# Under Control: A Control Theory Introduction for Computer Scientists

Quentin Guilloteau, Sophie Cerf, Raphaël Bleuse, Bogdan Robu, Eric Rutten

# Under Control: A Control Theory Introduction for Computer Scientists

Quentin Guilloteau[4] Sophie Cerf[2] Raphaël Bleuse[1] Bogdan Robu[3] Eric Rutten[1]

***Abstract*—This hands-on tutorial aims at introducing the basis of Control Theory, applied to the runtime management of computing systems, through a practical example. This tutorial is composed of two parts. In the first part, we present the motivation and classical tools of Control Theory: formulation, modeling, and controllers. During the second part, the participants will experiment with the concepts presented in the first part through *interactive* computational documents, via a Jupyter Notebook. Attendees will need a computer with an internet connection and a "recent" web browser, and only basic programming knowledge. The tutorial is available at the following URL: https://control-for-computing.gitlabpages.inria.fr/tutorial/**

***Index Terms*—autonomic computing, control of computing systems, control theory, feedback management**

## I. General Information

This hands-on tutorial is an introduction, addressed to beginners in control theory. Its duration may vary between participants, between 3 and 4 hours. It is aimed for students, academics, and industry. Table I summarizes the general information about this tutorial.

This tutorial does not have any strong prerequisite for the participants. On the technical side, participants are required to have access to a computer with an internet connection and a "recent" browser (Firefox 90+, Chromium 89+) which supports WebAssembly.

On the knowledge side, participants need basic programming knowledge. Some mathematical analysis knowledge can be useful to fully understand the section explaining the principles behind control theory guarantees, however it is not necessary to understand the key principles.

[1]: Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LIG, Grenoble, F-38000, France; [2]: Univ. Lille, Inria, CNRS, Centrale Lille, UMR 9189 CRIStAL, F-59000 Lille, France; [3]: Univ. Grenoble Alpes, CNRS, Grenoble INP, GIPSA-lab, Grenoble, F-38000, France; [4]: University of Basel, Switzerland

| Characteristics | Details |
|---|---|
| Content level | Beginner |
| Duration | 3 hours, up to 4 hours |
| Target Audience | Industry & Academia |
| Tutorial format | Hands-on |
| Prerequisites | Computer with internet |

TABLE I: General Information

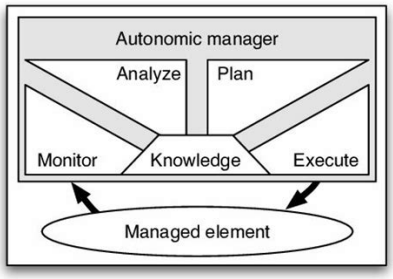



Fig. 1: MAPE-K Loop representation

## II. Introduction

### A. Motivation of Control for Computing

Computing systems are getting more and more complex. The software stacks are growing, and are executed on top of complex hardware. The behavior of applications is thus becoming extremely difficult to predict or model correctly in the design phase. We need regulation to provide Quality-of-Service guarantees.

One approach consists in building a theoretical model of the system, and design a complex configuration algorithm based on this model. Such an approach is limited by the quality of the model, and its ability to capture all runtime events. Modelling all the possible cases, and behaviors is tedious and error-prone.

A different approach to regulate the behavior of a computing system is to periodically take measurements of relevant metrics, and adapt the configuration based on these measurements. This runtime approach is called closed-loop, and this is the interest of the Autonomic Computing community [17].

The Autonomic Computing formulation relies on the *MAPE-K* loop (Figure 1), called after the different phases of the loop: **M**onitor, **A**nalyze, **P**lan, **E**xecute, and **K**nowledge. There are several ways to perform this runtime adaptation. One can use ad-hoc solutions based on arbitrary rules or IA for example [24], see Figure 2. These solutions, however, do not often come with guarantees on the closed-loop system. On the contrary, the Control Theory field has been applying math-proven methods to closed-loop physical systems for decades. But the application of Control Theory to computing systems is only quite recent. Figure 3 represents a feedback loop from the point of view of Control Theory.

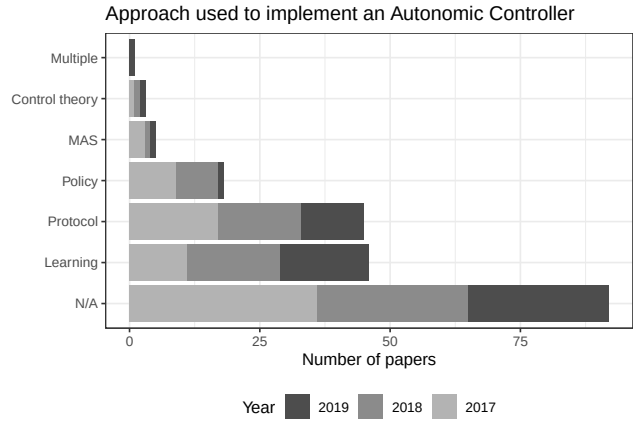



Fig. 2: Approaches used to implement an autonomic controller in 210 surveyed papers of three autonomic computing conferences. Regenerated from [24]

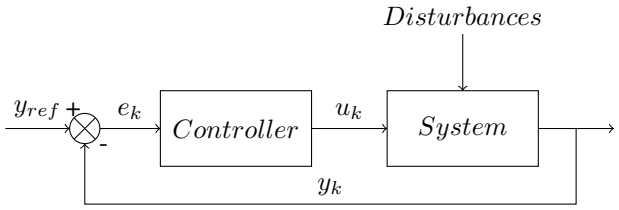



Fig. 3: Control Loop representation. $u_k$ is the control action, $y_k$ the sensed measure, $y_{ref}$ its desired reference value, and $e_k$ is the corresponding error.

### B. Motivation of this tutorial

Despite its promising results and guarantees, the use of Control Theory as tool for autonomic systems is only limited, possibly due to very few interactions between the Control Theory and Computer Science fields, in universities and industry. This tutorial aims at bringing accessible Control Theory concepts to computer scientists.

