EEET 3046 Control Systems (2020)

Lecture 1: Introduction

Dr. Yee Wei Law ⟨yeewei.law@unisa.edu.au⟩

Contents

Engineering students often speak of control engineering (not referring to this course!) as their favorite subject. This is because the application of control often leads to physical actuation that the students can readily see and understand. Unfortunately, it is also not uncommon for students to underappreciate the value of the subject, because somehow the mathematical nature of the subject often leads to the misconception that the kind of control taught at a university is impractical. The great Prof. Rudolf Kalman (see Figure 1) once said [GA15, p. 1]:

> Once you get the physics right, the rest is mathematics.

In the same vein, authors of the popular aeronautics text [JK93] wrote

> The kernel of our viewpoint is that effective control laws for mechanical systems are best designed by one who understands both the basic mechanics of the system under consideration and the control methodology being used to design the control law.

In other words, *both* physics *and* mathematics are essential knowledge in control engineering. Control is decidedly multidisciplinary, where mathematics is the underlying glue. As such, control is supposed to have a strong mathematical flavor. Renown experts Åström and Kumar [ÅK14, p. 33] have this to say:

Control is a field whose progress has been punctuated by several key theoretical contributions. These have involved a variety of mathematical sub-disciplines, such as complex analysis, differential equations, probability theory, differential geometry, optimization and graph theory. As such, control is currently one of the most mathematized fields of engineering.

Figure 1: Prof. Rudolf Kalman is a Hungarian-born American control engineer most famous for the creation of the Kalman filter, one of the essential technologies in the Apollo program and a wide spectrum of applications that involve state estimation. As one of the pioneers of modern control, he has created a deep body of knowledge, a fraction of which is covered in the follow-up course EEET 4071 Advanced Control. Throughout his career, he has received numerous awards, the most recent of which being The President's National Medal of Science in 2009.

1 What is control?

Consider a process/system/plant with inputs and outputs. *Control* is the action of adjusting the inputs to obtain desirable outputs. The entity performing the adjustment of the inputs is called the *controller*.

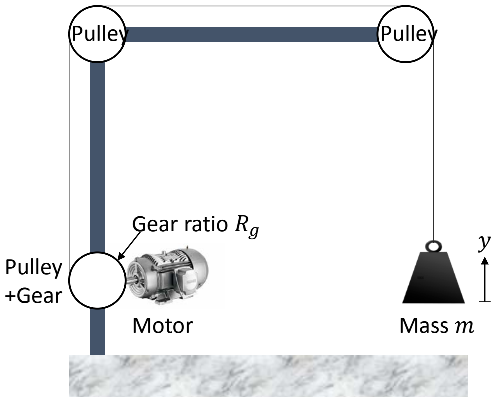

Figure 2: A motor lifting a weight.

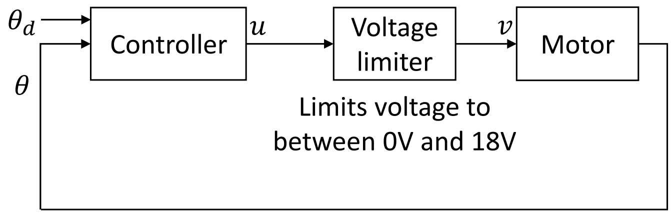

Figure 3: Block diagram of the motor control system in Figure 2.

Example 1

Consider the problem of controlling a DC motor to lift a weight to a certain height (see Figure 2). The motor has to rotate the amount of angle corresponding to the target height h. This is an example of a position control problem. Ignoring the mechanical transmission for simplicity, the required amount of angle θ_d is given by

$$\theta_d = h/r,$$

where r is the radius of the gear attached to the motor. Suppose the motor takes a voltage between 0 and $V_{\max}$, and can reach a maximum speed of $\omega_{\max}$. Imagine trying to come up with a control scheme:

- Since angle equals speed times time, can we just apply $V_{\max}$ to the motor for $t = \theta_d/\omega_{\max}$ amount of time? This does not work because the motor cannot change speed instantaneously, whether from 0 to $\omega_{\max}$ or vice versa. More importantly, there is *no* simple algebraic relationship between the input voltage v and the motor speed ω. In fact, ω is related to v via the ordinary differential equation (ODE):

$$v = Ri + L\frac{\mathrm{d}i}{\mathrm{d}t} + K_b\omega, \tag{1}$$

  where R is the armature resistance, i is the armature current, L is the armature inductance, and K_b is the back emf constant.

- Alternatively, if we try the naive algorithm in Listing 1, then we will end up with an angle response that looks like Figure 4.

Listing 1: Naive algorithm for motor position control.

```
if theta < theta_d
  u = Vmax;
elseif theta >= theta_d
  u = 0;
end
```

  The result in Figure 4, obtained using motor data from the 2001 Maxon A-max 32 datasheet[a], is interesting because it shows that even such a simple algorithm is able to track θ_d, which is 800 rad in this example. However, there is a catch: the simulation is in continuous time, while in practice the controller typically runs in discrete time (longer reaction time). Moreover, the chattering pattern in the motor voltage causes transient spikes, speeds up wear and tear, and consequently often leads to premature burnout of the motor.

In a way, we have been trying to reinvent control. The classical control approach is to implement a proportional-integral (PI) or proportional-integral-derivative (PID) controller. As shown in Figure 5, the motor voltage at the output of a PI controller with an integrator clamp is chattering-free. We will learn about the PI controller and the integrator clamp (an anti-windup measure) later in the course. We should keep in mind that control was invented in the pre-computer age, where the pioneers did not think in terms of algorithms like Listing 1; they also did not have the luxury of trying out code until something works.

[a] http://docs-asia.electrocomponents.com/webdocs/0111/0900766b80111e40.pdf

As living organisms, we are naturally evolved controllers with the ability to control/regulate our heart rate, body temperature, and countless other physiological parameters. As humans, we are

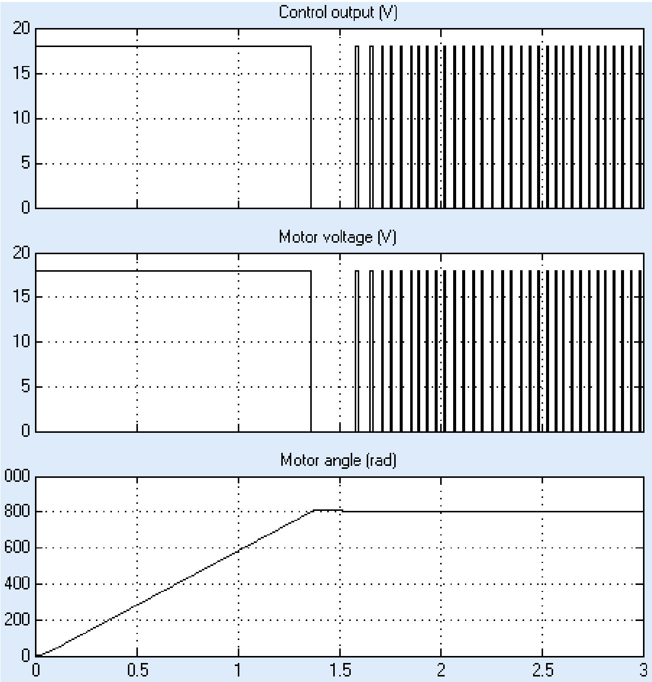

Figure 4: Angle response of the motor controlled using Listing 1.

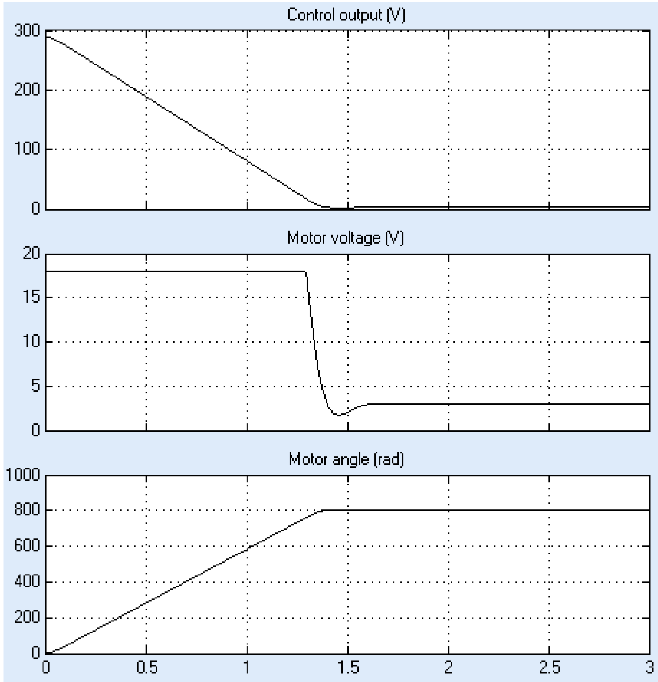

Figure 5: Angle response of the motor controlled using proportional-integral (PI) control with integrator clamp.

used to performing control subconsciously and consciously. For example, control can be as simple as adjusting the speaker volume of our audio equipment. For another example, when we shower, we control the water temperature by trying to mix the hot and cold water in the right proportions. The question is how does this control take place? The answer to this question consists of two parts: *learning* and *feedback*.

- *Learning* gives the controller (e.g., us) the ability to predict the output (e.g., shower temperature) corresponding to the supplied inputs (e.g., hot and cold water flow rates); such an ability exists in the form of a **model** of the controlled process.

- *Feedback* allows the controller to
  - observe via the sensors the critical parameters of the process (e.g., shower temperature), compare them with the desired values (e.g., desired temperature),
  - compute the corrective action (e.g., hot and cold water flow rates) as a function of the comparison result and the process model, and
  - actuate the process with the corrective action (see Figure 6).

