Systems Laboratory, Spring 2025

Damiano Varagnolo –

Recap of sub-module "Is this function a solution of this ODE?"

- a function is a solution of an ODE if it satisfies the equation for all values in its domain
- initial conditions are necessary to uniquely determine a solution

Recap of sub-module “which type of ODE is this one?”

- an ODE can be classified based on its structural properties (linearity, autonomy, time-invariance)
- linearity requires both additivity and homogeneity
- autonomous systems evolve solely based on their state, while non-autonomous systems depend on external inputs
- time-invariant systems have dynamics that do not explicitly depend on time, while time-varying systems do
- graphical representations help in identifying these properties visually

Recap of sub-module "compute the equilibria of the system"

- Equilibria in dynamical systems correspond to points where the system's state does not change over time.
- Autonomous time-varying ODEs can have equilibria, but their location may vary with time.
- Some dynamical systems may not have equilibria, particularly if they involve unbounded growth.
- Non-autonomous LTI ODEs can have equilibria only if the input $u(t)$ remains constant over time.

Recap of sub-module "building and interpreting phase portraits"

- A phase portrait is a graphical representation of a dynamical systems trajectories in state space.
- Phase portraits provide qualitative insight into system behavior without requiring explicit solutions.
- First-order systems have a one-dimensional state space, while second-order systems require two dimensions, etc.

Recap of sub-module “what is control”

- designing a controller means designing an algorithm that transforms information into decision
- there are several types of controllers, each with pros and cons
- taking decisions (i.e., actuating u) means modifying the dynamics of the system

Recap of sub-module "how to linearize an ODE"

- linearization requires following a series of steps (see the summary above)
- the model that one gets in this way is an approximation of the original model
- having a graphical understanding of what means what is essential to remember how to do things
- better testing a linear controller before a nonlinear one

Recap of sub-module "when is linearizing meaningful"

- be careful when using a linearized system - be always aware of where it comes from

Recap of sub-module
“what is the superposition principle, and what does it imply”

- superposition principle helps logically separating specific causes into specific effects
- linear ODEs $\implies$ superposition principle
- superposition principle $\implies$ "whole $=$ free $+$ forced"
- nonlinear systems WON’T satisfy this principle!

Recap of sub-module "what is an impulse response"

- impulse responses are directly connected to step responses
- actually this connection is valid only if the system is LTI

Recap of sub-module "1D convolution in continuous time"

- convolution is an essential operator, since it can be used for LTI systems to compute forced responses
- its graphical interpretation aids interpreting impulse responses as how the past inputs contribute to current outputs

Recap of sub-module "computing free evolutions and forced responses of LTI systems"

- finding such signals require knowing a couple of formulas by heart
- partial fraction decomposition is king here, one needs to know how to do that

Recap of sub-module “state space representations”

- a set of variables is a state vector if it satisfies for that model the separation principle, i.e., the current state vector “decouples” the past with the future
- state space models are finite, and first order vectorial models

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

Recap of sub-module "Connections between eigendecompositions and free evolution in continuous

- the eigenvalues of the system matrix A give the growth / decay rates of the modes $e^{\alpha t}$ of the free evolution of the system
- along eigenspaces, the trajectory of the free evolution is "simple", i.e., aligned with that eigenspace
- the kernel of the system matrix gives us the equilibria corresponding to $u = 0$

Recap of sub-module “explain and determine the marginal stability of an equilibrium”

- marginal stability / simple stability is the property that answers the question “can I bound the evolutions, i.e., arbitrarily constrain them to do not get “too far” from an equilibrium by starting opportunely closeby the original equilibrium?
- an equilibrium is marginally stable or not depending on whether one is able to ‘win’ the ‘choose your neighborhood’ game
- phase portraits are very interpretable, to this regards
- there is a sort of “downgrading” phenomenon that happens here: one has to have *all* the trajectories behaving in a good way to have a certain property. One not behaving is enough for the “downgrading” of the equilibrium

Recap of sub-module “explain and determine the convergence properties of an equilibrium”

- convergence is disconnected from “marginal stability”, since in general one may have one case and not the other, and viceversa, or both, or none
- the concept of convergence focuses on the limit behavior, ignoring the transient

Recap of sub-module "explain what BIBO stability means"

- BIBO stability means "a bounded input must imply a bounded output"
- it is a concept that in general it is disconnected to that of marginal stability / convergence of an equilibrium

Recap of the module “BIBO stability for LTI systems”

- for LTI systems BIBO stability is equivalent to the absolute integrability of the impulse response
- for ARMA systems BIBO stability is equivalent to having the impulse response so that all its exponential terms are vanishing in time
- for nonlinear systems one shall use more advanced tools that will be seen in later on courses

Recap of sub-module
"Is this time series a solution of this recurrence relation?"