While a theoretical approach can be given on the topic of control for computing systems [23], we advocate that a complementary hands-on, interactive experience with computational notebooks is a suitable tool for computer scientists to favor the actual usage of Control Theory in their systems.

### C. Impact of this tutorial

This tutorial aims at introducing Control Theory, a method particularly suited for autonomic management of dynamic systems. It explains Control Theory to computer scientists in a simple and (almost) mathematical-free way. It allows understanding the principles of the guarantees that Control Theory provides. After this tutorial, attendees will be able to apply basic Control Theory tools and practices to their own systems, but also will have the knowledge and understanding to discuss and collaborate with experts in Control Theory.

At the end of the tutorial, the attendees should have:

- understood the motivations, methodology, and basic tools (PID Controller) of Control Theory
- performed an "identification" of a system
- designed a Proportional (P) Controller for this system, and understood the limits of the controller
- designed a Proportional-Integral-Derivative (PID) Controller for the same system
- apprehend advanced control techniques
- formalize a control closed-loop of their own system.

## III. Examples of applications

In the following, examples of successfull application of control theory on computing systems are presented. Each time, we briefly describe the system, emphasize the sensor(s) and actuator(s) used, as well as the type of controller. Examples have been selected to reflect diversity.

### A. Runtime powercaping to regulate performance in HPC

In High Performance Computing, applications have varying behaviors, with more or less computing/memory/communication needs. Energy savings can be obtained, e.g., by leveraging the processors' power when the application is in a memory intensive phase. Proportional-Integral Control [3] and Adaptive Control [14] have been used to ensure applications' progress while acting on RAPL power limit.

### B. Resource harvesting at cluster level

A regulation problem arises in the context of CiGri [8], where the general objective is to harvest unused computing resources by injecting low-priority jobs in order to maximize the utilization of the computing cluster. However, filling up the CPUs can cause bottlenecks on other parts of the infrastructure, e.g., memory access. Therefore, the more specific objective is to control the load of the distributed file-system of a computing cluster by adapting the number of low-priority jobs to submit to the cluster's scheduler, using various controllers like PI [11], adaptive control [22], and Model-Free Control [12], as well as coordination with the system scheduler [9].

### C. Dynamic privacy preservation for mobile devices users

In the mobile computing context, users sharing their personal data to third-party services (i.e., position shared to a navigation app) can ensure that their privacy is respected, e.g., by using protection mechanisms. Users' privacy requirements can however vary in time, as it implies trading-off utility of the service. Control can be used to ensure that a privacy metric keeps a minimal value, by acting on protection mechanisms' parameters with a PI [4], or directly by computing the protected positions with optimal control [20].

### D. More examples

More examples can be found in the literature e.g., "brown-out" approaches to build more robust cloud applications by control for bounding response times [18, 21], feedback scheduling in real-time systems [27], MIMO control of CPU

and Memory in a web server [6], dynamic adaptation of the bit rate of a live video streaming in order to provide a "Quality of Experience" (quality of the image, fluidity, etc.) for the users[5]. More references are available in overview or survey papers like [26, 23, 19, 25].

## IV. Planned Content

### A. Introductory presentation (20 minutes)

The tutorial begins with a 15 to 20-minute presentation by the speaker. This presentation aims to motivate the use of Autonomic Computing [17] and the use of the feedback loop, as presented in Section II. It then exposes regulation problems that arise in Computer Science. We discuss the different approaches to answer such regulation problems: ad-hoc solutions, rule-based solutions, artificial intelligence based solutions, etc. We then introduce Control Theory, its principle, closed-loop formulation and its methodology [7]. Then, the different types of controllers are introduced, including the Proportional-Integral-Derivative (PID) Controller. The properties desired for a closed-loop system are discussed.

A version of the slides of this presentation is available online [10]. We recall here the essential elements in the Control Theory formulation, and the next sections dive more in details on the methodology steps and controllers. The theoretical part of the tutorial is based on the book of Joseph L. Hellerstein e.a. [13].

The main tool of Control Theory is the *Controller*, that links the control error, *i.e.*, the distance between the desired state of the system and the current state of the system, to the next value of the actuator. There exists a large corpus of controllers with different behavior and properties. We mainly focus on the classical *Proportional-Integral Controller*, as it is the widest used control technique, and also introduce the participants to mode advances controllers, such as Model Predictive Control and Feedforward control.

The presentation also aims at introducing important terminology, concepts, and notations of Autonomic Computing and Control Theory, such as *actuator* ($u$), *sensor* ($y$), *reference value* ($y_{ref}$), *control error* ($e$), etc.

After this presentation, the attendees are invited to connect on the tutorial website. Figure 4 depicts the visual of the website of the tutorial. The left panel shows the different steps of the tutorial. The right panel hosts the *interactive* notebooks where the attendees will interact via sliders with the simulated system. These notebooks do not require deploying an instance of JupyterLab, but instead rely on JupyterLite [15], a web-assembly powered version of JupyterLab executing directly in the attendees' browser without the need for a server. Hence, the tutorial is permanently available, even outside the tutorial time slots. Instructions on how to interact with a Jupyter notebook in recalled in the front page of the tutorial.