The concept of the feedback loop of sensing, computation and actuation, as depicted in Figure 6, is central to control [rM08, Sect. 1.2].
In Figure 6, the controller is designed based on *some* model of the plant — the difference between this model and the *true* model is called *model uncertainty*. Significant model uncertainty or disturbance can potentially derail the operation of the system — the keyword is actually "destabilize", meaning

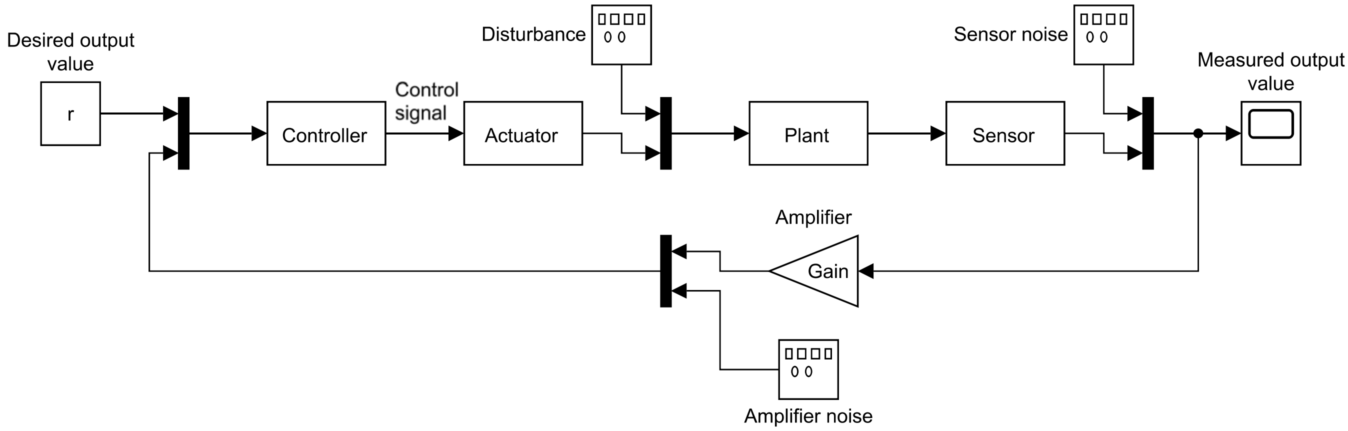

Figure 6: A typical feedback control loop [GGS00, Fig. 2.12]. The desired output value is commonly called the *reference input* or *set-point* or *command*. The control signal is also called the *control action* or *control output*. A disturbance is any uncontrollable external input to the plant. In the shower temperature control analogy, the desired output value is the desired shower temperature, the plant is the water heater and mixer, and the controller is the shower user. Can you identify the actuator and the sensor?

making the system diverge from the control objective, but we will only define "stability" formally in a later lecture.

Example 2

Consider the control of the shower temperature:

- The scenario where we mistake the cold water tap as the hot water tap and vice versa is an extreme example of model uncertainty. Under this extreme model uncertainty, we can never hope to even get close to our desired shower temperature.

- As an example of uncontrollable external disturbance, imagine your housemate flushing the toilet while you are showering — this disturbance to the shower temperature is outside your control. In the presence of this disturbance, you either frantically readjust the shower temperature to counter the disturbance, or wait it out. In practice, controllers are always designed to counter disturbances to maintain tracking of the set-point.

At this point, enough concepts have been introduced for the following definition to make sense:

Definition: The fundamental control problem [GGS00, Definition 2.1]

... is to find a technically feasible way to act on a given process so that the process behaves, as closely as possible, to some desired behavior. Furthermore, this approximate behavior should be achieved in the face of *uncertainty of the process* and in the presence of *uncontrollable external disturbances* acting on the process.

1.1 Model-based vs. intelligent control

Clearly, the process model plays an important role in control. A model can be learnt either offline or online. Whenever a control engineer designs a controller based on a process model, essentially the controller "learns" the model offline, although typically we do not refer to this procedure as learning.

A model can be represented in many ways; for example, our mental model of the shower temperature is decidedly different from any computer-implementable model. Nevertheless, picture trying to implement our mental model using IF-THEN-ELSE statements in a conventional programming language:

Listing 2: Imaginary model of the shower temperature implemented in a conventional programming language.

```
if cold_water_flow_rate < 0.1 && hot_water_flow_rate < 0.1
  shower_temperature = 24;
elseif cold_water_flow_rate < 0.1 && hot_water_flow_rate < 0.2
  shower_temperature = 25;
elseif cold_water_flow_rate < 0.1 && hot_water_flow_rate < 0.3
  shower_temperature = 26;
elseif ...
  ...
end
```

For Listing 2 to be able to represent the shower temperature accurately, one may assume we only need sufficiently many IF-THEN-ELSE statements. The fact is, regardless of the number of IF-THEN-ELSE statements, Listing 2 cannot model the temperature *dynamics* — a pair of flow rate values can correspond to a range of temperature values because temperature change is not instantaneous! In fact, shower temperature is an example of a *dynamical system*, i.e., a system that evolves with time.

Realizing the rate of change of the shower temperature is a function of the hot and cold water flow rates, we should model the shower temperature with a differential equation, e.g.,

$$\dot{y} = k(q_h - q_c)e^{-\lambda t}, \tag{2}$$

where y is the shower temperature, q_h is the hot water flow rate, q_c is the cold water flow rate, k and λ are constants. Although not necessarily true, Eq. 2 makes sense because it says the temperature gradient increases/decreases exponentially over time when we hold the flow rate differential $q_h - q_c$ constant.

Quiz 1

Given the initial temperature is $y(0) = y_0$, $q_h - q_c$ is constant, show that

$$y(t) = y_0 + \frac{k}{\lambda}(q_h - q_c)(1 - e^{-\lambda t}). \tag{3}$$

Note the curve of Eq. (3) goes from y_0 to $y_0 + \frac{k}{\lambda}(q_h - q_c)$ exponentially.

Most processes in engineering are dynamical systems that can readily be modeled using differential equations. For example, Newton's second law itself is a second-order differential equation.

We can classify controllers into two broad categories:

- **Model-based controllers**: These are controllers designed based on a set of differential equations that model the dynamics of the process to be controlled. In the next lecture, we will learn how to model some basic systems using differential equations.
- **Intelligent controllers**: Knowing that humans do not keep mental models in the form of Listing 2 or differential equations, the question of how we actually learn to control shower temperature or other processes remains. Observations on how we learn controlling by trial-and-error, imprecise reasoning, etc., have given birth to a class of bio-inspired controllers called the *intelligent controllers*. Rather than requiring a set of differential equations, intelligent controllers apply computational intelligence techniques, such as fuzzy logic, neural networks and evolutionary computation, to the learning and control of processes. Intelligent controllers are useful whenever the process to be controlled is difficult to model possibly due to
  - the process being too complex (nonlinear / distributed / incomplete / stochastic) to model mathematically; or
  - the process model being too physically challenging or costly to evaluate [Sid13].

Intelligent controllers may seem like an easy way out when the goal is "good enough" rather than optimum performance, since it does not require the control engineer to mathematically model the process, but most intelligent controllers do not provide any stability or performance guarantee, and their design requires extensive trial-and-error. Therefore, model-based controllers are preferred whenever the process model is attainable, and this is why the hybrid approach of *model-based intelligent control* exists to enhance model-based controllers with intelligence techniques. On the other hand, any stability or performance guarantee is only as good as the model itself.

The focus of this course is how to design and analyze model-based controllers using the *classical control* approach (more on this in Sect. 4).

2 Why study control?

Applications of control are ubiquitous. In fact, nature discovered feedback control (think homeostasis), before we put control systems everywhere, from household electronics to space-faring vessels. Control is the key to automation. Virtually all industries use some form of control. Expertise in control will help an engineer, especially a mechatronic engineer, go a long way.

> **Definition: Mechatronic engineering [dSKML15]**
>
> ... is the synergistic application of mechanics, electronics, control and software engineering to the development of electromechanical systems through an integrated design approach.

In the advent of the Internet of Things, more mechanical objects are getting connected to the Internet (e.g., cars), and more virtual objects are acquiring a mechanical embodiment (e.g., robot companions). A direct upshot of this trend is that mechatronic engineering skills — of which control is an essential part — will become a highly sought after commodity. Figure 7 shows two sample job ads seeking control expertise.

Control Systems Engineer

Experienced Systems Engineer

ZI-ARGUS is a market leading independent Systems Integrator providing total solutions in Process Control, Automation, Engineering, Process Consultancy and Plant Optimization. Our highly skilled and experienced specialists execute turn-key Industrial Automation projects throughout the Asia-Pacific region with offices in Australia, Indonesia, Philippines and Thailand.

Based from our Melbourne office, you will work in collaboration with skilled engineers and third parties to deliver high end solutions to a wide range of customers across many industries. Working on medium to large Industrial Automation / BMS systems projects, you will be responsible for system design, development, testing, validation, commissioning, training, optimisation and general post project service. You will also take the lead on projects, assist with tender quotations and mentor developing engineers.

To be considered for this role you must have:

• A degree in engineering such as electrical, mechatronics, electronic

• At least five years of PLC / SCADA systems development experience across several platforms

• Experience commissioning medium to large systems (> 3000 IO) across several industries from site establishment, IO testing through to process testing / final commissioning and system handover

• The ability to work well within a team environment, enjoy collaboration and shared learning

• Good communication and interpersonal skills with a proven ability to work with a diverse range of people at both a technical and non-technical level

• Excellent documentation skills

• Highly motivated with the initiative to take on work independently

Based on our track record of projects already successfully delivered, this role will provide you opportunity, experience, and variety across various industries and processes such as Oil and Gas, Petrochemical, Manufacturing, Food & Beverage, Pharmaceutical, Materials Handling and Smart Buildings.

RF Electronic Warfare Systems and Simulation Engineer

Simbiant is an Adelaide-based, Australian-owned company whose core business lies in the development of complex real world defence systems and technically complex commercial solutions. Simbiant have brought together experts in Software, Hardware, Electronics, Mathematics, Physics, Commercial and Military Systems, who work together to provide professional services and products to our clients. We are a small company with a can-do culture, and more than enough R&D projects to get involved with to fulfil your thirst for achievement.

Simbiant is currently seeking to fill Senior roles in Electronic Warfare Systems. This is a branch of defence science which incorporates many skillsets, including RF Electronics, Programming, Physics, Hardware, Cyber, Mechatronics, Control systems, simulation and real-time programming.

