```
format long e;
```

Q1

```
fprintf('Q1');
```

```
Q1
```

```
sizes = [5 4; 6 6; 8 7; 10 5];
for size=sizes'
    m = size(1);
    n = size(2);
    A = randn(size(1), size(2));
    [W, R] = polard1(A);
    if (m >= n)
        norm(W*R - A)
        norm(W'*W - eye(n))
    else
        norm(R*W - A)
        norm(W*W' - eye(m))
    end
    norm(R - R')
    max(abs(imag(eig(R))))
    min(real(eig(R)))
end
```

```
ans =
     3.842797079566486e-15
ans =
     2.298893533756662e-15
ans =
     1.255187498907224e-16
ans =
     0
ans =
     5.034054597380387e-01
ans =
     3.053436636578927e-15
ans =
     1.166415263071733e-15
ans =
     1.931110615645319e-16
ans =
     0
ans =
     8.178423215205403e-02
ans =
     5.934658694179852e-15
ans =
     1.537404709290552e-15
ans =
     3.433362460314130e-16
ans =
     0
ans =
     1.697350083467998e-01
ans =
```

```
     5.527099008464580e-15
ans =
     1.355359494449504e-15
ans =
     2.355138688025663e-16
ans =
     0
ans =
     1.896935563864549e+00
```

It is observed that for all the randomly-generated matrices, the errors are of the order of machine epsilon. Also, min(eig(R)) is non-negative.

Q2

```
fprintf('Q2');
```

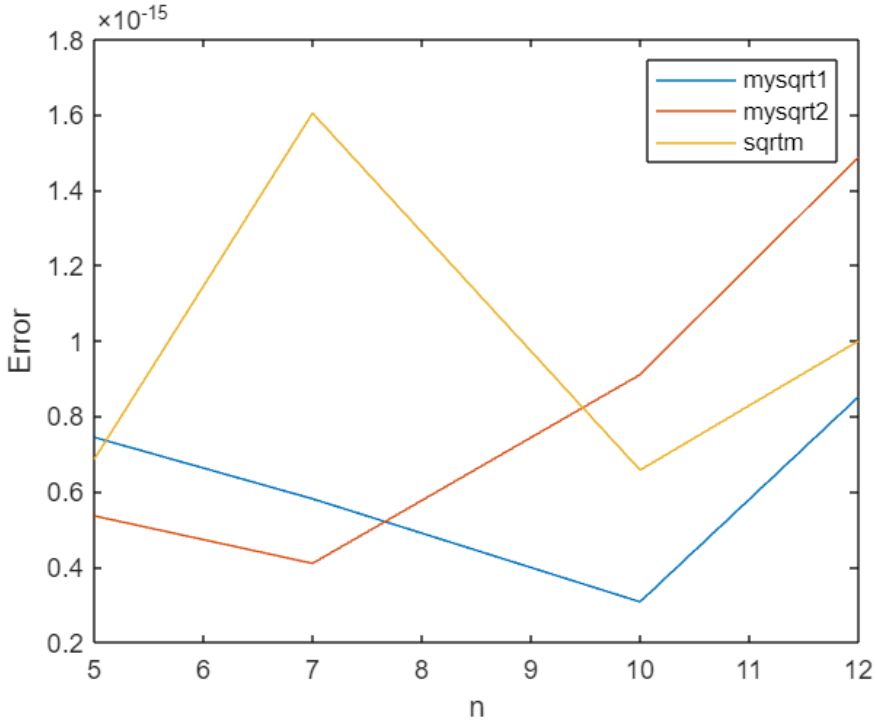
```
Q2
```

```
ns = [5, 7, 10, 12];

err1 = zeros(1, 4);
err2 = zeros(1, 4);
err3 = zeros(1, 4);
for i = 1:4
    n = ns(i);
    A = hilb(n);
    R1 = mysqrt1(A);
    R2 = mysqrt2(A);
    R3 = sqrtm(A);

    err1(i) = norm(A - (R1 * R1)) / norm(A);
    err2(i) = norm(A - (R2 * R2)) / norm(A);
    err3(i) = norm(A - (R3 * R3)) / norm(A);
end

plot(ns, err1);
hold on;
plot(ns, err2);
plot(ns, err3);
hold off;
xlabel('n');
ylabel('Error');
legend('mysqrt1', 'mysqrt2', 'sqrtm');
```




```
ns
```

```
ns = 1×4
     5     7    10    12
```

```
err1
```

```
err1 = 1×4
     7.436555054602137e-16     5.803992369242151e-16     3.070236471627202e-16 ···
```

```
err2
```

```
err2 = 1×4
     5.352003755622419e-16     4.090232051659431e-16     9.098181294552489e-16 ···
```

```
err3
```

```
err3 = 1×4
     6.851539234360993e-16     1.603702483437992e-15     6.566531941887708e-16 ···
```

For all given values of n, mysqrt1 gives the least error.

error ≈ 10^{-16} for all case

Q3

```
fprintf('Q3');
```

```
Q3
```

```
err1 = zeros(1, 15);
err2 = zeros(1, 15);
```

