State-Space Identification, Noiseless Data

5.1: Introduction to state-space models

- There are two main problems with performing system ID on transfer-function models:
  - Nonlinear optimization is required: Global optima not guaranteed, model order is not obvious, optimization can take a long time.
  - Multi-input multi-output (MIMO) systems are not easy to identify (well) using these methods.
- An alternative to transfer functions for modeling system dynamics is to use a "state space" form.
  - Prior knowledge of this form is not required (but is helpful).
  - We will spend some time discussing the most important features of state-space models from the point of view of system ID.
  - State-space models turn out to be extremely valuable for optimal estimator and control design as well, so the benefits of the form extend beyond a "simple" system ID (cf. ECE5520, 5530, 5550. . .).
- State-space system-ID methods find global optima under the assumed conditions, and work well for both SISO and MIMO systems.
- This chapter focuses on system-ID methods for noise-free systems.
  - We introduce state-space models and required related material;
  - We then look at three ID methods: Ho–Kalman, ERA, subspace.

- Next chapter will focus on state-space system ID methods for systems influenced by disturbance and sensor noise.

A quick introduction to state-space models

- Transfer functions provide a system's input–output mapping only:

$$u[k] \rightarrow G(z) \rightarrow y[k].$$

- State-space models provide access to what is going on *inside* the system, in addition to the input–output mapping.
  - What's going on inside the system is called the system's "state".

DEFINITION: The internal state of a system at time k_0 is the minimum amount of information at k_0 that, together with the input $u[k]$, $k \geq k_0$, uniquely determines the behavior of the system for all $k \geq k_0$.

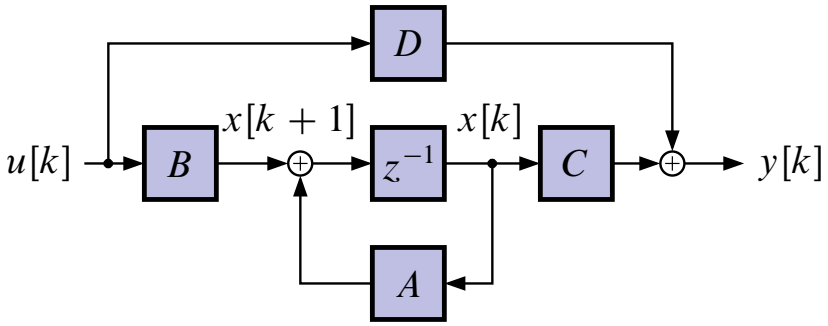
- State-space models describe a system's dynamics via two equations:
  - The "state equation" describes how the input influences the state;
  - The "output equation" describes how the state and the input both directly influence the output.
- Because the state summarizes the impact of all past inputs, we don't need to store past inputs to predict future outputs.
  - Contrast with unit-pulse-response (convolution) representation which requires all past history of $u[k]$, $y[k] = \sum_{m=0}^{\infty} g[m]u[k-m]$.
- Discrete-time LTI state-space models have the following form:

$$x[k+1] = Ax[k] + Bu[k]$$

$$y[k] = Cx[k] + Du[k],$$

where $u[k] \in \mathbb{R}^m$ is the input, $y[k] \in \mathbb{R}^p$ is the output, and $x[k] \in \mathbb{R}^n$ is the state vector.

- Different systems have different n, A, B, C, and D.
- A block diagram can help visualize the signal flows:



EXAMPLE: Convert the following single-input single-output difference equation into a discrete-time state-space form,

$$y[k]+a_1y[k-1]+a_2y[k-2]+a_3y[k-3] = b_1u[k-1]+b_2u[k-2]+b_3u[k-3].$$

- We're going to do the conversion by first recognizing that the transfer function of this system is,

$$G(z) = \frac{b_1z^2 + b_2z + b_3}{z^3 + a_1z^2 + a_2z + a_3} = \frac{Y(z)}{U(z)}.$$

- Break up transfer function into two parts. $G_p(z) = V(z)/U(z)$ contains all of the poles:

$$G_p(z) = \frac{1}{z^3 + a_1z^2 + a_2z + a_3} = \frac{V(z)}{U(z)}$$

$$\Rightarrow \quad v[k+3] + a_1v[k+2] + a_2v[k+1] + a_3v[k] = u[k].$$

- Choose current and advanced versions of $v[k]$ as state (this is a choice: there are other equally valid choices, as we will see)

$$x[k] = \begin{bmatrix} v[k+2] & v[k+1] & v[k] \end{bmatrix}^T.$$

- Then

$$x[k+1] = \begin{bmatrix} v[k+3] \\ v[k+2] \\ v[k+1] \end{bmatrix} = \begin{bmatrix} -a_1 & -a_2 & -a_3 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} v[k+2] \\ v[k+1] \\ v[k] \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} u[k].$$

- We now add zeros, $G(z) = \left(b_1 z^2 + b_2 z + b_3\right) G_p(z)$. Equivalently,

$$Y(z) = \left[b_1 z^2 + b_2 z + b_3\right] V(z),$$

  or, $y[k] = b_1 v[k+2] + b_2 v[k+1] + b_3 v[k]$.

- In summary, we have the state-space model:

$$x[k+1] = \begin{bmatrix} -a_1 & -a_2 & -a_3 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} x[k] + \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} u[k]$$

$$y[k] = \begin{bmatrix} b_1 & b_2 & b_3 \end{bmatrix} x[k] + \begin{bmatrix} 0 \end{bmatrix} u[k].$$

- Note: There are many other equally valid state-space models of this particular transfer function. We will soon see how they are related.
- Many discrete-time transfer functions are not strictly proper. Solve by polynomial long division, and setting D equal to the quotient.
- MATLAB command `[A,B,C,D]=tf2ss(num,den,Ts)` converts a rational-polynomial transfer function form to state-space form.

5.2: Working with state-space systems

State-space to transfer function

- In the prior example, we saw it is possible to convert from a difference equation (or transfer function) to a state-space form quite easily.
- Now, we'll see that the opposite translation is also straightforward.
- Start with the state equations

$$x[k+1] = Ax[k] + Bu[k]$$

$$y[k] = Cx[k] + Du[k].$$

- Take the z-transform of both sides of both equations

$$zX(z) - zx[0] = AX(z) + BU(z)$$

$$Y(z) = CX(z) + DU(z),$$

or

$$(zI - A)X(z) = BU(z) + zx[0]$$

$$X(z) = (zI - A)^{-1}BU(z) + (zI - A)^{-1}zx[0].$$

- This gives,

$$Y(z) = \underbrace{[C(zI - A)^{-1}B + D]}_{\text{transfer function of system}} U(z) + \underbrace{C(zI - A)^{-1}zx[0]}_{\text{response to initial conditions}}.$$

- So,

$$G(z) = \frac{Y(z)}{U(z)} = C(zI - A)^{-1}B + D.$$

- Note that $(zI - A)^{-1} = \dfrac{\text{adj}(zI - A)}{\det(zI - A)}$, so we can write a system's transfer function as

$$G(z) = \frac{C\,\mathrm{adj}(zI - A)B + D\det(zI - A)}{\det(zI - A)}.$$

- Extremely important observation: The poles of the system are where $\det(zI - A) = 0$, which (by definition) are the eigenvalues of A.

Transformation

- State-space representations of a particular system's dynamics are not unique. Selection of state $x[k]$ is somewhat arbitrary.
- To see this, analyze the transformation of

$$x[k+1] = Ax[k] + Bu[k]$$
$$y[k] = Cx[k] + Du[k],$$

  where we let $x[k] = Tw[k]$, where T is an invertible (similarity) transformation matrix. Then,

$$(Tw[k+1]) = A\,(Tw[k]) + Bu[k]$$
$$y[k] = C\,(Tw[k]) + Du[k].$$

- Multiplying the first equation by T^{-1} gives

$$w[k+1] = \underbrace{T^{-1}AT}_{\bar{A}}\,w[k] + \underbrace{T^{-1}B}_{\bar{B}}\,u[k]$$
$$y[k] = \underbrace{CT}_{\bar{C}}\,w[k] + \underbrace{D}_{\bar{D}}\,u[k]$$

so, $w[k+1] = \bar{A}w[k] + \bar{B}u[k]$

$$y[k] = \bar{C}w[k] + \bar{D}u[k].$$

- To show that $H_1(z) = H_2(z)$,

$$\begin{aligned} H_1(z) &= C(zI - A)^{-1}B + D \\ &= CTT^{-1}(zI - A)^{-1}TT^{-1}B + D \\ &= (CT)[T^{-1}(zI - A)T]^{-1}(T^{-1}B) + D \\ &= \bar{C}(zI - \bar{A})^{-1}\bar{B} + \bar{D} = H_2(z). \end{aligned}$$

- Transfer function not changed by similarity transform

CONCLUSION: Can arrive at state-space representations having identical input–output relationship but different (A, B, C, D) matrices.

EXAMPLE: Consider transforming the system

$$A = \begin{bmatrix} -a_1 & -a_2 & -a_3 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad \text{with} \quad T = T^{-1} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}.$$

$$C = \begin{bmatrix} b_1 & b_2 & b_3 \end{bmatrix}, \qquad D = \begin{bmatrix} 0 \end{bmatrix}$$

- Note that multiplying on the right by T flips the original entries left-to-right; multiplying on the left flips the original entries top-to-bottom.
- So, for this transformation matrix, we get:

$$\bar{A} = T^{-1}AT = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} -a_1 & -a_2 & -a_3 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -a_3 & -a_2 & -a_1 \end{bmatrix}$$

$$\bar{B} = T^{-1}B = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$\bar{C} = CT = \begin{bmatrix} b_1 & b_2 & b_3 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} b_3 & b_2 & b_1 \end{bmatrix}$$

$$\bar{D} = D = 0.$$

- We can find the transfer function of this new form as

$$\begin{aligned} G(z) &= \bar{C}(zI - \bar{A})^{-1}\bar{B} + \bar{D} \\ &= \begin{bmatrix} b_3 & b_2 & b_1 \end{bmatrix} \left(\left(\begin{bmatrix} z & 0 & 0 \\ 0 & z & 0 \\ 0 & 0 & z \end{bmatrix} - \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -a_3 & -a_2 & -a_1 \end{bmatrix} \right)^{-1} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} + 0 \\ &= \begin{bmatrix} b_3 & b_2 & b_1 \end{bmatrix} \left(\left(\begin{bmatrix} z & -1 & 0 \\ 0 & z & -1 \\ a_3 & a_2 & z + a_1 \end{bmatrix} \right)^{-1} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right. \\ &= \frac{\begin{bmatrix} b_3 & b_2 & b_1 \end{bmatrix} \begin{bmatrix} z^2 + a_1 z + a_2 & a_1 + z & 1 \\ -a_3 & z^2 + a_1 z & z \\ -a_3 z & -a_2 z - a_3 & z^2 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}}{z^3 + a_1 z^2 + a_2 z + a_3} \\ &= \frac{\begin{bmatrix} b_3 & b_2 & b_1 \end{bmatrix} \begin{bmatrix} 1 \\ z \\ z^2 \end{bmatrix}}{z^3 + a_1 z^2 + a_2 z + a_3} = \frac{b_1 z^2 + b_2 z + b_3}{z^3 + a_1 z^2 + a_2 z + a_3}, \end{aligned}$$

  which was the transfer function we started with before transformation.

5.3: Discrete-time Markov parameters

- It can be proved by induction from the equation

$$x[k+1] = Ax[k] + Bu[k], \quad x[0]$$

that the state $x[k]$ evolves according to the relationship:

$$x[k] = A^k x[0] + \underbrace{\sum_{j=0}^{k-1} A^{k-1-j} Bu[j]}_{\text{convolution}}.$$

- So, because $y[k] = Cx[k] + Du[k]$, we get

$$y[k] = \underbrace{CA^k x[0]}_{\text{initial resp.}} + \underbrace{\sum_{j=0}^{k-1} CA^{k-1-j} Bu[k]}_{\text{convolution}} + \underbrace{Du[k]}_{\text{feedthrough}}.$$

- It turns out that the discrete unit-pulse response of a state-space system has a special form that is important to us later.
- For example, let's look at the unit-pulse response of a single-input state-space system. (Note that, by definition, $x[0] = 0$ when finding a unit-pulse response).
- We find that

$$\begin{aligned}
y[0] &= Cx[0] + Du[0] = D, & x[1] &= Ax[0] + Bu[0] = B \\
y[1] &= Cx[1] + Du[1] = CB, & x[2] &= Ax[1] + Bu[1] = AB \\
y[2] &= Cx[2] + Du[2] = CAB, & x[3] &= Ax[2] + Bu[2] = A^2B \\
&\vdots & &\vdots \\
y[k] &= CA^{k-1}B, & k &\geq 1.
\end{aligned}$$

- These unit-pulse-response values, $\{D, CB, CAB, CA^2B, CA^3B, \ldots\}$ are called the Markov parameters of the system.

- They are very important for some of our system-ID methods.

- Specifically, we define the Markov parameters to be:

$$g_k = \begin{cases} D, & k = 0; \\ CA^{k-1}B, & k > 0. \end{cases}$$

CLARITY ISSUE:

- For SISO systems, the Markov parameters are scalars.
- For a single-input multi-output (SIMO) system the Markov parameters are (column) vectors.
  - The ith entry (row) of each Markov parameter is computed as the unit-pulse response from the input to the ith output.
  - Equivalently, the entire vector Markov parameter is the unit-pulse response from the input to the vector output.
- For multi-input single-output (MISO) systems, the Markov parameters are row vectors.
  - The jth entry (column) of each Markov parameter is computed via the unit-pulse response from the jth input to the output.
- For multi-input multi output (MIMO) systems, the Markov parameters are matrices.
  - The (i, j)th entries yield the the unit-pulse response from the jth input to the ith output.
  - Equivalently, the jth column of each Markov parameter is vector (as in the SIMO case) which is computed via the unit-pulse response from the jth input to the vector output.

EXAMPLE: Given the following discrete-time system, with zero initial condition, find the unit-pulse response:

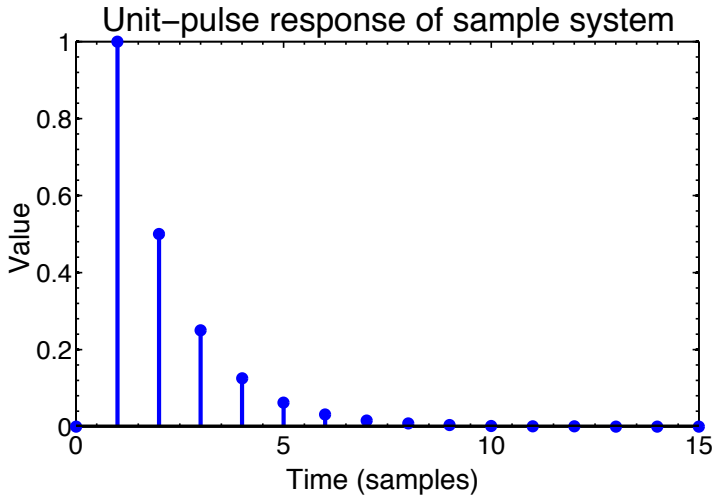
$$A = \begin{bmatrix} 0.5 & 0 \\ 0 & 1 \end{bmatrix}, \qquad B = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \qquad C = \begin{bmatrix} 1 & -1 \end{bmatrix}, \qquad D = 0.$$

- The Markov parameters are given by

$$\begin{aligned} g_k &= \{D, CB, CAB, CA^2B, \ldots\} \\ &= \{0, 1, 0.5, 0.25, \ldots\}. \end{aligned}$$

- MATLAB's `impulse.m` command confirms this result:

```
A = [0.5 0; 0 1];
B = [1 ; 0];
C = [1 -1]; D = 0;
sys = ss(A,B,C,D,-1);
y = impulse(sys,0:15);
stem(0:15,y,'filled');
```



5.4: Discrete-time controllability and observability

- Two related concepts are very important when studying control of state-space systems: controllability and observability.
- These particular *concepts* are not directly important to the problem of system identification, but some of the *mathematical forms* show up in many of the derivations that we will encounter.
  - So, we have a brief look at them, to see what these forms are.