The positions will form part of a team in a collaborative environment to develop RF EW systems together with DSTG. The ideal candidate will show an interest and understanding of radio frequency propagation, exposure to programming in languages such as C/C++, exposure to modelling tools such as MATLAB and SIMULINK, and an interest or experience in military systems and platforms.

The project involves using state of the art hardware to perform physics simulations that until recently have been impossible to easily achieve in real-time.

What you'll do
Be a member of a tight-knit team providing science and technology support to DST Group. In the first instance, there will be a steep learning curve, where you will come up to speed with modern military threat and countermeasure systems and techniques.

What we are looking for

- Degree in Electronic, Control Engineering, Physics, Mechatronics, Communications technologies or other STEM-based discipline which provides you with the skills we need.
- must have knowledge and experience in processing of RF seeker or radar signals at baseband and IF.
- knowledge and experience using Simulink for modelling of RF seekers or radar
- C++ or similar
- working in, or leading, teams to deliver M&S capability and applications.

Desirable:

- fixed point design and conversion
- FPGA or HDL exposure
- Simulink automatic code generation targets (HDL, C/C++, Simulink Real Time®).

Figure 7: Sample control-related job ads found on `seek.com.au` on 20 December 2016. Note the highlighted keywords.

A large collection of sample control systems/applications are available in

- the course textbook and
- the IEEE Control Systems Society website: `http://ieeecss.org/general/impact-control-technology`.

In Supplementary Lecture A (a different document), we will catch a glimpse of how control is used in various systems and industries. To fully appreciate Supplementary Lecture A though, the basic control terminology introduced in the next section should be understood first.

Attention: Supplementary Lecture A

Supplementary Lecture A is a separate document. Especially for those who need to see the applications to appreciate the course, PLEASE take some time to go through that document. Supplementary Lecture A describes a long list of sample control applications to satisfy the students' curiosity, but do not worry about writing long essays about these applications in the exam.

3 How do we do control?

Model-based control engineering involves the following steps:

- **Modeling**: This is for establishing the mathematical equations, based on domain knowledge, that describe the dynamics of the process to be controlled. This step is usually the most challenging and time-consuming part of the process.

- **Design/synthesis**: This is for investigating ways of constructing controllers with certain properties. This involves
  - determining the suitable types of controllers, and subsequently
  - the values of the controller parameters to achieve the desired response.

- **Analysis**: This is for investigating the impact that a controller has on the target system when they interact in feedback. This involves
  - determining the stability of the system;
  - determining the output of the system, in response to some input;
  - determining the transient response (timing characteristics) and steady-state response (steady-state error) of the system;
  - determining the robustness of the system to disturbances and model error.

  Analysis is typically interleaved with design.

- **Simulation and testing**: After the controller design has been finalized, except for really simple systems, the performance of the control system is typically evaluated by simulation before it is implemented.

- **Implementation and validation**: The control system is realized and validated in hardware and software.

Figure 8 shows a sample control design workflow for aircrafts with flexible structures (e.g., hypersonic aircrafts). The ensuing two subsections explain some basic control terminology to help the readers understand the control design process a bit more.

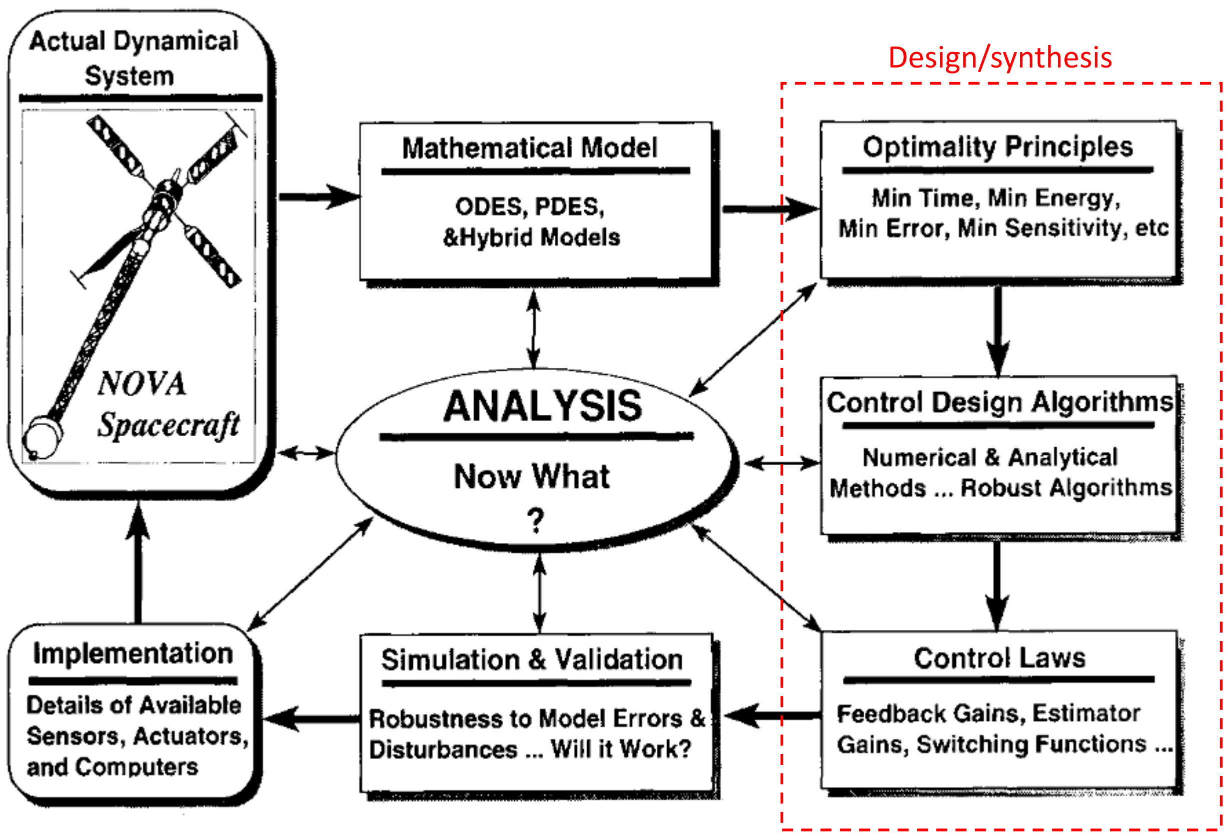

Figure 8: Sample control design workflow for aircrafts with flexible structures [JK93, Figure 1.1]. Most techniques involved in this workflow go beyond the scope of our introductory control course.

3.1 Modeling

A good control system design starts with a good system model. To build a good model, the control engineer needs to be familiar with the terminology, which is the subject of this subsection.

3.1.1 Symbolic vs. numeric

In a model, if the system parameters appear explicitly as symbols in the equations, the model is *symbolic*; otherwise, if the system parameters appear as numbers, then the model is *numeric*. The difference between the two matters, because a numeric model cannot express the role/effect of any system parameter in the system. For example, as a model, Newton's second law is more informative in the symbolic form:

$$\ddot{x} = \frac{f}{m},$$

than in the numeric form, where $m = 1$:

$$\ddot{x} = f.$$

The numeric form captures just one out of infinitely many possible systems. With the numeric form, we cannot tell how the mass (in general, system parameters) affects the dynamics of the system. It is realistic to expect a real-world model to be partly symbolic and partly numeric.

3.1.2 Analytical vs. empirical

A model derived from first principles (physical laws) is an *analytical* model, whereas a model derived by fitting equations to empirically obtained data is an *empirical* model.

If the physics of a system is well understood, an analytical model is clearly preferred. Analytical models are naturally symbolic. However, for most systems, an analytical model can be so complex that a modeling software tool is required to derive it, and such a tool typically provides the model in the numeric form. Nowadays, it is a basic requirement for the software to support joint modeling of components from different engineering domains, e.g., electrical, mechanical, thermodynamic, hydraulic — such a modeling tool is so-called *multi-domain*, or *multi-physics*, or *multi-engineering*. An example of a multi-domain modeling tool is MATLAB. As a part of MATLAB, Simulink facilitates a block diagram approach to modeling. Within Simulink, we can find an assortment of model libraries for modeling different kinds of systems, e.g.,

- the Aerospace Blockset for modeling aerospace systems;
- the Communications System Toolbox for modeling the physical layer of communications systems;
- Simscape Multibody for modeling rigid multibody systems;
- Simscape Electrical (formerly SimElectronics and SimPowerSystems) for modeling electronics and electrical power systems.

Within Simulink, we can use StateFlow to model finite state machines and event-triggered systems. We can even use Simulink to generate computer code from the block diagrams. Nevertheless, besides Simulink, Modelica-based tools such as Dymola, Simplorer, MapleSim, SimulationX and SystemModeler also support multi-domain modeling. Modelica (`http://modelica.org`) is a non-proprietary, object-oriented, declarative and equation-based language. In terms of popularity, Modelica has surpassed the more traditional modeling approach of *bond graph*.

If the physics of a system is not well understood, *system identification techniques* can be used to derive a numeric model. For complex systems, it is reasonable to expect the model to have a mix of analytically derived components and empirically identified components.

3.1.3 Continuous vs. discrete time

When modeling a system/process/plant, sometimes it is more natural to treat time as a continuum, in which case we get a *continuous-time* model; but sometimes it is more natural to treat time as a series of discrete time intervals, in which case we get a *discrete-time* model. We will go through some examples of continuous-time and discrete-time models in the next lecture.

- For a **continuous-time** plant, the controller can be an analog controller (which typically operates in continuous time) or a digital controller (which operates in discrete time). Analog controllers used to be abundant in the process control industry, but have been in constant decline for the past few decades, being replaced by more flexible digital platforms. These digital controllers gave rise to *sampled-data systems* — a sampled-data system is a system that operates in continuous time, but with some of its continuous-time signals sampled at discrete time instants [CF95, Ch. 1]. A sampled-data control system is a sampled-data system in which a continuous-time plant is controlled by a discrete-time controller (see Figure 9). A sampled-data system is by definition a *hybrid system*, but traditionally, to design the digital controller, instead of using *hybrid control* techniques, the control engineer either
  - design the controller in continuous time first and then discretize it; or

– discretize the plant model first and then design the controller in discrete time.