- a function is a solution of a RR if it satisfies the equation for all values in its domain
- initial conditions are necessary to uniquely determine a solution

Recap of sub-module “how to get a RR from an ODE”

- there are several ways of solving ODEs in a computer
- Euler is the most simple one, but it does not work well with stiff ODEs
- more advanced schemes have better numerical properties

Recap of sub-module "which type of RR is this one?"

- a RR can be classified based on its structural properties (linearity, autonomy, time-invariance)
- linearity requires both additivity and homogeneity
- autonomous systems evolve solely based on their state, while non-autonomous systems depend on external inputs
- time-invariant systems have dynamics that do not explicitly depend on time, while time-varying systems do
- graphical representations help in identifying these properties visually

Recap of sub-module "compute the equilibria of the system"

- Equilibria in dynamical discrete time systems correspond to points where the system's state does not change over time.
- Autonomous time-varying RRs can have equilibria, but their location may vary with time.
- Some dynamical systems may not have equilibria, particularly if they involve unbounded growth.
- Non-autonomous LTI RRs can have equilibria only if the input $u(t)$ remains constant over time.

Recap of sub-module "building and interpreting phase portraits for RRs"

- A phase portrait is a graphical representation of a dynamical systems trajectories in state space.
- Phase portraits provide qualitative insight into system behavior without requiring explicit solutions.
- First-order systems have a one-dimensional state space, while second-order systems require two dimensions, etc.
- The smaller the sampling period T, the closer the discrete-time phase portraits is to the one one would get from the continuous time version of the system

Recap of sub-module “what is control”

- designing a controller means designing an algorithm that transforms information into decision
- there are several types of controllers, each with pros and cons
- taking decisions (i.e., actuating u) means modifying the dynamics of the system

Recap of sub-module "how to linearize a RR"

- linearization requires following a series of steps (see the summary above)
- the model that one gets in this way is an approximation of the original model
- having a graphical understanding of what means what is essential to remember how to do things
- better testing a linear controller before a nonlinear one

Recap of sub-module “when is linearizing meaningful”

- be careful when using a linearized system - be always aware of where it comes from

Recap of sub-module "what is the superposition principle, and what does it imply"

- superposition principle helps logically separating specific causes into specific effects
- linear RRs $\Longrightarrow$ superposition principle
- superposition principle $\Longrightarrow$ "whole $=$ free $+$ forced"
- nonlinear systems WON'T satisfy this principle!

Recap of sub-module “what is an impulse response”

- impulse responses are directly connected to step responses
- actually this connection is valid only if the system is LTI

Recap of sub-module “1D convolution in discrete time”

- convolution is an essential operator, since it can be used for LTI systems to compute forced responses
- its graphical interpretation aids interpreting impulse responses as how the past inputs contribute to current outputs

Recap of sub-module "computing free evolutions and forced responses of LTI systems"

- finding such signals require knowing a couple of formulas by heart
- partial fraction decomposition is king here, one needs to know how to do that

Recap of sub-module "state space representations"

- a set of variables is a state vector if it satisfies for that model the separation principle, i.e., the current state vector "decouples" the past with the future
- state space models are finite, and first order vectorial models

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

Recap of sub-module

“Connections between eigendecompositions and free evolution in discrete tim

- the eigenvalues of the system matrix A give the growth / decay rates of the modes λ^k of the free evolution of the system
- along eigenspaces, the trajectory of the free evolution is “simple”, i.e., aligned with that eigenspace
- the kernel of the system matrix gives us the equilibria corresponding to $u = 0$

Recap of sub-module
"explain and determine the marginal stability of an equilibrium"

- marginal stability / simple stability is the property that answers the question "can I bound the evolutions, i.e., arbitrarily constrain them to do not get "too far" from an equilibrium by starting opportunely closeby the original equilibrium?
- an equilibrium is marginally stable or not depending on whether one is able to 'win' the 'choose your neighborhood' game
- phase portraits are very interpretable, to this regards
- there is a sort of "downgrading" phenomenon that happens here: one has to have *all* the trajectories behaving in a good way to have a certain property. One not behaving is enough for the "downgrading" of the equilibrium

Recap of sub-module
“explain and determine the convergence properties of an equilibrium”

- convergence is disconnected from “marginal stability”, since in general one may have one case and not the other, and viceversa, or both, or none
- the concept of convergence focuses on the limit behavior, ignoring the transient

Recap of sub-module “explain what BIBO stability means”

- BIBO stability means “a bounded input must imply a bounded output”
- it is a concept that in general it is disconnected to that of marginal stability / convergence of an equilibrium

Recap of the module “BIBO stability for LTI systems”

- for LTI systems BIBO stability is equivalent to the absolute summability of the impulse response
- for ARMA systems BIBO stability is equivalent to having the impulse response so that all its exponential terms are vanishing in time
- for nonlinear systems one shall use more advanced tools that will be seen in later on courses

Recap of sub-module "The Role of Filtering in Feedback Control Systems"

- Sensors are noisy and have limitations.
- Filtering is used to suppress measurement noise.
- Strong filtering slows down system response.
- Good control design carefully balances filtering strength and responsiveness.