Discrete-time controllability

- Consider the problem of driving a system to some arbitrary state $x[n]$, starting at an arbitrary state $x[0]$. Recall $x[k+1] = Ax[k] + Bu[k]$, so

$$
\begin{aligned}
x[1] &= Ax[0] + Bu[0] \\
x[2] &= A\left[Ax[0] + Bu[0]\right] + Bu[1] \\
x[3] &= A\left[A^2x[0] + ABu[0] + Bu[1]\right] + Bu[2] \\
&\vdots \\
x[n] &= A^n x[0] + \underbrace{\begin{bmatrix} B & AB & A^2B & \cdots & A^{n-1}B \end{bmatrix}}_{\mathcal{C}} \begin{bmatrix} u[n-1] \\ \vdots \\ u[0] \end{bmatrix},
\end{aligned}
$$

  where $\mathcal{C}$ is called the controllability matrix. This leads to

$$
\begin{bmatrix} u[n-1] \\ \vdots \\ u[0] \end{bmatrix} = \mathcal{C}^{-1}\left[x[n] - A^n x[0]\right].
$$

- If $\mathcal{C}$ is not invertible then these control signals don't exist. In that case, the input is only *partially* effective in influencing the state.

- If $\mathcal{C}$ is invertible, input can achieve any *arbitrary* state $x[n]$ for *any* $x[0]$.
  - We say that $\{A, B\}$ form a "controllable pair" or that the system is controllable.
  - Some literature refers to this as reachable, in that any state can be reached from $x[0] = 0$, which is a subset of the problem that we looked at, but turns out to have the same criteria.
  - Turns out that if we cannot reach a state in n time steps from $x[0] = 0$, then we cannot do so with any number of time steps (Caley–Hamilton theorem, studied in ECE5520).

Discrete-Time Observability

- Another question we can ask is, can we reconstruct the state $x[0]$ from the output $y[k]$ and input $u[k]$? Recall, $y[k] = Cx[k] + Du[k]$ so

$$
\begin{aligned}
y[0] &= Cx[0] + Du[0] \\
y[1] &= C\left[Ax[0] + Bu[0]\right] + Du[1] \\
y[2] &= C\left[A^2x[0] + ABu[0] + Bu[1]\right] + Du[2] \\
&\vdots \\
y[n-1] &= C\left[A^{n-1}x[0] + A^{n-2}Bu[0] + \cdots + Bu[n-1]\right] + Du[n-1].
\end{aligned}
$$

- In vector form, we can write

$$
\begin{bmatrix} y[0] \\ y[1] \\ \vdots \\ y[n-1] \end{bmatrix} = \underbrace{\begin{bmatrix} C \\ CA \\ \vdots \\ CA^{n-1} \end{bmatrix}}_{\mathcal{O}} x[0] + \underbrace{\begin{bmatrix} D & 0 & \cdots & 0 \\ CB & D & \cdots & 0 \\ CAB & CB & \cdots & 0 \\ \vdots & \vdots & \ddots & D \end{bmatrix}}_{\Psi} \begin{bmatrix} u[0] \\ u[1] \\ \vdots \\ u[n-1] \end{bmatrix},
$$

  where $\mathcal{O}$ is the observability matrix.

- We re-arrange this expression to solve for $x[0]$

$$x[0] = \mathcal{O}^{-1}\left[\begin{bmatrix} y[0] \\ \vdots \\ y[n-1] \end{bmatrix} - \Psi \begin{bmatrix} u[0] \\ \vdots \\ u[n-1] \end{bmatrix}\right].$$

- If $\mathcal{O}$ is full-rank or nonsingular, $x[0]$ may be reconstructed with any $y[k]$, $u[k]$. We say that $\{C, A\}$ form an "observable pair."
- Do more measurements of $y[n]$, $y[n+1], \ldots$ help in reconstructing $x[0]$? No! (Caley-Hamilton theorem). So, if the original state is not "observable" with n measurements, it will not be observable with more than n measurements either.
- As we know $u[k]$ and the system dynamics, if system is observable we can determine the entire state sequence $x[k]$, $k \geq 0$ from $x[0]$.

$$\begin{aligned} x[n] &= A^n x[0] + \sum_{i=0}^{n-1} A^{n-1-i} B u[k] \\ &= A^n \mathcal{O}^{-1}\left[\begin{bmatrix} y[0] \\ \vdots \\ y[n-1] \end{bmatrix} - \Psi \begin{bmatrix} u[0] \\ \vdots \\ u[n-1] \end{bmatrix}\right] + \mathcal{C} \begin{bmatrix} u[n-1] \\ \vdots \\ u[0] \end{bmatrix}. \end{aligned}$$

EXAMPLE: Given the following discrete-time system, find the observability and controllability matrices

$$A = \begin{bmatrix} 0.5 & 0 \\ 0 & 1 \end{bmatrix}, \qquad B = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \qquad C = \begin{bmatrix} 1 & -1 \end{bmatrix}, \qquad D = 0.$$

- This is quite easily done:

$$\mathcal{O} = \begin{bmatrix} C \\ CA \end{bmatrix} = \begin{bmatrix} 1 & -1 \\ 0.5 & -1 \end{bmatrix}, \quad \text{and} \quad \mathcal{C} = \begin{bmatrix} B & AB \end{bmatrix} = \begin{bmatrix} 1 & 0.5 \\ 0 & 0 \end{bmatrix}.$$

- This particular system *is* observable but *is not* controllable.

5.5: State-space realization problem

- The system-ID problem for state-space systems is sometimes called the "realization problem."
- That is, we wish to find a realization (a set of A, B, C, and D matrices) that describe a system's dynamics.
- We consider two separate realization problems here:

PROBLEM A: From system's Markov parameters (or transfer function), find system dimension n and (A, B, C, D), up to similarity transforms.

- The Ho–Kalman and ERA methods in this chapter (and ERA-DC for noise-affected systems) address this problem.

PROBLEM B: Given non-specific input–output data, find the system dimension n and (A, B, C, D), up to similarity transforms.

- The subspace methods introduced in this chapter and continued in the next chapter address this problem.

The Ho–Kalman method

- One of the first (maybe *the* first) state-space realization methods was introduced by Ho and Kalman.[1]
- We study it because it allows a natural introduction of some key concepts from system theory and advanced linear algebra that will be needed for the remainder of the course.

[1] B.L. Ho and R.E. Kalman, "Effective Construction of Linear State Variable Models from Input/Output Functions," *Regelungstechnik*, vol. 14, no. 12, pp. 545–8, 1966.

- It is too restrictive for most practical system-ID problems (*i.e.*, Markov parameters are not generally available to us); however, there may be situations where it is still useful.
- Notice that something curious happens when we multiply the observability and controllability matrices together:

$$\mathcal{OC} = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{n-1} \end{bmatrix} \begin{bmatrix} B & AB & A^2B & \cdots & A^{n-1}B \end{bmatrix}$$

$$= \begin{bmatrix} CB & CAB & CA^2B & \cdots & CA^{n-1}B \\ CAB & CA^2B & CA^3B & & \\ CA^2B & CA^3B & CA^4B & & \\ \vdots & & & \ddots & \vdots \\ CA^{n-1}B & & & \cdots & CA^{2n-2}B \end{bmatrix}.$$

- Notice that we get a Hankel matrix—a matrix having constant skew diagonals (an upside-down Toeplitz matrix).
- Note also that the values on the skew diagonals are the Markov parameters of the system (excluding g_0 and g_k for $k > 2n-1$)

$$\mathcal{H} = \mathcal{OC} = \begin{bmatrix} g_1 & g_2 & \cdots & g_n \\ g_2 & g_3 & & \\ \vdots & & \ddots & \vdots \\ g_n & & \cdots & g_{2n-1} \end{bmatrix}.$$

- Ho–Kalman assumes that we know the Markov parameters.
  - Knowledge of g_0 gives us D directly.

- Knowledge of the rest of the Markov parameters will ultimately result in A, B, and C.

- To use Ho–Kalman, we must first form the Hankel matrix $\mathcal{H}$.
- The next step is to factor $\mathcal{H} = \mathcal{O}\mathcal{C}$ into its $\mathcal{O}$ and $\mathcal{C}$ components.
- The third step is to use $\mathcal{O}$ and $\mathcal{C}$ to find A, B, and C.

ISSUE I: We don't know n. So, how do we form $\mathcal{H}$ in the first place? That is, when do we stop adding unit-pulse-response values to $\mathcal{H}$?

ANSWER: If there is no noise, then the rank of $\mathcal{H}$ is equal to n. Keep adding data until the rank doesn't increase.

- If there is noise, then we must work a little harder.

ISSUE II: How do we compute A, B, and C from $\mathcal{O}$ and $\mathcal{C}$?

ANSWER: C is extracted as the first block row of $\mathcal{O}$; B is extracted as the first block column of $\mathcal{C}$. We'll see how to get A shortly.

ISSUE III: How do we do the factoring of $\mathcal{H}$ into $\mathcal{O}$ and $\mathcal{C}$?

ANSWER: It doesn't matter, at least in principle. Any matrices $\mathcal{O}$ and $\mathcal{C}$ such that $\mathcal{O}\mathcal{C} = \mathcal{H}$ are okay.

- To see this latter point, consider what happens to the observability and controllability matrices when the state-space model undergoes a similarity transformation.
  - Recall that $\bar{A} = T^{-1}AT$, $\bar{B} = T^{-1}B$, and $\bar{C} = CT$.
  - The observability and controllability matrices of the new representation are

$$\bar{\mathcal{O}} = \begin{bmatrix} \bar{C} \\ \bar{C}\bar{A} \\ \vdots \\ \bar{C}\bar{A}^{n-1} \end{bmatrix} = \begin{bmatrix} CT \\ CTT^{-1}AT \\ \vdots \\ CT(T^{-1}AT)^{n-1} \end{bmatrix} = \mathcal{O}T$$

$$\bar{\mathcal{C}} = \begin{bmatrix} \bar{B} & \bar{A}\bar{B} & \cdots & \bar{A}^{n-1}\bar{B} \end{bmatrix}$$

$$= \begin{bmatrix} T^{-1}B & T^{-1}ATT^{-1}B & \cdots & (T^{-1}AT)^{n-1}T^{-1}B \end{bmatrix} = T^{-1}\mathcal{C}.$$

- Therefore, $\bar{\mathcal{O}}\bar{\mathcal{C}} = (\mathcal{O}T)(T^{-1}\mathcal{C}) = \mathcal{O}\mathcal{C}$.
  - If we factor $\mathcal{H}$ one way, we end up with a representation that has one set of $\mathcal{O}$ and $\mathcal{C}$.
  - If we factor $\mathcal{H}$ any other way, we end up with a representation that has an alternate set of $\bar{\mathcal{O}}$ and $\bar{\mathcal{C}}$.
  - But, these representations are related via a similarity transformation T.
- That is, no matter how we factor $\mathcal{H}$, we end up with different A, B, and C matrices, but the same input–output relationship (same transfer function, same unit-pulse response, but different state descriptions).
  - For example, we could choose to let $\mathcal{O} = I$, and then $\mathcal{C} = \mathcal{H}$. This will result in an A, B, and C that are in "observability canonical form." (cf. ECE5520)
  - Or, we could choose to let $\mathcal{C} = I$, and then $\mathcal{O} = \mathcal{H}$. This will result in an A, B, and C that are in "controllability canonical form."

ISSUE IV: Is there a "best" way to factor $\mathcal{H}$? Probably yes, but first we need to understand the SVD algorithm.

5.6: Quadratic forms

- We take a diversion to study/review some linear algebra concepts that are critical in a lot of advanced control-systems study, and are directly applicable to the problem at hand.
- The connections will not be immediately obvious, but we will soon show how they apply specifically to the Ho–Kalman problem, and will use these properties for much of the remainder of the course.

*Eigenvalues and eigenvectors of symmetric matrices (*i.e., $A = A^T$*)*

FACT: The eigenvalues of symmetric matrix $A \in \mathbb{R}^{n\times n}$ are real.

- To see this, suppose $Av = \lambda v$, $v \neq 0$, $v \in \mathbb{C}^n$.
  - Then, $(v^*)^T Av = (v^*)^T (Av) = \lambda (v^*)^T v = \lambda \sum_{i=1}^{n} |v_i|^2$.
  - Also, $(v^*)^T Av = ((Av)^*)^T v = ((\lambda v)^*)^T v = \lambda^* \sum_{i=1}^{n} |v_i|^2$.
- So, $\lambda = \lambda^*$, which means that $\lambda \in \mathbb{R}$ (hence, we can assume $v \in \mathbb{R}^n$).

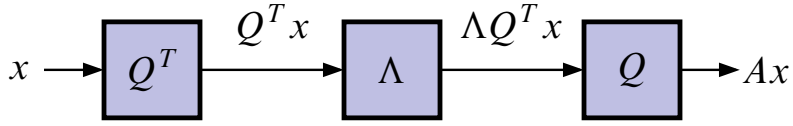
FACT: There is a set of orthonormal eigenvectors of A. *i.e.*, $q_1, \ldots, q_n$, such that $Aq_i = \lambda_i q_i$ and $q_i^T q_j = \delta_{ij}$. We'll show this for λ_i distinct.

- Suppose $v_1, \ldots, v_n$ is a set of linearly independent eigenvectors of A.
- That is, suppose $Av_i = \lambda_i v_i, \qquad \|v_i\| = 1.$
- Then, we have $v_i^T (Av_j) = \lambda_j v_i^T v_j = (Av_i)^T v_j = \lambda_i v_i^T v_j$.
- This gives $\left(\lambda_i - \lambda_j\right) v_i^T v_j = 0$ for $i \neq j$ and $\lambda_i \neq \lambda_j$. Hence, $v_i^T v_j = 0$.
  - In this case, we can say that the eigenvectors *are* orthogonal.

  - In the general case (λ_i not distinct) we must say that the eigenvectors *can be chosen* to be orthogonal.

- Grouping the eigenvectors into an (orthogonal) matrix Q allows us to write $Q^{-1}AQ = Q^TAQ = \Lambda$.
- Hence, we can express A as $A = Q\Lambda Q^T = \sum_{i=1}^{n} \lambda_i q_i q_i^T$.

INTERPRETATION: This allows us to consider $Ax = Q\Lambda Q^T x$ as



- Rotation of x by Q^T (which resolves the input x into q_i coordinates).
- Diagonal real scale (dilation by Λ, scaling each coordinate by λ_i).
- Rotate back by Q (reconstituting with basis q_i).

Quadratic forms

- A function $f : \mathbb{R}^n \to \mathbb{R}$ of the form

$$f(x) = x^TAx = \sum_{i,j}^{n} A_{ij}x_ix_j$$

  is called a quadratic form.

- In a quadratic form, we may assume A is symmetric since $y = y^T$, so

$$y = x^TAx = x^TA^Tx = x^T((A + A^T)/2)x,$$

  where $(A + A^T)/2$ is called the symmetric part of A.

- Suppose that $A = A^T$ and $A = Q\Lambda Q^T$ with eigenvalues sorted so $\lambda_1 \geq \cdots \geq \lambda_n$. Then,

$$\begin{aligned} x^T Ax &= x^T Q \Lambda Q^T x \\ &= (Q^T x)^T \Lambda (Q^T x) \\ &= \sum_{i=1}^{n} \lambda_i (q_i^T x)^2 \\ &\leq \lambda_1 \sum_{i=1}^{n} (q_i^T x)^2 = \lambda_1 \left[(Q^T x)^T I (Q^T x) \right] = \lambda_1 x^T x \\ &= \lambda_1 \|x\|^2 . \end{aligned}$$

- That is, we have $x^T Ax \leq \lambda_1 x^T x$.
- By a similar argument, we can find that $x^T Ax \geq \lambda_n \|x\|^2$, so we have

$$\lambda_n x^T x \leq x^T Ax \leq \lambda_1 x^T x.$$

- Sometimes λ_1 is called $\lambda_{\max}$ and λ_n is called $\lambda_{\min}$.
- Note also that $q_1^T Aq_1 = \lambda_1 \|q_1\|^2$ and $q_n^T Aq_n = \lambda_n \|q_n\|^2$, so the inequalities are tight.