This course focuses on the former approach, and hence in the next lecture, we will be mostly concerned with continuous-time rather than discrete-time modeling.

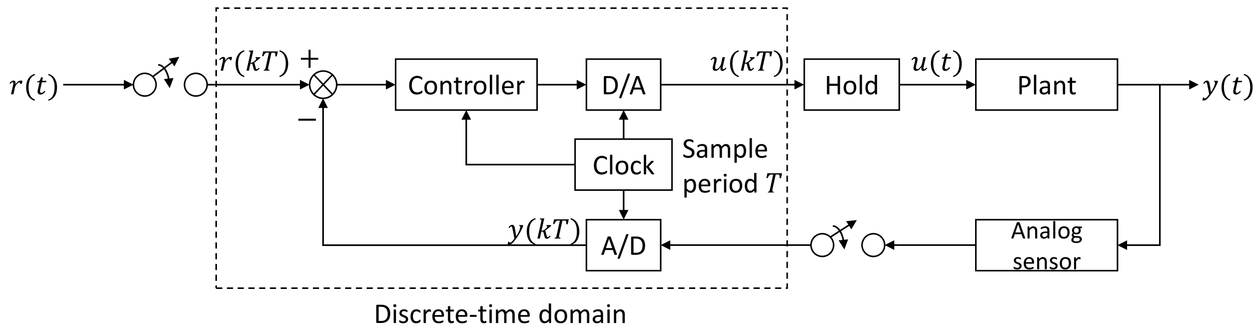

Figure 9: A sampled-data control system.

- For a **discrete-time** plant, which readily lends itself to discrete-time control, design of the digital controller is directly based on the plant model.

3.1.4 SISO vs. multivariable

A process model is either single-input, single-output (SISO) or multivariable:

- A **SISO** process has exactly one input and exactly one output.
- A **multivariable** process has at least two inputs and/or at least two outputs. A multivariable process that has at least two inputs and at least two outputs is a multiple-input, multiple-output (MIMO) process.

For SISO processes, both classical and modern control design methodologies are applicable; whereas for multivariable processes, only modern control design methodologies are applicable.

3.1.5 Linear vs. nonlinear

A model is either *linear* or *nonlinear*.

- A **linear** system is a system that satisfies the *superposition principle*: if an input consists of the weighted sum of several signals, then the output is the weighted sum of the responses of the system to each of those signals (see Figure 10). Mathematically, we write

$$y(ax_1(t) + bx_2(t), t) = ay(x_1(t), t) + by(x_2(t), t), \quad \forall a, b \in \mathbb{R}, \tag{4}$$

  where $y(x(t), t)$ is the output of the linear system; $x_1(t)$, $x_2(t)$ are inputs; a, b are arbitrary weights. An implication of the superposition principle is

$$x(t) = 0 \implies y(x(t), t) = 0, \tag{5}$$

  i.e., zero input implies zero output.

- A **nonlinear** system is a system that is not linear.

A model is typically nonlinear, so unless nonlinear control techniques are used, the model is to be linearized.

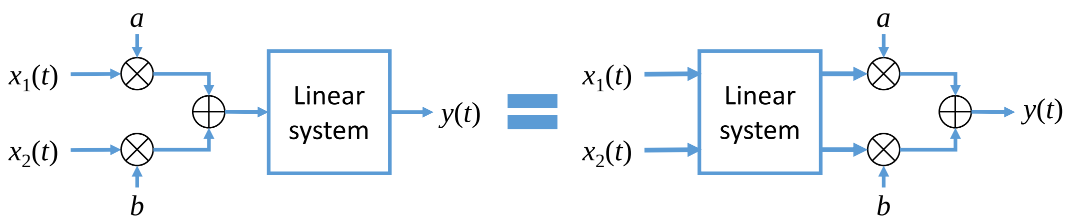

Figure 10: The superposition principle.

Example 3

The system $y(x(t), t) = t \cdot x(t)$ is linear, because

$$\begin{aligned} y(ax_1(t) + bx_2(t), t) = t(ax_1(t) + bx_2(t)) &= atx_1(t) + btx_2(t) \\ &= ay(x_1(t), t) + by(x_2(t), t), \quad \forall a, b \in \mathbb{R}. \end{aligned}$$

Example 4

The system $y(x(t), t) = \int_{-\infty}^{t} x(\tau)\, \mathrm{d}\tau$ is linear, because

$$\begin{aligned} y(ax_1(t) + bx_2(t), t) = \int_{-\infty}^{t} (ax_1(\tau) + bx_2(\tau))\, \mathrm{d}\tau &= a \int_{-\infty}^{t} x_1(\tau)\, \mathrm{d}\tau + b \int_{-\infty}^{t} x_2(\tau)\, \mathrm{d}\tau \\ &= ay(x_1(t), t) + by(x_2(t), t), \quad \forall a, b \in \mathbb{R}. \end{aligned}$$

Note that using Leibniz's integral rule, $y(x(t), t) = \int_{-\infty}^{t} x(\tau)\, \mathrm{d}\tau \implies \dot{y}(x(t), t) = x(t)$.

Example 5

The system $y(x(t), t) = 2x(t) + 1$ is *nonlinear*, because

$$\begin{aligned} y(ax_1(t) + bx_2(t), t) = 2(ax_1(t) + bx_2(t)) + 1 &\neq a(2x_1(t) + 1) + b(2x_2(t) + 1) \\ &= ay(x_1(t), t) + by(x_2(t), t), \quad \forall a, b \in \mathbb{R},\ a + b \neq 1. \end{aligned}$$

Alternatively, notice when $x(t) = 0$, $y(x(t), t) \neq 0$, so we can conclude the system is nonlinear. Notice the fact that the plot of $y(x(t), t)$ against $x(t)$ is a straight line does not imply the system is linear.

Usually, when we talk about linear control systems, we talk about systems represented by linear *ordinary* differential equations of the form

$$a_n(t)y^{(n)}(t) + a_{n-1}(t)y^{(n-1)}(t) + \cdots + a_0(t)y(t) = b_m(t)x^{(m)}(t) + \cdots + b_0(t)x(t), \tag{6}$$

where $x(t)$ and $y(t)$ represent the input and output respectively. Note that

- Coefficient $a_i(t)$, $i = 0, \ldots, n$, is either a time-dependent function or a constant.
- Derivative $y^{(i)}(t)$, $i = 0, \ldots, n$, has a power of exactly 1, if it exists.

- Coefficient $b_j(t)$, $j = 0, \dots, m$, is either a time-dependent function or a constant.
- Derivative $x^{(j)}(t)$, $j = 0, \dots, m$, has a power of exactly 1, if it exists.

Example 6

The system $\dot{y}(x(t), t) = e^{-t}x(t)$ is linear, because it is of the form of Eq. (6).

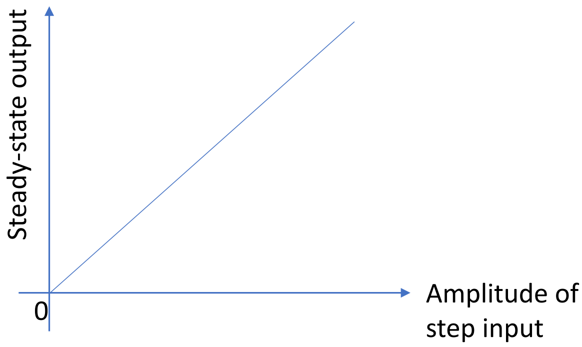

Figure 11: Is there enough evidence for the system to be considered linear?

Example 7

Consider the plot of steady-state output versus magnitude of step input in Figure 11. Can we conclude the system represented by the plot is linear? If not, why?

Solution: No, because the plot in Figure 11 does not take the transient response into account. To say whether a system is linear, we *really* need to check if its model is linear. For example, consider the state-space model

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} (-x_1 + x_2)^2 \\ 0 \end{bmatrix}, \tag{7}$$

where x_1 and x_2 are the output and input respectively. The model is nonlinear because of the square term, but in steady state, $x_1 = x_2$ (a linear relationship). Figure 12 shows when the input is doubled, the transient response is not double the amount of transient response associated with the original input.

3.1.6 Time-invariant vs. time-varying

A linear system is easier to analyze than a nonlinear one. Likewise, a time-invariant system is easier to analyze than a time-varying one.

- A **time-invariant** system is a system whose properties do not change with time. In other words, advancing/delaying the input by τ does nothing more than advancing/delaying the output by τ. Mathematically, we write

$$y(x(t-\tau), t) = y(x(t-\tau), t-\tau), \quad \forall \tau \in \mathbb{R}. \tag{8}$$

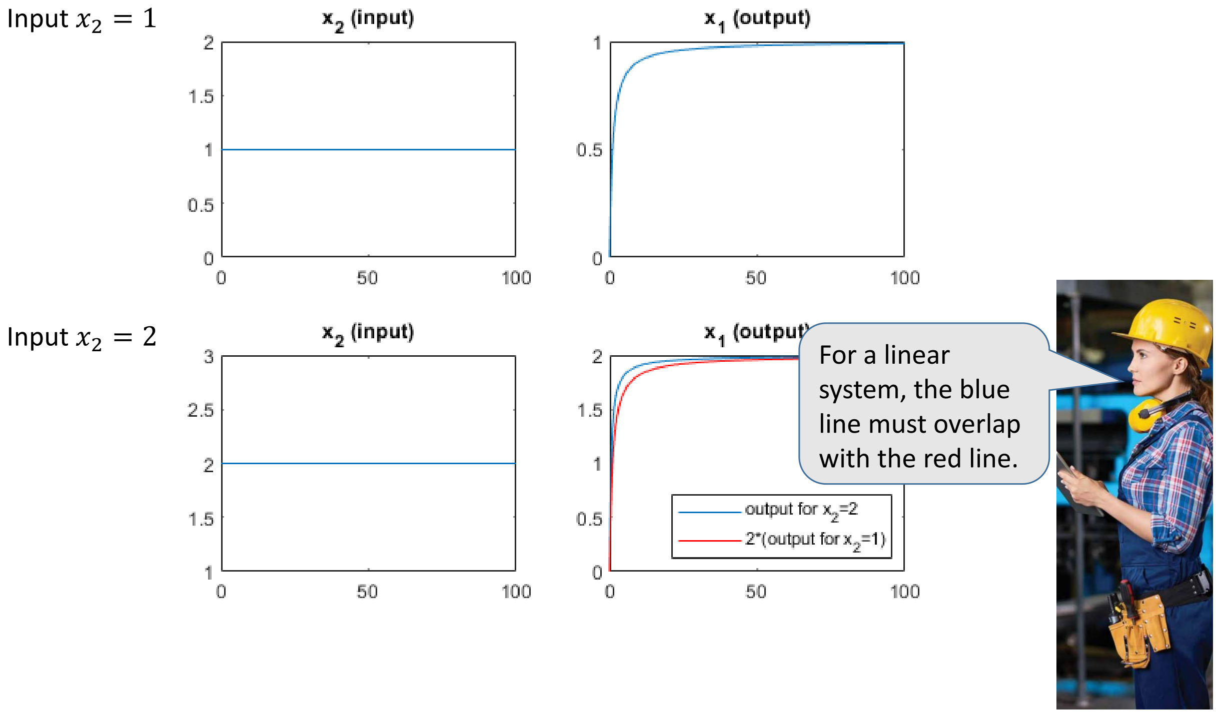

Figure 12: Step response plots for the sample system (7). If the system is linear, the blue line would have overlapped with the red line in the bottom right plot.

