Lecture 16

Numerical Methods for Eigenvalues

As mentioned above, the eigenvalues and eigenvectors of an $n \times n$ matrix where $n \geq 4$ must be found numerically instead of by hand. The numerical methods that are used in practice depend on the geometric meaning of eigenvalues and eigenvectors which is equation (14.1). The essence of all these methods is captured in the Power method, which we now introduce.

The Power Method

In the command window of MATLAB enter

```
≫ A = hilb(5)
≫ x = ones(5,1)
≫ x = A*x
≫ el = max(x)
≫ x = x/el
```

Compare the new value of `x` with the original. Repeat the last three lines (you can use the scroll up button). Compare the newest value of `x` with the previous one and the original. Notice that there is less change between the second two. Repeat the last three commands over and over until the values stop changing. You have completed what is known as the *Power Method*. Now try the command

```
≫ [v e] = eig(A)
```

The last entry in `e` should be the final `el` we computed. The last column in `v` is `x/norm(x)`. Below we explain why our commands gave this eigenvalue and eigenvector.

For illustration consider a 2×2 matrix whose eigenvalues are $1/3$ and -2 and whose corresponding eigenvectors are $\mathbf{v}_1$ and $\mathbf{v}_2$. Let $\mathbf{x}_0$ be any vector which is a combination of $\mathbf{v}_1$ and $\mathbf{v}_2$, e.g.,

$$\mathbf{x}_0 = \mathbf{v}_1 + \mathbf{v}_2.$$

Now let $\mathbf{x}_1$ be A times $\mathbf{x}_0$. It follows from (14.1) that

$$\begin{aligned} \mathbf{x}_1 &= A\mathbf{v}_1 + A\mathbf{v}_2 \\ &= \frac{1}{3}\mathbf{v}_1 + -2\mathbf{v}_2. \end{aligned} \tag{16.1}$$

Thus the $\mathbf{v}_1$ part is shrunk while the $\mathbf{v}_2$ is stretched. If we repeat this process k times then

$$\begin{aligned} \mathbf{x}_k &= A\mathbf{x}_{k-1} \\ &= A^k\mathbf{x}_0 \\ &= \left(\frac{1}{3}\right)^k \mathbf{v}_1 + (-2)^k\mathbf{v}_2. \end{aligned} \tag{16.2}$$

Clearly, $\mathbf{x}_k$ grows in the direction of $\mathbf{v}_2$ and shrinks in the direction of $\mathbf{v}_1$. This is the principle of the Power Method, vectors multiplied by A are stretched most in the direction of the eigenvector whose eigenvalue has the largest absolute value.

The eigenvalue with the largest absolute value is called the *dominant* eigenvalue. In many applications this quantity will necessarily be positive for physical reasons. When this is the case, the MATLAB code above will work since `max(v)` will be the element with the largest absolute value. In applications where the dominant eigenvalue may be negative, the program must use flow control to determine the correct number.

Summarizing the Power Method:

- Repeatedly multiply $\mathbf{x}$ by A and divide by the element with the largest absolute value.
- The element of largest absolute value converges to largest absolute eigenvalue.
- The vector converges to the corresponding eigenvector.

Note that this logic only works when the eigenvalue largest in magnitude is real. If the matrix and starting vector are real then the power method can never give a result with an imaginary part. Eigenvalues with imaginary part mean the matrix has a rotational component, so the eigenvector would not settle down either.

Try

```
≫ A = randn(15,15);
≫ e = eig(A)
```

You can see that for a random square matrix, many of the eigenvalues are complex.

However, matrices in applications are not just random. They have structure, and this can lead to real eigenvalues as seen in the next section.

Symmetric, Positive-Definite Matrices

As noted in the previous paragraph, the power method can fail if A has complex eigenvalues. One class of matrices that appear often in applications and for which the eigenvalues are always real are called the symmetric matrices. A matrix is *symmetric* if

$$A' = A,$$

i.e. A is symmetric with respect to reflections about its diagonal.

Try

```
>> A = rand(5,5)
>> C = A'*A
>> e = eig(C)
```

You can see that the eigenvalues of these symmetric matrices are real.

