# Systems Laboratory, Spring 2025

Damiano Varagnolo – CC-BY-4.0



notes

- welcome to the course!
- on this side of this document you will find notes that accompany the text typically visualized in class
- these notes are meant to convey the messages that are not displayed in the text on the side, and basically constitute what the teacher intends to say in class

## Table of Contents I


- state space from ARMA (and viceversa)
  - From state space to ARMA
  - From ARMA to SS
  - Most important python code for this sub-module
  - Self-assessment material




notes

- this is the table of contents of this document; each section corresponds to a specific part of the course

# state space from ARMA (and viceversa)



notes

## Contents map

| developed content units | taxonomy levels |
|---|---|
| realization | u1, e1 |

| prerequisite content units | taxonomy levels |
|---|---|
| ARMA model | u1, e1 |
| state space model model | u1, e1 |
| matrix inversion | u1, e1 |
| Laplace transforms | u1, e1 |



notes

## Main ILO of sub-module "state space from ARMA (and viceversa)"

**Determine** the state space structure of an
LTI system starting from an ARMA ODE



notes

- by the end of this module you shall be able to do this

## ARMA models

$$y^{(n)} = a_{n-1}y^{(n-1)} + \ldots + a_0 y + b_m u^{(m)} + \ldots + b_0 u$$



notes

- the $a_{n-1}y^{(n-1)} + \ldots + a_0 y$ part is called Auto-Regressive
- the $b_m u^{(m)} + \ldots + b_0 u$ part is called Moving-Average
- these names make more sense in discrete time systems of the type $y(k+n) = a_{n-1}y(k+n-1) + \ldots + a_0 y(k) + b_m u^(k+m) + \ldots + b_0 u(k)$ and $k$ a discrete time index. Here we see that the $a$'s correspond to an autoregression, and the $b$'s to the coefficients of a moving average. In any case we use ARMA for both continuous and discrete dynamics of these types

## State space representations - Notation

$$
\begin{aligned}
\dot{x}_1 &= f_1\left(x_1, \ldots, x_n, u_1, \ldots, u_m\right) \\
&\vdots \\
\dot{x}_n &= f_n\left(x_1, \ldots, x_n, u_1, \ldots, u_m\right) \\
y_1 &= g_1\left(x_1, \ldots, x_n, u_1, \ldots, u_m\right) \\
&\vdots \\
y_p &= g_p\left(x_1, \ldots, x_n, u_1, \ldots, u_m\right)
\end{aligned}
$$

$$
\begin{aligned}
\dot{\boldsymbol{x}} &= \boldsymbol{f}\left(\boldsymbol{x}, \boldsymbol{u}\right) \\
\boldsymbol{y} &= \boldsymbol{g}\left(\boldsymbol{x}, \boldsymbol{u}\right)
\end{aligned}
$$

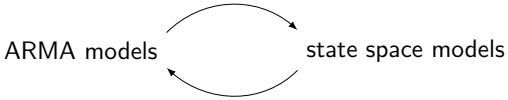
- $\boldsymbol{f}$ = state transition map
- $\boldsymbol{g}$ = output map



notes

- notation wide, remember that state space means first order ODEs
- they will thus look like these ones, in general
- we can also compress the notation in this way
- remember that bold non-capital fonts mean vectors in this course
- and we give to the various things these names

## This module:




*But why do we study this?*

because from physical laws we get ARMA,
but with state space we get more explainable models



notes

- we will learn how to do two simple operations
- we will only scratch the surface though, there is a lot of material to cover here and you will do it much better in other modules / courses
- and often one does the "ARMA to SS" operation

From state space to ARMA



notes

## SS to ARMA

Tacit assumption: $\boldsymbol{x}(0) = \mathbf{0}$

$$
\begin{aligned}
&\begin{cases} \dot{\boldsymbol{x}} &= A\boldsymbol{x} + Bu \\ y &= C\boldsymbol{x} + Du \end{cases} \\
\rightarrow \quad & \mathcal{L}\left( \begin{cases} \dot{\boldsymbol{x}} &= A\boldsymbol{x} + Bu \\ y &= C\boldsymbol{x} + Du \end{cases} \right) \\
\rightarrow \quad & \begin{cases} sX &= AX + BU \\ Y &= CX + DU \end{cases} \\
\rightarrow \quad & \begin{cases} (sI - A)X &= BU \\ Y &= CX + DU \end{cases} \\
\rightarrow \quad & \begin{cases} X &= (sI - A)^{-1}BU \quad (*) \\ Y &= CX + DU \end{cases} \\
\Rightarrow \quad & Y = \Big( C(sI - A)^{-1}B + D \Big) U \\
\Rightarrow \quad & Y(s) = \frac{\text{polynomial in } s}{\text{polynomial in } s} U(s)
\end{aligned}
$$



