PID tuning
&
UNICOS PID auto-tuning

Benjamin Bradu
Ruben Marti Martinez
CERN, BE-ICS

On behalf of UNICOS-CPC Team

19th April 2016

Content

- Regulation loops and feedback control

- Manual PID tuning in open loop
  - Stable system
  - Unstable system

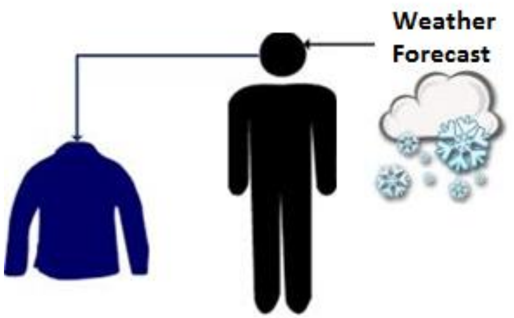
- PID auto-tuning tool in UNICOS
  - Relay Method
  - SIMC method
  - IFT method

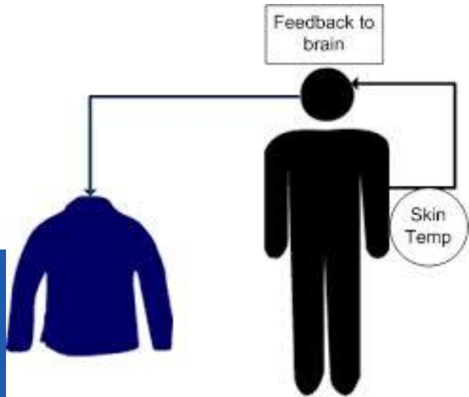
Regulation loops

- Allow to regulate **process variables** given by sensors (current, temperature, pressure, …)

- Make use of **actuators** (valves, pumps, heaters…) to fit different objectives in real-time:
  - Stay between acceptable ranges
  - Keep a constant set-point rejecting disturbances
  - Follow a set-point (stairs or ramps)

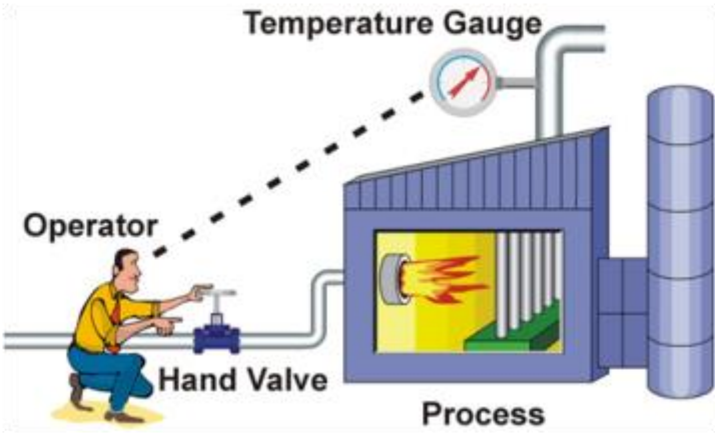
- 2 main techniques :
  1. Feed-Forward Control
  2. Feedback control



