Recitation 9

Recitation Instructor: Shivam Verma
Email: shivamverma@nyu.edu
Ph: 718-362-7836
Office hours: WWH 605 (2.50 - 4.50 pm, Tuesdays)

Brief Overview

- Sturm Sequence
- QR method (description, MATLAB)
- Inverse Iteration method (description, MATLAB)
- Introduction to LAPACK
- Discussion about how LAPACK solves eigenvalue problems
- HW discussion

Sturm's sequence property

We used Householder's method last week to reduce a real symmetric matrix to a tridiagonal matrix. This time, given a tridiagonal matrix, we will find its eigenvalues.

$$T = \begin{pmatrix} a_1 & b_2 & & & & & & \\ b_2 & a_2 & b_3 & & & & & \\ & b_3 & a_3 & b_4 & & & & \\ & & \cdots & \cdots & \cdots & & & \\ & & & \cdots & \cdots & \cdots & & \\ & & & & \cdots & \cdots & \cdots & \\ & & & & & b_{n-1} & a_{n-1} & b_n \\ & & & & & & b_n & a_n \end{pmatrix}$$

We define the characteristic polynomials of the leading principal minors:

$p_1(x) = det([a_1] - xI) = a_1 - x$
$p_2(x) = det([a_1\ b_2;\ b_2\ a_2] - xI) = (a_2 - x)(a_1 - x) - {b_2}^2$
…
$p_r(x) = (a_r - x)p_{r-1}(x) - {b_r}^2 p_{r-2}(x), \quad r = 2, 3, ..., n$ with $p_0(x) = 1$.

The bisection method is given by:

- Consider an interval $[a_k, b_k]$, and take the midpoint $c_k = (a_k + b_k)/2$.
- Evaluate the Sturm chain/sequence at $c_0 : \{p_1(c_k), p_2(c_k), ..., p_n(c_k)\}$
- The number of sign agreements gives the number of eigenvalues strictly greater than $c_k \Rightarrow If\ r = \#\ of\ sign\ agreements\ \{+\ +\ -\ +\ ...\ -\} \rightarrow \lambda_1, ..., \lambda_r > c_k$
- To find the largest eigenvalue, choose the interval $[c_k, b_k]\ i.e.\ a_{k+1} = c_k,\ b_{k+1} = b_k,\ c_{k+1} = (a_{k+1} + b_{k+1})/2$
- Repeat till $r = 1$, and $|c_{k+1} - c_k| < \varepsilon$.
- $\lambda_1 = c_{k+1}$

Similarly, we can find the 2nd, 3rd, … m-th largest eigenvalues by choosing the lower interval $[a_k, c_k]$ when $r = m - 1$.

Besides textbook, see ref. [1, 2] for complete proof and explanation of Sturm's sequence property. Also see sec. 4.6.2 in [3], and [15] for a good understanding of the method.

QR method

- Basic QR method - $O(n^3)$
- Hessenberg QR - $O(n^2)$
    - Convert A to tridiagonal/Hessenberg form: $O(n^2)$ steps
    - Apply basic QR method
- See [5, 6] for an excellent discussion.
- Watch this YouTube video for an interesting depiction of how QR algorithm reaches the solution (using Gershgorin's theorem)!

