Document: Math 226B (Winter 2012)
Professor: Freund
Latest Update: May 7, 2012
Author: Jeff Irion
http://www.math.ucdavis.edu/~jlirion

Contents

1 1-9-12

`http://www.math.ucdavis.edu/~freund/226B`

Office Hours: MW 12:30-1:30 MSB 2140

1.1 Large Scale Matrix Computations

Typical matrix computations:

- Linear systems: $Ax = b$, A is $n \times n$
- Eigenvalue problem: $Ax = \lambda x$, A is $n \times n$
- Linear programs: $\min c^T x = c_1 x_1 + c_2 x_2 + \cdots + c_n x_n$,
  x: $Ax = b$, A is $n \times n$, $x \geq 0$

Large scale case: n is "large" or n, m are "large."

For all problems that arise in practice, the large matrices exhibit special structures.

Definition 1.1. ***Large-Scale***

A matrix computation problem is called *large-scale* if it can be solved only by methods that exploit its special structures.

special structures $\swarrow\!\!\nearrow$ sparsity / structured dense matrices

1.2 Sparsity

Definition 1.2. ***Sparse***

A matrix $A = [a_{jk}] \in \mathbb{C}^{m \times n}$ is said to be *sparse* if only a small function of its entries a_{jk} are nonzero.

Example 1.3. ***Sparse Matrices***

- Discretization of (elliptic) partial differential equations.

$$Lu = f \quad \rightarrow \quad Ax = b$$
$$(\text{e.g.} \quad \underbrace{-\Delta u}_{-u_{xx}-u_{yy} \text{ in 2-D}} = f \ )$$

$$A = \begin{bmatrix} * & * & * & & * & & & \\ & * & & * & * & * & & \\ & & & & & \ddots & & \\ & & & & & * & * & * \end{bmatrix}$$

- Network problems
- Web search (e.g. the "Google matrix")

Example 1.4. ***Graph of a (Square) Matrix***

Let $A = [a_{jk}] \in \mathbb{C}^{n \times n}$. We can associate with A a directed graph $G(A)$:
Nodes: $N = \{1, 2, \ldots, n\}$
Edges: $E = \{(j,k) \mid j \in N,\ k \in N,\ a_{jk} \neq 0\}$

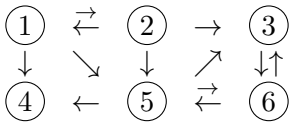
For example:

$$A = \begin{bmatrix} 0 & * & 0 & * & * & 0 \\ * & 0 & * & 0 & * & 0 \\ 0 & 0 & 0 & 0 & 0 & * \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & * & * & 0 & * \\ 0 & 0 & * & 0 & * & 0 \end{bmatrix} \in \mathbb{C}^{6 \times 6}$$

$G(A)$:

$N = \{1, 2, 3, 4, 5, 6\}$

$E = \{(1,2),\ (1,4),\ (1,5),\ (2,1),\ (2,3),\ (2,5),\ (3,6),\ (5,3),\ (5,4),\ (5,6),\ (6,3),\ (6,5)\}$



The graph is directed because the matrix is not symmetric.
$\Rightarrow$ If the matrix is symmetric, there is no need for the graph to be directed.

If $a_{ii} \neq 0$ then the edge is a loop.

2 1-11-12

2.1 The Google Matrix

View the web as a graph G:
$N = \{1, 2, 3, \ldots, n\},\ n =$ number of websites that are visible to the world.

From Example 1.4, we have

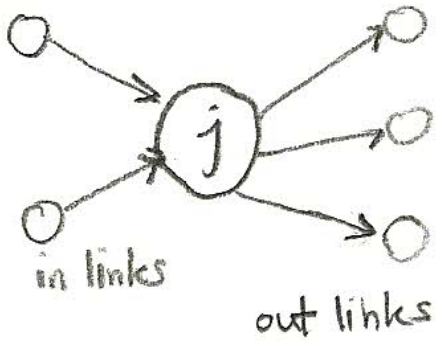
$$d_1 = 3 \quad d_2 = 3 \quad d_3 = 1 \quad d_4 = 0 \quad d_5 = 3 \quad d_6 = 2$$
$$E = \{(j,k) \mid j \in N,\ k \in N, \text{ and there is a link from website } j \text{ to website } k\}$$

Corresponding sparse matrix $Q = [q_{jk}] \in \mathbb{R}^{n \times n}$ such that $q_{jk} \neq 0 \ \Leftrightarrow\ (j,k) \in E$.

Sparsity structure of Q = connectivity of the web.

Values of $q_{jk} \neq 0$?

For each $j \in N$, the *out degree* d_j of j is the number of edges (j,k).



Google's "contribution:"

$$q_{jk} = \begin{cases} \frac{1}{d_j} & (j,k) \in E \\ 0 & \text{otherwise} \end{cases}$$

$$Q = \begin{bmatrix} 0 & \frac{1}{3} & 0 & \frac{1}{3} & \frac{1}{3} & 0 \\ \frac{1}{3} & 0 & \frac{1}{3} & 0 & \frac{1}{3} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{3} & \frac{1}{3} & 0 & \frac{1}{3} \\ 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 \end{bmatrix}$$

The values in row j are $\frac{1}{d_j}$ and their positions are the k corresponding to links.

All rows except row 4 sum up to 1 $\Rightarrow$ a visitor of website 4 will be stuck there!

To fix this:

$$A = [a_{jk}] \in \mathbb{R}^{n \times n}$$
$$a_{jk} = \begin{cases} q_{jk} & d_j > 0 \\ \frac{1}{n} & d_j = 0 \end{cases}$$

A is the "Google matrix:"

$$A = \begin{bmatrix} 0 & \frac{1}{3} & 0 & \frac{1}{3} & \frac{1}{3} & 0 \\ \frac{1}{3} & 0 & \frac{1}{3} & 0 & \frac{1}{3} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \\ 0 & 0 & \frac{1}{3} & \frac{1}{3} & 0 & \frac{1}{3} \\ 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 \end{bmatrix}$$

The Google matrix A is *row stochastic*:

- $a_{jk} \geq 0$ for all $j, k = 1, 2, \ldots, n$
- $\sum_{k=1}^{n} a_{jk} = 1$ for all $j = 1, 2, \ldots, n$

Random walk $\rightarrow$ pages that you are most likely to end up on are ranked higher.

$$Ae = e, \qquad e = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \in \mathbb{R}^n$$

$\lambda = 1$ is an eigenvalue of A with eigenvalue e.

$\Rightarrow \lambda = 1$ is an eigenvalue of A^T (since the eigenvalue of A and A^T are the same).

One can show that there is an eigenvector x such that

$$A^T x = x, \quad x \neq 0$$

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \geq 0 \quad \text{and} \quad x_1 + x_2 + \cdots + x_n = 1$$

The entries of x determine the page rank.

2.2 A Class of Dense Structured Matrices

Definition 2.1. ***Toeplitz Matrix***

A matrix $T \in \mathbb{C}^{n \times n}$ of the form

$$T = \begin{bmatrix} t_0 & t_{-1} & t_{-2} & \cdots & t_{-(n-1)} \\ t_1 & t_0 & t_{-1} & & \\ t_2 & t_1 & \ddots & & \\ \vdots & \ddots & \ddots & \ddots & \\ t_{n-1} & \cdots & t_2 & t_1 & t_0 \end{bmatrix}$$

is called a *Toeplitz matrix*.

Notes

1. $A = [a_{jk}] \in \mathbb{C}^{n\times n}$ is a Toeplitz matrix $\Leftrightarrow a_{jk} = t_{j-k}$ for all $j, k = 1, 2, \ldots, n$
2. An $n \times n$ Toeplitz matrix is dense in general, but we only need to store the $2n - 1$ numbers:

$$t_{-(n-1)},\ t_{-(n-2)},\ \cdots, t_{-2},\ t_{-1},\ t_0,\ t_1,\ t_2,\ \cdots,\ t_{n-1}$$

instead of n^2 numbers.

3 1-13-12

3.1 Toeplitz Matrix Example

Example 3.1.

Let ξ_j be a discrete-time stochastic process.
Time: $j = 1, 2, 3, \ldots, n$
E = expectation of the stochastic process
$m_j = E[\xi_j]$ = mean at time $t = j$
Covariance: $\sigma_{jk} = E[(\xi_j - m_j)\overline{(\xi_k - m_k)}]$
Covariance Matrix: $\Lambda = [\sigma_{jk}] \in \mathbb{C}^{n \times n}$

The stochastic process is said to be *weakly stationary* if

$$\sigma_{jk} = t_{j-k} \qquad \text{for all } j, k = 1, 2, \ldots, n$$

(i.e., 10 and 5 is the same as 11 and 6.)

$\Leftrightarrow \Lambda$ is a Toeplitz matrix.

3.2 Solution of Linear Systems

Problem: Solve $Ax = b$ where $A \in \mathbb{C}^{n \times n}$ is nonsingular and $b \in \mathbb{C}^n$.

3.2.1 Cholesky Factorization

Special case of LU factorization for the case that A is Hermitian positive definite (HPD).

Definition 3.2. ***Hermitian Positve Definite (HPD)***

$A \in \mathbb{C}^{n \times n}$ is *Hermitian positive definite* ($A \succ 0$) if

1. $A = A^H$
2. $x^H A x > 0$ for all $x \in \mathbb{C}^n,\ x \neq 0$

Notes:

1. $\overline{x^H A x} = (x^H A x)^H = x^H A^H x = x^H A x \ \in \mathbb{R}$
   The number is the same as the complex conjugate $\Rightarrow$ real.
2. $A = [a_{jk}] \succ 0 \ \Rightarrow \ a_{jj} = e_j^T A e_j > 0$ (choose $x = e_j$ above) for all j
   $\Rightarrow$ HPD matrices have positive diagonal entries.

Theorem 3.3.

Let $A = [a_{jk}] \in \mathbb{C}^{n\times n}$. Then

1. For any nonsingular $X \in \mathbb{C}^{n\times n}$, $A \succ 0 \;\Leftrightarrow\; X^H A X \succ 0$.
2. If $A \succ 0$, then $\tilde{A} = [a_{jk}]_{j,k\in I} \succ 0$ for all $I \subset \{1, 2, 3, \ldots, n\}$ (any subset).
3. $A \succ 0 \;\Leftrightarrow$ there exists a unique lower triangular matrix

$$L = \begin{bmatrix} l_{11} & 0 & \cdots & 0 \\ l_{21} & l_{22} & \ddots & \vdots \\ \vdots & & \ddots & 0 \\ l_{n1} & \cdots & \cdots & l_{nn} \end{bmatrix}$$

with $l_{jj} > 0,\ j = 1, 2, \ldots, n$ such that

$$A = LL^H \tag{3.1}$$

Notes:

1. (3.1) is called the *Cholesky factorization* of A.
2. No pivoting is needed.

Proof.

1.

$$\begin{aligned} A = A^H \quad &\Leftrightarrow \quad X^H A X = X^H A^H X = (X^H A X)^H \\ 0 < X^H A X &= x^H X^{-H} X^H A X \underbrace{X^{-1} x}_{=\tilde{x}\neq 0} \qquad (x \neq 0) \\ 0 < x^H A x &= \tilde{x}^H (X^H A X) \tilde{x} \end{aligned}$$

2. There is a permutation matrix P such that

$$\begin{aligned} P^T A P = \begin{bmatrix} \tilde{A} & * \\ * & * \end{bmatrix} \succ 0 \qquad &\text{(by part 1)} \\ x = \begin{bmatrix} \tilde{x} \\ 0 \end{bmatrix} \in \mathbb{C}^n, \quad \tilde{x} \neq 0 \quad &\Rightarrow \quad 0 < x^H P^T A P x = \tilde{x}^H \tilde{A} \tilde{x} \end{aligned}$$

3. Induction on n
   $n = 1$:

$$\begin{aligned} A = [a_{11}] \succ 0 \quad &\Rightarrow \quad a_{11} = 0 \\ l_{11} = \sqrt{a_{11}} > 0, \qquad &L := [l_{11}] \\ a_{11} = l_{11}^2 = l_{11}\overline{l_{11}} &= LL^* \end{aligned}$$

$\underline{n-1 \;\Rightarrow\; n}$:

$$A \in \mathbb{C}^{n\times n}, \quad A \succ 0$$

$$A = \begin{bmatrix} a_{11} & --- & w^H & --- \\ | & & & \\ w & & A_{22} & \\ | & & & \end{bmatrix}, \qquad \text{where } a_{11} > 0, \quad w := \begin{bmatrix} a_{21} \\ a_{31} \\ \vdots \\ a_{n1} \end{bmatrix}$$

$$A_{22} := [a_{jk}]_{2\le j,k\le n}$$

$$A = \begin{bmatrix} \sqrt{a_{11}} & 0 \\ \frac{w}{\sqrt{a_{11}}} & I \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & \tilde{A}_{22} \end{bmatrix} \begin{bmatrix} \sqrt{a_{11}} & \frac{w^H}{\sqrt{a_{11}}} \\ 0 & I \end{bmatrix}$$

where $\tilde{A}_{22} := A_{22} - \frac{ww^H}{a_{11}} \in \mathbb{C}^{n-1\times n-1}$.

Part 1 $\Rightarrow \begin{bmatrix} 1 & 0 \\ 0 & \tilde{A}_{22} \end{bmatrix} \succ 0$

Part 2 $\Rightarrow \tilde{A}_{22} \succ 0$

$$\tilde{A}_{22} = \tilde{L}\tilde{L}^H$$

where

$$\tilde{L} = \begin{bmatrix} l_{22} & 0 & \cdots & 0 \\ l_{32} & l_{33} & \ddots & \vdots \\ \vdots & & \ddots & 0 \\ l_{n2} & \cdots & \cdots & l_{nn} \end{bmatrix} \in \mathbb{C}^{n-1\times n-1}, \qquad l_{jj} > 0$$

$$l_{11} = \sqrt{a_{11}}$$

$$\begin{bmatrix} l_{21} \\ \vdots \\ l_{n1} \end{bmatrix} := \frac{w}{\sqrt{a_{11}}}$$

$$A = \begin{bmatrix} l_{11} & 0 \\ l_{21} & \\ \vdots & I \\ l_{n1} & \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & \tilde{L}\tilde{L}^H \end{bmatrix} \begin{bmatrix} l_{11} & \overline{l_{21}} & \cdots & \overline{l_{n1}} \\ & & & \\ 0 & & I & \\ & & & \end{bmatrix}$$

$$= \underbrace{\begin{bmatrix} l_{11} & 0 & \cdots & 0 \\ l_{21} & l_{22} & \ddots & \vdots \\ \vdots & & \ddots & 0 \\ l_{n1} & \cdots & \cdots & l_{nn} \end{bmatrix}}_{=L} L^H = LL^H$$

□

4 1-18-12

4.1 Cholesky Factorization

From last time: for $A \succ 0$, Cholesky factorization gives us

$$A = LL^H$$

We proved this as:

$$A = \begin{bmatrix} a_{11} & \overline{a_{21}} & \cdots & \overline{a_{n1}} \\ a_{21} & & & \\ \vdots & & A_{22} & \\ a_{n1} & & & \end{bmatrix}$$

$$L = \begin{bmatrix} l_{11} & 0 & \cdots & 0 \\ l_{21} & & & \\ \vdots & & \tilde{L} & \\ l_{n1} & & & \end{bmatrix}$$

$$l_{11} = \sqrt{a_{11}}$$

$$l_{21} = \frac{a_{21}}{l_{11}}$$

$$l_{31} = \frac{a_{31}}{l_{11}}$$

$$\vdots$$

$$l_{n1} = \frac{a_{n1}}{l_{11}}$$

$$\tilde{L}\tilde{L}^H = \tilde{A}_{22} = A_{22} - \begin{bmatrix} l_{21} \\ l_{31} \\ \vdots \\ l_{n1} \end{bmatrix} \begin{bmatrix} \overline{l_{21}} & \overline{l_{31}} & \cdots & \overline{l_{n1}} \end{bmatrix}$$

We can translate this proof into an algorithm.

$$A = \begin{bmatrix} a_{11} & * & \cdots & * \\ a_{21} & a_{22} & \ddots & \vdots \\ \vdots & & \ddots & * \\ a_{n1} & \cdots & \cdots & a_{nn} \end{bmatrix} \rightarrow L = \begin{bmatrix} l_{11} & 0 & \cdots & 0 \\ l_{21} & l_{22} & \ddots & \vdots \\ \vdots & & \ddots & 0 \\ l_{n1} & \cdots & \cdots & l_{nn} \end{bmatrix}$$

After $k-1$ steps:

$$L = \begin{bmatrix} l_{11} & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ l_{21} & l_{22} & \ddots & & & & \vdots \\ \vdots & & \ddots & \ddots & & & \vdots \\ \vdots & & & l_{k-1,k-1} & \ddots & & \vdots \\ \vdots & & & & l_{kk} & \ddots & \vdots \\ \vdots & & & & \vdots & \ddots & 0 \\ l_{n1} & \cdots & \cdots & l_{n,k-1} & l_{nk} & \cdots & l_{nn} \end{bmatrix}$$

(The values in black are final values.)

Notation:
For $L = [l_{jk}] \in \mathbb{C}^{n\times n}$,

$$l_{j_1:j_2,k} := \begin{bmatrix} l_{j_1,k} \\ l_{j_1+1,k} \\ \vdots \\ l_{j_2,k} \end{bmatrix}$$

Remark 4.1. ***Algorithm: Cholesky Factorization***

Input: The elements $a_{jk},\ j \geq k$, of $A = [a_{jk}] \in \mathbb{C}^{n\times n},\ A \succ 0$.

Set $l_{jk} = a_{jk}$ for all $j \geq k,\ j,k = 1,2,\ldots,n$.
For $k = 1,2,\ldots,n$, **do:**

- Set $l_{kk} = \sqrt{l_{kk}}$
- Set $l_{k+1:n,k} = \frac{1}{l_{kk}} l_{k+1:n,k}$
- **For** $j = k+1, k+2, \ldots, n$, **do:**
  - Set $l_{j:n,j} = l_{j:n,j} - l_{j:n,k}\overline{l_{jk}}$
- **End**(j)

End(k)

Output: The Cholesky factor $L = [l_{jk}] \in \mathbb{C}^{n\times n}$ of A.

4.2 Cholesky Factorization of Sparse Matrices

Let $A \succ 0$ be sparse. How can we ensure that the Cholesky factor L is also sparse?

Example 4.2. ***"Arrow" Matrices***

$$A = \begin{bmatrix} * & * & \cdots & \cdots & \cdots & * \\ * & * & 0 & \cdots & \cdots & 0 \\ \vdots & 0 & \ddots & \ddots & & \vdots \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \vdots & & \ddots & \ddots & 0 \\ * & 0 & \cdots & \cdots & 0 & * \end{bmatrix} \succ 0 \qquad (\nwarrow) \qquad \Rightarrow \qquad L = \begin{bmatrix} * & 0 & \cdots & \cdots & \cdots & 0 \\ * & * & \ddots & & & \vdots \\ \vdots & * & \ddots & \ddots & & \vdots \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \vdots & & \ddots & \ddots & 0 \\ * & * & \cdots & \cdots & * & * \end{bmatrix}$$

i.e., all sparsity is lost!

