# Iterative methods (for linear systems)

MATH 6610 Lecture 22

November 4, 2020

Trefethen & Bau: Lectures 32, 35, 38

## Direct methods

We have previously discssed *direct* methods, i.e., methods that

- orthogonalize vectors
- solve linear systems
- compute eigenvalues

assuming that operations like $x \mapsto Ax$ are efficiently computable.

## Direct methods

We have previously discssed *direct* methods, i.e., methods that

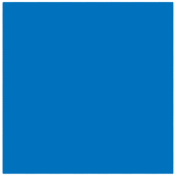
- orthogonalize vectors
- solve linear systems
- compute eigenvalues

assuming that operations like $x \mapsto Ax$ are efficiently computable.

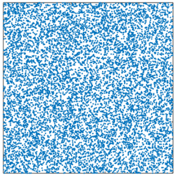
For very large matrices, e.g., $A \in \mathbb{C}^{10^6 \times 10^6}$, such procedures are not practical.

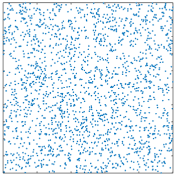
For such large matrices of general type, there is not much that can be done.

But in practice, matrices are *sparse*, having a reasonably small percentage of nonzero entries.


Dense matrix

10% density


1% density


0.2% density

## Direct methods and sparse matrices

Sparse matrices arise frequently in applications.

This is fortunate since fundamental operations like $x \mapsto Ax$ are very efficient for sparse matrices.

## Direct methods and sparse matrices

Sparse matrices arise frequently in applications.

This is fortunate since fundamental operations like $x \mapsto Ax$ are very efficient for sparse matrices.

Unfortunately, direct methods operating on sparse matrix frequently result in dense matrices:

- QR factorizations
- LU factorizations
- Eigenvalue algorithms

Even just storing such matrices can be impossible in practice.

## Iterative methods

In contrast to direct methods, iterative methods produce answers that gradually approach the solution, performing only operations that generally don't require large dense matrices.

There are two major problems that iterative methods generally seek to solve:

- Solve linear systems
- Compute eigenvalues/eigenvectors

We'll briefly discuss the general ideas for the first class of problems.

## Themes of iterative methods

$$Ax = b$$

Many iterative methods focus on ensuring the following properties

- A sequence of solution approximations, $x_0, x_1, x_2, \ldots$ is constructed, with a new approximation formed at every iteration.
- The update $x_k \mapsto x_{k+1}$ typically involves only efficient matrix-vector multiplications (e.g., exploiting sparsity)
- The sequence $\{x_k\}$ gradually approaches the solution as $k \uparrow \infty$.
- Termination is frequently judged by inspecting the residual $\|Ax_k - b\|$.

## Stationary iterative methods

$$Ax = b$$

Stationary iterative methods (also “relaxation methods") are among the first types of iterative methods developed.

The basic idea of stationary iterative methods is as follows: we update solutions linearly:

$$Bx_{k+1} = Cx_k + d,$$

where $B$ and $C$ are matrices, and $d$ is a vector.

## Stationary iterative methods

$$Ax = b$$

Stationary iterative methods (also “relaxation methods") are among the first types of iterative methods developed.

The basic idea of stationary iterative methods is as follows: we update solutions linearly:

$$Bx_{k+1} = Cx_k + d,$$

where $B$ and $C$ are matrices, and $d$ is a vector. In order to prove convergence, we attempt to choose $B$, $C$, and $d$ so that

$$e_{k+1} = Fe_k, \qquad\qquad e_k := x_k - x$$

We can ensure this relation if we make the choice,

$$B - C = A, \qquad\qquad d = b.$$

## Stationary iterative methods and convergence

$$e_{k+1} = F e_k, \qquad\qquad e_k := x_k - x$$

With the above relation, then we obtain convergence if

$$\lim_{k \to \infty} F^k = 0.$$

Ensuring this condition typically depends on the matrix $A$ and on what kind of decomposition $A = B - C$ is chosen.

## Some choices

$$Bx_{k+1} = Cx_k + b, \qquad\qquad A = B - C.$$

Suppose that $A = L + D + U$ where

- $L$ and $U$ are the lower- and upper-triangular portions of $A$, respectively,
- $D$ is the diagonal portion of $A$.