notes

- This slide shows the step-by-step derivation of the transfer function from the state-space representation using Laplace transforms.
- the assumption $\boldsymbol{x}(0) = \mathbf{0}$ simplifies the Laplace transform of the derivative.
- the key step where $X = (sI - A)^{-1}BU$ is derived is the foundation for the transfer function.
- the final result, $Y(s) = \frac{\text{polynomial in } s}{\text{polynomial in } s} U(s)$, is the ARMA representation of the system.
- For computations, I recommend using tools like `simpy` for symbolic algebra, but you should be able to handle 2x2 systems by hand.

## A note on the last formula

$$Y(s) = \frac{\text{polynomial in } s}{\text{polynomial in } s} U(s) \quad \mapsto \quad \text{ARMA:}$$

$$Y(s) = \frac{s+3}{2s^3+3s} U(s) \quad \mapsto \quad 2\dddot{y} + 3\dot{y} = \dot{u} + 3u$$



notes

- This slide connects the transfer function to the ARMA model in the time domain.
- the numerator and denominator polynomials in $s$ directly translate to differential equations in the time domain.
- the example shows how the transfer function $Y(s) = \frac{s+3}{2s^3+3s} U(s)$ corresponds to the differential equation $2\dddot{y} + 3\dot{y} = \dot{u} + 3u$.
- this is a key step in understanding the relationship between the Laplace domain and time domain.

## A note on the second to last formula

$$Y = \Big( C(sI - A)^{-1} B + D \Big) U$$

DISCLAIMER: in this course we consider SISO systems, thus $C$ and $B$ = vectors, and $D$ = scalar (if present)



notes

- that this course focuses on Single Input Single Output (SISO) systems, which simplifies the matrices $C$, $B$, and $D$.
- $C$ and $B$ are vectors, and $D$ is a scalar (often zero in many systems).
- this simplification is important for understanding the structure of the transfer function.
- MIMO (Multiple Input Multiple Output) systems in other courses!

## Numerical Example: $2 \times 2$ State-Space to ARMA

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \quad B = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 0 \end{bmatrix}, \quad D = \begin{bmatrix} 0 \end{bmatrix}$$



notes

- this numerical example is used to illustrate the conversion from state-space to ARMA.
- this is a 2x2 system, which is manageable by hand and helps students understand the process.
- the matrices $A$, $B$, $C$, and $D$ are chosen for simplicity, but the method is general.

## Numerical Example: $2 \times 2$ State-Space to ARMA

### Step 1: State-Space Equations

$$\begin{cases} \dot{x}_1 = x_1 + 2x_2 + u \\ \dot{x}_2 = 3x_1 + 4x_2 \\ y = x_1 \end{cases}$$



notes

- the state-space equations explicitly using the given matrices.
- $\dot{x}_1$ and $\dot{x}_2$ are linear combinations of the states and the input $u$.
- the output $y$ is simply the first state variable $x_1$.

## Numerical Example: $2 \times 2$ State-Space to ARMA

Step 2: Laplace Transform

$$\begin{cases} sX_1(s) = X_1(s) + 2X_2(s) + U(s) \\ sX_2(s) = 3X_1(s) + 4X_2(s) \\ Y(s) = X_1(s) \end{cases}$$



notes

- Apply the Laplace transform to the state-space equations, assuming zero initial conditions.
- the Laplace transform converts differential equations into algebraic equations in $s$.
- $Y(s) = X_1(s)$, which connects the output directly to the first state variable.