Remedy: turn this into a downward-pointing arrow $\Rightarrow$ reorder the rows and columns.

$$1, 2, \ldots, n \quad \rightarrow \quad 2, 3, \ldots, n, 1$$

Permuation matrix:

$$P^T A P = \tilde{A} = \begin{bmatrix} * & 0 & \cdots & \cdots & 0 & * \\ 0 & * & \ddots & & \vdots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ \vdots & & \ddots & \ddots & 0 & \vdots \\ 0 & \cdots & \cdots & 0 & \ddots & * \\ * & \cdots & \cdots & \cdots & * & * \end{bmatrix} \succ 0,$$

where

$$P = \begin{bmatrix} 0 & 0 & \cdots & \cdots & 0 & 1 \\ 1 & \ddots & & & \vdots & \vdots \\ 0 & 1 & \ddots & & \vdots & \vdots \\ \vdots & 0 & \ddots & \ddots & \vdots & \vdots \\ \vdots & \vdots & & \ddots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 1 & 0 \end{bmatrix} \in \mathbb{R}^{n \times n}$$

Then: $P^T A P = \tilde{A} = \tilde{L}\tilde{H}$, where

$$\tilde{L} = \begin{bmatrix} * & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & * & \ddots & & & \vdots \\ \vdots & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & 0 & * & 0 \\ * & * & \cdots & \cdots & * & * \end{bmatrix}$$

$\Rightarrow$ All sparsity is preserved!

But in general, you cannot always expect to find a sparse Cholesky factorization for a sparse matrix.

Remark 4.3. ***Algorithm: Sparse Cholesky Factorization***

Input: sparse matrix $A \succ 0$

1. (Symbolic factorization) Determine a permutation matrix P such that the Cholesky factor L of
$$P^T AP = LL^H$$
is sparse, and determine the sparsity structure of L.
2. (Numerical factorization) Compute the entries of L.

Output: A permutation matrix P and a lower triangular matrix L such that
$$P^T AP = LL^H$$

Question: How to find P?

5 1-20-12

5.1 Cholesky Factorization for Sparse Matrices (Continued)

Suppose we have $A \succ 0$. We want to find P such that

$$P^T AP = LL^H$$

where L is sparse.

Notation:
For a matrix $A = [a_{jk}] \in \mathbb{C}^{m \times n}$, $\text{nnz}(A)$ = number of nonzero entries $a_{jk} \neq 0$ in A.

Optimal choice of P: the Cholesky factor L of $P^T AP$ is such that $\text{nnz}(L)$ is minimal.

Theorem 5.1.

The problem of determining an optimal P is NP-complete.

Consequence: In practice, only heuristics for finding a "good" P are feasible.

The problem of finding P can be viewed as a graph.

Let $A = A^H \succ 0$.

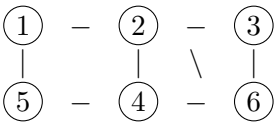
Conventions:

- For $A = A^H$, we view $G(A)$ as an undirected graph.
- For $A \succ 0$, omit edges (j, j) corresponding to $a_{jj} > 0$.

Example 5.2.

$$A = \left[\begin{array}{c|cccc} * & * & 0 & 0 & * & 0 \\ \hline * & * & * & * & 0 & * \\ 0 & * & * & 0 & 0 & * \\ 0 & * & 0 & * & * & * \\ * & 0 & 0 & * & * & 0 \\ 0 & * & * & * & 0 & * \end{array}\right] \in \mathbb{C}^{6\times 6}$$

$G(A)$:



First step of Cholesky factorization:

$$\tilde{A}_{22} = \begin{bmatrix} * & * & * & * & * \\ * & * & 0 & 0 & * \\ * & 0 & * & * & * \\ * & 0 & * & * & 0 \\ * & * & * & 0 & * \end{bmatrix}$$

where the $*$'s indicate fill-in elements.

Let $A = [a_{jk}] \in \mathbb{C}^{n\times n},\ A \succ 0$. The first step of Cholesky factorization is:

$$A \to \tilde{A}_{22} = [\tilde{a}_{jk}]_{j,k=2,3,\ldots,n}$$

where $\tilde{a}_{jk} = a_{jk} - \frac{a_{j1}-\overline{a_{k1}}}{a_{11}}$.

Generic case:
$\tilde{a}_{jk} \neq 0 \ \Leftrightarrow\ a_{jk} \neq 0$ or $(a_{j1} \neq 0$ and $a_{k1} \neq 0)$.
$\tilde{a}_{jk} \neq 0$ is called a *fill-in element* if $a_{jk} = 0$ but $a_{j1} \neq 0$ and $a_{k1} \neq 0$.

Interpretation in terms of the graphs $G(A)$ and $G(\tilde{A}_{22})$:

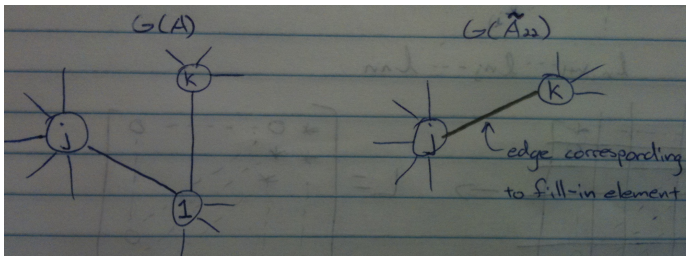


Figure 1: A fill-in element is created because the eliminated node, 1, is connected to both j and k.

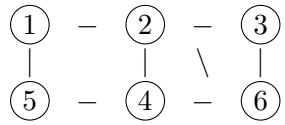
Remark 5.3. ***Minimal Degree Algorithm***

Notation: d_j = (out) degree of node j = number of edges of the form $(j,k),\ k \neq j$.

Order the nodes such that the node corresponding to the kth step of Cholesky factorization has minimal degree.

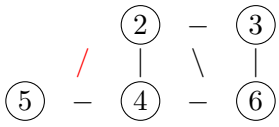
Example 5.4. ***Example 5.2 Continued***

$G^0 := G(a)$



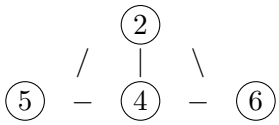
The lowest degree is 2, which occurs at nodes 1, 3, and 5.
Tie-breaker: If more than one node has minimal degree, pick the node with the lowest node number.
Thus, we choose node 1 and eliminate it:

G^1 :



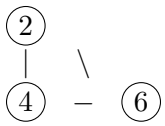
Eliminate node 3:

G^2 :

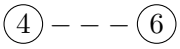


Eliminate node 5:

G^2 :



Eliminate node 2:



Eliminate node 4:

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$P^T A P = L L^H$$

New ordering: 1,3,5,2,4,6.
$\text{nnz}(LL^H) = \text{nnz}(A) + 2$.

6 1-23-12

6.1 Minimal Degree Algorithm (Continued)

General Case:
Let $G = (N, E)$ be an undirected graph and $i \in N$.

$$G_i := (N_i, E_i), \qquad \text{where } N_i := N \setminus \{i\}$$

and

$$E := \{(j,k) \in E \mid j \neq i \text{ and } k \neq i\} \cup \{(j,k) \notin E \mid j \neq i,\ (j,i) \in E,\ k \neq i,\ (k,i) \in E\}$$

Remark 6.1. ***Minimal Degree Algorithm***

Input: The undirected graph $G^0 = (N^0, E^0)$ associated with $A \in \mathbb{C}^{n\times n},\ A \succ 0$.

For $k = 1, 2, \ldots, n$:

1. Determine a node $i_k \in N^{k-1}$ of minimal degree in $G^{k-1} = (N^{k-1}, E^{k-1})$. (Possible tie-breaker: smallest i_k)
2. Set $G^k = (N^k, E^k) := G^{k-1}_{i_k}$

End k

Output: a reordering of the n rows and columns of A,

$$\underbrace{1, 2, 3, \ldots, n}_{A=LL^H} \quad \rightarrow \quad \underbrace{i_1, i_2, i_3, \ldots, i_n}_{P^TAP=LL^H}$$

Notes:

1. In general, the minimal degree ordering does not minimize the sparsity of the Cholesky factor L, i.e. it is not optimal.
2. There are many other heuristics for reordering the rows and columns of A.

6.2 LU Factorization

Let $A \in \mathbb{C}^{n\times n}$, A nonsingular.

Special case: no pivoting needed.

$$A = LU$$

$$L = \begin{bmatrix} 1 & & & 0 \\ l_{21} & \ddots & & \\ \vdots & \ddots & \ddots & \\ l_{n1} & \cdots & l_{n,n-1} & 1 \end{bmatrix}$$

$$U = \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ & u_{22} & \ddots & \vdots \\ & & \ddots & u_{n-1,n} \\ 0 & & & u_{nn} \end{bmatrix}$$

LU Algorithm (No Pivoting)

$A \in \mathbb{C}^{n \times n},\ A \to U,\ I \to L.$

After $k-1$ steps:

$$L = \begin{bmatrix} 1 & & & & & & 0 \\ l_{21} & \ddots & & & & & \\ \vdots & \ddots & 1 & & & & \\ \vdots & & l_{k,k-1} & \ddots & & & \\ \vdots & & \vdots & 0 & \ddots & & \\ \vdots & & \vdots & \vdots & \ddots & \ddots & \\ l_{n1} & \cdots & l_{n,k-1} & 0 & \cdots & 0 & 1 \end{bmatrix}$$

$$U = \begin{bmatrix} u_{11} & \cdots & \cdots & \cdots & \cdots & u_{1n} \\ 0 & \ddots & & & & \vdots \\ \vdots & \ddots & u_{k-1,k-1} & \cdots & \cdots & u_{k-1,n} \\ \vdots & & 0 & u_{kk} & \cdots & u_{kn} \\ \vdots & & \vdots & u_{k+1,k} & \cdots & u_{k+1,n} \\ \vdots & & \vdots & \vdots & \ddots & \ddots \\ 0 & \cdots & 0 & u_{nk} & \cdots & u_{nn} \end{bmatrix}$$

where the values in black are final values.

Remark 6.2. ***Algorithm: LU Factorization Without Pivoting***

Input: $A \in \mathbb{C}^{n\times n}$

Set $U = A, L = I$.

For $k = 1, 2, \ldots, n-1$, **do**

- **If** $u_{kk} = 0$: stop, pivoting is needed, or A is singular (and thus pivoting will not help)
- **For** $j = k+1, k+2, \ldots, n$, **do**
  - Set $l_{jk} = \frac{u_{jk}}{u_{kk}}$
  - Set $u_{j,k:n} = u_{j,k:n} - l_{jk} u_{k,k:n}$
- **End** j

End k

Output: L and U such that

$$A = LU$$

General case:

1. $u_{kk} = 0$ can occur even if A is nonsingular
2. Numerical instability if $u_{kk} \neq 0$, but $|u_{kk}| \ll |u_{jk}|$ ($\Rightarrow$ we create a very large quotient of these numbers)

Remedies:
To be done in each k-th step.

1. Partial pivoting: find an $r \in \{k, k+1, \ldots, n\}$ such that

   $$|u_{rk}| = \max_{i=k,k+1,\ldots,n} |u_{ik}|$$

   and interchange rows r and k. Thus, the final factorization is

   $$PA = LU$$

   where P is a permutation matrix. This is all that people ever use for dense matrices.

2. Complete pivoting: find $r, c \in \{k, k+1, \ldots, n\}$ such that

   $$|u_{rc}| = \max_{i,l=k,k+1,\ldots,n} |u_{il}|$$

   and interchange rows r and k and columns c and k.

Result:

$$PAQ = LU,$$

where P and Q are permutation matrices.

7 1-25-12

Remarks about the Homework:

- Problem 3 has been clarified

$$A$$
$$Ax$$
$$A^T,\ A_\alpha^T$$
$$A = Q + \frac{1}{n} v e^T$$
$$A^T x = Q^T x + \frac{1}{n} e v^T x$$

Don't store A. Q you can store.

$$\frac{x}{x_j} = \tilde{x}, \qquad \|\tilde{x}\|_\infty = 1, \quad \tilde{x}_j = 1$$
$$|x_j| = \|x\|_\infty$$
$$\frac{x}{\|x\|_\infty} = \tilde{\tilde{x}}$$

You need to write a routine that figures out what x_j is $\Rightarrow$ find the largest element of x and its index.

$$\tilde{\tilde{x}}_k = \pm 1$$

7.1 Sparse LU Factorization

Let $A \in \mathbb{C}^{n \times n}$ be sparse.
Goal: LU factorization

$$PAQ = LU,$$

where L and U are sparse.
Difficulty: In general, P and Q cannot be determined by symbolic factorization alone, since we also need to pivot for stability!
Instead: P and Q are determined during the actual (numerical) factorization.
Note: Symbolic factorization (minimal degree, ...) can be used as a preprocessing step:

$$A, G(A) \quad \rightarrow \quad \text{permutation matrices } P_0 \text{ and } Q_0 \text{ (not final)}$$

Run sparse LU factorization on the reordered version of A: $P_0 A Q_0$.
Step k of sparse LU factorization:

$$U^{(k)} = [u_{il}]_{i,l=k,k+1,\ldots,n} = \begin{bmatrix} u_{kk} & u_{k,k+1} & \cdots & \cdots & \cdots & u_{kn} \\ u_{k+1,k} & u_{k+1,k+1} & \cdots & \cdots & \cdots & u_{k+1,n} \\ \vdots & \vdots & & & & \vdots \\ \vdots & \vdots & & u_{il} \neq 0 & & \vdots \\ \vdots & \vdots & & & & \vdots \\ u_{nk} & u_{n,k+1} & \cdots & \cdots & \cdots & u_{nn} \end{bmatrix}$$

Swap rows k and l and columns k and i.

Any entry $u_{il} \neq 0$ of the submatrix is a candidate for the kth *pivot element*.

Example 7.1.

$$\begin{bmatrix} \textcircled{*} & 0 & * & 0 & 0 & * \\ * & * & 0 & * & 0 & * \\ 0 & \textcircled{*} & * & 0 & 0 & 0 \\ * & 0 & 0 & 0 & * & 0 \\ * & 0 & 0 & 0 & * & 0 \\ * & 0 & * & * & 0 & * \end{bmatrix}$$

5 fill-in elements (red)

swap rows 1 and 3 and swap columns 1 and 2

$$\begin{bmatrix} \textcircled{*} & 0 & * & 0 & 0 & 0 \\ * & * & 0 & * & 0 & * \\ 0 & * & * & 0 & 0 & * \\ 0 & * & 0 & 0 & * & 0 \\ 0 & * & 0 & 0 & * & 0 \\ 0 & * & * & * & 0 & * \end{bmatrix}$$

only 1 fill-in element

7.2 Markowitz Criterion

$r_i = r_i^{(k)} = \#$ of nonzero entries in the ith row $u_{i,k:n}$ of $U^{(k)}$
$c_l = c_l^{(k)} = \#$ of nonzero entries in the lth column $u_{k:n,l}$ of $U^{(k)}$

If u_{il} is used as the pivot element, then in the worst case the number of fill-in elements we're creating in step k is

$$(r_i^{(k)} - 1)(c_l^{(k)} - 1). \tag{7.1}$$

Basic Idea: Choose i, l to minimize (7.1).

General Case: To guarantee numerical stability, we need to make sure that u_{il} is not "too" small.

Remark 7.2. ***Practical Markowitz Criterion***

Among all $u_{il} \neq 0$, $i, l \in \{k, k+1, \ldots, n\}$, with

$$|u_{il}| \geq \alpha \max_{j \geq k} |u_{ij}| \qquad \text{(row)}$$

or

$$|u_{il}| \geq \alpha \max_{j \geq k} |u_{jl}| \qquad \text{(column)},$$

choose u_{il} such that

$$(r_i - 1)(c_l - 1)$$

is minimal. Tie-breaker: choose the smallest i, then (if necessary) choose the smallest l.

Here, α is a parameter, $0 < \alpha \leq 1$. Typical choice is $\alpha = 0.1$.

8 1-27-12

8.1 Storage of Sparse Matrices

Let $A = [a_{jk}] \in \mathbb{C}^{m\times n}$ $(\in \mathbb{R}^{m\times n})$ be sparse, with nnz := nnz(A) nonzero entries $a_{jk} \neq 0$.

Definition 8.1. ***Coordinate (COO) Format***

(used in Matlab)

Three arrays:

1. VA: complex (or real) array of length nnz that contains the values $a_{jk} \neq 0$ in any order
2. JA: integer array of length nnz that contains the (row) indices j in the same order as in VA
3. KA: integer array of length nnz that contains the (column) indices k in the same order as in VA

Then $a_{JA(i),KA(i)} = VA(i),\ i = 1, 2, \ldots, \text{nnz}$.

Example 8.2.

$$A = \begin{bmatrix} 0 & 1.27 & -1.5 & 0 & 3 \\ 2.3 & 0 & 0 & -5 & 0 \\ 0 & -7.1 & 0 & 0 & 2 \\ 3.3 & -2 & 0 & 1.2 & 0 \end{bmatrix} \in \mathbb{R}^{4\times 5}$$

nnz $= 10$

VA	2.3	-5	1.27	1.2	3.3	-7.1	-1.5	2	-2	3
JA	2	2	1	4	4	3	1	3	4	1
KA	1	4	2	4	1	2	3	5	2	5

One thing that is convenient about this format is that if we have new nonzero entries, we can simply tack them on.

A downside is that working with the matrix is not straightforward.

In practice, people don't use this format. What we want is a format where we can easily find elements in the same row (or column).

Definition 8.3. ***Compressed Sparse Row (CSR) Format***

Using Example (8.2):

VA	1.27	−1.5	3	−5	2.3	−7.1	2	1.2	−2	3.3	
KA	2	3	5	4	1	2	5	4	2	1	
IA	1			4		6		8			11 = nnz +1

(If we have a row of all zeros, then that row's number is repeated in KA.)

General case
3 Arrays:

1. VA: complex (or real) array of length nnz that contains the values $a_{jk} \neq 0$ stored row by row; within each row, the order is arbitrary
2. KA: integer array that contains the corresponding column indices k in the same order as in VA
3. IA: integer array of length $m+1$ ($A \in \mathbb{R}^{m\times n}$) that contains pointers to the beginning of each row in VA and KA
   - IA(j) = index of first entry of row j, $j = 1, 2, \ldots, m$
   - IA($m+1$) := IA(1) + nnz

Note:

- nonzero = "potentially" nonzero ⇒ sometimes we store a zero in VA, e.g. subtract two sparse matrices with the same structure and with a matching entry

Properties:

- IA($j+1$)-IA(j) = # of nonzero entries in the jth row, $a_{j,1:n}$ of A, $j = 1, 2, \ldots, m$
- For all $j = 1, 2, \ldots, m$, the nonzero entries in row j are given by

$$a_{j,KA(i)} = VA(i), \qquad i = IA(j),\ IA(j)+1,\ \ldots, IA(j+1)-1.$$

8.2 Fast Elliptic Solvers

Large sparse systems,

$$Av = b,$$

often exhibit special structures that can be exploited in their solution.