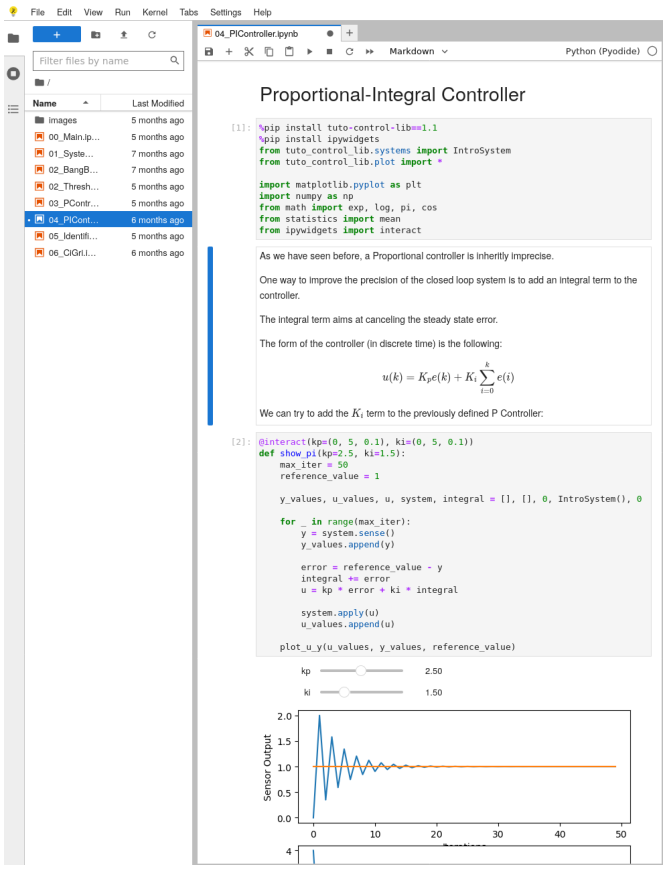



Fig. 4: Screenshot of the tutorial web interface

**Take Away: Introductory Presentation**

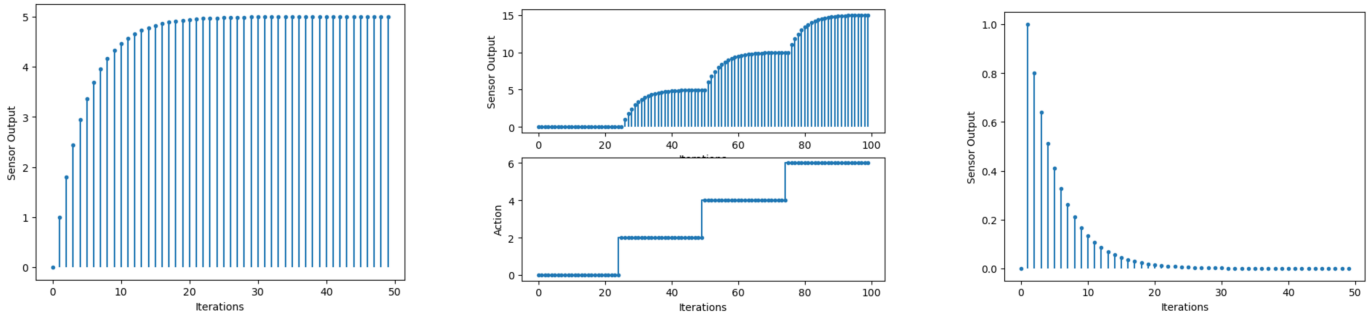
At the end of this step, attendees should have understood the main motivation and high level concepts of Autonomic Computing and Control Theory, and understand the closed-loop approach.

### B. Introduction to the simulated system (15 minutes)

In this first step on the notebooks, attendees get acquainted with the objects that they will manipulate in the rest of the tutorial. We introduce a simulated system that the attendees will learn to control/regulate in the following steps of this tutorial. The simulated system takes the form of a Python class with three methods:

- `sense`: takes no argument, and returns the value of the sensor of the system. This signal is often referred to as measurement or output signal in the control terminology.
- `apply`: takes one argument – the desired value of the actuator – and apply this value to the actuator. This method also *causes simulation time to progress*. The control action is usually called input signal in Control Theory.
- `get_time`: returns the current time, or iteration of the system.

Here is a small scenario:




(a) Single step response (b) Multiple steps response (c) Dirac response

Fig. 5: Response of the system for different types of control actions in open-loop.

```
> system = IntroSystem()
> system.get_time()
0
> system.sense()
0
> system.apply(2)
> system.get_time()
1
> system.sense()
1.0
> system.apply(1)
> system.get_time()
2
> system.sense()
1.3
```

Attendees observe the behavior of the system when applying a constant actuator value, and when the value of the actuator abruptly changes. Figure 5 shows different types of control signals. Figure 5a depicts the response of the system to a constant control action, and Figure 5b depicts the response of the system to a *step* signal. We observe a *transitory* state before the system reaches a converged state. This dynamics, a sort of inertia in the system, is a key behavior that makes the configuration challenging, particularly if one needs to avoid unsafe transient states. If instead of submitting the system to a constant control signal, we can also submit an impulse, or Dirac, we obtain the response of Figure 5c. Note that this behavior corresponds to a first order system. Details on the model is, however, set aside for the attendee at this point, since it is only needed for the demonstration on the tuning of the P controller in Section IV-D.

**Take Away: System under study**

At the end of this step, attendees should have understood how the simulated system works, its dynamics behavior, and how to interact with it through a sensor and an actuator.