## Numerical Example: $2 \times 2$ State-Space to ARMA

Step 3: Rearrange in Matrix Form

$$\begin{cases} (sI - A)X(s) = BU(s) \\ Y(s) = CX(s) + DU(s) \end{cases}$$

implies

$$\begin{cases} \begin{bmatrix} s-1 & -2 \\ -3 & s-4 \end{bmatrix} \begin{bmatrix} X_1(s) \\ X_2(s) \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} U(s) \\ Y(s) = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} X_1(s) \\ X_2(s) \end{bmatrix} \end{cases}$$



notes

- Rearrange the Laplace-transformed equations into matrix form.
- the output equation $Y(s) = CX(s)$ remains simple due to the choice of $C$.

## Numerical Example: $2 \times 2$ State-Space to ARMA

Step 4: Solve for $X(s)$

$$X(s) = (sI - A)^{-1} B U(s)$$

$$(sI - A) = \begin{bmatrix} s-1 & -2 \\ -3 & s-4 \end{bmatrix}$$

$$(sI - A)^{-1} = \frac{1}{(s-1)(s-4) - (-2)(-3)} \begin{bmatrix} s-4 & 2 \\ 3 & s-1 \end{bmatrix}$$

$$\det(sI - A) = (s-1)(s-4) - 6 = s^2 - 5s - 2$$

$$(sI - A)^{-1} = \frac{1}{s^2 - 5s - 2} \begin{bmatrix} s-4 & 2 \\ 3 & s-1 \end{bmatrix}$$



notes

- Solve for $X(s)$ by computing $(sI - A)^{-1}$.
- the determinant $\det(sI - A)$, which appears in the denominator of the transfer function, is key.
- the step-by-step computation of the inverse matrix is assumed as a given skill.

## Numerical Example: $2 \times 2$ State-Space to ARMA

Step 5: Multiply by $B$

Now, multiply by $B$:

$$X(s) = \frac{1}{s^2 - 5s - 2} \begin{bmatrix} s-4 & 2 \\ 3 & s-1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} U(s) = \frac{1}{s^2 - 5s - 2} \begin{bmatrix} s-4 \\ 3 \end{bmatrix} U(s)$$



notes

- Multiply $(sI - A)^{-1}$ by $B$ to obtain $X(s)$.
- this step simplifies the expression for $X(s)$.
- $X_1(s)$ and $X_2(s)$ are now expressed in terms of $U(s)$.

## Numerical Example: $2 \times 2$ State-Space to ARMA

Step 6: Solve for $Y(s)$

Substitute $X(s)$ into the output equation:

$$Y(s) = CX(s) + DU(s) = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} X_1(s) \\ X_2(s) \end{bmatrix} = X_1(s)$$

Thus:

$$Y(s) = \frac{s-4}{s^2 - 5s - 2} U(s)$$



notes

- Substitute $X(s)$ into the output equation to find $Y(s)$.
- $Y(s)$ is directly proportional to $X_1(s)$.

## Numerical Example: $2 \times 2$ State-Space to ARMA

Step 7: Final Result

Transfer function $H(s)$:

$$H(s) = \frac{Y(s)}{U(s)} = \frac{s-4}{s^2 - 5s - 2}$$

and from this we get the ARMA representation of the system as before



notes

- this is the ARMA representation of the system.

# From ARMA to SS



notes

## Starting point (blending Laplace notation with time notation)

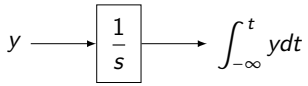
$$y(t) = \frac{b(s)}{a(s)}u(t) = \frac{b_1 s^{n-1} + \ldots + b_n}{s^n + a_1 s^{n-1} + \ldots + a_n}u(t)$$

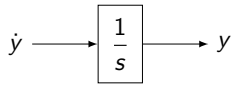


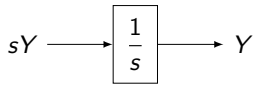
notes

- our goal is now that of converting an ARMA model to a state-space representation.
- the starting point is the transfer function in the Laplace domain.
- the numerator and denominator polynomials define the ARMA model.

## Building block = the integrator (block)






notes

- the integrator block is a fundamental building block for state-space representations.
- the integrator relates to differentiation and integration in the time domain.
- the integrator is key to constructing state variables.

## How do we use integrators?

$$\dddot{y} + a_1\ddot{y} + a_2\dot{y} + a_3 y = b_1 u$$
$$\downarrow$$
$$\dddot{y} = -a_1\ddot{y} - a_2\dot{y} - a_3 y + b_1 u$$



notes

- This shows how to rearrange a higher-order differential equation into a form suitable for state-space representation.
- the highest derivative is expressed as a function of lower derivatives and the input.
- this step is crucial for defining the state variables.

