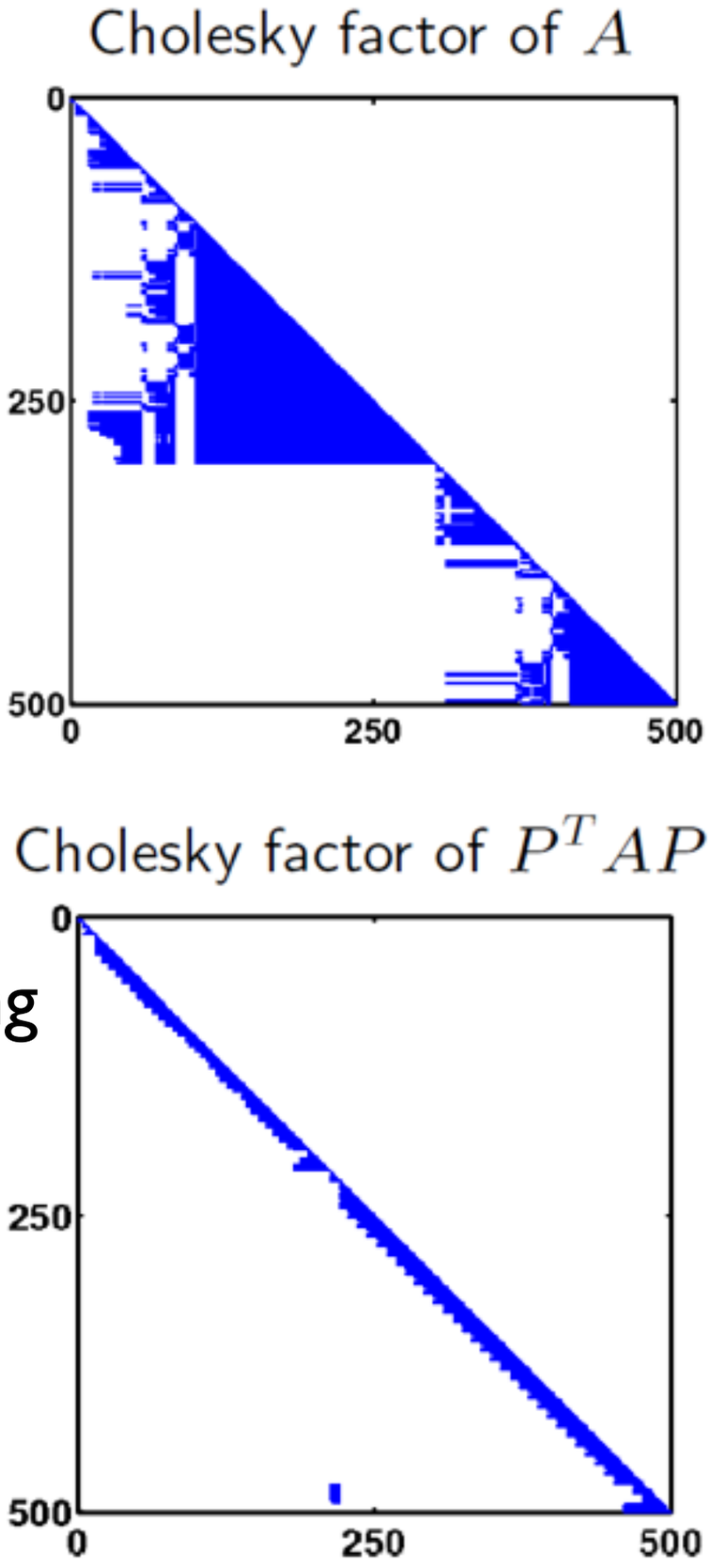
use_umfpack *: bool (optional)*
if True (default) then use umfpack for the solution. This is only referenced if b is a vector.

Returns:

x *: ndarray or sparse matrix*
the solution of the sparse linear equation. If b is a vector, then x is a vector of size A.shape[1] If b is a matrix, then x is a matrix of size (A.shape[1], b.shape[1])