5.7: Matrix gain

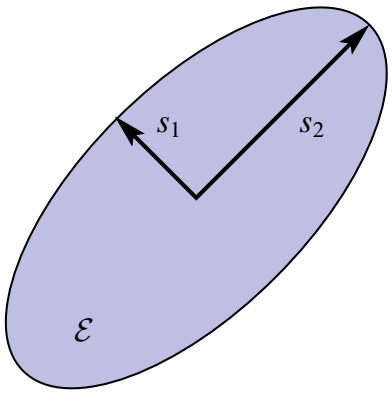
- Suppose now that $A \in \mathbb{R}^{m \times n}$ (not necessarily square or symmetric).
- For $x \in \mathbb{R}^n$, $\|Ax\| \,/\, \|x\|$ gives the amplification factor or gain of A in the direction x.
- Since this gain varies with direction of the input x, we might be interested in knowing:
  - What is the maximum gain and corresponding gain direction of A?
  - What is the minimum gain of A (and corresponding gain direction)?
  - How does gain of A vary with direction?
- The maximum gain $\max_{x \neq 0} \|Ax\| \,/\, \|x\|$ is called the matrix norm or spectral norm of A, and is denoted $\|A\|$.
- We have already developed the machinery to evaluate this quantity:

$$\max_{x \neq 0} \frac{\|Ax\|^2}{\|x\|^2} = \max_{x \neq 0} \frac{x^T A^T A x}{\|x\|^2} = \lambda_{\max}(A^T A),$$

  so we have $\|A\| = \sqrt{\lambda_{\max}(A^T A)}$.
- Similarly, the minimum gain is given by $\min_{x \neq 0} \|Ax\| \,/\, \|x\| = \sqrt{\lambda_{\min}(A^T A)}$.
- Note that $A^T A \in \mathbb{R}^{n \times n}$ is symmetric and $A^T A \geq 0$ so $\lambda_{\min}$ and $\lambda_{\max} \geq 0$.
- The "max gain" input direction is $x = q_1$, the eigenvector of $A^T A$ associated with $\lambda_{\max}$.
- The "min gain" input direction is $x = q_n$, the eigenvector of $A^T A$ associated with $\lambda_{\min}$.

EXAMPLE: Consider the set $\mathcal{E} = \{x \mid x^T A x \leq 1\}$, illustrated by the figure

- The semi-axes of this ellipse are governed by $s_i = \lambda_i^{-1/2} q_i$. That is,
  - The eigenvectors determine the directions of the semiaxes;
  - The eigenvalues determine the lengths of the semiaxes.



- In direction q_1, $x^T Ax$ is *large*, hence the ellipsoid is *thin* in direction q_1.
- In direction q_2, $x^T Ax$ is *small*, hence ellipsoid is *wide* in direction q_2.
- $\sqrt{\lambda_{\max}/\lambda_{\min}}$ gives the maximum *eccentricity*.

EXAMPLE: Consider the matrix

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} \qquad \text{so} \qquad A^T A = \begin{bmatrix} 35 & 44 \\ 44 & 56 \end{bmatrix}.$$

- For this matrix, we have

$$A^T A = \begin{bmatrix} 0.620 & 0.785 \\ 0.785 & -0.620 \end{bmatrix} \begin{bmatrix} 90.7 & 0 \\ 0 & 0.265 \end{bmatrix} \begin{bmatrix} 0.620 & 0.785 \\ 0.785 & -0.620 \end{bmatrix}^T.$$

- So, we have $\|A\| = \sqrt{\lambda_{\max}(A^T A)} = 9.53$,

$$\left\| \begin{bmatrix} 0.620 \\ 0.785 \end{bmatrix} \right\| = 1, \qquad \left\| A \begin{bmatrix} 0.620 \\ 0.785 \end{bmatrix} \right\| = \left\| \begin{bmatrix} 2.18 \\ 4.99 \\ 7.78 \end{bmatrix} \right\| = 9.53.$$

- We also have minimum gain $\sqrt{\lambda_{\min}(A^T A)} = 0.514$.

$$\left\| \begin{bmatrix} 0.785 \\ -0.620 \end{bmatrix} \right\| = 1, \qquad \left\| A \begin{bmatrix} 0.785 \\ -0.620 \end{bmatrix} \right\| = \left\| \begin{bmatrix} 0.46 \\ 0.14 \\ -0.18 \end{bmatrix} \right\| = 0.514.$$

- And, for all $x \neq 0$, we then have $0.514 \leq \|Ax\| / \|x\| \leq 0.953$.

5.8: Singular value decomposition (SVD) of a matrix

- A more complete picture of the gain properties of A is given by the singular value decomposition (SVD) of $A \in \mathbb{R}^{m\times n}$, where $\mathrm{rank}(A) = r$:

$$A = U\Sigma V^T.$$

  - $U = [u_1, \ldots, u_r] \in \mathbb{R}^{m\times r}$, and $U^TU = I$, and u_i are the left or output singular vectors of A.
  - $V = [v_1, \ldots, v_r] \in \mathbb{R}^{n\times r}$, and $V^TV = I$, and v_i are the right or input singular vectors of A.
  - $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_r)$ where $\sigma_1 \geq \cdots \geq \sigma_r > 0$, and σ_i are the (nonzero) singular values of A.

- The above is called a compact SVD. Most often, we compute a full SVD, where

  - $U = [u_1, \ldots, u_m] \in \mathbb{R}^{m\times m}$, and $U^TU = I$,
  - $V = [v_1, \ldots, v_n] \in \mathbb{R}^{n\times n}$, and $V^TV = I$,
  - The matrix $\Sigma \in \mathbb{R}^{m\times n}$ is "diagonal"

$$\Sigma = \begin{bmatrix} \sigma_1 & & 0 & 0 \\ & \ddots & & 0 \\ 0 & & \sigma_m & 0 \end{bmatrix} \text{ or } \Sigma = \begin{bmatrix} \sigma_1 & & 0 \\ & \ddots & \\ 0 & & \sigma_n \end{bmatrix} \text{ or } \Sigma = \begin{bmatrix} \sigma_1 & & 0 \\ & \ddots & \\ 0 & & \sigma_n \\ 0 & 0 & 0 \end{bmatrix}$$

    when $m < n$, $m = n$ and $m > n$, respectively.
  - In this case, $\sigma_1 \geq \cdots \geq \sigma_r > 0$, and $\sigma_i = 0$ for $i > r$.
  - In MATLAB, `svd.m` and `svds.m`

- We often write the full SVD as partitioned:

$$A = \left[\begin{array}{c:c} U_1 & U_2 \end{array}\right] \left[\begin{array}{c:c} \Sigma_1 & 0_{r\times(n-r)} \\ \hdashline 0_{(m-r)\times r} & 0_{(m-r)\times(n-r)} \end{array}\right] \left[\begin{array}{c} V_1^T \\ \hdashline V_2^T \end{array}\right],$$

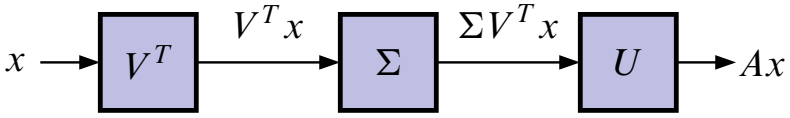
where $A = U_1\Sigma_1 V_1^T$ is the compact SVD.

- The matrices U and V form orthonormal bases for the four fundamental subspaces
  - The first r columns of V form a basis for the row space of A;
    - The row space of a matrix is the set of all possible linear combinations of its row vectors. It's the span of the input vectors x to $y = Ax$ that result in nonzero y.
  - The last $n - r$ columns of V form a basis for the nullspace of A;
    - The null space of a matrix is the set of all possible vectors x such that $y = Ax = 0$.
  - The first r columns of U form a basis for the column space of A;
    - The column space is also the range of the matrix. It's the set of all possible outputs y generated as $y = Ax$.
  - The last $m - r$ columns of U form a basis for the nullspace of A^T.
    - The nullspace of A^T is the set of all outputs y that *cannot* be generated as $y = Ax$.
- So, immediately we see some value to the SVD: It gives us the rank of the matrix, and these four bases, which can give us range space, nullspace, etc.
- Also, the SVD is computationally much simpler than an eigensystem calculation, which makes finding matrix norms far simpler:

$$A^T A = (U\Sigma V^T)^T(U\Sigma V^T) = V\Sigma^2 V^T$$

so $\sigma_i = \sqrt{\lambda_i(A^T A)}$ and hence $\|A\| = \sigma_1$.

- Can view operation $y = Ax$ as $y = (U\Sigma V^T)x$, decomposing the operation into



  - Computing coefficients of x along the input directions $v_1, \ldots, v_r$ (rotating by V^T)
  - Scaling the coefficients by σ_i (dilation)
  - Reconstituting along output directions $u_1, \ldots, u_r$.
- The difference between this and the eigenvalue decomposition for symmetric A is that the input and output directions are different for a general matrix A.
  - v_1 is the most sensitive (highest gain) input direction;
  - u_1 is the highest gain output direction. $Av_1 = \sigma_1 u_1$.
- The SVD gives clearer picture of gain as a function of input/output directions.

EXAMPLE: Consider $A \in \mathbb{R}^{4\times 4}$ with $\Sigma = \mathrm{diag}(10, 7, 0.1, 0.05)$.

- Input components along directions v_1 and v_2 are amplified (by about 10) and come out mostly along the plane spanned by u_1 and u_2.
- Input components along directions v_3, v_4 are attenuated (by about 10).
- $\|Ax\| \,/\, \|x\|$ can range between 10 and 0.05; A is nonsingular.
- For some applications you might say that A is *effectively* rank 2 (this will be important for us later).

5.9: Moore–Penrose pseudo-inverse

- Suppose we have a matrix $A \in \mathbb{R}^{m \times n}$ with $A \neq 0$ and SVD given by $A = U \Sigma V^T$.
- Then, there exists a unique solution X that satisfies the four Moore–Penrose conditions for a pseudo-inverse of A:

  1. $AXA = A$; 2. $XAX = X$; 3. $(AX)^T = AX$; 4. $(XA)^T = XA$.
- The unique solution is given by:

$$X = A^\dagger = V_1 \Sigma_1^{-1} U_1^T,$$

  where $U_1 \Sigma_1 V_1^T$ is the compact SVD of A.
- In this case, $X = A^\dagger$ is called the Moore–Penrose generalized inverse, or the pseudo-inverse, of A.

PROOF: Direct substitution shows that the four conditions are satisfied. For example, consider the first case:

$$\begin{aligned} AXA &= (U_1 \Sigma_1 \underbrace{V_1^T)(V_1}_{I} \Sigma_1^{-1} U_1^T) A \\ &= (U_1 \underbrace{\Sigma_1 \Sigma_1^{-1}}_{I} U_1^T) A \\ &= (\underbrace{U_1 U_1^T}_{I}) A \\ &= A. \end{aligned}$$

- The pseudo-inverse has many of the traditional matrix inverse properties, but is not limited to square matrices.
- If A is tall and full rank, then $A^\dagger = (A^T A)^{-1} A^T$, and $A^\dagger$ gives the least-squares approximate solution to the problem $y = Ax + v$ as $x_{\text{ls}} = A^\dagger y$.

- If A is wide and full rank, then $A^\dagger = A^T(AA^T)^{-1}$ gives the least-norm solution to the problem $y = Ax + v$ as $x_{\text{ln}} = A^\dagger y$.
- In general, $x_{\text{pinv}} = A^\dagger y$ is the minimum-norm, least-squares approximate solution.
- This is an incredibly handy result—least squares optimization finds the global optimum solution with one line of MATLAB code!

Low-rank approximations

- Suppose that $A \in \mathbb{R}^{m\times n}$ and $\text{rank}(A) = r$, with SVD $A = U\Sigma V^T = \sum_{i=1}^{r} \sigma_i u_i v_i^T$.
- We want to approximate A by $\hat{A}$, where $\text{rank}(\hat{A}) \leq p < r$ such that $\hat{A} \approx A$ in the sense that $\left\|A - \hat{A}\right\|$ is minimized.
- The optimal rank p approximator is $\hat{A} = \sum_{i=1}^{p} \sigma_i u_i v_i^T$ and hence

$$\left\|A - \hat{A}\right\| = \left\|\sum_{i=p+1}^{r} \sigma_i u_i v_i^T\right\| = \sigma_{p+1}$$

because σ_{p+1} is the maximum remaining singular value.

INTERPRETATION I: SVD dyads $u_i v_i^T$ are ranked in order of 'importance'; take p of them to get a rank p approximant.

INTERPRETATION II: We can now form another interpretation of σ_i

$$\sigma_i = \min\{\|A - B\| \mid \text{rank}(B) \leq i - 1\},$$

i.e., the distance (measured by matrix norm) to the nearest rank $i-1$ matrix.

- For example, if $A \in \mathbb{R}^{n \times n}$, then $\sigma_n = \sigma_{\min}$ is the distance to the nearest singular matrix.
- Hence, small $\sigma_{\min}$ means that A is near to a singular matrix.

APPLICATION: We can use this idea to simplify models (very useful). Suppose that

- $y = Ax + v$ where $A \in \mathbb{R}^{100 \times 30}$ has SVs 10, 7, 2, 0.5, 0.01, . . . , 0.0001.
- $\|x\|$ is on the order of 1, and unknown error or noise v has norm on the order of 0.1.
- Then, the terms $\sigma_i u_i v_i^T x$ for $i = 5, \ldots, 30$ are substantially smaller than the noise term v.
- So, we can approximate $y = Ax + v$ by the much simplified model $y = \sum_{i=1}^{4} \sigma_i u_i v_i^T x + v$.

5.10: Back to Ho–Kalman

- Recall Ho–Kalman "ISSUE I," how do we form the Hankel matrix $\mathcal{H}$ if we don't know the dimension of the system state n?
- To address this issue, consider the infinite, skew-diagonal matrix $\mathcal{H}$:

$$\mathcal{H} = \begin{bmatrix} g_1 & g_2 & g_3 & g_4 & \cdots \\ g_2 & g_3 & g_4 & g_5 & \cdots \\ g_3 & g_4 & g_5 & g_6 & \cdots \\ g_4 & g_5 & g_6 & g_7 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

  where the entries g_k correspond to the Markov parameters for the given system.
- This form is called an infinite Hankel matrix, or Hankel operator.
- We can also define a finite Hankel matrix, formed by the first k rows and l columns of $\mathcal{H}$:

$$\mathcal{H}_{k,l} = \begin{bmatrix} g_1 & g_2 & g_3 & \cdots & g_l \\ g_2 & g_3 & g_4 & \cdots & g_{l+1} \\ g_3 & g_4 & g_5 & \cdots & g_{l+2} \\ \vdots & \vdots & \vdots & & \vdots \\ g_k & g_{k+1} & g_{k+2} & \cdots & g_{k+l-1} \end{bmatrix}.$$

- This finite Hankel matrix factors into $\mathcal{H}_{k,l} = \mathcal{O}_k \mathcal{C}_l$ where:

$$\mathcal{O}_k = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{k-1} \end{bmatrix}, \quad \mathcal{C}_l = \begin{bmatrix} B & AB & A^2B & \cdots & A^{l-1}B \end{bmatrix}.$$

- The approach we will take is to make a $\mathcal{H}_{k,l}$ of larger size than we expect for a hypothesized value of n. That is, $k > n$ and $l > n$.
  - Therefore $\mathcal{O}_k \neq \mathcal{O}$ and $\mathcal{C}_l \neq \mathcal{C}$ even though the matrices have the same general form. We call $\mathcal{O}_k$ the extended observability matrix and $\mathcal{C}_l$ the extended controllability matrix.
- We then apply the SVD to $\mathcal{H}_{k,l}$

$$\begin{aligned}\mathcal{H}_{k,l} &= U\Sigma V^T = U\Sigma^{1/2}\Sigma^{1/2}V^T \\ &= U\Sigma^{1/2}TT^{-1}\Sigma^{1/2}V^T \\ &= \underbrace{(U\Sigma^{1/2}T)}_{\mathcal{O}_k}\underbrace{(T^{-1}\Sigma^{1/2}V^T)}_{\mathcal{C}_l}.\end{aligned}$$

- The first n non-zero singular values provide insight into model order.
  - Problem: Noisy data yields more than n non-zero singular values.
  - Need to look at a few and determine when there is a "significant" drop off in the magnitude of the SVDs.
- Note that this approach also gives us $\mathcal{O}_k$ and $\mathcal{C}_l$ automatically in a "balanced realization". Solves "ISSUE III" and "ISSUE IV".
  - T must be invertible, but selection of T is otherwise arbitrary. Usually use $T = I$.
- How to decompose further into (A, B, C) to solve "ISSUE II"?
- Note the shift property of a Hankel matrix. If we shift $\mathcal{H}$ up by one block row, we get $\mathcal{H}^{\uparrow}_{k+1,l} = \mathcal{O}_k A \mathcal{C}_l$.

$$\mathcal{H}^{\uparrow}_{k+1,l} = \begin{bmatrix} g_2 & g_3 & g_4 & \cdots & g_{l+1} \\ g_3 & g_4 & g_5 & \cdots & g_{l+2} \\ \vdots & \vdots & \vdots & & \vdots \\ g_k & g_{k+1} & g_{k+2} & \cdots & g_{k+l-1} \\ g_{k+1} & g_{k+2} & g_{k+3} & \cdots & g_{k+l} \end{bmatrix}$$

$$= \begin{bmatrix} CAB & CA^2B & CA^3B & \cdots & CA^lB \\ CA^2B & CA^3B & CA^4B & & CA^{l+1}B \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ CA^{k-1}B & CA^kB & CA^{k+1}B & \cdots & CA^{k+l-2}B \\ CA^kB & CA^{k+1}B & CA^{k+2}B & \cdots & CA^{k+l-1}B \end{bmatrix}$$

$$= \mathcal{O}^{\uparrow}_{k+1}\mathcal{C}_l = \mathcal{O}_k\mathcal{C}^{\leftarrow}_{l+1} = \mathcal{O}_kA\mathcal{C}_l.$$

- Using the pseudo-inverse to solve for A gives $A = \mathcal{O}_k^{\dagger}\mathcal{H}^{\uparrow}_{k+1,l}\mathcal{C}_l^{\dagger}$.
- As before, we extract B from the first block column of the controllability matrix we derived via SVD.
- Also, extract C from the first block row of the observability matrix we derived via SVD, and set $D = g_0$.