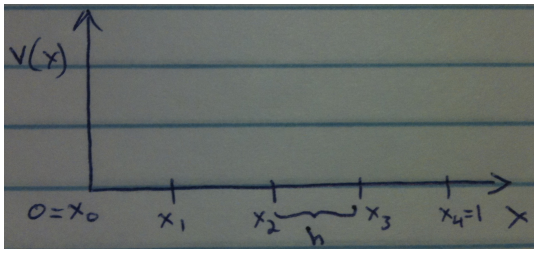
Standard example: Poisson's equation (on simple domains).

$$-\frac{\partial^2 u}{\partial x^2} - -\frac{\partial^2 u}{\partial y^2} = f(x,y), \qquad 0 < x, y < 1$$
$$u = 0 \qquad x = 0, 1 \text{ or } y = 0, 1$$

9 1-30-12

9.1 Poisson Equation: 1D

$$-\frac{d^2v(x)}{dx^2} = f(x), \quad 0 < x < 1 \tag{9.1}$$
$$v(0) = v(1) = 0$$



$$\begin{aligned}
h &= \frac{1}{m+1} \\
x_j &= jh = \frac{j}{m+1}, \qquad j = 0, 1, 2, \ldots, m+1 \\
v_j &\approx v(x_j) \\
-\left.\frac{d^2v(x)}{dx^2}\right|_{x=x_j} &\approx \frac{2v(x_j) - v(x_{j+1}) - v(x_{j-1})}{h^2} \quad \Rightarrow \quad \textit{centered-difference approximation} \\
&\approx \frac{2v_j - v_{j+1} - v_{j-1}}{h^2}
\end{aligned}$$

We get an approximate version of (9.1):

$$\begin{aligned}
2v_j - v_{j-1} - v_{j+1} &= h^2 f_j \qquad (f_j := f(x_j)) \\
v_0 &= 0 \\
v_{m+1} &= 0
\end{aligned}$$

We have m linear equations for m unknowns. We can write this in the compact form:

$$\underbrace{\begin{bmatrix} 2 & -1 & 0 & \cdots & \cdots & 0 \\ -1 & 2 & -1 & 0 & \cdots & 0 \\ 0 & -1 & 2 & -1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & \ddots & \ddots & -1 \\ 0 & \cdots & \cdots & 0 & -1 & 2 \end{bmatrix}}_{=:T_m} \underbrace{\begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_m \end{bmatrix}}_{=:v} = h^2 \underbrace{\begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_m \end{bmatrix}}_{=:f}$$

T_m is real and symmetric positive definite, so we can use banded Cholesky and solve it easily. The solution v of the system is:

$$v_j \approx v(x_j), \qquad j = 1, 2, \ldots, m.$$

This scheme is second order accurate:

$$v_j - v(x_j) = O(h^2).$$

This accuracy comes from our discretization scheme. We could use a higher order approximation, but we would not have such an easy matrix to work with.

Lemma 9.1.

The eigenvalues λ_l and the eigenvectors z_l of T_m are given by

$$\lambda_l = \underbrace{2\left(1 - \cos\frac{\pi l}{m+1}\right)}_{\geq 0}, \qquad l = 1, 2, \ldots, m$$

$$z_l = \sqrt{\frac{2}{m+1}} \begin{bmatrix} \sin\frac{l\pi}{m+1} \\ \sin\frac{2l\pi}{m+1} \\ \vdots \\ \sin\frac{ml\pi}{m+1} \end{bmatrix}$$

The z_l's are orthonormal: $z_l^T z_j = \delta_{lj}$.

Proof. Verify that $T_m z_l = \lambda_l z_l$. □

Compact formulation:

$$T_m Z = Z\Lambda,$$

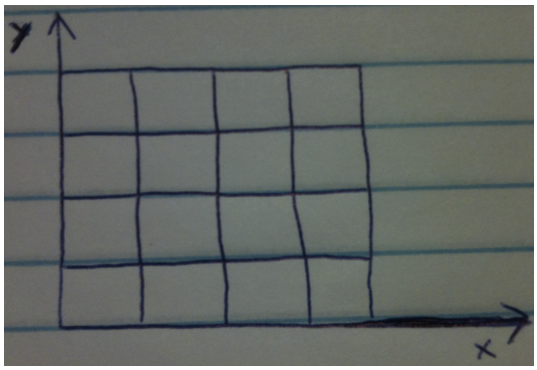
where $Z^T Z = ZZ^T = I$, $Z = \begin{bmatrix} z_1 & z_2 & \cdots & z_m \end{bmatrix} \in \mathbb{R}^{m\times m}$, $\Lambda = \operatorname{diag}(\lambda_1, \lambda_2, \ldots, \lambda_m) \in \mathbb{R}^{m\times m}$.

Corollary 9.2.

$$Z^T T_m Z = \Lambda$$

9.2 Poisson Equation: 2D

$$-\frac{\partial^2 v(x,y)}{\partial x^2} - \frac{\partial^2 v(x,y)}{\partial y^2} = f(x,y), \qquad 0 < x, y < 1$$
$$v = 0 \text{ on the boundary}$$



$$
\begin{aligned}
h &= \frac{1}{m+1} \\
x_j &= jh = \frac{j}{m+1} \\
y_k &= kh = \frac{k}{m+1} \\
v_{jk} &\approx v(x_j, y_k) \\
f_{jk} &= f(x_j, y_k)
\end{aligned}
$$

Centered-difference approximation:

$$
4v_{jk} - v_{j-1,k} - v_{j+1,k} - v_{j,k+1} - v_{j,k-1} = h^2 f_{jk}, \qquad j,k = 1,2,\ldots,m \tag{9.2}
$$

where $v_{0k} = v_{m+1,k} = v_{j0} = v_{j,m+1} = 0$.
Compact formulation of (9.2):

$$
T_m V + V T_m = h^2 F \tag{9.3}
$$

where $F := [f_{jk}]_{j,k=1,2,\ldots,m} \in \mathbb{R}^{m\times m}$ is given, and $V := [v_{jk}]_{j,k=1,2,\ldots,m} \in \mathbb{R}^{m\times m}$ is unknown. (9.3) represents a system of linear equations:

$$
v = \begin{bmatrix} v_{11} \\ v_{21} \\ \vdots \\ v_{m1} \\ v_{21} \\ v_{22} \\ \vdots \\ v_{mm} \end{bmatrix} \in \mathbb{R}^n, \quad \text{where } n = m^2, \qquad f = \begin{bmatrix} f_{11} \\ f_{21} \\ \vdots \\ f_{m1} \\ f_{21} \\ f_{22} \\ \vdots \\ v_{mm} \end{bmatrix} \in \mathbb{R}^n, \quad \underbrace{\begin{bmatrix} T_m + 2I_m & -I_m & & 0 \\ -I_m & T_m + 2I_m & \ddots & \\ & \ddots & \ddots & -I_m \\ 0 & & -I_m & T_m + 2I_m \end{bmatrix}}_{=:T_{m\times m} \in \mathbb{R}^{n\times n}}
$$

10 2-1-12

10.1 Poisson Equation (Continued)

$$\begin{aligned} T_m V + V T_m &= h^2 F \qquad (10.1)\\ Z^T T_m Z &= \Lambda = \operatorname{diag}(\lambda_1, \dots, \lambda_m)\\ Z Z^T &= I\\ h &= \frac{1}{m+1} \end{aligned}$$

Rewrite (10.1) as

$$\begin{aligned} \underbrace{Z^T T_m Z}_{\Lambda}\underbrace{Z^T V Z}_{V'} + \underbrace{Z^T V Z}_{V'}\underbrace{Z^T T_m Z}_{\Lambda} &= h^2 \underbrace{Z^T F Z}_{F'}\\ \Lambda V' + V'\Lambda &= h^2 F'\\ \lambda_j v'_{jk} + v'_{jk}\lambda_k &= h^2 f'_{jk} \quad \text{for all } j,k = 1,2,\dots,m\\ v'_{jk} &= \frac{h^2 f'_{jk}}{\lambda_j + \lambda_k} \quad \text{for all } j,k = 1,2,\dots,m \end{aligned}$$

Remark 10.1. ***Algorithm (for solving the two-dimensional Poisson equation)***

Input: F, h

1. Set $F' = Z^T F Z$
2. Set $v'_{jk} = \frac{h^2 f'_{jk}}{\lambda_j + \lambda_k}$ for all $j,k = 1,2,\dots,m$
3. Set $V = Z V' Z^T$

Flop count (naive implementation):

1. 2 matrix-matrix multiplications in $\mathbb{R}^{m\times m} \approx 4m^3$ flops
2. $3m^2$ flops
3. $\approx 4m^3$ flops (same as 1)

Total $= O(m^3) = O(n^{3/2})$ flops, where $n = m^2$

Totally naive solution: solve

$$T_{m\times m} v = h^2 f,$$

where $T_{m\times m} \in \mathbb{R}^{m^2\times m^2}$, $n = m^2$. This is $O(n^3)$ flops.

We can do better than $O(n^{3/2})$!

Recall: $Z = [z_{jk}] \in \mathbb{R}^{m\times m}$, where $z_{jk} := \sqrt{\frac{2}{m+1}} \sin\frac{jk\pi}{m+1}$.

Notes:

1. $Z = Z^T$

2. Z is related to the $2m+2 \times 2m+2$ DFT (*discrete Fourier transform*) matrix:

$$\Phi = [\phi_{jk}]_{j,k=0,1,2,\ldots,2m+1} \in \mathbb{C}^{2m+2\times 2m+2}$$
$$\phi_{jk} = e^{-jk\pi i/(m+1)} = \cos\frac{jk\pi}{m+1} - i\underbrace{\sin\frac{jk\pi}{m+1}}$$

In fact:

$$\Phi = \left[\begin{array}{c|c|c} *_{1\times 1} & *_{1\times m} & *_{1\times m+1} \\ \hline *_{m\times 1} & \hat{Z}_{m\times m} & *_{m\times m+1} \\ \hline *_{m+1\times 1} & *_{m+1\times m} & *_{m+1\times m+1} \end{array}\right]$$

where $\hat{Z} = [\hat{z}_{jk}]_{j,k=1,2,\ldots,m}$, $\hat{z}_{jk} := \cos\frac{jk\pi}{m+1} - i\sin\frac{jk\pi}{m+1}$. Thus,

$$Z = -\sqrt{\frac{2}{m+1}}\,\mathrm{Im}\,\hat{Z}$$

Proposition 10.2.

For any $x \in \mathbb{R}^m$, the matrix-vector product $y = Zx = Z^T x$ can be computed as follows:

1. $$\tilde{x} = \begin{bmatrix} 0_{1\times 1} \\ x_{m\times 1} \\ 0_{m+1\times 1} \end{bmatrix} \in \mathbb{R}^{2m+2}$$
2. Compute $\tilde{y} = \Phi\tilde{x}$, where Φ is the $2m+2 \times 2m+2$ DFT matrix.
3. $$\tilde{y} = \begin{bmatrix} *_{1\times 1} \\ \hat{y}_{m\times 1} \\ *_{m+1\times 1} \end{bmatrix}$$
   $$\hat{y} = \hat{Z}x$$
4. Set
   $$y = -\sqrt{\frac{2}{m+1}}\,\mathrm{Im}\,\hat{y}.$$

Corollary 10.3.

The product $y = Zx = Z^T x$ can be computed with $O(m\log m)$ flops using DFT.

Notes:

1. In Matlab, $\tilde{y} = \Phi\tilde{x} \;\Leftrightarrow\; \tilde{y} = \texttt{fft}(\tilde{x})$

2. For any $F \in \mathbb{R}^{m \times m}$, we can compute $F' = Z^T F Z = Z^T (Z^T F^T)^T$ with $O(m^2 \log m) = O(n \log n)$ flops using DFT's.

3. With these fast DFT computations, the above algorithm requires $O(n \log n)$ flops!

    - This is almost optimal! An optimal algorithm (multi-grid) requires $O(n)$ flops. ($n = m^2 =$ number of grid points)

11 2-3-12

11.1 Iterative Methods for the Solution of Linear Systems

$$Ax = b, \qquad A \in \mathbb{C}^{n\times n},\ \text{nonsingular},\ b \in \mathbb{C}^n$$

There are two types of methods:

1. Direct (e.g. LU factorization, Cholesky, fast elliptic solvers, etc.)
2. Iterative (e.g. Krylov subspace methods, multigrid, etc.)

11.2 Krylov Subspace Methods

If A is sparse with nnz potentially nonzero entries, then computing

$$y = Ax$$

for any $x \in \mathbb{C}^n$ is cheap: at most nnz multiplications and at most nnz additions.

Krylov subspace methods exploit this fact!

11.2.1 The conjugate gradient (CG) method

Assumption: $A \succ 0$ ($A \succ 0 \ \Leftrightarrow\ A = A^H$ and $p^H Ap > 0$ for all $p \in \mathbb{C}^n,\ p \neq 0$)

CG is the classical Krylov subspace method. It is an iterative method:

$$x_0 \in \mathbb{C}^n \ \to\ x_1 \ \to\ x_2 \ \to\ \cdots \ \to\ x_k \in \mathbb{C}^n \ \to\ \cdots$$

Notation

- x_k is the k-th iterate.
- $t_k := b - Ax_k$ is the corresponding residual vector
  - Note: $t_k = 0 \Leftrightarrow x_k = A^{-1}b =: x^*$ is the solution of $Ax = b$

Goal: Construct x_k such that $\|t_k\|$ is small, where $\|\cdot\|$ is some appropriate norm in $\mathbb{C}^n$.

$A \succ 0 \Rightarrow \|x\|_A := \sqrt{x^H Ax}$ is a norm on $\mathbb{C}^n$.
$A^{-1} \succ 0 \ \Rightarrow\ \|r\|_{A^{-1}} := \sqrt{r^H A^{-1} r}$ is a norm on $\mathbb{C}^n$.

Note: $\|x\|_A = \|Ax\|_{A^{-1}}$ for all $x \in \mathbb{C}^n$.
The CG method is based on the error norm

$$\|x^* - x\|_A = \|Ax^* - Ax\|_{A^{-1}} = \|b - Ax\|_{A^{-1}}$$

Suppose we have $x_k \in \mathbb{C}^n$, and we want to construct

$$x_{k+1} = x_k + \alpha_k p_k,$$

where $p_k \in \mathbb{C}^n,\ p_k \neq 0$ is a search direction and $\alpha_k \in \mathbb{R},\ \alpha_k > 0$.

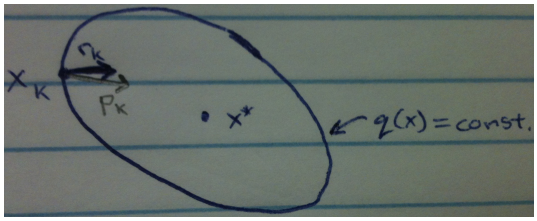


Figure 2: This is an ellipsoid (see below).

$$\begin{aligned} & \|x^* - x_k\|_A = \|x^* - x\|_A \\ \Leftrightarrow \quad & q(x_k) = q(x), \\ \text{where} \quad & q(x) := \frac{1}{2}(x^* - x)^H A(x^* - x) \\ & \nabla q(x) = 0 = -(A(x^* - x)) \end{aligned}$$

Steepest descent:

$$p_k = -\nabla q(x_k) = A(x^* - x_k) = b - Ax_k = r_k$$

We can do better. Instead of the negative gradient, use the conjugate gradient.

$$p_0 = r_0 \ (= b - Ax_0)$$

For $k = 0, 1, \ldots,$

$$\begin{aligned} & p_{k+1} = r_{k+1} + \beta_{k+1} p_k, \\ \text{where} \quad & \beta_{k+1} = \frac{r_{k+1}^H r_{k+1}}{r_k^H r_k} \end{aligned}$$

End (k)

One can show $p_j^H A p_k = 0$ for all $j \neq k$.

Note: The choice of $\beta_{k+1} \ \Leftrightarrow \ p_{k+1}^H A p_k = 0$.

The iterates are

$$\begin{aligned} & x_{k+1} = x_k + \alpha_k p_k, \\ \text{where} \quad & \alpha_k = \frac{r_k^H r_k}{p_k^H A p_k}. \end{aligned}$$

Remark 11.1.

In n dimensions, this algorithm will converge in n iterations.

12 2-6-12

12.1 Conjugate Gradient (Continued)

Remark 12.1. ***Algorithm: Conjugate Gradient Method***

Inputs:

- A routine to compute $z = Ap$ for any $p \in \mathbb{C}^n$ ($A \in \mathbb{C}^{n\times n}$, $A \succ 0$)
- $b \in \mathbb{C}^n$
- $x_0 \in \mathbb{C}^n$ (arbitrary)
- convergence tolerance `tol`, typically 10^{-6} or 10^{-9}

Set $r_0 = b - Ax_0$, $p_0 = r_0$.
for $k = 0, 1, 2, \ldots,$ **do:**

- If $\frac{\|r_k\|_2}{\|r_0\|_2} \leq$ `tol`, **stop:** $x_k \approx A^{-1}b$
- Set

$$
\begin{aligned}
z &= Ap_k \\
\alpha_k &= \frac{r_k^H r_k}{p_k^H z} \quad \left(= \frac{\|r_k\|^2}{p_k^H A p_k} > 0\right) \\
x_{k+1} &= x_k + \alpha_k p_k \\
r_{k+1} &= b - Ax_{k+1} = \underbrace{b - Ax_k}_{r_k} - \alpha_k \underbrace{Ap_k}_{z} \\
&= r_k - \alpha_k z \ (= b - Ax_{k+1}) \\
\beta_{k+1} &= \frac{r_{k+1}^H r_{k+1}}{r_k^H r_k} \quad \left(= \frac{\|r_{k+1}\|_2^2}{\|r_k\|_2^2}\right) \\
p_{k+1} &= r_{k+1} + \beta_{k+1} p_k \qquad \text{(Recall: } p_{k+1}^H A p_j = 0,\ j = 0, 1, \ldots, k)
\end{aligned}
$$

end (k)

Notes:

1. In exact arithmetic, $r_k = b - Ax_k$
2. Each k-th iteration involves the following operations:
   - 1 matrix-vector product: $z = Ap$
   - 2 inner products in $\mathbb{C}^n$: $p_k^H z$ and $r_{k+1}^H r_{k+1}$
   - 3 *SAXPY*'s (**S**ingle-precision real **A**lpha **X** **P**lus **Y**):

$$
\begin{aligned}
x_{k+1} &= x_k + \alpha_k p_k \\
r_{k+1} &= r_k - \alpha_k z \\
p_{k+1} &= r_{k+1} - \beta_k p_k
\end{aligned}
$$

3. Conjugate Gradient is a Krylov subspace method:

$$x_1 = x_0 + \alpha_0 \underbrace{p_0}_{=r_0} \in x_0 + \text{span}\{r_0\}$$

$$x_2 = x_1 + \alpha_1 p_1 = x_0 + \alpha_0 r_0 + \alpha_1 (\underbrace{r_1}_{=r_0 - \alpha_0 A r_0} + \beta_0 \underbrace{p_0}_{=r_0}) \in x_0 + \text{span}\{r_0, Ar_0\}$$

$$= (x_0 + \alpha_0 r_0 + \gamma_1 A r_0)$$

$$\vdots$$

$$x_k \in x_0 + \underbrace{\text{span}\left\{r_0, Ar_0, A^2 r_0, \ldots, A^{k-1} r_0\right\}}_{=: K_k(A, r_0)}$$

$K_k(A, r_0) = k$-th *Krylov subspace* (induced by A and r_0)

12.1.1 The Dimension of $K_k(A, r_0)$

$K_k(A, r_0)$ is defined for any $A \in \mathbb{C}^{n \times n}$ and $r_0 \in \mathbb{C}^n$.