## Some choices

$$Bx_{k+1} = Cx_k + b, \qquad\qquad A = B - C.$$

Suppose that $A = L + D + U$ where

- $L$ and $U$ are the lower- and upper-triangular portions of $A$, respectively,
- $D$ is the diagonal portion of $A$.

Many methods boil down to how $B$ and $C$ are chosen:

- Jacobi method: $B = D$, $C = -L - U$.

## Some choices

$$Bx_{k+1} = Cx_k + b, \qquad\qquad A = B - C.$$

Suppose that $A = L + D + U$ where

- $L$ and $U$ are the lower- and upper-triangular portions of $A$, respectively,
- $D$ is the diagonal portion of $A$.

Many methods boil down to how $B$ and $C$ are chosen:

- Jacobi method: $B = D$, $C = -L - U$.
- Gauss-Seidel method: $B = D + L$, $C = -U$.

## Some choices

$$Bx_{k+1} = Cx_k + b, \qquad\qquad A = B - C.$$

Suppose that $A = L + D + U$ where

- $L$ and $U$ are the lower- and upper-triangular portions of $A$, respectively,
- $D$ is the diagonal portion of $A$.

Many methods boil down to how $B$ and $C$ are chosen:

- Jacobi method: $B = D$, $C = -L - U$.
- Gauss-Seidel method: $B = D + L$, $C = -U$.
- Successive over-relaxation method: $B = D + \alpha L$, $C = (1 - \alpha)D - \alpha U$.

These methods have strong theory when applied to discretizations of Laplace's equation.

## Krylov subspace methods

Krylov subspace methods are a general class of iterative methods.

Krylov subspace methods build approximations from the subspace,

$$\operatorname{span}\left\{b, Ab, A^2 b, \ldots, A^{r-1} b\right\},$$

where $r$ is either adaptively or *a priori* specified.

The vector $b$ is either the right-hand side of a linear system, or an initial guess for an eigenvector.

## Krylov subspace methods

Krylov subspace methods are a general class of iterative methods.

Krylov subspace methods build approximations from the subspace,

$$\operatorname{span}\left\{b, Ab, A^2 b, \ldots, A^{r-1} b\right\},$$

where $r$ is either adaptively or *a priori* specified.

The vector $b$ is either the right-hand side of a linear system, or an initial guess for an eigenvector.

Of course there are some implementation details:

- Neither $A^k$ nor $A^k b$ are ever explicitly computed.
- Some orthogonalization is typically performed to avoid ill-conditioning.

## Some Krylov subspace methods

Krylov subspace methods are among the most popular and effective iterative methods.

For solving linear systems, examples are

- Conjugate Gradient (CG) and variants (CGStab, BiCGStab, etc.)
- Generalized minimum residuals (GMRES) and variants (MINRES, QMR, etc.)

## Some Krylov subspace methods

Krylov subspace methods are among the most popular and effective iterative methods.

For solving linear systems, examples are

- Conjugate Gradient (CG) and variants (CGStab, BiCGStab, etc.)
- Generalized minimum residuals (GMRES) and variants (MINRES, QMR, etc.)

For computing eigenvalues some iterative methods are

- Lanczos iteration (for Hermitian $A$)
- Arnoldi iteration (for general $A$)

## Preconditioning

$$Ax = b$$

The conditioning of solving systems depends on the condition number of $A$.

## Preconditioning

$$Ax = b$$

The conditioning of solving systems depends on the condition number of $A$.

Preconditioning is a technique where an "easily invertible" matrix $P$ is inserted in hopes of making the equation more well-conditioning:

$$P^{-1}Ax = P^{-1}b, \qquad AP^{-1}(Px) = b.$$

Typically, $P$ is chosen as an "approximate" inverse of $A$, such as its diagonal.

## Preconditioning

$$Ax = b$$

The conditioning of solving systems depends on the condition number of $A$.

Preconditioning is a technique where an "easily invertible" matrix $P$ is inserted in hopes of making the equation more well-conditioning:

$$P^{-1}Ax = P^{-1}b, \qquad AP^{-1}(Px) = b.$$

Typically, $P$ is chosen as an "approximate" inverse of $A$, such as its diagonal.

Effective preconditioning serves both to stabilize numerical computations, and to accelerate iterative methods (result in fewer iterations).