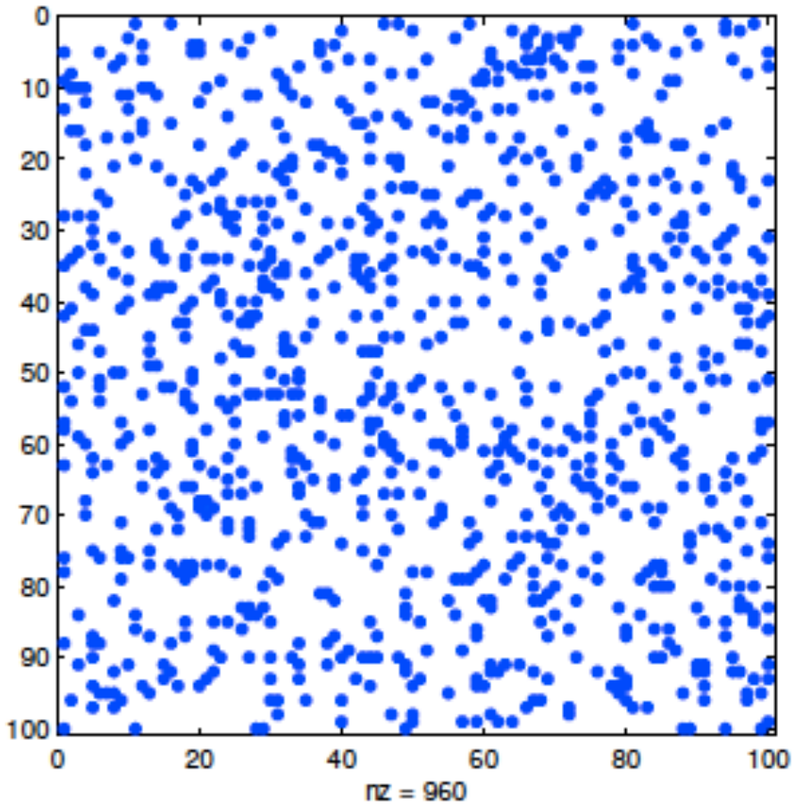
Factorization of a Sparse Matrix

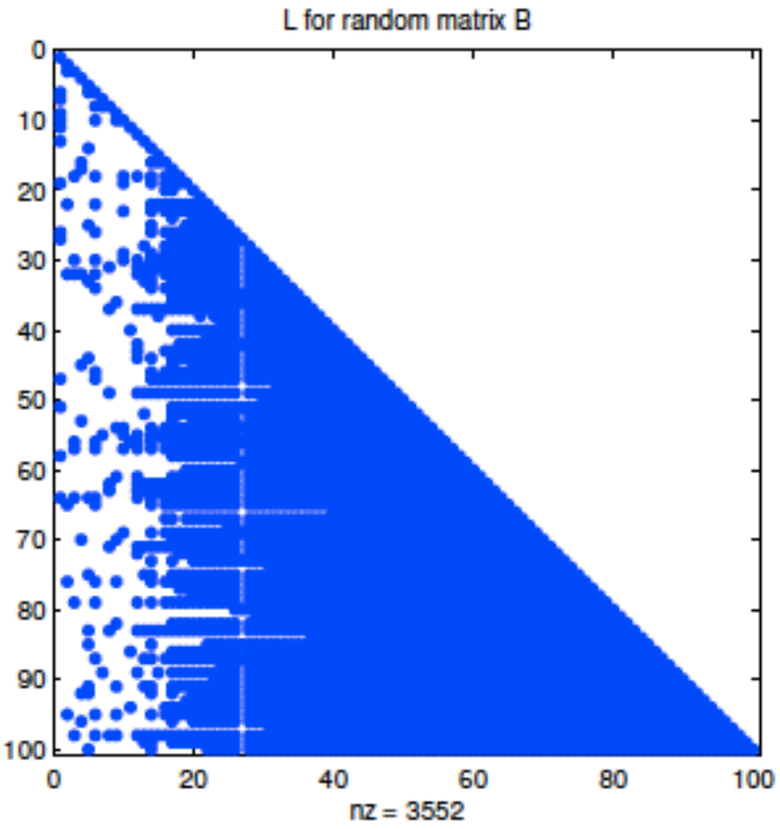




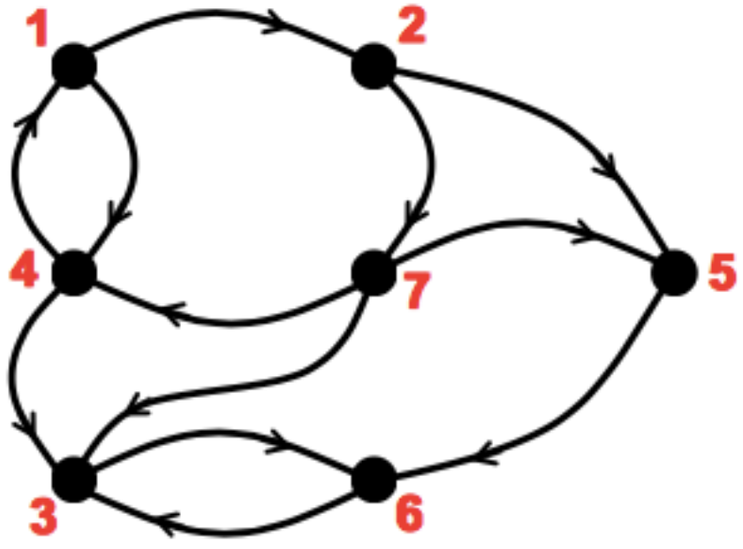
Reordering

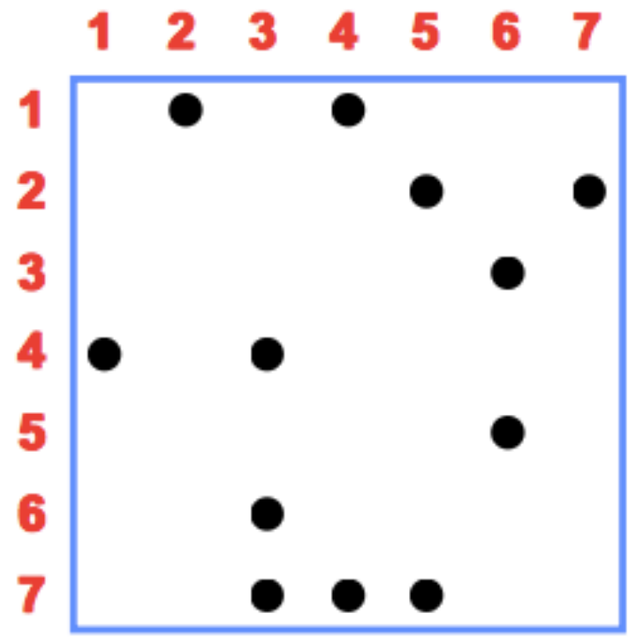
Factorization of a Random Sparse Matrix

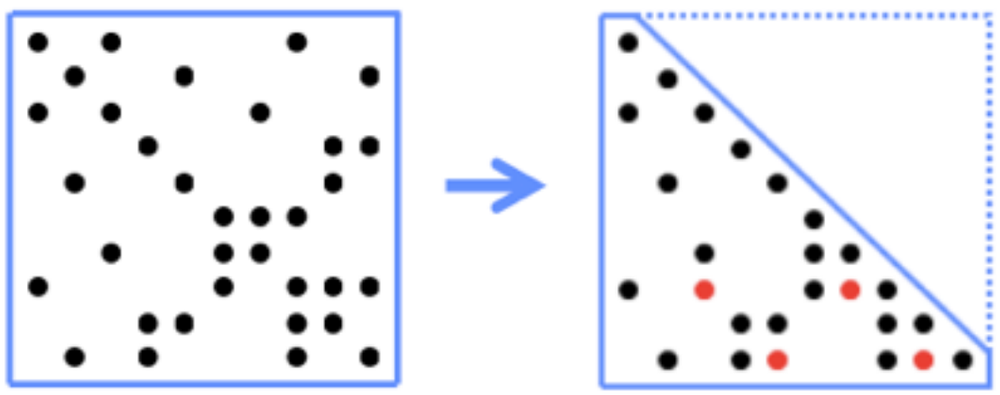




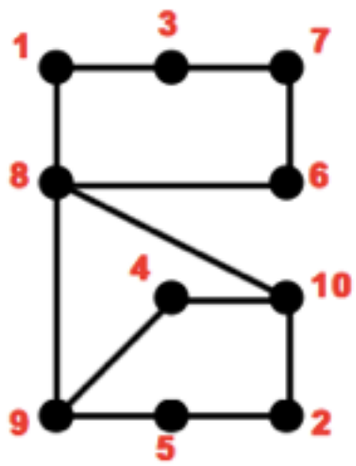
Graph Representation of a Sparse Matrix

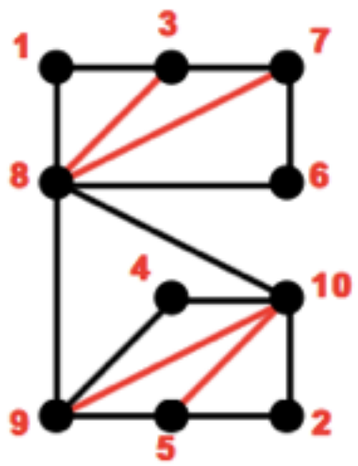






Fill in

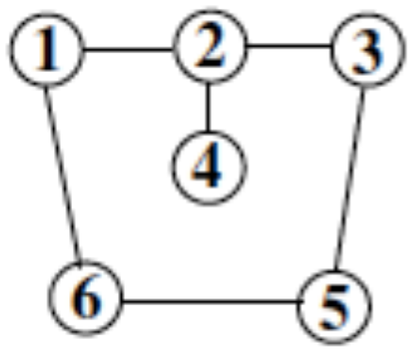




New edge

Elimination Graph

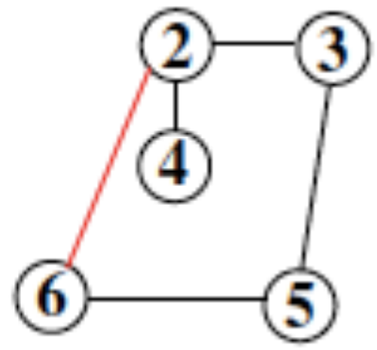
G0 :



H0 =

$$\begin{bmatrix} 1 & \times & & & & \times \\ \times & 2 & \times & \times & & \\ & \times & 3 & & \times & \\ & \times & & 4 & & \\ & & \times & & 5 & \times \\ \times & & & & \times & 6 \end{bmatrix}$$