Recall:

Definition 12.2. ***Minimal Polynomial (1)***

The *minimal polynomial* Ψ of A is a monic polynomial:

$$\Psi(z) = \alpha_0 + \alpha_1 z + \alpha_2 z^2 + \cdots + \alpha_{d-1} z^{d-1} + z^d$$

of smallest degree d such that

$$\Psi(A) = \gamma_0 I + \gamma_1 A + \gamma_2 A^2 + \cdots + \gamma_{k-1} A^{d-1} + A^d = 0.$$

Ψ is unique.

$r_0, Ar_0, \ldots, A^{k-1} r_0$ are linearly independent $\Leftrightarrow$ $\gamma_0 r_0 + \gamma_1 A r_0 + \cdots + \gamma_{k-1} A^{k-1} r_0 + \gamma_k A^k r_0 = 0$, not all $\gamma_i = 0$

Definition 12.3. ***Minimal Polynomial (2), Grade***

1. The minimal polynomial Φ of A with respect to r_0 is the unique monic polynomial of smallest possible degree d for which $\Phi(A) r_0 = 0$. (The degree of Φ is $\leq$ the degree of Ψ.)
2. The *grade* of A with respect to r_0 is given by

$$d(A, r_0) = \text{ degree } \Phi.$$

- Note: $d(A, r_0) \leq n$.
- In general, the number of steps needed for conjugate gradient is equal to the grade.

13 2-8-12

13.1 Lemma about the Grade of a Matrix

$A \in \mathbb{C}^{n\times n},\ r_0 \in \mathbb{C}^n,\ K_k(A, r_0) = \operatorname{span}\left\{r_0, Ar_0, A^2r_0, \ldots, A^{k-1}r_0\right\} = \{\Psi(A)r_0 \mid \Psi \text{ is any polynomial of degree } \leq k-1\}$.

Lemma 13.1.

1. $$\dim K_k(A, r_0) = \begin{cases} k & k \leq d(A, r_0) \\ d(A, r_0) & k > d(A, r_0) \end{cases}$$

2. $$d(A, r_0) = \operatorname{rank} \underbrace{\begin{bmatrix} r_0 & Ar_0 & A^2r_0 & \cdots & A^{n-1}r_0 \end{bmatrix}}_{\in \mathbb{C}^{n\times n}}$$

3. $d(A, r_0)$ is the number of eigenvectors in an *eigendecomposition* of r_0:

$$r_0 = \sum_{j=1}^{d(A,r_0)} \rho_j z_j, \qquad \rho_j \in \mathbb{C},\ \rho_j \neq 0$$
$$Az_j = \lambda_j z_j, \qquad z_j \neq 0$$
$$\lambda_j \neq \lambda_l \qquad \text{for all } j, l = 1, 2, \ldots, d(A, r_0),\ j \neq l$$

13.2 Back to the Case $A \succ 0$

Theorem 13.2.

1. In exact arithmetic, CG terminates after finitely many steps:

$$x_k \neq x^* = A^{-1}b \qquad \text{for all } k = 0, 1, 2, \ldots, d(A, r_0) - 1$$
$$x_k = x^* \qquad \text{for } k = d(A, r_0)$$

2. For all $k = 1, 2, \ldots, d(A, r_0)$, $x_k \in x_0 + K_k(A, r_0)$ is optimal in the following sense: (Recall: $\|x\|_A = \sqrt{x^H A x}$)

$$\|x^* - x_k\|_A = \min_{x \in x_0 + K_k(A, r_0)} \|x^* - x\|_A$$

   And x_k is the unique minimizer.

3. For all $k = 1, 2, \ldots, d(A, r_0)$,

$$\frac{\|x^* - x_k\|_A}{\|x^* - x_0\|_A} \leq 2 \left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \right)^k$$

   where

$$\kappa = \frac{\lambda_{\max}(A)}{\lambda_{\min}(A)}$$

   and $\lambda_{\max}(A)$ is the largest eigenvalue of A and $0 < \lambda_{\min}(A)$ is the smallest.

   Note: $\kappa \geq 1$, κ is the condition number of A with respect to the norm $\|\cdot\|_2$.

In some applications, one would like to minimize $\|b - Ax\|_2$ instead of $\|x^* - x\|_A$.

$$\|x^* - x_0\|_A = \sqrt{(x^* - x_0)^H \underbrace{A(x^* - x_0)}_{=b}} = \sqrt{(x^* - x_0)^H r_0}$$

the *conjugate residual method* (CR) is a variant of conjugate gradient that generates iterates $x_k \in x_0 + K_k(A, r_0)$, $k = 1, 2, \ldots, d(A, r_0)$,

$$\|b - Ax_k\|_2 = \min_{x \in x_0 + K_k(A, r_0)} \|b - Ax\|_2$$

Remark 13.3. ***CR Algorithm***

In CG, replace $r_k^H r_k$ by $r_k^H A r_k$, and $p_k^H \underbrace{A p_k}_{=z} = p_k^H z$ by $p_k^H A^2 p_k = z^H z$, where $z = A p_k$.

13.3 Preconditioning

$$\frac{\|x^* - x_k\|_A}{\|x^* - x_0\|_A} \leq 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^k \qquad \text{for CG}$$

$$\frac{\|b - Ax_k\|_2}{\|b - Ax_0\|_2} \leq 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^k \qquad \text{for CR}$$

$\Rightarrow$ Fast convergence of CG (or CR) if κ is small (Recall: $\kappa \geq 1$).

Basic idea of preconditioning:
Find an equivalent system,

$$Ax = b \quad \Leftrightarrow \quad A'x' = b' \qquad (A' \succ 0),$$

such that $(1 \leq)\ \kappa(A') \ll \kappa(A)$.

14 2-10-12

14.1 Preconditioning (Continued)

$$Ax = b \quad \Leftrightarrow \quad A'x' = b'$$
$$\kappa(A) \gg \kappa(A')$$

$A, A' \in \mathbb{C}^{n\times n},\ A, A' \succ 0.$

Suppose $M \in \mathbb{C}^{n\times n},\ M \succ 0,\ M$ "approximates" A. Write $M = LL^H$, where $L \in \mathbb{C}^{n\times n}$ (e.g. Cholesky, although L doesn't have to be lower triangular).

$$\begin{aligned} Ax = b \quad &\Leftrightarrow \quad \underbrace{L^{-1}AL^{-H}}_{=:A'\succ 0}\underbrace{L^H x}_{=:x'} = \underbrace{L^{-1}b}_{=:b'} \\ &\Leftrightarrow \quad A'x' = b' \end{aligned}$$

(So $x = L^{-H}x'$.)

Note:

$$\begin{aligned} M \approx A \quad &\Leftrightarrow \quad A' = L^{-1}AL^{-H} \approx L^{-1}\underbrace{M}_{=LL^H}L^{-H} = I \\ &\Leftrightarrow \quad \kappa(A') \gtrapprox \kappa(I) = 1 \end{aligned}$$

Remark 14.1. ***Preconditioned Conjugate Gradient***

- Set $b' = L^{-1}b,\ x'_0 = L^H x_0$
- Apply CG to $A'x' = b'$ with initial guess x'_0; stops with x'_k
- Set $x_k = L^{-H}x'_k$

In order to be more efficient than non-preconditioned CG, we need:

- faster convergence (guaranteed if $\kappa(A') \ll \kappa(A)$)
- computing matrix-vector products $z' = A'p'$ is cheap
  - forming A' would not be efficient because, in general, A' is not sparse (even when A is sparse)

Computation of

$$z' = A'p' = L^{-1}\underbrace{A\overbrace{L^{-H}p'}^{=q}}_{=t}$$

$\Leftrightarrow$

- solve $L^H q = p'$ for q
- compute $t = Aq$
- solve $Lz' = t$ for z'

$\Rightarrow$ Solution of linear systems with coefficient matrices L^H and L needs to be cheap.

14.2 Some Preconditioners

14.2.1 Diagonal preconditioning

$$A = \begin{bmatrix} a_{11} & * & \cdots & * \\ * & a_{22} & \ddots & \vdots \\ \vdots & \ddots & \ddots & * \\ * & \cdots & * & a_{nn} \end{bmatrix} \succ 0 \quad (\Rightarrow\ a_{jj} > 0,\ j = 1, 2, \ldots, n)$$

$$M = \operatorname{diag}(A) \succ 0$$

$$L = L^H = \begin{bmatrix} \sqrt{a_{11}} & & & 0 \\ & \sqrt{a_{22}} & & \\ & & \ddots & \\ 0 & & & \sqrt{a_{nn}} \end{bmatrix}$$

14.2.2 Incomplete Cholesky factorization

Instead of computing L with $A = LL^H$, compute a sparser L such that $A \approx LL^H$ $(= M)$. Typical approach: prescribe the sparsity of L.

- Choose $G \subset \{(j,k) \mid 1 \le k < j \le n\}$ (i.e., a subset of the lower triangular matrix, not including the diagonal, which *must* be nonzero).
- Construct L such that $l_{jk} \neq 0 \ \Rightarrow\ (j,k) \in G$.
- Example: $G = \{(j,k) \mid 1 \le k < j \le n \text{ and } a_{jk} \neq 0\}$. Then L has the same sparsity structure as the lower triangular part of A.

Recall: After $k-1$ steps of Cholesky factorization,

$$L = \begin{bmatrix} l_{11} & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ l_{21} & l_{22} & \ddots & & & & \vdots \\ \vdots & & \ddots & \ddots & & & \vdots \\ \vdots & & & l_{k-1,k-1} & \ddots & & \vdots \\ \vdots & & & & l_{kk} & \ddots & \vdots \\ \vdots & & & & \vdots & \ddots & 0 \\ l_{n1} & \cdots & \cdots & l_{n,k-1} & l_{nk} & \cdots & l_{nn} \end{bmatrix}$$

Remark 14.2. ***Algorithm: Incomplete Cholesky Factorization***

(Starting at step k)

- If $l_{kk} \leq 0$: **stop**
- Set $l_{kk} = \sqrt{l_{kk}}$
- **for** $j = k+1, k+2, \ldots, n$, **do**
  - Set
$$l_{jk} = \begin{cases} \frac{l_{jk}}{l_{kk}} & (j,k) \in G \\ 0 & \text{otherwise} \end{cases}$$
  - **for** $i = k+1, k+2, \ldots, j$, **do**
    - Set
$$l_{ji} = \begin{cases} l_{ji} - l_{jk}\overline{l_{ik}} & (j,i) \in G \\ 0 & \text{otherwise} \end{cases}$$
  - **end** i

  end j

Result: $L = [l_{jk}]$ such that $l_{jk} = 0$ if $(j,h) \notin G$ and $LL^H \approx A$.

15 2-13-12

15.1 Incomplete Cholesky Factorization (Continued)

In general, incomplete Cholesky factorization can break down due to $l_{kk} \leq 0$.

However, no breakdowns can occur if A is an H-matrix.

15.2 M-Matrices and H-Matrices

Definition 15.1. ***M-Matrix***

A matrix $A \in \mathbb{R}^{n\times n}$ is said to be an *M-matrix* if there exists an $s \in \mathbb{R}$, $s > 0$, and a nonnegative matrix $B \in \mathbb{R}^{n\times n}$ such that

$$A = sI - B$$

and

$$\rho(B) < s.$$

Notes:

- B is nonnegative $\Leftrightarrow b_{jk} \geq 0$ for all j, k
- $\rho(B) = \max_{\lambda\in\sigma(B)} |\lambda| =$ *spectral radius*

Example 15.2.

$$T_m = \begin{bmatrix} 2 & -1 & 0 & \cdots & \cdots & 0 \\ -1 & 2 & -1 & 0 & \cdots & 0 \\ 0 & -1 & 2 & -1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & \ddots & \ddots & -1 \\ 0 & \cdots & \cdots & 0 & -1 & 2 \end{bmatrix} \in \mathbb{R}^{m\times m}, \qquad B = \begin{bmatrix} 0 & 1 & 0 & \cdots & \cdots & 0 \\ 1 & 0 & 1 & 0 & \cdots & 0 \\ 0 & 1 & 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & \ddots & \ddots & 1 \\ 0 & \cdots & \cdots & 0 & 1 & 0 \end{bmatrix}$$

is an M-matrix (for any $m \geq 1$):

$$T_m = 2I - B.$$

The eigenvalues of B are given by

$$\lambda_l = 2\cos\frac{\pi l}{m+1}, \qquad l = 1, 2, \ldots$$

Its spectral radius is

$$\rho(B) = 2\cos\frac{\pi}{m+1} < 1 < 2 = s$$

Note:

- $T_{m\times m} \in \mathbb{R}^{n\times n}$, where $n = m^2$, is also an M-matrix for any $m \geq 1$

Definition 15.3. ***H-Matrix***

A matrix $A = [a_{jk}] \in \mathbb{C}^{n\times n}$ is said to be an *H-matrix* if the matrix $\hat{A} = [\hat{a}_{jk}] \in \mathbb{R}^{n\times n}$ defined by

$$\hat{a}_{jk} := \begin{cases} |a_{jk}| & j = k \\ -|a_{jk}| & j \neq k \end{cases} \qquad j,k = 1,2,\ldots,n$$

is an M-matrix.

Theorem 15.4.

Let $A \in \mathbb{C}^{n\times n}$, $A \succ 0$, be an H-matrix. Then for any set

$$G \subset \{(j,k) \mid 1 \leq k < j\},$$

the corresponding incomplete Cholesky factorization $A \approx LL^H$ (L sonsingular) exists.

15.3 Some Preconditioners (Continued)

15.3.3 SSOR-Type Preconditioning

Let $A \succ 0$. Write $A = D_0 - E - \underbrace{F}_{E^H}$, where

- $D_0 = \operatorname{diag}(A)$
- E is the strictly lower triangular part of A.
- $F = E^H$ is the strictly upper triangular part of A.

$$M = (D - E)D^{-1}(D - E^H),$$

where D is any diagonal matrix such that $D \succ 0$.

Motivation: If $D = D_0$, then

$$\begin{aligned} M &= (D_0 - E)D_0^{-1}(D_0 - E^H) \\ &= \underbrace{D_0 - E - E^H}_{=A} + ED_0^{-1}E^H \\ &\approx A \end{aligned}$$

if $ED_0^{-1}E^H$ is "small."

$$M = \underbrace{(D - E)D^{-1/2}}_{=L}\underbrace{D^{-1/2}(D - E^H)}_{=L^H}$$

where L is lower-triangular and nonsingular.

$$A' = L^{-1}AL^{-H} = D^{1/2}\underbrace{(D - E)^{-1}A(D - E)^{-1}}_{=:\tilde{A}}D^{1/2}$$

Computation of

$$z' = A'p' = D^{1/2}\,\underbrace{\tilde{A}(\overbrace{D^{1/2}p'}^{=\tilde{p}})}_{=\tilde{z}} = D^{1/2}\tilde{z}$$

where $\tilde{p} = D^{1/2}p'$ and $\tilde{z} = \tilde{A}\tilde{p}$.

16 2-15-12

16.1 SSOR Preconditioning (Continued)

$$\begin{aligned} A &= D_0 - E - E^H \succ 0 \\ M &= (D-E)D^{-1}(D-E)^H, \qquad D \succ 0 \\ A' &= D^{1/2}\tilde{A}D^{1/2} \\ \tilde{A} &= (D-E)^{-1}A(D-E^H)^{-1} \end{aligned}$$

How do we compute $\tilde{z} = \tilde{A}\tilde{p}$? Claim: this can be computed with roughly the same amount of work as $z = Ap$. (Thus, SSOR preconditioning is almost free!)

$$\begin{aligned} \tilde{A} &= (D-E)^{-1}(D_0 - E - E^H)(D-E^H)^{-1} \\ &= (D-E)^{-1}(\underbrace{D_0 - 2D}_{=:D_1} + D - E + D - E^H)(D-E^H)^{-1} \\ &= (D-E)^{-1}(I + D_1(D-E^H)^{-1} + (D-E^H)^{-1} \\ \tilde{z} = \tilde{A}\tilde{p} &= \underbrace{(D-E)^{-1}(\tilde{p} + D_1\overbrace{(D-E^H)^{-1}\tilde{p}}^{=v})}_{=w} + \overbrace{(D-E^H)^{-1}\tilde{p}}^{=v} \end{aligned}$$

Remark 16.1. ***Algorithm: Eisenstat Trick***

Input: $\tilde{p} \in \mathbb{C}^n$, a routine to multiply by D_1 (diagonal matrix), a routine to solve linear systems with matrix $D - E$, a routine to solve linear systems with matrix $D - E^H$

- Solve $(D - E^H)v = \tilde{p}$ for v
- Solve $(D - E)w = \tilde{p} + D_1 v$ for w
- Set $\tilde{z} = w + v$

Output: $\tilde{z} = \tilde{A}\tilde{p}$

Note: The two triangular solves with $D - E^H$ and $D - E$ require about the same amount of work as one multiplication by A.

16.2 Krylov Subspace Methods for General Linear Systems

$$Ax = b, \qquad \text{where } A \in \mathbb{C}^{n\times in} \text{ is nonsingular, } b \in \mathbb{C}^n \tag{16.1}$$

16.2.1 CGNE

Poor man's use of CG to solve (16.1):

$$(16.1) \;\Leftrightarrow\; A^H Ax = A^H b \tag{16.2}$$

normal equations

Note: $A^H A \succ 0$, since $x^H A^H A x = \|Ax\|_2^2 > 0$ for $x \neq 0$

Resulting method: *CGNE* (CG applied to the Normal Equations)

$$\|x^* - x\|_{A^H A} = \|Ax^* - Ax\|_2 = \|b - Ax\|_2$$
$$r_k = b - Ax_k$$

CGNE Iterates:

$$x_k \in x_0 + K_k(A^H A, A^H r_0)$$

such that

$$\|b - Ax\|_2 = \min_{x \in x_0 + K_k(A^H A, A^H r_0)} \|b - Ax\|$$

CGNE error bounds:

$$\frac{\|r_k\|_2}{\|r_0\|_2} = \frac{\|b - Ax_k\|_2}{\|b - Ax_0\|_2} = \frac{\|x^* - x_k\|_{A^H A}}{\|x^* - x_0\|_{A^H A}} \leq 2\left(\frac{\sqrt{\kappa(A^H A)} - 1}{\sqrt{\kappa(A^H A)} + 1}\right)^k$$
$$= 2\left(\frac{\kappa(A) - 1}{\kappa(A) + 1}\right)^k$$

(where we have used that $\sqrt{\kappa(A^H A)} = \kappa(A) = \frac{\sigma_{\max}(A)}{\sigma_{\min}(A)}$, $\sigma_{\max}(A)$ is the maximum singular value of A and is equal to $\|A\|_2$, and $\sigma_{\min}(A)$ is the minimum singular value of A and is equal to $\frac{1}{\|A^{-1}\|_2}$). $\Rightarrow$ very slow convergence in general.

