Householder transformation - Tridiagonalization

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

Definition: Given two vectors start from the origin $\vec{v}_a$ and $\vec{v}_b$, we could 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: $T = I - 2\hat{u}\hat{u}^T$

you 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)
```

```
    0.9962   -0.0361    0.0789
   -0.0361    0.6549    0.7548
    0.0789    0.7548   -0.6512
```

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

```
    0.0254    0.0971    0.2612
    0.2151    0.8235    0.2612
    0.1815    0.6948    0.2612
```

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

```
    0.1219    0.0318    3.8288
    1.0603    0.2769    3.8288
    0.1768    0.0462    3.8288
```

Tridiagonalization

Given symmetry matrix A_1, we construct

$$S_i = \begin{bmatrix} I_{i\times i} & \\ & T_{\vec{v}_a \to \vec{v}_b} \end{bmatrix}$$

where $I_{i\times i}$ is i by i unit matrix, $\vec{v}_a = A_i(i+1:end, i)$, $\vec{v}_b = [1,0,\cdots,0]^T$ and $T_{\vec{v}_a \to \vec{v}_b}$ is the Householder reflection transformation. and apply

$A_{i+1} = S_i A_i S_i$

$N-1$ times, we will have tridiagonal matrix

```
N0 = 5;
tmp1 = rand(N0,N0);
matA = tmp1 + tmp1';
disp(tridiagonal_householder(matA))
```

```
    0.6342    2.2638   -0.0000   -0.0000   -0.0000
    2.2638    3.5346    1.2073    0.0000   -0.0000
   -0.0000    1.2073    0.3397    0.8790   -0.0000
   -0.0000    0.0000    0.8790   -0.3758   -0.0541
   -0.0000   -0.0000    0.0000   -0.0541    1.1081
```

Sturm Sequence

```
% TO BE ADDED
```

functions

```
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


function [retA,vec_reflector,retB] = tridiagonal_householder(matA)
% transform matA into tri-diagonal form
% reference: https://en.wikipedia.org/wiki/Householder_transformation#Tridiagonalization
% matA(N1,N1): symmetric
% (ret1)retA: tri-diagonal matrix
% (ret2)vec_reflector: reflector vector used in transformation
% (ret3)retB: matA = retB*retA*retB', retB * retB' = I
% doc hess
assert(issymmetric(matA), "matrix for eig_jocabi should be symmetric");
N1 = size(matA,1);

vec_reflector = zeros(N1,N1-2);
retB = eye(N1);
for ind1 = 2:(N1-1)
    vec1 = matA(ind1:end,ind1-1);
    vec2 = zeros(size(vec1));
    vec2(1) = 1;
    [mat,vec_reflector(ind1:end,ind1-1)] = householder_matrix(vec1,vec2);
    T = blkdiag(eye(ind1-1), mat);
    retB = retB * T;
    matA = T * matA * T;
end
retA = matA;
end
```