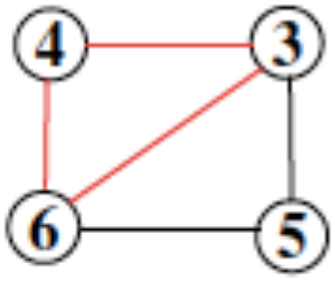
G1 :



H1 =

$$\begin{bmatrix} 2 & \times & \times & & + \\ \times & 3 & & \times & \\ \times & & 4 & & \\ & \times & & 5 & \times \\ + & & & \times & 6 \end{bmatrix}$$

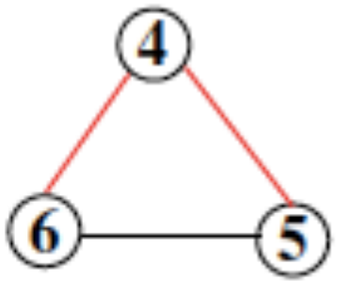
G2 :



H2 =

$$\begin{bmatrix} 3 & + & \times & + \\ + & 4 & & + \\ \times & & 5 & \times \\ + & + & \times & 6 \end{bmatrix}$$

G3 :

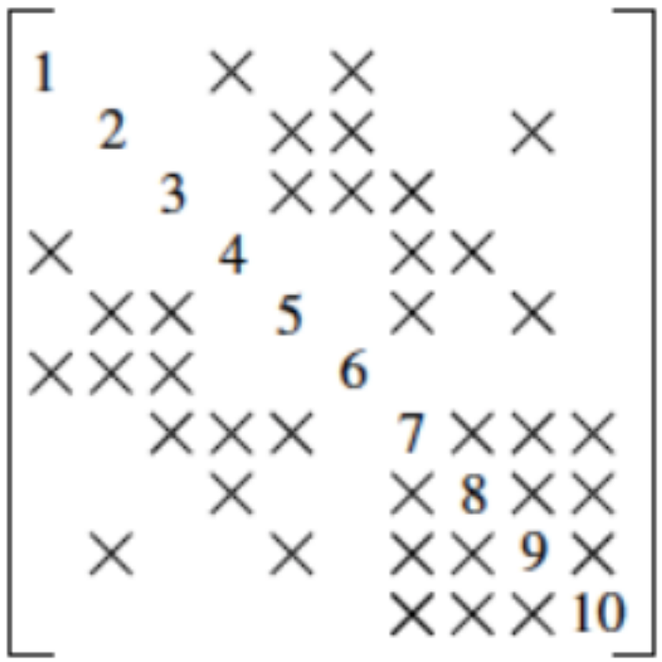


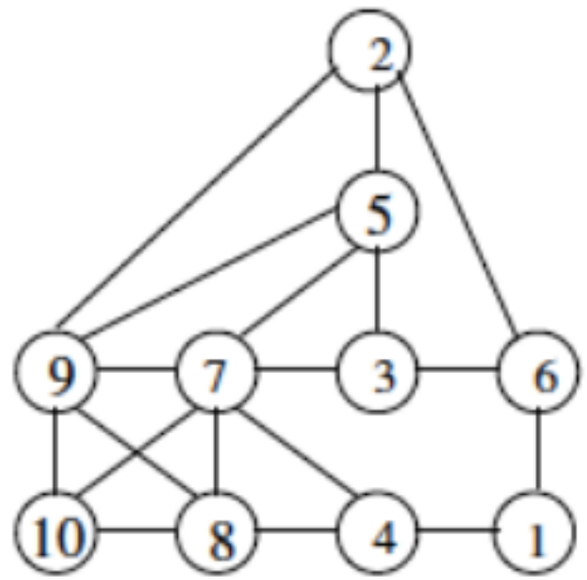
H3 =

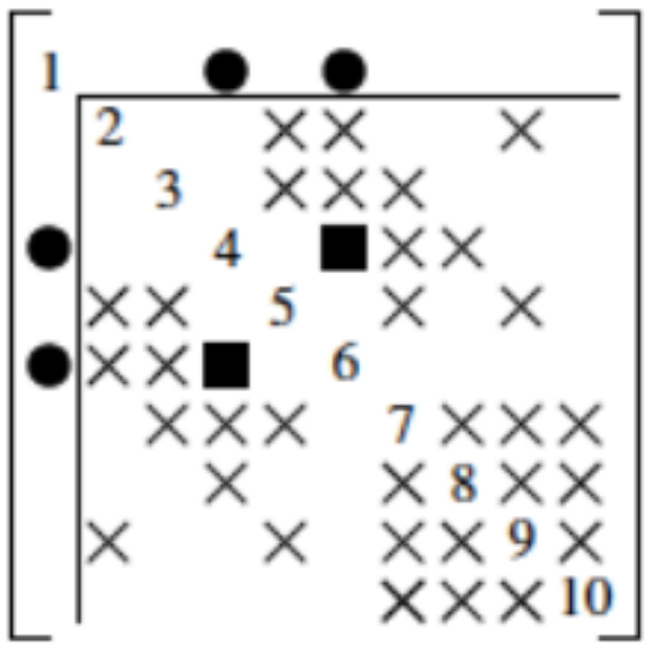
$$\begin{bmatrix} 4 & + & + \\ + & 5 & \times \\ + & \times & 6 \end{bmatrix}$$

