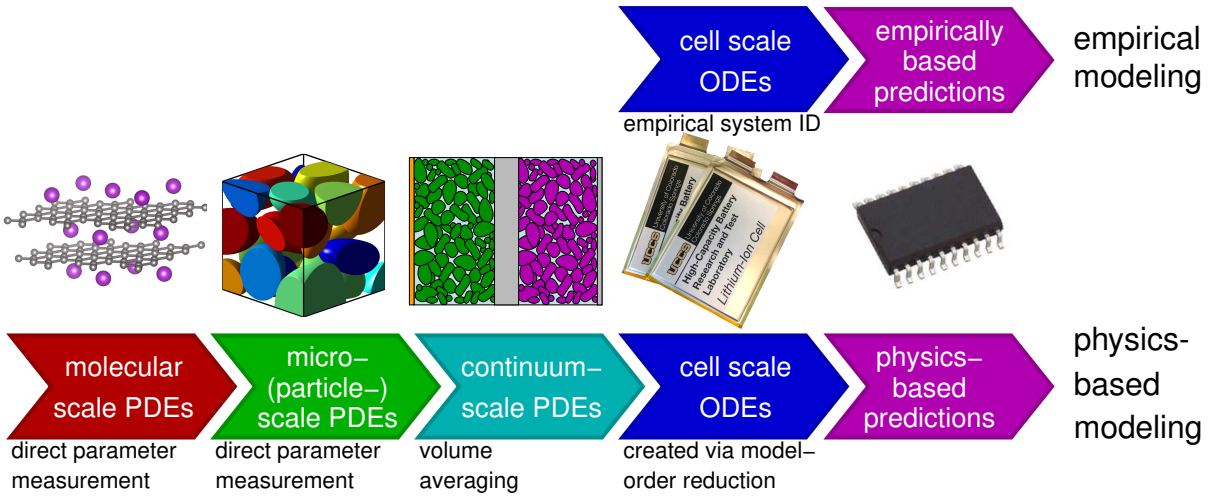
# State-Space Models and the Discrete-Time Realization Algorithm

## 5.1: Introduction to state-space models

- The coupled PDEs derived in earlier chapters of notes are too complex to be used in real-time applications.
  - They are "infinite dimensional." For every point in time $t$, there are an infinite number of $x$- and $r$- dimension variables to solve for.
  - *i.e.*, $c_s(x, r, t)$, $\bar{c}_e(x, t)$, $\bar{\phi}_s(x, t)$, $\bar{\phi}_e(x, t)$, for a pseudo-two dimensional model.




- We desire to create cell-scale ODEs that retain, as much as possible, the fidelity of the continuum-scale PDEs, but which reduce their order from infinite order to some (small) finite order.
  - The result is a small coupled set of ODEs, which can be simulated very easily and quickly.
- In this chapter, we introduce "state-space" models, which is the final form of the reduced-order models we will develop.

- We then preview the approach to generate the state-space models from the PDEs of the variables of interest:
  - We start by generating transfer functions for each PDE;
  - We then use the "discrete-time realization algorithm" to convert transfer functions to state-space form.

## 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.
- Discrete-time LTI state-space models have the following form:

$$\boldsymbol{x}[k+1] = \boldsymbol{A}\boldsymbol{x}[k] + \boldsymbol{B}\boldsymbol{u}[k]$$

$$\boldsymbol{y}[k] = \boldsymbol{C}\boldsymbol{x}[k] + \boldsymbol{D}\boldsymbol{u}[k],$$

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

- Different systems have different $n$, $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$, and $\boldsymbol{D}$.
- 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

$$\boldsymbol{x}[k+1] = \boldsymbol{A}\boldsymbol{x}[k] + \boldsymbol{B}\boldsymbol{u}[k]$$

$$\boldsymbol{y}[k] = \boldsymbol{C}\boldsymbol{x}[k] + \boldsymbol{D}\boldsymbol{u}[k].$$

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

$$z\boldsymbol{X}(z) - z\boldsymbol{x}[0] = \boldsymbol{A}\boldsymbol{X}(z) + \boldsymbol{B}\boldsymbol{U}(z)$$

$$\boldsymbol{Y}(z) = \boldsymbol{C}\boldsymbol{X}(z) + \boldsymbol{D}\boldsymbol{U}(z),$$

  or

$$(z\boldsymbol{I} - \boldsymbol{A})\boldsymbol{X}(z) = \boldsymbol{B}\boldsymbol{U}(z) + z\boldsymbol{x}[0]$$

$$\boldsymbol{X}(z) = (z\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{B}\boldsymbol{U}(z) + (z\boldsymbol{I} - \boldsymbol{A})^{-1}z\boldsymbol{x}[0].$$

- This gives,

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

- So,

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

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

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

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

## Transformation

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

$$\boldsymbol{x}[k+1] = \boldsymbol{A}\boldsymbol{x}[k] + \boldsymbol{B}\boldsymbol{u}[k]$$

$$\boldsymbol{y}[k] = \boldsymbol{C}\boldsymbol{x}[k] + \boldsymbol{D}\boldsymbol{u}[k],$$

  where we let $\boldsymbol{x}[k] = \boldsymbol{T}\boldsymbol{w}[k]$, where $\boldsymbol{T}$ is an invertible (similarity) transformation matrix. Then,

$$(\boldsymbol{T}\boldsymbol{w}[k+1]) = \boldsymbol{A}\,(\boldsymbol{T}\boldsymbol{w}[k]) + \boldsymbol{B}\boldsymbol{u}[k]$$

$$\boldsymbol{y}[k] = \boldsymbol{C}\,(\boldsymbol{T}\boldsymbol{w}[k]) + \boldsymbol{D}\boldsymbol{u}[k].$$

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

$$\boldsymbol{w}[k+1] = \underbrace{\boldsymbol{T}^{-1}\boldsymbol{A}\boldsymbol{T}}_{\bar{\boldsymbol{A}}}\,\boldsymbol{w}[k] + \underbrace{\boldsymbol{T}^{-1}\boldsymbol{B}}_{\bar{\boldsymbol{B}}}\,\boldsymbol{u}[k]$$

$$\boldsymbol{y}[k] = \underbrace{\boldsymbol{C}\boldsymbol{T}}_{\bar{\boldsymbol{C}}}\,\boldsymbol{w}[k] + \underbrace{\boldsymbol{D}}_{\bar{\boldsymbol{D}}}\,\boldsymbol{u}[k]$$

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

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

- To show that $\boldsymbol{H}_1(z) = \boldsymbol{H}_2(z)$,

$$\boldsymbol{H}_1(z) = \boldsymbol{C}(z\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{B} + \boldsymbol{D}$$

$$= \boldsymbol{C}\boldsymbol{T}\boldsymbol{T}^{-1}(z\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{T}\boldsymbol{T}^{-1}\boldsymbol{B} + \boldsymbol{D}$$

$$= (\boldsymbol{CT})[\boldsymbol{T}^{-1}(z\boldsymbol{I} - \boldsymbol{A})\boldsymbol{T}]^{-1}(\boldsymbol{T}^{-1}\boldsymbol{B}) + \boldsymbol{D}$$

$$= \bar{\boldsymbol{C}}(z\boldsymbol{I} - \bar{\boldsymbol{A}})^{-1}\bar{\boldsymbol{B}} + \bar{\boldsymbol{D}} = \boldsymbol{H}_2(z).$$

- Transfer function not changed by similarity transform

**CONCLUSION:** Can arrive at state-space representations having identical input-output relationship but different $(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D})$ matrices.

**EXAMPLE**: Consider transforming the system

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

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

- Note that multiplying on the right by $\boldsymbol{T}$ 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:

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

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

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

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

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

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

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

  which was the transfer function we started with before transformation.

## 5.3: Discrete-time Markov parameters

- It 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, $\boldsymbol{x}[0] = 0$ when finding a unit-pulse response).
- We find that

$$\begin{aligned}
\boldsymbol{y}[0] &= \boldsymbol{C}\boldsymbol{x}[0] + \boldsymbol{D}u[0] = \boldsymbol{D}, & \boldsymbol{x}[1] &= \boldsymbol{B} \\
\boldsymbol{y}[1] &= \boldsymbol{C}\boldsymbol{x}[1] + \boldsymbol{D}u[1] = \boldsymbol{C}\boldsymbol{B}, & \boldsymbol{x}[2] &= \boldsymbol{A}\boldsymbol{B} \\
\boldsymbol{y}[2] &= \boldsymbol{C}\boldsymbol{x}[2] + \boldsymbol{D}u[2] = \boldsymbol{C}\boldsymbol{A}\boldsymbol{B}, & \boldsymbol{x}[3] &= \boldsymbol{A}^2\boldsymbol{B} \\
&\vdots & &\vdots \\
\boldsymbol{y}[k] &= \boldsymbol{C}\boldsymbol{A}^{k-1}\boldsymbol{B}, & k &\geq 1.
\end{aligned}$$

- These unit-pulse-response values, $\{\boldsymbol{D}, \boldsymbol{C}\boldsymbol{B}, \boldsymbol{C}\boldsymbol{A}\boldsymbol{B}, \boldsymbol{C}\boldsymbol{A}^2\boldsymbol{B}, \boldsymbol{C}\boldsymbol{A}^3\boldsymbol{B}, \ldots\}$ are called the Markov parameters of the system.
  - This turns out to be of critical importance to realizing our transfer functions, as we will see.
- Specifically, we define the Markov parameters to be:

$$\boldsymbol{g}_k = \begin{cases} \boldsymbol{D}, & k = 0; \\ \boldsymbol{C}\boldsymbol{A}^{k-1}\boldsymbol{B}, & k > 0. \end{cases}$$

**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 $i$th entry (row) of each Markov parameter is computed as the unit-pulse response from the input to the $i$th 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 $j$th entry (column) of each Markov parameter is computed via the unit-pulse response from the $j$th 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 $j$th input to the $i$th output.
  - Equivalently, the $j$th column of each Markov parameter is vector (as in the SIMO case) which is computed via the unit-pulse response from the $j$th input to the vector output.

