Lecture 13: Biorthogonalization methods

School of Mathematical Sciences, Xiamen University

1. Tridiagonal biorthogonalization

- Assume that $\mathbf{A}$ is Hermitian, i.e., $\mathbf{A} = \mathbf{A}^*$. The Lanczos process is a process of tridiagonal orthogonalization. If no breakdown, then

$$\mathbf{A} = \mathbf{Q}\mathbf{T}\mathbf{Q}^*.$$

  Note that

$$\text{“unitary”} \quad + \quad \text{“tridiagonal”}.$$

- For non-Hermitian $\mathbf{A}$, we must give up either the unitary transformations or the tridiagonal structure.
- The Arnoldi process gives up the tridiagonal structure:

$$\mathbf{A} = \mathbf{Q}\mathbf{H}\mathbf{Q}^*.$$

- Tridiagonal biorthogonalization methods give up the unitary transformations: $\mathbf{V}$ is nonsingular but generally not unitary,

$$\mathbf{A} = \mathbf{V}\mathbf{T}\mathbf{V}^{-1}.$$

- The term "biorthogonal" refers to the fact that although the columns of $\mathbf{V}$ are not orthogonal to each other, they are orthogonal to the columns of $\mathbf{V}^{-*}$, as follows trivially from the identity

$$(\mathbf{V}^{-*})^*\mathbf{V} = \mathbf{V}^{-1}\mathbf{V} = \mathbf{I}.$$

- Define $\mathbf{W} = \mathbf{V}^{-*}$. Let $\mathbf{v}_j$ and $\mathbf{w}_j$ denote the jth columns of $\mathbf{V}$ and $\mathbf{W}$, respectively. These vectors are biorthogonal in the sense that

$$\mathbf{w}_i^*\mathbf{v}_j = \delta_{ij},$$

where δ_{ij} is the Kronecker delta function. Define

$$\mathbf{V}_j = \begin{bmatrix}\mathbf{v}_1 & \cdots & \mathbf{v}_j\end{bmatrix}, \quad \mathbf{W}_j = \begin{bmatrix}\mathbf{w}_1 & \cdots & \mathbf{w}_j\end{bmatrix}.$$

In matrix form, the biorthogonality condition can be written

$$\mathbf{W}_j^*\mathbf{V}_j = \mathbf{V}_j^*\mathbf{W}_j = \mathbf{I}_j.$$

- The Lanczos relation

$$\mathbf{A}\mathbf{Q}_j = \mathbf{Q}_{j+1}\widetilde{\mathbf{T}}_j, \qquad \mathbf{T}_j = \mathbf{Q}_j^*\mathbf{A}\mathbf{Q}_j.$$

- The Arnoldi relation

$$\mathbf{A}\mathbf{Q}_j = \mathbf{Q}_{j+1}\widetilde{\mathbf{H}}_j, \qquad \mathbf{H}_j = \mathbf{Q}_j^*\mathbf{A}\mathbf{Q}_j.$$

- Corresponding formulas for biorthogonalization methods:

$$\mathbf{A}\mathbf{V}_j = \mathbf{V}_{j+1}\widetilde{\mathbf{T}}_j, \qquad \mathbf{A}^*\mathbf{W}_j = \mathbf{W}_{j+1}\widetilde{\mathbf{S}}_j,$$

and

$$\mathbf{T}_j = \mathbf{S}_j^* = \mathbf{W}_j^*\mathbf{A}\mathbf{V}_j.$$

$\mathbf{T}_j$ is tridiagonal and is obtained by deleting the last row of $\widetilde{\mathbf{T}}_j$.

- Note that

$$\mathbf{A}\mathbf{V}_j = \mathbf{V}_{j+1}\widetilde{\mathbf{T}}_j$$

takes the form

$$\mathbf{A}\begin{bmatrix}\mathbf{v}_1 & \cdots & \mathbf{v}_j\end{bmatrix} = \begin{bmatrix}\mathbf{v}_1 & \cdots & \mathbf{v}_{j+1}\end{bmatrix}\begin{bmatrix}\alpha_1 & \gamma_1 & & & \\ \beta_1 & \alpha_2 & \gamma_2 & & \\ & \beta_2 & \alpha_3 & \ddots & \\ & & \ddots & \ddots & \gamma_{j-1} \\ & & & \beta_{j-1} & \alpha_j \\ & & & & \beta_j\end{bmatrix},$$

which corresponds to the three-term recurrence relation

$$\mathbf{A}\mathbf{v}_j = \gamma_{j-1}\mathbf{v}_{j-1} + \alpha_j\mathbf{v}_j + \beta_j\mathbf{v}_{j+1}.$$