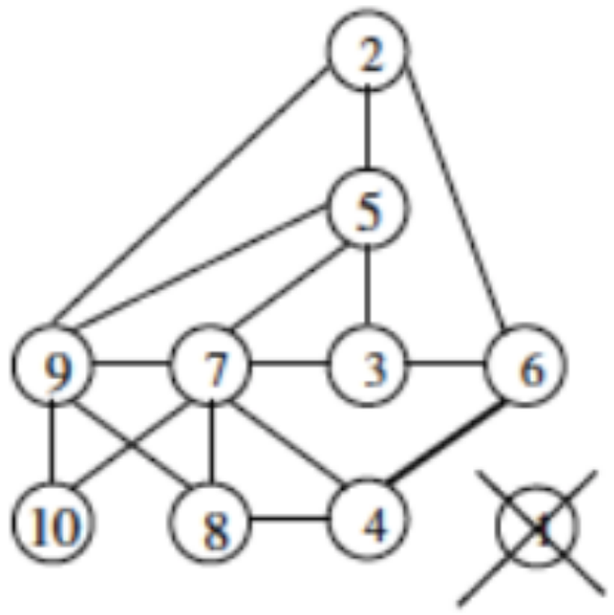
Minimum degree ordering

Select the vertex that possesses the smallest number of neighbors in G^0

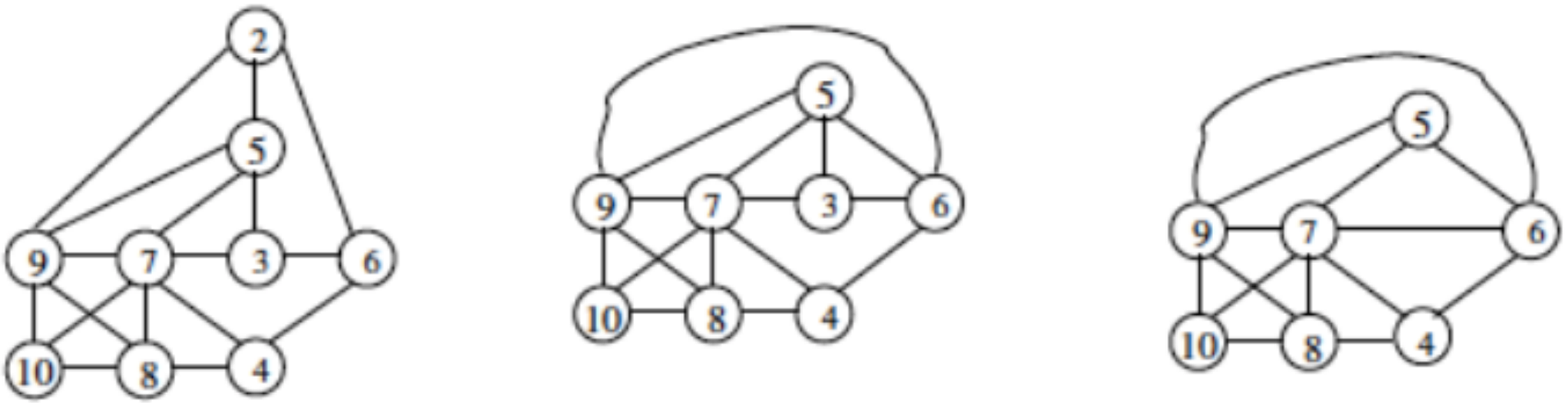




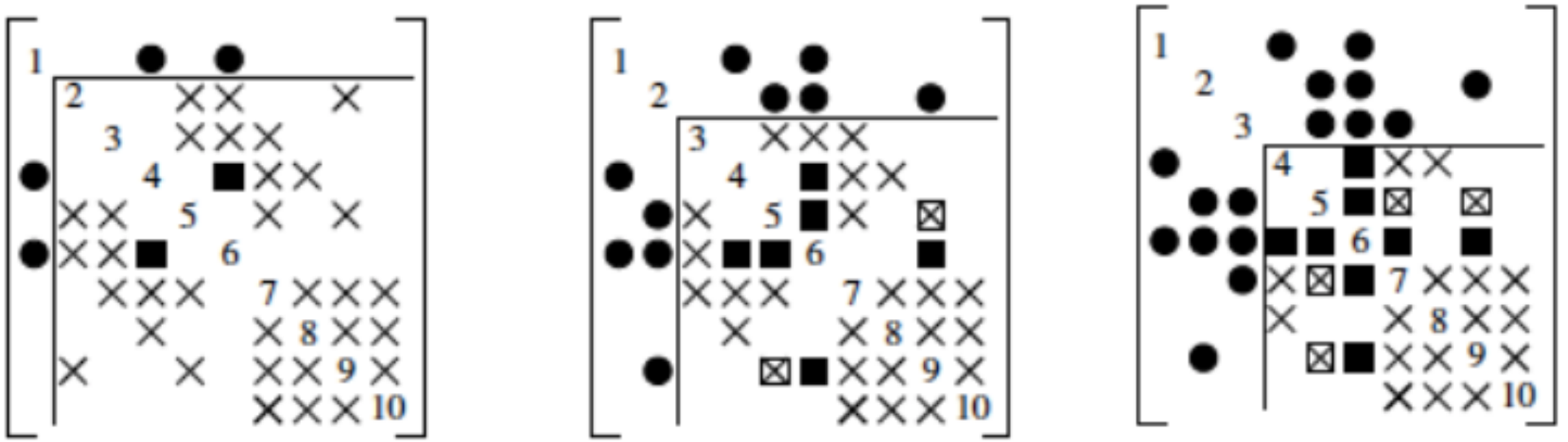




Minimum degree ordering



(a) Elimination graphs



(b) Factors and active submatrices

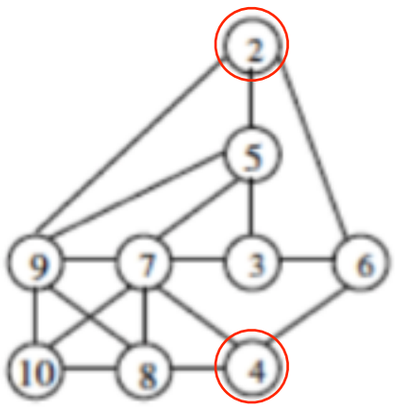
× **Original nonzero**

⊠ **Original nonzero modified**

■ **Fill–in**

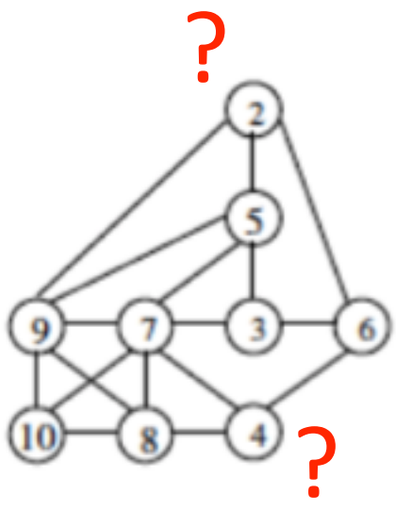
● **Nonzeros in factors**

Multiple Minimum Degree



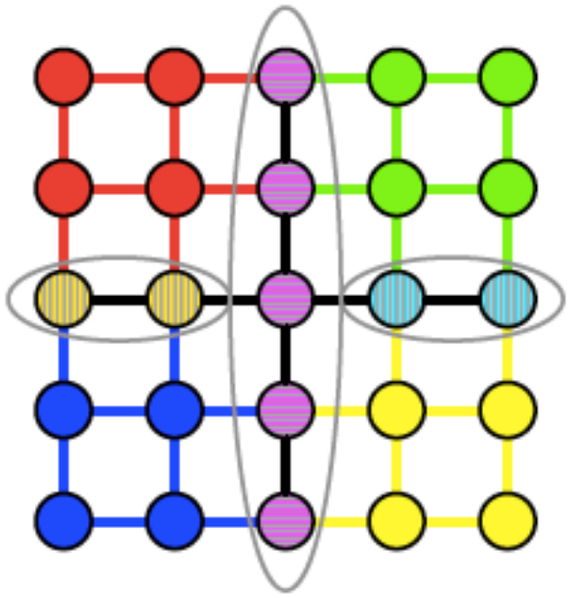
Choose the nodes that are not neighbors

Approximate Minimum Degree

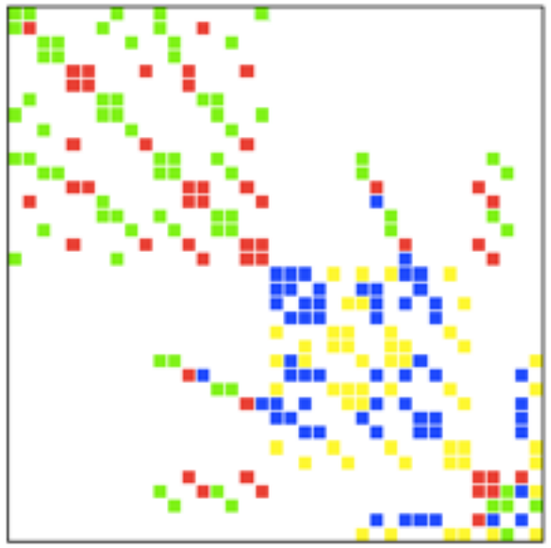


Only calculates the approximate minimum degree

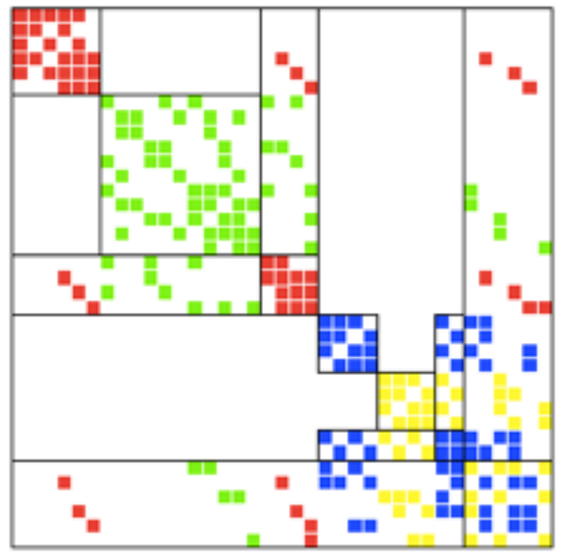
Nested dissection ordering