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

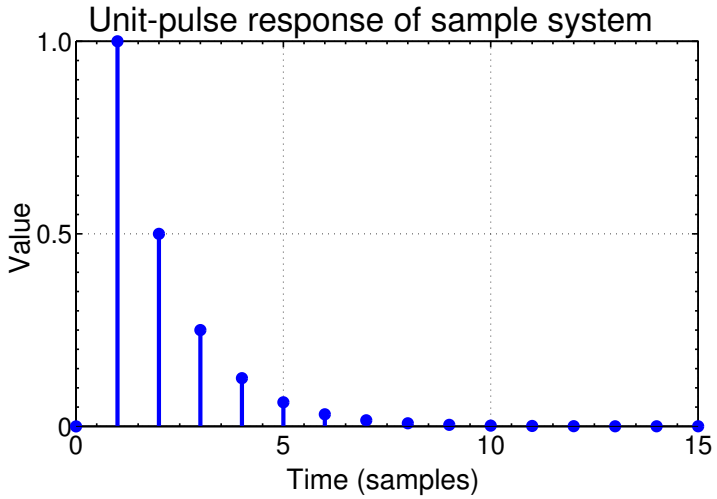
$$\boldsymbol{A} = \begin{bmatrix} 0.5 & 0 \\ 0 & 1 \end{bmatrix}, \qquad \boldsymbol{B} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \qquad \boldsymbol{C} = \begin{bmatrix} 1 & -1 \end{bmatrix}, \qquad \boldsymbol{D} = 0.$$