Recap of sub-module
“Performance indexes for filtering measurement noise”

- Filters are evaluated using performance indexes.
- Main indexes are: noise reduction efficiency, signal distortion, response delay, stability, and computational cost.
- Trade-offs between these indexes are inevitable.

Recap of sub-module
"Design and implement low-pass and weighted averaging filters"

- Low-pass filters allow smooth signals to pass while reducing noise.
- Weighted averaging filters assign importance to past samples.
- Moving average is the simplest example of a low-pass filter.
- Python makes it easy to simulate and test filters.

Recap of sub-module "Filtering Lab: Noise Reduction and Outlier Rejection"

- Filter choice depends on signal features (noise band, outliers, quantization).
- Always check frequency/impulse responses for unintended effects.
- Non-linear filters (median) handle outliers; linear filters (Butterworth) handle noise.

Recap of sub-module "Introduction to System Identification"

- Model-based control requires accurate models
- System identification builds models from data
- There are several tools to estimate model parameters, in this course we only scratch the surface

Recap of sub-module “Least squares estimators”

- Least squares aims to minimize the squared residuals between model predictions and observed data
- The geometric interpretation views system identification as finding the closest point on a model manifold to measurement vectors
- Normal equations provide an analytical solution for unconstrained linear least squares problems through $\Phi^T \Phi \theta = \Phi^T y$
- The pseudoinverse generalizes solutions for rank-deficient systems and connects with singular value decomposition
- Existence and uniqueness of LS solutions depend on hypothesis space topology and model structure identifiability
- Constrained LS problems require different approaches than normal equations when parameters must satisfy domain restrictions

Recap of sub-module "Ill conditioning"

- Ill-posed problems may lack a solution, have multiple solutions, or be highly sensitive to small changes in data
- Ill-conditioned problems have a solution, but it is numerically unstable and highly sensitive to input errors
- The condition number of a matrix quantifies the degree of ill-conditioning; a high condition number indicates poor numerical stability
- In system identification, slowly varying or insufficiently rich input signals can lead to ill-conditioning
- Regularization techniques can mitigate the effects of ill-conditioning by introducing stability through additional constraints
- Choosing appropriate input signals is critical to ensuring well-posed and well-conditioned identification problems
- Understanding the structure and properties of the data matrix (e.g., U in least squares problems) is essential to diagnose ill-conditioning

Recap of sub-module "Regularization"

- adding regularization and non-L2 costs noticeably extends capabilities of estimators, at the cost though of introducing some hyperparameters that need to be tuned too from the data

Recap of module "Visualizing systems with block schemes"

- block representations are alternative representations
- they enable graphical coding, that is used quite a lot in big companies

Recap of module "Introduction to Open-Loop Controller Design"

- open-loop control is structurally simple but not very robust

Recap of module "Introduction to closed-loop controller design"

- feedback control is more promising, but requires designing more things compared to open loop

Recap of sub-module "PID Controllers"

- Pole placement allows us to achieve desired dynamics
- PID gains shift the closed-loop poles
- Match desired characteristic polynomial with actual one
- Use symbolic or numerical tools to solve for K_P, K_I, K_D

Recap of sub-module “Full state feedback control”

- full state feedback enables placing the poles wherever one wants
- with respect to PID it has more flexibility
- this comes with the cost of having a sufficiently accurate model (and that the model can be written in control canonical form, something that is not always guaranteed!)

Recap of sub-module "Introduction to Luenberger observers"

- one may estimate the states of a system by means of making the estimated state be so that it dynamically matches the measured values
- this strategy though is as valid as the model is, as a description of the system
- the situation is though in practice not as simple as seen here - indeed the here presented case is for "fully observable" systems (a concept that you'll see in systems theory) and thus not applicable all the times (but extensible to!)

Recap of sub-module "Tuning MPC for LTI Systems"

- MPC performance depends on careful parameter selection
- Prediction horizon affects stability and computation
- Weight matrices balance state vs control objectives
- Systematic tuning follows an iterative procedure