16.2.2 Craig's Method

A second poor man's use of CG:

$$(16.1) \Leftrightarrow \underbrace{AA^H}_{\succ 0} y = b, \quad x = A^H y \tag{16.3}$$

Apply CG to (16.3). This is called *Craig's method*. Its iterates satisfy:

$$\|x^* - x_k\|_2 = \min_{x \in x_0 + K_k(A^H A, A^H r_0)} \|x^* - x\|_2$$

Error bound:

$$\frac{\|x^* - x_k\|_2}{\|x^* - x_0\|_2} \leq 2\left(\frac{\kappa(A) - 1}{\kappa(A) + 1}\right)^k$$

$\Rightarrow$ same slow convergence!

17 2-17-12

17.1 Krylov Subspace Methods for General Linear Systems (Continued)

$$Ax = b, \quad A \in \mathbb{C}^{n\times n} \text{ nonsingular}, \ b \in \mathbb{C}^n \tag{17.1}$$
$$K_k(A, r_0) = \text{span}\left\{r_0, Ar_0, A^2 r_0, \ldots, A^{k-1} r_0\right\}$$

$$\mathbb{C}^n \ni x_0 \to x_1 \to \cdots \to x_k \to \cdots \qquad (r_0 = b - Ax_0)$$

where $x_k \in x_0 + K_k(A, r_0)$ is such that

$$\|b - Ax_k\|_2 = \min_{x \in x_0 + K_k(A,r_0)} \|b - Ax\|$$

Recall: CG and CR are based on short recurrences:

$$\left.\begin{aligned} x_{k+1} &= x_k + \alpha_k P_k \\ r_{k+1} &= r_k - \alpha_k A p_k \\ P_{k+1} &= r_{k+1} + \beta_k P_k \end{aligned}\right\} \text{ coupled 2-term recurrences}$$

Theorem 17.1.

For general A, it is not possible to implement a minimal residual method with short recurrences: to generate x_{k+1}, vectors from all previous k iterations are needed.

Actual implementation?

17.2 The Arnoldi Process

Goal: Given $A \in \mathbb{C}^{n\times n}$ (not necessarily nonsingular) and $r_0 \in \mathbb{C}^n$, construct orthonormal basis vectors $v_1, v_2, \ldots, v_k, \ldots$ such that the first k basis vectors span $K_k(A, r_0)$, $k = 1, 2, \ldots$.

Remark 17.2. ***Algorithm: Arnoldi Process***

Input: $r_0 \in \mathbb{C}^n$ and a routine to compute $q = Av$ for any $v \in \mathbb{C}^n$.

Set $\beta_1 = \|r_0\|_2$.

$$r_0, Ar_0, A^2r_0, \ldots, A^{k-1}r_0, A^k r_0$$

$$v_1, \quad v_2, \quad v_3, \ldots, \quad v_k, \quad \tilde{v}_{k+1} = Av_k - \sum_{j=1}^{k} h_{jk} v_i$$

If $\beta_1 = 0$, **stop**: $r_0 = 0$
Otherwise, set $v_1 = \frac{r_0}{\beta_1}$
for $k = 1, 2, \ldots,$ **do**

- Compute $q = Av_k$
- **for** $j = 1, 2, \ldots, k$
  - Set $h_{jk} = v_j^H q$
  - Set $q = q - h_{jk} v_j$
  - (This is called *Modified Gram-Schmidt*)
- **end**(j)
- Set $h_{k+1,k} = \|q\|_2$
- If $h_{k+1,k} = 0$, **stop**: the Krylov subspace $K_k(A, r_0)$ has reached its maximal dimension, i.e. $k = d(A, r_0)$
- Set $v_{k+1} = \frac{q}{h_{k+1,k}}$

end(k)

Output: $v_1, v_2, \ldots, v_k, \ldots, v_{d(A,r_0)}$

Properties

1. $$v_k^H v_j = \begin{cases} 0 & k \neq j \\ 1 & k = j \end{cases}$$
   - This is equivalent to $V_k^H V_k = I_k$
2. The v_k's form a basis of the $(K_k(A, r_0))$'s,
   $$K_k(A, r_0) = \text{span}\{v_1, v_2, \ldots, v_k\}, \quad k = 1, 2, \ldots, d(A, r_0)$$
3. For all $k = 1, 2, \ldots, d(A, r_0)$,
   $$h_{k+1,k} v_{k+1} = Av_k - \sum_{j=1}^{k} h_{jk} v_j$$

- Note: this property can be written compactly as

$$
\begin{aligned}
& AV_k = V_{k+1}\tilde{H}_k, \\
\text{where} \quad & V_k = \begin{bmatrix} v_1 & v_2 & \cdots & v_k \end{bmatrix}, \quad V_{k+1} = \begin{bmatrix} V_k & v_{k+1} \end{bmatrix}, \\
& \tilde{H}_k = [h_{jl}]_{\substack{j=1,2,\ldots,k+1 \\ l=1,2,\ldots,k}} = \begin{bmatrix} h_{11} & h_{12} & \cdots & \cdots & h_{1k} \\ h_{21} & h_{22} & h_{23} & & \vdots \\ & h_{32} & \ddots & \ddots & \vdots \\ & & \ddots & \ddots & h_{k-1,k} \\ & & & \ddots & h_{k,k} \\ 0 & & & & h_{k+1,k} \end{bmatrix} \in \mathbb{C}^{k+1\times k}
\end{aligned}
$$

  $\tilde{H}_k$ is an upper Hessenberg matrix. $\operatorname{rank}\tilde{H}_k = k$.

4.

$$
V_k^H A V_k = H_k = \begin{bmatrix} h_{11} & h_{12} & \cdots & \cdots & h_{1k} \\ h_{21} & h_{22} & h_{23} & & \vdots \\ & h_{32} & \ddots & \ddots & \vdots \\ & & \ddots & \ddots & h_{k-1,k} \\ 0 & & & h_{k,k-1} & h_{k,k} \end{bmatrix} \in \mathbb{C}^{k\times k}
$$

18 2-22-12

18.1 GMRES

Recall:

Definition 18.1. ***Minimal Residual Method***

A *minimal residual method* is a method where

$$x_k \in x_0 + K_k(A, r_0)$$

such that

$$\|b - Ax_k\|_2 = \min_{x \in x_0 + K_k(A,r_0)} \|b - Ax\|_2$$

Here, $x_0 \in \mathbb{C}^n$, assume that $r_0 := b - Ax_0 \neq \mathbf{0}$ (i.e., $x_0 \neq A^{-1}b$).

GMRES is an implementation of the minimal residual method using the Arnoldi process.

Recall:

$$AV_k = V_{k+1}\tilde{H}_k, \qquad V_k^H V_k = I_k \text{ (orthonormal)}$$
$$K_k(A, r_0) = \{v = V_k z \mid z \in \mathbb{C}^k\}$$

At step k of GMRES:

$$x \in x_0 + K_k(A, r_0) \quad \Leftrightarrow \quad x = x_0 + V_k z,\ z \in \mathbb{C}^k$$

The corresponding residual vector is:

$$\begin{aligned}
b - Ax &= \underbrace{b - Ax_0}_{=r_0} - \underbrace{AV_k}_{=V_{k+1}\tilde{H}_k} z \\
v_1 &= \frac{r_0}{\beta_1} \quad \Rightarrow \quad r_0 = v_1 \beta 1 \\
b - Ax &= \beta_1 v_1 - V_{k+1}\tilde{H}_k z \\
&= V_{k+1}(\beta_1 e_1 - \tilde{H}_k z), \qquad \text{where } e_1 = \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix}^T \in \mathbb{R}^{k+1} \\
\Rightarrow \qquad \|b - Ax\|_2 &= \|V_{k+1}(\beta_1 e_1 - \tilde{H}_k z)\|_2 \qquad (\text{Recall: } V^H V = I \ \Rightarrow\ \|Vy\|_2 = \|y\|_2) \\
&= \|\beta_1 e_1 - \tilde{H}_k z\|_2 \\
\Rightarrow \qquad \|b - Ax_k\|_2 &= \min_{x \in x_0 + K_k(A,r_0)} \|b - Ax\|_2 = \min_{z \in \mathbb{C}^k} \|\beta_1 e_1 - \tilde{H}_k z\|_2
\end{aligned}$$

To compute the kth GMRES iterate:

1. Find $z_k \in \mathbb{C}^k$ such that
$$\|\beta_1 e_1 - \tilde{H}_k z_k\|_2 = \min_{z \in \mathbb{C}^k} \|\beta_1 e_1 - \tilde{H}_k z\|_2$$
This is a least squares problem (LS_k).
2. Set $x_k = x_0 + V_k z_k$

This is a least squares problem with a $(k+1) \times k$ matrix $\tilde{H}_k$. For $b \in \mathbb{C}^m$, $A \in \mathbb{C}^{m \times n}$, the minimizer of $\min_{x \in \mathbb{C}^n} \|b - Ax\|_2$ is unique if $\mathrm{rank}(A) = n$. For our problem, we will have a unique solution because $\tilde{H}_k$ is of full column rank k, and therefore x_k is also unique.

18.2 Solution of (LS_k)

Let $Q_k \in \mathbb{C}^{k+1\times k+1}$ be a unitary matrix (i.e., $Q_k^H Q_k = I_{k+1}$) such that

$$Q_k \tilde{H}_k = \begin{bmatrix} & R_k & \\ 0 & \cdots & 0 \end{bmatrix} \in \mathbb{C}^{k+1\times k},$$

where $R_k \in \mathbb{C}^{k\times k}$ is an upper-triangular matrix. The important thing is: rank $\tilde{H}_k = k \Rightarrow R_k$ is nonsingular. So

$$\begin{aligned} \|\beta_1 e_1 - \tilde{H}_k z\|_2 &= \|Q_k \beta_1 e_1 - \underbrace{R_k}_{=Q_k \tilde{H}_k} z\|_2 \\ Q_k \beta_1 e_2 &= \begin{bmatrix} f_k \ (\in \mathbb{C}^k) \\ \tau_{k+1} (\in \mathbb{C}) \end{bmatrix} \\ \min_{z\in\mathbb{C}^k} \|\beta_1 e_1 - \tilde{H}_k z\|_2 &= \min_{z\in\mathbb{C}^k} \|Q_k \beta_1 e_1 - R_k z\|_2 \\ &= \min_{z\in\mathbb{C}^k} \left\| \begin{bmatrix} f_k - R_k z \\ \tau_{k+1} \end{bmatrix} \right\|_2 \\ &= |\tau_{k+1}| \\ \Rightarrow \qquad z_k &= R_k^{-1} f_k \end{aligned}$$

Note: $\|b - Ax_k\|_2 = |\tau_{k+1}|$

Remark 18.2. ***Algorithm: GMRES***

Input: $x_0 \in \mathbb{C}^n$, a routine to compute matrix-vector products $q = Av$, a convergence tolerance `tol`.

Set $r_0 = b - Ax_0$ and $\beta_1 = \|r_0\|2$.
If $\beta_1 = 0$, **stop:** $x_0 = A^{-1}b$ is the solution of $Ax = b$.
Set $v_1 = \frac{r_0}{\beta_1}$.

For $k = 1, 2, \ldots,$, **do:**

1. Perform the kth step of the Arnoldi process. (So we have $\tilde{H}_k$, v_1, v_2, $\ldots$, v_{k+1}.)
2. Determine z_k and τ_{k+1} such that

$$|\tau_{k+1}| = \|\beta_1 e_1 - \tilde{H}_k z_k\|_2 = \min_{z\in\mathbb{C}^k} \|\beta_1 e_1 - \tilde{H}_k z\|_2$$

3. If $\frac{|\tau_{k+1}|}{\beta_1}$ $\left(= \frac{\|b-Ax_k\|_2}{\|b-Ax_0\|_2}\right) \leq$ `tol`, **stop**

end(k)

$x_k = x_0 + V_k z_k \approx A^{-1}b$

There cannot be a way of implementing GMRES without storing lots of "stuff" (i.e., the v_i's $\Leftrightarrow V_k$).

19 2-24-12

19.1 Computation of z_k and τ_{k+1}

$$x_k = x_0 + V_k z_k$$
$$\|\tau_{k+1}\| = \|b - Ax_k\|_2$$

Recall:

Definition 19.1. ***Givens Rotation***

An elementary *Givens rotation* is

$$G = \begin{bmatrix} c & s \\ -\overline{s} & \overline{c} \end{bmatrix} \in \mathbb{C}^{2\times 2},$$
$$\text{where} \qquad |c|^2 + |s|^2 = 1.$$

General form:

$$G = \begin{bmatrix} 1 & & & & & & & & 0 \\ & \ddots & & & & & & & \\ & & 1 & & & & & & \\ & & & c & & & s & & \\ & & & & 1 & & & & \\ & & & & & \ddots & & & \\ & & & & & & 1 & & \\ & & & -\overline{s} & & & \overline{c} & & \\ & & & & & & & 1 & \\ & & & & & & & & \ddots \\ 0 & & & & & & & & 1 \end{bmatrix}$$

Notes:

1. For any $h = \begin{bmatrix} h_1 \\ h_2 \end{bmatrix} \in \mathbb{C}^2$, G can be chosen such that $Gh = \begin{bmatrix} * \\ 0 \end{bmatrix}$
2. G is unitary: $G^H G = I$.

$\tilde{H}_k \to R_k$ by a product k Givens rotations.

Example 19.2. $k = 4$

$$\tilde{H}_4 = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ 0 & * & * & * \\ 0 & 0 & * & * \\ 0 & 0 & 0 & * \end{bmatrix}$$

$$G_1 = \begin{bmatrix} c & s & & \\ -\overline{s} & \overline{c} & & \\ & & 1 & \\ & & & 1 \end{bmatrix}$$

$$\tilde{H}_4 \xrightarrow{G_1} G_1\tilde{H}_4 = \begin{bmatrix} * & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \\ 0 & 0 & * & * \\ 0 & 0 & 0 & * \end{bmatrix}$$

$$\xrightarrow{G_2} G_2G_1\tilde{H}_k = \begin{bmatrix} * & * & * & * \\ 0 & * & * & * \\ 0 & 0 & * & * \\ 0 & 0 & * & * \\ 0 & 0 & 0 & * \end{bmatrix}$$

$$\xrightarrow{G_3} G_3G_2G_1\tilde{H}_k = \begin{bmatrix} * & * & * & * \\ 0 & * & * & * \\ 0 & 0 & * & * \\ 0 & 0 & 0 & * \\ 0 & 0 & 0 & * \end{bmatrix}$$

$$\xrightarrow{G_4} \underbrace{G_4G_3G_2G_1}_{=Q_4}\tilde{H}_k = \begin{bmatrix} * & * & * & * \\ 0 & * & * & * \\ 0 & 0 & * & * \\ 0 & 0 & 0 & * \\ 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} & R_4 & \\ 0 & \cdots & 0 \end{bmatrix}$$

$$\min_{z\in\mathbb{C}^k} \|\beta_1 e_1 - \tilde{H}_k z\|_2 = \min \left\| Q_k\beta_1 e_1 - \begin{bmatrix} & R_k & \\ 0 & \cdots & 0 \end{bmatrix} z \right\|_2 = |\tau_{k+1}|$$

$$Q_k\beta_1 e_1 = \begin{bmatrix} f_k \\ \tau_{k+1} \end{bmatrix}$$

$$z_k = R_k^{-1} f_k$$

The factorization

$$Q_k\tilde{H}_k = \begin{bmatrix} & R_k & \\ 0 & \cdots & 0 \end{bmatrix}$$

can easily be updated from $k-1 \to k$:

$$\tilde{H}_k = \left[\begin{array}{ccc|c} & \tilde{H}_{k-1} & & \begin{matrix} h_{1k} \\ \vdots \\ h_{kk} \end{matrix} \\ \hline 0 & \cdots & 0 & h_{k+1,k} \end{array}\right].$$

We have $Q_{k-1}\tilde{H}_{k-1} = \begin{bmatrix} & R_k & \\ 0 & \cdots & 0 \end{bmatrix}$, where $Q_{k-1} = G_{k-1}G_{k-2}\cdots G_1$.

$$\begin{bmatrix} & & & 0 \\ & Q_{k-1} & \vdots & \\ & & & 0 \\ 0 & \cdots & 0 & 1 \end{bmatrix} \tilde{H}_k = \left[\begin{array}{ccc|c} & & & r_{1k} \\ & & & r_{2k} \\ & R_{k-1} & & \vdots \\ & & & r_{k-1,k} \\ 0 & \cdots & 0 & \tilde{r}_{kk} \\ \hline 0 & \cdots & 0 & \tilde{r}_{k+1,k} \end{array}\right]$$

Select $\hat{G}_k = \begin{bmatrix} c_k & s_k \\ -\overline{s_k} & \overline{c_k} \end{bmatrix}$ such that $\hat{G}_k \begin{bmatrix} \tilde{r}_{kk} \\ \tilde{r}_{k+1,k} \end{bmatrix} = \begin{bmatrix} r_{kk} \\ 0 \end{bmatrix}$ and set

$$\begin{aligned} \underbrace{Q_k}_{k+1\times k+1} &= \begin{bmatrix} I_{k-1} & \mathbf{0} \\ \mathbf{0} & \hat{G}_k \end{bmatrix} \begin{bmatrix} Q_{k-1} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \\ \Rightarrow \qquad Q_k\tilde{H}_k &= \begin{bmatrix} & R_k & \\ 0 & \cdots & 0 \end{bmatrix} \\ R_k &= \left[\begin{array}{ccc|c} & & & r_{1k} \\ & R_{k-1} & & r_{2k} \\ & & & \vdots \\ & & & r_{k-1,k} \\ \hline 0 & \cdots & 0 & r_{kk} \end{array}\right] \\ \begin{bmatrix} f_k \\ \tau_{k+1} \end{bmatrix} &= Q_k(\beta_1 e_1) = \begin{bmatrix} I_{k-1} & \mathbf{0} \\ \mathbf{0} & \hat{G}_k \end{bmatrix} \begin{bmatrix} Q_{k-1} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \beta_1 e_1 \\ \begin{bmatrix} Q_{k-1} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \beta_1 e_1 &= \begin{bmatrix} Q_{k-1}\beta_1 e_1 \\ 0 \end{bmatrix} = \begin{bmatrix} f_{k-1} \\ \tau_k \\ 0 \end{bmatrix} \\ \begin{bmatrix} f_k \\ \tau_{k+1} \end{bmatrix} &= \begin{bmatrix} f_{k-1} \\ c_k\tau_k \\ -\overline{s_k}\tau_k \end{bmatrix} \\ f_k &= \begin{bmatrix} f_{k-1} \\ c_k\tau_k \end{bmatrix} \\ \tau_{k+1} &= -\overline{s_k}\tau_k \end{aligned}$$

Note:

$$\begin{aligned} |\tau_{k+1}| &= \|b - Ax_k\|_2 = \|r_k\|_2 \\ |\tau_k &= \|b - Ax_k\|_2 \leq |\tau_{k+1}| = \|r_{k-1}\|_2 \\ \|r_k\| &= \underbrace{|s_k|}_{\leq 1} \|r_{k-1}\|_2 \end{aligned}$$

20 2-27-12

20.1 Convergence of the Minimal Residual Method

$Ax = b$, A nonsingular, x_0, $r_0 = b - Ax_0$

$$\begin{aligned} K_k(A, r_0) &= \text{span}\left\{r_0, Ar_0, A^2r_0, \ldots, A^{k-1}r_0\right\} \\ &= \{v = \gamma_0 r_0 + \gamma_1 Ar_0 \cdots + \gamma_{k-1}A^{k-1}r_0 \mid \gamma_0, \gamma_1, \ldots, \gamma_{k-1} \in \mathbb{C}\} \\ &= \{v = p(A)r_0 \mid p \in \prod_{k-1}\} \\ \prod_{k-1} &:= \{p(\lambda) = \gamma_0 + \gamma_1\lambda + \cdots + \gamma_{k-1}\lambda^{k-1} \mid \gamma_0, \gamma_1, \ldots, \gamma_{k-1} \in \mathbb{C}\} \end{aligned}$$

$x \in K_k(A, r_0) \Leftrightarrow x = x_0 + p(A)r_0,\ p \in \prod_{k-1}$

$$\begin{aligned} b - Ax &= \underbrace{b - Ax_0}_{=r_0} - Ap(A)r_0 = \underbrace{I - Ap(A)}_{=q(A)} r_0 \\ &= q(A)r_0 \end{aligned}$$

where $q(\lambda) = 1 - \lambda p(\lambda) \in \prod_k$, $q(0) = 1$.