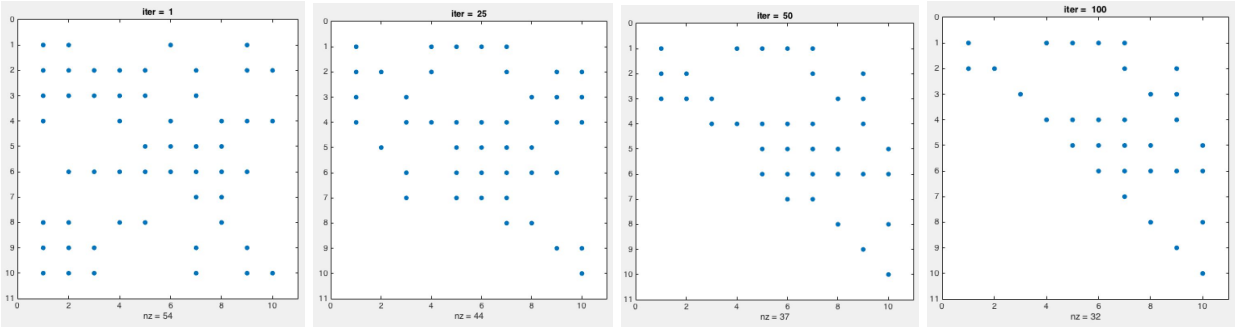
Next, we'll implement a simpler version of the QR method (without shifting) in MATLAB, and observe the sparsity of the resulting matrix. Try the following code in MATLAB:

```
D=diag(1:10); %original matrix with eig = [1,2,...,10]
rand('seed',10);
S=rand(10); %a random matrix
A=S*D*inv(S); %similarity transformation
thresh=1e-6;
numiters=500;
pshow = [1,25,50,100]; %show plots for these iterations
for i=1:numiters
[Q,R]=qr(A);  %Uses MATLAB's QR function
A=R*Q;     %qr step
B=A;     %need this for plotting
```

```
B(B<thresh)=0;
if any(i==pshow)
    figure;
    spy(B);  %plots matrix sparsity
    title(['iter =  ' num2str(i)])
end
end
```

General note: If you copy-paste code from a PDF to MATLAB and get an error, it is most likely due to a symbol such as " - " or "'" not being rendered correctly. Use the MATLAB editor's debugging function to remove these errors.

Compare the eigenvalues of A with that of D. You should get the following plots, where you'll see that A gradually becomes an upper triangular matrix.



Inverse Iteration Method

- Related to Power method [6].
- Jacobi method can give us eigenvectors -Sturm sequence don't
- Inverse iteration gives eigenvectors, and often used alongside Sturm/Bisection method in practice.

Let ν be an approximation of an eigenvalue, and $v^{(0)}$ the corresponding **approximation** to the eigenvector. Then using the inverse iteration method:

$$\begin{aligned}(A-\vartheta I)\boldsymbol{w}^{(k)} &= \boldsymbol{v}^{(k)},\\ \boldsymbol{v}^{(k+1)} &= c_k\boldsymbol{w}^{(k)},\end{aligned}$$

where $c_k = 1/\|w^{(k)}\|_2$, the sequence $\{v^{(k)}\}$ converges to the normalized eigenvector $\overline{v}$ for the eigenvalue λ closest to ν.

[For proof, see Th. 5.10 in book.]

Computing $w^{(k)}$ at every step requires solving a linear system of equations. Here, we can:

- Use LU decomposition of A
- Convert A to tridiagonal T using Householder's (more efficient)