## Towards SS with a useful trick

$$y(t) = \frac{b(s)}{a(s)}u(t) = \frac{b_1 s^{n-1} + \ldots + b_n}{s^n + a_1 s^{n-1} + \ldots + a_n}u(t) \rightarrow \begin{cases} x_n(t) = \dfrac{1}{a(s)}u(t) \\ y(t) = b(s)x_n(t) \end{cases}$$



notes

- We then use the trick of defining an intermediate variable $x_n(t)$ to simplify the conversion process.
- $x_n(t)$ is the output of the denominator dynamics driven by the input $u(t)$.
- this trick separates the AR (denominator) and MA (numerator) parts of the system.

## This is an AR model on $x_n$

$$x_n(t) = \frac{1}{a(s)}u(t) \qquad \Longrightarrow \qquad a(s)x_n(t) = u(t)$$

implies

$$x_n = \frac{1}{s}x_{n-1} \qquad \rightarrow \qquad x_{n-1} = sx_n \qquad \rightarrow \qquad \begin{cases} x_{n-1} &= sx_n \\ x_{n-2} &= s^2 x_n \\ \vdots & \\ x_2 &= s^{n-2}x_n \\ x_1 &= s^{n-1}x_n \end{cases}$$



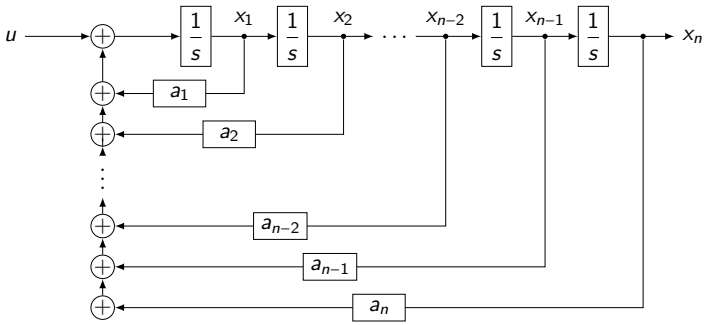
notes

- The intermediate variable $x_n(t)$ leads to the definition of state variables $x_1, x_2, \ldots, x_n$.
- each state variable is a scaled version of the next, with the scaling factor being $s$.
- this step defines the state vector $\boldsymbol{x}$.

## This is an AR model on $x_n$

$$x_n(t) = \frac{1}{a(s)}u(t) \qquad \Longrightarrow \qquad a(s)x_n(t) = u(t)$$

implies



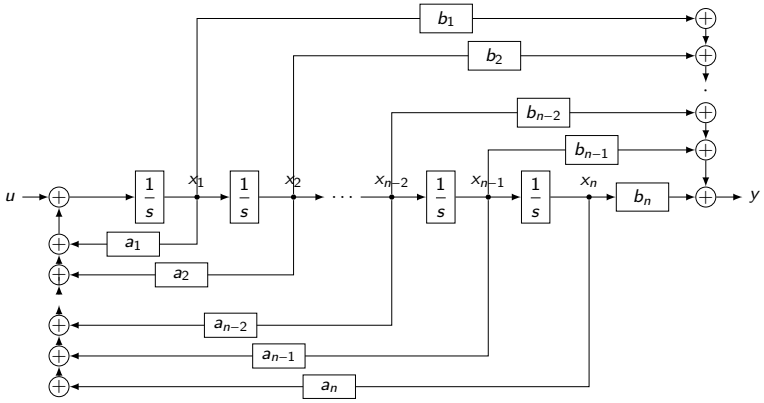



notes

- We now use a block diagram to illustrate the relationship between the state variables.
- the state variables are interconnected through integrators.
- this structure is the foundation of the state-space representation.

## Completing the picture (a MA from $x_n$ to $y$)

$$y(t) = b(s)x_n(t) = b_1x_1(t) + \ldots + b_nx_n(t)$$






notes

- the output $y(t)$ is constructed as a linear combination of the state variables.
- the coefficients of the linear combination are the numerator coefficients $b_1, b_2, \ldots, b_n$.
- this step completes the state-space representation.