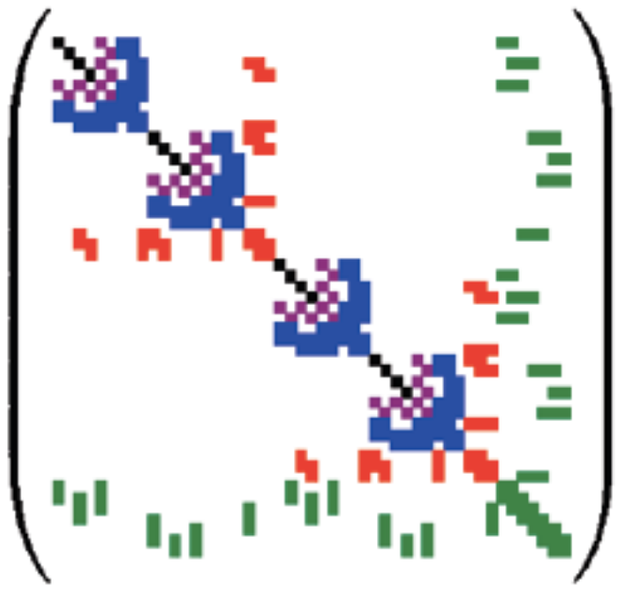
A



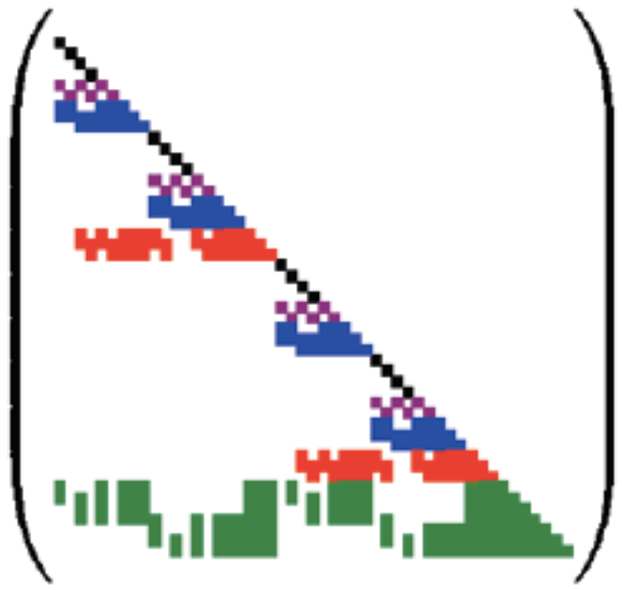
reordered A



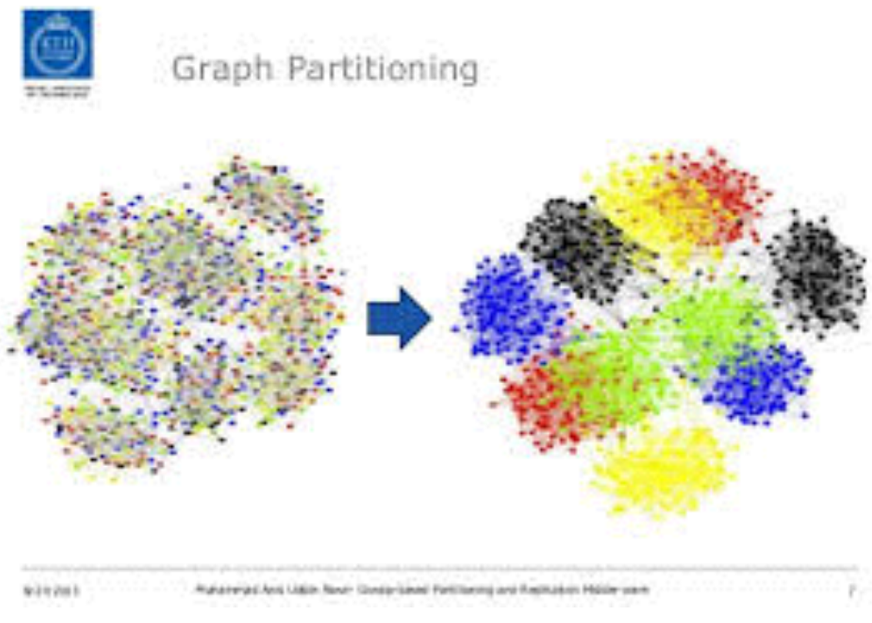
reordered A

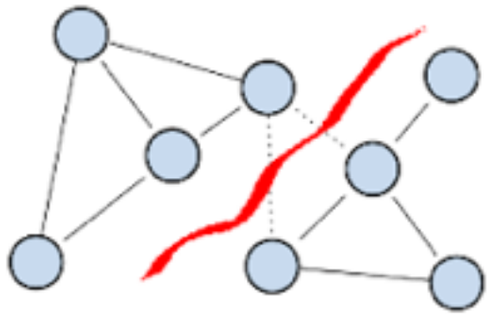


L



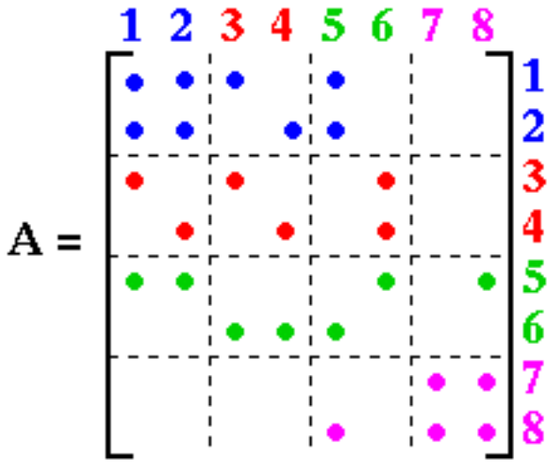
Graph partitioning

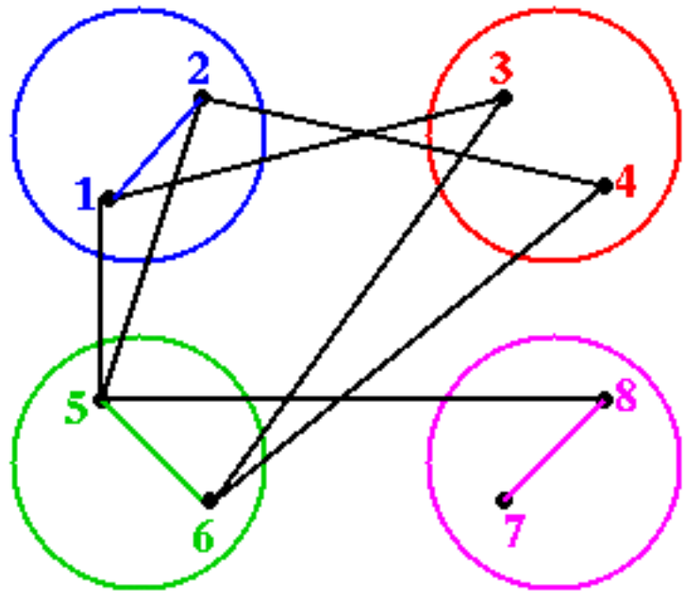




METIS

Partitioning a Sparse Symmetric Matrix





Sparse data format

Sparse matrices (scipy.sparse)

SciPy 2-D sparse matrix package for numeric data.

Contents¶

Sparse matrix classes

bsr_matrix(arg1[, shape, dtype, copy, blocksize])	Block Sparse Row matrix
coo_matrix(arg1[, shape, dtype, copy])	A sparse matrix in COOrdinate format.
csc_matrix(arg1[, shape, dtype, copy])	Compressed Sparse Column matrix
csr_matrix(arg1[, shape, dtype, copy])	Compressed Sparse Row matrix