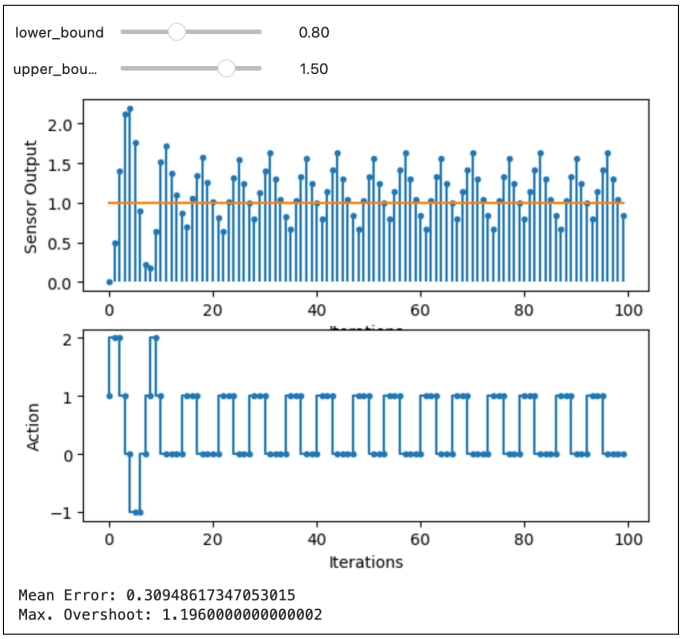


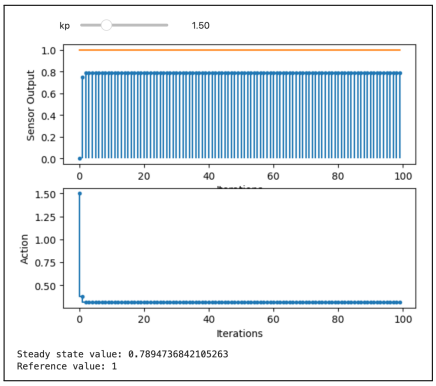

Fig. 6: Behavior of a threshold-based regulation

### C. Designing a threshold-based solution (20 minutes)

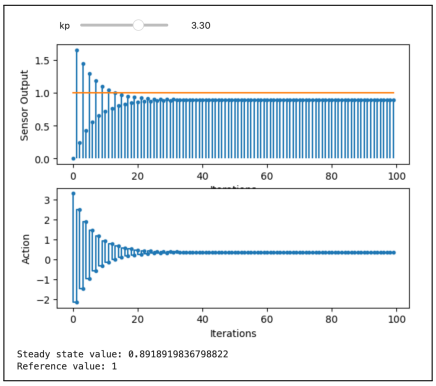
In this step, attendees are asked to design a first solution to regulate the system's measure around a desired value. The objective is to highlight that choosing the adequate action is not straightforward, especially due to the dynamics of the system. At first, a threshold-based approach is proposed, as a simple solution that do not require Control Theory knowledge, however it shows some limitations.

The threshold-based algorithm is already implemented and can be described as: If the value of the sensor is greater than `upper_bound` then decrease the value of the actuator by 1. If the value of the sensor is smaller than `lower_bound` then increase the value of the actuator by 1. Else, keep the current value of the actuator. The values of `upper_bound` and `lowerer_bound` can be set by the attendee, see Figure 6.

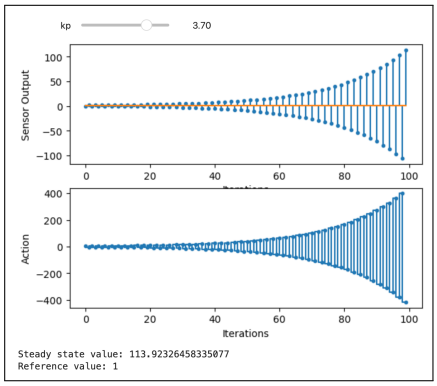
The objective is to show to attendees that finding acceptable values for the thresholds is a tedious and experiments-driven




(a) Small gain ($K_p = 1.5$)




(b) Medium gain ($K_p = 3.3$)




(c) Large gain ($K_p = 3.7$)

Fig. 7: Behavior of a Proportional Controller for different values of the gain $K_p$for the simulated system.

task, and always results in an oscillating sensor signal. Moreover, such threshold-based solutions lead to non-converging systems which oscillate (see Figure 6), and are thus non-satisfactory.

Based on our experience with the previous sessions of this tutorial, attendees appreciate this step as they usually set up a small competition to see who can find the threshold values yielding the best control.

**Take Away: Threshold-based Solution**

At the end of this step, attendees should have understood that threshold-based solutions are tedious to set up, that there is no methodology to do so, and that the resulting measure signal has an oscillatory behavior.

### *D. Designing a Proportional Controller (30 minutes)*

We introduce a first controller, that overcomes the limitations of the threshold-based approach by providing a design methodology and tunable measurement behavior. In this step, we present the simplest controller of Control Theory: the *Proportional Controller*.

The formulation of the Proportional Controller is as follows:

$$u_k = K_p \times e_k \tag{1}$$

where $u_k$ is the value of the actuator, $e_k$ is the control error at iteration $k$ ($e_k = y_{ref} - y_k$), and $K_p$ is a gain.

This controller has a single parameter to define ($K_p$), and a limited set of behaviors, but its study lays the bedrock for the following steps of the tutorial.

We then introduce the attendees to important properties of a controller and closed-loop systems (depicted in Figure 8):

- *Stability*: the capacity for the controlled system to reach a *converged state*, *i.e.*, no divergence.
- *Precision*: the capacity for the controlled system to converge close to the reference, measured as an error.
- *Settling Time*: the time required by the controlled system to reach stay within $\pm 5\%$ of the reference value.

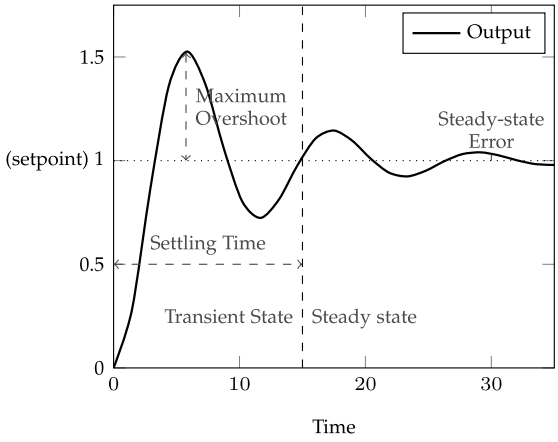



Fig. 8: Representation of the important properties of a controller and closed-loop system. Taken from [26]

- *Maximum Overshoot*: the maximum point above the reference value.

We ask the attendees to experiment with different value of the gain $K_p$. Figure 7 depicts 3 types of behavior of a Proportional Controller. With small gains (Figure 7a), the controller quickly reaches a stable state, but does not converge close to the desired reference value (straight orange line). When we increase the gain (Figure 7b), the controller gets closer to the reference value, but we see some oscillations and overshoots the reference value, but the system still ends up converging. When we push even further the gain $K_p$ (Figure 7c), then the oscillations are large and increasing: there is no convergence, the system is unstable.