- The Markov parameters are given by

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

- 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');
```




## Before proceeding...

- We have now quickly previewed state-space models, with the claim that there will be a method to represent our battery models in that particular form.
- We now begin to investigate that claim—the first step is to create transfer-function models for the variables of interest.
- In this chapter, we look at representing $c_s$ as a transfer function; in the next chapter we look at the remainder of the model equations.
  - Note that in chapter 3 we used symbols without an over-line to indicate point-wise values for variables of interest: *i.e.*, $c_s$, $c_e$, $\phi_s$, $\phi_e$.
  - In chapter 4 we used symbols with an over-line to indicate volume average versions of these point-wise variables: *i.e.*, $\bar{c}_e$, $\bar{\phi}_s$, and $\bar{\phi}_e$.
  - We now drop the over-line notation, because otherwise the equations get so highly decorated that they are impossible to parse. We are still talking about the volume-average quantities of chapter 4.

## 5.4: Equations describing the solid dynamics

### Finding the transfer function $\widetilde{C}_{s,e}(s)/J(s)$

- To find the transfer function for $c_s$, we follow the approach by Jacobsen and West[1]
- We start with the underlying partial-differential equation,

$$\frac{\partial c_s(r,t)}{\partial t} = \frac{1}{r^2}\frac{\partial}{\partial r}\left(D_s r^2 \frac{\partial c_s(r,t)}{\partial r}\right),$$

  with standard boundary conditions,

$$D_s\frac{\partial c_s(0,t)}{\partial r} = 0, \qquad \text{and} \qquad D_s\frac{\partial c_s(R_s,t)}{\partial r} = -j(t), \qquad t \geq 0,$$

  and with initial equilibrium concentration,

$$c_s(r,0) = c_{s,0}, \qquad 0 \leq r \leq R_s.$$

- Note that we run into problems solving this PDE directly if $c_{s,0} \neq 0$.
- So, to enforce a homogeneous PDE in later steps, we define $\tilde{c}_s(r,t) = c_s(r,t) - c_{s,0}$. The "tilde" notation denotes the difference between an absolute quantity and its equilibrium set-point.
- If we assume constant $D_s$, the differential equations become:

$$\frac{\partial \tilde{c}_s(r,t)}{\partial t} = \frac{D_s}{r^2}\frac{\partial}{\partial r}\left(r^2 \frac{\partial \tilde{c}_s(r,t)}{\partial r}\right),$$

  with boundary conditions,

$$D_s\frac{\partial \tilde{c}_s(0,t)}{\partial r} = 0, \qquad \text{and} \qquad D_s\frac{\partial \tilde{c}_s(R_s,t)}{\partial r} = -j(t), \qquad t \geq 0,$$

  and with initial equilibrium concentration,

$$\tilde{c}_s(r,0) = 0, \qquad 0 \leq r \leq R_s.$$

[1] Jacobsen, T., and West, K., "Diffusion Impedance in Planar, Cylindrical and Spherical Symmetry," *Electrochimica Acta*, 40(2), 1995, pp. 255–62.

- We continue by taking the Laplace transform of the PDE:

$$s\widetilde{C}_s(r,s) - \tilde{c}_0 = \frac{D_s}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial}{\partial r}\widetilde{C}_s(r,s)\right)$$

$$s\widetilde{C}_s(r,s) = \frac{D_s}{r^2}\left(2r\frac{\partial\widetilde{C}_s(r,s)}{\partial r} + r^2\frac{\partial^2\widetilde{C}_s(r,s)}{\partial r^2}\right).$$

- This is a 2nd-order ordinary differential equation in $r$, which may be written

$$\boxed{\frac{\partial^2\widetilde{C}_s(r,s)}{\partial r^2} + \frac{2}{r}\frac{\partial\widetilde{C}_s(r,s)}{\partial r} - \frac{s}{D_s}\widetilde{C}_s(r,s) = 0.}$$

- This homogeneous differential equation has a solution of the form

$$\widetilde{C}_s(r,s) = \frac{A}{r}\exp\left(r\sqrt{\frac{s}{D_s}}\right) + \frac{B}{r}\exp\left(-r\sqrt{\frac{s}{D_s}}\right)$$

$$= \frac{A}{r}\exp(\beta(r)) + \frac{B}{r}\exp(-\beta(r)),$$

  where we define $\beta(r) = r\sqrt{s/D_s}$. We note that $\beta(r)$ is also a function of $s$, but we omit this dependence in the notation for compactness.

- The constants $A$ and $B$ are chosen to satisfy the boundary conditions.

- Consider first the outer boundary condition at $r = R_s$, which is

$$D_s\left.\frac{\partial\tilde{c}_s(r,t)}{\partial r}\right|_{r=R_s} = -j(t).$$

- The equivalent Laplace-domain boundary condition is

$$D_s\left.\frac{\partial\widetilde{C}_s(r,s)}{\partial r}\right|_{r=R_s} = -J(s).$$

- To substitute this in, we will need to compute $\partial\widetilde{C}_s(r,s)/\partial r$

$$\frac{\partial\widetilde{C}_s(r,s)}{\partial r} = \frac{A\sqrt{\frac{s}{D_s}}r\exp(\beta(r)) - B\exp(-\beta(r))}{r^2}$$

$$-\frac{A\exp(\beta(r)) + B\sqrt{\frac{s}{D_s}}r\exp(-\beta(r))}{r^2}$$
$$= \frac{A(\beta(r)-1)\exp(\beta(r)) - B(1+\beta(r))\exp(-\beta(r))}{r^2}.$$

- We substitute $r = R_s$ and the boundary condition

$$\left.\frac{\partial \widetilde{C}_s(r,s)}{\partial r}\right|_{r=R_s} = \frac{A(\beta(R_s)-1)\exp(\beta(R_s)) - B(1+\beta(R_s))\exp(-\beta(R_s))}{R_s^2}$$
$$-\frac{J(s)}{D_s} = \frac{A(\beta(R_s)-1)\exp(\beta(R_s)) - B(1+\beta(R_s))\exp(-\beta(R_s))}{R_s^2}.$$

- This gives us an expression for $J(s)$,

$$J(s) = -\frac{D_s}{R_s^2}\left(A(\beta(R_s)-1)\exp(\beta(R_s)) - B(1+\beta(R_s))\exp(-\beta(R_s))\right).$$

- If we immediately substitute the second boundary condition at $r = 0$, we run into some divide-by-zero issues.

- So, instead, we substitute $r = r_\delta$, which we think of as a very small value. We will then later take the limit as $r_\delta \to 0$.

$$0 = \frac{A(\beta(r_\delta)-1)\exp(\beta(r_\delta)) - B(1+\beta(r_\delta))\exp(-\beta(r_\delta))}{r_\delta^2}.$$

- This allows us to write

$$\frac{A(\beta(r_\delta)-1)\exp(\beta(r_\delta))}{r_\delta^2} = \frac{B(1+\beta(r_\delta))\exp(-\beta(r_\delta))}{r_\delta^2}$$
$$A = B\frac{(1+\beta(r_\delta))\exp(-\beta(r_\delta))}{(\beta(r_\delta)-1)\exp(\beta(r_\delta))}.$$

- We now take the limit as $r_\delta \to 0$, and find that $A = -B$.

- We are now ready to construct the transfer function $\widetilde{C}_s(s,r)/J(s)$

$$\frac{\widetilde{C}_s(r,s)}{J(s)} = \frac{-R_s^2}{D_s r}\left[\frac{A\exp(\beta(r))+B\exp(-\beta(r))}{A(\beta(R_s)-1)\exp(\beta(R_s))-B(1+\beta(R_s))\exp(-\beta(R_s))}\right]$$
$$= \frac{-R_s^2}{D_s r}\left[\frac{A}{-A}\right]\left[\frac{\exp(\beta(r))-\exp(-\beta(r))}{(1-\beta(R_s))\exp(\beta(R_s))-(1+\beta(R_s))\exp(-\beta(R_s))}\right]$$
$$= \frac{R_s^2}{D_s r}\left[\frac{\exp(\beta(r))-\exp(-\beta(r))}{(1-\beta(R_s))\exp(\beta(R_s))-(1+\beta(R_s))\exp(-\beta(R_s))}\right].$$

- This expression can be used to determine the lithium concentration anywhere within the particle.
- However, we are most interested in determining the concentration at the *surface* of the particle, where $r = R_s$. So, we substitute $r = R_s$

$$\frac{\widetilde{C}_{s,e}(s)}{J(s)} = \frac{R_s}{D_s}\left[\frac{\exp(\beta(R_s)) - \exp(-\beta(R_s))}{(1-\beta(R_s))\exp(\beta(R_s)) - (1+\beta(R_s))\exp(-\beta(R_s))}\right].$$

- To compact the notation yet again, write $\beta(R_s)$ as simply $\beta$,

$$\frac{\widetilde{C}_{s,e}(s)}{J(s)} = \frac{R_s}{D_s}\left[\frac{\exp(\beta) - \exp(-\beta)}{(1-\beta)\exp(\beta) - (1+\beta)\exp(-\beta)}\right]$$
$$= \frac{R_s}{D_s}\left[\frac{\exp(\beta) - \exp(-\beta)}{\exp(\beta) - \exp(-\beta) - \beta\left[\exp(\beta) + \exp(-\beta)\right]}\right]$$
$$= \frac{R_s}{D_s}\left[\frac{\frac{\exp(\beta)-\exp(-\beta)}{\exp(\beta)+\exp(-\beta)}}{\frac{\exp(\beta)-\exp(-\beta)}{\exp(\beta)+\exp(-\beta)} - \beta}\right]$$
$$= \frac{R_s}{D_s}\left[\frac{\tanh(\beta)}{\tanh(\beta) - \beta}\right] = \frac{R_s}{D_s}\left[\frac{1}{1-\beta\coth(\beta)}\right].$$

- To recap to this point, re-expanding notation, where $\beta(s,r) = r\sqrt{s/D_s}$,

$$\boxed{\widetilde{C}_{s,e}(s) = \frac{R_s}{D_s}\left[\frac{1}{1-\beta(s,R_s)\coth\left(\beta(s,R_s)\right)}\right]J(s).}$$

## 5.5: Removing the integrator pole

- While not immediately obvious by looking at the transfer function, it turns out that $\widetilde{C}_{s,e}(s)/J(s)$ is unstable: There is a pole at $s = 0$.
  - This is intuitively clear, however, because we know that a step input will result in ever-increasing concentration.
  - This will be important when we look at how to convert the transfer function to a state-space model.
- To make a stable transfer function, define $\Delta\widetilde{C}_{s,e}(s) = \widetilde{C}_{s,e}(s) - \widetilde{C}_{s,\text{avg}}(s)$, where $\widetilde{C}_{s,\text{avg}}(s)$ is the bulk (average) concentration in the solid, less $c_{s,0}$.
- Note that we can write $\tilde{c}_{s,\text{avg}}(t_1)$ for some arbitrary point in time $t_1$ as

$$\tilde{c}_{s,\text{avg}}(t_1) = \int_0^{t_1} \frac{\text{Influx of Li, [mol s}^{-1}\text{]}}{\text{Volume of particle [m}^3\text{]}}\,\mathrm{d}t.$$

- Note two things:
  - The volume of a sphere of radius $R_s$ is $\frac{4}{3}\pi R_s^3$ [m$^3$];
  - The influx of lithium is $-j(t)$ [mol m$^{-2}$ s$^{-1}$], occurring over the surface area $4\pi R_s^2$ [m$^2$].
- This gives

$$\begin{aligned}\tilde{c}_{s,\text{avg}}(t_1) &= \int_0^{t_1} \frac{-j(t)\cdot 4\pi R_s^2}{\frac{4}{3}\pi R_s^3}\,\mathrm{d}t \\ &= -\frac{3}{R_s}\int_0^{t_1} j(t)\,\mathrm{d}t \\ \frac{\mathrm{d}}{\mathrm{d}t}\tilde{c}_{s,\text{avg}}(t) &= -\frac{3}{R_s}j(t).\end{aligned}$$

- Note that this result is perfectly general. We made no assumptions on how the lithium concentration is distributed inside the particle.
- Taking Laplace transforms, we find:

$$\frac{\widetilde{C}_{s,\text{avg}}(s)}{J(s)} = -\frac{3}{R_s}\frac{1}{s}.$$

- Therefore,

$$\begin{aligned}
\frac{\Delta\widetilde{C}_{s,e}(s)}{J(s)} &= \frac{\widetilde{C}_{s,e}(s)}{J(s)} - \frac{\widetilde{C}_{s,\text{avg}}(s)}{J(s)} \\
&= \frac{R_s}{D_s}\left[\frac{\tanh(\beta)}{\tanh(\beta)-\beta}\right] + \frac{3}{R_s s} \\
&= \frac{R_s}{D_s}\left[\frac{\tanh(\beta) + \frac{3D_s}{sR_s^2}\left(\tanh(\beta)-\beta\right)}{\tanh(\beta)-\beta}\right] \\
&= \frac{R_s}{D_s}\left[\frac{\tanh(\beta) + \frac{3}{\beta^2}\left(\tanh(\beta)-\beta\right)}{\tanh(\beta)-\beta}\right] \\
&= \frac{R_s}{D_s}\left[\frac{\beta^2\tanh(\beta) + 3\left(\tanh(\beta)-\beta\right)}{\beta^2\left(\tanh(\beta)-\beta\right)}\right] \\
&= \frac{R_s}{D_s}\left[\frac{(\beta^2+3)\tanh(\beta) - 3\beta}{\beta^2\left(\tanh(\beta)-\beta\right)}\right].
\end{aligned}$$

## State-space realization problem

- It turns out that for this specific case, we can find all the poles and zeros using a simple numeric method, and use that information to make a discrete-time state-space model.
- For the transfer functions we develop in the next chapter, however, this cannot be done.
- So, we must turn to alternative implementation approaches.

- One method is to use nonlinear optimization to select poles and residues to attempt to match the frequency response of the transfer functions.
- This is fraught with problems.
- We next introduce another approach, which directly gives us a discrete-time state-space approximate model of our transfer functions.

- This system-identification problem for state-space systems is sometimes called the “realization problem.”
  - That is, we wish to find a realization (a set of $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$, and $\boldsymbol{D}$ matrices) that describe a system’s dynamics.

## 5.6: State-space realization problem: Ho–Kalman method

- For now, we assume that we are able to find the Markov parameters of our transfer functions.

**PROBLEM:** Given a system's Markov parameters, find the system dimension $n$ and $(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D})$, up to similarity transforms.

- One of the first (maybe *the* first) state-space realization methods was introduced by Ho and Kalman.[2]
- It is key to the discrete-time realization algorithm we will develop.
- Notice that something curious happens when we multiply the following matrices together:

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

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

  - For reasons beyond the scope of our discussion here, $\mathcal{O}$ is called the "observability matrix" and $\mathcal{C}$ is called the "controllability matrix."

[2] 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.

- 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 $k > 2n-1$)

$$\mathcal{H} = \mathcal{O}\mathcal{C} = \begin{bmatrix} \boldsymbol{g}_1 & \boldsymbol{g}_2 & \cdots & \boldsymbol{g}_n \\ \boldsymbol{g}_2 & \boldsymbol{g}_3 & & \\ \vdots & & \ddots & \vdots \\ \boldsymbol{g}_n & & \cdots & \boldsymbol{g}_{2n-1} \end{bmatrix}.$$

- Ho–Kalman assumes that we know the Markov parameters.
  - Knowledge of $\boldsymbol{g}_0$ gives us $\boldsymbol{D}$ directly.
  - Knowledge of the rest of the Markov parameters will ultimately result in $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{C}$.
- 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 $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{C}$.

**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 $\mathcal{H}$?

**PRELIMINARY ANSWER:** The rank of $\mathcal{H}$ is equal to $n$. Keep adding data until the rank doesn't increase.

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

**ANSWER:** $\boldsymbol{C}$ is extracted as the first block row of $\mathcal{O}$; $\boldsymbol{B}$ is extracted as the first block column of $\mathcal{C}$. We'll see how to get $\boldsymbol{A}$ 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 $\mathcal{OC} = \mathcal{H}$ are okay.

- To see this latter point, consider what happens to $\mathcal{O}$ and $\mathcal{C}$ 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

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

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

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

- Therefore, $\overline{\mathcal{O}}\,\overline{\mathcal{C}} = (\mathcal{O}T)(T^{-1}\mathcal{C}) = \mathcal{OC}$
  - 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 $\mathcal{O} = \boldsymbol{I}$, and then $\mathcal{C} = \mathcal{H}$. This will result in an $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{C}$ that are in "observability canonical form." (cf. ECE5520)
- Or, we could choose to let $\mathcal{C} = \boldsymbol{I}$, and then $\mathcal{O} = \mathcal{H}$. This will result in an $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{C}$ that are in "controllability canonical form."

**ISSUE IV:** Is there a "best" way to factor $\mathcal{H}$? Yes... enter the SVD.

## 5.7: Singular value decomposition

**FACT:** Any rectangular matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, where $\mathrm{rank}(\boldsymbol{A}) = r$, can be factored into the form:

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T.$$

- $\boldsymbol{U} = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_r] \in \mathbb{R}^{m\times r}$, and $\boldsymbol{U}^T\boldsymbol{U} = \boldsymbol{I}$, and $\boldsymbol{u}_i$ are the left or output singular vectors of $\boldsymbol{A}$.
- $\boldsymbol{V} = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_r] \in \mathbb{R}^{n\times r}$, and $\boldsymbol{V}^T\boldsymbol{V} = \boldsymbol{I}$, and $\boldsymbol{v}_i$ are the right or input singular vectors of $\boldsymbol{A}$.
- $\boldsymbol{\Sigma} = \mathrm{diag}(\sigma_1, \ldots, \sigma_r)$ where $\sigma_1 \geq \cdots \geq \sigma_r > 0$, and $\sigma_i$ are the (nonzero) singular values of $\boldsymbol{A}$.

- The above is called a compact SVD. Most often, we compute a full SVD, where
  - $\boldsymbol{U} = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_m] \in \mathbb{R}^{m\times m}$, and $\boldsymbol{U}^T\boldsymbol{U} = \boldsymbol{I}$,
  - $\boldsymbol{V} = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n] \in \mathbb{R}^{n\times n}$, and $\boldsymbol{V}^T\boldsymbol{V} = \boldsymbol{I}$,
  - The matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{m\times n}$ is "diagonal"

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

  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:

$$\boldsymbol{A} = \left[\begin{array}{c:c} \boldsymbol{U}_1 & \boldsymbol{U}_2 \end{array}\right] \left[\begin{array}{c:c} \boldsymbol{\Sigma}_1 & \boldsymbol{0}_{r\times(n-r)} \\ \hdashline \boldsymbol{0}_{(m-r)\times r} & \boldsymbol{0}_{(m-r)\times(n-r)} \end{array}\right] \left[\begin{array}{c} \boldsymbol{V}_1^T \\ \hdashline \boldsymbol{V}_2^T \end{array}\right],$$

where $\boldsymbol{A} = \boldsymbol{U}_1\boldsymbol{\Sigma}_1\boldsymbol{V}_1^T$ is the compact SVD.

- Note that the singular values are related to matrix norm. In particular, $\|\boldsymbol{A}\| = \sigma_1$.
- Can view operation $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$ as $\boldsymbol{y} = (\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T)\boldsymbol{x}$, decomposing the operation into
  - Computing coefficients of $\boldsymbol{x}$ along the input directions $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_r$ (rotating by $\boldsymbol{V}^T$)
    - $\boldsymbol{v}_1$ is the most sensitive (highest gain) input direction
  - Scaling the coefficients by $\sigma_i$ (dilation)
  - Reconstituting along output directions $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_r$.
    - $\boldsymbol{u}_1$ is the highest gain output direction. $\boldsymbol{A}\boldsymbol{v}_1 = \sigma_1\boldsymbol{u}_1$.
- SVD gives a picture of gain as a function of input/output directions.

**EXAMPLE:** Consider $\boldsymbol{A} \in \mathbb{R}^{4\times4}$ with $\boldsymbol{\Sigma} = \mathrm{diag}(10, 7, 0.1, 0.05)$.

- Input components along directions $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ are amplified (by about 10) and come out mostly along the plane spanned by $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$.
- Input components along directions $\boldsymbol{v}_3$, $\boldsymbol{v}_4$ are attenuated (by about 10).
- $\|\boldsymbol{A}\boldsymbol{x}\| \,/\, \|\boldsymbol{x}\|$ can range between 10 and 0.05; $\boldsymbol{A}$ is nonsingular.
- For some applications you might say that $\boldsymbol{A}$ is *effectively* rank 2 (this will be important for us later).

## Low-rank approximations

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

$$\left\| \boldsymbol{A} - \hat{\boldsymbol{A}} \right\| = \left\| \sum_{i=p+1}^{r} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T \right\| = \sigma_{p+1}$$

  because $\sigma_{p+1}$ is the maximum remaining singular value.

**INTERPRETATION:** SVD dyads $\boldsymbol{u}_i \boldsymbol{v}_i^T$ are ranked in order of 'importance'; take $p$ of them to get a rank $p$ approximant.

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

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

## 5.8: 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 $\mathcal{H}_\infty$:

$$\mathcal{H}_\infty = \begin{bmatrix} \boldsymbol{g}_1 & \boldsymbol{g}_2 & \boldsymbol{g}_3 & \boldsymbol{g}_4 & \cdots \\ \boldsymbol{g}_2 & \boldsymbol{g}_3 & \boldsymbol{g}_4 & \boldsymbol{g}_5 & \cdots \\ \boldsymbol{g}_3 & \boldsymbol{g}_4 & \boldsymbol{g}_5 & \boldsymbol{g}_6 & \cdots \\ \boldsymbol{g}_4 & \boldsymbol{g}_5 & \boldsymbol{g}_6 & \boldsymbol{g}_7 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

  where the entries $\boldsymbol{g}_k$ 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 $\mathcal{H}$

$$\mathcal{H}_{k,l} = \begin{bmatrix} \boldsymbol{g}_1 & \boldsymbol{g}_2 & \boldsymbol{g}_3 & \cdots & \boldsymbol{g}_l \\ \boldsymbol{g}_2 & \boldsymbol{g}_3 & \boldsymbol{g}_4 & \cdots & \boldsymbol{g}_{l+1} \\ \boldsymbol{g}_3 & \boldsymbol{g}_4 & \boldsymbol{g}_5 & \cdots & \boldsymbol{g}_{l+2} \\ \vdots & \vdots & \vdots & & \vdots \\ \boldsymbol{g}_k & \boldsymbol{g}_{k+1} & \boldsymbol{g}_{k+2} & \cdots & \boldsymbol{g}_{k+l-1} \end{bmatrix}.$$

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

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

- 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 $\mathcal{H}_{k,l}$

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

- 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".
  - $\boldsymbol{T}$ must be invertible, but selection of $\boldsymbol{T}$ is otherwise arbitrary. Usually use $\boldsymbol{T} = \boldsymbol{I}$.
- How to decompose further into $(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C})$ to solve "ISSUE II"?
- Note the shift property of a Hankel matrix. If we shift $\mathcal{H}$ up by one block row, we get $\mathcal{H}^{\uparrow}_{k+1,l} = \mathcal{O}_k \boldsymbol{A} \mathcal{C}_l$.

$$\mathcal{H}^{\uparrow}_{k+1,l} = \begin{bmatrix} \boldsymbol{g}_2 & \boldsymbol{g}_3 & \boldsymbol{g}_4 & \cdots & \boldsymbol{g}_{l+1} \\ \boldsymbol{g}_3 & \boldsymbol{g}_4 & \boldsymbol{g}_5 & \cdots & \boldsymbol{g}_{l+2} \\ \vdots & \vdots & \vdots & & \vdots \\ \boldsymbol{g}_k & \boldsymbol{g}_{k+1} & \boldsymbol{g}_{k+2} & \cdots & \boldsymbol{g}_{k+l-1} \\ \boldsymbol{g}_{k+1} & \boldsymbol{g}_{k+2} & \boldsymbol{g}_{k+3} & \cdots & \boldsymbol{g}_{k+l} \end{bmatrix}$$

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

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

- Using the pseudo-inverse to solve for $A$ gives $A = \mathcal{O}_k^{\dagger}\mathcal{H}_{k+1,l}^{\uparrow}\mathcal{C}_l^{\dagger}$.
- In MATLAB, we can compute either

