AM 205: lecture 25

- Last time: SOR, multigrid
- Today: Multigrid, Krylov subspace methods

Krylov Subspace Methods

We now give an overview of the role of Krylov[1] subspace methods in Scientific Computing

Given a matrix A and vector b, a Krylov sequence is the set of vectors

$$\{b, Ab, A^2b, A^3b, \ldots\}$$

The corresponding Krylov subspaces are the spaces spanned by successive groups of these vectors

$$\mathcal{K}_m(A, b) \equiv \text{span}\{b, Ab, A^2b, \ldots, A^{m-1}b\}$$

[1] Aleksey Krylov, 1863–1945, wrote a paper on this idea in 1931

Krylov Subspace Methods

Krylov subspaces are the basis for iterative methods for eigenvalue problems (and also for solving linear systems)

An important advantage: Krylov methods do not deal directly with A, but rather with matrix–vector products involving A

This is particularly helpful when A is large and sparse, since matrix–vector multiplications are relatively cheap

Also, Krylov sequence is closely related to power iteration, hence not surprising it is useful for solving eigenproblems

Arnoldi Iteration

Arnoldi Iteration

We define a matrix as being in Hessenberg form in the following way:

- A is called upper-Hessenberg if $a_{ij} = 0$ for all $i > j + 1$
- A is called lower-Hessenberg if $a_{ij} = 0$ for all $j > i + 1$

The Arnoldi iteration is a Krylov subspace iterative method that reduces A to upper-Hessenberg form

As we'll see, we can then use this simpler form to approximate some eigenvalues of A

Arnoldi Iteration

For $A \in \mathbb{C}^{n\times n}$, we want to compute $A = QHQ^*$, where H is upper Hessenberg and Q is unitary (*i.e.* $QQ^* = \mathrm{I}$)

However, we suppose that n is huge! Hence we do not try to compute the full factorization

Instead, let us consider just the first $m \ll n$ columns of the factorization $AQ = QH$

Therefore, on the left-hand side, we only need the matrix $Q_m \in \mathbb{C}^{n\times m}$:

$$Q_m = \left[\begin{array}{c|c|c|c} & & & \\ q_1 & q_2 & \dots & q_m \\ & & & \end{array}\right]$$

Arnoldi Iteration

On the right-hand side, we only need the first m columns of H

More specifically, due to upper-Hessenberg structure, we only need $\widetilde{H}_m$, which is the $(m+1) \times m$ upper-left section of H:

$$\widetilde{H}_m = \begin{bmatrix} h_{11} & & \cdots & h_{1m} \\ h_{21} & h_{22} & & \\ & \ddots & \ddots & \vdots \\ & & h_{m,m-1} & h_{mm} \\ & & & h_{m+1,m} \end{bmatrix}$$

$\widetilde{H}_m$ only interacts with the first $m+1$ columns of Q, hence we have

$$AQ_m = Q_{m+1}\widetilde{H}_m$$

Arnoldi Iteration

$$\left[\begin{array}{c} \\ A \\ \\ \end{array}\right]\left[\begin{array}{c|c|c} & & \\ q_1 & \cdots & q_m \\ & & \end{array}\right] = \left[\begin{array}{c|c|c} & & \\ q_1 & \cdots & q_{m+1} \\ & & \end{array}\right]\begin{bmatrix} h_{11} & \cdots & h_{1m} \\ h_{21} & \cdots & h_{2m} \\ & \ddots & \vdots \\ & & h_{m+1,m} \end{bmatrix}$$

The m^{th} column can be written as

$$Aq_m = h_{1m}q_1 + \cdots + h_{mm}q_m + h_{m+1,m}q_{m+1}$$

Or, equivalently

$$q_{m+1} = (Aq_m - h_{1m}q_1 - \cdots - h_{mm}q_m)/h_{m+1,m}$$

Arnoldi iteration is just the Gram–Schmidt method that constructs the h_{ij} and the (orthonormal) vectors q_j, $j = 1, 2, \ldots$

Arnoldi Iteration

```
 1: choose $b$ arbitrarily, then $q_1 = b/\|b\|_2$
 2: for $m = 1, 2, 3, \ldots$ do
 3:     $v = Aq_m$
 4:     for $j = 1, 2, \ldots, m$ do
 5:         $h_{jm} = q_j^* v$
 6:         $v = v - h_{jm} q_j$
 7:     end for
 8:     $h_{m+1,m} = \|v\|_2$
 9:     $q_{m+1} = v/h_{m+1,m}$
10: end for
```

