Power method

En la práctica, los valores propios de las matrices extensas no se calculan usando el polinomio característico. Calcular el polinomio resulta muy costoso, y extraer las raíces exactas de un polinomio de grado alto puede ser difícil de calcular y expresar. En consecuencia, los algoritmos generales para encontrar vectores propios y valores propios son iterativos. La manera más fácil es el método de las potencias: se escoge un vector aleatorio y se calcula una secuencia de vectores unitarios. Esta sucesión casi siempre convergerá a un vector propio correspondiente al mayor valor propio.

Multiplying by a matrix Arepeatedly will exponentially amplify the largest $|\lambda|$ eigenvalue. This is the basis for many algorithms to compute eigenvectors and eigenvalues, the most basic of which is known as the power method.

The simplest version of this is to just start with a random vector x and multiply it by A repeatedly. This works, but has the practical problem that the vector quickly becomes very large or very small, and eventually becomes too big/small for the computer to represent (this is known as "oveflow/underflow"). The fix is easy: normalize the vector after each multiplication by A. That is:

Starting with a random x, repeatedly compute $x = \frac{\mathrm{Ax}}{\|\mathrm{Ax}\|}$

If x was the exact eigenvector, we could just multiply Ax and see how much each component increased: they would all increase (or decrease) by the same factor. But, for an approximate eigenvector, each component of Ax will increase by a slightly different amount. Instead, the most common approach is to use the Rayleigh quotient. If we have an exact eigenvector, so that $\mathrm{Ax} = \lambda x$, then the Rayleigh quotient will gives us exactly λ. Otherwise, it is a kind of weighted-average (weighted by the components x_i^2) and is a reasonable approximation

$\lambda = \frac{x \cdot \mathrm{Ax}}{x \cdot x}$

```
A = gallery(3);
[V,D] = eig(A)
```

```
V = 3×3
    0.3162   -0.4041   -0.1391
   -0.9487    0.9091    0.9740
   -0.0000    0.1010   -0.1789
D = 3×3
    1.0000         0         0
         0    2.0000         0
         0         0    3.0000
```

```
spectralRadius = max(abs(diag(D)))
```

```
spectralRadius = 3.0000
```

```
[x,lambda] = powerMethod(A)
```

```
x = 3×1
   -0.1391
    0.9740
```

```
   -0.1789
lambda = 3.0000
```

Largest or dominant eigenvalue λ_1(spectral radius). Strictly greater (in absolute value) than the other eigenvalues.

The method can converge very slowly if $\left|\frac{\lambda_2}{\lambda_1}\right|$ is close to 1. And if the two eigenvalues have equal magnitude, the method may not converge at all.

The smallest eigenvalue can be determined by applying the power method to the matrix inverse of A. For this case, the power method will converge on the largest value of $\frac{1}{\lambda}$ - in other words, the smallest value of λ.

```
function [x, lambda] = powerMethod(A)  % power iteration
    REL_TOL = eps;
    MAX_ITER = 50;
    Ax = A * rand(1, length(A))';
    x = Ax / norm(Ax);

    i = 0;
    flag = true;
    while flag
        xp = x;
        Ax = A * x;
        x = Ax / norm(Ax);
        i = i + 1;
        flag = i < MAX_ITER && norm((x - xp) / x) > REL_TOL;
    end

    lambda = dot(x, Ax) / dot(x, x);
end
```