Recap of Eigenvalue methods for Symmetric Real Matrices

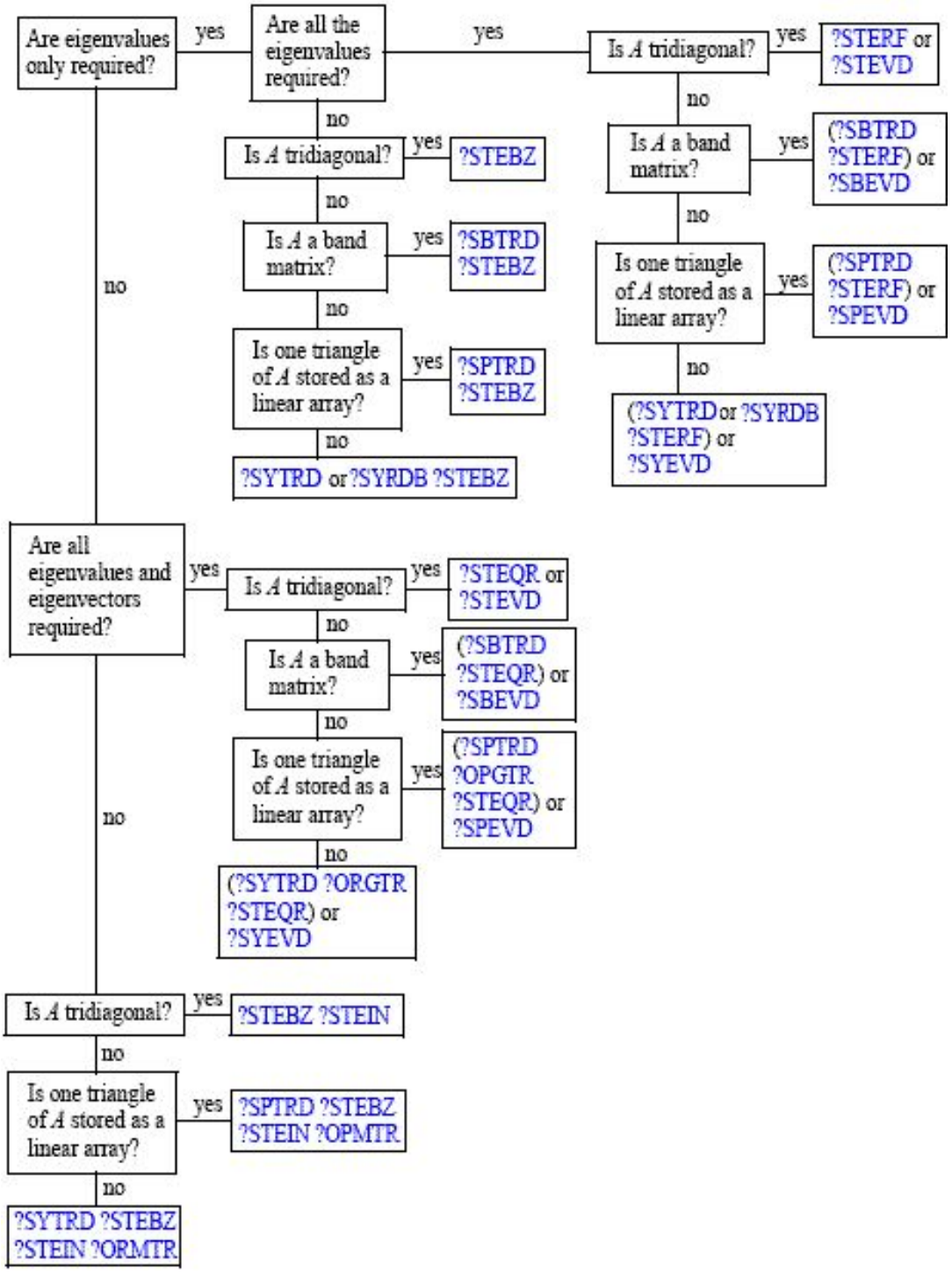
Method	Idea	Algorithm
Jacobi	• Use orthogonal transformations (pre- and post-multiply) to convert matrix to diagonal form.	$R(\varphi) = \begin{pmatrix} \cos\varphi & \sin\varphi \\ -\sin\varphi & \cos\varphi \end{pmatrix}$ $\varphi = \frac{1}{2}\tan^{-1}\frac{2a_{pq}}{a_{qq}-a_{pp}}$
QR	• Take symmetric tridiagonal matrix, and convert to upper-diagonal	• $A_k = Q_k R_k$ • $A_{k+1} = R_k Q_k$ • Can add shift parameter for faster convergence