Minimal residual property:

$$\begin{aligned} \|b - Ax_k\|_2 &= \min_{x \in x_0 + K_k(A, r_0)} \|b - Ax\|_2 \\ &= \min_{q \in \prod_k,\ q(0)=1} \|q(A)r_0\|_2 \\ &\le \|r_0\|_2 \min_{q \in \prod_k,\ q(0)=1} \|q(A)\|_2 \end{aligned}$$

Theorem 20.1.

The iterates x_k that are generated by the minimal residual method (such as GMRES) satisfy

1. $$\frac{\|b - Ax_k\|_2}{\|b - Ax_0\|_2} = \frac{\|r_k\|_2}{\|r_0\|_2} \le \min_{q \in \prod_k,\ q(0)=1} \|q(A)\|_2$$
2. Moreover, if A is diagonalizable, i.e., $A = U\Lambda U^{-1}$, $U \in \mathbb{C}^{n \times n}$ nonsingular and $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_n)$, then

$$\frac{\|r_k\|_2}{\|r_k\|_0} \le \kappa_2(U) \min_{q \in \prod_k,\ q(0)=1} \max_{j=1,2,\ldots,n} |q(\lambda_j)|$$

Notes:

1. $\kappa_2(U) = \|U\|_2\|U\|_2^{-1}$ = condition number of U with respect to the $\|\cdot\|_2$ norm.
2. $U = \begin{bmatrix} u_1 & u_2 & \cdots & u_n \end{bmatrix}$,

$$A = U\Lambda U^{-1} \Leftrightarrow AU = U\Lambda \Leftrightarrow Au_i = \lambda_i u_i,\ i = 1, 2, \ldots, n$$

Proof. (Of #2; we already proved #1.)

$$\begin{aligned}
A &= U\Lambda U^{-1} \\
A^2 &= U\Lambda U^{-1}U\Lambda U^{-1} = U\Lambda^2 U^{-1} \\
&\vdots \\
A^j &= U\Lambda^j U^{-1} \\
q(A) &= Uq(\Lambda)U^{-1}, \\
q(\Lambda) &= \operatorname{diag}(q(\lambda_1), q(\lambda_2), \ldots, q(\lambda_n)) \\
\|q(A)\|_2 &= \|Uq(\Lambda)U^{-1}\|_2 \\
&\leq \|U\|_2 \|q(\Lambda)\|_2 \|U^{-1}\|_2 \\
&= \kappa_2(U)\|q(\Lambda)\|_2 \\
&= \kappa_2(U) \max_{j=1,2,\ldots,n} |q(\lambda_j)|
\end{aligned}$$

Together with #1:

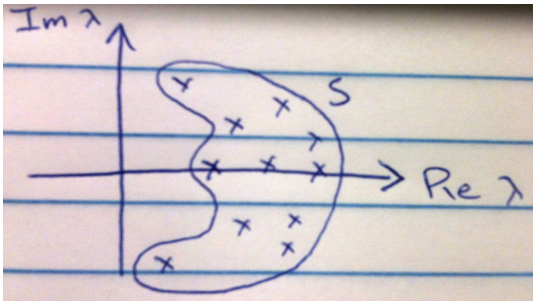
$$\frac{\|r_k\|}{\|r_0\|_2} \leq \kappa_2(U) \min_{q\in\prod_k,\ q(0)=1} \max_{j=1,2,\ldots,n} |q(\lambda_j)|$$

□

Corollary 20.2.

If A is diagonalizable, $\sigma(A) \subset S$, $0 \notin S$, S is compact. Then

$$\frac{\|r_k\|_2}{\|r_0\|_2} \leq \kappa(U) \min_{q\in\prod_k,\ q(0)=1} \max_{\lambda\in S} |q(\lambda)|$$



Consequence: We get fast convergence of GMRES if the eigenvalues of A are clustered away from 0.

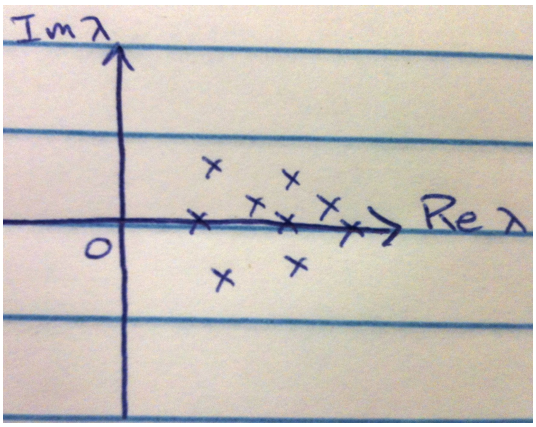


Figure 3: "Good" case.

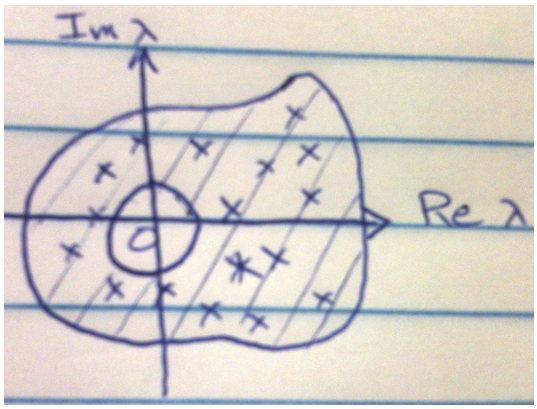


Figure 4: "Bad" case.

$Ax = b$ "bad" $\rightarrow A'x' = b'$ "good"

21 2-29-12

21.1 Convergence Results (Continued)

Example 21.1. ***Eigenvalues in a disk S***

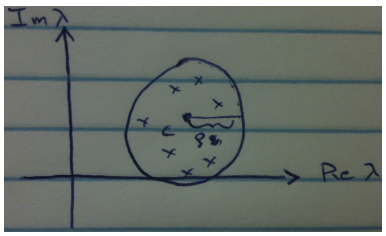


Figure 5: $S = \{\lambda \mid |\lambda - c| \leq \rho\}$, where $\rho < |c| \Leftrightarrow 0 \notin S$, $\sigma(A) \subset S$.

$$\frac{\|r_k\|_2}{\|r_0\|_2} \leq \kappa_2(U) \min_{q \in \prod_k,\ q(0)=1} \max_{\lambda \in X} |q(\lambda)|$$

$$q(\lambda) = \left(1 - \frac{\lambda}{c}\right)^k \in \prod_k, \quad q(0) = 1$$

$$\max_{\lambda \in S} |q(\lambda)| = \max_{\lambda \in \partial S} |q(\lambda)|$$

$$\partial S = \{\lambda = c + \rho e^{i\phi} \mid 0 \leq \phi < 2\pi\}$$

$$\max_{\lambda \in S} |q(\lambda)| = \max_{0 \leq \phi < 2\pi} \left| -\frac{\rho}{c} e^{i\phi} \right|^k$$

$$= \left(\frac{\rho}{|c|}\right)^k$$

$$\frac{\|r_k\|_2}{\|r_0\|_2} \leq \kappa_2(U) \min_{q \in \prod_k,\ q(0)=1} \max_{\lambda \in X} |q(\lambda)| \underbrace{\leq}_{\text{actually, } =} \kappa_2(U) \left(\frac{\rho}{|c|}\right)^k$$

21.2 A Convergence Bound Based on Bendixson's Theorem

Recall:

- $A = A^H \Rightarrow \sigma(A)$ is real
- $A = -A^H \Rightarrow \sigma(A)$ is purely imaginary

For any $A \in \mathbb{C}^{n \times n}$, we can write

$$A = \underbrace{\frac{A + A^H}{2}}_{=:A_H} + \underbrace{\frac{A - A^H}{2}}_{=:A_s}$$

$A_H = A_H^H =$ the Hermitian part of A

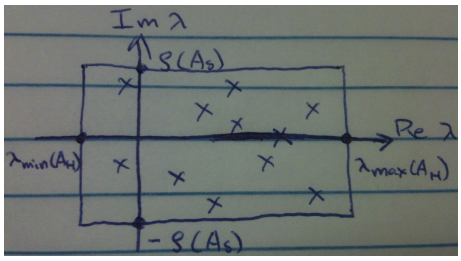
$A_S = -A_S^H =$ the skew-Hermitian part of A

This is the unique decomposition of A into Hermitian and skew-Hermitian matrices.

Theorem 21.2. ***Bendixson's Theorem***

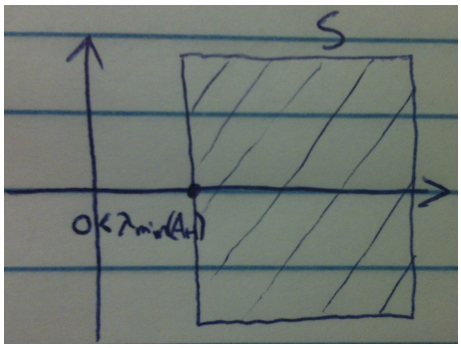
For all $\lambda \in \sigma(A)$,

$$\lambda_{\min}(A_H) \le \operatorname{Re}\lambda \le \lambda_{\max}(A_H)$$
$$-\rho(A_S) \le \operatorname{Im}\lambda \le \rho(A_S) := \max_{\lambda_j \in \sigma(A_S)} |\lambda_j|$$



Theorem 21.3.

If $A_H \succ 0$,



then

$$\frac{\|r_k\|_2}{\|r_0\|_2} \le \left(1 - \frac{1}{\kappa_2(A_H) + \left(\frac{\rho(A_S)}{\lambda_{\min}(A_H)}\right)}\right)^{k/2} \qquad \text{for } k = 1, 2, \ldots$$

where $\kappa_2(A_H) = \dfrac{\lambda_{\max}(A_H)}{\lambda_{\min}(A_H)}$

21.3 Preconditioned GMRES

$$Ax = b, \qquad A \in \mathbb{C}^{n\times n} \text{ nonsingular}, \ b \in \mathbb{C}^n \tag{21.1}$$

Preconditioner: $M \in \mathbb{C}^{n\times n}$ nonsingular such that $M = M_1 M_2$, where $M_1, M_2 \in \mathbb{C}^{n\times n}$ and systems with M_1 and M_2 are "easy" to solve, and $M \approx A$ (in some sense).

Let $x_0 \in \mathbb{C}^n$ be any initial guess for (21.1). Then

$$\begin{aligned} Ax = b &\Leftrightarrow A(x - x_0) = b - Ax_0 \\ &\Leftrightarrow \underbrace{M_1^{-1} A M_2^{-1}}_{=:A'} \underbrace{M_2(x - x_0)}_{=:x'} = \underbrace{M_1^{-1}(b - Ax_0)}_{=:b'} \\ x &= x_0 + M_2^{-1} x' \end{aligned}$$

$$\underbrace{M}_{=M_1 M_2} \approx A \Leftrightarrow M_1^{-1} A M_2^{-1} \approx I \Leftrightarrow A' \approx I$$

Thus, GMRES applied to $A'x' = b'$ converges faster than for $Ax = b$. (e.g., $\sigma(A')$ is clustered away from 0.)

22 3-2-12

22.1 Preconditioning with GMRES (Continued)

$A \in \mathbb{C}^{n \times n}$ nonsingular

$$\begin{aligned} Ax = b \quad &\Leftrightarrow \quad A'x' = b' \\ M &= M_1 M_2 \approx A \\ A' &= M_1^{-1} A M_2^{-1} \\ x' &= M_2(x - x_0) \\ b' &= M_1^{-1}(b - Ax_0) \end{aligned}$$

Remark 22.1. ***Algorithm (Preconditioned GMRES)***

Input: $x_0 \in \mathbb{C}^n$, a routine to compute $q = Av$, a routine to solve systems with M_1, and a routine to solve systems with M_2, a convergence tolerance `tol`

1. Solve $M_1 b' = b - Ax_0$ for b'
2. Set $x_0' := \mathbf{0} \in \mathbb{C}^n$ and $r_0' = b'$
3. Run GMRES to solve $A'x' = b'$ (with initial guess x_0') to residual accuracy:

$$\frac{\|r_k'\|_2}{\|r_0'\|_2} \leq \texttt{tol}$$

4. Solve $M_2 w = x_k'$ for w and set $x_k = x_0 + w$

Output: Approximate solution x_k for $Ax = b$.

Note: Step #3 requires matrix vector products

$$q' = a'v' \ (= M_1^{-1} A M_2^{-1} v')$$

and each such product requires 1 solve with M_2, 1 multiplication with A, and 1 solve with M_1.

22.2 Some Preconditioners

1. Diagonal Preconditioning. $A = D_0 - E - F$, where $D_0 = \text{diag}(A)$ is nonsingular, E is strictly lower-triangular, and F is strictly upper-triangular. Set $M = D_0$. Typically, we set $M_1 = D_0$ and $M_2 = I$ (left preconditioning), or $M_1 = I$ and $M_2 = D_0$ (right preconditioning). In general, this will work quite well if A is diagonally dominant.

2. Incomplete LU Factorization. $PAQ \approx LU$, where P and Q are permutation matrices (e.g. Markowitz criterion), L is lower-triangular, and U is upper-triangular. Then

$$A \approx \underbrace{P^T L}_{=M_1} \underbrace{U Q^T}_{=M_2}$$

   Systems with M_1 and M_2 are easy to solve, provided that L and U are sparse enough.

3. SSOR-Type Preconditioner. Based on $A = D_0 - E - F$ and proceed as before, but replace E^H with F.

22.3 Restarted GMRES

WLOG, assume $A = A'$ is already the preconditioned matrix, $A \in \mathbb{C}^{n \times n}$.

Recall: The kth step of the Arnoldi process,

$$h_{k+1,k} v_{k+1} = A v_k - h_{1k} v_1 - h_{2k} v_2 - \cdots - h_{kk} v_k,$$

requires:

- $k+1$ inner products (in $\mathbb{C}^n$)
- k SAXPY's (in $\mathbb{C}^n$)
- 1 multiplication with A
- storage of all vectors $v_1, v_2, \ldots, v_{k+1}$

For very large n, the Arnoldi process becomes too expensive quickly (as k increases).

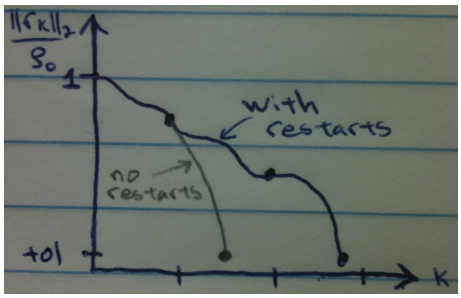
Remedy: Restarts.

Let k be the largest number of Arnoldi steps that one is willing to run. Typical values: $k_0 = 50$ or $k_0 = 100$.

Remark 22.2. ***Algorithm (Restarted GMRES)***

Input: $x_0 \in \mathbb{C}^n$, $b \in \mathbb{C}^n$, a routine to compute $q = Av$, a convergence tolerance `tol`, the restart parameter k_0

1. Set $\rho_0 := \|b - Ax_0\|_2$
2. Run GMRES until
   (a) $\frac{\|r_k\|_2}{\rho_0} \leq$ `tol` $\Rightarrow$ **stop**, $x_k \approx A^{-1}b$.
   (b) $k = k_0$ is reached $\Rightarrow$ set $x_0 := x_{k_0}$ and repeat step #2.



23 3-5-12

23.1 Domain Decomposition

Basic idea: Solve the PDE

$$\begin{aligned} Lu &= f \quad \text{in } R \\ u &= g \quad \text{on } \partial R \end{aligned} \tag{23.1}$$

by solving p subproblems:

$$\begin{aligned} Lu_i &= f \quad \text{in } R_i, \qquad i = 1, 2, \ldots, p \\ u_i &= g_i \quad \text{on } \partial R_i \end{aligned}$$

where $R = R_1 \cup R_2 \cup \cdots \cup R_p$.

Motivation:

- Parallel computing
- "Different" physics in subdomains R_i (e.g. different constants)
- Proof technique (historical)

23.1.1 Classical Alternating Schwarz Method

(Schwarz, 1870)

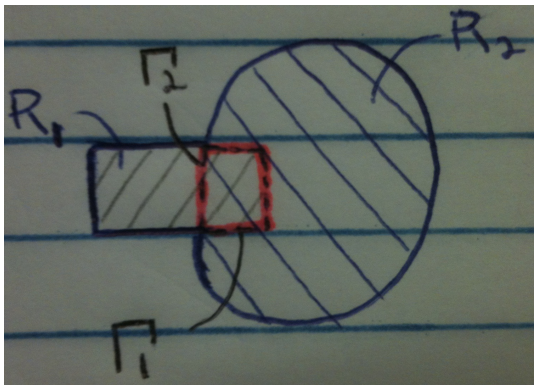


Figure 6: $p = 2$. $R = R_1 \cup R_2$. Γ_1, Γ_2 are artificial boundaries: $\Gamma_1, \Gamma_2 \cap \partial R = \emptyset$.

Guess $u_2^{(0)}\big|_{\Gamma_1} \approx u\big|_{\Gamma_1}$. For $n = 1, 2, \ldots,$

1. Solve $Lu_1^{(n)} = f$ in R_1

$$u_1^{(n)} = \begin{cases} g & \text{on } \partial R_1 \setminus \Gamma_1 \\ u_2^{(n-1)} & \text{on } \Gamma_1 \end{cases}$$

   for $u_1^{(n)}$.