This is akin to the modified Gram–Schmidt method because the updated vector v is used in line 5 (vs. the "raw vector" Aq_m)

Also, we only need to evaluate Aq_m and perform some vector operations in each iteration

Arnoldi Iteration

The Arnoldi iteration is useful because the q_j form orthonormal bases of the successive Krylov spaces

$$\mathcal{K}_m(A, b) = \text{span}\{b, Ab, \dots, A^{m-1}b\} = \text{span}\{q_1, q_2, \dots, q_m\}$$

We expect $\mathcal{K}_m(A, b)$ to provide good information about the dominant eigenvalues/eigenvectors of A

Note that this looks similar to the QR algorithm, but the QR algorithm was based on QR factorization of

$$\left[\begin{array}{c|c|c|c} & & & \\ A^k e_1 & A^k e_2 & \dots & A^k e_n \\ & & & \end{array}\right]$$

Arnoldi Iteration

Question: How do we find eigenvalues from the Arnoldi iteration?

Let $H_m = Q_m^* A Q_m$ be the $m \times m$ matrix obtained by removing the last row from $\widetilde{H}_m$

Answer: At each step m, we compute the eigenvalues of the Hessenberg matrix H_m (via, say, the QR algorithm)[2]

This provides estimates for m eigenvalues/eigenvectors ($m \ll n$) called Ritz values, Ritz vectors, respectively

Just as with the power method, the Ritz values will typically converge to extreme eigenvalues of the spectrum

[2]This is how `eigs` in Python/Matlab works

Arnoldi Iteration

We now examine why eigenvalues of H_m approximate extreme eigenvalues of A

Let[3] $\mathbb{P}^m_{\text{monic}}$ denote the monic polynomials of degree m

Theorem: The characteristic polynomial of H_m is the unique solution of the approximation problem: find $p \in \mathbb{P}^m_{\text{monic}}$ such that

$$\|p(A)b\|_2 = \text{minimum}$$

Proof: See Trefethen & Bau

[3] Recall that a monic polynomial has coefficient of highest order term of 1

Arnoldi Iteration

This theorem implies that Ritz values (*i.e.* eigenvalues of H_m) are the roots of the optimal polynomial

$$p^* = \arg \min_{p \in \mathbb{P}^m_{\text{monic}}} \|p(A)b\|_2$$

Now, let's consider what p^* should look like in order to minimize $\|p(A)b\|_2$

We can illustrate the important ideas with a simple case, suppose:

- A has only m ($\ll n$) distinct eigenvalues
- $b = \sum_{j=1}^m \alpha_j v_j$, where v_j is an eigenvector corresponding to λ_j

Arnoldi Iteration

Then, for $p \in \mathbb{P}^m_{\text{monic}}$, we have

$$p(x) = c_0 + c_1 x + c_2 x^2 + \cdots + x^m$$

for some coefficients $c_0, c_1, \ldots, c_{m-1}$

Applying this polynomial to a matrix A gives

$$\begin{aligned}
p(A)b &= \left(c_0 \mathrm{I} + c_1 A + c_2 A^2 + \cdots + A^m\right) b \\
&= \sum_{j=1}^{m} \alpha_j \left(c_0 \mathrm{I} + c_1 A + c_2 A^2 + \cdots + A^m\right) v_j \\
&= \sum_{j=1}^{m} \alpha_j \left(c_0 + c_1 \lambda_j + c_2 \lambda_j^2 + \cdots + \lambda_j^m\right) v_j \\
&= \sum_{j=1}^{m} \alpha_j p(\lambda_j) v_j
\end{aligned}$$

Arnoldi Iteration

Then the polynomial $p^* \in \mathbb{P}^m_{\text{monic}}$ with roots at $\lambda_1, \lambda_2, \ldots, \lambda_m$ minimizes $\|p(A)b\|_2$, since $\|p^*(A)b\|_2 = 0$

Hence, in this simple case the Arnoldi method finds p^* after m iterations

The Ritz values after m iterations are then exactly the m distinct eigenvalues of A

Arnoldi Iteration

Suppose now that there are more than m distinct eigenvalues (as is generally the case in practice)

It is intuitive that in order to minimize $\|p(A)b\|_2$, p^* should have roots close to the dominant eigenvalues of A