These properties are then explained mathematically, and how they are linked to the value of the $K_p$ – the gain of the Proportional Controller. The tutorial guides the attendees to theoretically find the value of the gain $K_p$ that meets the different closed-loop properties. Note that these mathematical explanations assume the system is a first-order one. The conditions on the gain $K_p$ can then be observed on the interactive plot of the Proportional controller, to illustrate the

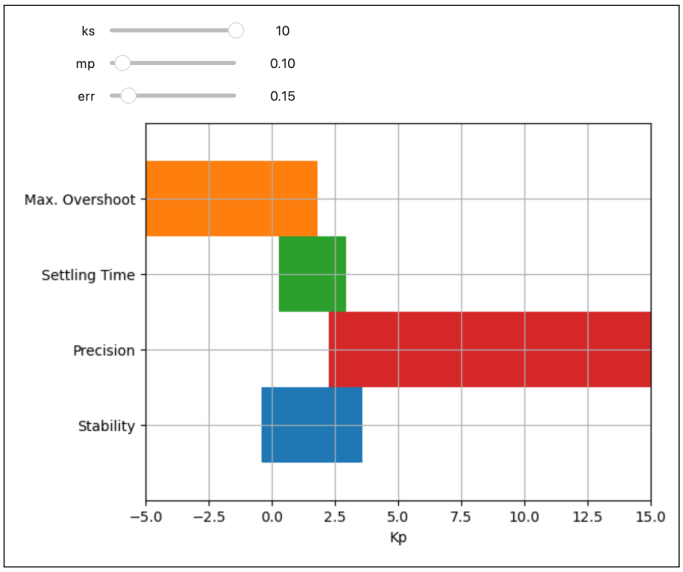



Fig. 9: Interactive representation of the tradeoff between the different properties of a Proportional controller to find a suitable value of $K_p$

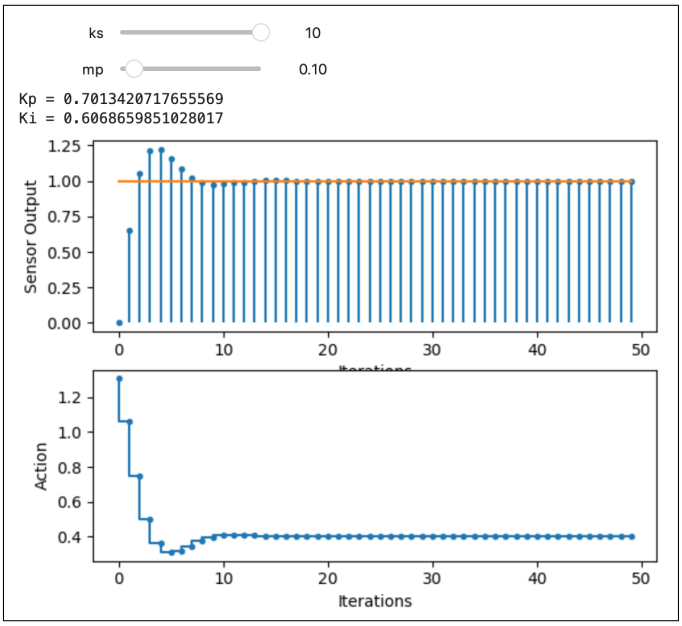



Fig. 10: Proportional-Integral controller with sliders on the desired properties.

considered properties and the link with the gain.

Figure 9 depicts the tradeoff between the different properties of a Proportional controller to find a suitable value of $K_p$. Attendees can then use the sliders on the properties of the controller (maximum overshoot, settling time, precision) to find a value of $K_p$ that satisfies their properties. Attendees can trade-off some precision and find a $K_p$ value around 1.25, or increase the maximum overshoot and find a $K_p$ value of around 2.5. The example provided is a first order system: $y_{k+1} = ay_k + bu_k$ with $a = 0.8$ and $b = 0.5$.

**Take Away: P Controller**

At the end of this step, attendees should have understood the behavior and tuning of the Proportional Controller, and its limitations: its precision, and tradeoffs between the different properties.

### *E. Designing a Proportional-Integral Controller (20 minutes)*

To overcome the precision limitation of the P controller, we introduce the *Proportional-Integral Controller* (PI).

The formulation of the PI (in discrete time) is as follows:

$$u_k = K_p \times e_k + K_i \times \sum_{i=0}^{k} e_i \tag{2}$$

where $K_i$ is the gain of the integral part.

There are now two gains to set up: $K_p$ and $K_i$. Contrary to the previous step of the tutorial for the Proportional controller, we do not dive into the mathematical proof of how to find the relations to set up the gains, but directly use the classical pole placement method [13]. The attendees can set their desired settling time $k_s$ and maximum overshoot $mp$, and the computed gains $K_p$ and $K_i$ are computed, see Figure 10. The objective here is not to focus on the tuning method, but rather to show that the integral action cancels the steady-state error.

Attendees can interact with the sliders for the gains $K_p$ and $K_i$ and observe the impact of the Integral term. Interesting configurations of the gains could be: $(K_p \neq 0, K_i = 0)$, $(K_p = 0, K_i \neq 0)$, or $(K_p \neq 0, K_i \neq 0)$.

**Take Away: PI Controller**

At the end of this step, attendees should have understood that using a PI instead of a P controller brings precision to the closed loop system. Attendees should also be introduced to the limits of the PI, namely that a large value of $K_i$ can lead to unstable systems.

### *F. Proportional-Integral-Derivative Control (15 minutes)*

One limitation of the Proportional control is the trade-off between rapidity and overshoot. As one increases the $K_p$ parameter, the settling time lowers, while overshoot increases. The PI controller suffers the same drawback.