2. Solve $Lu_2^{(n)} = f$ in R_2

$$u_2^{(n)} = \begin{cases} g & \text{on } \partial R_2 \setminus \Gamma_2 \\ u_1^{(n)} & \text{on } \Gamma_2 \end{cases}$$

3. (Repeat up to p...)

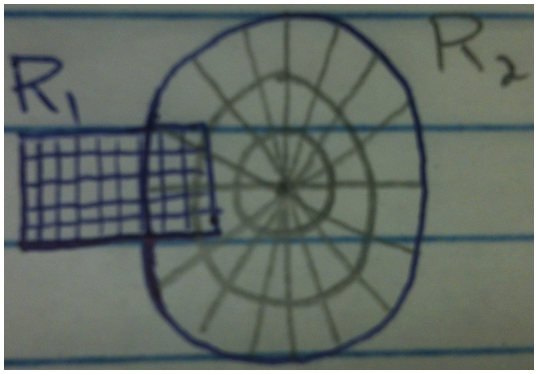
Schwarz showed that

$$\lim_{n\to\infty} u_n = u = \text{ solution of (23.1)},$$

where

$$u_n = \begin{cases} u_1^{(n)} & \text{in } R_1 \setminus R_2 \\ u_2^{(n)} & \text{in } R_2 \setminus R_1 \end{cases}$$

23.1.2 Discretized Problem (Nonmatching Grids)



We define

$$\Sigma_i = \partial R_i \setminus \Gamma_i.$$

We have 3 types of points.

$$u_i \to v_i = \begin{bmatrix} v_{R_i} \\ v_{\Sigma_i} \\ v_{\Gamma_i} \end{bmatrix} \begin{array}{l} \leftarrow \text{ interior grid points of } R_i \\ \leftarrow \text{ true boundary grid points} \\ \leftarrow \text{ artificial boundary grid points} \end{array}$$

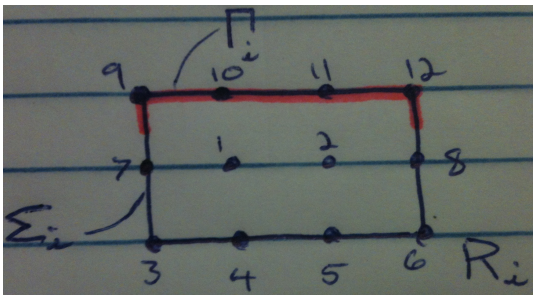
$Lu_i = f$ on $R_i \to A_i v_i = f_i$

Example 23.1.

$$Lu = -u_{xx} - u_{yy}$$

5-point stencil.

$$A_i = \begin{bmatrix} A_{R_i} & A_{\Sigma_i} & A_{\Gamma_i} \end{bmatrix}$$



$$\left[\begin{array}{cc|cccccc|cccc} 4 & -1 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 \\ -1 & 4 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 \end{array}\right] \left[\begin{array}{c} v_1 \\ v_2 \\ \hline v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \\ v_8 \\ \hline v_9 \\ v_{10} \\ v_{11} \\ v_{12} \end{array}\right] = h^2 \begin{bmatrix} f_1 \\ f_2 \end{bmatrix}$$

Notes:

1. A_i is rectangular; A_{R_i} is square (and nonsingular)
2. We need interpolation operators, $I_{R_1}^{\Gamma_2}$ and $I_{R_2}^{\Gamma_1}$, to obtain discrete artificial boundary conditions on Γ_1 and Γ_2

24 3-7-12

24.1 Domain Decomposition (Continued)

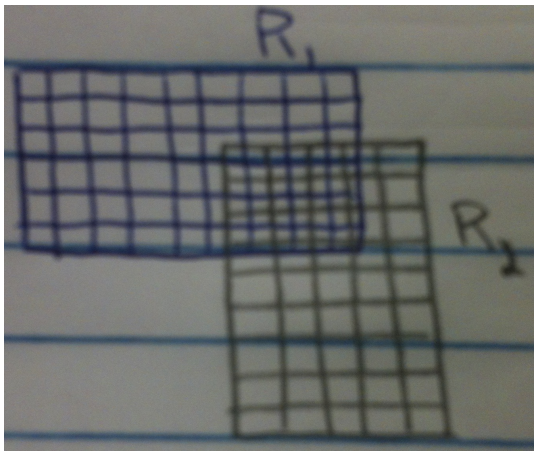


Figure 7: $R = R_1 \cup R_2,\ \Sigma_1 = \partial R_1 \setminus R_2$

Original Problem:

$$\begin{aligned} Lu &= f \qquad \text{in } R \\ u &= g \qquad \text{on } \partial R \end{aligned}$$

Discretized Problem:

on R_1	on R_2
$A_1 v_1 = f_1$	$A_2 v_2 = f_2$
$v_{\Sigma_1} = g_1$	$v_{\Sigma_2} = g_2$
$v_{\Gamma_1} = I_{R_2}^{\Gamma_1}(v_2)_{R_2}$	$v_{\Gamma_2} = I_{R_1}^{\Gamma_2}(v_1)_{R_1}$

Remark 24.1. ***Algorithm (Discretized Alternating Schwarz Method for 2 Subdomains)***

Choose initial guess for $v_{R_2}^{(0)}$, e.g., $v_{R_2}^{(0)} = \mathbf{0}$.

for $n = 1, 2, \ldots,$:

1. Solve

$$\begin{aligned} A_1 v_1^{(n)} &= f_1 \\ v_{\Sigma_1}^{(n)} &= g_1 \\ v_{\Gamma_1} &= I_{R_2}^{\Gamma_1} v_{R_2}^{(n-1)} \end{aligned}$$

   for $v_1^{(n)}$.

2. Solve

$$\begin{aligned} A_2 v_2^{(n)} &= f_2 \\ v_{\Sigma_2}^{(n)} &= g_2 \\ v_{\Gamma_2}^{(n)} &= I_{R_1}^{\Gamma_2} v_{R_1}^{(n)} \end{aligned}$$

   for $v_2^{(n)}$.

3. If $\|v_i^{(n)} - v_i^{(n-1)}\|$ and $\|v_{\Gamma_i}^{(n)} - v_{\Gamma_i}^{(n-1)}\|$, $i = 1, 2, \ldots,$ are "small," **stop**

end (n)

Note:

$$\left.\begin{aligned} A_1 v_1^{(n)} &= f_1 \\ v_{\Sigma_1}^{(n)} &= g_1 \\ v_{\Gamma_1}^{(n)} &= I_{R_2}^{\Gamma_1} v_{R_2}^{(n-1)} \end{aligned}\right\} \quad \Leftrightarrow \quad \begin{aligned} A_{R_1} v_{R_1}^{(n)} &= f_1 - A_{\Sigma_1} \underbrace{v_{\Sigma^2}^{(n)}}_{=s_1} - A_{\Gamma_1} v_{\Gamma_1}^{(n)} \\ &= \tilde{f}_1 - A_{\Gamma_1} I_{R_2}^{\Gamma_1} v_{R_2}^{(n-1)} \\ &\text{where } \tilde{f}_1 = f_1 - A_{\Sigma_1} g_1 \end{aligned}$$

2 solves in step n of the above algorithm:

$$\begin{aligned} A_{R_1} v_{R_1}^{(n)} &= \tilde{f}_1 - A_{\Gamma_1} I_{R_2}^{\Gamma_1} v_{R_2}^{(n-1)} \\ A_{R_2} v_{R_2}^{(n)} &= \tilde{f}_2 - A_{\Gamma_2} I_{R_1}^{\Gamma_2} v_{R_1}^{(n)} \end{aligned}$$

$\Leftrightarrow$ step n of block Gauss-Seidel applied to the linear system

$$\underbrace{\begin{bmatrix} A_{R_1} & A_{\Gamma_1} I_{R_2}^{\Gamma_1} \\ A_{\Gamma_2} I_{R_1}^{\Gamma_2} & A_{R_2} \end{bmatrix}}_{=: \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}} \begin{bmatrix} v_{R_1} \\ v_{R_2} \end{bmatrix} = \begin{bmatrix} \tilde{f}_1 \\ \tilde{f}_2 \end{bmatrix}$$

$$v_{R_i} = \lim_{n \to \infty} v_{R_i}^{(n)}, \qquad i = 1, 2, \ldots$$

Block Gauss-Seidel for $Av = \tilde{f}$, where

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}, \qquad \text{with } A_{11} \text{ and } A_{22} \text{ nonsingular,}$$

$$\tilde{f} = \begin{bmatrix} \tilde{f}_1 \\ \tilde{f}_2 \end{bmatrix}$$

$$M := \begin{bmatrix} A_{11} & 0 \\ A_{21} & A_{22} \end{bmatrix}$$

$$Av = \tilde{f} \quad \Leftrightarrow \quad Mv = (M - A)v + \tilde{f}$$

So at step n of Gauss-Seidel, we get $v^{(n)}$ from

$$Mv^{(n)} = (M - A)v^{(n-1)} + \tilde{f}.$$

$$\begin{bmatrix} A_{11} & 0 \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} v_1^{(n)} \\ v_2^{(n)} \end{bmatrix} = \begin{bmatrix} \tilde{f}_1 - A_{12} v_2^{(n-1)} \\ \tilde{f}_2 \end{bmatrix}$$

Remark 24.2. ***Algorithm (Alternating Schwarz as Block Gauss-Seidel)***

1. Choose initial guesses for $v_{R_1}^{(0)}$ and $v_{R_2}^{(0)}$, e.g., set $v_{R_1}^{(0)} = 0$ and $v_{R_2}^{(0)} = 0$. Set $\tilde{f}_i = f_i - A_{\Sigma_i} g_i$.
2. Solve

$$\begin{bmatrix} A_{R_1} & A_{\Gamma_1} I_{R_2}^{\Gamma_1} \\ A_{\Gamma_2} I_{R_1}^{\Gamma_2} & A_{R_2} \end{bmatrix} \begin{bmatrix} v_{R_1} \\ v_{R_2} \end{bmatrix} = \begin{bmatrix} \tilde{f}_1 \\ \tilde{f}_2 \end{bmatrix}$$

by block Gauss-Seidel, with $\begin{bmatrix} v_{R_1}^{(0)} \\ v_{R_2}^{(0)} \end{bmatrix}$ as initial guesses.

Notes:

1. A is not symmetric in general.
2. For self-adjoint elliptic PDE's: $A_{R_1} \succ 0$ and $A_{R_2} \succ 0$

25 3-9-12

25.1 Homework 5 Comments

Generalize $T_m V + V T_m = h^2 F$ to an $m_x \times m_y$ grid, $m_x \neq m_y$. We have $h_x = \frac{1}{m_x+1}$, $h_y = \frac{1}{m_y+1}$,

$$\underbrace{T_{m_y}}_{m_y \times m_y} V + \alpha V \underbrace{T_{m_x}}_{m_x \times m_x} = \tilde{F}$$

Also, at some point we will need to use the feature of GMRES in Matlab where we can specify a routine instead of a matrix A.

25.2 Alternating Schwarz as a Preconditioner

Remark 25.1. ***Algorithm***

Input: your favorite Krylov subspace method for nonsymmetric $A, M \Rightarrow$ e.g. GMRES

1. Choose initial guesses for $v_{R_1}^{(0)}$ and $v_{R_2}^{(0)}$, e.g. $v_{R_1}^{(0)} = v_{R_2}^{(0)} = \mathbf{0}$. Set $\tilde{f}_1 = f_1 - A_{\Sigma_1} g_1$ and $\tilde{f}_2 = f_2 - A_{\Sigma_2} g_2$. (Recall: $\partial R_i = \Sigma_i \cup \Gamma_i$.)
2. Solve

$$\underbrace{\begin{bmatrix} A_{R_1} & A_{\Gamma_1} I_{R_2}^{\Gamma_1} \\ A_{\Gamma_2} I_{R_1}^{\Gamma_2} & A_{R_2} \end{bmatrix}}_{=A} \begin{bmatrix} v_{R_1} \\ v_{R_2} \end{bmatrix} = \begin{bmatrix} \tilde{f}_1 \\ \tilde{f}_2 \end{bmatrix}$$

with the chosen Krylov subspace method with initial guess $\begin{bmatrix} v_{R_1}^{(0)} \\ v_{R_2}^{(0)} \end{bmatrix}$ and preconditioner

$$M = \begin{bmatrix} A_{R_1} & 0 \\ A_{\Gamma_2} I_{R_1}^{\Gamma_2} & A_{R_2} \end{bmatrix}$$

Notes:

1. The preconditioner is applied from the left:

$$A' = M^{-1}A = \begin{bmatrix} * & * \\ * & * \end{bmatrix} \qquad \text{(can be solved by hand)}$$

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$$

$$M = \begin{bmatrix} A_{11} & \mathbf{0} \\ A_{21} & A_{22} \end{bmatrix}$$

$$M^{-1} = \begin{bmatrix} A_{11}^{-1} & \mathbf{0} \\ \mathbf{X} & A_{22}^{-1} \end{bmatrix}$$

$$A_{21}A_{11}^{-1} + A_{22}\mathbf{X} = \mathbf{0}$$

$$\mathbf{X} = -A_{22}^{-1}A_{21}A_{11}^{-1}$$

$$M^{-1} = \begin{bmatrix} A_{11}^{-1} & \mathbf{0} \\ -A_{22}^{-1}A_{21}A_{11}^{-1} & A_{22}^{-1} \end{bmatrix}$$

$$M^{-1}A = \begin{bmatrix} A_{11}^{-1} & \mathbf{0} \\ -A_{22}^{-1}A_{21}A_{11}^{-1} & \end{bmatrix} \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} = \begin{bmatrix} \mathbf{0} & A_{11}^{-1}A_{12} \\ \cancel{-A_{22}^{-1}A_{21} + A_{22}^{-1}A_{21}} & I - A_{22}^{-1}A_{21}A_{11}^{-1}A_{12} \end{bmatrix}$$

$$= \mathbf{0}$$

$$M^{-1}A = \begin{bmatrix} I & A_{11}^{-1}A_{12} \\ \mathbf{0} & I - A_{22}^{-1}A_{21}A_{11}^{-1}A_{12} \end{bmatrix}$$

$$M^{-1}A \begin{bmatrix} p_1 \\ p_2 \end{bmatrix} = \begin{bmatrix} p_1 + \overbrace{A_{11}^{-1}A_{12}p_2}^{t_1} \\ p_2 - \underbrace{A_{22}^{-1}A_{21}A_{11}^{-1}A_{12}p_2}_{t_2} \end{bmatrix}$$

$$A_{11}t_1 = A_{12}p_2 \qquad A_{R_1} \rightarrow \text{ fast solver}$$

$$A_{22}t_2 = A_{21}t_1$$

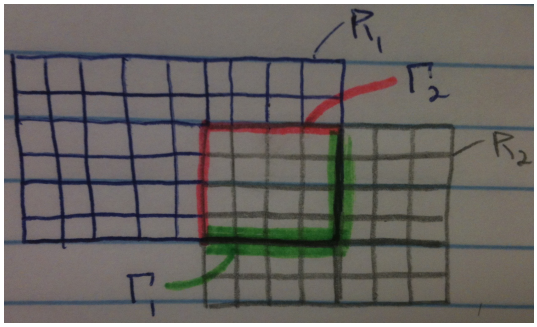
$$q = M^{-1}Ap = \begin{bmatrix} p_1 + t_1 \\ p_2 - t_2 \end{bmatrix}$$

Alternating Schwarz is not parallel ⇒ like multiplicative Schwarz for matching grids.
Apply GMRES to A'!

2. If grids in R_1 and R_2 match, then $I_{R_2}^{\Gamma_1}$ and $I_{R_1}^{\Gamma_2}$ are just "pruned" identity matrices. For example,

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

25.3 Matching Grids



$$
\begin{aligned}
& R = R_1 \cup R_2 \\
& Lu = f \qquad \text{in } R = R_1 \cup R_2 \\
& u = g \qquad \text{on } \partial R \\
\Rightarrow \qquad & Av = b
\end{aligned}
$$

Often $A \succ 0$, e.g. $Lu = -u_{xx} - u_{yy}$.

Order the unknowns vector v such that

$$
v = \begin{bmatrix} v_{R_1 \setminus \overline{R_2}} \\ v_{\Gamma_2} \\ v_{R_1 \cap R_2} \\ v_{\Gamma_1} \\ V_{R_2 \setminus \overline{R_1}} \end{bmatrix}
$$

Take $I_1 = \begin{bmatrix} I & \mathbf{0} \end{bmatrix}$ such that

$$
Iv_1 = \begin{bmatrix} v_{R_1 \setminus \overline{R_2}} \\ v_{\Gamma_2} \\ v_{R_1 \cap R_2} \end{bmatrix} = \text{ unknowns in } R_1.
$$

Similarly, take $I_2 = \begin{bmatrix} \mathbf{0} & I \end{bmatrix}$ such that

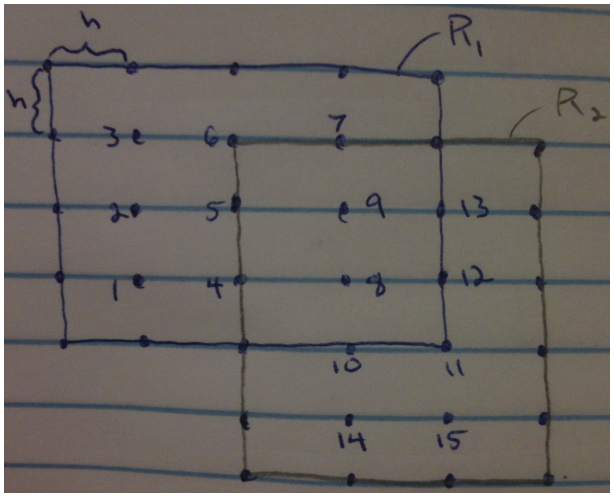
$$
I_2 v = \begin{bmatrix} v_{R_1 \cap R_2} \\ v_{\Gamma_1} \\ v_{R_2 \setminus \overline{R_1}} \end{bmatrix} = \text{ unknowns in } R_2.
$$

- $A_1 := I_1 A I_1^T$ (discretization in R_1)
- $A_2 := I_2 A I_2^T$ (discretization in R_2)

$$
A = \left[\begin{array}{c|c} A_1 & * \\ \hline * & A_2 \end{array}\right]
$$

Example 25.2.

$$\begin{aligned} -u_{xx} - u_{yy} &= f \qquad \text{in } R = R_1 \cup R_2 \\ u &= g \qquad \text{on } \partial R \end{aligned}$$



$$v = \begin{bmatrix} v_{R_1 \setminus \overline{R_2}} & = 1:3 \\ v_{\Gamma_2} & = 4:7 \\ v_{R_1 \cap R_2} & = 8:9 \\ v_{\Gamma_1} & = 10:13 \\ V_{R_2 \setminus \overline{R_1}} & = 14:15 \end{bmatrix}$$

26 3-12-12

26.1 Alternating Schwarz Example (Continued)

Continuing from last time...

