

**Problem [1].** For  $PQ$ , consider one eigenvalue  $\lambda_i$  and its corresponding eigenvector  $v_i$ :

$$PQv_i = \lambda_i v_i \Rightarrow \lambda_i = v_i^T PQv_i = v_i^T P v_i v_i^T Q v_i$$

$$P > 0 \Rightarrow v_i^T P v_i > 0$$

$$Q > 0 \Rightarrow v_i^T Q v_i > 0$$

Therefore:  $\lambda_i > 0$ .

Now, we will show that the given matrices have the same eigenvalues.

Apply the similarity transformations  $T_1 = P^{-\frac{1}{2}}$  and  $T_2 = Q^{\frac{1}{2}}$  to  $PQ$ :

$$T_1(PQ)T_1^{-1} = P^{-\frac{1}{2}}(PQ)P^{\frac{1}{2}} = P^{\frac{1}{2}}QP^{\frac{1}{2}}$$

$$T_2(PQ)T_2^{-1} = Q^{\frac{1}{2}}(PQ)Q^{-\frac{1}{2}} = Q^{\frac{1}{2}}PQ^{\frac{1}{2}}$$

Eigenvalues are preserved under similarity transformations  $\Rightarrow \lambda(PQ) = \lambda(P^{\frac{1}{2}}QP^{\frac{1}{2}}) = Q^{\frac{1}{2}}PQ^{\frac{1}{2}}$ .

The generalized eigenvalues of  $(P, Q^{-1})$ :

$$Pv_i = \lambda_i Q^{-1}v_i \Rightarrow (PQ - \lambda_i I)Q^{-1}v_i = (PQ - \lambda_i I)w_i$$

$$\Rightarrow \lambda(P, Q^{-1}) = \lambda(PQ).$$

Let  $PQ$  have the following eigenvalue decomposition:  $PQ = V\Lambda V^{-1}$ . We can relate the eigenvectors of the matrices in the following way:

$$P^{\frac{1}{2}}QP^{\frac{1}{2}} = (P^{-\frac{1}{2}}V)\Lambda(P^{-\frac{1}{2}}V)^{-1}$$

$$Q^{\frac{1}{2}}PQ^{\frac{1}{2}} = (Q^{\frac{1}{2}}V)\Lambda(Q^{\frac{1}{2}}V)^{-1}$$

$$QP = (QV)\Lambda(QV)^{-1}$$

```
clear all, clc, close all, format compact
```

```
P = [ 1    -1    0;
      -1   2    2;
        0   2    6];
```

```
Q = [ 7    6   -2;
      6    6   -2;
      -2   -2    1];
```

```
EVals1 = eig(P*Q)
EVals2 = eig(sqrtm(P)*Q*sqrtm(P))
EVals3 = eig(sqrtm(Q)*P*sqrtm(Q))
EVals4 = eig(P,inv(Q))
```

**Output:**

```
EVals1 =
     2
     1
     2
```

```

EVals2 =
    1.0000
    2.0000
    2.0000
EVals3 =
    1.0000
    2.0000
    2.0000
EVals4 =
    1.0000
    2.0000
    2.0000

```

**Problem [2].** The following MATLAB code provides the solution for this problem:

---

```

clear all, clc, close all, format short e

A = [ -0.746  0.006  -0.999  0.0369;
      -12.9   -0.746  0.387   0;
       4.31   0.024  -0.174  0;
       0      1      0      0];

B = [ 0.0012  0.0092;
      6.05    0.952;
      -0.416 -1.76;
      0      0];

C = [ 0  0  1  0;
      0  0  0  1];

%
%   a)
%

U = lyapchol(A,B);
P = U'*U;
[V1,D1] = eig(P);

% the state transformation that diagonalizes P
% (T1*P*T1 = [e1 e2 e3 e4]*D1*[e1 e2 e3 e4]')
T1 = V1';

% check
T1*P*T1'

% transform to the new basis and truncate by keeping the columns of V1
% corresponding to the 2 LARGEST eigenvalues of P

A1 = T1(3:4,:)*A*T1(3:4,:); % i.e. A1 = T1*A*inv(T1); A1(3:4,3:4)

