Sparse Matrices in Numerical Mathematics

Miroslav Tůma

Faculty of Mathematics and Physics

Charles University

mirektuma@karlin.mff.cuni.cz

Praha, September 18, 2022

Outline

1 Introduction

Introductory notes

- Created as a material supporting online lectures of NMNV533.
- Assuming basic understanding of algebraic iterative (Krylov space) and direct (dense) solvers (elimination/factorization/solve) (A lot of these is repeated)
- The text deal prevailably with purely algebraic techniques. Such techniques often serve as building blocks for more complex approaches. In particular, some important techniques are mentioned at most. Like:
  - Multigrid/multilevel preconditioners,
  - Domain decomposition,
  - Row projection techniques.
- Only preconditioning of real systems considered here, although extension to complex field is typically straightforward.
- Orientation in variants of Cholesky and LU decompositions is assumed.

Introductory notes: resources / history

- The main resource is:
  Jennifer Scott and Miroslav Tůma: Algorithms for sparse linear systems, Birkhäuser- Springer, 2022, to appear.
- Printed parts of the resource will be provided to students until it will appear (expected open access then).
- Traditional material also the course text in Czech (nowadays outdated, not supported); see the web page of the course.

Introductory notes: resources and history of the course

- A few other resources:
  Davis, T. A. (2006). Direct Methods for Sparse Linear Systems. Fundamentals of Algorithms. SIAM, Philadelphia, PA.
  Davis, T. A., Rajamanickam, S., & Sid-Lakhdar, W.M. (2016). A survey of direct methods for sparse linear systems. Acta Numer., 25, 383-566.
  Duff, I. S., Erisman, A.M., & Reid, J. K. (2017). Direct Methods for Sparse Matrices (Second ed.). Oxford University Press, Oxford.
  George, A. & Liu, J. W. H. (1981). Computer Solution of Large Sparse Positive Definite Systems. Prentice Hall, Englewood Cliffs, NJ.
  Saad, Y. (2003b). Iterative Methods for Sparse Linear Systems (Second ed.). SIAM, Philadelphia, PA.

Motivation

- Most of our activities around solving

$$\mathbf{Ax} = \mathbf{b}$$

- Direct methods
- Iterative methods
- Practical boundaries between them more and more fuzzy.
- Principially different.

Motivation

Direct methods

- Direct methods: Transform A using a finite sequence of elementary transformations: An approach based on factorization (decomposition) and subsequent substitutions.
- The most simple case: $A \to LL^T$ or LDL^T or LU

  In principal $=$ Gaussian elimination. Modern (decompositional) form based a lot on the work of Householder (end of 1950's)
  - Solving systems with triangular matrices like L, U is generally much cheaper and more straightforward that using A.
  - Factorizations are backbone of direct methods.
  - Occasionally other factorizations than LU or LL^T or LDL^T
  - Most of the work is in the (Cholesky, indefinite, LU) decomposition.
  - But: also the computer model (sequential, concurrent processors, multicore, GPU) decides about relative complexity of the two steps.
- The algorithms can be made more efficient/stable using additional techniques before, after or during factorization.
- For example, the solution can be made more accurate by an auxiliary iterative method.

Motivation

Iterative methods

- Compute a sequence of approximations

$$x^{(0)}, x^{(1)}, x^{(2)}, \ldots$$

  that (hopefully) converge to the solution x of the linear system.
- Iterative method are usually accompanied by a problem transformation based on a direct method called preconditioner.
- Usually have to be accompanied by a problem transformation based on a direct method called preconditioner.

Motivation

Iterative methods

- Algebraic preconditioners are tools to convert the problem $Ax = b$ into the one which is easier to solve. They are typically expressed in matrix form as a transformation like:

$$MAx = Mb$$

- M can be then used to apply approximation to A^{-1} to vectors used in the iterative method.
- In practice, it can store approximation to A or A^{-1} (approximate inverse).