```
Ahat = pinv(Ok)*HankelUp*pinv(Cl);
```

  or

```
Ahat = (Ok\HankelUp)/Cl;
```

- 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 $D = g_0$.

## 5.9: Ho–Kalman summary and example

**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 $\boldsymbol{T} = \boldsymbol{I}_n$

**STEP IV:** Identify the system matrices $(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C})$. $\boldsymbol{D} = \boldsymbol{g}_0$.

**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:

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

- 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 $\boldsymbol{U}$ and $\boldsymbol{V}$

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

- Compute the extended observability and controllability matrices

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

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

```
% 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 $(\widehat{\boldsymbol{A}}, \widehat{\boldsymbol{B}}, \widehat{\boldsymbol{C}})$ up to similarity transform

$$\widehat{\boldsymbol{A}} = \mathcal{O}_k^{\dagger}\mathcal{H}_{k+1,l}^{\uparrow}\mathcal{C}_l^{\dagger} = \begin{bmatrix} 1.6180 & 0 \\ 0 & -0.6180 \end{bmatrix}$$

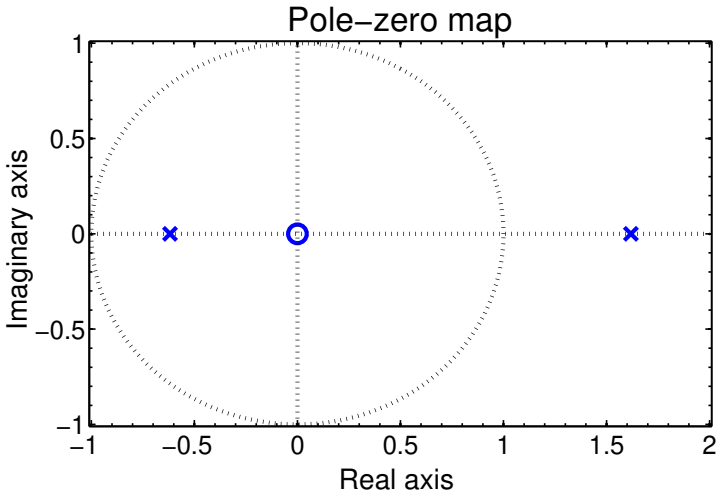
$$\widehat{\boldsymbol{B}} = \mathcal{C}_l(1:n, 1:m) = \mathcal{C}_l(1:2, 1) = \begin{bmatrix} -0.8057 \\ 0.5257 \end{bmatrix}$$

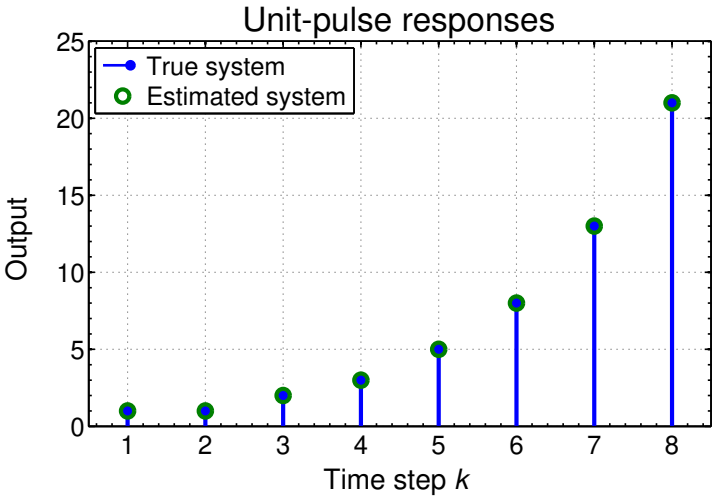
$$\widehat{\boldsymbol{C}} = \mathcal{O}_k(1:p, 1:n) = \mathcal{O}_k(1, 1:2) = \begin{bmatrix} -0.8057 & 0.5257 \end{bmatrix}$$

$$\widehat{\boldsymbol{D}} = g_0 = 0.$$

```
% 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 $\boldsymbol{A} = \boldsymbol{A}^T$ and that $\boldsymbol{B} = \boldsymbol{C}^T$ 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:** From whence come the $\boldsymbol{g}_k$?

- This is key to making the DRA work.

## 5.10: Discrete-Time Realization Algorithm (DRA)

- Given a continuous-time transfer function in the Laplace domain, $\boldsymbol{H}(s) = \boldsymbol{Y}(s)/\boldsymbol{U}(s)$, and a sampling period, $T_s$, we want to derive a reduced-order discrete-time state-space realization of the form

$$\boldsymbol{x}[k+1] = \boldsymbol{A}\boldsymbol{x}[k] + \boldsymbol{B}\boldsymbol{u}[k]$$

$$\boldsymbol{y}[k] = \boldsymbol{C}\boldsymbol{x}[k] + \boldsymbol{D}\boldsymbol{u}[k],$$

- A *sufficient* condition for the DRA to operate is that $\boldsymbol{H}(s)$ be an element of the Hardy space $\mathscr{H}_\infty$, which implies that it is a strictly stable and proper system.
- This is not a *necessary* condition, however, as we will later generalize the method to work with systems having isolated pole(s) on the imaginary axis.
- Note that we do not restrict $\boldsymbol{H}(s)$ to be formulated as a quotient of polynomials in the Laplace variable "$s$" (for which well-known methods exist to find the discrete-time system).
- We describe the algorithm in four steps, which we preview here, and discuss in more detail in the following subsections.

  **STEP 1:** Sample the continuous-time transfer function $H(s)$ in the frequency domain at a high rate, and take the inverse discrete Fourier transform (IDFT) of the samples to get an approximation to the continuous-time impulse response $h(t)$.

  **STEP 2:** Use $h(t)$ to approximate the continuous-time step response $h_{\text{step}}(t)$, also sampled at the high rate.

**STEP 3:** Compute discrete-time unit-pulse response $g_k$ with inter-sample period $T_s$ from continuous-time step response $h_{\text{step}}(t)$, assuming a sample and hold circuit connected to system input.

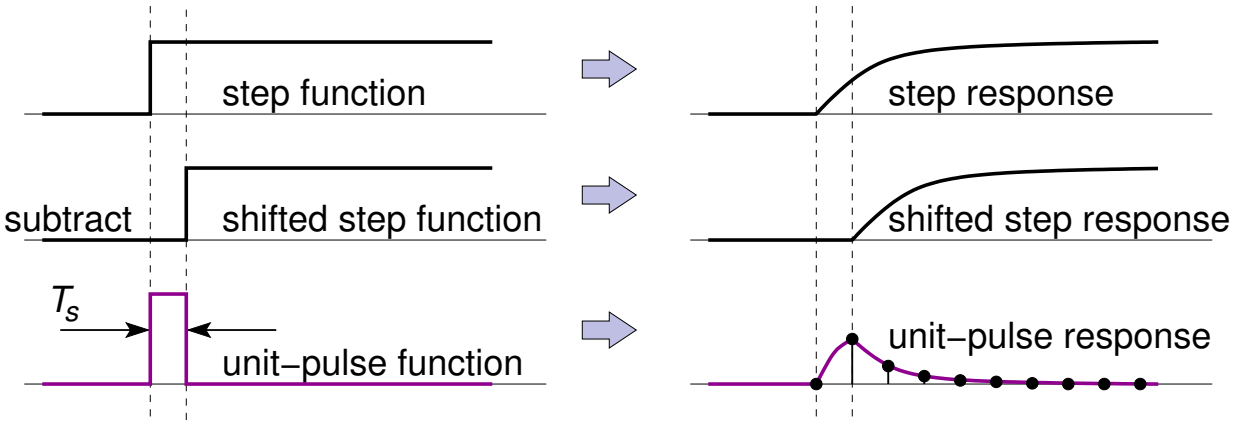
**STEP 4:** Generate a discrete-time state-space realization using the deterministic Ho–Kalman algorithm.

- We note that a system having a pole at the origin does not meet the strictly-stable requirement. However, we also show that this pole can be automatically accounted for.

## Building the DRA from the end to the beginning

**STEP 3:** If we have the system's unit-pulse response, we can use Ho–Kalman to find a state-space representation.

- But, how to find the unit pulse response? Let's assume that we know the continuous-time step response $h_{\text{step}}(t)$:




- The continuous-time response to a unit pulse of length $T_s$ seconds is $h_{\text{pulse}}(t) = h_{\text{step}}(t) - h_{\text{step}}(t - T_s)$.
- The discrete-time response is found by sampling: $g_k = h_{\text{pulse}}(kT_s)$.

**STEP 2:** If we have the system's continuous-time step response, we can find a state-space representation.

- But, how to find the step response? Let's assume that we know the continuous-time impulse response $h(t)$. Then,

$$h_{\text{step}}(t) = \int_0^t h(\tau)\,\mathrm{d}\tau.$$

- In fact, since the DRA is a numeric algorithm, we can't deal with continuous time directly. Instead, we select a fast sample frequency $F_1$ such that $T_1 = \dfrac{1}{F_1} \ll T_s$.
- Then, the finely sampled continuous-time step response is:

$$h_{\text{step}}(kT_1) = T_1 \sum_{i=0}^{k-1} h(iT_1).$$

**STEP 1:** Given the system's finely sampled continuous-time impulse response, we can find a state-space representation.

- How to find the finely sampled continuous-time impulse response?
- We approximate the continuous-time impulse response via a "discrete equivalent" approach (frequency-domain emulation).
- We use the bilinear transform to write a high-sample-rate discrete-time approximation to the original continuous-time transfer function

$$H(z) \approx H(s)\big|_{s=\frac{2}{T_1}\frac{z-1}{z+1}},$$

where $T_1$ is the same emulation sampling period as before.[3]

[3] In order to arrive at an accurate estimation of the continuous time transfer function, the sampling frequency, $F_1 = 1/T_1$, must be high enough to capture the system dynamics. As a rule of thumb, the sampling frequency must be at least 20 times the as great as the bandwidth of the system to get an rough approximation in the frequency domain. A higher emulation sampling frequency gives more accurate results.

- We now recognize that the discrete Fourier transform (DFT) of a sequence is related to its $z$-transform via the relationship

$$H[f] = H(z)\Big|_{z=\exp(j2\pi f/N)} = H(s)\Big|_{s=\frac{2}{T_1}\left[\frac{e^{j2\pi f/N}-1}{e^{j2\pi f/N}+1}\right]}$$
$$= H(s)\Big|_{s=\frac{2j}{T_1}\tan(\pi f/N)}, \quad 0 \leq f < N,$$

  where $N$ is the number of points chosen for the underlying sequence, and is usually chosen to be a power of $2$ for efficient computations.

- The inverse DFT of $H[f]$ gives $h(nT_1)$, which is the approximation of the continuous-time impulse response at the emulation sampling period, $T_1$

$$h(nT_1) = \frac{1}{N}\sum_{f=0}^{N-1} H[f]e^{j2\pi fn/N},$$

  which is indexed from $n = 0$ to $n = N-1$.

## Examples of the DRA

- We will ultimately look at three examples to illustrate the DRA.
- The first two are rational-polynomial transfer functions, which we use because we can calculate the exact solution using other methods.
  - We can then compare the exact solutions to the approximate solutions obtained by the DRA.
- The third does not have a closed-form solution, but we can use a 1-D parabolic-elliptic partial differential equation solver to find an accurate near-exact solution against which to compare the DRA solution.
- We find excellent agreement between the exact solutions and DRA solutions in all cases.

## 5.11: Example 1: Rational polynomial transfer function

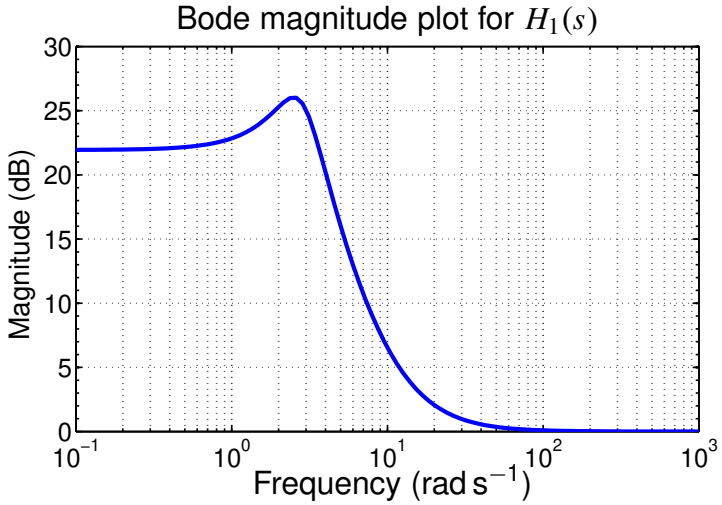
- The DRA method is first applied to a simple second-order system.
- We require a discrete-time realization with the a sampling period of $T_s = 0.1$ seconds from the continuous-time transfer function

$$H_1(s) = \frac{s^2 + 20s + 100}{s^2 + 2s + 8}.$$

- We compute the Bode plot to estimate the system bandwidth.