## From concepts to formulas

$$\begin{cases} y(t) = b_1 x_1(t) + \ldots + b_n x_n(t) \\ \dot{x}_1(t) = -a_1 x_1(t) - \ldots - a_n x_n(t) + u(t) \\ \dot{x}_i(t) = x_{i-1}(t) \end{cases} \rightarrow \begin{cases} \dot{\boldsymbol{x}} &= A\boldsymbol{x} + Bu \\ y &= C\boldsymbol{x} + Du \end{cases}$$

$$\dot{\boldsymbol{x}} := \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \vdots \\ \dot{x}_n \end{bmatrix} = \begin{bmatrix} -a_1 & -a_2 & \ldots & \ldots & -a_n \\ 1 & 0 & \ldots & \ldots & 0 \\ 0 & 1 & 0 & \ldots & 0 \\ & \ddots & \ddots & \ddots & \\ & & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} u$$



notes

- This presents the final state-space equations in matrix form.
- the structure of the $A$ matrix is then in control canonical form.
- the $B$ vector has a single non-zero entry, corresponding to the input $u(t)$.

## And $y$?

$$\begin{cases} y(t) = b_1 x_1(t) + \ldots + b_n x_n(t) \\ \dot{x}_1(t) = -a_1 x_1(t) - \ldots - a_n x_n(t) + u(t) \\ \dot{x}_i(t) = x_{i-1}(t) \end{cases} \rightarrow \begin{cases} \dot{\boldsymbol{x}} &= A\boldsymbol{x} + Bu \\ y &= C\boldsymbol{x} + Du \end{cases}$$

$$y = \begin{bmatrix} b_1 & b_2 & b_3 & \ldots & b_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}$$



notes

- The output equation is constructed from the state variables.
- the $C$ matrix contains the numerator coefficients $b_1, b_2, \ldots, b_n$.
- this step completes the state-space representation.

## From ARMA to state space (in Control Canonical Form)

$$
\begin{cases}
\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \vdots \\ \dot{x}_n \end{bmatrix} = \begin{bmatrix} -a_1 & -a_2 & \dots & \dots & -a_n \\ 1 & 0 & \dots & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ & \ddots & \ddots & \ddots & \\ & & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} u \\
y = \begin{bmatrix} b_1 & b_2 & b_3 & \dots & b_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}
\end{cases}
$$



notes

- The state-space representation in control canonical form.
- the structure of the $A$ matrix becomes upper Hessenberg with a diagonal of ones.
- this form is particularly useful for control design and analysis, you will see it very often.

## Matlab / Python implementation

```
[A, B, C, D] = tf2ss([b1 b2 .. bn], [1 a1 a2 .. an])
```



notes

- the MATLAB/Python function `tf2ss` is used for converting transfer functions to state-space form.
- the input arguments are the numerator and denominator coefficients of the transfer function.
- this function automates the process of deriving the state-space matrices.
- you can use this function to verify your hand calculations only for small examples, at work don't do computations by hand

## Summarizing

**Determine** the state space structure of an LTI system starting from an ARMA ODE

- there are some formulas, that you may simply know by heart, or that you may want to understand
- for understanding there is the need to get how the transformations work, and what is what
- likely the most important point is that to go from ARMA to SS the (likely) most simple strategy is to build the states as a chain of integrators, and ladder on top of that



notes

- you should now be able to do this, following the pseudo-algorithm in the itemized list

Most important python code for this sub-module



notes

## These functions have also their opposite, i.e., tf2ss

- `https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.ss2tf.html`
- `https://python-control.readthedocs.io/en/latest/generated/control.ss2tf.html`



notes

- in the references you will see much more information than what is given in this module

## Self-assessment material



notes

## Question 1

What is the role of $(sI - A)^{-1}$ in the derivation of the transfer function from a state-space model?

**Potential answers:**

I: **(wrong)** It represents the output matrix $C$.
II: **(correct)** It is used to solve for the state vector $X(s)$ in the Laplace domain.
III: **(wrong)** It defines the input matrix $B$.
IV: **(wrong)** It is the Laplace transform of the state transition matrix.
V: **(wrong)** I do not know

**Solution 1:**

$(sI - A)^{-1}$ is used to solve for the state vector $X(s)$ in the Laplace domain. It allows us to express $X(s)$ in terms of the input $U(s)$, which is then used to derive the transfer function.



notes

- see the associated solution(s), if compiled with that ones :)

## Question 2

What is the structure of the $A$ matrix in the control canonical form of a state-space model?