Motivation

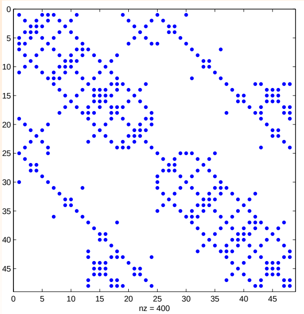
Contrast: direct versus iterative methods

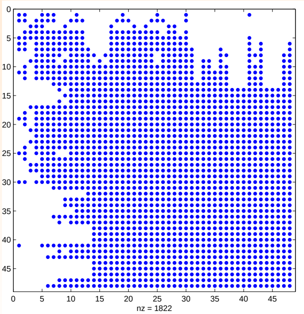
- Direct methods: designed to be robust, designed to solve
  - Properly implemented, they can be used as block-box solvers for computing solutions with predictable accuracy.
  - As we have seen, they can be expensive, requiring large amounts of memory, which increases with the size of A.
- Iterative methods: designed to approximate
  - The number of iterations depends on the initial guess $x^{(0)}$, A and b
  - Use the matrix A only indirectly, through matrix-vector products $\rightarrow$ memory requirements are limited to a (small) number of vectors of length the size of A
  - A does not need to be available explicitly.
  - They can be terminated as soon as the required accuracy in the computed solution is achieved.
  - Typically must be preconditioned. Preconditioner computation is sometimes based on a relaxation of a direct method.

Motivation

Where is the problem with direct methods?

- For example: sparse matrices and resulting factorizations may look like as follows:





Where is the problem with direct methods?

- For example: and they can look like as:

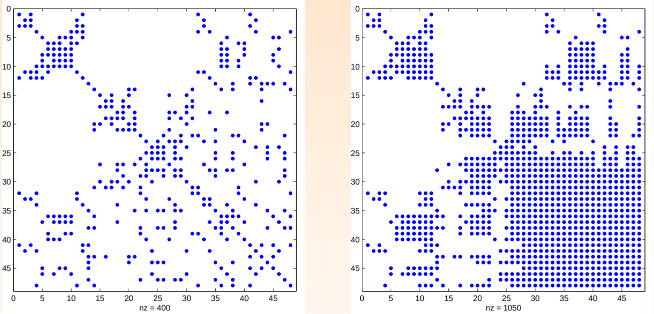


Figure: *The locations of the nonzero entries in a symmetric permutation of the matrix from Figure* **??** *(left) and in $\bar{L} + \bar{L}^T$ (right), where $\bar{L}$ is the Cholesky factor of the permuted matrix.*

Motivation

Where is the problem with direct methods?

- For example: and they can look like as:

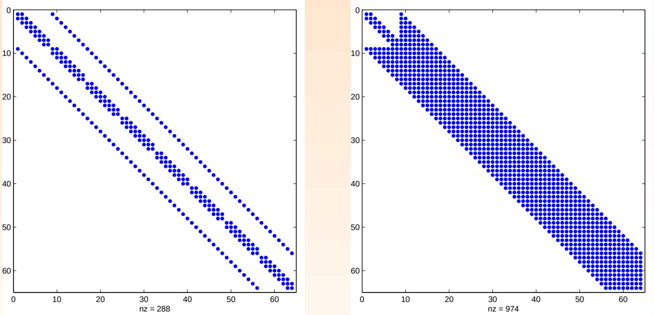


Figure: *The locations of the nonzero entries in a symmetric permutation of the matrix from Figure* **??** *(left) and in $\bar{L} + \bar{L}^T$ (right), where $\bar{L}$ is the Cholesky factor of the permuted matrix.*

Motivation

Where is the problem with direct methods?

- We need exploit sparsity (mentioned later)
- See the figures above
- We need sparse (complete) factorizations $A = LL^T$, LU (up to the floating-point model)

Where is the problem with iterative methods?

- We must transform (precondition)
- We need sparse (incomplete) factorizations $A = LL^T$, LU (up to the floating-point model) like
  - incomplete decompositions ($A \approx LL^T$, LU etc.)
  - incomplete inverse decompositions ($A^{-1} \approx ZZ^T$, WZ^T etc.)
- Or specific (PDE-based, model-based) approaches.