- Similarly,

$$\mathbf{A}^*\mathbf{W}_j = \mathbf{W}_{j+1}\widetilde{\mathbf{S}}_j$$

takes the form

$$\mathbf{A}^*\begin{bmatrix}\mathbf{w}_1 & \cdots & \mathbf{w}_j\end{bmatrix} = \begin{bmatrix}\mathbf{w}_1 & \cdots & \mathbf{w}_{j+1}\end{bmatrix}\begin{bmatrix}\overline{\alpha}_1 & \overline{\beta}_1 & & & \\ \overline{\gamma}_1 & \overline{\alpha}_2 & \overline{\beta}_2 & & \\ & \overline{\gamma}_2 & \overline{\alpha}_3 & \ddots & \\ & & \ddots & \ddots & \overline{\beta}_{j-1} \\ & & & \overline{\gamma}_{j-1} & \overline{\alpha}_j \\ & & & & \overline{\gamma}_j \end{bmatrix},$$

which corresponds to the three-term recurrence relation

$$\mathbf{A}^*\mathbf{w}_j = \overline{\beta}_{j-1}\mathbf{w}_{j-1} + \overline{\alpha}_j\mathbf{w}_j + \overline{\gamma}_j\mathbf{w}_{j+1}.$$

- These recurrence relations suggest an algorithm. Begin with vectors $\mathbf{v}_1$ and $\mathbf{w}_1$ that are arbitrary except for satisfying $\mathbf{v}_1^*\mathbf{w}_1 = 1$, and set $\beta_0 = \gamma_0 = 0$ and $\mathbf{v}_0 = \mathbf{w}_0 = \mathbf{0}$. Now for each $j = 1, 2, \ldots,$ set

$$\alpha_j = \mathbf{w}_j^*\mathbf{A}\mathbf{v}_j.$$

  The vectors $\mathbf{v}_{j+1}$ and $\mathbf{w}_{j+1}$ are then determined by the recurrence relations up to scalar factors. These factors may be chosen arbitrarily, subject to the normalization

$$\mathbf{w}_{j+1}^*\mathbf{v}_{j+1} = 1.$$

- The vectors generated by the procedure just described satisfy

$$\mathbf{v}_j \in \mathcal{K}_j(\mathbf{A}, \mathbf{v}_1), \qquad \mathbf{w}_j \in \mathcal{K}_j(\mathbf{A}^*, \mathbf{w}_1).$$

- Breakdowns? $\mathbf{v}_j = \mathbf{0}$ or $\mathbf{w}_j = \mathbf{0}$ or $\mathbf{w}_j^*\mathbf{v}_j = 0$

2. Bi-conjugate gradients (Bi-CG)

- The principle of Bi-CG is to pick

$$\mathbf{x}_j \in \mathbf{x}_0 + \mathcal{K}_j(\mathbf{A}, \mathbf{r}_0)$$

subject to the orthogonality condition

$$\mathbf{r}_j \perp \mathcal{K}_j(\mathbf{A}^*, \mathbf{w}_1),$$

where $\mathbf{w}_1 \in \mathbb{C}^n$ is an arbitrary vector satisfying $(\mathbf{v}_1 = \mathbf{r}_0)$

$$\mathbf{w}_1^* \mathbf{r}_0 = 1.$$

- Proceeding in the same manner as for the derivation of CG, we can derive the scheme for Bi-CG. See Saad's book – *Iterative methods for sparse linear systems* (Chapters 6 and 7) for details. Here we provide an alternative based on preconditioned CG.

- Consider the following linear system

$$\begin{bmatrix} \mathbf{0} & \mathbf{A} \\ \mathbf{A}^\top & \mathbf{0} \end{bmatrix} \begin{bmatrix} \widetilde{\mathbf{x}} \\ \mathbf{x} \end{bmatrix} = \begin{bmatrix} \mathbf{b} \\ \widetilde{\mathbf{b}} \end{bmatrix}.$$