**Potential answers:**

I: **(correct)** An upper Hessenberg matrix with a lower diagonal of ones and coefficients on the first row from the denominator polynomial.
II: **(wrong)** A diagonal matrix with the eigenvalues of the system.
III: **(wrong)** A lower triangular matrix with zeros on the diagonal.
IV: **(wrong)** A symmetric matrix with off-diagonal elements equal to zero.
V: **(wrong)** I do not know

**Solution 1:**

The $A$ matrix in control canonical form is an upper Hessenberg matrix with a lower diagonal of ones and coefficients from the denominator polynomial. This structure is particularly useful for control design and analysis.



notes

- see the associated solution(s), if compiled with that ones :)

## Question 3

What is the purpose of the integrator block in the conversion from ARMA to state-space models?

**Potential answers:**

I: **(wrong)** To differentiate the input signal.
II: **(wrong)** To invert the Laplace transform of the output.
III: **(correct)** To construct the state variables as a chain of scaled integrators.
IV: **(wrong)** To compute the determinant of the state matrix.
V: **(wrong)** I do not know

**Solution 1:**

The integrator block is used to construct the state variables as a chain of scaled integrators. This allows us to define the state vector $\boldsymbol{x}$ and build the state-space representation.



notes

- see the associated solution(s), if compiled with that ones :)

## Question 4

What does the transfer function $H(s) = \dfrac{Y(s)}{U(s)}$ represent in the context of state-space models?

**Potential answers:**

I: **(wrong)** The state transition matrix.
II: **(wrong)** The input matrix $B$.
III: **(wrong)** The determinant of the state matrix.
IV: **(correct)** The relationship between the input $U(s)$ and the output $Y(s)$ in the Laplace domain.
V: **(wrong)** I do not know

**Solution 1:**

The transfer function $H(s) = \dfrac{Y(s)}{U(s)}$ represents the relationship between the input $U(s)$ and the output $Y(s)$ in the Laplace domain. It is derived from the state-space model and encapsulates the system's dynamics.



notes

- see the associated solution(s), if compiled with that ones :)

## Question 5

In the context of SISO systems, what are the dimensions of the matrices $C$ and $B$ in a state space representation?

**Potential answers:**

I: **(wrong)** $C$ is a scalar, and $B$ is a vector.
II: **(correct)** $C$ is a row vector, and $B$ is a column vector.
III: **(wrong)** $C$ is a square matrix, and $B$ is a scalar.
IV: **(wrong)** $C$ is a column vector, and $B$ is a row vector.
V: **(wrong)** I do not know

**Solution 1:**

In SISO systems, $C$ is a row vector ($1 \times n$), and $B$ is a column vector ($n \times 1$). This is because $C$ maps the state vector to the output, and $B$ maps the input to the state vector.



notes

- see the associated solution(s), if compiled with that ones :)

## Question 6

Given the state-space matrices $A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$, $B = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, $C = \begin{bmatrix} 1 & 0 \end{bmatrix}$, and $D = \begin{bmatrix} 0 \end{bmatrix}$, what is the transfer function $H(s)$?

**Potential answers:**

I: **(correct)** $H(s) = \dfrac{s-4}{s^2-5s-2}$
II: **(wrong)** $H(s) = \dfrac{s-1}{s^2-5s-2}$
III: **(wrong)** $H(s) = \dfrac{s+3}{s^2-5s-2}$
IV: **(wrong)** $H(s) = \dfrac{s-2}{s^2-5s-2}$
V: **(wrong)** I do not know

**Solution 1:**



The transfer function is $H(s) = \dfrac{s-4}{s^2-5s-2}$. This is derived by solving the state-

notes

- see the associated solution(s), if compiled with that ones :)

## Recap of sub-module "state space from ARMA (and viceversa)"

- one can go from ARMA to state space and viceversa
- we did not see this, but watch out that the two representations are not equivalent: there are systems that one can represent with state space and not with ARMA, and viceversa
- typically state space is more interpretable, and tends to be the structure used when doing model predictive control



notes

- the most important remarks from this sub-module are these ones