Ho–Kalman Method: Algorithm Steps

STEP I: Collect the unit-pulse response values into two Hankel matrices

1. An original finite Hankel matrix
2. A shifted version matrix of the original Hankel matrix (same size)

STEP II: Compute the SVD of the (unshifted) Hankel matrix

- Identify system order from the singular values

- May need to iterate on choice of Hankel matrix (discussed later)

STEP III: Compute the extended observability and controllability matrices

- Use appropriately dimensioned SVD components
- Typically use $T = I_n$

STEP IV: Identify the system matrices (A, B, C). $D = g_0$.

5.11: Fibonacci example

EXAMPLE: Suppose that a unit pulse yields the following response:

$$y = (0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, \cdots).$$

- We recognize this output as the Fibonacci sequence generated by $g_k = g_{k-1} + g_{k-2}$ with initial conditions $g_0 = 0$ and $g_1 = 1$.
- A typical realization for this sequence is given by the state-space system:

$$A = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 0 \end{bmatrix}, \quad D = 0.$$

- We'll try to come up with an equivalent realization based on only the unit-pulse response.

```
% Define true system, compute the Markov parameters as "y"
A  = [0 1; 1 1]; B  = [1; 1]; C  = [1 0]; D  = 0; dt = 1;
sysTrue = ss(A,B,C,D,dt);      % "typical" Fibonacci ss model
y = dt*impulse(sysTrue);       % scale by dt to get unit-pulse response
```

- The Hankel matrices that we will require are:

$$\mathcal{H}_{4,4} = \begin{bmatrix} 1 & 1 & 2 & 3 \\ 1 & 2 & 3 & 5 \\ 2 & 3 & 5 & 8 \\ 3 & 5 & 8 & 13 \end{bmatrix}, \quad \mathcal{H}_{5,4}^{\uparrow} = \begin{bmatrix} 1 & 2 & 3 & 5 \\ 2 & 3 & 5 & 8 \\ 3 & 5 & 8 & 13 \\ 5 & 8 & 13 & 21 \end{bmatrix}.$$

```
% Form H{4,4} and shifted H{5,4}. Note:  Do not include "zero-th"
% parameter (first element of y), which corresponds to the matrix D.
bigHankel = hankel(y(2:end)); % don't forget to omit h(0) term = y(1)
H = bigHankel(1:4,1:4);       % for this example, keep only 4x4 portion
Hup = bigHankel(2:5,1:4);     % shifted H{5,4}
```

- The SVD yields

$$\sigma_1 = 54.56 \quad \sigma_2 = 0.43988 \quad \sigma_i = 0, \; i \geq 3$$

  which indicates that $n = 2$.

```
% Compute singular values of Hankel matrix
[U,S,V] = svd(H);

% Identify system order off-line as n = 2 based on values of S
n = 2;
```

- We now extract the two left columns of U and V

$$U = V = \begin{bmatrix} -0.1876 & 0.7947 \\ -0.3035 & -0.4911 \\ -0.4911 & 0.3035 \\ -0.7947 & -0.1876 \end{bmatrix}.$$

- Compute the extended observability and controllability matrices

$$\mathcal{C}_l = \Sigma^{1/2} V^T = \begin{bmatrix} -0.8507 & -1.3764 & -2.2270 & -3.6034 \\ 0.5257 & -0.3249 & 0.2008 & -0.1241 \end{bmatrix}$$

$$\mathcal{O}_k = U \Sigma^{1/2} = \mathcal{C}_l^T.$$

```
% Compute extended observability and controllability matrices, sized to
% the order for the system inferred by the singular values.
Us = U(:,1:n); Ss = S(1:n,1:n); Vs = V(:,1:n);
Ok = Us*sqrtm(Ss); Cl = sqrtm(Ss)*Vs';
```

- Identify the system matrices $(\hat{A}, \hat{B}, \hat{C})$ up to similarity transform

$$\hat{A} = O_k^{\dagger} H_{k+1,l}^{\uparrow} C_l^{\dagger} = \begin{bmatrix} 1.6180 & 0 \\ 0 & -0.6180 \end{bmatrix}$$

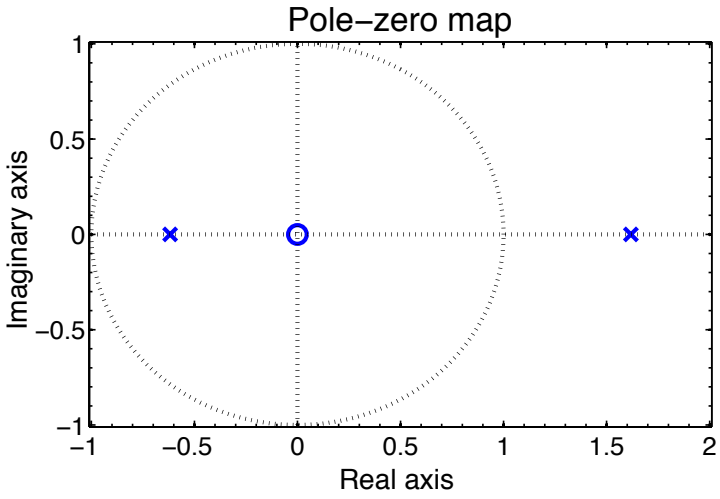
$$\hat{B} = \mathcal{C}_l(1:n, 1:m) = \mathcal{C}_l(1:2, 1) = \begin{bmatrix} -0.8057 \\ 0.5257 \end{bmatrix}$$

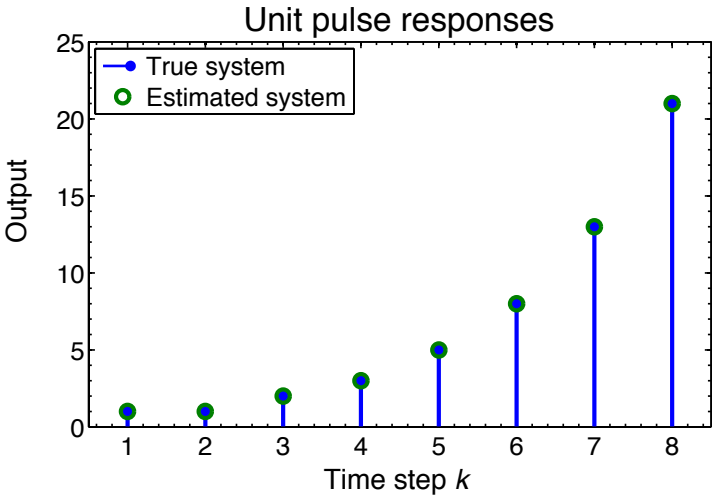
$$\hat{C} = \mathcal{O}_k(1:p, 1:n) = \mathcal{O}_k(1, 1:2) = \begin{bmatrix} -0.8057 & 0.5257 \end{bmatrix}$$

$$\hat{D} = g_0 = 0.$$

```
% Identify system assuming p = m = 1 (SISO), using shifted Hankel matrix
Ahat = (Ok\Hup)/Cl; Bhat = Cl(:,1); Chat = Ok(1,:); Dhat = y(1);
sysEst = ss(Ahat, Bhat, Chat, Dhat, dt);
```

- Now, let's compare the true and identified ("estimated") systems
  - Same pole-zero mapping (eigenvalues...transfer function)
  - Same unit-pulse responses





COMMENTS: Selecting an appropriate amount of output data to store may require iteration ("how big an $\mathcal{H}$ do I need?")

- Until $\text{rank}(\mathcal{H}_{k,l}) = \text{rank}(\mathcal{H}_{k-1,l-1})$, or
- Until the next singular value is "insignificant."

- Interesting to note that $A = A^T$ and that $B = C^T$ for the identified system in the example.
  - This property holds for square Hankel matrices

- The identification process will work so long as the Hankel matrix dimensions exceed the system order ($\mathcal{H}$ need not be square)

REMAINING QUESTION: How bad is it if there is noise on the g_k values?

- Ho–Kalman is an "output error" approach. If noise is white, estimate will be decent. Determine order n by guessing at the division between true system singular values and noise singular values.
- If noise is not white, then the method will try to fit the correlation of the noise with system dynamics. This can be quite poor (see examples in last chapter on fitting an OE model to a non-OE system).
- Even if noise is white, unit-pulse response will decay for a stable system while noise power does not.
  - Noise will eventually dominate pulse response—augmenting $\mathcal{H}$ with extra data not helpful.
  - Really need a better way to handle general input/output data instead of specific unit-pulse response data.
  - This is the purpose of the subspace algorithms, which we look at shortly.

REMAINING QUESTION: From whence come the g_k?

- We may be able to measure these directly from a pulse response (but issues when there is noise).
- Can also perform inverse DFT of a frequency response, perhaps identified by `etfe.m` or `spa.m`.
- Alternately, can use general input–output data, organized as $Y = \mathcal{Y}U$, where

$$Y = \begin{bmatrix} y[0] & y[1] & \cdots & y[N-1] \end{bmatrix}$$

$$\mathcal{Y} = \begin{bmatrix} D & CB & CAB & \cdots & CA^{N-2}B \end{bmatrix}$$

$$U = \begin{bmatrix} u[0] & u[1] & u[2] & \cdots & u[N-1] \\ 0 & u[0] & u[1] & \cdots & u[N-2] \\ 0 & 0 & u[0] & \cdots & u[N-3] \\ \vdots & & & \vdots & \\ 0 & 0 & 0 & \cdots & u[0] \end{bmatrix},$$

where $Y \in \mathbb{R}^{m \times N}$, $\mathcal{Y} \in \mathbb{R}^{m \times pN}$, and $U \in \mathbb{R}^{pN \times N}$.

- Then, solve for $\mathcal{Y}$ via least squares: $\mathcal{Y} = YU^{\dagger}$.
- Note: U must be reasonably well conditioned for the pseudo-inverse to turn out well, and for lightly damped systems the number of parameters that must be solved for can be very high.

5.12: Eigensystem Realization Algorithm (ERA)

- Juang and Pappa formulated an extension of the Ho–Kalman method, and applied it to identification of the dynamics of flexible structures.[2]
  - They wanted to identify a realization (A, B, C, and D) and then to determine a continuous-time model of the modes (eigenvalues of the continuous-time A matrix) from that.
  - They called their algorithm the "Eigensystem Realization Algorithm" because it results in an eigensystem (modal description of the system).
  - Here, we are interested in only the first part of the algorithm, the realization part, but we still call it the ERA.
- The major feature of ERA that is of interest to us is that it is able to deal with faulty sensors or incomplete unit-pulse-response data.
  - For example: saturation of the sensors, data dropout from a communications link, an intermittent bit failure, or an external noise source that temporarily affects the data measured by the system.
  - We will still use a Hankel matrix approach that bears some similarity to Ho–Kalman. But, we will remove all rows and columns from the Hankel matrix that correspond to "bad" data.
  - Therefore, this algorithm is best for systems having long impulse responses where cutting out data doesn't cause too much degradation due to other noise issues.

[2] J.N. Juang and R.S. Pappa, "Eigensystem Realization Algorithm for Modal Parameter Identification and Model Reduction," *Journal of Guidance, Control, and Dynamics*, vol. 8, no. 5, pp. 620–27, 1985.

- We begin by forming an $r \times s$ generalized Hankel matrix

$$\mathcal{H}_{r,s}[k-1] = \begin{bmatrix} g_k & g_{k+t_1} & \cdots & g_{k+t_{s-1}} \\ g_{j_1+k} & g_{j_1+k+t_1} & \cdots & g_{j_1+k+t_{s-1}} \\ \vdots & \vdots & & \vdots \\ g_{j_{r-1}+k} & g_{j_{r-1}+k+t_1} & \cdots & g_{j_{r-1}+k+t_{s-1}} \end{bmatrix}_{rp \times ms}$$

  where j_i $(i=1,\ldots,r-1)$ and t_i $(i=1,\ldots,s-1)$ are arbitrary integers.

- Note that this is the same as the Hankel matrix used by the Ho–Kalman method, except that we have deleted all **block** rows and all **block** columns that have "bad" data in them. (We don't delete individual rows and columns in the non SISO case).
  - We will assume that $\mathcal{H}_{r,s}[0]$ and $\mathcal{H}_{r,s}[1]$ are available.

- We now define generalized controllability and observability matrices

$$\mathcal{C}_s = \begin{bmatrix} B, & A^{t_1}B, & \cdots, & A^{t_{s-1}}B \end{bmatrix}_{n \times ms}, \qquad \mathcal{O}_r = \begin{bmatrix} C \\ CA^{j_1} \\ \vdots \\ CA^{j_r-1} \end{bmatrix}_{rp \times n}.$$

- With these definitions, we note that $\mathcal{H}_{r,s}[k] = \mathcal{O}_r A^k \mathcal{C}_s$.
  - In particular, $\mathcal{H}_{r,s}[0] = \mathcal{O}_r \mathcal{C}_s$ and $\mathcal{H}_{r,s}[1] = \mathcal{O}_r A \mathcal{C}_s$.

- The method for finding a realization is similar to Ho–Kalman. We fist take the compact SVD of $\mathcal{H}_{r,s}[0]$:

$$\mathcal{H}_{r,s}[0] = U \Sigma V^T,$$

  where $U \in \mathbb{R}^{rp \times n}$, $V \in \mathbb{R}^{ms \times n}$, and $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_n)$.

- As with Ho–Kalman, we set $\mathcal{O}_r = U \Sigma^{1/2}$ and $\mathcal{C}_s = \Sigma^{1/2} V^T$.

- We then set B to be the left block column of $\mathcal{C}_s$, C to be the top block row of $\mathcal{O}_r$, and $D = g_0$.
- Finally, we set $A = \mathcal{O}_r^{\dagger}\mathcal{H}_{r,s}[1]\mathcal{C}_s^{\dagger} = \Sigma^{-1/2}U^T\mathcal{H}_{r,s}[1]V\Sigma^{-1/2}$.