Apply the PCG scheme with the corresponding inner product replaced by the bilinear form

$$\left\langle \begin{bmatrix} \widetilde{\mathbf{x}} \\ \mathbf{x} \end{bmatrix}, \begin{bmatrix} \widetilde{\mathbf{y}} \\ \mathbf{y} \end{bmatrix} \right\rangle_{\text{Bi-CG}} = \begin{bmatrix} \widetilde{\mathbf{x}} \\ \mathbf{x} \end{bmatrix}^\top \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ \mathbf{I} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \widetilde{\mathbf{y}} \\ \mathbf{y} \end{bmatrix}$$

and the preconditioner

$$\begin{bmatrix} \mathbf{0} & \mathbf{I} \\ \mathbf{I} & \mathbf{0} \end{bmatrix},$$

then we obtain the Bi-CG scheme for $\mathbf{Ax} = \mathbf{b}$:

$$
\begin{aligned}
&\mathbf{r}_0 = \mathbf{p}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0; && \widetilde{\mathbf{r}}_0 = \widetilde{\mathbf{p}}_0 = \widetilde{\mathbf{b}} - \mathbf{A}^\top \widetilde{\mathbf{x}}_0;\\
&\mathbf{x}_j = \mathbf{x}_{j-1} + \alpha_j \mathbf{p}_{j-1}; && \widetilde{\mathbf{x}}_j = \widetilde{\mathbf{x}}_{j-1} + \alpha_j \widetilde{\mathbf{p}}_{j-1};\\
&\mathbf{r}_j = \mathbf{r}_{j-1} - \alpha_j \mathbf{A}\mathbf{p}_{j-1}; && \widetilde{\mathbf{r}}_j = \widetilde{\mathbf{r}}_{j-1} - \alpha_j \mathbf{A}^\top \widetilde{\mathbf{p}}_{j-1};\\
&\mathbf{p}_j = \mathbf{r}_j + \beta_j \mathbf{p}_{j-1}; && \widetilde{\mathbf{p}}_j = \widetilde{\mathbf{r}}_j + \beta_j \widetilde{\mathbf{p}}_{j-1};
\end{aligned}
$$

where

$$
\alpha_j = \frac{\widetilde{\mathbf{r}}_{j-1}^\top \mathbf{r}_{j-1} + \mathbf{r}_{j-1}^\top \widetilde{\mathbf{r}}_{j-1}}{\mathbf{p}_{j-1}^\top \mathbf{A}^\top \widetilde{\mathbf{p}}_{j-1} + \widetilde{\mathbf{p}}_{j-1}^\top \mathbf{A}\mathbf{p}_{j-1}} = \frac{\widetilde{\mathbf{r}}_{j-1}^\top \mathbf{r}_{j-1}}{\widetilde{\mathbf{p}}_{j-1}^\top \mathbf{A}\mathbf{p}_{j-1}};
$$

and

$$
\beta_j = \frac{\widetilde{\mathbf{r}}_j^\top \mathbf{r}_j + \mathbf{r}_j^\top \widetilde{\mathbf{r}}_j}{\widetilde{\mathbf{r}}_{j-1}^\top \mathbf{r}_{j-1} + \mathbf{r}_{j-1}^\top \widetilde{\mathbf{r}}_{j-1}} = \frac{\widetilde{\mathbf{r}}_j^\top \mathbf{r}_j}{\widetilde{\mathbf{r}}_{j-1}^\top \mathbf{r}_{j-1}}.
$$

Obviously, by writing $\mathbf{s}_j = \overline{\widetilde{\mathbf{r}}}_j$, $\mathbf{q}_j = \overline{\widetilde{\mathbf{p}}}_j$, $\widehat{\mathbf{b}} = \overline{\widetilde{\mathbf{b}}}$, and $\widehat{\mathbf{x}}_j = \overline{\widetilde{\mathbf{x}}}_j$, we have

$$\begin{aligned}
&\mathbf{r}_0 = \mathbf{p}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0; && \mathbf{s}_0 = \mathbf{q}_0 = \widehat{\mathbf{b}} - \mathbf{A}^*\widehat{\mathbf{x}}_0;\\
&\mathbf{x}_j = \mathbf{x}_{j-1} + \alpha_j \mathbf{p}_{j-1}; && \widehat{\mathbf{x}}_j = \widehat{\mathbf{x}}_{j-1} + \alpha_j \mathbf{q}_{j-1};\\
&\mathbf{r}_j = \mathbf{r}_{j-1} - \alpha_j \mathbf{A}\mathbf{p}_{j-1}; && \mathbf{s}_j = \mathbf{s}_{j-1} - \overline{\alpha}_j \mathbf{A}^*\mathbf{q}_{j-1};\\
&\mathbf{p}_j = \mathbf{r}_j + \beta_j \mathbf{p}_{j-1}; && \mathbf{q}_j = \mathbf{s}_j + \overline{\beta}_j \mathbf{q}_{j-1};
\end{aligned}$$