dia_matrix(arg1[, shape, dtype, copy])	Sparse matrix with DIAgonal storage
dok_matrix(arg1[, shape, dtype, copy])	Dictionary Of Keys based sparse matrix.
lil_matrix(arg1[, shape, dtype, copy])	Row-based linked list sparse matrix
spmatrix([maxprint])	This class provides a base class for all sparse matrices.

Benchmarking

Code	Ordering options								Factorization Algorithm
	MD	AMD	MMD	ND	METIS	MS	MF	User	
BCSLIB-EXT	×	×	√	×	√*	×	×	√	Multifrontal
MA57	√	√*	×	×	√*	×	×	√	Multifrontal
MUMPS	√	√*	×	×	√*	√	√	√	Multifrontal
Oblio	×	×	√	×	√*	×	×	√	Left-looking, right-looking, multifrontal
PARDISO	×	×	√	×	√*	×	×	√	Left-right looking
SPOOLES	×	×	√	√*	×	√*	×	√	Left-looking
SPRSBLKLLT	×	×	√*	×	×	×	×	√	Left-looking
TAUCS	√	√	√	×	√*	×	×	√	Left-looking, multifrontal
UMFPACK	×	√*	×	×	×	×	×	×	Unsymmetric multifrontal
WSMP	×	×	×	√*	×	×	√*	√	Multifrontal