The system (6) is time-invariant if all the coefficients $a_n(t), \ldots, a_0(t)$ and $b_m(t), \ldots, b_0(t)$ are constants, but the converse is not necessarily true.

- A **time-varying** system is the opposite of a time-invariant system.

In this course, we only deal with *linear time-invariant* (LTI) systems.

Example 8

The system $y(x(t), t) = \sin(x(t))$ is nonlinear (can you show this?), and time-invariant because

$$y(x(t-\tau), t) = \sin(x(t-\tau)) = y(x(t-\tau), t-\tau), \quad \forall \tau \in \mathbb{R}.$$

Example 9

The system $y(x(t), t) = t \cdot x(t)$ is linear as shown in Example 3, and time-varying because

$$y(x(t-\tau), t) = t \cdot x(t-\tau) \neq (t-\tau) \cdot x(t-\tau) = y(x(t-\tau), t-\tau), \quad \forall \tau \in \mathbb{R}.$$

Quiz 2

Given $x(t)$ as input, $y(x(t), t)$ as output, determine mathematically whether the system is linear or nonlinear, time-invariant or time-varying:

$$y(x(t), t) = \ln(x(t-1)). \tag{9}$$

3.2 Design/synthesis and analysis

When designing or synthesizing a controller, we are mostly trying to build a *closed-loop system*, as opposed to an *open-loop system* (see Figure 13), because:

- An **open-loop system** does not feed the output signal back to the controller, so the output variables do not affect the input variables of the plant. The system can track the reference command given an accurate enough plant model, but it cannot compensate for unexpected disturbances. Nevertheless, the controller does not change the stability of the system.

- A **closed-loop system** feeds the output signal back to the controller in order to track the reference command, so the output variables do affect the input variables of the plant. The controller computes the control output based on the control error such that the control error is minimized. A closed-loop system can reject the effect of disturbances, but could stabilize or destabilize a system.

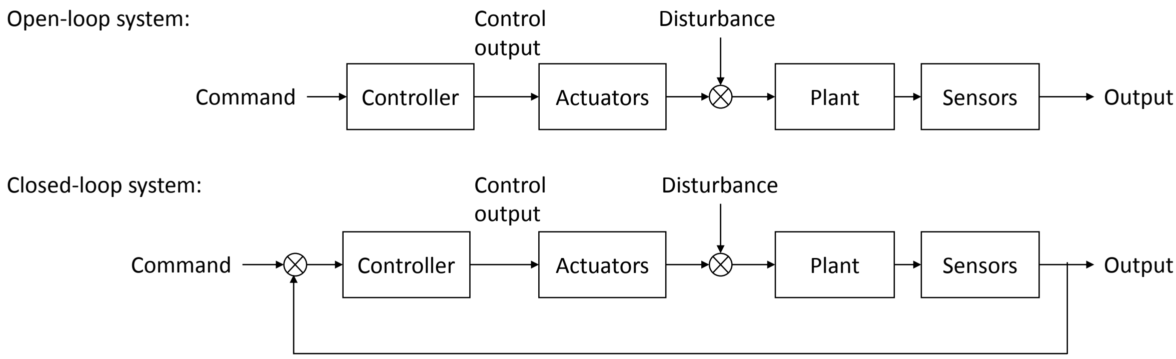

Figure 13: An open-loop and a closed-loop system.

When designing a closed-loop system, we are designing either a *servomechanism* or a *regulator*:

- A **servomechanism**[1] or **tracker** ensures the system output approximate a variable reference signal. For example, the motor lifting a weight up a specified height in Example 1 is a servomechanism.

- A **regulator** ensures the system response to disturbances decays at a desired rate, so that the system returns to the equilibrium state dictated by the initial condition. A regulator is basically a servomechanism with a constant (typically zero) reference signal. For example, an active suspension system is a regulator that tries to reject variation in the road surface and driving dynamics to return to the equilibrium state.

Quiz 3

For each of the sample control systems in Supplementary Lecture A, say whether it uses a servomechanism or a regulator.

In this course, we focus on the following types of servomechanisms or regulators:

- proportional-integral (PI) controller and lag compensator;
- proportional-derivative (PD) controller and lead compensator;

[1] "Servo" can be a noun or an adjective. When used as a noun, it is short for "servomechanism". When used as an adjective, it serves as a prefix, e.g., "servosystem". For the origin of this word, see Sect. 4.

- proportional-integral-derivative (PID) controller and lag-lead compensator.

We will apply the following design methodologies:

1. root locus;
2. heuristic PID tuning;
3. frequency response.

3.3 Simulation, testing, implementation and validation

The problem with testing the actual system prototype is that overlooked design or implementation errors can damage the hardware, causing financial loss and introducing delays to the project. The problem is more pronounced for control systems that comprise modules or components developed by different engineers. The standard solution is to test the control system first virtually, then physically. Therefore, except for really really simple projects, controller development and testing typically goes through these three phases [Cet15, Sect. 1.3]:

Phase 1 Simulation in a non-real-time environment.

- The control system is developed using a control design software such as MATLAB/Simulink, Maple/ MapleSim, CATIA/Dymola, etc., and simulated on a non-real-time computer.

Phase 2 Hardware-in-the-loop (HIL) simulation and testing in a real-time environment (see Figure 14). The goal of HIL simulation is to work out the errors before moving to final hardware testing, which is supposed to validate good design, rather than uncover bad design. In general, the steps are [Mat17, p. 24-4]:

1. Develop a model that represents the environment or system under development. The challenge lies in creating a simulation at a suitable balance between computational efficiency and model accuracy.
2. Generate an executable for the environment model.
3. Download the executable for the environment model to the HIL simulation platform.
4. Replace software representing a system component with the corresponding hardware. This system component is typically the controller. The controller's C code can be generated from the system model, and run on the target *embedded controller module* (ECM), which can contain a high-performance microprocessor and some FPGA components. Examples of ECM include general-purpose microcontrollers, programmable logic controllers (PLCs) for industrial applications, electronic control units for automotive applications, the National Instruments GPIC for power electronics.
5. Test the hardware in the context of the HIL system. If the hardware is the controller, then instead of the actual plant, the controller interfaces with a high-performance computer called the *target computer* running a simulation of the plant.
6. Repeat steps 4 and 5 until all components that require testing have been tested in integrated simulation.

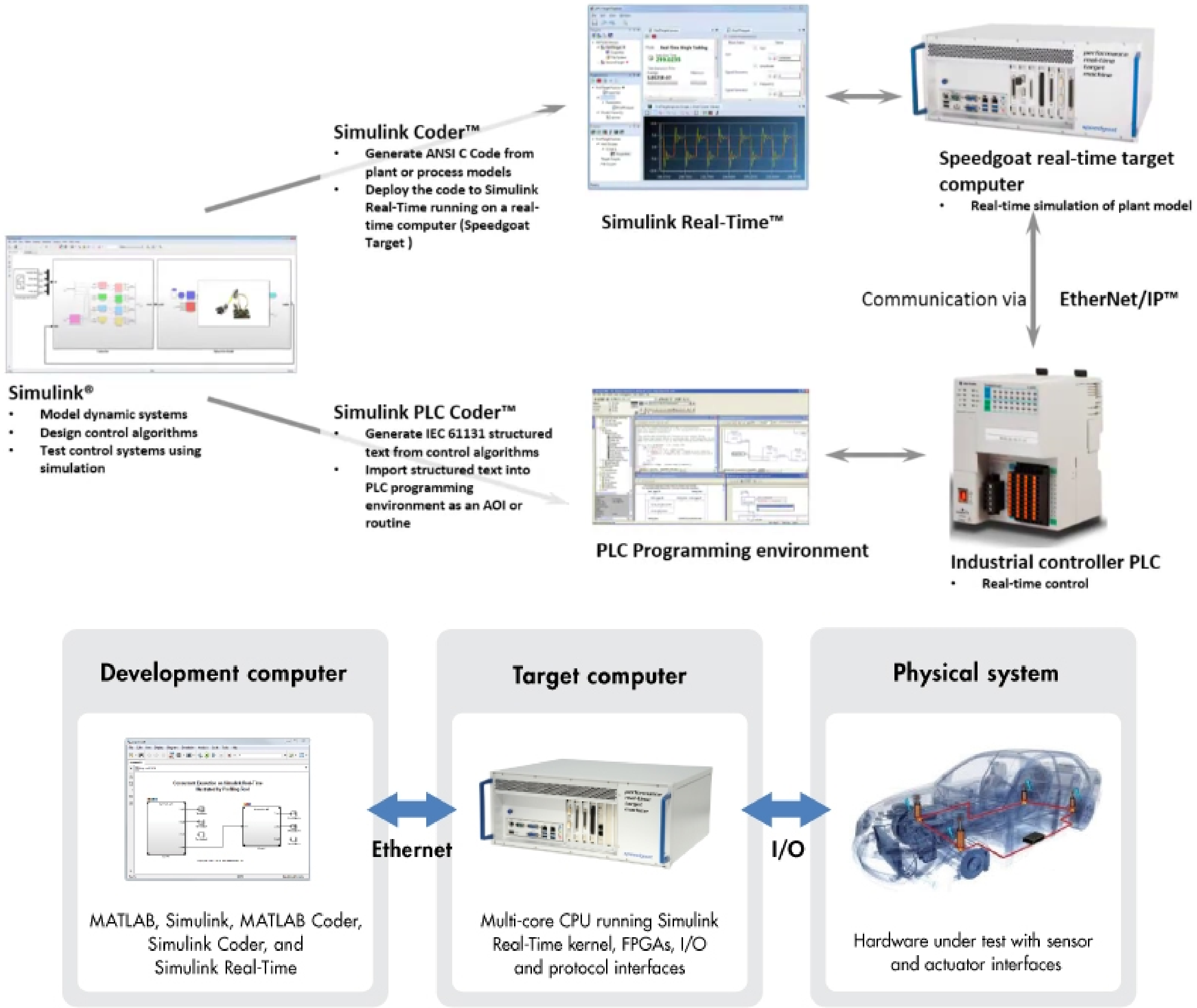

Figure 14: An example of HIL simulation/testing using Simulink and associated toolboxes.

