Tutorial 5: The power method

You learned in class that you can compute the eigenvalues of a matrix **A** by multiplying an arbitrary vector repeatedly from the left by **A**. In this Tutorial you will learn how to implement this method in Matlab and how to extend it to compute more than one eigenvalue of a matrix.

The power method to find the dominant eigenvalue.

We apply the power method to the matrix

```
A = [2 -1  0
    -1  2 -1
     0 -1  2];
```

The method starts by selecting an arbitrary vector $\mathbf{x}_0$ whose largest entry is 1. You can generate an aribitrary vector using the randn command. You can learn how to use the randn command via the help function:

```
help randn
```

We need an arbitrary vector of length 3 (because **A** is 3 x 3)

```
xo = randn(3,1)
```

Next, we need to make sure that the largest entry of $\mathbf{x}_0$ is equal to one. One way of doing this is to find the maximum entry of $\mathbf{x}_0$ and divide each element of $\mathbf{x}_0$ by this maximum entry:

```
max_xo = max(abs(xo)); % find maximum entry in xo
xo = xo/max_xo;
xo
```

We can now start the power method which requries that we obtain $\mathbf{x}_{k+1}$ from $\mathbf{x}_k$ in two steps:

1. Multiply $\mathbf{x}_k$ by **A** and define μ_k to be the largest entry of $\mathbf{A}\mathbf{x}_k$.
2. Set $\mathbf{x}_{k+1} = (1/\mu_k)\mathbf{A}\mathbf{x}_k$

This can be implemented with the following code:

```
x = xo; % initialize the method via xo
for kk = 1:100 % do 100 iterations
    x = A*x;
    m = max(abs(x));
    x = x*(1/m);
end
fprintf('Largest eigenvalue computed by the power method %g\n',m)
```

You can check the result by using the eig command:

```
L = eig(A);
fprintf('Largest eigenvalue computed by eig %g\n',max(L))
```

You can modify the code to save the value of μ_k and $\mathbf{x}_k$ during the iteration.

```
MuSave = zeros(20,1);
EvecSave = zeros(3,20);
x = xo; % initialize the method via xo
for kk = 1:20 % do 20 iterations
    x = A*x;
    m = max(abs(x));
    x = x*(1/m);
    EvecSave(:,kk) = x;
    MuSave(kk) = m;
end
```

You can then plot the results and see how quickly μ_k approaches the largest eigenvalue of **A**

```
figure
plot([1 20],L(end)*[1 1],'LineWidth',2)
hold on, plot(MuSave,'-','LineWidth',2)
xlabel('Iteration number')
ylabel('Largest eigenvalue of A')
legend('eig',...
    'Power method')
```

After only 6 iterations, you already have a pretty accurate approximation of the largest eigenvalue of **A**.

You can also see how quickly the elements of $\mathbf{x}_k$ approach the elements of the eigenvector corresponding to the largest eigenvector of **A**

```
figure
plot(EvecSave(1,:),'LineWidth',2)
hold on, plot(EvecSave(2,:),'LineWidth',2)
hold on, plot(EvecSave(3,:),'LineWidth',2)
xlabel('Iteration number')
ylabel('Entries of the eigenvector')
```

Computing another eigenvalue of A.

The power method is useful for finding the largest eigenvalue of a matrix. But what if you are also interested in the second and third largest eigenvalue of **A**?

The idea is based in the following two facts:

1. If $\lambda \neq 0$ is an eigenvalye of an invertible matrix $\mathbf{A}$, then $1/\lambda$ is an eigenvalue of $\mathbf{A}^{-1}$
2. If λ is an eigenvalye of $\mathbf{A}$, then $\lambda - \alpha$ is an eigenvalue of $\mathbf{A} - \alpha\mathbf{I}$

Equipped with these two facts, it is easy to show that:

If λ is an eigenvalye of $\mathbf{A}$, then $1/(\lambda - \alpha)$ is an eigenvalue of $(\mathbf{A} - \alpha\mathbf{I})^{-1}$.

The proof is as follows. By Fact 2, it is clear that If λ is an eigenvalye of $\mathbf{A}$, then $(\lambda - \alpha)$ is an eigenvalue of $(\mathbf{A} - \alpha\mathbf{I})$. Combining this with Fact 1, proofs the claim.

The idea behind the "inverse power method" is to find eigenvalues of the matrix

$$\mathbf{B} = (\mathbf{A} - \alpha\mathbf{I})^{-1}$$

which, by the reasoning above are

$$\frac{1}{\lambda_1 - \alpha}, \frac{1}{\lambda_2 - \alpha}, \ldots, \frac{1}{\lambda_n - \alpha},$$

where $\lambda_1, \lambda_2, \ldots, \lambda_n$ are the eigenvalues of $\mathbf{A}$.Thus, if we set α to be approximately equal to λ_i, then the largest eigenvalue of $\mathbf{B}$ is

$$\mu_i = \frac{1}{\lambda_i - \alpha},$$

and the power method, applied to **B**, will find this eigenvalue. Given μ_i, we can then compute λ_i via:

$$\lambda_i = \alpha + \frac{1}{\mu_i}.$$

You can implement the inverse power method in Matlab as follows.

You decide on an alpha:

```
a = 1.5;
```

Then you apply the power method to the inverse of

```
B = (A-a*eye(3));
```

To search for eigenvalues near a.

```
MuSave = zeros(20,1);
EvecSave = zeros(3,20);
x = xo; % initialize the method via xo
for kk = 1:20 % do 20 iterations
    x = B\x;
    m = max(abs(x));
```

```
    x = x*(1/m);
    EvecSave(:,kk) = x;
    MuSave(kk) = a+1/m;
end
figure
plot([1 20],L(end-1)*[1 1],'LineWidth',2)
hold on, plot(MuSave,'-','LineWidth',2)
xlabel('Iteration number')
ylabel('Second eigenvalue of A')
legend('eig','Power method')
```

You can see that the method works quite well for this example. We already found the first two eigenvalues which are, approximately, $\lambda_1 \approx 3.41$, $\lambda_2 \approx 2$. Assuming that the third eigenvalues is even smaller we try the inverse power method with a = 0.1.

```
a = .1;

B = (A-a*eye(3));
MuSave = zeros(20,1);
EvecSave = zeros(3,20);
x = xo; % initialize the method via xo
for kk = 1:20 % do 20 iterations
    x = B\x;
    m = max(abs(x));
    x = x*(1/m);
    EvecSave(:,kk) = x;
    MuSave(kk) = a+1/m;
end
figure
plot([1 20],L(1)*[1 1],'LineWidth',2)
hold on, plot(MuSave,'-','LineWidth',2)
xlabel('Iteration number')
ylabel('Third eigenvalue of A')
legend('eig','Power method')
```

Exercises

1. Proof the following statements:

    1. If $\lambda \neq 0$ is an eigenvalye of an invertible matrix $\mathbf{A}$, then $1/\lambda$ is an eigenvalue of $\mathbf{A}^{-1}$
    2. If λ is an eigenvalye of $\mathbf{A}$, then $\lambda - \alpha$ is an eigenvalue of $\mathbf{A} - \alpha\mathbf{I}$

2. Use the power method and inverse power method to compute the two largest eigenvalues of the 100 x 100 matrix.

```
clear % clear old variables
n = 100;
A = 2*eye(n,n)-diag(ones(n-1,1),1)-diag(ones(n-1,1),-1);
```

Check your answer by using Matlab's eig command.