A derivative action can be added to improve the quality of the measure signal, i.e., reduce the oscillations. Indeed, adding a component in the control action formula that is proportional to the rate of change of the error allows damping of the action when close to convergence, and boosting it when there are large changes in the error (e.g., at initial moments). The derivation acts as an anticipation action, predicting a continuation in the error signal trend (increase or decrease).

The derivative control is however sensitive to noise, as it amplifies high frequency signals, and is thus not always used in practice.

**Take Away: PID Controller**

By now, the attendees should understand which properties are improves by each element of a Proportional-Integral-Derivative controller, and know the trade-offs tuning allows.

### G. Performing the identification of a system (30 minutes)

In the previous steps, the attendees were implicitly relying on the parameters of the underlying model for the tuning of the controller gains. In practice on computing systems, these parameters are unknown and must be found through what control theorists call, *identification*. This step of the tutorial presents how to design the experimental identification of our simulated system, and find the parameters of the model via the Least Mean Square algorithm.

First, attendees focus on a system without any noise, where the identification of the parameters is straightforward. Then, we introduce noise in the system to make the process more realistic. The noise makes the identification more difficult, and the estimation of the model parameters less precise. However, we show that even with less precise parameters, a PI Controller is still able to regulate the system, illustrating that modeling is only an intermediary step in the Control Theory methodology, focussed on closed-loop performances.

**Take Away: Identification**

At the end of this step, attendees should have understood that the model parameters needed for controllers' tuning can be found experimentally by identification, and apprehend its challenges.

### H. Envisioning the variety of Controllers (30 minutes)

So far, the tutorial focused on PID feedback controllers, that compute the action based on the measured system states. Other types of controllers can be used, either to provide further guarantees, or when the system shows challenging properties (e.g., non-constant sampling time, varying model parameters).

In cases where the system undergoes some disturbances, i.e., a non-controllable signal affects the measure signal, it can be useful to design a controller able to *anticipate* rather than just *react*. Measuring the disturbance, one can adjust the control action accordingly with a so-called feedforward controller. This allows for a perfect compensation of the disturbance, provided some hypothesis are fulfilled (such as the control acts faster than the disturbance), while a purely feedback control, as a PID, would only compensate for the disturbance once it affects the measure. Feedforward control is often combined with feedback controllers, to ensure the tracking of the reference.

Another approach when designing a controller is to solve an optimization problem. Model Predictive Control considers the problem of choosing a set of control actions over a future horizon, so as to minimize a cost function (often composed of elements such as reference tracking $y_{ref} - y_k$ and control amplitude $u_k$). The first control action is applied, and the MPC is solved again in the next time step. MPC has the advantage of providing optimal *trajectories*. However, as it relies on future steps prediction, its performance can degrade if the modeling is not reliable. This limitation is mitigated by the re-computation of the optimal solution at each time step.

**Take Away: Advanced controllers**

At this point, attendees should understand that a controller is simply a function giving the action to perform. The different controllers enable to have various guarantees on different types of systems.

### I. Formulate your own control problem (1 hour)

Now that the steps of design of a controller (and model) are understood, the attendees are invited to consider their own self-adaptive systems as a control loop. Those with no specific system under study can join other groups. Following the methodology of applying control theory to computing systems [7, 3], the attendees are guided with a list of questions to identify the goals of the adaptation, the challenges that make it non-straightforward, the potential actions that can be performed at runtime, and the suitable measure of the state of the system. Actuators and sensors are analyzed to ensure that they fit the requirements for control, e.g., continuous-like signals and inertia in the actions. A first attempt of analytical modeling is performed. A reflection on the specificities and challenges of the system aims at selecting a suitable control form.

**Take Away: Formulating a control problem**

At the end of this part, the attendees should have all the elements formulated (e.g., action, measure) in order to apply identification and control design on their own system.

### J. Application to more complex (simulated) system (optional)

In the last step of the tutorial, the attendees are invited to use what they learned in the previous steps to design a controller for a more realistic system inspired from the one mentioned in III-B. The system is a simulation of CiGri [8], a computing grid middleware, to harvest the idle resources of a set of computing machines. CiGri takes as input Bag-of-tasks applications, *i.e.*, a set of numerous independent jobs with similar characteristics. These tasks cannot be submitted directly to the scheduler of the computing cluster, as they would overload the scheduling algorithm. Hence, the role of CiGri is to submit periodically batches of tasks to the scheduler in order to use the idle resources of the computing cluster while not overloading the scheduler.

**Take Away: Application to more complex system**

At the end of this step, attendees should be able to apply the Control Theory methodology on a new system and design a controller.

## V. Previous Sessions

As of June 2024, there have been **7** sessions of this tutorial, for a total number of attendees around **110**. Table II summarizes the information about the previous sessions.

## VI. Perspectives

Perspectives of this tutorial include the development of an extension where attendees would be able to apply what they learned on a *real system*. We envision that attendees could connect to a testbed such as Grid'5000 [2] or Chameleon [16], deploy a real computing system, and design a Proportional Controller for this system.

The system described in [1] represents an interesting case for experimenting on a real system. In this system, a "for" loop is parallelized between threads: each thread receives a number of iterations of the loop to execute – a *task* –, and, once done, will ask for a new task until all the iterations are executed. In general, small task sizes lead to low load imbalance and increased performance. However, in [1], the authors observed that for their application, having load imbalance improved the performance of the application as the synchronization patterns of the loop were slightly desynchronized, and lead to less congestion on memory. Once, set up, the attendees would need to dynamically adapt the size of tasks based on the bandwidth of the memory to control the congestion on memory in order to exploit the trade-off between load imbalance and memory congestion.

A companion tutorial could also be offered to control scientists, to introduce them to computing systems as a subject of regulation. The objectives would be to illustrate the challenges specific to computing systems, such as variability between runs, multiplicity of sensing and actuation options, choice of the sampling frequency, unknown sources of disturbances/noise/variability, hardware dependence, etc. Such tutorial could consist in a simple interface (e.g. in Python or Matlab) allowing to perform open-loop and closed-loop experimentation, modify sensors/actuators, while hiding the complexity of the deployment of the computing application on a real testbed. We hope that such a setup could allow control theorist to easily design identification and control, tackling the challenges of computing systems with dedicated or novel control techniques.