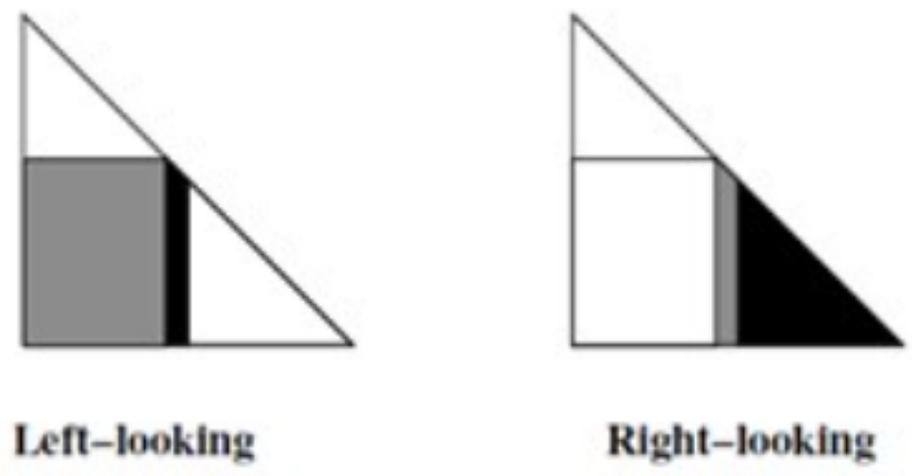
Factorization

left-looking

for k = 1,…,n do
  for i = k,…,n do
    for j = 1,…k-1 do
      $a_{ik}^{(k)} = a_{ik}^{(k)} - l_{ij} \cdot l_{kj}$
    end for
  end for
  $l_{kk} = \sqrt{a_{kk}^{(k-1)}}$
  for i = k+1,…,n do
    $l_{ik} = a_{ik}^{(k-1)} / l_{kk}$
  end for
end for

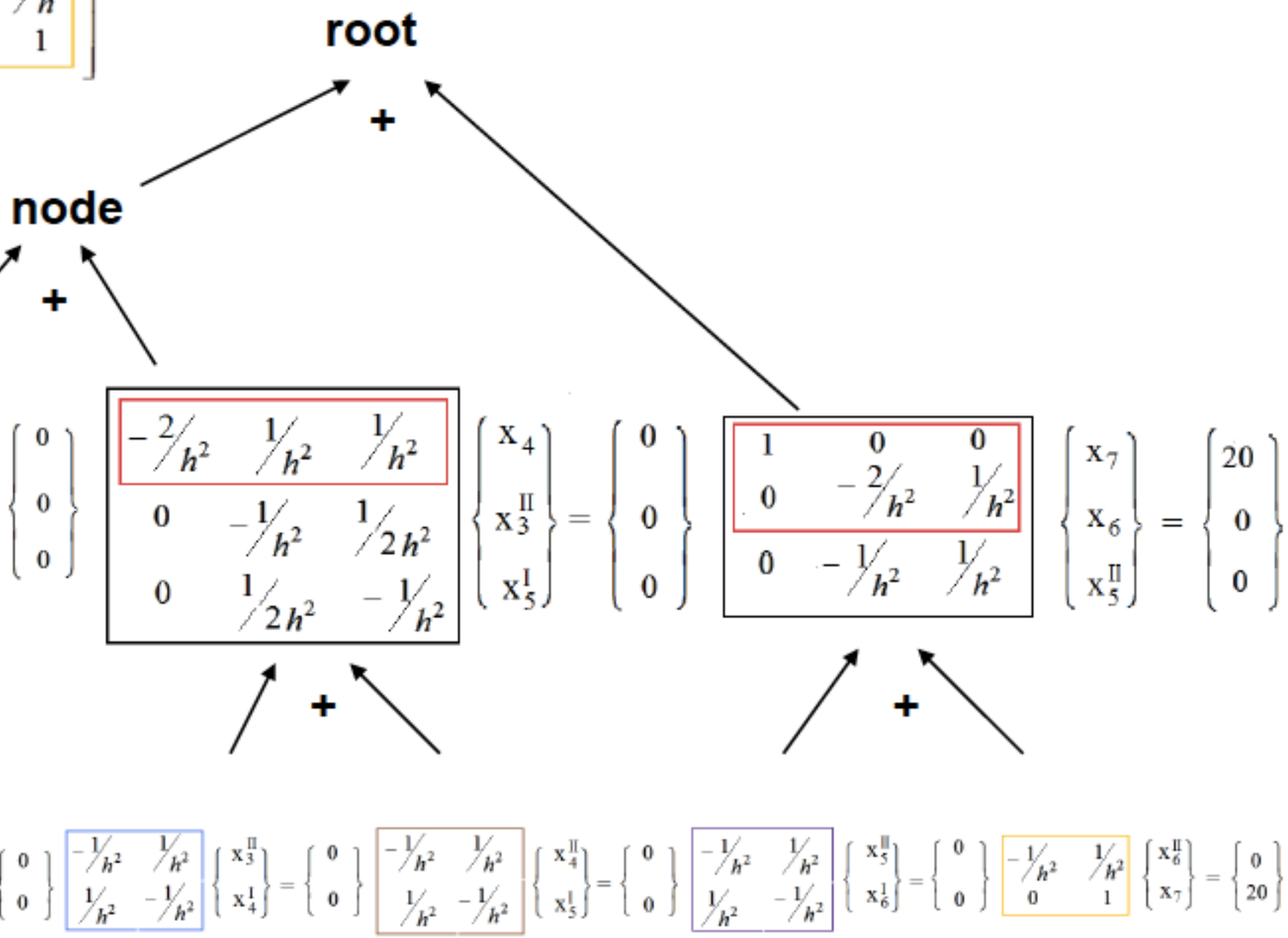
right-looking

for k = 1,…,n do
  $l_{kk} = \sqrt{a_{kk}^{(k-1)}}$
  for i = k+1,…,n do
    $l_{ik} = a_{ik}^{(k-1)} / l_{kk}$
    for j = k+1,…,i do
      $a_{ij}^{(k)} = a_{ij}^{(k)} - l_{ik} \cdot l_{jk}$
    end for
  end for
end for



Multifrontal solver

$$\begin{bmatrix}
1 & 0 & \dots & 0 & \dots & 0 & 0 \\
1/h^2 & -2/h^2 & 1/h^2 & 0 & \dots & 0 & 0 \\
0 & 1/h^2 & -2/h^2 & 1/h^2 & 0 & \dots & 0 \\
0 & \dots & 1/h^2 & -2/h^2 & 1/h^2 & \dots & 0 \\
0 & \dots & 0 & 1/h^2 & -2/h^2 & 1/h^2 & 0 \\
0 & 0 & \dots & 0 & 1/h^2 & -2/h^2 & 1/h^2 \\
0 & 0 & \dots & 0 & \dots & 0 & 1
\end{bmatrix}$$