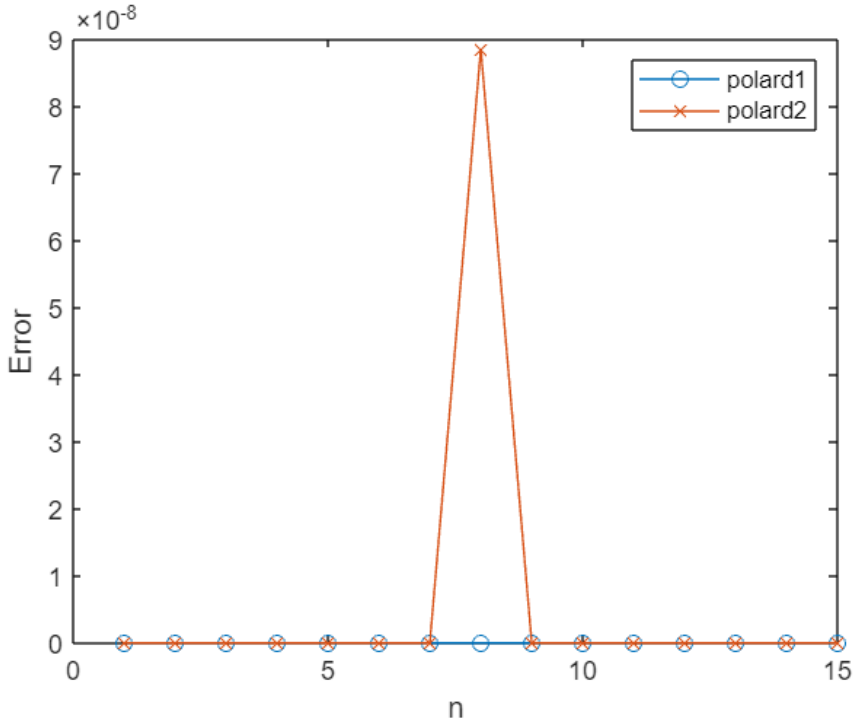
```
for i = 1:15
    A = randn(20, 20);
    [U, S, V] = svd(A);
    S = sort(diag(S), 'descend');
    S = (S / S(end)) * (10^(-i + 6));
    S = diag(S);
    A = U * S * V';

    [W, R] = polard1(A);
    [X, T] = polard2(A);

    err1(i) = norm(W' * W - eye(20));
    err2(i) = norm(X' * X - eye(20));
end

plot(1:15, err1, '-o');
hold on;
plot(1:15, err2, '-x');
hold off;

xlabel('n');
ylabel('Error');
legend('polard1', 'polard2');
```



As it can be clearly seen from the graphs, polard1 is must better than polard2.