## VII. Authors Biographies

### A. Quentin Guilloteau

Quentin Guilloteau holds a Ph.D. in Computer Science from Univ. Grenoble Alpes in France (2023) on the topic of applying Control Theory approaches to the regulation of High Performance Computing systems. He is currently a Post-doc at the University of Basel in Switzerland, working on multi-level scheduling. He is also interested in reproducible research and autonomic computing in HPC.

### B. Sophie Cerf

Sophie Cerf is a Research Scientist in the Spirals team at Inria center of the University of Lille. She got her Ph.D. from the Univ. Grenoble Alpes in 2019. Her research interests are Control Theory for Distributed Systems. She focuses her research on sustainable and social computing.

### C. Raphaël Bleuse

Raphaël Bleuse is an Associate Professor at Univ. Grenoble Alpes, France since 2019. He received is Ph.D. in 2017 from Univ. Grenoble Alpes. His research focuses on the combination of control theory with scheduling theory, aiming at designing pragmatic solutions for the allocation of resources on HPC systems and alike. His research ranges from modeling computing systems, designing controllers and scheduling algorithms to conducting reproducible experiments by simulation or on real hardware.

### D. Bogdan Robu

Bogdan Robu is an Associate Professor in the GIPSA-lab research laboratory of the Univ. Grenoble Alpes. He received his Ph.D. in 2010 from the University of Toulouse. His research focus is on applying Control Theory techniques (discrete, continuous, hybrid) to Distributed Systems for the runtime dynamic management of learning algorithms and neural networks, Cloud/Fog software, parallel computing systems as well as IoT systems.

### E. Eric Rutten

Éric Rutten (Ph.D. 90, Habil. 99 at U. Rennes, France) is researcher at Inria in Grenoble, France, where he heads the Ctrl-A team. After working in the past on embedded, real-time and robotic systems, contributing methods and tools to support computing for control systems, he then reversed the perspective by considering control for computing systems, exploring the potentials of Control Theory (discrete and continuous) in the runtime management of self-adaptive Cloud/Edge or High Performance Computing systems, with objectives of self-(re)configuration, self-optimization, self-healing, and self-protection.


## Acknowledgment

Experiments presented in this paper were carried out using the Grid'5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations (see https://www.grid5000.fr).

| Event | Date | Speaker | Attendees | Duration | Details |
|---|---|---|---|---|---|
| FlexScience 2024 | 03/06/24 | Quentin Guilloteau | $\simeq 10$ | 1 hour | Attendees of a conference in parallel and distributed computing, beginners |
| ANR ADAPT | 14/03/24 | Sophie Cerf | $\simeq 10$ | 1 hour | Research project seminar, beginners |
| VELVET Days 2023 | 13/12/23 | Sophie Cerf | $\simeq 25$ | 1 hour | Event on reconfiguration, adaptation, and DevOps, beginners |
| ComPas 2023 | 04/07/23 | Quentin Guilloteau | $\simeq 10$ | 2 hours | Attendees of a conference in parallel and distributed computing, beginners |
| Spirals Seminar | 27/06/23 | Sophie Cerf | $\simeq 35$ | 2 hours | Research team seminar, audience of self-adaptive researchers |
| WAX GLSI | 13/06/23 | Quentin Guilloteau | $\simeq 10$ | 2 hours | Research lab seminar, beginners |
| CtrlA Seminar | 21/04/23 | Quentin Guilloteau | $\simeq 10$ | 2 hours | Research team seminar, initiated attendees |

TABLE II: Summary of the previous sessions

## REFERENCES


[1] A. Afzal, G. Hager, and G. Wellein. Desynchronization and wave pattern formation in MPI-parallel and hybrid memory-bound programs. In *High Performance Computing: 35th International Conference, ISC High Performance 2020, Frankfurt/Main, Germany, June 22–25, 2020, Proceedings 35*, pages 391–411. Springer, 2020.

[2] D. Balouek, A. C. Amarie, G. Charrier, F. Desprez, E. Jeannot, E. Jeanvoine, A. Lèbre, D. Margery, N. Niclausse, L. Nussbaum, et al. Adding virtualization capabilities to the grid'5000 testbed. In *Cloud Computing and Services Science: Second International Conference, CLOSER 2012, Porto, Portugal, April 18-21, 2012. Revised Selected Papers 2*, pages 3–20. Springer, 2013.

[3] S. Cerf, R. Bleuse, V. Reis, S. Perarnau, and E. Rutten. Sustaining performance while reducing energy consumption: a control theory approach. In *Euro-Par 2021: Parallel Processing: 27th International Conference on Parallel and Distributed Computing, Lisbon, Portugal, September 1–3, 2021, Proceedings 27*, pages 334–349. Springer, 2021.

[4] S. Cerf, B. Robu, N. Marchand, and S. Bouchenak. Privacy protection control for mobile apps users. *Control Engineering Practice*, 134(May):105456, May 2023. DOI: 10.1016/j.conengprac.2023.105456. URL: https://hal.science/hal-03977386.

[5] L. De Cicco, S. Mascolo, and V. Palmisano. Feedback control for adaptive live video streaming. In *Proceedings of the second annual ACM conference on Multimedia systems*, pages 145–156, 2011.

[6] Y. Diao, N. Gandhi, J. L. Hellerstein, S. Parekh, and D. M. Tilbury. Using mimo feedback control to enforce policies for interrelated metrics with application to the apache web server. In *NOMS 2002. IEEE/IFIP Network Operations and Management Symposium.'Management Solutions for the New Communications World'(Cat. No. 02CH37327)*, pages 219–234. IEEE, 2002.