Next we consider an even more specialized class for which the eigenvalues are not only real, but positive. A symmetric matrix is called *positive definite* if for all vectors $\mathbf{v} \neq \mathbf{0}$ the following holds:

$$A\mathbf{v} \cdot \mathbf{v} > 0.$$

Geometrically, A does not rotate any vector by more than $\pi/2$. In summary:

- If A is symmetric then its eigenvalues are real.
- If A is symmetric positive definite, then its eigenvalues are positive numbers.

Notice that the B matrices in the previous section were symmetric and the eigenvalues were all real. Notice that the Hilbert and Pascal matrices are symmetric.

The residual of an approximate eigenvector-eigenvalue pair

If $\mathbf{v}$ and λ are an eigenvector-eigenvalue pair for A, then they are supposed to satisfy the equations: $A\mathbf{v} = \lambda\mathbf{v}$. Thus a scalar residual for approximate $\mathbf{v}$ and λ would be

$$r = \|A\mathbf{v} - \lambda\mathbf{v}\|.$$

The Inverse Power Method

In the application of vibration analysis, the mode (eigenvector) with the lowest frequency (eigenvalue) is the most dangerous for the machine or structure. The Power Method gives us instead the largest eigenvalue, which is the least important frequency. In this section we introduce a method, the *Inverse Power Method* which produces exactly what is needed.

The following facts are at the heart of the Inverse Power Method:

- If λ is an eigenvalue of A then $1/\lambda$ is an eigenvalue for A^{-1}.
- The eigenvectors for A and A^{-1} are the same.

Thus if we apply the Power Method to A^{-1} we will obtain the largest absolute eigenvalue of A^{-1}, which is exactly the reciprocal of the smallest absolute eigenvalue of A. We will also obtain the corresponding eigenvector, which is an eigenvector for both A^{-1} and A. Recall that in the application of vibration mode analysis, the smallest eigenvalue and its eigenvector correspond exactly to the frequency and mode that we are most interested in, i.e. the one that can do the most damage.

Here as always, we do not really want to calculate the inverse of A directly if we can help it. Fortunately, multiplying $\mathbf{x}_i$ by A^{-1} to get $\mathbf{x}_{i+1}$ is equivalent to solving the system $A\mathbf{x}_{i+1} = \mathbf{x}_i$, which can be done

efficiently and accurately. Since iterating this process involves solving a linear system with the same A but many different right hand sides, it is a perfect time to use the LU decomposition to save computations. The following function program does n steps of the Inverse Power Method.

```
function [x e] = myipm(A,n)
    % Performs the inverse power method.
    % Inputs: A -- a square matrix,   n --  number of iterations.
    % Outputs: x -- estimated eigenvector,  e -- estimated smallest eigenvalue.
    [L U P] = lu(A);  % LU decomposition of A with pivoting
    m = size(A,1); % determine the size of A
    x = ones(m,1); % make an initial vector with ones
    for i = 1:n
        px = P*x;  % Apply pivot
        y = L\px;  % solve via LU
        x = U\y;
        % find the maximum entry in absolute value, retaining its sign
        M = max(x);
        m = min(x);
        if abs(M) >= abs(m)
           el = M;
        else
           el = m;
        end
        x = x/el; % divide by the estimated eigenvalue of the inverse of A
    end
    e = 1/el; % reciprocate to get an eigenvalue of A
end
```

Create a 5×5 Hilbert matrix H and use the program to find its smallest eigenvalue and corresponding. Also do `[V, E] = eig(H)` and compare.

Exercises

16.1 For each of the matrices

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \quad \text{and} \quad B = \begin{bmatrix} -2 & 1 & 0 \\ 1 & -2 & 1 \\ 0 & 1 & -3 \end{bmatrix},$$

perform two iterations of the power method by hand starting with a vector of all ones. State the resulting approximations of the eigenvalue and eigenvector.

16.2 (a) Write a well-commented MATLAB **function** program `mypm.m` that inputs a matrix and a tolerance, applies the power method until the scalar residual is less than the tolerance, and outputs the estimated eigenvalue and eigenvector, the number of steps, and the scalar residual.

(b) Test your program on the matrices A and B in the previous exercise and check that you get the same results as your hand calculations for the first 2 iterations. Turn in your program only.