ECE 717
Homework 3: Realization theory

due: Wednesday October 23, 2019

1. **Jordan normal form.** Any matrix $A \in \mathbb{R}^{n\times n}$ can be transformed (using a similarity transform) into *Jordan normal form*, which is a block-diagonal matrix that looks like:

$$T^{-1}AT = \begin{bmatrix} J_1 & & 0 \\ & \ddots & \\ 0 & & J_r \end{bmatrix} \qquad \text{with} \quad J_i = \begin{bmatrix} \lambda_i & 1 & \cdots & 0 \\ \vdots & \lambda_i & \ddots & \vdots \\ 0 & \vdots & \ddots & 1 \\ 0 & 0 & \cdots & \lambda_i \end{bmatrix}.$$

Each "Jordan block" J_i is constant along the diagonal and this diagonal entry is an eigenvalue of A. The diagonal immediately above the main diagonal is all 1's. Jordan blocks can only exist when eigenvalues are repeated, and the size of each Jordan block depends on the geometric multiplicity of the corresponding eigenvalue. In the special case where A happens to be diagonalizable, each Jordan block is 1×1 and $T^{-1}AT$ is a diagonal matrix. So we recover the standard eigenvalue decomposition in this case.

Suppose that a SISO LTI system (A, B, C, D) has an A-matrix that is a single Jordan block:

$$\left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right] = \left[\begin{array}{ccccc|c} \lambda & 1 & 0 & \cdots & 0 & b_1 \\ 0 & \lambda & 1 & \cdots & 0 & b_2 \\ \vdots & \vdots & \lambda & \ddots & \vdots & \vdots \\ 0 & 0 & \vdots & \ddots & 1 & b_{n-1} \\ 0 & 0 & 0 & \cdots & \lambda & b_n \\ \hline c_1 & c_2 & c_3 & \cdots & c_n & d \end{array}\right]$$

a) Show that (A, B) is controllable if and only if $b_n \neq 0$.

b) Show that (C, A) is observable if and only if $c_1 \neq 0$.

2. **Realizing a transfer function.** For the transfer function $H(s) = \frac{s+1}{s^2+2}$, find:

a) an uncontrollable and observable realization,

b) a controllable and unobservable realization,

c) an uncontrollable and unobservable realization,

d) a controllable and observable (minimal) realization.

3. **Kalman canonical form.** In this problem, you will write a MATLAB function that transforms a system into Kalman canonical form. We will do it in three parts. The following MATLAB commands might come in handy as you write your code:

- `P = ctrb(A,B)` returns the controllability matrix.
- `Q = obsv(A,C)` returns the observability matrix.
- `T = orth(M)` returns a matrix whose columns are a basis for range(M).
- `T = null(M)` returns a matrix whose columns are a basis for null(M).

a) Given two subspaces of $\mathbb{R}^n$ with bases given by the columns of T_1 and T_2 respectively, the MATLAB function below returns a matrix with columns that form a basis for the intersection of the two subspaces. Explain how it works.

```
function T = intersect_subspaces(T1,T2)
    [n,r] = size(T1);
    S = null( [T1 T2] );
    T = T1 * S(1:r,:);
end
```

b) Given two subspaces of $\mathbb{R}^n$ with bases given by the columns of T_1 and T_2 respectively, with range(T_1) $\subseteq$ range(T_2), the MATLAB function below returns a matrix T with columns that complete the basis. That is, range$\left(\begin{bmatrix} T_1 & T \end{bmatrix}\right)$ = range(T_2). Explain how it works.

```
function T = complete_basis(T1,T2)
    T1bar = null(T1');
    T = intersect_subspaces(T2,T1bar);
end
```

c) Write a MATLAB function `[T1,T2,T3,T4] = kcf(A,B,C)` that takes as input the matrices for a state-space realization and returns the blocks of the matrix T that transforms the system into to Kalman canonical form. Use the convention from class that $T = \begin{bmatrix} T_{c\bar{o}} & T_{co} & T_{\bar{c}\bar{o}} & T_{\bar{c}o} \end{bmatrix}$.

d) Write a MATLAB function `[Am,Bm,Cm] = reduce(A,B,C)` that returns a minimal realization via the Kalman canonical form. To test your function, find a minimal realization for

$$\left[\begin{array}{cccc|c} 5 & 9 & 2 & 1 & 2 \\ -5 & -6 & -1 & 0 & -1 \\ -7 & -13 & -5 & -2 & -3 \\ 14 & 5 & 2 & -4 & 4 \\ \hline 7 & 16 & 3 & 3 & 0 \end{array}\right]$$

Compare your result to the MATLAB command `minreal(ss(A,B,C,0))`. You can see if the realizations are the same by comparing their transfer function computed using `tf(...)`.