Householder transformation

wiki: https://en.wikipedia.org/wiki/Householder_transformation

Definition: Given two vectors start from the origin $\vec{v}_a$ and $\vec{v}_b$, find a linear transformation that reflect one vector $\vec{v}_a\left(\vec{v}_b\right)$ into the direction of the other $\vec{v}_b\left(\vec{v}_a\right)$

steps:

1. normalize two vectors: $\hat{v}_a = \frac{\vec{v}_a}{\left|\vec{v}_a\right|}, \hat{v}_b = \frac{\vec{v}_b}{\left|\vec{v}_b\right|}$
2. differnece of two unit vectors: $\vec{u} = \hat{v}_a - \hat{v}_b$
3. normalize: $\hat{u} = \frac{\vec{u}}{\left|\vec{u}\right|}$
4. linear transformation: $T = I - 2\hat{u}\hat{u}^T$

could verify: $T\vec{v}_a = s\vec{v}_b, T\vec{v}_b = t\vec{v}_a$, here s, t are some scalar constants.

NOTICE: the examples in lecture notes are just special case of here where $\vec{v}_b = [1,0,\cdots,0]^T$

property:

1. symmetry: $T = T^T$
2. unitary: $T^T = T^{-1}$, or $TT^T = T^TT = I$

such property guarantee that such linear transformation applying on matrix TAT will not change eigenvalues of the origin matrix.

```
vec1 = rand(3,1);
vec2 = rand(3,1);
[T,~] = householder_matrix(vec1,vec2);
disp(T)
```

```
   -0.6932    0.1127    0.7119
    0.1127    0.9925   -0.0474
    0.7119   -0.0474    0.7007
```

```
disp([T*vec1,vec2,T*vec1./vec2])
```

```
    0.1302    0.1112    1.1712
    0.9139    0.7803    1.1712
    0.4565    0.3897    1.1712
```

```
disp([T*vec2,vec1,T*vec2./vec1])
```

```
    0.2883    0.3377    0.8538
    0.7685    0.9001    0.8538
    0.3153    0.3692    0.8538
```

```
function [mat,vec] = householder_matrix(source_vec, target_vec)
% find Householder transformation that transform source_vec (direction) to target_vec (directio
% reference: https://en.wikipedia.org/wiki/Householder_transformation#Transformation
% sepcial case: target_case==[1,0,0,...]
%   see lecture note and wiki. Actually, exactly the same as below
% source_vec/target_vec(N1,1)
% mat(N1,N1)
% vec(N1,1)

N1 = size(source_vec,1);
assert(size(target_vec,1)==N1);

source_vec = source_vec / sqrt(sum(source_vec.^2,1));
target_vec = target_vec / sqrt(sum(target_vec.^2,1));
vec = source_vec - target_vec;
vec = vec/sqrt(sum(vec.^2,1));
mat = eye(N1) - 2*(vec*vec.');
end
```