```
omega = logspace(-1,3,100);          % create freq. axis in rad/sec
s = 1j*omega;                        % create s = j*omega
H = (s.^2+20*s+100)./(s.^2+2*s+8);   % compute cplx. freq. response
semilogx(omega,20*log10(abs(H)));    % display the magnitude response
```

- Poles at $-1 \pm \text{j}2.65\,\text{rad}\,\text{s}^{-1}$, two zeros at $10\,\text{rad}\,\text{s}^{-1}$.
- The magnitude response of $H_1(s)$ is shown in the figure.
- The system bandwidth is on the order of $3\,\text{rad}\,\text{s}^{-1}$ (about 0.5 Hz).




**STEP 1:** The sampling frequency is selected as 256 Hz which is (much) greater than 20 times the system bandwidth.

- Transfer function is sampled at discrete frequencies; inverse DFT yields an approximate continuous-time impulse response.

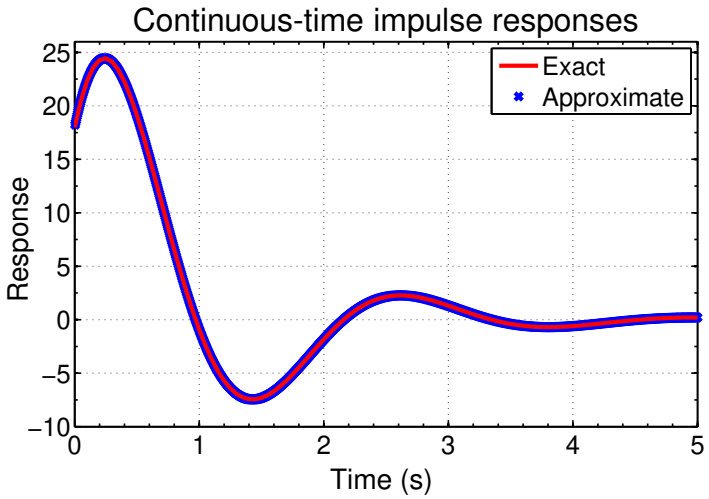
```
F1 = 256; T1 = 1/F1;                 % Interp. freq. of 256 Hz
minTlen = 6.5;                       % min. h(t) length in sec.
N = 2^(ceil(log2(minTlen*F1)));      % # of samples at rate F1
f = 0:N-1;                           % normalized freq. vector
s = (2j/T1)*tan(pi*f/N);             % substitute to get Hd[f]
```

```
Hd = (s.^2+20*s+100)./(s.^2+2*s+8);       % Hd[f]
hd = real(ifft(Hd)) * F1;                 % approximation to h(t)
td = T1*(0:N-1);                          % time vector for h(t)
plot(td,hd,'bx','markersize',8); hold on % plot h(t)

H1 = tf([1 20 100],[1 2 8]);              % true transfer function
[himpTrue,timpTrue] = impulse(H1,5);      % true impulse response
plot(timpTrue,himpTrue,'r'); axis([0 5 -10 26]); % plot on top
```

- The figure compares the approximate continuous-time impulse response computed via the inverse DFT to the exact continuous-time impulse response of $H_1(s)$.
- The solutions are coincident.

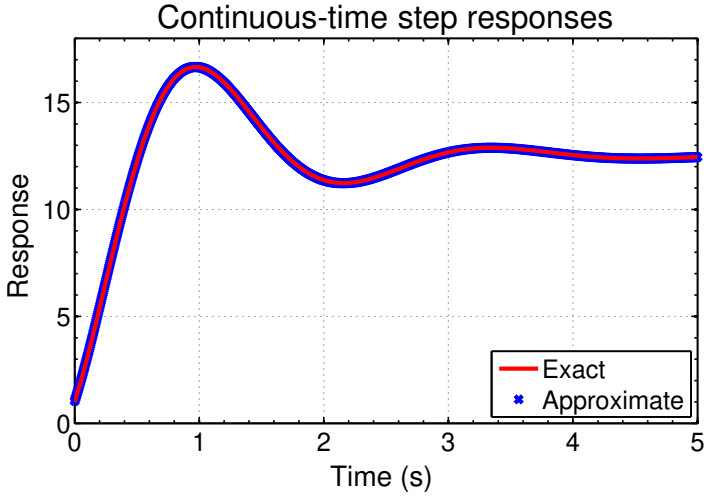



**STEP 2:** The approximation to the continuous-time step response is found by doing a cumulative summation of the impulse response.

```
hstep = T1*cumsum(hd);                          % h_s(t)
plot(td,hstep,'bx','markersize',8); hold on     % plot h_s(t)

[hstepTrue,tstepTrue] = step(H1,5);             % true step resp.
plot(tstepTrue,hstepTrue,'r'); axis([0 5 0 18]); % plot on top
```

- The results are shown in the figure and show excellent agreement with the exact step response of the continuous time system.

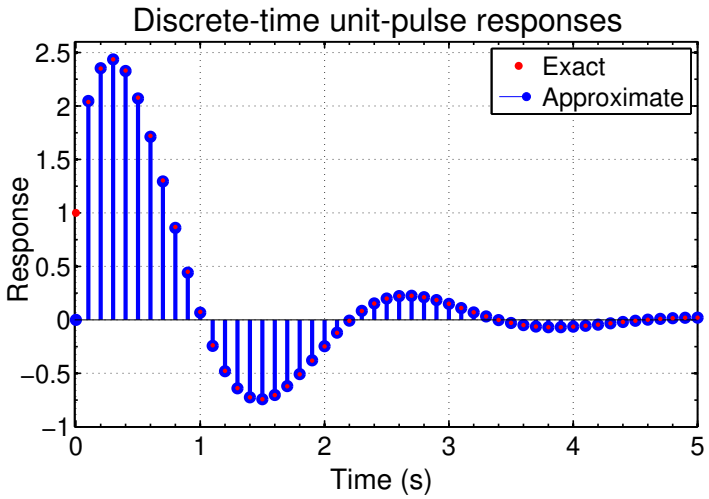


**STEP 3:** We now resample the continuous-time approximate step response at the final sample rate $T_s$, and compute the discrete-time unit-pulse response as $h_{\text{step}}[k] - h_{\text{step}}[k-1]$

```
Ts = 0.1; tdisc = 0:Ts:6.5;                    % final time vector
hdisc = [0 diff(interp1(td,hstep,tdisc))];     % h[k]
stem(tdisc,hdisc,'filled'); hold on

[himpDiscTrue,timpDiscTrue] = impulse(c2d(H1,Ts),5);
% next line scales IMPULSE in new MATLAB to give unit-pulse resp.
himpDiscTrue = Ts*himpDiscTrue;
plot(timpDiscTrue,himpDiscTrue,'r.','markersize',8);
axis([-0.01 5 -1 2.6]);
```

- Note that in new versions of MATLAB, the "impulse" command works differently from old versions for discrete-time systems.
- We need to scale MATLAB's output by $T_s$ to compute the unit-pulse response that we desire.




- Again, there is excellent agreement between the approximate unit-pulse response and the exact solution, except at single point $t = 0$.
- This is often the case because of some properties of the inverse DFT.
- But it causes no problems since the unit-pulse response value at $t = 0$ is computed differently, using

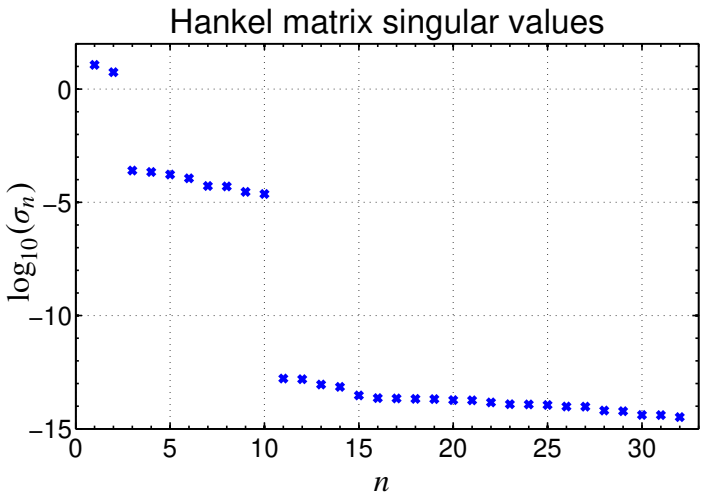
$$D = g_0 = \lim_{s \to \infty} H(s).$$

**STEP 4:** The Ho–Kalman algorithm is used to find state-space realization from approximate discrete-time unit-pulse response of Step 3.

- 64 points from the discrete-time unit-pulse response are used, which allows a maximum Hankel matrix of $32 \times 32$.
- We first compute and plot the singular values of the Hankel matrix.