Example 10

Besides modeling, Simulink can be used for HIL simulations (see Figure 14). Given a Simulink model, Simulink PLC Coder can be used generate hardware-independent IEC 61131-3 Structured Text and Ladder Diagrams for PLCs, whereas Simulink Coder can be used to generate C code for a range of non-PLC microcontroller platforms, including BeagleBone Blue, dSPACE ACE Kit, NXP FRDM-K64F, NXP FRDM-KL25Z, some STMicroelectronics Nucleo boards, FlexECU.

For constructing an HIL simulation platform, the MathWorks recipe prescribes Simulink Real-Time for the simulator software, and Speedgoat for the simulator hardware (target computer). The role of Simulink Real-Time is twofold: (i) to create a real-time application from a Simulink model, to be downloaded onto the target computer, and (ii) to serve as the real-time kernel of the target computer. Simulink Real-Time provides driver blocks for I/O (e.g., analog, digital, pulse train, encoders, transformers, passive components, serial, audio, shared memory, reconfigurable FPGA), and data buses (e.g., Raw Ethernet, EtherCAT, Ethernet/IP, real-time UDP, CAN).

Speedgoat is a Simulink-centric offering. dSPACE is a well-known alternative to Speedgoat, and it has the advantage of supporting a wide range of models including Simulink and Modelica models.

Phase 3 Testing and validation on the actual hardware.

- The controller running on the ECM is tested with the actual plant.
- The sensors and actuators are calibrated.
- The controller parameters are *tuned* to obtain the best possible dynamic performance based on expert operator and end-user comments.

Phase 1 and Phase 2 are obviously useful especially when testing the controller on the actual plant straightaway is either too costly or hazardous (e.g., rocket control). HIL sits comfortably between pure software-based simulation and pure hardware-based testing, and has been experiencing spectacular growth in recent years.

4 Brief history of control

Control is a relatively young field. Understanding the historical events helps us appreciate and understand why current methodologies are what they are, as well as what can or cannot be done with current control technologies. Poetically put, the beauty of control is best appreciated through the historical lens. The history of control can be divided into the following periods [Ben96]:

1. Early period
2. Pre-classical period
3. Classical period
4. Modern period
5. Present

Except for the second lecture which covers the state-space modeling approach of the Modern period, our introductory control course covers knowledge established in the Early, Pre-classical and Classical periods.

4.1 Early period

This period is pre-1900 approximately. Within this period, feedback control is said to have originated with the float valve regulators of the Hellenic and Arab worlds, which were reinvented in Europe during the industrial revolution. Engineers in this age were still experimenting with feedback mechanisms, which were purely mechanical in nature. For the historically inclined, the Smithsonian has an extensive catalog of feedback mechanisms invented in this period [May71].

The most significant development in the 18th century was the steam engine governor. James Watt's steam engine was fundamental to the industrial evolution, as the *centrifugal governor* was fundamental to the steam engine. A centrifugal governor (also called a flyball governor, see Figure 15) is a mechanical device that uses mainly the properties of centrifugal force to measure and regulate the speed of a machine. Watts' original design was only capable of proportional action, but by the end

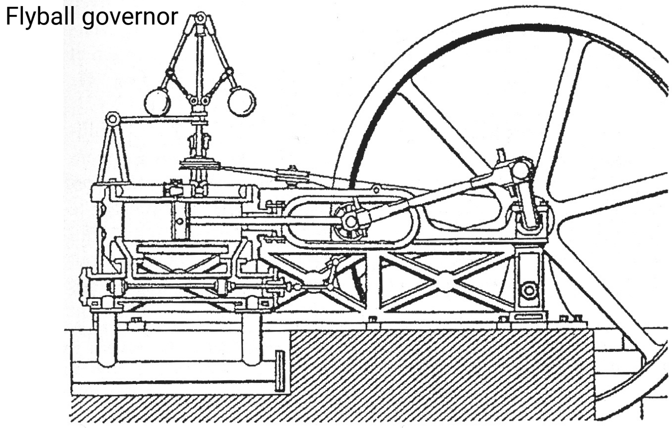

Figure 15: A steam engine with a flyball governor [rM08, Figure 1.2]. A demonstration of how a modern-day diesel engine governor works can be found on YouTube: `https://youtu.be/OiHb2L8ei8E`.

of the 19th century, governors of various designs — including the Siemens chronometric governor which was capable of integral and differential actions — were available to a range of applications [Bis09]. There was no interest in the analysis of the dynamics of these governors until there emerged a need to keep a telescope directed at a particular star as the Earth rotated.

In his 1868 paper "On governors", James Maxwell (the physicist who gave us Maxwell's equations of electromagnetism) analyzed the stability of governor-regulated systems up to the third order, and thereby co-founded the theory of automatic control [Żak03]. Edward Routh and Adolf Hurwitz completed Maxwell's quest to generalize his stability analysis to systems of any order, by independently arriving at the celebrated **Routh-Hurwitz stability criterion**.

Around that time, the first closed-loop steering engine for steamships appeared in the U.S. In France, Jean Farcot designed a range of steering engines and other closed-loop position control systems, and called them "servo-moteur" or "moteur asservi" [Ben96]. This is the origin of the terms "servomotors" and "servomechanisms".

In 1892, Lyapunov of Russia published his groundbreaking work on the stability of nonlinear systems in his thesis entitled "The General Problem of Stability of Motion", but it was little known outside Russia until the Modern period. Lyapunov stability analysis is covered in EEET 4071 Advanced Control.

4.2 Pre-classical period

This period lasted from 1900 till 1935 approximately. The beginning of the 20th century saw the acceleration of automation and manufacturing, driving the rapid and widespread application of feedback controllers to a wide range of applications, including boiler control for steam generation, and electric motor speed control.

In the power industry, the first industrial research laboratory — the General Electric Research Laboratory — was established, to work on voltage and frequency regulation in power grids.

In the transportation industry, major advances under the label of the "gyro culture" occurred in ship steering. Elmer Sperry of the U.S. invented a range of gyroscope-based devices for ships and aircrafts: the gyro compass for steering (see Figure 16), active stabilizer for stabilization[2], and gyroscope autopilot (gyro-pilot) for autopiloting [Ben96]. Sperry's gyro-pilots were so successful that they were used in hundreds of systems. The amazing fact about these gyro-pilots is that they achieved adaptive PID control based entirely on Sperry's intuition rather than advanced theoretical understanding

[2]See `https://youtu.be/_ncTy6ny9vI` for a visualization of how gyroscopic stabilization works.

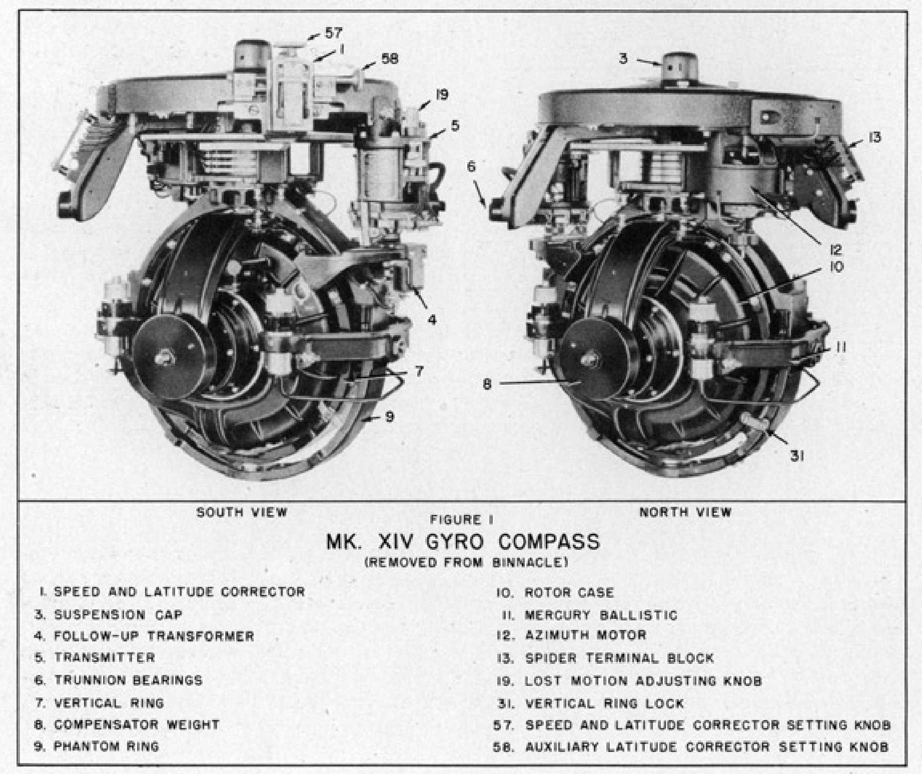

Figure 16: A great achievement of the “gyro culture”: a Sperry gyro compass, which can be used for autopiloting. Source: `http://www.maritime.org/doc/gyromk14/img/fig001.jpg`.