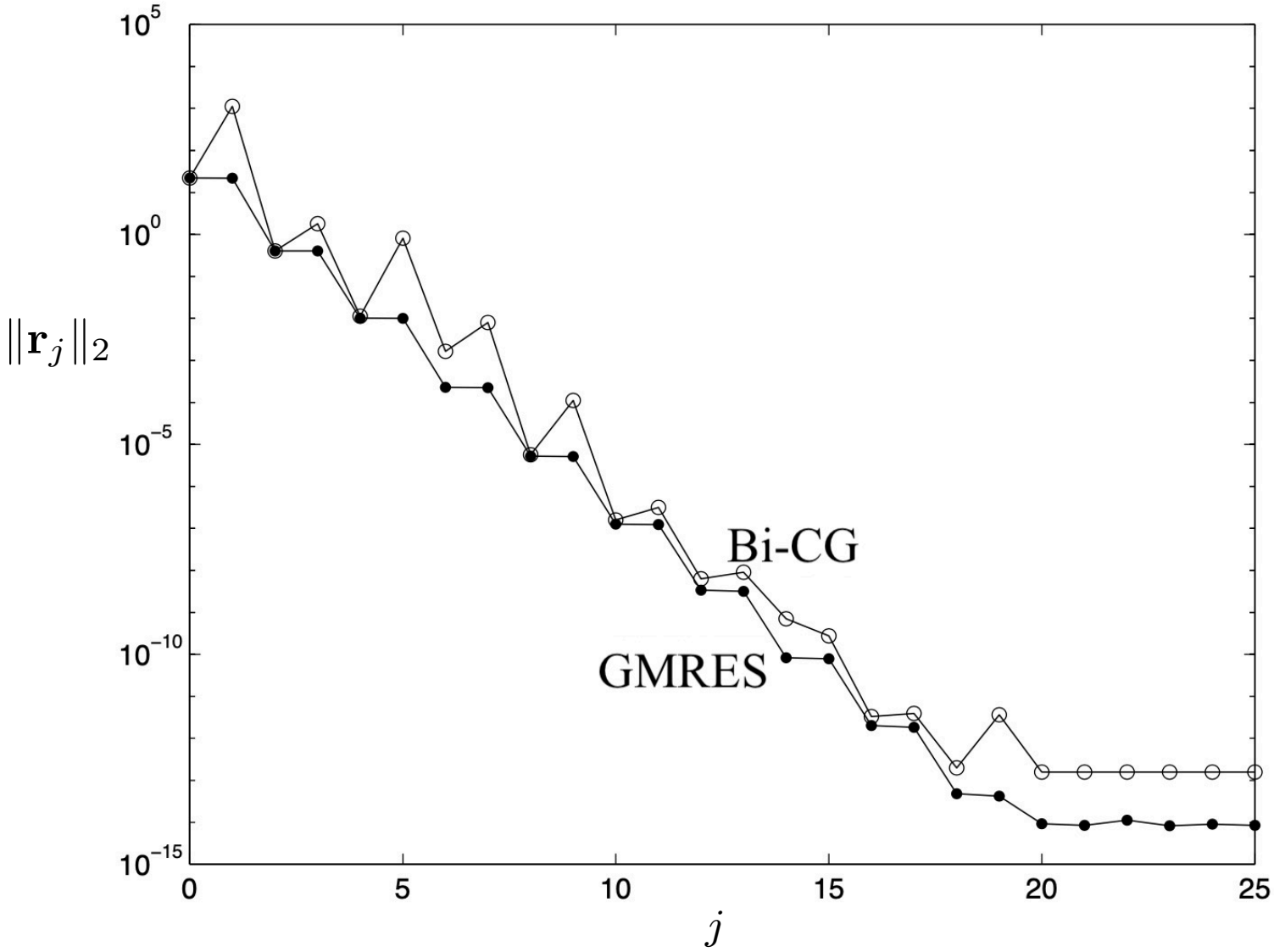
where

$$\alpha_j = \frac{\mathbf{s}_{j-1}^* \mathbf{r}_{j-1}}{\mathbf{q}_{j-1}^* \mathbf{A}\mathbf{p}_{j-1}}; \qquad \beta_j = \frac{\mathbf{s}_j^* \mathbf{r}_j}{\mathbf{s}_{j-1}^* \mathbf{r}_{j-1}}.$$

As the case in CG, it is readily shown that $\mathbf{s}_j^* \mathbf{r}_i = 0$ and $\mathbf{q}_j^* \mathbf{A}\mathbf{p}_i = 0$ for $i < j$.

- Discussion: What happens if Bi-CG is applied to a linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$ with Hermitian positive definite $\mathbf{A}$?