[7] A. Filieri, M. Maggio, K. Angelopoulos, N. d'Ippolito, I. Gerostathopoulos, A. B. Hempel, H. Hoffmann, P. Jamshidi, E. Kalyvianaki, C. Klein, et al. Software engineering meets control theory. In *2015 IEEE/ACM 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, pages 71–82. IEEE, 2015.

[8] Y. Georgiou, O. Richard, and N. Capit. Evaluations of the lightweight grid cigri upon the grid5000 platform. In *Third IEEE International Conference on e-Science and Grid Computing (e-Science 2007)*, pages 279–286. IEEE, 2007.

[9] Q. Guilloteau. *Control-based runtime management of HPC systems with support for reproducible experiments*. Theses, Université Grenoble Alpes, Dec. 2023. URL: https://hal.science/tel-04389290.

[10] Q. Guilloteau, S. Cerf, E. Rutten, R. Bleuse, and B. Robu. Under Control: A Control Theory Introduction for Computer Scientists, Apr. 2023. URL: https://hal.science/hal-04460285. Tutorial to introduce Control Theory to Computer scientists.

[11] Q. Guilloteau, O. Richard, B. Robu, and E. Rutten. Controlling the Injection of Best-Effort Tasks to Harvest Idle Computing Grid Resources. In *ICSTCC 2021 - 25th International Conference on System Theory, Control and Computing*, pages 1–6, Iași, Romania, Oct. 2021. DOI: 10.1109/ICSTCC52150.2021.9607292. URL: https://hal.inria.fr/hal-03363709.

[12] Q. Guilloteau, B. Robu, C. Join, M. Fliess, É. Rutten, and O. Richard. Model-free control for resource harvesting in computing grids. In *Conference on Control Technology and Applications, CCTA 2022*, Trieste, Italy. IEEE, Aug. 2022. URL: https://hal.archives-ouvertes.fr/hal-03663273.

[13] J. L. Hellerstein, Y. Diao, S. Parekh, and D. M. Tilbury. *Feedback control of computing systems*. John Wiley & Sons, 2004.

[14] C. Imes, H. Zhang, K. Zhao, and H. Hoffmann. Copper: soft real-time application performance using hardware power capping. In *2019 IEEE International Conference on Autonomic Computing (ICAC)*, pages 31–41. IEEE, 2019.

[15] [Software] jupyterlite, 2024. URL: https://github.com/jupyterlite/jupyterlite, SWHID: ⟨swh:1:dir:9b5c2e6b11a3acab3bfa3772a082c930cbeb6ba9;origin=https://github.com/jupyterlite/jupyterlite⟩.

[16] K. Keahey, J. Anderson, Z. Zhen, P. Riteau, P. Ruth, D. Stanzione, M. Cevik, J. Colleran, H. S. Gunawi, C. Hammock, et al. Lessons learned from the chameleon testbed. In *2020 USENIX annual technical conference (USENIX ATC 20)*, pages 219–233, 2020.

[17] J. O. Kephart and D. M. Chess. The vision of autonomic computing. *Computer*, 36(1):41–50, 2003.

[18] C. Klein, M. Maggio, K.-E. Årzén, and F. Hernández-Rodriguez. Brownout: building more robust cloud applications. In *Proceedings of the 36th International Conference on Software Engineering*, pages 700–711, 2014.

[19] M. Litoiu, M. Shaw, G. Tamura, N. M. Villegas, H. Müller, H. Giese, R. Rouvoy, and E. Rutten. What Can Control Theory Teach Us About Assurances in Self-Adaptive Software Systems? In R. de Lemos, D. Garlan, C. Ghezzi, and H. Giese, editors, *Software Engineering for Self-Adaptive Systems 3: Assurances*. Volume 9640, LNCS. Springer, May 2017. URL: https://inria.hal.science/hal-01281063.

[20] E. Molina, M. Fiacchini, S. Cerf, and B. Robu. React to the Worst: Lightweight and proactive protection of location privacy. *IEEE Control Systems Letters*, 7:2371–2376, 2023. DOI: 10.1109/LCSYS.2023.3286989. URL: https://hal.science/hal-04128118.

[21] T. Nylander, C. Klein, K.-E. Årzén, and M. Maggio. Brownout cc: cascaded control for bounding the response times of cloud applications. In *2018 Annual American Control Conference (ACC)*, pages 3354–3361. IEEE, 2018.

[22] R. Pagano, S. Cerf, B. Robu, Q. Guilloteau, R. Bleuse, and É. Rutten. Making Control in High Performance Computing for Overload Avoidance Adaptive in Time and Job Size. In *Conference on Control Technology and Applications, CCTA 2024*, Newcastle upon Tyne, UK. IEEE, Aug. 2024.

[23] A. V. Papadopoulos. Designing self-adaptive software systems with control theory: an overview. In *2022 IEEE International Conference on Autonomic Computing and Self-Organizing Systems Companion (ACSOS-C)*, 2022. DOI: 10.1109/ACSOSC56246.2022.00027.

[24] B. Porter, R. R. Filho, and P. Dean. A survey of methodology in self-adaptive systems research. In *2020 IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS)*, pages 168–177, 2020. DOI: 10.1109/ACSOS49614.2020.00039.

[25] E. Rutten, N. Marchand, and D. Simon. Feedback control as mape-k loop in autonomic computing. In *Software Engineering for Self-Adaptive Systems III. Assurances: International Seminar, Dagstuhl Castle, Germany, December 15-19, 2013, Revised Selected and Invited Papers*, pages 349–373. Springer, 2017.

[26] S. Shevtsov, M. Berekmeri, D. Weyns, and M. Maggio. Control-theoretical software adaptation: a systematic literature review. *IEEE Transactions on Software Engineering*, 44(8):784–810, 2017.

[27] D. Simon, A. Seuret, and O. Sename. Real-time control systems: feedback, scheduling and robustness. *International Journal of Systems Science*, 48(11):2368–2378, 2017. DOI: 10.1080/00207721.2017.1316879. URL: https://hal-lirmm.ccsd.cnrs.fr/lirmm-01515226.