```

```

B1 = T1(3:4,:)*B;
C1 = C*T1(3:4,:)' ;

%
%      b)
%

L = lyapchol(A',C');
Q = L'*L;
[V2,D2] = eig(Q);

% the state transformation that diagonalizes Q
% (T2*Q*T2 = [e1 e2 e3 e4]*D2*[e1 e2 e3 e4]')
T2 = V2';

% check
diagQ = inv(T2')*Q*inv(T2) % this also reveals the degree of observability

% transform to the new basis and truncate by keeping the columns of V2
% corresponding to the 2 LARGEST eigenvalues of Q

A2 = T2(3:4,:)*A*T2(3:4,:)'
B2 = T2(3:4,:)*B;
C2 = C*T2(3:4,:)' ;

%
%      c)
%

U = U';
L = L';
[W,S,V] = svd(U'*L);

% keep the 2 states which are easiest to reach and observe

W = W(:,1:2);
V = V(:,1:2);
S = S(1:2,1:2);

T = sqrtm(S)\(V'*L');
Ti = (U*W)/sqrtm(S);

% balance

Ar = T*A*Ti;
Br = T*B;
Cr = C*Ti;

%
%      d)
%
```

```
figure,clf
bode(ss(A,B,C,0),'r',ss(A1,B1,C1,0),'b--')
title('Problem 2, point a')
```

```
figure,clf
bode(ss(A,B,C,0),'r',ss(A2,B2,C2,0),'b--')
title('Problem 2, point b')
```

```
figure,clf
bode(ss(A,B,C,0),'r',ss(Ar,Br,Cr,0),'b--')
title('Problem 2, point c')
```

**Remark:** Notice that the gramians  $P$  and  $Q$  were computed using the `lyapchol()` routine. This was not needed for this small example,  $P = \text{lyap}(A, B*B')$  would have worked just as fine. In general, it is useful for larger systems to use `lyapchol()`, because the gramians will be positive definite by construction.

**Remark:** Also, in point c), we have truncated first and then balanced. This is a subtle point. Theoretically, truncating first and then balancing is the same thing as first balancing the system and then truncating. However, computationally it is more efficient to truncate first and then balance (the operation count is lower).

**Problem [3].** The following MATLAB code provides the solution for this problem:

```
clear all, clc, close all, format short e

%
%      (1)
%

[A,B,C,D] = butter(6,1,'s');

k = 4;

U = lyapchol(A,B);
L = lyapchol(A',C');

P = U'*U;

U = U';
L = L';
[W,S,V] = svd(U'*L);

% truncate

W = W(:,1:k);
V = V(:,1:k);
S = S(1:k,1:k);

T = sqrtm(S)\(V'*L');
Ti = (U*W)/sqrtm(S);
```

```

% balance

Ar = T*A*Ti;
Br = T*B;
Cr = C*Ti;

%
%      (2)
%

T = P^(-.5);
Ti = P^(.5);

% check
norm(T*P*T' - eye(6))

A3 = T*A*Ti;
B3 = T*B;
C3 = C*Ti;

% truncate

A3red = A3(1:k,1:k);
B3red = B3(1:k,:);
C3red = C3(:,1:k);

%
%      (3)
%

figure,clf
bode(ss(A,B,C,0),'r',ss(Ar,Br,Cr,0),'b--',ss(A3red,B3red,C3red,0),'g-.')
title('Problem 3, point (3)')

```

---

**Problem [4].** The following MATLAB code provides the solution for this problem:

---

```

clear all, clc, close all, format short

%
%      (a)
%

A1 = [0 1; -1 -2]; B1 = [0; 1]; C1 = [0 1];
A2 = [-1 2; 0 -1]; B2 = [-1; 1]; C2 = [-.5 .5];
A3 = [-1 0; 0 -1]; B3 = [-1; 1]; C3 = [-.5 .5];
A4 = [-.25 -1; 1 -.5]; B4 = [1; -1]; C4 = [1 1];

% compute the gramians and the Hankel Singular Values
P1 = lyap(A1,B1*B1');