[Ben96]. The lack of systematic understanding of these devices has spurred Nicholas Minorsky to make the first theoretical analysis of automatic ship steering in 1922, providing historically the first theoretical basis and recommendation for PID control, although the term “PID” itself was not used.

In the process control industry, the main concern was the control of temperature, pressure, and flow. Boilers, reactors, distillation columns, mixers and blenders were instrumented with a wide range of newly developed sensors and actuators. Pneumatic controllers emerged as general-purpose devices, and came equipped with dials for adjusting controller parameters. The demand for new sensors, actuators and controllers gave birth to companies such as Foxboro, Honeywell, Siemens, Taylor Instrument, and Yokogawa. In 1931, Clesson Mason of Foxboro produced a pneumatic negative feedback amplifier called Stabilog, which was also the world’s first general-purpose PID controller [Ben96, ÅK14].

In the electronics and telecommunications sectors, Harold Black of AT&T’s Bell Labs invented the negative feedback amplifier, reducing distortion significantly for transcontinental telephony. For the problem of determining the stability of feedback loops, Black enlisted Harry Nyquist’s help, who published the celebrated **Nyquist stability criterion** in his paper “Regeneration Theory” in 1932. The Nyquist stability criterion has a significant advantage over the Routh-Hurwitz stability criterion.

Overall, the range of devices designed, built, and manufactured in this period was large, but most were designed without any clear understanding of the dynamics of both the system to be controlled, and the sensors and actuators used for control [Ben96]. Although the process control industry has created a huge drive for control technologies, ideas were protected as trade secrets [ÅK14]. Theoretical progress was slow, and it was not until WWII, that advances made in electronics and telecommunications in the 1920s and 1930s were translated into the control field [Bis09].

Figure 17: Harry Nyquist, a key figure in the pre-classical period.

Figure 18: Hendrik Bode, a key figure in the classical period.

4.3 Classical period

This period lasted from 1935 till 1950 approximately. This is a period of intense theoretical development driven by rapid growth of the telecommunications sector and the onset of WWII. The most significant contributions were made by three groups in the U.S., namely

1. The Bell Labs group: In its quest for ever higher telecommunications bandwidth, AT&T set its most brilliant minds on the problem of achieving a constant gain over a wide bandwidth with a sharp cut-off and a small phase lag. One of them is Hendrik Bode (see Figure 18), one of the founders of classical control, who is known for many contributions. He investigated the relationship between attenuation and phase, and introduced the concepts of gain margin and phase margin, and the notion of minimum phase. He discovered the fundamental limitation that makes control system design inherently an exercise of compromise: making the *sensitivity function* smaller for some frequencies increases it for other frequencies; in other words, disturbances at some frequencies can only be attenuated if disturbances at other frequencies are amplified [rM08, Ch. 11]. More importantly, Bode developed the graphical design framework of *loop shaping* based on what we call **Bode plots** to design feedback amplifiers. His deep insights on the application of control to designing feedback amplifiers were documented in his 1945 book "Network Analysis and Feedback Amplifier Design". To commemorate Bode's contributions, IEEE founded the Hendrik W. Bode Lecture Prize.

2. The MIT group: The onset of WWII created a strong demand for artillery control technologies. The firing control problem was a challenging problem that involved the (radar-based) detection of an enemy aircraft, the calculation of the aircraft's current and future positions, and the precise aiming of a heavy gun on the moving aircraft [Bis09]. MIT's Servomechanisms Laboratory and the Radiation Laboratory played a major role in solving this problem.
   - Gordon Brown pioneered the usage of block diagrams, and developed improved hydraulic systems for steering turrets [Ben96, ÅK14].
   - Albert Hall showed that by treating the blocks as transfer functions (using Laplace transform), the system transfer locus could be drawn, allowing the gain and phase margins to be obtained [Ben96].
   - Out of the Radiation Laboratory's work on the SCR-584 radar system, Nathaniel Nichols' work on Nichols chart was born; these results and many others were compiled in a book called "Theory of Servomechanisms" after the war.

3. The process industry group: The American Society of Mechanical Engineers (ASME) formed an Industrial Instruments and Regulators Committee in 1936, and became the first major professional body to form a section specializing in automatic control [Ben96]. In 1942, Ziegler and Nichols of Taylor Instrument published what are now known as the **Ziegler-Nichols tuning rules** for tuning PID controllers based on measurements of the process to be controlled. Hundreds of variations of these rules have been proposed ever since [O'D09].

The **frequency response** methodology based on the work of Nyquist, Bode, Nichols, etc. assessed performance in terms of bandwidth, resonance, gain margin, and phase margin. There was no straightforward way in which the designer could relate system parameter changes to time behavior changes, until at the end of the 1940s, when Walter Evans proposed the **root locus** methodology, which relates performance in terms of rise time, settling time, percent overshoot, steady-state error, etc. to system parameters.

The classical approaches of frequency response and root locus had tremendous successes in the war. However, they are only applicable to linear, deterministic SISO systems. Real systems are nonlinear, difficult to model accurately, and produce noisy measurements. The quest for design techniques that are applicable to nonlinear, stochastic multivariable systems, and that are robust to noise and model uncertainty, ushered the control community into the modern period.

4.4 Modern period (Golden Age)

This period is considered to have commenced in the early 1950s, but nobody has put an end date to it. The beginning of this Golden Age — so called because of an explosion of successes in various areas of control — was marked by the establishment of some professional bodies and conferences:

- the "Automatic Control" conference in Cranfield in 1951;
- the "Frequency Response Symposium" in New York in 1953;
- the International Federation of Automatic Control (IFAC) in 1957, and its first conference in Moscow in 1960.

It was at the Moscow conference that Kalman presented his theory that laid down the foundation of modern control theory.

Activities in this period were driven by two major developments:

1. the cold war and space race between the U.S. and USSR, and
2. the advent of the digital computers.

Many problems encountered in aerospace applications (launching, maneuvering, guiding, and tracking of missiles and spacecrafts), process control, and econometric applications are time-varying multivariable problems that classical control cannot handle. This triggered interest in the state-space approach to solving differential equations, which can be traced back to Poincaré and Lyapunov at the end of the 19th century.

In the early 1950s, the design of autopilots for high-performance aircrafts required new control technologies, because aircraft dynamics changes over large flight envelopes, e.g., altitude changes aircraft dynamics. The need to adjust controller parameters dynamically gave birth to *adaptive control*. During this time which earnt the moniker "brave era", NASA experimented with adaptive control on its X-15 aircrafts. The program was stunted by the fatal crash of X-15-3 which had an adaptive controller without stability proof, but was revived upon the introduction of *Lyapunov-based model-reference adaptive control*. The Lyapunov theory of stability is covered in EEET 4071 Advanced Control.

In the late 1950s, the design problem evolved from simply achieving a stable controller to that of achieving the "best" controller, e.g., best in terms of minimizing the time of flight or fuel consumption of a rocket, giving birth to the field of *optimal control*.

In the 1960s, Kalman and Bellman in the U.S. and Pontryagin in the USSR pioneered methods for optimal estimation and control. These state-space methods, such as the Kalman filter and dynamic programming, were sophisticated but efficient for practical applications thanks to the digital computers.

In the 1970s, the issue of robustness (the ability to achieve control goals despite uncertainty in the process model) came into focus, and the field of *robust control* began to blossom.

By the 1980s, various control approaches including adaptive control, optimal control and robust control reached an advanced level. Most importantly, these mathematically sophisticated control approaches have found practical applications in diverse areas (see Supplementary Lecture A).

The history of this period is further discussed in EEET 4071 Advanced Control.

4.5 Present

The penetration of the Internet has a profound impact on the field of control. In the late 1990s, the Smart Dust project at the University of California at Berkeley opened up a field called *wireless sensor and actuator networks* (often just *wireless sensor networks* for short). The ability to deploy a control system over a large-scale network of sensors and actuators gave rise to *control over networks* or *networked control* [ÅK14]. Conversely, networks can benefit from control to achieve good performance, and since loops are readily closed over the communication network, *control of networks* is a natural consequence of applying control to networking. The term "cyber–physical system" describes a networked system of computational elements controlling physical plants; and signifies the increasingly tight coupling of control, computing and communications.

As the control community tries to conquer networked systems, we see a confluence of ideas from multiple disciplines, including computer science, communication and life sciences. For example,

- The field of *hybrid systems* aims at controlling systems with continuous-time and discrete-time subsystems using automata theories from computer science.
- The field of *systems biology* aims at modeling complex biological systems to understand how to disrupt the feedback of harmful biological pathways that cause disease [ÅK14].

From the early days, control has come a long way and is constantly evolving. Taking an introductory control course is merely taking the first step in a very long march toward becoming an expert.

5 Roadmap, tools and learning resources

Figure 19 summarizes the scope of this course, and Figure 20 shows the coverage of this course compared to common topics covered at a more control-focused university. Below are several important points:

- This course is about classical control, which deals with LTI SISO systems.
- Historically speaking, continuous-time control came before discrete-time control. As a rule of thumb, we can readily convert any continuous-time controller to a discrete-time one as long as the sampling rate is at least 30 times the system's *bandwidth* [FPW98, p. 4]. Therefore, discrete-time control is covered if time allows (there is no point covering the syllabus for the sake of covering the syllabus).

> **Detail: Sampling rate**
>
> The truth about the sampling rate is actually more complicated than the above. The rules of thumb for the choice of sampling rate include [Mou07, Sect. 8.1]:

- The sampling rate should be at least 10 times the Nyquist rate.
- There should be 4 to 10 samples per rise time of the closed-loop system.
- There should be 15 to 45 samples per period of the oscillating closed-loop system.
- The sampling rate should be at least 30 times the bandwidth.
- $\omega_{gc}T$ should be between 0.15 and 0.5, where ω_{gc} is the *gain crossover frequency*, and T is the sample period.
- T should be such that the decrease in *phase margin* of the discretized system from the original is not more than $15°$.

- After going through Sect. 3, and based on Figure 20, you would realize most of the advanced techniques are not covered in this course or EEET 4071 Advance Control. Think about doing a final-year project and even a research-based postgraduate course if you want to pick up those techniques.

