EE447: Assignment 5
Linear Algebra I

Harishankar Ramachandran
EE Dept, IIT Madras

November 26, 2015

1 Reading Assignment

Chapter 11, on eigensystems.

2 The given matrices to diagonalize.

1 $\langle * \text{ 1}\rangle\equiv$ 4▷

```
from matplotlib.pylab import *


x=linspace(0,pi,9)
x=x[0:-1]
A=c_[ones_like(x),cos(x),cos(2*x),cos(3*x),cos(4*x), \
     cos(5*x),cos(6*x),cos(7*x)]
B=array([(15, -3, -5, -7),(-3.0, 25, -4, -6), \
         (-5.0, -4, 10, -5),(-7.0, -6, -5, 20)])
c=(1+1j)/sqrt(2.0)
d=(1-1j)/sqrt(2.0)
C=array([(1, 1, 1, 1),(1, 1, -1j, d),(1, 1j, -1, -1j), \
         (1, c, 1j, 1)])
```

The matrices look like this:

$$A = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \cos(\pi/8) & \cos(\pi/4) & \dots & \cos(7\pi/8) \\ 1 & \cos(\pi/4) & \cos(\pi/2) & \dots & \cos(7\pi/4) \\ \dots & \dots & \dots & \dots & \dots \\ 1 & \cos(7\pi/8) & \cos(7\pi/4) & \dots & \cos(\pi/8) \end{pmatrix}$$

$$B = \begin{pmatrix} 15 & -3 & -5 & -7 \\ -3 & 25 & -4 & -6 \\ -5 & -4 & 10 & -5 \\ -7 & -6 & -5 & 20 \end{pmatrix}$$

$$C = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -j & e^{-j\pi/4} \\ 1 & j & -1 & -j \\ 1 & e^{j\pi/4} & j & 1 \end{pmatrix}$$

These matrices are symmetric (the third matrix is Hermitian) and we know that their eigenvalues and eigenvectors exist.

1. Use python to extract the eigen vectors and eigen values.

3 Reducing the matrices to tridiagonal form

We define the householder matrix of order k (original matrix of order n by n). The matrix is defined by

$$P = \begin{bmatrix} 1 & \dots & 0 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & 1 & 0 & \dots & 0 \\ 0 & \dots & 0 & & & \\ \dots & \dots & \dots & & P_{n-k} & \\ 0 & \dots & 0 & & & \end{bmatrix}$$

1. Define a function that will create $P_{n-k}(v)$, where v is the vector we want to nullify.

2 $\langle Pk\ 2\rangle \equiv$

```
def Pk(n,k,v):

  w=c_[v];x=w.transpose()

  M=eye(n)-2*dot(w,x)/dot(x,w)

 u=householder(v);

  M=(eye()-2*u*u');

endfunction
```

2. Now define the $P(k,v)$ matrix.

3 $\langle P\ 3\rangle\equiv$

```
function M=P(n,k,A)

  M=eye(n,n);

  M(k+1:n,k+1:n)=Pk(n,k+1,A(k+1:n,k))

endfunction
```

3. Apply the transformation on matrix A till it is reduced to tri-diagonal.
4. Apply the transformation to matrix B till it is reduced to tri-diagonal
5. Repeat for matrix C. You will have problems since the code above is for real matrices. Change the code and make it work for Hermitian matrices.

4 Eigenvalues

The method is to do the QR transformation and iterate, i.e.,

- $M_i = Q_i R_i$, where M_i is the i^{th} iteration of the matrix, Q_i is a unitary matrix and R_i is an upper triangular matrix.
- $M_{i+1} = R_i Q_i$ finds the matrix for the next iteration.

This works for all matrices.

1. Apply the algorithm to the original matrix B and see how many iterations are required to bring the off-diagonal elements to 10^{10} less than the diagonal elements. Note that each such iteration costs $\mathcal{O}\left(n^3\right)$ operations.
2. Repeat for the tridiagonal matrix, for which the QR operation costs only $\mathcal{O}(n)$ operations.

   You find that the same number of iterations are required, but the cost is far lower if we could first make the matrix tridiagonal.
3. Now repeat for matrices A and C

   Note that the "error" in the iterations of A do not reduce as they do for B. The reason can be seen by running `spec` on the matrix. You find five degenerate eigenvalues. The algorithm converges to block-diagonal in the case of degenerate matrices.
4. Print the final matrices A and A_{new} after zeroing small variables.

   Note that the block corresponding to the 5 degenerate λ values is different for A and for A_{new}. There is nothing sacred about the block. Any rotation leaves the eigenvalues unchanged.

5 Comments

- Note that we have not done pivoting. This would be essential to keep the accuracy of the computations.
- The method works for any square, symmetric matrix. Such matrices have real eigenvalues, since if
$$Ax = \lambda x$$
it follows that
$$x^H A^H = \lambda^* x^H$$
But $A = A^H$ since A is real and symmetric. Since left and right eigenvalues are the same for a symmetric matrix, it follows that $\lambda = \lambda^*$ is real.
- What the method shows is that any symmetric matrix can be reduced by symmetric transformations to a diagonal matrix. This is so, since if
$$A = QR$$
then
$$R = Q^T A$$
and
$$A_{\text{new}} = RQ = Q^T AQ$$
is a symmetric transformation of A. Thus we have a surefire method to extract eigenvalues and eigenvectors for a real symmetric matrix. The same approach also works for Hermitian matrices, which is very important for SVD.
- The convergence is very slow. It took 100 iterations to converge for a 4 by 4 matrix. This is because we did not scale the matrices to improve convergence. NR claims that we might need 4 to 5 iterations for the first eigenvalue and much less for later ones. Over all an average of 1.3 to 1.6 iterations per eigenvalue are required. Which says that we should have required about 6 iterations instead of 100! Ofcourse this result is only claimed for large n. But still the tuning that NR talks about it is very important.

4 $\langle * \ 1 \rangle + \equiv$ $\triangleleft 1$