EXAMPLE: Suppose again our Fibonacci system.

- This time, the pulse response experiment yields the following outputs

$$y = (0, 1, 1, 2, -32.12, 724.1, -87.4, 13, 21, 34, 55, 89, 144, 233, 377, 610 \cdots)$$

- The float-valued quantities in entries y_4, y_5, y_6 are suspicious (all others are integers) and thus will be discarded (label them $*$).

$$y = (0, 1, 1, 2, *, *, *, 13, 21, 34, 55, 89, 144, 233, 377, 610 \cdots)$$

- We can immediately set $D = g_0 = 0$. We can then form a 7×9 Hankel matrix from g_k, $k > 0$:

$$\mathcal{H}_{7,9} = \begin{bmatrix} 1 & 1 & 2 & * & * & * & 13 & 21 & 34 \\ 1 & 2 & * & * & * & 13 & 21 & 34 & 55 \\ 2 & * & * & * & 13 & 21 & 34 & 55 & 89 \\ * & * & * & 13 & 21 & 34 & 55 & 89 & 144 \\ * & * & 13 & 21 & 34 & 55 & 89 & 144 & 233 \\ * & 13 & 21 & 34 & 55 & 89 & 144 & 233 & 377 \\ 13 & 21 & 34 & 55 & 89 & 144 & 233 & 377 & 610 \end{bmatrix}.$$

- We're going to delete rows 2:6 and columns 3:6. This gives us index sets: $t_k = \{1, 6, 7, 8\}$ and $s_k = \{6, 7, 8, 9\}$.
- This deletion leaves us with 4×4 Hankel matrices

$$\mathcal{H}_{4,4}[0] = \begin{bmatrix} g_1 & g_2 & g_7 & g_8 \\ g_7 & g_8 & g_{13} & g_{14} \\ g_8 & g_9 & g_{14} & g_{15} \\ g_9 & g_{10} & g_{15} & g_{16} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 13 & 21 \\ 13 & 21 & 233 & 377 \\ 21 & 34 & 377 & 610 \\ 34 & 55 & 610 & 987 \end{bmatrix}$$

$$\mathcal{H}_{4,4}[1] = \begin{bmatrix} g_2 & g_3 & g_8 & g_9 \\ g_8 & g_9 & g_{14} & g_{15} \\ g_9 & g_{10} & g_{15} & g_{16} \\ g_{10} & g_{11} & g_{16} & g_{17} \end{bmatrix} = \begin{bmatrix} 1 & 2 & 21 & 34 \\ 21 & 34 & 377 & 610 \\ 34 & 55 & 610 & 987 \\ 55 & 89 & 987 & 1597 \end{bmatrix}.$$

- The SVD of $\mathcal{H}_{4,4}[0]$ yields

$$\sigma_1 = 1436.6 \quad \sigma_2 = 0.326 \quad \sigma_i = 0, \ i \geq 3,$$

  which indicates that $n = 2$. Note that the split of SVD magnitude has dramatically changed.

- We now extract the two left columns of U and V

$$U = \begin{bmatrix} -0.017218 & 0.99763 \\ -0.30897 & 0.045008 \\ -0.49993 & -0.051566 \\ -0.8089 & -0.0065579 \end{bmatrix}, \qquad V = \begin{bmatrix} -0.029259 & 0.84933 \\ -0.047328 & -0.52492 \\ -0.52492 & 0.047328 \\ -0.84933 & -0.029259 \end{bmatrix}.$$

- Again, note that U no longer equals V.

- The extended observability and controllability matrices are

$$\mathcal{O}_k = U\Sigma^{1/2} = \begin{bmatrix} -0.65263 & 0.56961 \\ -11.711 & 0.025698 \\ -18.949 & -0.029442 \\ -30.66 & -0.0037443 \end{bmatrix}$$

$$\mathcal{C}_l = \Sigma^{1/2} V^T = \begin{bmatrix} -1.109 & -1.7939 & -19.896 & -32.192 \\ 0.48494 & -0.29971 & 0.027022 & -0.016706 \end{bmatrix}.$$

- Identify $(\hat{A}, \hat{B}, \hat{C})$, the system matrices up to similarity transform

$$\hat{A} = \Sigma^{-1/2} U^T \mathcal{H}_{r,s}[1] V \Sigma^{-1/2} = \begin{bmatrix} 1.6180 & 0.0011579 \\ 2.6275 \times 10^{-7} & -0.61803 \end{bmatrix}$$

$$\hat{B} = \mathcal{C}_l(1:n, 1:m) = \mathcal{C}_l(1:2, 1) = \begin{bmatrix} -1.109 \\ 0.48494 \end{bmatrix}$$

$$\hat{C} = \mathcal{O}_k(1:p, 1:n) = \mathcal{O}_k(1, 1:2) = \begin{bmatrix} -0.65263 & 0.56961 \end{bmatrix}$$

- Note that these are quite different from the values we got for the Ho–Kalman example, but give identical transfer function, unit-pulse response, and pole-zero mapping. They both match the true system.
- Note that the realization is in general no longer balanced, even for square systems.

5.13: The subspace methods, data matrices

- Both the Ho–Kalman and ERA methods assumed that we had knowledge of the Markov parameters of the system.
- This is a limiting assumption. Better if we could work with general input–output data instead (state-space realization "PROBLEM B").
- The subspace methods do just this. They assume a model of the form

$$x[k+1] = Ax[k] + Bu[k] + w[k]$$

$$y[k] = Cx[k] + Du[k] + v[k],$$

where $u[k] \in \mathbb{R}^m$, $x[n] \in \mathbb{R}^n$, $y[k] \in \mathbb{R}^p$, $w[k] \in \mathbb{R}^n$, $v[k] \in \mathbb{R}^p$, and $w[k]$ and $v[k]$ are zero-mean stationary white processes with

$$\mathbb{E}\left[\begin{bmatrix} w[i] \\ v[i] \end{bmatrix}\begin{bmatrix} w^T[j] & v^T[j] \end{bmatrix}\right] = \begin{bmatrix} Q & S \\ S^T & R \end{bmatrix}\delta_{ij} \geq 0.$$

- In this chapter of notes, we look at the "deterministic" problem, with $w[k] \equiv 0$ and $v[k] \equiv 0$.
- In the next chapter, we look first at the "stochastic" problem with $u[k] \equiv 0$, then at the "combined deterministic-stochastic" problem.
  - We could discuss only the combined problem, and show that the pure deterministic and stochastic problems are simply special cases, but this would probably give us all aneurysms at trying to grapple with the entire complexity in one step.
  - Easier to "sneak up on" the general solution in a stepwise manner.
- For clarity, we use superscripts "d" and "s" to distinguish between the deterministic and stochastic problems. The deterministic model is

$$x^d[k+1] = Ax^d[k] + Bu[k]$$

$$y[k] = Cx^d[k] + Du[k].$$

Data matrices

- Key to the derivation and operation of the subspace methods is how the input–output data is organized into matrices.
- For $i > n$, we define a block Hankel control matrix:

$$U_{0|2i-1} \triangleq \left[\begin{array}{ccccc} u[0] & u[1] & u[2] & \cdots & u[N-1] \\ u[1] & u[2] & u[3] & \cdots & u[N] \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ u[i-1] & u[i] & u[i+1] & \cdots & u[i+N-2] \\ \hdashline u[i] & u[i+1] & u[i+2] & \cdots & u[i+N-1] \\ u[i+1] & u[i+2] & u[i+3] & \cdots & u[i+N] \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ u[2i-1] & u[2i] & u[2i+1] & \cdots & u[2i+N-2] \end{array}\right] \begin{array}{l} \uparrow \\ \\ \\ \\ \downarrow \\ \uparrow \\ \\ \\ \\ \downarrow \end{array} \begin{array}{l} \\ \\ \\ \text{“past”} \\ \\ \\ \\ \\ \text{“future”} \end{array}$$

$$\triangleq \left[\begin{array}{c} U_{0|i-1} \\ \hdashline U_{i|2i-1} \end{array}\right] \triangleq \left[\begin{array}{c} U_p \\ \hdashline U_f \end{array}\right],$$

  where the subscripts indicate the (first)|(last) elements of the left-hand column in each matrix.
- Note that the division between “past” and “future” is arbitrary, as all of the data in the matrices has already been collected.
- We choose to split the data set so that the first half is prior to some “present” point, and the second half is subsequent to that point.
- Note also that both the “past” and “future” inputs have many elements in common.

- For example, $u[i]$ can be found in both U_p and U_f.
- However, the corresponding columns of U_p and U_f have no elements in common; columnwise, U_f is future to U_p.

- We also find value in splitting the data one block row beneath the "present" point:

$$U_{0|2i-1} \triangleq \left[\begin{array}{ccccc} u[0] & u[1] & u[2] & \cdots & u[N-1] \\ u[1] & u[2] & u[3] & \cdots & u[N] \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ u[i-1] & u[i] & u[i+1] & \cdots & u[i+N-2] \\ u[i] & u[i+1] & u[i+2] & \cdots & u[i+N-1] \\ \hdashline u[i+1] & u[i+2] & u[i+3] & \cdots & u[i+N] \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ u[2i-1] & u[2i] & u[2i+1] & \cdots & u[2i+N-2] \end{array}\right] \begin{array}{l} \uparrow \\ \\ \\ \\ \\ \\ \\ \downarrow \end{array} \begin{array}{l} \\ \\ \\ \\ \text{“past”} \\ \\ \\ \\ \end{array}$$

$$\begin{array}{l} \\ \\ \\ \\ \\ \uparrow \\ \\ \downarrow \end{array} \begin{array}{l} \\ \\ \\ \\ \\ \text{“future”} \\ \\ \\ \end{array}$$

$$\triangleq \left[\begin{array}{c} U_{0|i} \\ \hdashline U_{i+1|2i-1} \end{array}\right] \triangleq \left[\begin{array}{c} U_p^+ \\ \hdashline U_f^- \end{array}\right].$$

- We can similarly define a block Hankel observation matrix:

$$Y_{0|2i-1} \triangleq \left[\begin{array}{ccccc} y[0] & y[1] & y[2] & \cdots & y[N-1] \\ y[1] & y[2] & y[3] & \cdots & y[N] \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ y[i-1] & y[i] & y[i+1] & \cdots & y[i+N-2] \\ \hdashline y[i] & y[i+1] & y[i+2] & \cdots & y[i+N-1] \\ y[i+1] & y[i+2] & y[i+3] & \cdots & y[i+N] \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ y[2i-1] & y[2i] & y[2i+1] & \cdots & y[2i+N-2] \end{array}\right] \begin{array}{l} \uparrow \\ \\ \\ \\ \text{“past”} \\ \\ \\ \downarrow \end{array} \begin{array}{l} \\ \\ \\ \\ \\ \uparrow \\ \\ \\ \end{array}$$

$$\begin{array}{l} \\ \\ \\ \\ \\ \text{“future”} \\ \\ \\ \end{array}$$

$$\triangleq \left[\begin{array}{c} Y_{0|i-1} \\ \hdashline Y_{i|2i-1} \end{array}\right] \triangleq \left[\begin{array}{c} Y_p \\ \hdashline Y_f \end{array}\right].$$

- This can also be split one block row beneath the "present" point:

$$Y_{0|2i-1} \triangleq \left[\begin{array}{ccccc} y[0] & y[1] & y[2] & \cdots & y[N-1] \\ y[1] & y[2] & y[3] & \cdots & y[N] \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ y[i-1] & y[i] & y[i+1] & \cdots & y[i+N-2] \\ y[i] & y[i+1] & y[i+2] & \cdots & y[i+N-1] \\ \hdashline y[i+1] & y[i+2] & y[i+3] & \cdots & y[i+N] \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ y[2i-1] & y[2i] & y[2i+1] & \cdots & y[2i+N-2] \end{array}\right] \begin{array}{l} \uparrow \\ \updownarrow \\ \downarrow \\ \uparrow \\ \updownarrow \\ \downarrow \end{array} \begin{array}{l} \\ \\ \text{“past”} \\ \\ \\ \text{“future”} \end{array}$$

$$\triangleq \left[\begin{array}{c} Y_{0|i} \\ \hdashline Y_{i+1|2i-1} \end{array}\right] \triangleq \left[\begin{array}{c} Y_p^+ \\ \hdashline Y_f^- \end{array}\right].$$

- Finally, we define block Hankel matrices consisting of inputs *and* outputs

$$W_{0|i-1} \triangleq \left[\begin{array}{c} U_{0|i-1} \\ Y_{0|i-1} \end{array}\right] = \left[\begin{array}{c} U_p \\ Y_p \end{array}\right] = W_p, \qquad \text{and} \qquad W_p^+ \triangleq \left[\begin{array}{c} U_p^+ \\ Y_p^+ \end{array}\right].$$

- We'll use these block matrices to form matrix input–output equations.
- To get there, we first use the basic state-space form repeatedly for advancing time:

$$\left[\begin{array}{c} y[k] \\ y[k+1] \\ \vdots \\ y[k+i-1] \end{array}\right] = \underbrace{\left[\begin{array}{c} C \\ CA \\ \vdots \\ CA^{i-1} \end{array}\right]}_{\mathcal{O}_i \in \mathbb{R}^{ip \times n}} x^d[k] + \underbrace{\left[\begin{array}{cccc} D & & & \\ CB & D & & \\ \vdots & \ddots & \ddots & \\ CA^{i-2}B & \cdots & CB & D \end{array}\right]}_{\Psi_i \in \mathbb{R}^{ip \times im}} \left[\begin{array}{c} u[k] \\ u[k+1] \\ \vdots \\ u[k+i-1] \end{array}\right].$$

- We can introduce some notation to simplify manipulation of the above equation:

$$y_i[k] = \begin{bmatrix} y[k] \\ y[k+1] \\ \vdots \\ y[k+i-1] \end{bmatrix} \in \mathbb{R}^{ip}, \quad u_i[k] = \begin{bmatrix} u[k] \\ u[k+1] \\ \vdots \\ u[k+i-1] \end{bmatrix} \in \mathbb{R}^{im}.$$

- The block control and observation matrices can now be expressed as

$$U_p = U_{0|i-1} = \begin{bmatrix} u_i[0] & u_i[1] & \cdots & u_i[N-1] \end{bmatrix}$$

$$Y_p = Y_{0|i-1} = \begin{bmatrix} y_i[0] & y_i[1] & \cdots & y_i[N-1] \end{bmatrix}.$$

- This notation leaves us with a relationship that is fundamental for ongoing state-space system-identification work:

$$y_i[k] = \mathcal{O}_i x^d[k] + \Psi_i u_i[k].$$

- Repeating this equation for $k = 0 \ldots N-1$ gives a matrix input–output equation based on past inputs and outputs

$$\boxed{Y_p = \mathcal{O}_i X_p^d + \Psi_i U_p,}$$

where X_p^d is a matrix of state vectors

$$X_p^d = \begin{bmatrix} x^d[0] & x^d[1] & \cdots & x^d[N-1] \end{bmatrix} \in \mathbb{R}^{n \times N}.$$

- Following the same argument, we can also show

$$\boxed{Y_f = \mathcal{O}_i X_f^d + \Psi_i U_f,}$$

where X_f^d is a matrix of state vectors

$$X_f^d = \begin{bmatrix} x^d[i] & x^d[i+1] & \cdots & x^d[i+N-1] \end{bmatrix} \in \mathbb{R}^{n \times N}.$$

- We'll need one more relationship connecting X_f^d to X_p^d.
- Recall from page 5–12 the general state-space relationship

$$x[n] = A^n x[0] + \underbrace{\begin{bmatrix} B & AB & A^2B & \cdots & A^{n-1}B \end{bmatrix}}_{\mathcal{C}} \begin{bmatrix} u[n-1] \\ \vdots \\ u[0] \end{bmatrix}.$$

- We now re-write this as (note that the $u[k]$ are in reverse order now)

$$x^d[k] = A^i x^d[k-i] + \underbrace{\begin{bmatrix} A^{i-1}B, & \cdots & A^2B, & AB, & B \end{bmatrix}}_{\mathcal{O}_i^d} \begin{bmatrix} u[k-i] \\ \vdots \\ u[k] \end{bmatrix},$$

  where $\mathcal{O}_i^d$ is the reversed extended controllability matrix. This gives

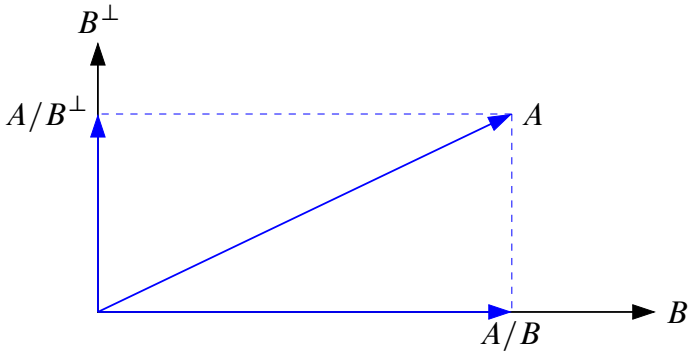
$$\boxed{X_f^d = A^i X_p^d + \mathcal{O}_i^d U_p.}$$

- These three boxed equations form the foundation for all of the subspace methods. We will find that we can determine the state sequence X_f^d directly from the given data $u[k]$ and $y[k]$ *without* knowledge of A, B, C, or D.
- We can also determine the extended observability matrix $\mathcal{O}_i$ directly from the given input–output data.
- From X_f^d and $\mathcal{O}_i$ there are a number of ways to find A, B, C, or D.
- Much like Ho–Kalman, we wish to factor the boxed equations;
  - However, now there is not a single matrix to be factored as in $\mathcal{H} = \mathcal{O}\mathcal{C}$, but a direct (*i.e.*, not orthogonal) sum of two matrix products such as $Y_f = \mathcal{O}_i X_f^d + \Psi_i U_f$.
  - We will use geometric projections to break the direct sum into its two constituent parts.