Example 26.1. ***Example 25.2 Continued...***

$$\begin{aligned} -u_{xx} - u_{yy} &= f \qquad \text{in } R = R_1 \cup R_2 \\ u &= g \qquad \text{on } \partial R \end{aligned}$$

Using the 5 point stencil, we get a linear system $Av = b$.

$$\left[\begin{array}{ccc|ccc|ccc|cccc|cc}
4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 4 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
-1 & 0 & 0 & 4 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & -1 & 4 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 0 & -1 & 4 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 4 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & -1 & 0 & 0 & 0 & 4 & -1 & -1 & 0 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 0 & -1 & -1 & 4 & 0 & 0 & 0 & -1 & 0 & 0 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 4 & -1 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 4 & -1 & 0 & 0 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 4 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 4 & 0 & 0 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 4 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 4
\end{array}\right]
\left[\begin{array}{c} v_1 \\ v_2 \\ v_3 \\ \hline v_4 \\ v_5 \\ v_6 \\ v_7 \\ \hline v_8 \\ v_9 \\ \hline v_{10} \\ v_{11} \\ v_{12} \\ v_{13} \\ \hline v_{14} \\ v_{15} \end{array}\right]
= h^2
\left[\begin{array}{c} f_1 + g_{11} + g_{12} \\ f_2 + g_{21} \\ f_3 + g_{31} + g_{32} \\ f_4 + g_{41} \\ f_5 \\ f_6 + g_{61} \\ f_7 + g_{71} + g_{72} \\ f_8 \\ f_9 \\ f_{10} + g_{10,1} \\ f_{11} + g_{11,1} \\ f_{12} + g_{12,1} \\ f_{13} + g_{13,1} + g_{13,2} \\ f_{14} + g_{14,1} + g_{14,2} \\ f_{15} + g_{15,1} + g_{15,2} \end{array}\right]$$

Example 26.2. ***Example 25.2 Continued (Again)...***

Note that our A matrix has 2 blocks. The upper left block corresponds to A_1, and the lower right block corresponds to A_2. (If we had p subdomains, then A would have p blocks.)

$$\begin{aligned} I_1 &= \begin{bmatrix} I_{9\times 9} & \mathbf{0}_{9\times 6} \end{bmatrix}, \qquad I_1 : R \to R_1 \\ I_2 &= \begin{bmatrix} \mathbf{0}_{8\times 7} & I_{8\times 8} \end{bmatrix}, \qquad I_2 : R \to R_2 \\ A_1 &= I_1 A I_1^T \\ A_2 &= I_2 A I_2^T \end{aligned}$$

26.2 Multiplicative Schwarz Method

Initial guess $v^{(0)}$.

$$
\begin{aligned}
Av = b, \qquad & M \approx A,\ Mv = (M-A)v + b \\
Mv^{(n+1)} &= (M-A)v^{(n)} + b \\
v^{(n+1)} &= v^{(n)} + M^{-1}(b - Av^{(n)}) \\
e^{(n)} &= A^{-1}b - v^{(n)} \qquad\qquad \text{(error)} \\
e^{(n+1)} &= (I - M^{-1}A)e^{(n)}
\end{aligned}
$$

for $n = 0, 1, 2, \ldots$**:**

$$
\begin{aligned}
v^{\left(n+\frac{1}{2}\right)} &= v^{(n)} + I_1^T A_1^{-1} I_1 \underbrace{(b - Av^{(n)})}_{=Ae^{(n)}} \\
v^{(n+1)} &= v^{\left(n+\frac{1}{2}\right)} + I_2^T A_2^{-1} I_2 (b - Av^{\left(n+\frac{1}{2}\right)})
\end{aligned}
$$

The corresponding iteration matrix:

$$
\begin{aligned}
e^{\left(n+\frac{1}{2}\right)} &= A^{-1}b - v^{\left(n+\frac{1}{2}\right)} = e^{(n)} - I_1^T A^{-1} I_1 A e^{(n)} = (I - I_1^T A^{-1} I_1 A)e^{(n)} \\
e^{(n+1)} &= (I - I_2^T A_2^{-1} I_2 A)e^{\left(n+\frac{1}{2}\right)}
\end{aligned}
$$

The error propogation of this method is

$$
\begin{aligned}
e^{(n+1)} &= (I - P_2)(I - P_1)e^{(n)}, \\
\text{where} \qquad P_i &:= I_i^t A_i^{-1} I_i A, \qquad i = 1, 2, \ldots
\end{aligned}
$$

The iteration matrix is a product, which is why this method has come to be known as the multiplicative Schwarz method.

For "sufficient" overlap of R_1 and R_2, we get linear convergence:

$$
\|e^{(n)}\| \leq \rho^n \|e^{(0)}\|,
$$

where $\rho < 1$ is <u>independent</u> of the mesh sizes h_x and h_y.

27 3-14-12

27.1 Multiplicative Schwarz as a Preconditioner

$$e^{(n+1)} = (I - P_2)(I - P_1)e^{(n)} \tag{27.1}$$
$$\text{where} \qquad P_i = B_i A, \qquad B_i := I_i^T A_i^{-1} I_i$$

General iterative method:

$$e^{(n+1)} = (I - M^{-1}A)e^{(n)}, \qquad M \ (\approx A) \text{ is nonsingular}$$

Accelerate converge: Krylov subspace method with preconditioner M. What is M for (27.1)?

$$\begin{aligned}(I - P_2)(I - P_1) &= I - P_1 - P_2 + P_2P_1 \\ &= I - \underbrace{(B_1 + B_2 - B_2AB_1)}_{=M^{-1}} A \\ M &:= (B_1 + B_2 - B_2AB_1)^{-1} \\ B_1 &= \begin{bmatrix} A_1^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \\ B_1 + B_2 &= \begin{bmatrix} A_1^{-1} & & \\ & & A_2^{-1} \end{bmatrix} \qquad \text{(Note: there is some overlap)}\end{aligned}$$

Use M as a preconditioner.

$$Mq = v \qquad \Leftrightarrow \qquad q = (B_1 + B_2 - B_2AB_1)v$$

Note: $A \succ 0$, $B_1 + B_2 \succ 0$, but in general $M \neq M^T$ so $M \not\succ 0$. Thus, we cannot use CG! So we have to use a nonsymmetric Krylov method, e.g. GMRES. This isn't so bad because we shouldn't need a whole lot of iterations.

The B_2AB_1 term is a "bad guy" because it destroys the symmetry of M and it prevents parallel computing.

27.2 Additive Schwarz Method

Intial guess $v^{(0)}$.
for $n = 0, 1, 2, \ldots,$**:**

$$\begin{aligned}v^{(n+\frac{1}{2})} &= v^{(n)} + B_1(b - Av^{(n)}) \\ v^{(n+1)} &= v^{(n+\frac{1}{2})} + B_2(b - Av^{(n)})\end{aligned}$$

Now:

$$\begin{aligned}e^{(n+1)} &= (I - (B_1 + B_2)A)^{-1}e^{(n)} \\ M &:= (B_1 + B_2)^{-1} \succ 0\end{aligned}$$

We can use M as a preconditioner for CG. M is called the additive Schwarz preconditioner.

Note:

$$Mq = v \qquad \Leftrightarrow \qquad q = (B_1 + B_2)v = \left(\begin{bmatrix} A_1^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & A_2^{-1} \end{bmatrix} \right) v$$

$$v = \begin{bmatrix} v_1 \\ (\text{overlap}) \\ v_2 \end{bmatrix}$$

$$q = \begin{bmatrix} A_1^{-1}v_1 \\ \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ A_2^{-1}v_2 \end{bmatrix}$$

The subdomain solves for R_1 and R_2 can be done in parallel!

Note: All of this extends to p subdomains: $R_1, R_2, \ldots, R_p$.

Notes for the Homework and Final Project

- Matlab's backslash would work, but it is not allowed!
- We can use Matlab's built-in GMRES and CG.

27.3 The Lanczos Process

Given: $A \in \mathbb{C}^{n\times n},\ r_0 \in \mathbb{C}^n$.
Goal: Generate vectors $v_1, v_2, \ldots, v_k, \ldots$, such that

$$K_k(A, r_0) = \operatorname{span}\{v_1, v_2, \ldots, v_k\}, \qquad k = 1, 2, \ldots, d(A, r_0)$$

using recurrences of fixed length. (The Arnoldi process produces orthonormal basis vectors, but at each step it requires information from all prior steps. It does not minimize residuals because, as we know, that would not be possible with fixed recurrences.)

$$\begin{aligned} v_j &= A^{j-1} r_0 \\ v_{k+1} &= A v_k \\ \gamma_{k+1} v_{k+1} &= A v_k - \alpha_k v_k - \beta_k v_{k-1} \end{aligned}$$

28 3-16-12

28.1 The Lanczos Process (Continued)

$A \in \mathbb{C}^{n\times n},\ r_0 \in \mathbb{C}^n,\ K_k(A, r_0) = \operatorname{span}\{v_1, v_2, \ldots, v_k\}$.

Recall: The Arnoldi process generates orthonormal v_k's using recurrences (in matrix form):

$$\begin{aligned} AV_k &= V_{k+1}\tilde{H}_k \\ \text{where} \quad V_k &= \begin{bmatrix} v_1 & v_2 & \cdots & v_k \end{bmatrix} \\ \text{and} \quad \tilde{H}_k &= \begin{bmatrix} h_{11} & h_{12} & \cdots & \cdots & h_{1k} \\ h_{21} & h_{22} & h_{23} & & \vdots \\ & h_{32} & \ddots & \ddots & \vdots \\ & & \ddots & \ddots & h_{k-1,k} \\ & & & \ddots & h_{k,k} \\ 0 & & & & h_{k+1,k} \end{bmatrix} \in \mathbb{C}^{k+1\times k},\ h_{j+1,j} > 0 \end{aligned} \tag{28.1}$$

Orthonormality of the v_k's $\Leftrightarrow V_k^H V_k = I_k,\ V_k^H V_{k+1} = \begin{bmatrix} I_k & \mathbf{0} \end{bmatrix}$.

Multiply (28.1) on the left by V_k^H:

$$V_k^H A V_k = \begin{bmatrix} I_k & \mathbf{0} \end{bmatrix} \tilde{H}_k = \begin{bmatrix} h_{11} & * & \cdots & * \\ h_{21} & \ddots & & \vdots \\ & \ddots & \ddots & * \\ 0 & & h_{k,k-1} & h_{kk} \end{bmatrix} =: H_k \tag{28.2}$$

Theorem 28.1.

The matrix H_k (produced by k steps of the Arnoldi process) is the projection of A onto the kth Krylov subspace:

$$V_k^H A V_k = H_k.$$

Special case: $A = A^H$.
Then $H_k = V_k^H A V_k = V_k^H A^H V_k = (V_k^H A V_k)^H = H_k^H$. Thus, H_k is Hermitian $\Rightarrow$ tridiagonal, by (28.2).

Since $h_{j+1,j} > 0,\ j = 1, 2, \ldots, k-1$, it follows that

$$(H_k =:) \underbrace{T_k}_{\text{tridiagonal}} = \begin{bmatrix} \alpha_1 & \beta_2 & & & \\ \beta_2 & \alpha_2 & \beta_3 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \beta_k \\ & & & \beta_k & \alpha_k \end{bmatrix} \in \mathbb{R}^{k\times k}$$

(28.1) for $A = A^H$:
$$\begin{aligned} AV_k &= V_k T_k + \begin{bmatrix} 0 & \cdots & 0 & \beta_{k+1} v_{k+1} \end{bmatrix} \\ Av_j &= \beta_j v_{j-1} + \alpha_j v_j + \beta_{j+1} v_{j+1}, \qquad j = 1, 2, \ldots, k \\ \beta_{j+1} v_{j+1} &= Av_j - \alpha_j v_j - \beta_j v_{j-1}, \qquad j = 1, 2, \ldots, k \end{aligned}$$

This means that for $A = A^H$, the orthonormal vectors v_k can be generated via a 3-term recurrence!

Thus, the Arnoldi process for $A = A^H$ is the Hermitian Lanczos process.

Remark 28.2. ***Algorithm (Hermitian Lanczos Process)***

Input: $r_0 \in \mathbb{C}^n$, a routine to compute $q = Av$ for any $v \in \mathbb{C}^n$ (where $A = A^H$)

Set $\beta_1 = \|r_0\|_2$.
if $\beta_1 = 0$**, stop:** $r_0 = 0$
Otherwise, set $v_1 = \frac{r_0}{\beta_1}$, $v_0 := \mathbf{0} \in \mathbb{C}^n$.
for $k = 1, 2, \ldots,$ **do:**

- Compute $q = Av$
- Set $q = q - \beta_k v_{k-1}$ $(= Av_k - \beta_k v_{k-1})$
- Compute $\alpha_k = v_k^H q$
- Set $q = q - \alpha_k v_k$ $(= Av_k - \beta_k v_{k-1} - \alpha_k v_k)$
- Set $\beta_{k+1} = \|q\|_2$
- **if** $\beta_{k+1} = 0$**, stop:** the Krylov subspace $K_k(A, r_0)$ has reached its maximum dimension, i.e., $k = d(A, r_0)$
- Set $v_{k+1} = \frac{q}{\beta_{k+1}}$

end (k)

Output: Orthonormal vectors $v_1, v_2, \ldots, v_k$ and projection T_k.

Notes:

1. Unlike Arnoldi, the work per kth iteration is constant:
   - 1 product Av
   - 2 inner products
   - 2 SAXPY's
   - 1 division of a vector by a scalar
2. For $A \succ 0$, then $r_{k-1}^{\mathrm{CG}} = \underbrace{\mu_k}_{\in\mathbb{C}} v_k,\ k = 1, 2, \ldots,$.
3. In finite-precision arithmetic, the vectors v_k gradually lose orthogonality $\Rightarrow$ you may need to run more iterations than the theory predicts.

29 3-19-12

Office Hours:

- Today 12:30-1:30
- Wednesday 10-11
- Friday 11-12

29.1 The Lanczos Process (Continued)

The general case: $A \in \mathbb{C}^{n\times n}$. We can still have

$$AV_k = V_k T_k + \underbrace{\begin{bmatrix} 0 & \cdots & 0 & \beta_{k+1} v_{k+1} \end{bmatrix}}_{=\beta_{k+1} v_{k+1} e_k^T} \tag{29.1}$$

where

$$V_k = \begin{bmatrix} v_1 & v_2 & \cdots & v_k \end{bmatrix}$$

$$e_k := \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} \in \mathbb{R}^k$$

$$T_k = \text{ tridiagonal}$$

But the v_k's are no longer orthonormal. We introduce a second sequence,

$$w_1, w_2, \ldots, w_k, \ldots$$

such that

$$\text{span}\,\{w_1, w_2, \ldots, w_k\} = K_k(A^T, c), \qquad k = 1, 2, \ldots, d(A^T, c)$$

where $c \in \mathbb{C}^n$ is a "left" starting vector. (In the complex case, $A^T \to A^H,\ c \to \overline{c},\ w_j \to \overline{w_j}$.)

The recurrence relations in compact form are:

$$A^T W_k = W_k \tilde{T}_k + \gamma_{k+1} w_{k+1} e_k^T \tag{29.2}$$

where $W_k = \begin{bmatrix} w_1 & w_2 & \cdots & w_k \end{bmatrix}$ and $\tilde{T}_k$ is tridiagonal.

Notation: The v_k's are called the right Lanczos vectors, and the w_k's are called the left Lanczos vectors.

The v_k's and w_k's are constructed to be *bi-orthogonal*:

$$w_j^T v_k = 0 \qquad \text{for all } j \neq k,\ j, k = 1, 2, \ldots.$$

If $A = A^H$, $c = \overline{r_0}$, then $w_j = \overline{w_j}$, and thus

$$v_j^H v_k = 0 \qquad \text{for all } j \neq k.$$

We can still choose a normalization of the v_k's and w_k's. We use:

$$\|v_k\|_2 = 1, \quad \|w_k\|_2 = 1 \qquad \text{for all } k$$

Remark 29.1. ***Algorithm (Nonsymmetric Lanczos Process)***

Input: $r_0 \in \mathbb{C}^n$, $c \in \mathbb{C}^n$, a routine to compute $q = Av$ for any $v \in \mathbb{C}^n$, a routine to compute $s = A^T w$ for any $w \in \mathbb{C}^n$ (here $A \in \mathbb{C}^{n\times n}$)

Set $\beta_1 = \|r_0\|_2$ and $\gamma_1 = \|c\|_2$.
If $\beta_1 = 0$ **or** $\gamma_1 = 0$**, stop:** $r_0 = 0$ or $c = 0$.
Otherwise, set $v_1 = \frac{r_0}{\beta_1}$, $w_1 = \frac{c}{\gamma_1}$, $v_0 = w_0 = \mathbf{0} \in \mathbb{R}^n$, and $\delta_0 = 1$.

for $k = 1, 2, \ldots,$ **do:**

- Compute $\delta_k = w_k^T v_k$
- **If** $\delta_k = 0$**, stop:** "breakdown" of the algorithm
- Compute $q = Av_k$ and $s = A^T w_k$
- Set $q = q - \left(\gamma_k \frac{\delta_k}{\delta_{k-1}}\right) v_{k-1}$ ($\Leftarrow$ this guarantees that $w_{k-1}^T q = 0$)
- Set $\alpha_k = \frac{w_k^T q}{\delta_k}$, $q = q - \alpha_k v_k$ ($\Leftarrow$ this guarantees that $w_k^T q = 0$)
- Set $s = s - \alpha_k w_k$ and $s = s - \left(\beta_k \frac{\delta_k}{\delta_{k-1}}\right) w_{k-1}$ ($\Leftarrow$ this guarantees that $s^T v_k = 0,\ s^T v_{k-1} = 0$)
- Set $\beta_{k+1} = \|q\|_2$ and $\gamma_{k+1} = \|s\|_2$.
- **If** $\beta_{k+1} = 0$**, stop:** the Krylov subspace $K_k(A, r_0)$ has reached its maximum dimension, $k = d(A, r_0)$
- **If** $\gamma_{k+1} = 0$**, stop:** the Krylov subspace $K_k(A^T, c)$ has reached its maximum dimension, $k = d(A^T, c)$
- Set $v_{k+1} = \frac{q}{\beta_{k+1}}$ and $w_{k+1} = \frac{s}{\gamma_{k+1}}$

$$T_k = \begin{bmatrix} \alpha_1 & \eta_2 & & & 0 \\ \beta_2 & \alpha_2 & \eta_3 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \eta_k \\ 0 & & & \beta_k & \alpha_k \end{bmatrix}$$

$$\beta_{k+1} v_{k+1} = Av_k - \alpha_k v_k - \underbrace{\left(\gamma_k \frac{\delta_k}{\delta_{k-1}}\right)}_{=:\eta_k} v_{k-1}$$

$$\tilde{T}_k = \begin{bmatrix} \alpha_1 & \xi_2 & & & 0 \\ \gamma_2 & \alpha_2 & \xi_3 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \xi_k \\ 0 & & & \gamma_k & \alpha_k \end{bmatrix}$$

$$\xi_k := \beta_k \frac{\delta_k}{\delta_{k-1}}$$

Index