Q4

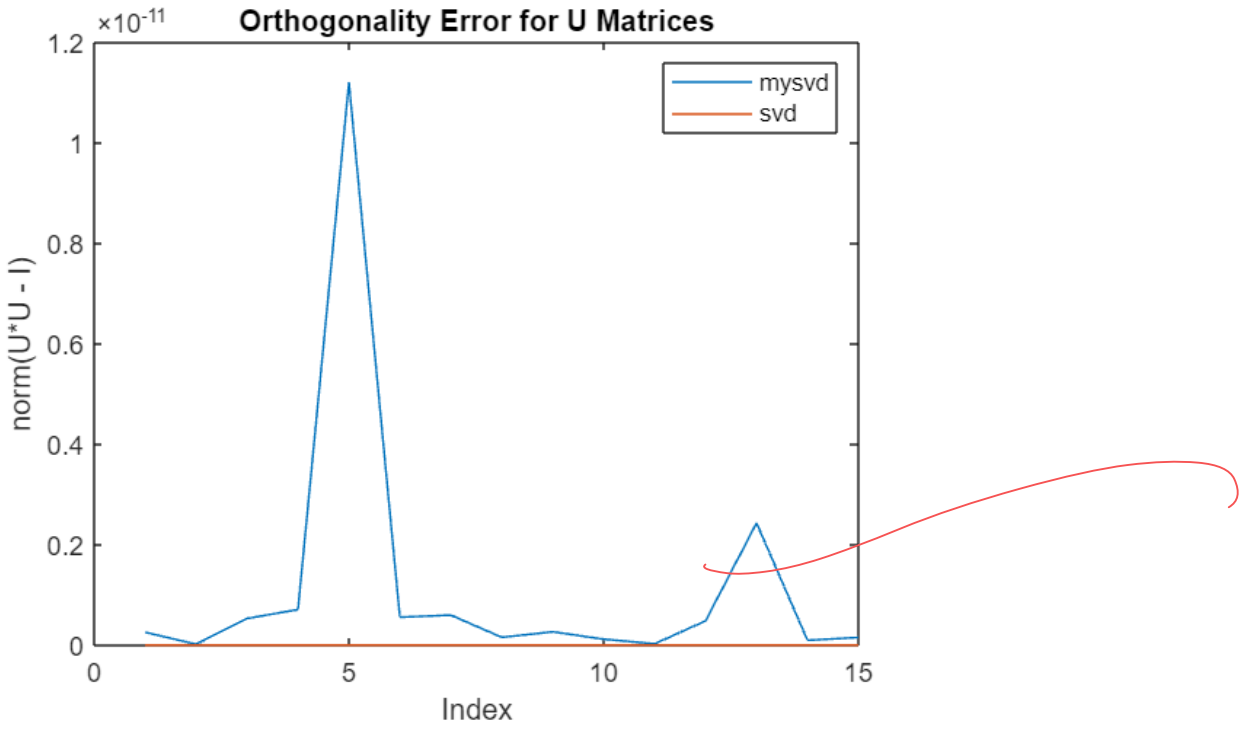
```
fprintf('Q4');
```

```
Q4
```

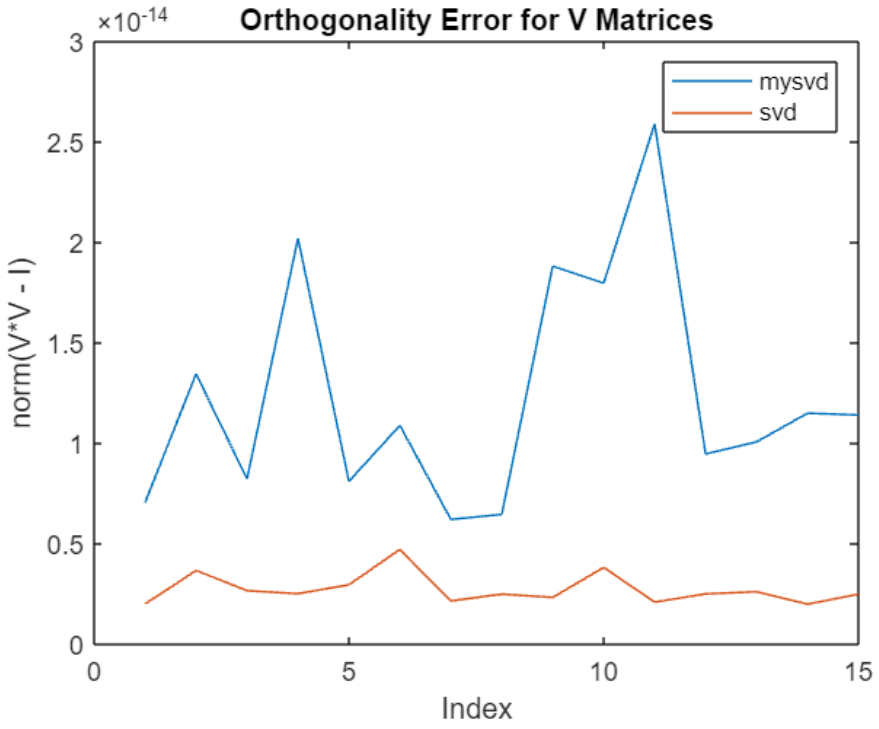
```
errU = zeros(15, 2);
errV = zeros(15, 2);
for i = 1:15
    A = randn(20, 20);
    [U, S, V] = svd(A);
    S = sort(diag(S), 'descend');
    S = (S / S(end)) * (10^(-i + 6));
    S = diag(S);
    A = U * S * V';

    [my_U, ~, my_V] = mysvd(A);
    [U, ~, V] = svd(A);

    errU(i, :) = [norm(my_U' * my_U - eye(20)), norm(U' * U - eye(20))];
    errV(i, :) = [norm(my_V' * my_V - eye(20)), norm(V' * V - eye(20))];
end
plot_orthogonality(1:15, errU, 'U');
```




```
plot_orthogonality(1:15, errV, 'V');
```



The builtin svd function gives much lesser error than the mysvd function

```
function [W, R] = polard1(A)
    m = size(A, 1);
    n = size(A, 2);
    if m >= n
        [U, S, V] = svd(A, 0);
        R = V * S * V';
        W = U * V';
    else
        [U, S, V] = svd(A, "econ");
        R = U * S * U';
        W = U * V';
    end
end

function R1 = mysqrt1(A)
    [X, D] = eig(A);
    R1 = (X * sqrt(D)) / X;
end

function R2 = mysqrt2(A)
    R = chol(A);
    [~, S, V] = svd(R);
    R2 = V * S * V';
end

function [W, R] = polard2(A)
```

```
    R = mysqrt1(A * A');
    W = R \ A;
end

function [U, S, V] = mysvd(A)
    [W, R] = polard2(A);

    [V, S] = eig(R, 'vector');
    [S, I] = sort(S, 'descend');
    S = diag(S);

    V = V(:, I);
    U = W * V;
end

function plot_orthogonality(x, data, label)
    figure;
    plot(x, data(:, 1));
    hold on;
    plot(x, data(:, 2));
    hold off;

    xlabel('Index');
    ylabel(['norm(' label '*' label ' - I)']);
    title(['Orthogonality Error for ' label ' Matrices']);
    legend('mysvd', 'svd');
end
```

check: norm(A - U*S*V') ≈ 0(u)