5.14: Geometric projections that we will need

Orthogonal Projections

- You are probably familiar with orthogonal projection, writing a matrix A in terms of an orthogonal basis set B and $B^\perp$.



- That is, $A = A/B + A/B^\perp$ where A/B is the orthogonal projection of the row space of A onto the row space of B, and $A/B^\perp = A - A/B$ is the orthogonal projection of the row space of A onto the null space of B.[3]
- Note that in the notation A/B, the boldface B indicates that the result of the operation lies in the row space of B.
- The projection is a linear relationship: $A/B = PB$.
- Note that $A/B^\perp$ is orthogonal to B, so $(A/B^\perp)B^T = 0$. That is,

$$(A - PB)B^T = 0$$

$$AB^T = PBB^T$$

$$P = AB^T(BB^T)^\dagger.$$

- So, $A/B = AB^T(BB^T)^\dagger B = A\Pi_B$ where $\Pi_B \triangleq B^T(BB^T)^\dagger B$.
- Similarly, we can define $\Pi_{B^\perp}$ to be the geometric operation that projects the row space of a matrix onto the null space of B.
  - That is, $A/B^\perp \triangleq A\Pi_{B^\perp}$ where $\Pi_{B^\perp} = I - \Pi_B$.

[3] Note: In this section, A, B, and C are generic. They are not the state-space matrices.

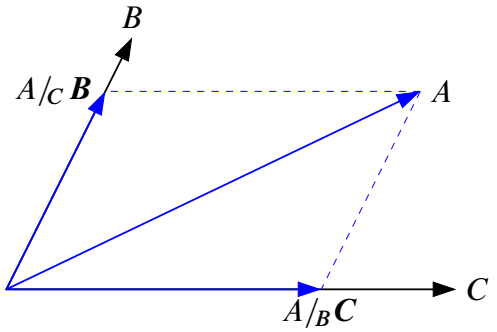
- These combinations decompose a matrix into two matrices of which the row spaces are orthogonal: $A = A\Pi_B + A\Pi_{B^\perp}$.
- Note that both geometrically and algebraically we have

$$B\Pi_B = BB^T(BB^T)^\dagger B = B$$

$$B\Pi_{B^\perp} = B(I - B^T(BB^T)^\dagger B) = 0.$$

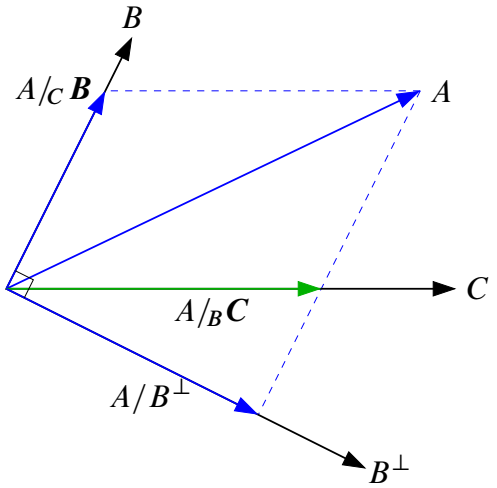
Oblique projections

- You may be less familiar with oblique projection (or parallel projection), writing a matrix A in terms of linear combinations of the rows of two non-orthogonal matrices B and C.



- We write $A/_B C$ as the oblique projection of the row space of A *along* the row space of B *onto* the row space of C.
- Oblique projections turn out to be very important for us, since, for example, $Y_f = \mathcal{O}_i X_f^d + \Psi_i U_f$ where we know Y_f and want to break it down into its oblique components $\mathcal{O}_i X_f^d$ and $\Psi_i U_f$.
- Note that $A/_B C$ is a linear combination of C ($A/_B C = PC$) and lies on the line joining A and $A/B^\perp$.

$$\begin{aligned} A/B^\perp &= [A/_B C]/B^\perp \\ &= P\,\underbrace{C((B^\perp)^T(B^\perp(B^\perp)^T)^\dagger B^\perp)}_{C/B^\perp} \\ P &= [A/B^\perp][C/B^\perp]^\dagger. \end{aligned}$$

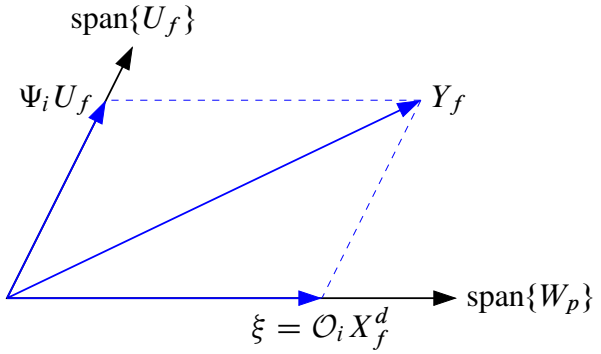


- Therefore, $A/_B C = [A/B^\perp][C/B^\perp]^\dagger C$.

- In long form, $A/_B\boldsymbol{C} = [A - AB^T(BB^T)^\dagger B][C - CB^T(BB^T)^\dagger B]^\dagger C$.
- Note that there are efficient and numerically stable ways to calculate this, which we won't focus on. Suffice it to say that we can calculate the oblique projections given only the matrices involved.
- Also note that when the row space of B is perpendicular to the row space of C (*i.e.*, $CB^T = 0$) or when $B = 0$ this relationship becomes $A/_B\boldsymbol{C} = A/C$.

5.15: Deterministic subspace system identification

- We are going to discover that we can recover the state sequence X_f^d and the extended observability matrix $\mathcal{O}_i$ from $u[k]$ and $y[k]$, without knowledge of A, B, C, and/or D.
- We then explore two ways to find A, B, C, and D from X_f^d and $\mathcal{O}_i$.
- We will assume that the input sequence $u[k] \in \mathbb{R}^m$ is persistently exciting of order $2i$, which means that $U_{0|2i-1}$ is full rank, which is $2mi$.
- We will also assume that the intersection of the row space of U_f and the row space of X_p^d is empty. This implies that there is no feedback.
- The basic approach is first to perform an oblique projection of Y_f along U_f onto W_p.
- The idea is that $Y_f = \mathcal{O}_i X_f^d + \Psi_i U_f$, and we're trying to recover the $\xi = \mathcal{O}_i X_f^d$ part of this.



- We'll then see how to split up ξ into its constituent $\mathcal{O}_i$ and X_f^d portions.
- Start with the relationship $Y_p = \mathcal{O}_i X_p^d + \Psi_i U_p$. Solving for X_p^d gives

$$X_p^d = \mathcal{O}_i^\dagger [Y_p - \Psi_i U_p].$$

- Substitute this into $X_f^d = A^i X_p^d + \Upsilon_i^d U_p$

$$\begin{aligned} X_f^d &= A^i \mathcal{O}_i^\dagger [Y_p - \Psi_i U_p] + \Upsilon_i^d U_p \\ &= [\Upsilon_i^d - A^i \mathcal{O}_i^\dagger \Psi_i] U_p + [A^i \mathcal{O}_i^\dagger] Y_p \\ &= \underbrace{\left[\Upsilon_i^d - A^i \mathcal{O}_i^\dagger \Psi_i \,\middle|\, A^i \mathcal{O}_i^\dagger \right]}_{L_p} W_p. \end{aligned}$$

- We can now write $Y_f = \mathcal{O}_i L_p W_p + \Psi_i U_f$. Multiply on the right by $\Pi_{U_f^\perp}$

$$Y_f \Pi_{U_f^\perp} = \mathcal{O}_i L_p W_p \Pi_{U_f^\perp} + \Psi_i \underbrace{U_f \Pi_{U_f^\perp}}_{0}.$$

- Multiply on the right by $[W_p \Pi_{U_f^\perp}]^\dagger W_p$

$$\underbrace{[Y_f \Pi_{U_f^\perp}][W_p \Pi_{U_f^\perp}]^\dagger W_p}_{\xi = Y_f / U_f \boldsymbol{W_p}} = \mathcal{O}_i L_p \underbrace{[W_p \Pi_{U_f^\perp}][W_p \Pi_{U_f^\perp}]^\dagger W_p}_{W_p}$$

$$\xi = \mathcal{O}_i L_p W_p = \mathcal{O}_i X_f^d.$$

  - Note that the second under-brace is not obviously equal to W_p since $W_p \Pi_{U_f^\perp}$ is rank deficient in purely deterministic systems. The text Appendix A.1 does a formal proof.

- The next step is to take ξ and split it up into $\mathcal{O}_i$ and X_f^d parts.

- To be very general, we first define weighting matrices $W_1 \in \mathbb{R}^{pi \times pi}$ and $W_2 \in \mathbb{R}^{N \times N}$ such that W_1 is full rank and $\text{rank}(W_p) = \text{rank}(W_p W_2)$.

  - The choice of the weighting matrices is fairly arbitrary.
  - At this point the optimal (if there is one) set is not known.
  - Different subspace ID methods use different W_1 and W_2.
  - The matrices essentially determine the state-space basis for which the model will be identified.

- We next perform the SVD of

$$W_1 \xi W_2 = \begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} \Sigma_1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} V_1^T \\ V_2^T \end{bmatrix}$$

$$= U_1 \Sigma_1 V_1^T.$$

- Note that $W_1 \xi W_2 = W_1 \mathcal{O}_i X_f^d W_2$ is the product of two matrices, $W_1 \mathcal{O}_i$ having n columns and $X_f^d W_2$ having n rows.
- Both matrices are full rank ($= n$), so the product is also full rank. Therefore, we determine that $n = \dim(\Sigma_1)$.
- We choose to set $W_1 \mathcal{O}_i = U_1 \Sigma_1^{1/2} T$ and $X_f^d W_2 = T^{-1} \Sigma_1^{1/2} V_1^T$, where T is an arbitrary similarity transformation matrix. Note that by choosing T, we can create any state space basis of equivalent systems.
- Thus, we have $\mathcal{O}_i = W_1^{-1} U_1 \Sigma_1^{1/2} T$ and $X_f^d = \mathcal{O}_i^\dagger \xi$.

REMARKS: Notice that

- $\text{rank}(Y_f /_{U_f} \boldsymbol{W_p}) = n$;
- The row space of $Y_f /_{U_f} \boldsymbol{W_p}$ is equal to the row space of X_f^d;
- The column space of $Y_f /_{U_f} \boldsymbol{W_p}$ is equal to the column space of $\mathcal{O}_i$.

- This is why these are "subspace methods." They retrieve system related matrices as subspaces of projected data matrices.

Computing the system matrices

- Now that we have found X_f^d and $\mathcal{O}_i$, we look at two different methods to find A, B, C, and D.

METHOD I: Recall that $X_f^d = \begin{bmatrix} x^d[i] & x^d[i+1] & \cdots & x^d[i+N-1] \end{bmatrix}$.

- Define (extracting data from X_f^d as needed),

$$\overline{X}_{i+1}^d = \begin{bmatrix} x^d[i+1] & x^d[i+2] & \cdots & x^d[i+N-1] \end{bmatrix}$$

$$\overline{X}_i^d = \begin{bmatrix} x^d[i] & x^d[i+1] & \cdots & x^d[i+N-2] \end{bmatrix}$$

$$\overline{U}_{i|i} = \begin{bmatrix} u[i] & u[i+1] & \cdots & u[i+N-2] \end{bmatrix}$$
$$\overline{Y}_{i|i} = \begin{bmatrix} y[i] & y[i+1] & \cdots & y[i+N-2] \end{bmatrix}.$$

- Then,

$$\begin{bmatrix} \overline{X}^d_{i+1} \\ \overline{Y}_{i|i} \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} \overline{X}^d_i \\ \overline{U}_{i|i} \end{bmatrix}$$
$$\begin{bmatrix} \hat{A} & \hat{B} \\ \hat{C} & \hat{D} \end{bmatrix} = \begin{bmatrix} \overline{X}^d_{i+1} \\ \overline{Y}_{i|i} \end{bmatrix} \begin{bmatrix} \overline{X}^d_i \\ \overline{U}_{i|i} \end{bmatrix}^\dagger.$$

- In the van Overshee toolbox, `det_stat.m` contains a MATLAB implementation of this algorithm.

METHOD II: There are two steps to METHOD II.

- First, we find A and C from $\mathcal{O}_i$ much the same as in Ho–Kalman. Then, we find B and D.
- To find A and C we rely on the shift structure of $\mathcal{O}_i$.
- Note that $\mathcal{O}_i^{\downarrow} A = \mathcal{O}_i^{\uparrow}$. Therefore, one way to find A is $A = (\mathcal{O}_i^{\downarrow})^{\dagger} \mathcal{O}_i^{\uparrow}$.
- C is set to the first block row of $\mathcal{O}_i$.
- To find B and D, recall that $Y_f = \mathcal{O}_i X_f^d + \Psi_i U_f$. Multiply on the left by $\mathcal{O}_i^{\perp}$, where $\mathcal{O}_i^{\perp}$ is a full row rank matrix satisfying $\mathcal{O}_i^{\perp} \mathcal{O}_i = 0$.
- This gives $\mathcal{O}_i^{\perp} Y_f = \mathcal{O}_i^{\perp} \Psi_i U_f$.
- With known matrices A, C, $\mathcal{O}_i^{\perp}$, U_f and Y_f, this equation is linear in B and D. Multiply on the right by $U_f^{\dagger}$

$$\mathcal{O}_i^{\perp} Y_f U_f^{\dagger} = \mathcal{O}_i^{\perp} \Psi_i.$$

- For simplicity of notation, denote the left-hand-side by $\mathcal{M}$ and $\mathcal{O}_i^{\perp}$ by $\mathcal{L}$. Then, we can write this equation as

$$\begin{bmatrix} \mathcal{M}_1 & \mathcal{M}_2 & \cdots & \mathcal{M}_i \end{bmatrix} = \begin{bmatrix} \mathcal{L}_1 & \mathcal{L}_2 & \cdots & \mathcal{L}_i \end{bmatrix} \times \begin{bmatrix} D & 0 & 0 & \cdots & 0 \\ CB & D & 0 & \cdots & 0 \\ CAB & CB & D & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ CA^{i-2}B & CA^{i-3}B & CA^{i-4}B & \cdots & D \end{bmatrix}.$$

- We can reorganize this as

$$\begin{bmatrix} \mathcal{M}_1 \\ \mathcal{M}_2 \\ \vdots \\ \mathcal{M}_i \end{bmatrix} = \begin{bmatrix} \mathcal{L}_1 & \mathcal{L}_2 & \cdots & \mathcal{L}_{i-1} & \mathcal{L}_i \\ \mathcal{L}_2 & \mathcal{L}_3 & \cdots & \mathcal{L}_i & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathcal{L}_i & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} I & 0 \\ 0 & \mathcal{O}_i^{\uparrow} \end{bmatrix} \begin{bmatrix} D \\ B \end{bmatrix}.$$

- We can now solve for B and D via least squares.
- In the van Overshee toolbox, `det_alt.m` contains a MATLAB implementation of this algorithm.