Bisection method using Sturm's sequence property	• Step 1: Householder method for reducing to tridiagonal form
	• Step 2:
	  ○ # of consecutive sign agreements in sequence $p(\lambda)$ = # of eig. Values > λ
	  • Take interval using Gershgorin theorem, use bisection method to find any eig. value
		$T = \begin{pmatrix} a_1 & b_2 & & & & & & \\ b_2 & a_2 & b_3 & & & & & \\ & b_3 & a_3 & b_4 & & & & \\ & & \cdots & \cdots & \cdots & & & \\ & & & \cdots & \cdots & \cdots & & \\ & & & & \cdots & \cdots & \cdots & \\ & & & & & b_{n-1} & a_{n-1} & b_n \\ & & & & & & b_n & a_n \end{pmatrix}$.
		$p_r(\lambda) = (a_r - \lambda)p_{r-1}(\lambda) - b_r^2 p_{r-2}(\lambda)$,
		$r = 2, 3, \ldots, n$

Inverse Iteration	• Take an estimate of eigenvalue • Iterate to find corresponding eigenvector	$(A - \vartheta I)\boldsymbol{w}^{(k)} = \boldsymbol{v}^{(k)}$, $\boldsymbol{v}^{(k+1)} = c_k \boldsymbol{w}^{(k)}$, where $c_k = 1/\|w^{(k)}\|_2$, the sequence $\{v^{(k)}\}$ converges to the normalized eigenvector $\bar{v}$ for the eigenvalue λ closest to ν.

Solving symmetric eigenvalue problems in LAPACK

LAPACK (**L**inear **A**lgebra **PACK**age) is one of the most popular numerical linear algebra libraries in the world, used for solving systems of linear equations, eigenvalue problems and finding matrix factorizations. It is used inside MATLAB, Numpy (Python) and almost every other scientific computing environment. It is written in Fortran 90, and is highly optimized for efficiency, speed and performance on CPUs. See [13] for a general overview. The following chart and table illustrate how LAPACK decides which algorithm to use for a particular eigenvalue-related operation on a symmetric matrix. The terms in **blue** are the Fortran routines (functions) which run a specific algorithm. For example, **?STEBZ** indicates: **ST** = symmetric triangular, **EBZ** = bisection method. Similarly, **?SBTRD** indicates: **SB** = symmetric banded, **TRD** = reduce to tridiagonal form. You can read about the naming convention in [16].


Are eigenvalues only required?
yes
Are all the eigenvalues required?
yes
Is A tridiagonal?
yes
?STERF or ?STEVD
no
Is A a band matrix?
yes
(?SBTRD ?STERF) or ?SBEVD
no
Is one triangle of A stored as a linear array?
yes
(?SPTRD ?STERF) or ?SPEVD
no
(?SYTRD or ?SYRDB ?STERF) or ?SYEVD
Is A tridiagonal?
yes
?STEBZ
no
Is A a band matrix?
yes
?SBTRD ?STEBZ
no
Is one triangle of A stored as a linear array?
yes
?SPTRD ?STEBZ
no
?SYTRD or ?SYRDB ?STEBZ
no
Are all eigenvalues and eigenvectors required?
yes
Is A tridiagonal?
yes
?STEQR or ?STEVD
no
Is A a band matrix?
yes
(?SBTRD ?STEQR) or ?SBEVD
no
Is one triangle of A stored as a linear array?
yes
(?SPTRD ?OPGTR ?STEQR) or ?SPEVD
no
(?SYTRD ?ORGTR ?STEQR) or ?SYEVD
no
Is A tridiagonal?
yes
?STEBZ ?STEIN
no
Is one triangle of A stored as a linear array?
yes
?SPTRD ?STEBZ ?STEIN ?OPMTR
no
?SYTRD ?STEBZ ?STEIN ?ORMTR