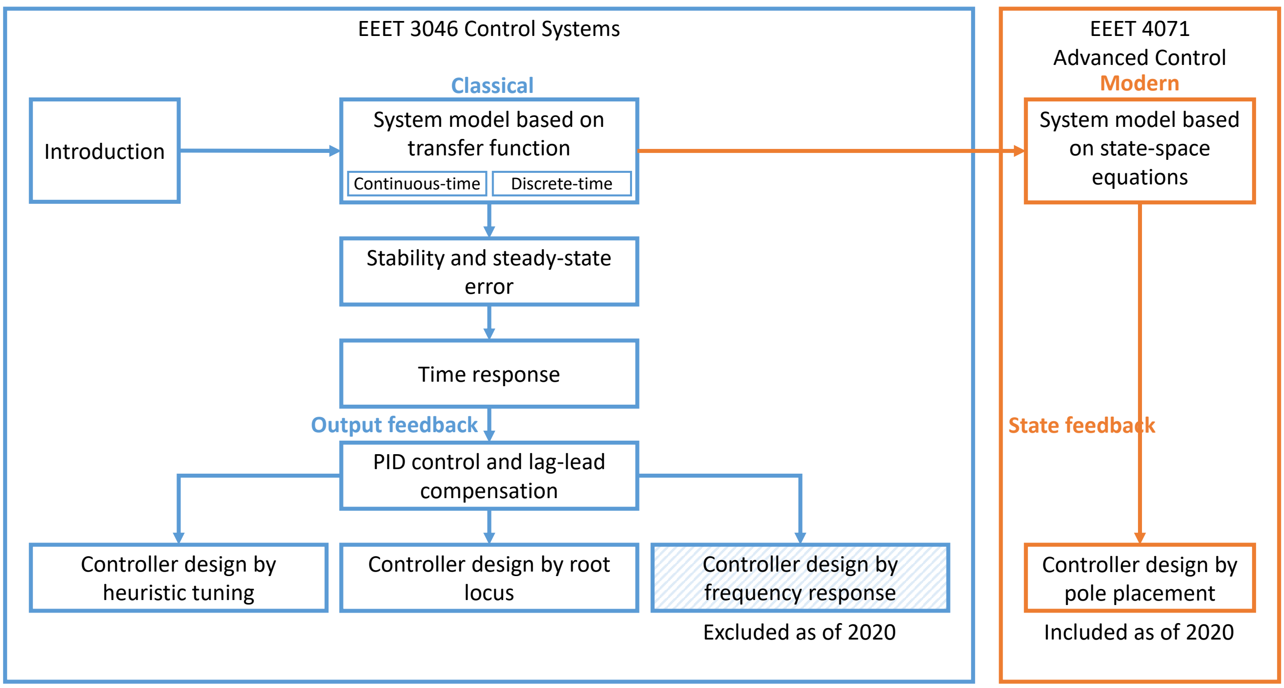

Figure 19: Course syllabus.

The following software tools are used in this course:

- MATLAB/Simulink for controller design.
- Arduino for controller implementation.

The adoption rate of Modelica-based tools such as Dymola is increasing in the industry, but for practical reasons, they are not taught or used in this course.

For a major engineering discipline like control, learning resources are aplenty. Besides the lecture notes, eReadings have been made available to enhance the students' learning. Journals reporting the latest research results in control are available electronically through the university library (see Table 1). Students who are fond of learning from videos may find the following YouTube channels useful:

- Brian Douglas's "Control System Lectures" videos: `https://www.youtube.com/user/ControlLectures`

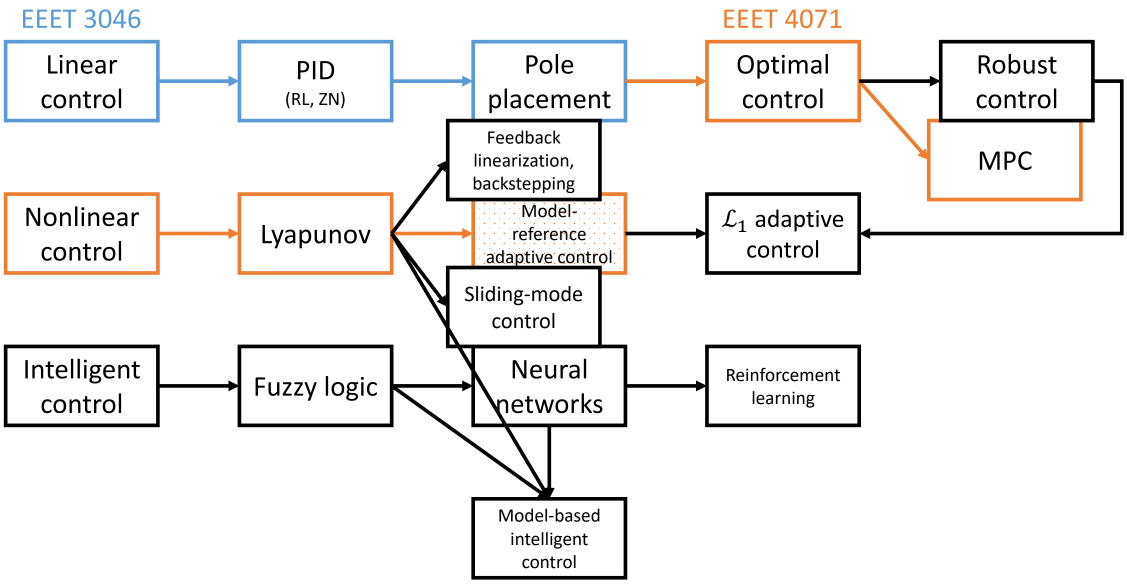

Figure 20: Coverage of this course compared to common topics covered at a more control-focused university, such as Lund University (`http://www.control.lth.se/education/engineering-program/`).

- Georgia Tech's "Control of Mobile Robots" videos: `https://www.youtube.com/user/freeonlinecourses1`

References

[ÅK14] Karl J. Åström and P.R. Kumar. Control: A perspective. *Automatica*, 50(1):3 – 43, 2014.

[Ben96] S. Bennett. A brief history of automatic control. *IEEE Control Syst. Mag.*, 16(3):17–25, June 1996.

[Bis09] Christopher Bissell. A history of automatic control. In *Springer Handbook of Automation*, pages 53–69. Springer, 2009.

[Cet15] Sabri Cetinkunt. *Mechatronics with Experiments*. John Wiley & Sons Ltd, 2015.

[CF95] Tongwen Chen and Bruce Allen Francis. *Optimal Sampled-Data Control Systems*. Springer London, 1995.

[dSKML15] Clarence W. de Silva, Farbod Khoshnoud, and Saman K. Halgamuge Maoqing Li, editors. *Mechatronics: Fundamentals and Applications*. CRC Press, 2015.

[FPW98] Gene Franklin, J. David Powell, and Michael Workman. *Digital Control of Dynamic Systems*. Addison Wesley Longman, Inc., third edition, 1998.

[GA15] Mohinder S. Grewal and Angus P. Andrews. *Kalman Filtering: Theory and Practice Using MATLAB®*. John Wiley & Sons, Inc., 4th edition, 2015.

[GGS00] Graham C. Goodwin, Stefan F. Graebe, and Mario E. Salgado. *Control System Design*. Prentice Hall, 2000.

[JK93] John L Junkins and Youdan Kim. *Introduction to dynamics and control of flexible structures*. AIAA, 1993.

[Mat17] The MathWorks, Inc. *Simulink®Coder™User's Guide*, r2017b edition, 2017.

Table 1: Sample scientific journals related to control.

Journal	URL
IEEE Transactions on Automatic Control	`https://ieeexplore-ieee-org.access.library.unisa.edu.au/xpl/RecentIssue.jsp?punumber=9`
IEEE Transactions on Control Systems Technology	`https://ieeexplore-ieee-org.access.library.unisa.edu.au/xpl/RecentIssue.jsp?punumber=87`
IEEE Control Systems	`https://ieeexplore-ieee-org.access.library.unisa.edu.au/xpl/RecentIssue.jsp?punumber=5488303`
IEEE Transactions on Control of Network Systems	`https://ieeexplore-ieee-org.access.library.unisa.edu.au/xpl/RecentIssue.jsp?punumber=6509490`
IEEE Transactions on Fuzzy Systems	`https://ieeexplore-ieee-org.access.library.unisa.edu.au/xpl/RecentIssue.jsp?punumber=91`
IET Control Theory & Applications	`https://ieeexplore-ieee-org.access.library.unisa.edu.au/xpl/RecentIssue.jsp?punumber=4079545`
International Journal of Control	`https://www-tandfonline-com.access.library.unisa.edu.au/loi/tcon20`
International Journal of Robust and Nonlinear Control	`https://onlinelibrary-wiley-com.access.library.unisa.edu.au/journal/10.1002/(ISSN)1099-1239`
Control Engineering Practice	`https://www-sciencedirect-com.access.library.unisa.edu.au/science/journal/09670661`
Mechatronics	`https://www-sciencedirect-com.access.library.unisa.edu.au/science/journal/09574158`
ASME Journal of Dynamic Systems, Measurement, and Control	`https://dynamicsystems-asmedigitalcollection-asme-org.access.library.unisa.edu.au/issues.aspx`
AIAA Journal of Guidance, Control, and Dynamics	`https://arc-aiaa-org.access.library.unisa.edu.au/journal/jgcd`

[May71] Otto Mayr. *Feedback Mechanisms: In the Historical Collections of the The National Museum of History and Technology.* Smithsonian Institution Press, 1971.

[Mou07] Kannan M. Moudgalya. *Digital Control.* John Wiley & Sons Ltd, 2007.

[O'D09] Aidan O'Dwyer. *Handbook of PI and PID controller tuning rules*, volume 57. World Scientific, 2009.

[rM08] K.J. Åström and R.M. Murray. *Feedback Systems: An Introduction for Scientists and Engineers.* Princeton University Press, 2008.

[Sid13] Nazmul Siddique. *Intelligent Control: A Hybrid Approach Based on Fuzzy Logic, Neural Networks and Genetic Algorithms.* Springer International Publishing, 2013.

[Żak03] Stanislaw H. Żak. *Systems and Control.* Oxford University Press, 2003.