MATH 270A: Numerical Linear Algebra

Direct Methods, recapitulated

Martin W. Licht

UC San Diego

Fall 2018

First, there is a new optional programming assignment.

SVD, parting words

- The singular value decomposition $A = W\Sigma V$ is a helpful theoretical tool, but of limited importance in practical computations.
- Algorithms to compute the SVD are iterative at best: we compute better and better approximations of W, Σ, and V.
- If the matrix size A is large, then storing the orthogonal matrices W and V requires prohibitive amounts of memory.

In-Place Computations

Important for high performance computing problems.

- **LU decomposition without pivoting, Cholesky decomposition** In-place computation natural and possible.
- **QR decomposition via Givens rotations** matrix R stored in place, Givens rotation encoded in lower triangular part.
- **QR decomposition via Householder transformations** Householder vectors stored in lower triangular part, upper triangular matrix stored in upper triangular part and auxiliary vector
- **QR decomposition via Gram-Schmidt** Generally no in-place computation feasible.

Numerical stability

Numerical stability, that is, the influence of rounding errors, is not discussed in any detail in this course.

- **LU decomposition without pivoting** Rounding errors may have disastrous effects.
- **LU decomposition with pivoting** Numerical stable in practice, theoretical understanding still incomplete.
- **Cholesky decomposition** Stable for symmetric positive definite matrices.
- **QR decomposition via Givens rotations or Householder transformations** Stable.
- **QR decomposition via Gram-Schmidt** Stable if modified Gram-Schmidt is used (see programming exercise).

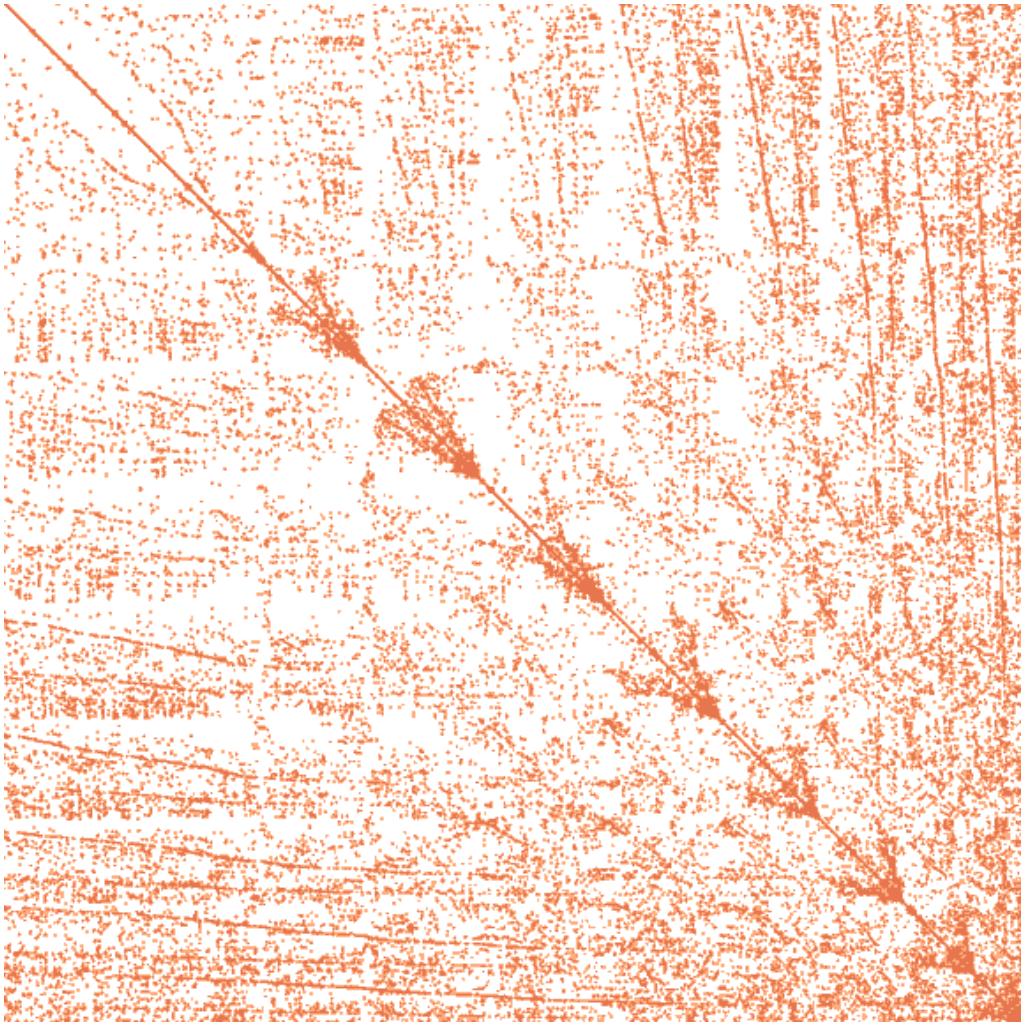
Computational Costs

Solving a linear system of equations $Ax = b$ with $A \in \mathbb{R}^{n \times n}$.

1. Factorization phase: cubic growth of costs
2. Solution phase: quadratic grwoth of costs.

Huge pay-off if $Ax = b$ is solved with the same matrix A but for a sequence of right-hand sides.

Sparse Matrix



Sparse Matrix

Let $A \in \mathbb{R}^{n \times n}$ and let $nz(A)$ be the number of non-zero entries. Loosely, we call A *sparse* if $nz(A) \ll n^2$.

Many matrices, such as arising in numerical partial differential equations, are sparse. $nz(A) \leq c \cdot n$ for some fixed $c > 0$.

In practice, only a list of non-zero entries is saved (for each entry, the row index, the column index, and the entry value).

Major problem of direct methods: Sparsity is not preserved. The factorizations suffer from fill-in, that is , the factoriations and the auxiliary results have much more non/zero entries than the original matrix.

Example

Let $A \in \mathbb{R}^{n \times n}$ with 8 non-zero entries per row. Let $n = 10^9$.

Suppose each integer occupies 4 bytes of memory and each floating-point number occupies 8 bytes of memory.

Storage of full matrix: $8 \cdot 10^{18}$ bytes, that is, 6 Exabyte.

Sparse storage, if we save row/column index and value per entry: $16 \cdot 10^9$, that is, 16 Gigabyte.

Enter **Iterative Methods**

Iterative Methods

Basic idea of iterative methods

- Given a first approximation $x_0 \in \mathbb{R}^n$ to the solution $x \in \mathbb{R}^n$ of the system $Ax = b$,
- we recursively construct a sequence of approximate solutions

$$x^{(k+1)} = \Phi(x^{(k)}), \quad k \in \mathbb{N}_0$$

  by some iteration rule $\Phi : \mathbb{R}^n \to \mathbb{R}^n$,
- and hope for convergence $x^{(k)} \to x$ as $k \to \infty$.

Iterative Methods

Typically, the iteration step

$$x^{(k+1)} = \Phi(x^{(k)}), \quad k \in \mathbb{N}_0$$

involves auxiliary memory in form of a few vectors of size $\mathbb{R}^n$, and involves computing a few matrix vector multiplications.

$\Longrightarrow$ Memory consumpition at each time step linear in n.

$\Longrightarrow$ Time complexity for each time step quadratic in n.

In typical applications, we get a sufficiently good approximation x^l after relatively few iterations, $l \lll n$.

Iterative Methods

- **Classical Fixpoint Methods:** Classical form of an iterative method as just described.
- **Krylov subspace methods:** Special class of *semi-iterative methods* that is the state-of-the-art in numerical linear algebra.