- Comparsion of GMRES and Bi-CG for a 500×500 matrix



3. Conjugate orthogonal conjugate gradients (COCG)

- If the system matrix $\mathbf{A}$ is complex symmetric, i.e.,

$$\mathbf{A} \in \mathbb{C}^{m\times m}, \qquad \mathbf{A} = \mathbf{A}^{\top},$$

with the choice $\mathbf{s}_0 = \bar{\mathbf{r}}_0$, where $\mathbf{s}_0$ is the initial residual of the dual system

$$\mathbf{A}^{*}\widehat{\mathbf{x}} = \widehat{\mathbf{b}}$$

arising in Bi-CG and $\bar{\mathbf{r}}_0$ denotes the complex conjugate of $\mathbf{r}_0$, Bi-CG reduces to COCG because

$$\mathbf{s}_j = \overline{\mathbf{r}}_j, \qquad \mathbf{q}_j = \overline{\mathbf{p}}_j.$$

- The details of COCG and its preconditioned version (PCOCG) are given below.

Algorithm COCG: $\mathbf{Ax} = \mathbf{b}$

Choose arbitrary $\mathbf{x}_0$; set $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$ and $\mathbf{p}_0 = \mathbf{r}_0$;

for $k = 1, 2, \ldots,$ **do** until convergence or breakdown:

$$\mathbf{x}_j = \mathbf{x}_{j-1} + \alpha_j \mathbf{p}_{j-1};$$
$$\mathbf{r}_j = \mathbf{r}_{j-1} - \alpha_j \mathbf{A}\mathbf{p}_{j-1};$$
$$\mathbf{p}_j = \mathbf{r}_j + \beta_j \mathbf{p}_{j-1};$$

where

$$\alpha_j = \frac{\mathbf{r}_{j-1}^\top \mathbf{r}_{j-1}}{\mathbf{p}_{j-1}^\top \mathbf{A}\mathbf{p}_{j-1}}; \quad \beta_j = \frac{\mathbf{r}_j^\top \mathbf{r}_j}{\mathbf{r}_{j-1}^\top \mathbf{r}_{j-1}}.$$

- The only essential change of COCG with respect to CG is the replacement of the standard inner product $\mathbf{y}^*\mathbf{x}$ by the bilinear form $\mathbf{y}^\top\mathbf{x}$.
- Discussion: What happens if COCG is applied to a linear system $\mathbf{Ax} = \mathbf{b}$ with Hermitian positive definite $\mathbf{A}$?

Algorithm PCOCG: $\mathbf{AM}^{-1}\mathbf{z} = \mathbf{b}$, $\mathbf{x} = \mathbf{M}^{-1}\mathbf{z}$

Choose $\mathbf{x} = \mathbf{x}_0$;

set $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$ and $\mathbf{p}_0 = \mathbf{M}^{-1}\mathbf{r}_0$;

for $k = 1, 2, \ldots,$ do until convergence or breakdown:

$$\mathbf{x}_j = \mathbf{x}_{j-1} + \alpha_j \mathbf{p}_{j-1};$$
$$\mathbf{r}_j = \mathbf{r}_{j-1} - \alpha_j \mathbf{A}\mathbf{p}_{j-1};$$
$$\mathbf{p}_j = \mathbf{M}^{-1}\mathbf{r}_j + \beta_j \mathbf{p}_{j-1};$$

where

$$\alpha_j = \frac{\mathbf{r}_{j-1}^\top \mathbf{M}^{-1}\mathbf{r}_{j-1}}{\mathbf{p}_{j-1}^\top \mathbf{A}\mathbf{p}_{j-1}}; \quad \beta_j = \frac{\mathbf{r}_j^\top \mathbf{M}^{-1}\mathbf{r}_j}{\mathbf{r}_{j-1}^\top \mathbf{M}^{-1}\mathbf{r}_{j-1}}.$$

- The preconditioner $\mathbf{M}$ in PCOCG has to be complex symmetric. The only essential change of PCOCG with respect to COCG is the replacement of the bilinear form $\mathbf{y}^\top\mathbf{x}$ by the bilinear form $\mathbf{y}^\top\mathbf{M}^{-1}\mathbf{x}$.

4. Advantage and disadvantages of Bi-CG

- One great advantage over GMRES:

  It involves three-term recurrences, enabling the work per step and the storage requirements to remain under control even when many steps are needed.

- Two disadvantages:

  (1) Its convergence is slower and often erratic. It may have the consequence of reducing the ultimately attainable accuracy because of rounding error. In the extreme, it becomes the phenomenon of breakdown of the iteration, where an inner product becomes zero and no further progress is possible.

