



HAL
open science

Under Control: A Control Theory Introduction for Computer Scientists

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

► **To cite this version:**

Quentin Guilloteau, Sophie Cerf, Eric Rutten, Raphaël Bleuse, Bogdan Robu. Under Control: A Control Theory Introduction for Computer Scientists. 2023, pp.1-11. hal-04460285

HAL Id: hal-04460285

<https://hal.science/hal-04460285v1>

Submitted on 15 Feb 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Distributed under a Creative Commons Attribution 4.0 International License

Under Control: A Control Theory Introduction for Computer Scientists

Session @ VELVET 2023

Quentin GUILLOTEAU¹ , Sophie CERF² , Eric RUTTEN¹ ,
Raphaël BLEUSE¹ , Bogdan ROBU³

¹ Univ. Grenoble Alpes, Inria, CNRS, LIG
Firstname.Lastname@inria.fr

² Univ. Lille, Inria, Central Lille, CRISTAL
Sophie.Cerf@inria.fr

³ Univ. Grenoble Alpes, CNRS, G-INP, Gipsa Lab
Bogdan.Robu@grenoble-inp.fr

2023-12-13

Goal of this tutorial

- Present the need for regulation of computing systems
- Show how computing systems can benefit from the Feedback loop approach
- Show what Control Theory can bring
- Quick presentation of the methodology
- Let you implement some Control Theory controllers!

The need for Regulation

- Complex systems
 - Crazy software stacks on crazy hardware
 - Many sources of unpredictability
 - \leadsto need for dynamic management
- Usual approaches:
 - simple rules
 - *if this then that*
 - very arbitrary constants
 - no guarantee
 - or model as much as possible
 - then solve “optimally”
 - very complex
 - guarantee only if model stands

We need another approach (guarantees + simplicity)

The Feedback Loop

The idea

Use the knowledge about the current state of the system to respond

Autonomic Computing (IBM 2003)

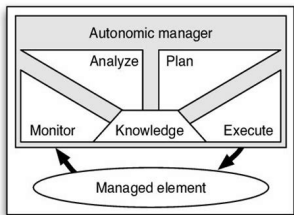


Figure: MAPE-K Loop

- auto-*
 - configuration
 - healing
 - **optimization**
 - protection
- main tool: MAPE-K loop
- Separation of concerns
- ≠ implementations of AC (rules, AI, control)

Some Examples

- Regulate the **heat of a processor** based on its frequency
 - Sensor: CPU thermometer
 - Actuator: DVFS
 - Reference: Desired CPU temperature
- Regulate the **waiting time of users** based on the number of available servers
 - Sensor: Waiting time of a request
 - Actuator: Number of servers to add/remove
 - Reference: Mean waiting time for the requests
- Regulate the **FPS of an online video rendering** based on the quality
 - Sensor: FPS
 - Actuator: depth of computation
 - Reference: 60 FPS

Control Theory

Goal

Act on dynamic system to reach desired behavior with guarantees

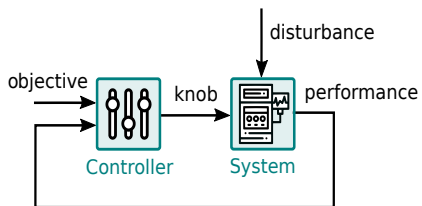


Figure: Control Loop

- Proven Guarantees
 - Stability
 - Robustness
 - Precision
- Long history in physical systems
- Only recent for CS
- Reusable methodology

Methodology

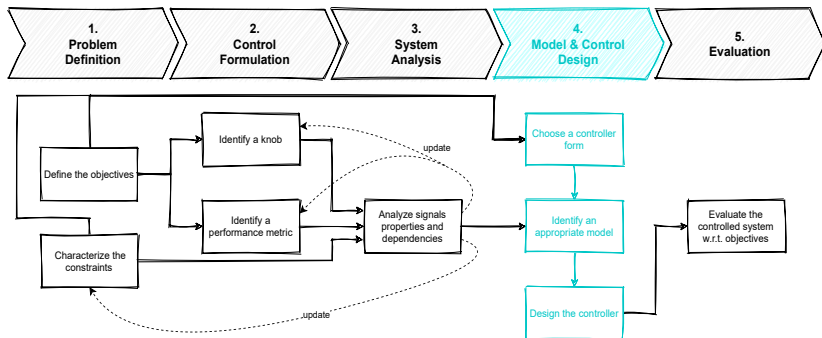


Figure: Control Theory Methodology

The (famous) PID Controller (discretized)

First, a **Model** ... (i.e., how does the system behave (Open-Loop))

$$\mathbf{y}(k+1) = \sum_{i=0}^k a_i \mathbf{y}(k-i) + \sum_{j=0}^k b_j \mathbf{u}(k-j)$$

... then a **PID Controller** (i.e., the Closed-Loop behavior)

$$\mathbf{u}_k = \mathbf{K}_p \times \text{Error}_k + \mathbf{K}_i \times \sum_k \text{Error}_k + \mathbf{K}_d \times (\text{Error}_k - \text{Error}_{k-1})$$

Sensors & Actuators

- Actuator: \mathbf{u}
- Sensor: \mathbf{y}
- Error: *Reference* – *Sensor*

Method

- 1 Open-Loop expe (predefined \mathbf{u})
- 2 Model parameters (a_i, b_j)
- 3 Choice controller behavior (\mathbf{K}_*)

Settling Time, Overshoot and Error

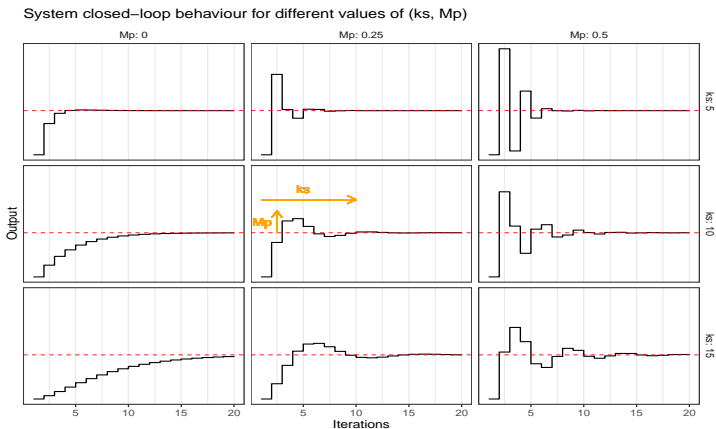


Figure: Closed Loop Behavior

The controller gains define this behavior!

Controllers

Goal

Map the error to the next action to reach desired system state with guarantees

Many (many) types of Controllers

- Feedback (PID, MFC, RST, etc.)
- Feedforward (proactive reaction to disturbances)
- Adaptive (change behavior at runtime)
- Model Predictive
- Event Based
- Optimal
- and more!

Your turn!

What you will do now

- 1 Play with a dummy system
- 2 Implement a naive Threshold-based controller
- 3 First introduction with Control Theory: P Controller
- 4 Implement a more precise controller (PI)
- 5 Perform an identification phase
- 6 **Implement a PI controller on a “real” system**

Starting Point

<https://tinyurl.com/Control4ComputingVELVET>