```

```

Q1 = lyap(A1',C1'*C1);
hsv1 = sqrt(eig(P1*Q1))

P2 = lyap(A2,B2*B2');
Q2 = lyap(A2',C2'*C2);
hsv2 = sqrt(eig(P2*Q2))

P3 = lyap(A3,B3*B3');
Q3 = lyap(A3',C3'*C3);
hsv3 = sqrt(eig(P3*Q3))

P4 = lyap(A4,B4*B4');
Q4 = lyap(A4',C4'*C4);
hsv4 = sqrt(eig((P4*Q4)))

% We notice that only systems 1 and 2 have the same HSV therefore we can find a
% (similarity) transformation T1

% solving the system
%      T1*A1*inv(T1) = A2
%      T1*B1 = B2
%      C1*inv(T1) = C2
% we get the following T1

T1 = [1 -1; 1 1];

% check
T1*A1*inv(T1) - A2

%
%      (b)
%

k = 1;

U = lyapchol(A1,B1);
L = lyapchol(A1',C1');

U = U';
L = L';
[W,S,V] = svd(U'*L);

% truncate

W = W(:,1:k);
V = V(:,1:k);
S = S(1:k,1:k);

T = sqrtm(S)\(V'*L');
Ti = (U*W)/sqrtm(S);

% balance

```

```
Ar = T*A1*Ti;
Br = T*B1;
Cr = C1*Ti;
```

---

**Output:**

```
hsv1 =
    0.2500
    0.2500
hsv2 =
    0.2500
    0.2500
hsv3 =
     0
    0.5000
hsv4 =
    2.0000
    1.0000
ans =
     0     0
     0     0
```

**Problem [5].** The following MATLAB code provides the solution for this problem:

---

```
clear all, clc, close all, format short

K = [2 1; 1 2];
L = [3 1; 1 3];

[X,EVals] = eig(K,L);
EVals

% First, if we would like to diagonalize K*L, we would apply T1

[V,D] = eig(K*L);
T1 = V';

M = T1*K*L*T1' % = T1*K*T1'*T1*L*T1'

% We notice that we have obtained a diagonal matrix, but M ~ = I.
% Now, we need to rescale the final matrix M s.t. M == I. This can
% be achieved by using the following T

T = (D^-0.25)*V'; % notice that now T'*T ~ = I

% check
M = T*K*T'*T*L*T'
```

---

**Output:**

```
EVals =  
    0.5000         0  
         0    0.7500  
M =  
    2.0000         0  
         0   12.0000  
M =  
    1.0000         0  
         0    1.0000
```

**Problem [6].** The following MATLAB code provides the solution for this problem:

---

```
clear all, clc, close all, format short e  
  
A4 = [-.25 -1; 1 -.5]; B4 = [1; -1]; C4 = [1 1];  
  
%  
%      (a)  
%  
  
U = lyapchol(A4,B4);  
P = U'*U  
  
L = lyapchol(A4',C4');  
Q = L'*L  
  
U = U';  
L = L';  
[W,S,V] = svd(U'*L);  
  
T = sqrtm(S)\(V'*L');  
Ti = (U*W)/sqrtm(S);  
  
Abal = T*A4*Ti;  
Bbal = T*B4;  
Cbal = C4*Ti;  
  
%  
%      (b)  
%  
  
T = Q^(.5);  
Ti = Q^(-.5);  
  
% check
```

```
Ti'*Q*Ti - eye(2)
```

```
A44 = T*A4*Ti;
```

```
B44 = T*B4;
```

```
C44 = C4*Ti;
```

```
% truncate
```

```
A4red = A44(1,1);
```

```
B4red = B44(1,:);
```

```
C4red = C44(:,1);
```

---

### Output:

```
P =
```

```
2.0000e+00    0
    0    1.0000e+00
```

```
Q =
```

```
2.0000e+00 -6.2804e-16
-6.2804e-16    1.0000e+00
```

```
ans =
```

```
2.2204e-16 -3.4513e-31
-3.2047e-31    0
```