  (2) It requires multiplication by $\mathbf{A}^*$ as well as $\mathbf{A}$. Depending on how these products are implemented both mathematically and in terms of computer architecture, this may be anything from a minor addition burden to effectively impossible.

5. Other variants

- Desired properties

  Smoothed convergence curves + breakdown free + transpose free
- Look-ahead Lanczos

  avoid breakdowns
- CGS = conjugate gradients squares

  transpose free, but is twice as erratic
- QMR = quasi-minimal residuals

  Pronounced effect on the smoothness of convergence
- Bi-CGSTAB = stabilized Bi-CG (via stabilizing CGS)

  Significantly smooths the convergence of Bi-CG, transpose free
- TFQMR = tanspose-free QMR

  transpose free and smooth convergence

6. Overview of iterative methods

6.1. Why iterate?

- The importance of iterative algorithms in linear algebra stems from a simple fact: noniterative or "direct" algorithms for general matrices requires $\mathcal{O}(m^3)$ work.
- It is too much both in the absolute sense that m^3 is huge when m is large, and in the relative sense that since the input to most matrix problems involves only $\mathcal{O}(m^2)$ numbers, it seems unreasonable that $\mathcal{O}(m^3)$ work must be expended in solving them.

6.2. Structure, Sparsity, and Black box

- Sparsity:

  A finite difference discretization of a PDE may lead to a matrix of dimension $m = 10^5$ with only $\nu = 10$ nonzero entries per row.

- Black box:

$$\mathbf{x} \longrightarrow \boxed{\begin{array}{c}\textbf{BLACK}\\ \textbf{BOX}\end{array}} \longrightarrow \mathbf{Ax}$$

  The iterative algorithm requires nothing more than the ability to determine $\mathbf{Ax}$ for any $\mathbf{x}$, which in a computer program will be effected by a procedure whose internal workings need be of no concern to the designer of the iterative algorithm. (Some iterative algorithms also require the computation of $\mathbf{A}^*\mathbf{x}$.)

  For the example of a sparse matrix $\mathbf{A}$, it is easy to design a procedure to compute $\mathbf{Ax}$ in only $\mathcal{O}(\nu m)$ rather than $\mathcal{O}(m^2)$ operations.

6.3. Number of steps, Work per step

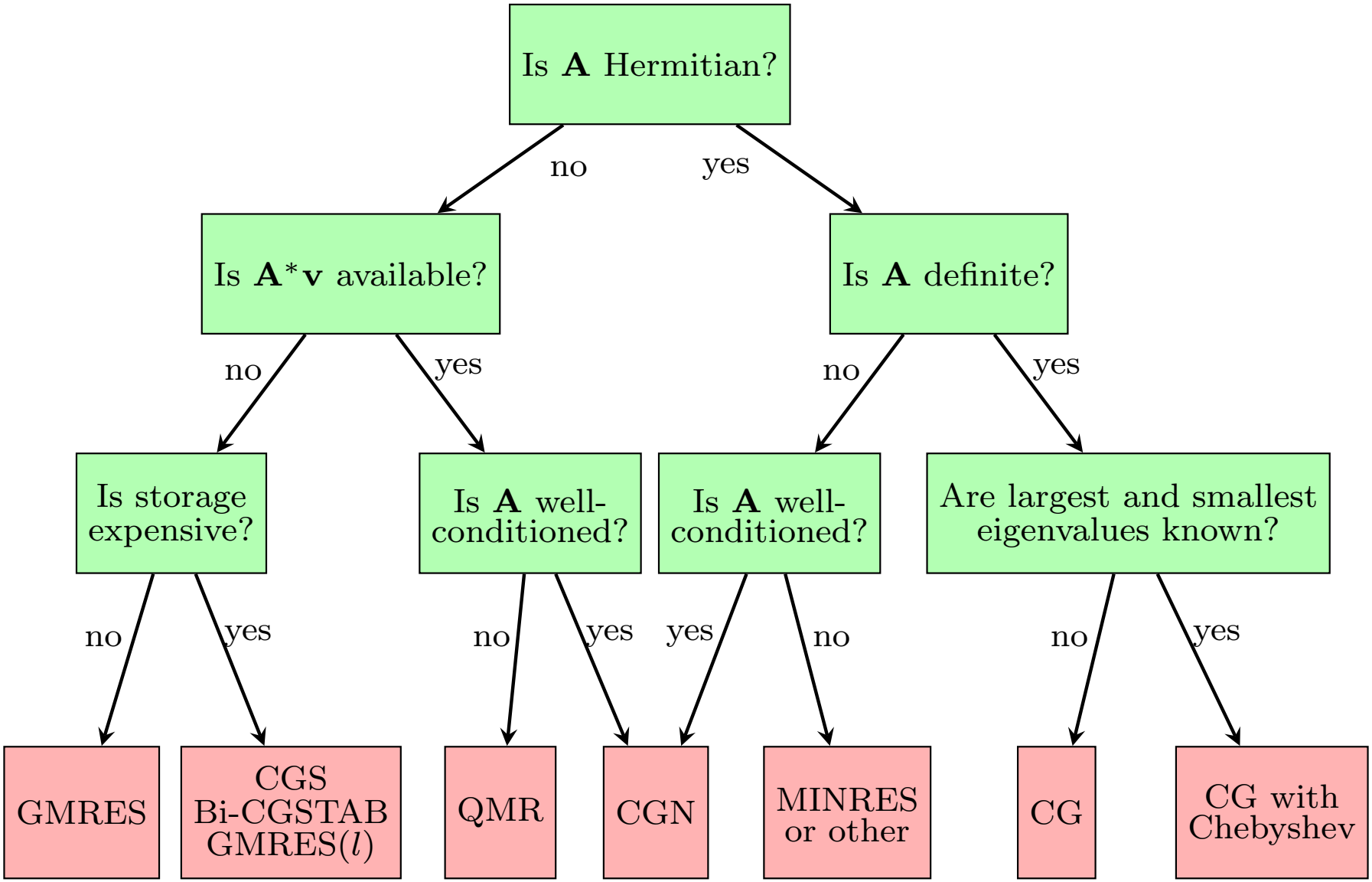
- Gaussian elimination, QR factorization, and most other algorithms of dense linear algebra fit the following pattern: there are $\mathcal{O}(m)$ steps, each requiring $\mathcal{O}(m^2)$ work, for a total work of $\mathcal{O}(m^3)$.