Summary of the deterministic subspace method

1. Form the data matrices W_p, U_f, and Y_f.
2. Compute the oblique projection $\xi = Y_f /_{U_f} \boldsymbol{W_p}$.
3. Compute the SVD: $U_1 \Sigma_1 V_1^T = W_1 \xi W_2$. (For now, simply use $W_1 = I$, $W_2 = I$, and $T = I$.) Set $n = \dim(\Sigma_1)$.
4. Set $\mathcal{O}_i = W_1^{-1} U_1 \Sigma_1^{1/2} T$ and $X_f^d = \mathcal{O}_i^{\dagger} \xi$.
5. Use either METHOD I or METHOD II to solve for A, B, C, and D.

5.16: Data matrices for continuous-time realization

- The course focus is on system identification for discrete-time models.
- However, subspace methods can also be used to identify continuous-time models, if special care is taken in the linear algebra.
- Here, we look at a method based on van Overschee and de Moor.[4]
- A continuous-time state-space model looks like:

$$\dot{x}(t) = Ax(t) + Bu(t)$$

$$y(t) = Cx(t) + Du(t),$$

  - Where $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{l\times n}$, and $D \in \mathbb{R}^{l\times m}$;
  - The continuous-time A and B matrices have different values from the corresponding discrete-time system's A and B matrices;
  - We can define controllability and observability matrices having the same algebraic form as for discrete-time, but with the continuous-time A and B matrices used instead.
- Define unit vector $E_1 = \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix}^T$, with similar E_2, etc.
- Then, in steady state, if the input to the system is $u(t) = E_1 e^{j\omega t}$, then $x_{ss}(t) = M_1 e^{j\phi_1} e^{j\omega t}$ and we have

$$\underbrace{(j\omega) M_1 e^{j\phi_1} e^{j\omega t}}_{\dot{x}_{ss}(t)} = A \underbrace{M_1 e^{j\phi_1} e^{j\omega t}}_{x_{ss}(t)} + BE_1 e^{j\omega t}.$$

- We can cancel $e^{j\omega t}$ from both sides and write

$$(j\omega) \underbrace{M_1 e^{j\phi_1}}_{X_1} = A \underbrace{M_1 e^{j\phi_1}}_{X_1} + BE_1.$$

[4] P. van Overschee and B. de Moor, "Continuous-time frequency domain subspace system identification," *Signal Processing*, vol. 52, 1996, pp. 179–194.

- Repeating with inputs $u(t) = E_2 e^{j\omega t}$, $u(t) = E_3 e^{j\omega t}$, etc, we can assemble the following form

$$(j\omega)\underbrace{\begin{bmatrix} X_1 & \cdots & X_m \end{bmatrix}}_{X(j\omega)} = A\underbrace{\begin{bmatrix} X_1 & \cdots & X_m \end{bmatrix}}_{X(j\omega)} + B\underbrace{\begin{bmatrix} E_1 & \cdots & E_m \end{bmatrix}}_{I_m}$$

$$(j\omega)X(j\omega) = AX(j\omega) + BI_m,$$

  where $X(j\omega) \in \mathbb{C}^{n\times m}$ is a matrix of constant complex coefficients.

- When the input is a frequency, the steady-state output is the frequency response:

$$y_{ss}(t) = Cx_{ss}(t) + Du_{ss}(t)$$

$$\underbrace{Y_1 e^{j\theta_1}}_{H_1} e^{j\omega t} = CM_1 e^{j\phi_1} e^{j\omega t} + DE_1 e^{j\omega t}$$

$$H_1 = CX_1 + DE_1.$$

- Collecting frequency responses resulting from every input together

$$\underbrace{\begin{bmatrix} H_1 & \cdots & H_m \end{bmatrix}}_{H(j\omega)} = C\underbrace{\begin{bmatrix} X_1 & \cdots & X_m \end{bmatrix}}_{X(j\omega)} + DI_m,$$

  where $H(j\omega) \in \mathbb{C}^{l\times m}$.

- Now, notice that differentiating the output gives $\dot{y}(t) = C\dot{x}(t) + D\dot{u}(t)$.
  - With a frequency input, in steady state $\dot{y}_{ss}(t) = (j\omega)y_{ss}(t)$ and $\dot{u}_{ss}(t) = (j\omega)u_{ss}(t)$; furthermore, the state equation gives $\dot{x}_{ss}(t) = Ax_{ss}(t) + Bu_{ss}(t)$.

- For multiple differentiations of the output, we can then write

$$(j\omega)^{i-1}H(j\omega) = CA^{i-1}X(j\omega) + CA^{i-2}BI_m + (j\omega)CA^{i-3}BI_m$$

$$+ \cdots + (j\omega)^{i-1}DI_m.$$

- Collecting multiple frequency responses and differentiated frequency responses in one complex data matrix, we can write

$$H^c = \begin{bmatrix} H(j\omega_1) & H(j\omega_2) & \cdots & H(j\omega_N) \\ (j\omega_1)H(j\omega_1) & (j\omega_2)H(j\omega_2) & \cdots & (j\omega_N)H(j\omega_N) \\ \vdots & & \ddots & \\ (j\omega_1)^{i-1}H(j\omega_1) & (j\omega_2)^{i-1}H(j\omega_2) & \cdots & (j\omega_N)^{i-1}H(j\omega_N) \end{bmatrix}.$$

- We also define

$$I^c = \begin{bmatrix} I_m & I_m & \cdots & I_m \\ (j\omega_1)I_m & (j\omega_2)I_m & \cdots & (j\omega_N)I_m \\ \vdots & & \ddots & \\ (j\omega_1)^{i-1}I_m & (j\omega_2)^{i-1}I_m & \cdots & (j\omega_N)^{i-1}I_m \end{bmatrix}$$

$$X^c = \begin{bmatrix} X(j\omega_1) & X(j\omega_2) & \cdots & X(j\omega_N) \end{bmatrix}.$$

- Then, we can further write the input–output equations

$$H^c = \mathcal{O}_i X^c + \Psi_i I^c,$$

where, as before

$$\mathcal{O}_i = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{i-1} \end{bmatrix} \quad \text{and} \quad \Psi_i = \begin{bmatrix} D & & & \\ CB & D & & \\ \vdots & \ddots & \ddots & \\ CA^{i-2}B & \cdots & CB & D \end{bmatrix}.$$

- It can be more convenient to work with real data matrices rather than complex data matrices, so define

$$H^r = \begin{bmatrix} \mathbb{R}(H^c) & \mathbb{I}(H^c) \end{bmatrix} \qquad X^r = \begin{bmatrix} \mathbb{R}(X^c) & \mathbb{I}(X^c) \end{bmatrix}$$

$$I^r = \begin{bmatrix} \mathbb{R}(I^c) & \mathbb{I}(I^c) \end{bmatrix}.$$

  - Then, we also have $H^r = \mathcal{O}_i X^r + \Psi_i I^r$.

5.17: Simple continuous-time realization

- If we conduct frequency-response experiments on a system using multiple input frequencies, it is a simple matter to construct the data matrices H^r and I^r.
- Then, if we could somehow solve $H^r = \mathcal{O}_i X^r + \Psi_i I^r$ for $\mathcal{O}_i$, we could find A and C from $\mathcal{O}_i$.
- Finally, we could use least squares to determine B and D from

$$H(j\omega) = \begin{bmatrix} C((j\omega)I - A)^{-1} & I \end{bmatrix} \begin{bmatrix} B \\ D \end{bmatrix}.$$

  - The trick then is in isolating $\mathcal{O}_i$ somehow.
- Note first what happens to the product $\mathcal{O}_i X M$ upon a state transformation, where M is some arbitrary matrix.
- Define $x = Tz$, where T is an invertible similarity transformation matrix. The state equation will change as:

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ T\dot{z}(t) &= ATz(t) + Bu(t) \\ \dot{z}(t) &= \underbrace{T^{-1}AT}_{\bar{A}} z(t) + \underbrace{T^{-1}B}_{\bar{B}} u(t). \end{aligned}$$

- The output equation will change as

$$\begin{aligned} y(t) &= Cx(t) + Du(t) \\ &= \underbrace{CT}_{\bar{C}} z(t) + Du(t). \end{aligned}$$

- The observability matrix for the z-state system will be

$$\bar{\mathcal{O}}_i = \begin{bmatrix} \bar{C} \\ \bar{C}\bar{A} \\ \vdots \\ \bar{C}\bar{A}^{n-1} \end{bmatrix} = \begin{bmatrix} CT \\ CTT^{-1}AT \\ \vdots \\ CT\underbrace{T^{-1}AT}_{n-1 \text{ times}} \end{bmatrix} = \mathcal{O}_i T.$$

- The state data matrix for the original system was

$$X^c = \begin{bmatrix} X(j\omega_1) & X(j\omega_2) & \cdots & X(j\omega_N) \end{bmatrix},$$

where

$$X(j\omega) = ((j\omega)I - A)^{-1} BI_m.$$

- For the transformed system,

$$\begin{aligned} Z(j\omega) &= \left((j\omega)I - \bar{A}\right)^{-1} \bar{B} I_m \\ &= \left((j\omega)I - T^{-1}AT\right)^{-1} T^{-1}BI_m \\ &= T^{-1}((j\omega)I - A)^{-1} TT^{-1}BI_m \\ &= T^{-1}X(j\omega), \end{aligned}$$

so $Z^c = T^{-1}X^c$.

- Putting these results together,

$$\mathcal{O}_i X(j\omega)M = \bar{\mathcal{O}}_i Z(j\omega)M.$$

- In words, no matter how we factor $\mathcal{O}_i X(j\omega)M$ into two pieces—the $\mathcal{O}_i$ piece and the $X(j\omega)M$ piece—we end up with a correct state-space realization of the original system, with a transformed state.
- Back to the original problem: We would like to solve $H^r = \mathcal{O}_i X^r + \Psi_i I^r$ for $\mathcal{O}_i$.

- We do so by first projecting H^r onto the orthogonal complement of I^r.

$$H^r/I_r^\perp = \mathcal{O}_i X^r/I_r^\perp + \Psi_i \underbrace{I_r/I_r^\perp}_{0}$$

$$= \mathcal{O}_i X^r \underbrace{I_r^T(I_r I_r^T)^\dagger I_r}_{\text{“}M\text{”}}.$$

- So, if we take the SVD of $H^r/I_r^\perp$, and factor into the $\mathcal{O}_i$ part and the remainder part, we can extract A and C from $\mathcal{O}_i$.
- The “simple” continuous-time algorithm is then:

1. Construct H^r and I_r from the given frequencies ω_k and frequency-response points $H(j\omega_k)$.
2. Compute $H^r/I_r^\perp$ and its singular-value decomposition

$$H^r/I_r^\perp = \begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} \Sigma_1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} V_1^T \\ V_2^T \end{bmatrix}.$$

3. Determine the order from number of significant singular values in Σ_1.
4. Determine $\mathcal{O}_i = U_1 \Sigma_1^{1/2}$, which is one possible estimate for the extended-observability matrix.
5. Determine A and C as

$$C = \mathcal{O}_i(1:l,:)$$

$$A = \left(\mathcal{O}_i^\downarrow\right)^\dagger \mathcal{O}_i^\uparrow.$$

6. Determine B and D through the least-squares solution to the linear equations

$$\begin{bmatrix} H(j\omega_1) \\ \vdots \\ H(j\omega_N) \end{bmatrix} = \begin{bmatrix} C(j\omega_1 I_n - A)^{-1} & I_l \\ \vdots & \vdots \\ C(j\omega_N I_n - A)^{-1} & I_l \end{bmatrix} \begin{bmatrix} B \\ D \end{bmatrix}.$$

5.18: Numeric problems with simple algorithm

- For low-order systems and small data matrices, the simple algorithm works well. However, for high-order systems and large data matrices, it utterly fails. Why?
- An investigation shows that the H^r and I_r matrices become "poorly conditioned" because they are by nature composed via block-Vandermonde matrices.
  - While we never directly perform a matrix inverse, the matrix operations we do perform are sensitive to conditioning.
  - Floating-point arithmetic cannot retain enough significant figures, and round-off error leads to poor results.
- In scalar form, a Vandermonde matrix has the form

$$\mathcal{V} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_q \\ x_1^2 & x_2^2 & \cdots & x_q^2 \\ \vdots & \vdots & & \vdots \\ x_1^{p-1} & x_2^{p-1} & \cdots & x_q^{p-1} \end{bmatrix}.$$

- The determinant of the Vandermonde matrix has a special form:

$$\det(\mathcal{V}) = \prod_{1 \le \rho < \mu \le p} (x_\rho - x_\mu).$$

- A simple example illustrates this notation. For an arbitrary four-parameter Vandermonde matrix:

$$\mathcal{V} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ a & b & c & d \\ a^2 & b^2 & c^2 & d^2 \\ a^3 & b^3 & c^3 & d^3 \end{bmatrix}$$

$$\det(\mathcal{V}) = (a-b)(a-c)(a-d)(b-c)(b-d)(c-d).$$

- If any of $\{a, b, c, d\}$ are nearly the same, the determinant is nearly zero, and the matrix is poorly conditioned.
- In our problem (notation not precise),

$$H^c = \mathcal{V}(\omega_k)\text{blkdiag}(H)$$

$$I^c = \mathcal{V}(\omega_k).$$

- If frequencies ω_k are small and/or close to each other, $\mathcal{V}$ is poorly conditioned.
- Further, multiplying by $\text{blkdiag}(H)$ requires the frequency-response matrix itself to be well conditioned, which it will not be if some frequency-response values are near zero.
- In either case, differences between elements become small, and the matrices are poorly conditioned.

Toward a better conditioning

- We would like to somehow manipulate $H^r/I_r^\perp$ so that it is better conditioned.
- The issue is not so much with the *columns* of $H^r/I_r^\perp$, which could be scaled for better conditioning via multiplying this quantity on the right; it is rather with the *rows* of $H^r/I_r^\perp$, which must be improved by multiplying on the left.
- That is, we seek to find some L_H that improves the situation, where

$$L_H H^r = L_H \mathcal{O}_i X^r + L_H I_r$$

$$L_H H^r/I_r^\perp = L_H \mathcal{O}_i X^r I_r^T (I_r I_r^T)^\dagger I_r.$$

- There are two problems with this:
  1. L_H itself will be poorly conditioned;
  2. Since L_H multiplies on the left, it messes up the nice factoring of $H^r/I_r^\perp \neq L_H H^r/I_r^\perp$, and we have to account for the multiplication on the left when we are attempting to extract $\mathcal{O}_i$ from the result.
- We avoid the first problem by never explicitly computing L_H; rather, we compute the well-conditioned results $L_H H^r$ and $L_H I_r$ directly.
- We solve the second problem by finding a method to untangle the effect of multiplying by L_H in the first place.

Forsythe recursion

- Ideally, $L_H H^r$ and $L_H I_r$ would both be block orthonormal matrices as a result of the scaling.
  - We can accomplish this for systems having a single output, and can get close for multi-output systems.
- The basic algorithm sets up new data matrices as follows:
  1. Initialize

$$R_0 = \begin{bmatrix} H(j\omega_1) & H(j\omega_2) & \cdots & H(j\omega_N) \end{bmatrix}$$

$$Z_0 = R_0 R_0^*$$

$$R_1 = R_0 D_\omega$$

$$Z_1 = R_1 R_1^*,$$

where $D_\omega = \mathrm{diag}\Big[\big((j\omega_1)\,(j\omega_2)\,\cdots\,(j\omega_N)\big) \otimes \underbrace{\begin{pmatrix} 1 & 1 & \cdots & 1 \end{pmatrix}}_{\in \mathbb{R}^{1\times m}}\Big]$.