Matrix Type	What is required?	How many?	Kind of matrix	LAPACK routine	Algorithm	Avg. FLOPs
Symmetric	E (values)	All	Tridiagonal	sterf	root-free QR	$14n^2$
			Banded	sbtrd + sterf	Banded to tridiagonal (Givens) + root-free QR	$6n^2k + 14n^2$
			Packed	sptrd+ sterf	Packed to tridiagonal (Householder) + root-free QR	$\frac{4}{3}n^3 + 14n^2$
			Otherwise	sytrd + sterf	To tridiagonal (Householder) + root-free QR	$\frac{4}{3}n^3 + 14n^2$
		Few	Tridiagonal	stebz	Bisection method	$log_2\frac{b-a}{\varepsilon}$
			Banded	sbtrd + stebz	Banded to tridiagonal + bisection	$6n^2k + log_2\frac{b-a}{\varepsilon}$
			Packed	sptrd + stebz	Packed to tridiagonal + bisection	$\frac{4}{3}n^3 + log_2\frac{b-a}{\varepsilon}$
			Otherwise	sytrd + stebz	To tridiagonal + bisection	$\frac{4}{3}n^3 + log_2\frac{b-a}{\varepsilon}$
	E (values) + V (vectors)	All	Tridiagonal	steqr / stevd	QR algorithm / divide-and-conquer	$7n^3$
			Banded	sbtrd + steqr	Banded to tridiagonal + QR	$6n^2k + 7n^3$
			Packed	sptrd + opgtr + steqr	Packed to tridiagonal + generate Q + QR	$\frac{4}{3}n^3 + \frac{4}{3}n^3 + 7n^3$
			Otherwise	sytrd + orgtr + steqr	To tridiagonal + generate Q + QR	$\frac{4}{3}n^3 + \frac{4}{3}n^3 + 7n^3$
		Few	Tridiagonal	stebz + stein	Bisection + inverse iteration	$log_2\frac{b-a}{\varepsilon} + O(n^3)$
			Packed	sptrd + stebz + stein + opmtr	Packed to tridiagonal + bisection + inverse iteration + apply Q	$\frac{4}{3}n^3 + log_2\frac{b-a}{\varepsilon} + O(n^3) + 2m^2n$
			Otherwise	sytrd + stebz + stein + ormtr	To tridiagonal + bisection + inverse iteration + apply Q	$\frac{4}{3}n^3 + log_2\frac{b-a}{\varepsilon} + O(n^3) + 2m^2n$

You'll notice in the table that for the (E+V, All, Tridiagonal) case, **STEQR** can be replaced by **STEVD**. The latter algorithm is known as the *divide-and-conquer* algorithm, and uses recursion to divide the eigenvalue problem into smaller eigenvalue problems, and then 'conquering' or solving them. I'll briefly discuss this in class, but you can read more about it in [10, 11, 12].

Also, note that LAPACK only deals with dense and banded matrices. For sparse matrices, we'll have to use another library, such as ARPACK [14]. However, if you're using MATLAB, you don't need to worry about these issues, since it takes care of everything behind the scenes! You'll find an exhaustive list of the most popular linear algebra libraries in [14].

Note: This section on LAPACK is to help you understand how MATLAB solves (symmetric) eigenvalue problems, and give you a flavor of the practical/computational aspects of numerical analysis. You do not need to know how LAPACK works for the exam (unless you want to!).

Helpful links

1. Sturm's theorem
2. Proof of Sturm's sequence property
3. Using Sturm's theorem for finding eigenvalues (sec. 4.6.2)
4. QR algorithm explained quite well
5. QR algorithm
6. Power method for find largest eigenvalue
7. LAPACK: symmetric eigenvalue problems
8. LAPACK decision tree for symmetric matrices
9. LAPACK decision tree for nonsymmetric matrices
10. Wiki on divide-and-conquer algorithm
11. Extended discussion on QR, root-free QR and divide-conquer
12. Quick intro to divide-conquer from LAPACK website
13. Wiki on LAPACK
14. Overview of numerical linear algebra libraries
15. Presentation on QR, Sturm sequence
16. LAPACK naming scheme