- For iterative methods, the same figures still apply, but now they represent a typical worst-case behavior. When these methods succeed, they may do so by reducing one or both of these factors.
- We see that the number of steps required for convergence to a satisfactory precision typically depends on spectral properties of the matrix $\mathbf{A}$, if the word "spectral" is interpreted broadly. For example: CG for HPD $\mathbf{Ax} = \mathbf{b}$ converges quickly if the eigenvalues of $\mathbf{A}$ are clustered well away from the origin.
- The work per step in a matrix iteration depends mainly on the structure of the matrix and on what advantage is taken of this structure in the $\mathbf{x} \mapsto \mathbf{Ax}$ black box.
- The ideal iterative method in linear algebra reduces the number of steps from $\mathcal{O}(m)$ to $\mathcal{O}(1)$ and the work per step from $\mathcal{O}(m^2)$ to $\mathcal{O}(m)$, reducing the total work from $\mathcal{O}(m^3)$ to $\mathcal{O}(m)$. Such extraordinary speedups do occur in practical problems, but a more typical improvement is perhaps from $\mathcal{O}(m^3)$ to $\mathcal{O}(m^2)$.

6.4. Preconditioning for linear systems

- Diagonal scaling or Jacobi.
- Incomplete Cholesky or LU factorization.
- Coase-grid approximation.
- Local approximation.
- Block preconditioners and domain decomposition.
- Low-order discretization.
- Constant-coefficient or symmetric approximation.
- Splitting of a multi-term operator.
- Dimensional splitting or ADI.
- One step of a classical iterative method.
- Periodic or convolution approximation.
- Unstable direct method.
- Polynomial preconditioners.

6.5 Decision tree for choosing an iterative solver for $\mathbf{Ax} = \mathbf{b}$



Iterative Methods for Linear Systems

R2020a

One of the most important and common applications of numerical linear algebra is the solution of linear systems that can be expressed in the form `A*x = b`. When `A` is a large sparse matrix, you can solve the linear system using iterative methods, which enable you to trade-off between the run time of the calculation and the precision of the solution. This topic describes the iterative methods available in MATLAB® to solve the equation `A*x = b`.