```
bigHankel = hankel(hdisc(2:66));  % don't forget to omit h(0) term!
% for this example, keep only 32x32 portion
Hankel = bigHankel(1:32,1:32);
HankelUp = bigHankel(2:33,1:32);  % shifted Hankel matrix
[U, S, V] = svd(Hankel);          % compute singular values
plot(log10(diag(S)),'bx','markersize',8); axis([0 33 -20 5]);
```

- Hankel-matrix SVD gives insight into the system's order.
- A log plot of the singular values is shown in the figure.
- The first two are almost three orders of magnitude greater than the third, so we select a reduced-order model dimension $p = 2$.




```
n = 2;            % select via singular values
Us = U(:,1:n);    % Compute extended observability, controlability
Ss = S(1:n,1:n); % matrices, sized to the order for the system
Vs = V(:,1:n);    % inferred by the singular values.
Ok = Us*sqrtm(Ss); Cl = sqrtm(Ss)*Vs';

Ahat = (Ok\HankelUp)/Cl;                % calculate A from Ok, Cl
Bhat = Cl(1:n,1); Chat = Ok(1,1:n);     % calculate B and C
Dhat = 1;                               % calculated manually
sysDRA = ss(Ahat,Bhat,Chat,Dhat,Ts);    % final DRA ss model
```

- Truncating to the first two states only, the Ho–Kalman algorithm gives a state-space realization with the following $A$, $B$, and $C$ matrices

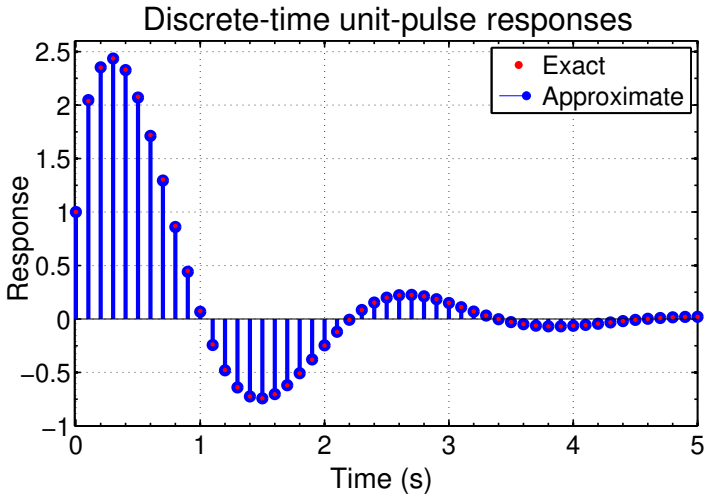
$$\widehat{\boldsymbol{A}} = \begin{bmatrix} 0.8656 & -0.2367 \\ 0.2367 & 0.8811 \end{bmatrix}, \widehat{\boldsymbol{B}} = \begin{bmatrix} -1.624 \\ 0.7692 \end{bmatrix}, \widehat{\boldsymbol{C}} = \begin{bmatrix} -1.624 & -0.7692 \end{bmatrix}.$$

- The $\widehat{\boldsymbol{D}}$ matrix is found from the initial value theorem and, for this example, is $\widehat{\boldsymbol{D}} = [1]$.

- We compare the true discrete-time unit-pulse response and the final DRA model unit-pulse response:

```
% next line scales IMPULSE in new MATLAB to give unit-pulse resp.
[himpDRA,timpDRA] = impulse(sysDRA,5); himpDRA = Ts*himpDRA;
stem(timpDRA,himpDRA,'filled'); hold on
plot(timpDiscTrue,himpDiscTrue,'r.','markersize',8);
axis([-0.01 5 -1 2.6]);
```

- The results agree very well (note that $h[0]$ has been corrected by the correct calculation of the $\widehat{\boldsymbol{D}}$ matrix in Step 4).




- Because the unit-pulse responses agree very well, the response of the reduced-order model will also agree well with the exact response for any input signal $u[k]$.

## 5.12: Example 2: Dealing with a pole in $H(s)$ at the origin

- This example has a pole in $H(s)$ at $s = 0$, so is not strictly stable, and violates the necessary conditions that make the DRA work.
- However, it is quite simple to deal with this case.
  - We first subtract the pole at the origin from the transfer function,
  - Then execute the DRA on the residual system,
  - Then compute a final discrete-time state-space model that augments the DRA result with additional dynamics to implement the function of the $s$-domain pole at the origin.
- A pole at the origin is removed by first calculating the residue of this pole and then subtracting it from the original transfer function.

$$H^*(s) = H(s) - \frac{\text{res}_0}{s} \qquad \text{where} \qquad \text{res}_0 = \lim_{s\to 0} sH(s).$$

- The remainder of the DRA is executed using $H^*(s)$ instead of $H(s)$.
- To re-incorporate the effect of the pole at $s = 0$ into the final reduced-order model, recall that this pole corresponds to an integrator. The discrete-time equivalent can be implemented as

$$x_i[k+1] = x_i[k] + T_s u[k].$$

- We combine this with the DRA-produced state-space form

$$\underbrace{\left[\begin{array}{c} \boldsymbol{x}[k+1] \\ \hdashline x_i[k+1] \end{array}\right]}_{\boldsymbol{x}_{\text{aug}}[k+1]} = \underbrace{\left[\begin{array}{c:c} \widehat{\boldsymbol{A}} & \boldsymbol{0} \\ \hdashline \boldsymbol{0} & 1 \end{array}\right]}_{\widehat{\boldsymbol{A}}_{\text{aug}}} \underbrace{\left[\begin{array}{c} \boldsymbol{x}[k] \\ \hdashline x_i[k] \end{array}\right]}_{\boldsymbol{x}_{\text{aug}}[k]} + \underbrace{\left[\begin{array}{c} \widehat{\boldsymbol{B}} \\ \hdashline T_s \end{array}\right]}_{\widehat{\boldsymbol{B}}_{\text{aug}}} \boldsymbol{u}[k]$$

$$\boldsymbol{y}[k] = \underbrace{\left[\begin{array}{c:c} \widehat{\boldsymbol{C}} & \text{res}_0 \end{array}\right]}_{\widehat{\boldsymbol{C}}_{\text{aug}}} \left[\begin{array}{c} \boldsymbol{x}[k] \\ \hdashline x_i[k] \end{array}\right] + \boldsymbol{D}\boldsymbol{u}[k]$$

where dotted lines delineate boundaries between block sub-matrices of the overall augmented state-space matrices $\widehat{\boldsymbol{A}}_{\text{aug}}$, $\widehat{\boldsymbol{B}}_{\text{aug}}$, and $\widehat{\boldsymbol{C}}_{\text{aug}}$.

## Example 2: Rational polynomial transfer function with pole at origin

- In this example, we demonstrate how to handle a single pole at the origin. The continuous-time transfer function is given by

$$H_2(s) = \frac{1}{s}\left(\frac{1}{s^2 + 6s + 8}\right).$$

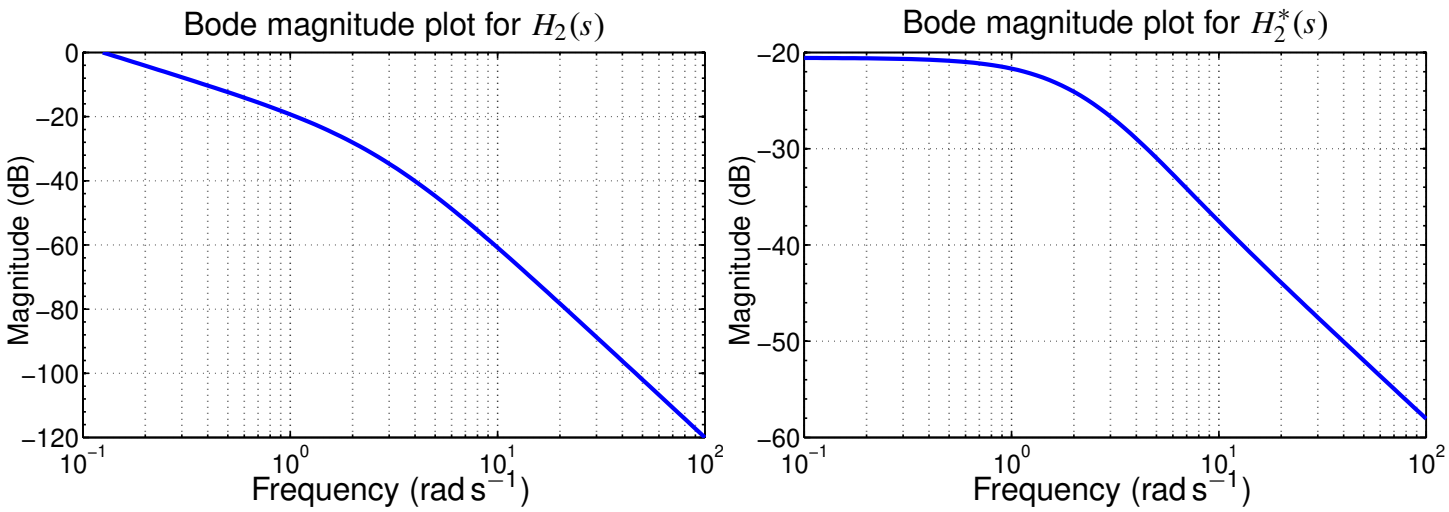
- This system has real poles at 0, 2 and 4 rad s$^{-1}$.
- Desire a discrete-time transfer function with sample period $T_s = 0.1$ s.
- Prior to Step 1 we remove the pole at the origin.
- This is accomplished by first calculating the residue for this pole.
- In this example, the residue can be computed analytically as

$$\text{res}_0 = \lim_{s\to 0} s\,H(s) = 0.125.$$

- In general, we find this residue by selecting a very small value for $s$ and numerically computing $\text{res}_0$, or by using a software tool like Mathematica to compute the limit.
- The reduced transfer function, $H_2^*(s)$ with the pole at the origin removed is

$$H_2^*(s) = \frac{1}{s}\left(\frac{1}{s^2 + 6s + 8}\right) - \frac{0.125}{s}.$$

- The figures below shows the magnitude plot of the original system and the system with the pole at the origin removed.




**STEP 1.** $H_2^*(s)$ is sampled at 256 Hz which is (much) more than 50 times greater than the system bandwidth. We could implement either

