Text reference page 370.

Iterative Methods for Eigenvalues

Purpose
To demonstrate some simple iterative methods for finding eigenvalues and eigenvectors.

MATLAB Functions
`max, abs, eig, inv`

In MATLAB, the *power method* can be represented by first multiplying a vector v by A

```
v = A*v
```

Next, find the entry of `v` that is largest in absolute value:

```
[s,k] = max(abs(v))
```

Keep track of that entry:

```
mu = v(k)
```

Scale by `mu`

```
v = v/mu
```

Then repeat. We can code 10 passes of this iteration with a simple loop.

```
for i = 1:10
    v = A*v,
    [s,k] = max(abs(v));
    mu = v(k);
    v = v/mu;
end
```

The text tells us that the sequence of values given by `mu` will converge to an eigenvalue, when there is a strictly dominating eigenvalue. Since $\mathbf{v}$ is an approximate eigenvector we can find another way to estimate the eigenvalue. The discussion before Exercise 11 on page 369 of the text shows that the least-squares solution to $A\mathbf{v} = \lambda\mathbf{v}$ (note that the variable in this equation is λ) is given by the *Rayleigh quotient,*

$$\lambda = \frac{\mathbf{v}^T A\mathbf{v}}{\mathbf{v}^T\mathbf{v}}$$

Using the Rayleigh Quotient provides better accuracy than using `mu` as shown above. Thus we can modify our code:

```
for i = 1:10
    v = A*v,
    s = max(abs(v));
    v = v/s;
end
r = (v'*(A*v))/(v'*v)
```

The *inverse power method* is described in Section 5.8 of the text. This method attempts to estimate an eigenvalue near a parameter s by running the power method on the inverse of the shifted matrix $A - sI_n$. While this method naturally tends toward an eigenvector of $(A - sI_n)^{-1}$, that vector is also an eigenvector of A. Thus, in MATLAB, we can write a few iterations of the loop and conclude with the Rayleigh quotient *applied to the original matrix A.*

```
B = inv(A - s*eye(n));
for i = 1:10
    v = B*v;
    s = max(abs(v));
    v = v/s;
end
r = (v'*(A*v))/(v'*v)
```

We have computed the inverse of $A - sI_n$, since it is convenient in MATLAB. In practice for large problems, a more efficient means is to compute an LU factorization of $A - sI_n$ and use it in the loop to solve $(A - sI_n)\mathbf{x} = \mathbf{b}$ rather than multiplying to find $\mathbf{x} = (A - sI_n)^{-1}\mathbf{b}$.

You may be wondering about how to choose the shift s. If you have a reasonable guess about an eigenvalue, then using that guess as the shift makes sense. Frequently the diagonal entries of the matrix A make good shifts. Another approach is to choose

$$s = \frac{\mathbf{x}^T A \mathbf{x}}{\mathbf{x}^T \mathbf{x}}$$

the Rayleigh quotient, if you have a reasonable guess for an eigenvector.

MATLAB Exercises

1. Use the power method with the Rayleigh quotient to find an eigenvalue for each of the following matrices. Then check for accuracy by comparing the result from MATLAB's `eig`.

a. `ones(5)`
b. `pascal(5)`
c. `hilb(5)`
d. `A` formed with `v = rand(5,1); A = eye(5) - (2/(v'*v))*(v*v')`
e. Describe what happened in part d, and explain why.

2. Use the inverse power method to find the smallest (in absolute value) eigenvalue of each matrix:

a. `pascal(5)`
b. `hilb(5)`
c. `magic(5)`
d. Describe what happened in part c, and explain why.

3. Use the inverse shift method to find all of the eigenvalues of `A = hilb(4)` by carefully choosing the shift s. First, try using the diagonal entries of `A` for the shift s. If you do not succeed in getting all of the eigenvalues this way, try using `eig(A)` as a guide for choosing s.

4. The matrix formed by

```
A = magic(7); A = A/175;
```

is a *stochastic matrix,* that is, the sum of each column is 1. Check this with

```
sum(A)
```

Use the power method to find the largest eigenvalue of A. What is the eigenvector?

5. The Laydata function `randstoc` will make a random stochastic matrix:

```
A = randstoc(5)
```

Check that this matrix is stochastic using

```
sum(A)
```

Use the inverse power method to find all the eigenvalues of this matrix.

6. The discussion above suggested using the Rayleigh quotient

$$\frac{\mathbf{x}^T A\mathbf{x}}{\mathbf{x}^T\mathbf{x}}$$

to approximate the eigenvalue. The other approach is to use the value of `mu`, which we can obtain with

```
v = A*v
[s,k] = max(abs(v))
mu = v(k)
```

Try the following:

```
A = hilb(5); e = max(eig(A));
```

Run the power method for 5 iterations and compare the errors from using the Rayleigh quotient and `mu`. Now run 10 iterations. Which method converges faster?