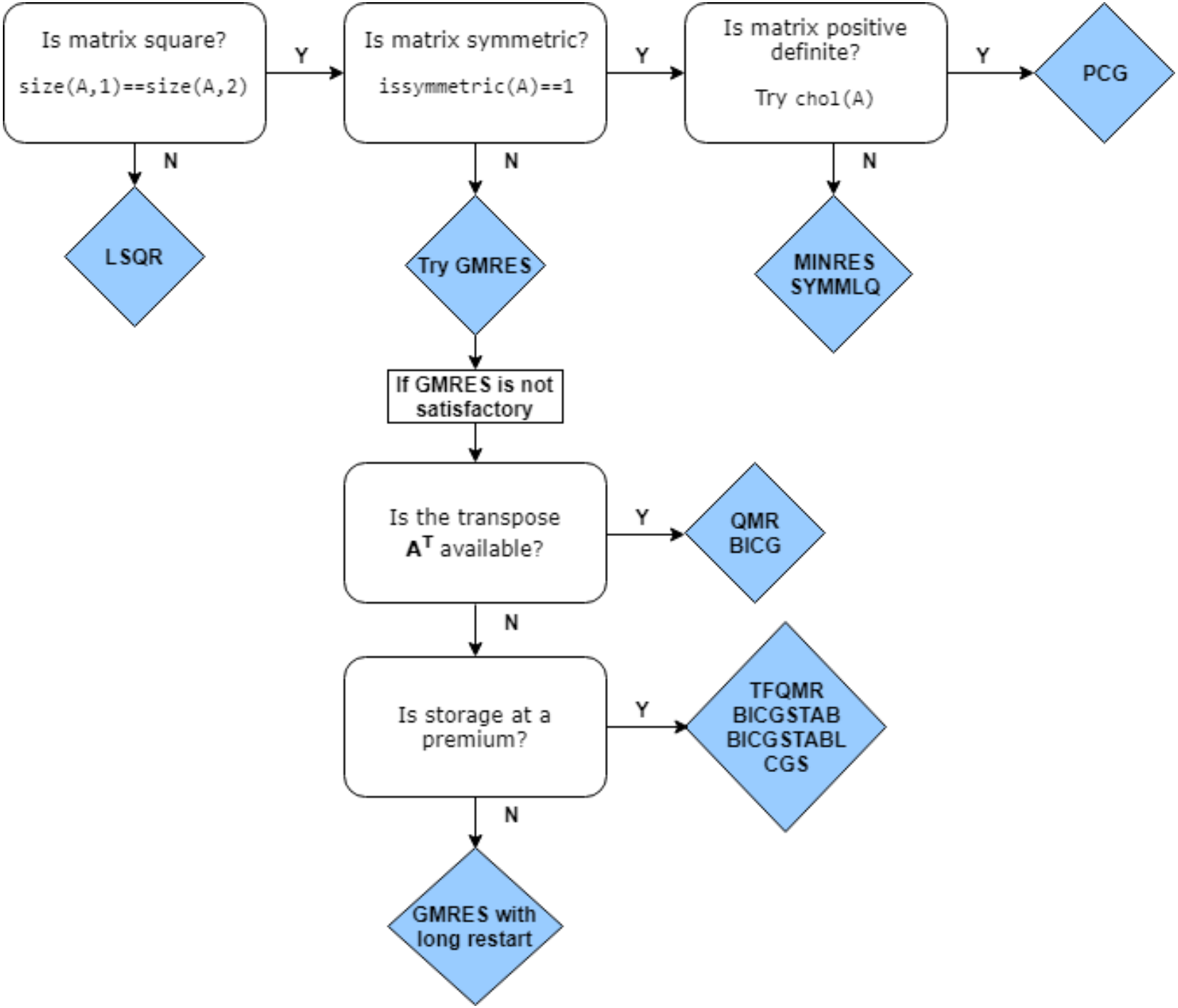
Direct vs. Iterative Methods

There are two types of methods for solving linear equations `A*x = b`:

- **Direct methods** are variants of Gaussian elimination. These methods involve the individual matrix elements directly, through matrix operations such as LU, QR, or Cholesky factorization. You can use direct methods to solve linear equations with a high level of precision, but these methods can be slow when operating on large sparse matrices. The speed of solving a linear system with a direct method strongly depends on the size of the coefficient matrix.

  For example, this code solves a small linear system.

```
A = magic(5);
b = sum(A,2);
x = A\b;
norm(A*x-b)
```


Is matrix square?
size(A,1)==size(A,2)
Y
Is matrix symmetric?
issymmetric(A)==1
Y
Is matrix positive definite?
Try chol(A)
Y
PCG
N
LSQR
N
Try GMRES
N
MINRES
SYMMLQ
If GMRES is not satisfactory
Is the transpose A^T available?
Y
QMR
BICG
N
Is storage at a premium?
Y
TFQMR
BICGSTAB
BICGSTABL
CGS
N
GMRES with long restart