```
Hd = 1./(s.^3+6*s.^2+8*s) - 0.125./s;      % Hd[f]
Hd(1) = -6/64;                             % analytic solution
```

where $\lim_{s\to 0} H_2^*(s) = -6/64$, or compute by hand

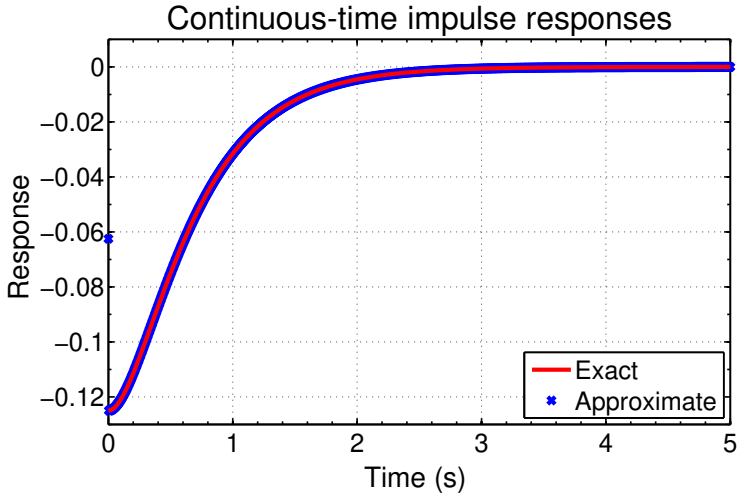
$$H_2^*(s) = \frac{1}{s}\left(\frac{1}{s^2+6s+8}\right) - \frac{0.125}{s}\left(\frac{s^2+6s+8}{s^2+6s+8}\right)$$

$$= -\frac{0.125}{s}\left(\frac{s^2+6s}{s^2+6s+8}\right) = -0.125\left(\frac{s+6}{s^2+6s+8}\right),$$

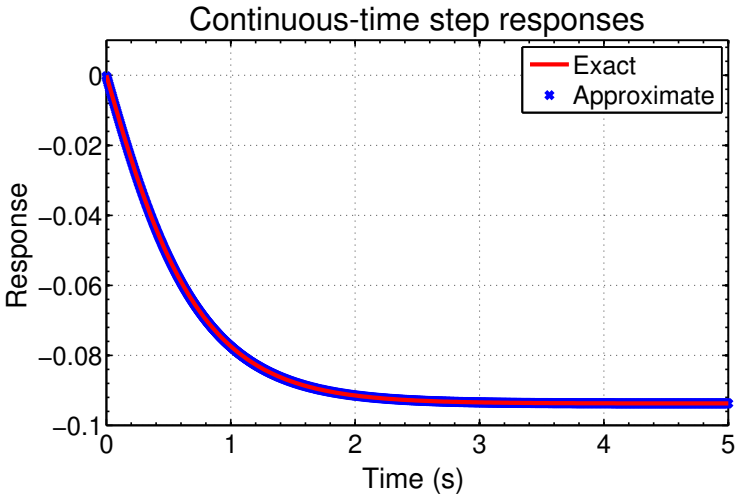
and implement

```
Hd = -0.125*(s+6)./(s.^2+6*s+8);
```

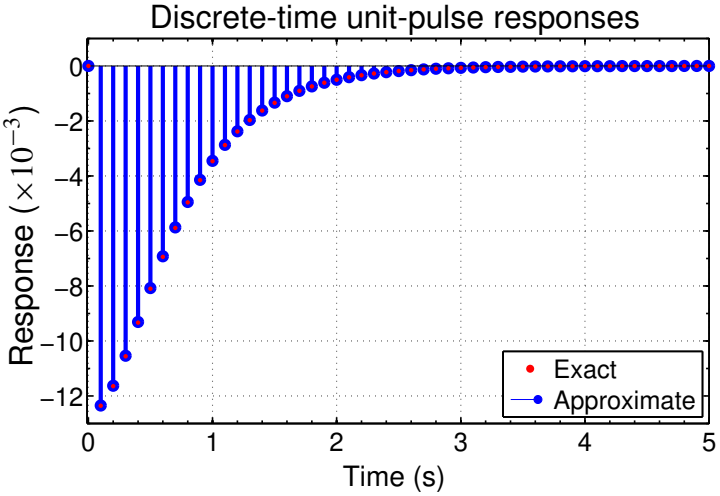
- The approximate continuous-time impulse response is computed and plotted.



**STEP 2:** The approximation to the continuous-time step response of $H_2^*(s)$ is calculated as in the first example and plotted.

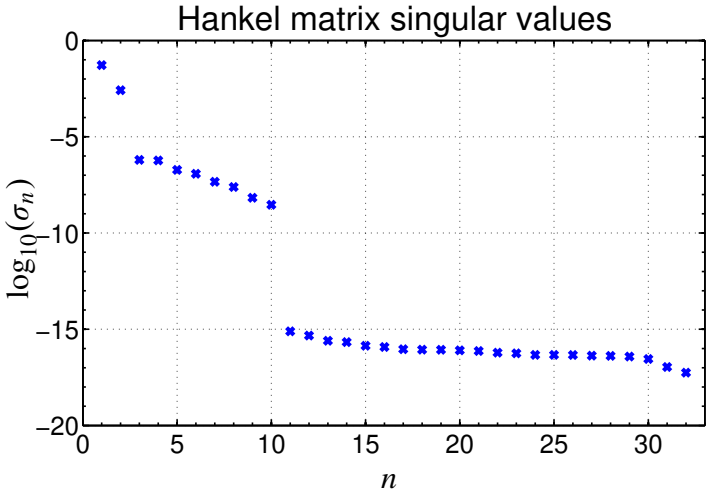



**STEP 3:** This step response is sampled at $T_s = 0.1$ seconds, and differenced to yield the discrete-time unit-pulse response, plotted in the figure.




**STEP 4.** The system Hankel matrix is generated from the discrete-time unit-pulse response found in Step 3.

- 64 discrete time points are used, resulting in a 32×32 Hankel matrix.
- The figure depicts the 32 singular values of the system Hankel matrix.
- The first two singular values are two orders of magnitude greater than the third, indicating that $H_2^*(s)$ is a second order system.



- The Ho–Kalman algorithm generates the $\widehat{A}$, $\widehat{B}$, and $\widehat{C}$ matrices after truncating all but the first two states. We find that

$$\widehat{A} = \begin{bmatrix} 0.8617 & -0.0906 \\ 0.0906 & 0.6274 \end{bmatrix}, \qquad \widehat{B} = \begin{bmatrix} 0.1162 \\ -0.0340 \end{bmatrix}$$

$$\widehat{C} = \begin{bmatrix} -0.1162 & -0.0340 \end{bmatrix}.$$

  In this example, we also compute $\widehat{D} = \lim_{s\to\infty} H_2^*(s) = 0$, which can also be quite easily seen in the high-frequency response of $H_2^*(s)$.

- The state-space representation for $H_2^*(s)$ is augmented to include the pole at the origin to create a representation for $H_s(s)$.

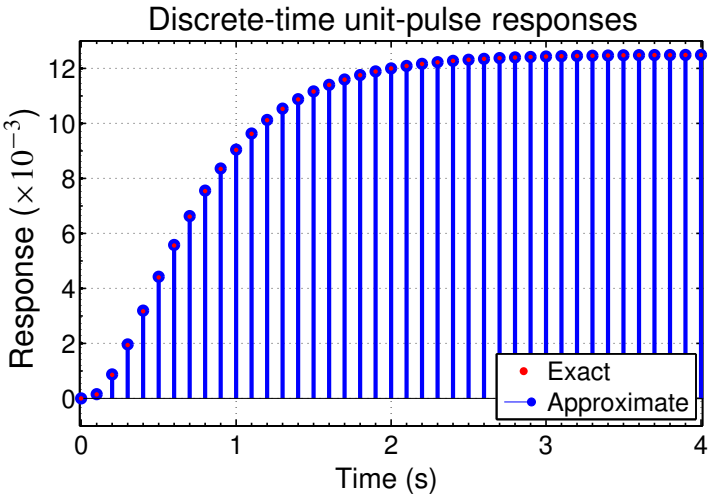
```
Aaug = [Ahat, zeros(n,1); zeros(1,n), 1];
Baug = [Bhat; Ts];
Caug = [Chat, res0];
sysDRA = ss(Aaug,Baug,Caug,Dhat,Ts)   % final DRA state-space sys.
```

- The discrete-time realization of $H_2(s)$ is

$$\widehat{A}_{\text{aug}} = \left[\begin{array}{cc|c} 0.8617 & -0.0906 & 0 \\ 0.0906 & 0.6274 & 0 \\ \hline 0 & 0 & 1 \end{array}\right], \qquad \widehat{B}_{\text{aug}} = \left[\begin{array}{c} 0.1162 \\ -0.0340 \\ \hline 0.1 \end{array}\right]$$

$$\widehat{C}_{\text{aug}} = \left[\begin{array}{cc|c} -0.1162 & -0.0340 & 0.125 \end{array}\right], \qquad \widehat{D} = [0].$$

- The figure shows close comparison of the unit-pulse response found from the DRA and the exact solution.



## 5.13: Example 3: Transcendental transfer function

- In the first two examples, we used rational polynomials to illustrate the DRA method where order of the system is known *a priori*, and the exact answer could be calculated analytically.
- We now demonstrate the DRA with an infinite-order distributed-parameter system: Specifically the Jacobsen–West transfer function of lithium diffusion in a single particle, where

$$H_3(s) = \frac{\widetilde{C}_{s,e}(s)}{J(s)} = \frac{R_s}{D_s}\left[\frac{1}{1 - R_s\sqrt{s/D_s}\coth(R_s\sqrt{s/D_s})}\right],$$

and where the integrator-removed transfer function is

$$\begin{aligned} H_3^*(s) = \frac{\Delta\widetilde{C}_{s,e}(s)}{J(s)} &= \frac{\widetilde{C}_{s,e}(s)}{J(s)} - \frac{\widetilde{C}_{s,\text{avg}}(s)}{J(s)} \\ &= \frac{\frac{sR_s^2}{D_s} + 3 - 3R_s\sqrt{\frac{s}{D_s}}\coth\left(R_s\sqrt{\frac{s}{D_s}}\right)}{sR_s\left(1 - R_s\sqrt{\frac{s}{D_s}}\coth\left(R_s\sqrt{\frac{s}{D_s}}\right)\right)}, \end{aligned}$$

where we have used the relationship

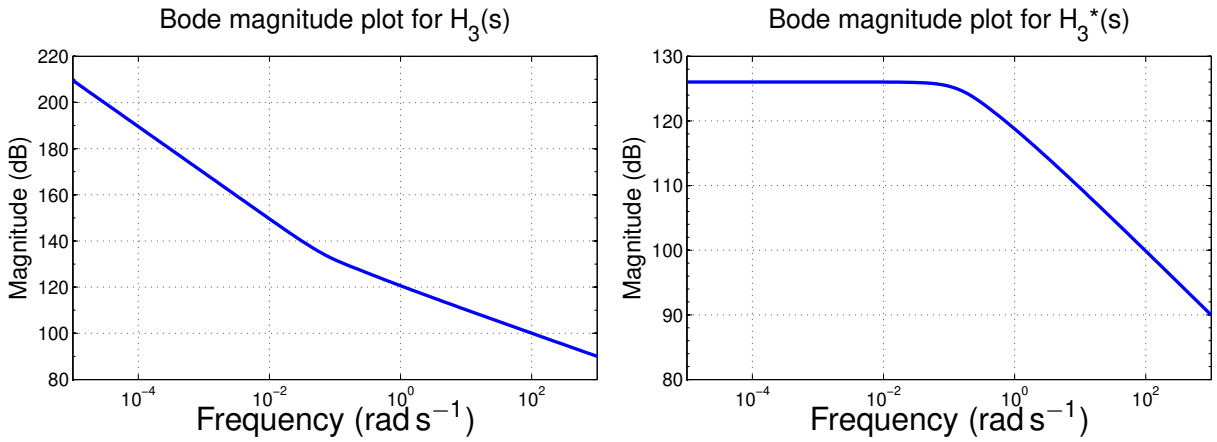
$$\frac{\widetilde{C}_{s,\text{avg}}(s)}{J(s)} = \frac{\text{res}_0}{s} = \frac{-3/R_s}{s}.$$

- Parameter values for the transfer functions used in this example are listed in the table, from which we can compute that $\text{res}_0 = -3 \times 10^5$.

| Parameter name | Interpretation | Value |
|---|---|---|
| $T_s$ | Sampling period | 1 s |
| $R_s$ | Particle radius | $10^{-5}$ m |
| $D_s$ | Diffusivity | $10^{-12}$ m$^2$ s$^{-1}$ |
| $c(r,0)$ | Initial lithium concentration | 10 000 mol m$^{-3}$ |

**STEP 1.** The magnitude responses of $H_3(s)$ and $H_3^*(s)$ are shown below:




- $H_3^*(s)$ is sampled at 256 Hz for a total of 256 seconds.
- The frequency vector for $H_3^*(s)$ can be calculated as