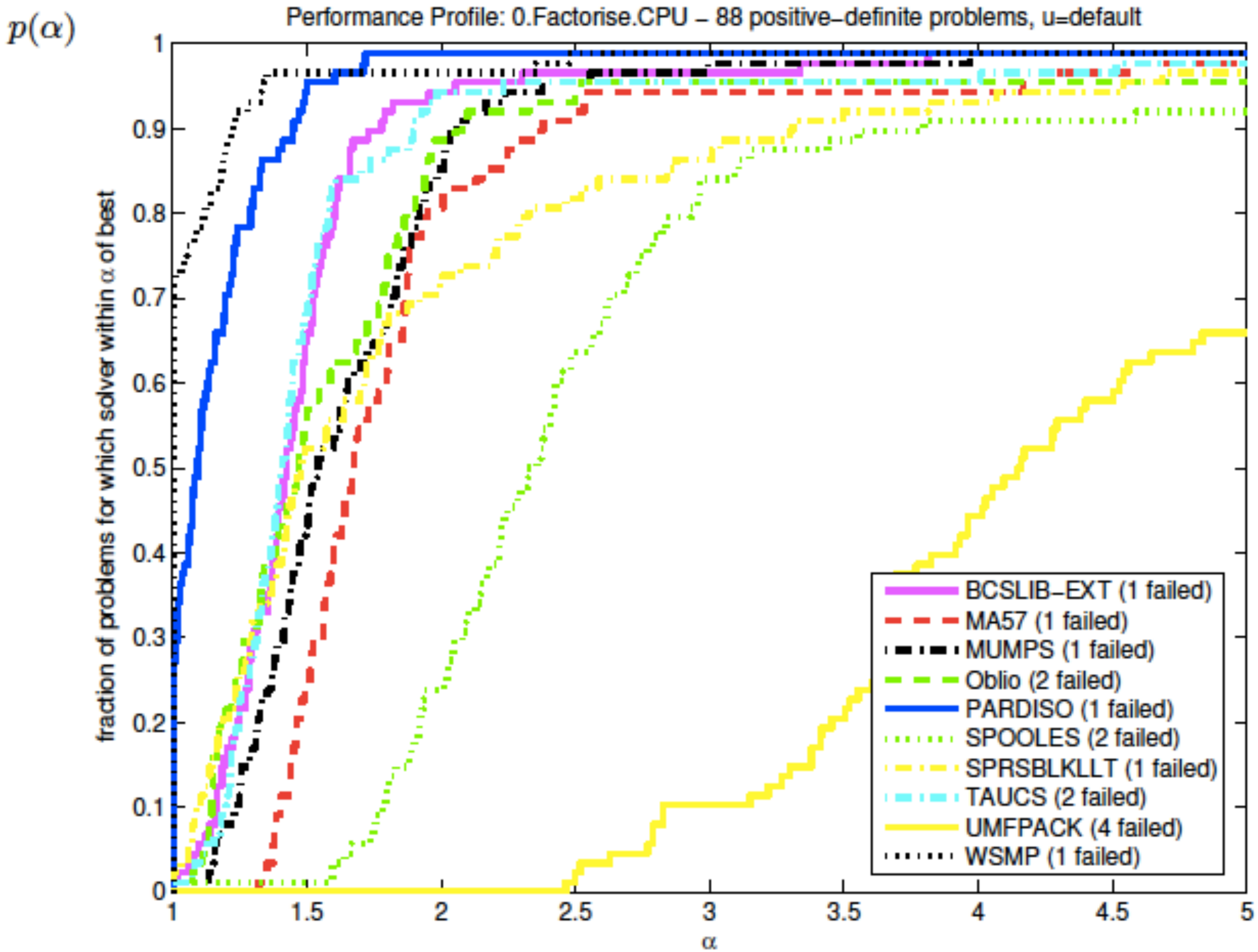
$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -2/h^2 & 1/h^2 \\ 0 & 1/h^2 & -1/h^2 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3^{I} \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ 0 \end{Bmatrix}$$

$$\begin{bmatrix} -2/h^2 & 1/h^2 & 1/h^2 \\ 0 & -1/h^2 & 1/2h^2 \\ 0 & 1/2h^2 & -1/h^2 \end{bmatrix} \begin{Bmatrix} x_4 \\ x_3^{II} \\ x_5^{I} \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ 0 \end{Bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -2/h^2 & 1/h^2 \\ 0 & -1/h^2 & 1/h^2 \end{bmatrix} \begin{Bmatrix} x_7 \\ x_6 \\ x_5^{II} \end{Bmatrix} = \begin{Bmatrix} 20 \\ 0 \\ 0 \end{Bmatrix}$$

$$\begin{bmatrix} 1 & 0 \\ 1/h^2 & -1/h^2 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2^{I} \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix} \quad \begin{bmatrix} -1/h^2 & 1/h^2 \\ 1/h^2 & -1/h^2 \end{bmatrix} \begin{Bmatrix} x_2^{II} \\ x_3^{I} \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix} \quad \begin{bmatrix} -1/h^2 & 1/h^2 \\ 1/h^2 & -1/h^2 \end{bmatrix} \begin{Bmatrix} x_3^{II} \\ x_4^{I} \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix} \quad \begin{bmatrix} -1/h^2 & 1/h^2 \\ 1/h^2 & -1/h^2 \end{bmatrix} \begin{Bmatrix} x_4^{II} \\ x_5^{I} \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix} \quad \begin{bmatrix} -1/h^2 & 1/h^2 \\ 1/h^2 & -1/h^2 \end{bmatrix} \begin{Bmatrix} x_5^{II} \\ x_6^{I} \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix} \quad \begin{bmatrix} -1/h^2 & 1/h^2 \\ 0 & 1 \end{bmatrix} \begin{Bmatrix} x_6^{II} \\ x_7 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 20 \end{Bmatrix}$$

Benchmarking



05/09	Class 9	Dense direct solvers	Understand the principle of LU decomposition and the optimization and parallelization techniques that lead to the LINPACK benchmark.
05/12	Class 10	Dense eigensolvers	Determine eigenvalues and eigenvectors and understand the fast algorithms for diagonalization and orthonormalization.
05/16	Class 11	Sparse direct solvers	Understand reordering in AMD and nested dissection, and fast algorithms such as skyline and multifrontal methods.
05/19	Class 12	Sparse iterative solvers	Understand the notion of positive definiteness, condition number, and the difference between Jacobi, CG, and GMRES.
05/23	Class 13	Preconditioners	Understand how preconditioning affects the condition number and spectral radius, and how that affects the CG method.
05/26	Class 14	Multigrid methods	Understand the role of smoothers, restriction, and prolongation in the V-cycle.
05/30	Class 15	Fast multipole methods, H-matrices	Understand the concept of multipole expansion and low-rank approximation, and the role of the tree structure.