Also, we expect Ritz values to converge more rapidly for extreme eigenvalues that are well-separated from the rest of the spectrum

(We'll see a concrete example of this for a symmetric matrix A shortly)

Lanczos Iteration

Lanczos Iteration

Lanczos iteration is the Arnoldi iteration in the special case that A is hermitian

However, we obtain some significant computational savings in this special case

Let us suppose for simplicity that A is symmetric with real entries, and hence has real eigenvalues

Then $H_m = Q_m^T A Q_m$ is also symmetric $\implies$ Ritz values (*i.e.* eigenvalue estimates) are also real

Lanczos Iteration

Also, we can show that H_m is tridiagonal: Consider the ij entry of H_m, $h_{ij} = q_i^T A q_j$

Recall first that $\{q_1, q_2, \ldots, q_j\}$ is an orthonormal basis for $\mathcal{K}_j(A, b)$

Then we have $Aq_j \in \mathcal{K}_{j+1}(A, b) = \text{span}\{q_1, q_2, \ldots, q_{j+1}\}$, and hence $h_{ij} = q_i^T (Aq_j) = 0$ for $i > j + 1$ since

$$q_i \perp \text{span}\{q_1, q_2, \ldots, q_{j+1}\}, \text{ for } i > j + 1$$

Also, since H_m is symmetric, we have $h_{ij} = h_{ji} = q_j^T (Aq_i)$, which implies $h_{ij} = 0$ for $j > i + 1$, by the same reasoning as above

Lanczos Iteration

Since H_m is now tridiagonal, we shall write it as

$$T_m = \begin{bmatrix} \alpha_1 & \beta_1 & & & \\ \beta_1 & \alpha_2 & \beta_2 & & \\ & \beta_2 & \alpha_3 & \ddots & \\ & & \ddots & \ddots & \beta_{m-1} \\ & & & \beta_{m-1} & \alpha_m \end{bmatrix}$$

The consequence of tridiagonality: Lanczos iteration is much cheaper than Arnoldi iteration!

Lanczos Iteration

The inner loop in Lanczos iteration only runs from $m-1$ to m, instead of 1 to m as in Arnoldi

This is due to the three-term recurrence at step m:

$$Aq_m = \beta_{m-1}q_{m-1} + \alpha_m q_m + \beta_m q_{m+1}$$

(This follows from our discussion of the Arnoldi case, with $\widetilde{T}_m$ replacing $\widetilde{H}_m$)

As before, we rearrange this to give

$$q_{m+1} = (Aq_m - \beta_{m-1}q_{m-1} - \alpha_m q_m)/\beta_m$$

Lanczos Iteration

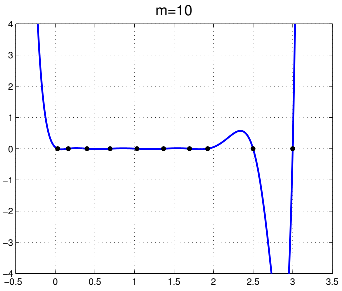
Which leads to the Lanczos iteration

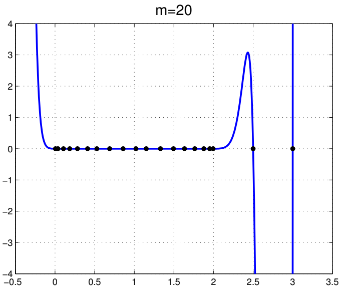
```
1: $\beta_0 = 0,\ q_0 = 0$
2: choose $b$ arbitrarily, then $q_1 = b/\|b\|_2$
3: **for** $m = 1, 2, 3, \ldots$ **do**
4:     $v = Aq_m$
5:     $\alpha_m = q_m^T v$
6:     $v = v - \beta_{m-1} q_{m-1} - \alpha_m q_m$
7:     $\beta_m = \|v\|_2$
8:     $q_{m+1} = v/\beta_m$
9: **end for**
```

Lanczos Iteration

Python demo: Lanczos iteration for a diagonal matrix

Lanczos Iteration





We can see that Lanczos minimizes $\|p(A)b\|_2$:

- p is uniformly small in the region of clustered eigenvalues
- roots of p match isolated eigenvalues very closely

Note that in general p will be very steep near isolated eigenvalues, hence convergence for isolated eigenvalues is rapid!