```
beta = Rs*sqrt(s/Ds);
Hd = (Rs/Ds)*(1./(1-beta.*coth(beta))) + (3/Rs)./s;
Hd(1) = -Rs/(5*Ds); % analytic solution
```

  where MATLAB numerically removes the integrator pole, or as

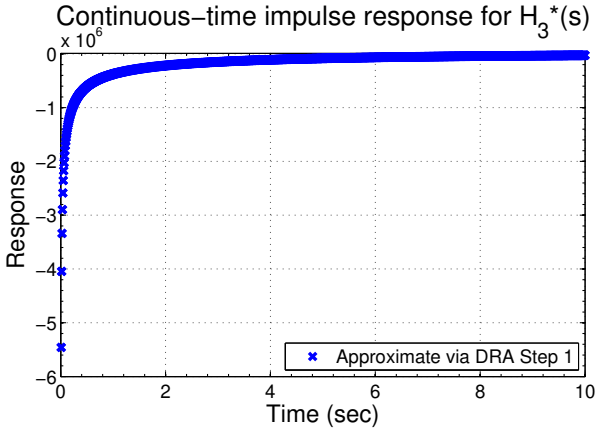
```
beta = Rs*sqrt(s/Ds);
Hd = (beta.^2+3-3*beta.*coth(beta))./(s.*Rs.*(1-beta.*coth(beta)));
Hd(1) = -Rs/(5*Ds); % analytic solution
```

- Note that both computations of `Hd` initially produce `NaN` for $s = 0$ due to numeric attempts to evaluate zero divided by zero.
- This entry must be manually replaced by a value computed analytically

$$\lim_{s\to 0} H_3^*(s) = \lim_{s\to 0} \frac{\frac{sR_s^2}{D_s} + 3 - 3R_s\sqrt{\frac{s}{D_s}}\coth\left(R_s\sqrt{\frac{s}{D_s}}\right)}{sR_s\left(1 - R_s\sqrt{\frac{s}{D_s}}\coth\left(R_s\sqrt{\frac{s}{D_s}}\right)\right)} = -\frac{R_s}{5D_s}.$$

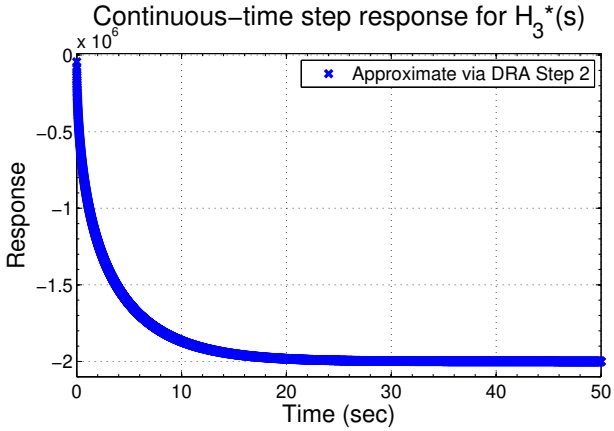
- Direct by-hand computation returns $0/0$. We must use l'Hôpital's rule repeatedly until an answer is reached.
- When using transcendental transfer functions, we recommend computer tools such as Mathematica for symbolic manipulation.

- The approximate continuous-time impulse response is shown.
- There is no known exact solution against which to compare this result.

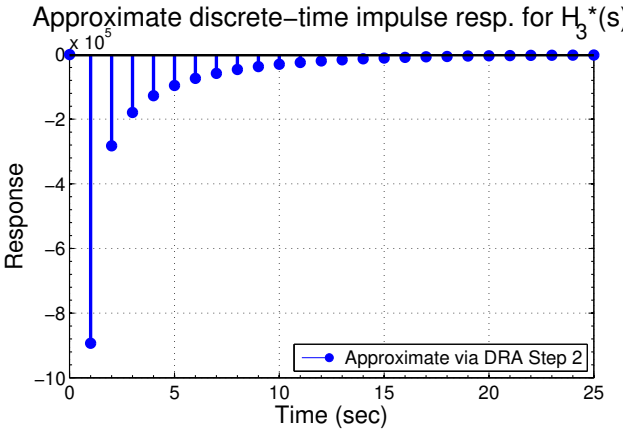



**STEP 2.** The approximate continuous-time step response is calculated by performing a cumulative sum of the impulse response of Step 1.

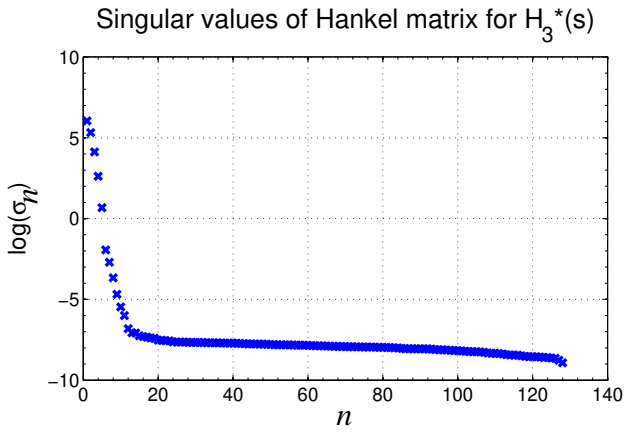
- The figure shows approximated continuous-time step response.
- Again, there is no known exact solution against which to compare this result.




**STEP 3:** The approximate continuous-time step response is sampled at $T_s = 1$ second, and differenced to produce the discrete-time unit-pulse response, shown here.



**STEP 4.** Hankel matrix is formed; singular values are plotted.




- $H_3^*(s)$ represents a distributed-parameter system that actually has an infinite number of poles.
- However, only a few of them are significant to the solution.
- In particular, we choose to use a reduced-order model dimension $n = 2$ in the results we present here, imposing a tradeoff between the complexity and accuracy of the solution.
- The Ho–Kalman algorithm generates the $\widehat{\boldsymbol{A}}$, $\widehat{\boldsymbol{B}}$, and $\widehat{\boldsymbol{C}}$ matrices to approximate $H_3^*(s)$ after truncating all but the first two states.

$$\widehat{\boldsymbol{A}} = \begin{bmatrix} 0.4695 & 0.3296 \\ 0.3296 & 0.4355 \end{bmatrix}, \qquad \widehat{\boldsymbol{B}} = \begin{bmatrix} 919.1 \\ -220 \end{bmatrix}$$

$$\widehat{\boldsymbol{C}} = \begin{bmatrix} -919.1 & 220 \end{bmatrix}.$$

- In this example, we also compute $\widehat{\boldsymbol{D}} = \lim_{s\to\infty} H_3^*(s) = 0$, which can also be quite easily seen in the high-frequency response of $H_3^*(s)$.
- This state-space realization is augmented with the integrator state to give the final third-order model of the diffusion equation $H_3(s)$.

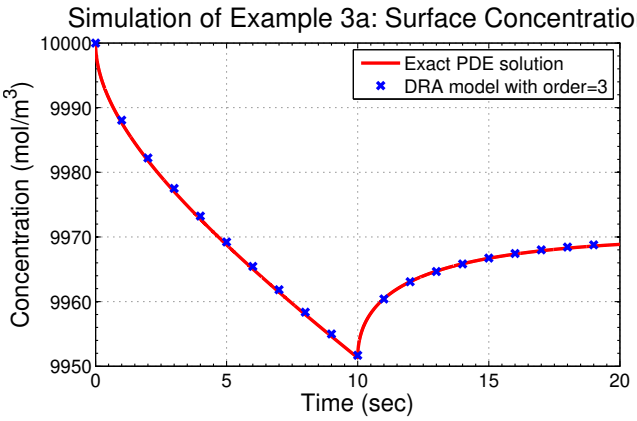
$$\widehat{\boldsymbol{A}}_{\text{aug}} = \left[\begin{array}{cc:c} 0.4695 & 0.3296 & 0 \\ 0.3296 & 0.4355 & 0 \\ \hdashline 0 & 0 & 1 \end{array}\right], \qquad \widehat{\boldsymbol{B}}_{\text{aug}} = \left[\begin{array}{c} 919.1 \\ -220 \\ \hdashline 1 \end{array}\right]$$

$$\widehat{\boldsymbol{C}}_{\text{aug}} = \left[\begin{array}{cc:c} -919.1 & 220 & -3\times 10^5 \end{array}\right], \qquad \widehat{\boldsymbol{D}} = [0].$$

- We demonstrate the the DRA-produced model by simulating a 10 s discharge pulse where the surface lithium flux (leaving the particle) was $j = 1 \times 10^{-5}\,\text{mol}\,\text{m}^{-2}\,\text{s}^{-1}$, followed by a 10 s rest.
- The augmented state-space model was simulated with this input to produce $\tilde{c}_{s,e}[k]$, and $c_{s,e}[k]$ was computed as $c_{s,e}[k] = \tilde{c}_{s,e}[k] + c_{s,0}$.

```
cs0 = 10000;
uk = 1e-5*[ones(1,10),zeros(1,10)];
[cseTilde,tk] = lsim(sysDRA,uk);
cse = cseTilde + cs0;
```

- All discrete-time model states are initialized to zero.
- The output of this discrete-time realization to a 10 second discharge followed by a 10 second rest is shown.




- We compare this result against the "truth" produced by simulating the PDE using MATLAB's 1-D parabolic-elliptic PDE solver.

```
function [cse,t] = simCsePDE
  dr = 0.1e-6; % Radial resolution = 0.1 micro-meter
  dt = 0.001;  % Time step in simulation, s
  Tfinal = 20; % Length of simulation, s
  Rp = 10e-6;  % Radius of particle = 10 micro-meters
  Ds = 1e-12;  % Solid diffusivity, m^2/s
  j = 1e-5;    % mol/m^2/s

  x = 0:dr:Rp;      % locations for solution
  t = 0:dt:Tfinal; % time steps for solution

  options = odeset('RelTol',1e-8,'AbsTol',1e-10);
  sol = pdepe(2,@csefun,@cseic,@csebc,x,t,options);
  cse = sol(:,end,1);

  function[c,f,s] = csefun(~,~,~,DuDx)
```

```
    c = 1/Ds; f = DuDx; s = 0;
  end

  function u0 = cseic(~,~)
    c0 = 10000; u0 = c0;
  end

  function[pl,ql,pr,qr] = csebc(~,~,~,~,t)
    pl = 0; ql = 1; qr = Ds; pr = 0;
    if t<Tfinal/2, pr=j; end
  end

end
```

- The code comprises nested functions, where the main function initializes variables and calls MATLAB's solver with pointers (function handles) to nested helper functions:
  - `csefun` implements the parameter values of the PDE;
  - `cseic` implements the initial conditions; and
  - `csebc` implements the boundary conditions.
- Note that we achieve good results with the PDE solver only if a fine time-step is used: here, we have used a 1 ms step size, which makes the PDE solver execute much more slowly than the DRA-produced model.

## Where from here?

- We have now seen the form that the final model will take, and examples of the general methodology to go from the PDE continuum-scale model to the reduced-order model.
- We now proceed to develop transfer functions for all cell variables of interest, and see how well the overall cell model works.