2. Recursively compute, for $k = 2$ to $i - 1$,

$$R_k = R_{k-1} D_\omega + Z_{k-1} Z_{k-2}^{-1} R_{k-2}$$

$$Z_k = R_k R_k^*.$$

3. Then, define

$$H_F^c = \begin{bmatrix} (\sqrt{Z_0})^{-1} R_0 \\ (\sqrt{Z_1})^{-1} R_1 \\ \vdots \\ (\sqrt{Z_{i-1}})^{-1} R_{i-1} \end{bmatrix},$$

where the matrix square root is defined as a square lower-triangular matrix such that $(\sqrt{Z_k})(\sqrt{Z_k})^* = Z_k$, which can be computed in MATLAB as `zk=chol(Zk,'lower')`.

4. Similarly, compute I_F^c by initializing the recursion with

$$R_0 = \begin{bmatrix} I_m & I_m & \cdots & I_m \end{bmatrix}.$$

- In the next sections, we first show that H_F^c and I_F^c are better conditioned; we then see how to extract A and C from the resulting data matrices.

5.19: Proof of better conditioning

- Ideally, H_F^r would be orthonormal. This implies the need for

$$H_F^r (H_F^r)^T = \mathbb{R}\left[H_F^c (H_F^c)^*\right] = I$$

or

$$\mathbb{R}\left[\left(z_r^{-1} R_r\right)\left(z_s^{-1} R_s\right)^*\right] = \delta_{rs},$$

where z_k is the square-root factor of Z_k such that $z_k z_k^* = Z_k$.

- To show normality for $r = s$,

$$\mathbb{R}\left[\left(z_r^{-1} R_r\right)\left(z_r^{-1} R_r\right)^*\right] = \mathbb{R}\Big[z_r^{-1} \underbrace{R_r R_r^*}_{Z_r} (z_r^{-1})^*\Big]$$

$$= \mathbb{R}\left[z_r^{-1} z_r z_r^* (z_r^{-1})^*\right] = I\,.$$

- To check for orthogonality, we use proof by induction.

Initialization: $\mathbb{R}[\left(z_r^{-1} R_r\right)\left(z_0^{-1} R_0\right)^*] = 0$ *for* $r = 1, \ldots, i-1$

- Let R_k^v be the vth column of R_k. Then, for $r = 1$,

$$\mathbb{R}[\left(z_1^{-1} R_1\right)\left(z_0^{-1} R_0\right)^*] = \mathbb{R}[z_1^{-1} \underbrace{R_0 D_\omega}_{R_1} R_0^* (z_0^{-1})^*].$$

- For a single output, z_0 and z_1 are real scalars. Therefore, we need only check that $\mathbb{R}[R_0 D_\omega R_0^*] = 0$.

$$\mathbb{R}[R_0 D_\omega R_0^*] = \mathbb{R}\left[j \sum_{v=1}^{mN} \omega_k \left|R_0^v\right|^2\right] = 0.$$

- For multiple outputs, z_0 and z_1 are complex, and we do *not* have $\mathbb{R}[\left(z_1^{-1} R_1\right)\left(z_0^{-1} R_0\right)^*] = 0$. However, the conditioning is still better than without our enhanced method.

- For $r = 2$, we have

$$\mathbb{R}[z_2^{-1} R_2 R_0^*(z_0^{-1})^*] = \mathbb{R}[z_2^{-1} \underbrace{\left(R_1 D_\omega + Z_1 Z_0^{-1} R_0\right)}_{R_2} R_0^*(z_0^{-1})^*]$$

$$= \mathbb{R}[z_2^{-1} \left(R_1 D_\omega R_0^* + Z_1 \underbrace{Z_0^{-1} R_0 R_0^*}_{I} \right) (z_0^{-1})^*]$$

$$= \mathbb{R}[z_2^{-1} \left(- \underbrace{R_1 (\overbrace{R_0 D_\omega}^{R_1})^*}_{Z_1} + Z_1 \right) (z_0^{-1})^*] = 0.$$

Induction

- We now assume that it is given that $\mathbb{R}[z_r^{-1} R_r R_s^*(z_s^{-1})^*] = 0$ for $s = 0, \ldots, k-1$ and $r = s+1, \ldots, i-1$.
  - Note that if $\mathbb{R}[M] = 0$, then $\mathbb{R}[M^*] = 0$ too, implying that $\mathbb{R}[z_r^{-1} R_r R_s^*(z_s^{-1})^*] = 0$ for $r = 0, \ldots, k-1$ and $s = r+1, \ldots, i-1$.
- We seek to prove that $\mathbb{R}[z_r^{-1} R_r R_k^*(z_k^{-1})^*] = 0$ for $r = k+1, \ldots, i-1$.
- For $r = k+1$,

$$\mathbb{R}[z_{k+1}^{-1} R_{k+1} R_k^*(z_k^{-1})^*] = \mathbb{R}[z_{k+1}^{-1} \left(R_k D_\omega + Z_k Z_{k-1}^{-1} R_{k-1}\right) R_k^*(z_k^{-1})^*]$$

$$= \mathbb{R}[z_{k+1}^{-1} \left(R_k D_\omega R_k^* + Z_k Z_{k-1}^{-1} R_{k-1} R_k^*\right) (z_k^{-1})^*].$$

- For a single output, z_{k+1}, z_k, and z_{k-1} are real scalars. Also, $\mathbb{R}[R_{k-1} R_k^*] = 0$ by assumption. Therefore, we need only check that $\mathbb{R}[R_k D_\omega R_k^*] = 0$.

$$\mathbb{R}[R_k D_\omega R_k^*] = \mathbb{R}\left[j \sum_{v=1}^{mN} \omega_k \, |R_k^v|^2 \right] = 0.$$

- For multiple outputs, z_{k+1}, z_k, and z_{k-1} are complex, and we do *not* have $\mathbb{R}[z_{k+1}^{-1} R_{k+1} R_k^*(z_k^{-1})^*] = 0$. However, the conditioning is still better than without our enhanced method.
- For $r = k+2$,

$$
\begin{aligned}
&\mathbb{R}[z_{k+2}^{-1} R_{k+2} R_k^*(z_k^{-1})^*] \\
&= \mathbb{R}[z_{k+2}^{-1} \left(R_{k+1} D_\omega + Z_{k+1} Z_k^{-1} R_k\right) R_k^*(z_k^{-1})^*] \\
&= \mathbb{R}[z_{k+2}^{-1} \left(R_{k+1} D_\omega R_k^* + Z_{k+1} \underbrace{Z_k^{-1} R_k R_k^*}_{I} \right) (z_k^{-1})^*] \\
&= \mathbb{R}[z_{k+2}^{-1} \left(-R_{k+1}(R_k D_\omega)^* + R_{k+1} R_{k+1}^*\right) (z_k^{-1})^*] \\
&= \mathbb{R}[z_{k+2}^{-1} \left(-R_{k+1}(R_{k+1} - Z_k Z_{k-1}^{-1} R_{k-1})^* + R_{k+1} R_{k+1}^*\right) (z_k^{-1})^*] \\
&= \mathbb{R}[z_{k+2}^{-1} \left(R_{k+1} R_{k-1}^* \left(Z_k Z_{k-1}^{-1}\right)^* \right) (z_k^{-1})^*].
\end{aligned}
$$

- For a single output, $\mathbb{R}[R_{k+1} R_{k-1}^*] = 0$, and we have the desired result. Otherwise, this is not zero, but we again appear to have better conditioning than without the recursions.
- So, we have shown that
  - For single-output systems, H_F^r is orthonormal, and so has condition number 1. This is as good as it gets.
  - For multi-output systems, H_F^r is normal, but not orthogonal. The condition number of H_F^r is generally much better than for H^r.
- We can repeat the same steps for I_F^r with the same fundamental results, except that the z_k are completely real, meaning that orthonormality is true for I_F^r even for multi-output systems.

5.20: Improved continuous-time system ID

- We have shown that H_F^c and I_F^c are better conditioned than H^c and I^c. But, we don't yet know how to perform system ID with the revised matrices.
- Here, we show how the recursions that produce the modified data matrices can be unraveled to allow us to solve for A and C.
- Let h_k and h_k^F be the kth block rows of H^c and H_F^c. Similarly, let γ_k and γ_k^F be the kth block rows of $\mathcal{O}_i$ and $\mathcal{O}_i^F$.
- Then, from the untransformed system, we know

$$h_k = h_{k-1} D_\omega$$

$$\gamma_k = \gamma_{k-1} A.$$

- The general recursions give us

$$R_k = R_{k-1} D_\omega + Z_{k-1} Z_{k-2}^{-1} R_{k-2}$$

$$\underbrace{z_k^{-1} R_k}_{h_k^F} = z_k^{-1} z_{k-1} \underbrace{z_{k-1}^{-1} R_{k-1}}_{h_{k-1}^F} D_\omega + z_k^{-1} Z_{k-1} Z_{k-2}^{-1} z_{k-2} \underbrace{z_{k-2}^{-1} R_{k-2}}_{h_{k-2}^F}$$

$$h_k^F = z_k^{-1} z_{k-1} h_{k-1}^F D_\omega + z_k^{-1} Z_{k-1} Z_{k-2}^{-1} z_{k-2} h_{k-2}^F.$$

- Also, as $L_H H^c$ and $L_H \mathcal{O}_i X^c M$ can be written as the same linear combinations of $H^c$and $\mathcal{O}_i X^c M$, we have

$$h_k^F = \sum_{v=0}^{k} \alpha_v h_v, \qquad \text{and} \qquad \gamma_k^F = \sum_{v=0}^{k} \alpha_v \gamma_v,$$

  for the same blending matrices α_v.
- Working from h_k^F,

$$h_k^F = z_k^{-1} z_{k-1} h_{k-1}^F D_\omega + z_k^{-1} Z_{k-1} \underbrace{Z_{k-2}^{-1} z_{k-2}}_{\left(z_{k-2}^{-1}\right)^*} h_{k-2}^F$$

$$= z_k^{-1} z_{k-1} \left(\sum_{v=0}^{k-1} \alpha_v h_v D_\omega \right) + z_k^{-1} Z_{k-1} \left(z_{k-2}^{-1}\right)^* h_{k-2}^F$$

$$= z_k^{-1} z_{k-1} \left(\sum_{v=0}^{k-1} \alpha_v h_{v+1} \right) + z_k^{-1} Z_{k-1} \left(z_{k-2}^{-1}\right)^* h_{k-2}^F.$$

- The same basic relationship must hold for γ_k^F as well, so

$$\gamma_k^F = z_k^{-1} z_{k-1} \left(\sum_{v=0}^{k-1} \alpha_v \gamma_{v+1} \right) + z_k^{-1} Z_{k-1} \left(z_{k-2}^{-1}\right)^* \gamma_{k-2}^F$$

$$= z_k^{-1} z_{k-1} \left(\sum_{v=0}^{k-1} \alpha_v \gamma_v \right) A + z_k^{-1} Z_{k-1} \left(z_{k-2}^{-1}\right)^* \gamma_{k-2}^F$$

$$= z_k^{-1} z_{k-1} \gamma_{k-1}^F A + z_k^{-1} Z_{k-1} \left(z_{k-2}^{-1}\right)^* \gamma_{k-2}^F.$$

- Rearranging,

$$z_k^{-1} z_{k-1} \gamma_{k-1}^F A = \gamma_k^F - z_k^{-1} Z_{k-1} \left(z_{k-2}^{-1}\right)^* \gamma_{k-2}^F.$$

- Writing out a few terms in matrix form

$$\begin{bmatrix} z_2^{-1} z_1 \gamma_1^F \\ z_3^{-1} z_2 \gamma_2^F \\ z_4^{-1} z_3 \gamma_3^F \\ \vdots \end{bmatrix} A = \begin{bmatrix} \gamma_2^F \\ \gamma_3^F \\ \gamma_4^F \\ \vdots \end{bmatrix} - \begin{bmatrix} z_2^{-1} Z_1 (z_0^{-1})^* \gamma_0^F \\ z_3^{-1} Z_2 (z_1^{-1})^* \gamma_1^F \\ z_4^{-1} Z_3 (z_2^{-1})^* \gamma_2^F \\ \vdots \end{bmatrix}.$$

- If we define new scaling matrices

$$D_1 = \text{blkdiag}\left[\begin{array}{cccc} z_2^{-1} z_1 & z_3^{-1} z_2 & \cdots & z_{i-1}^{-1} z_{i-2} \end{array} \right]$$

$$D_2 = \text{blkdiag}\left[\begin{array}{cccc} z_2^{-1} Z_1 (z_0^{-1})^* & z_3^{-1} Z_2 (z_1^{-1})^* & \cdots & z_{i-1}^{-1} Z_{i-2} (z_{i-3}^{-1})^* \end{array} \right],$$

then we can write

$$D_1 \underbrace{\begin{bmatrix} \gamma_1^F \\ \gamma_2^F \\ \vdots \\ \gamma_{i-2}^F \end{bmatrix}}_{(\mathcal{O}_i^F)^{\updownarrow}} A = \underbrace{\begin{bmatrix} \gamma_2^F \\ \gamma_3^F \\ \vdots \\ \gamma_{i-1}^F \end{bmatrix}}_{(\mathcal{O}_i^F)^{\uparrow\uparrow}} - D_2 \underbrace{\begin{bmatrix} \gamma_0^F \\ \gamma_1^F \\ \vdots \\ \gamma_{i-3}^F \end{bmatrix}}_{(\mathcal{O}_i^F)^{\downarrow\downarrow}}.$$

- So, we can solve for C as the top block row of $\mathcal{O}_i^F$ and for A as

$$A = \left(D_1 \left(\mathcal{O}_i^F\right)^{\updownarrow}\right)^{\dagger} \left(\left(\mathcal{O}_i^F\right)^{\uparrow\uparrow} - D_2 \left(\mathcal{O}_i^F\right)^{\downarrow\downarrow}\right).$$

- We can solve for B and D using least squares, as before.
- The "improved" continuous-time algorithm is then:

1. Construct H_F and I_F and the matrices Z_0 through Z_{i-1}.
2. Compute $H_F/I_F^{\perp} = H_F - H_F I_F^T \left(\underbrace{I_F I_F^T}_{I_{mi}}\right)^{-1} I_F = H_F - H_F I_F^T I_F$ and its singular-value decomposition

$$H_F/I_F^{\perp} = \begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} \Sigma_1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} V_1^T \\ V_2^T \end{bmatrix}.$$

3. Determine the order from number of significant singular values in Σ_1.
4. Determine $\mathcal{O}_i^F = U_1 \Sigma_1^{1/2}$, which is one possible estimate for the extended-observability matrix.
5. Determine A and C as

$$C = \mathcal{O}_i^F(1:l,:)$$

$$A = \left(D_1 \left(\mathcal{O}_i^F\right)^{\updownarrow}\right)^{\dagger} \left(\left(\mathcal{O}_i^F\right)^{\uparrow\uparrow} - D_2 \left(\mathcal{O}_i^F\right)^{\downarrow\downarrow}\right).$$

6. Determine B and D through the least-squares solution to the linear equations

$$\begin{bmatrix} H(j\omega_1) \\ \vdots \\ H(j\omega_N) \end{bmatrix} = \begin{bmatrix} C(j\omega_1 I_n - A)^{-1} & I_l \\ \vdots & \vdots \\ C(j\omega_N I_n - A)^{-1} & I_l \end{bmatrix} \begin{bmatrix} B \\ D \end{bmatrix}.$$

- Our experience is that this improved algorithm works very well.
  - Results are somewhat sensitive to the frequencies ω_k—optimal selection depends on dynamics of the system.
  - The original paper gives some extra tuning parameters that could be explored, which place different emphases on different frequency-response values.

Where from here

- We've made a good start at looking at state-space methods.
- However, none of these methods explicitly takes measurement noise or (correlated) disturbance into account.
- Our next step is to look at some state-space identification methods that do.