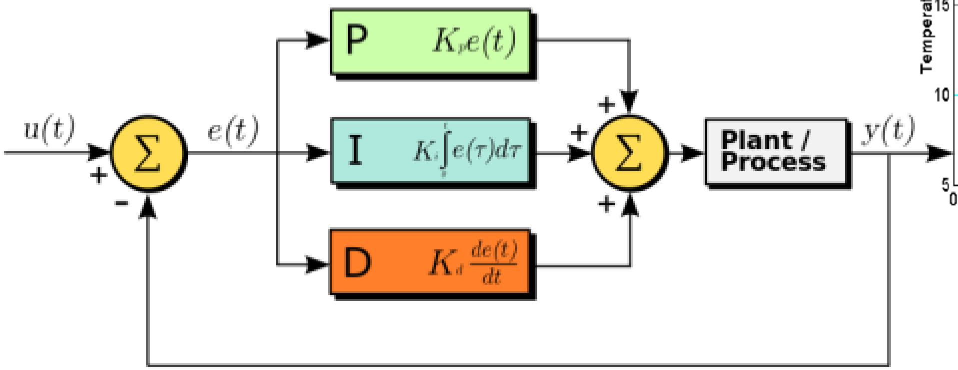


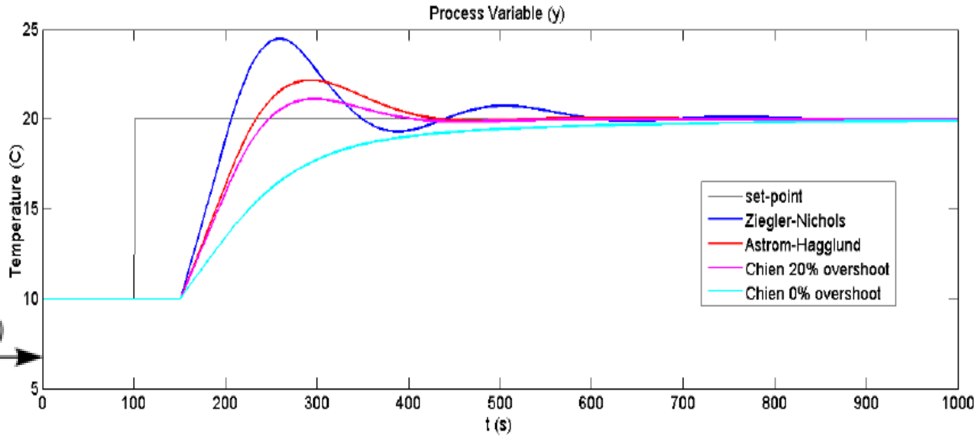
Feedback Control

- **Regulation loop** using a feedback (sensor) to compute the actuator position
  - y: Sensor to be controlled (process value)
  - u: Actuator to be manipulated (manipulated value)
  - r: Set-point to be respected (reference)



- **PID**: most common algorithm
  - Compute the **error**
  - **P**roportional action
  - **I**ntegral action
  - **D**erivative action



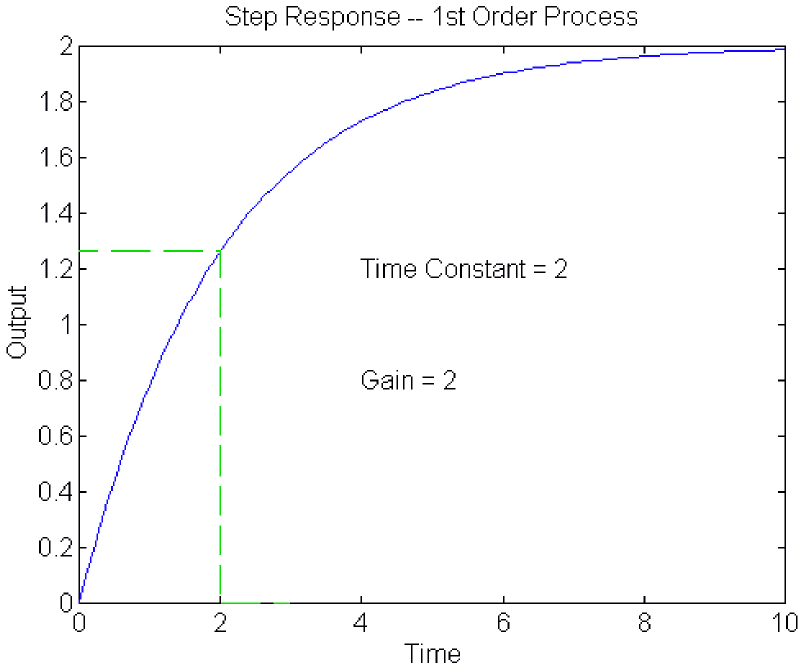


➔Then, you have to Tune your PID and find a compromise overshoot Vs time response.

Manual PID tuning: hypothesis

- Process is a linear 1st order system (stable or unstable)
  - Be sure to make the test around operation point to limit non-linear effects.
  - Most of processes can be approximated by a 1st order system.

Reminder about a **linear system**:
- A valve step between 10% ➔ 20% has the same effect than a step between 80%➔90%
- A valve step of 10% at 20 C or at 30 C has the same effect
- Conclusion: Nature is never linear (unfortunately for control)



Manual PID tuning: hypothesis

- Tuning is done for a Single Input/Single Output (SISO) system
  - Try to minimize the other regulation impacts when you do tuning.
  - For cascade tuning:
    - First: Tune the internal loop with the external loop disabled
    - Second: Tune the external loop with the internal loop enabled and tuned
  - For Split-Range tuning, use only one actuator during the tuning.

- Actuator speed is infinite
  - Be sure that your actuator can follow the controller output.

- There is no disturbance on the measurement
  - Try to reduce external disturbances during the tuning.
  - If measurement is noisy, it is necessary to apply a first order filter on the measured value.

PID tuning principle in open loop

1. Identify your process dynamics
    1. Open regulation loop (Controller in Positioning mode)
    2. Put a constant PID output close to your operation point
    3. Wait for stable situation (can be long)
    4. Apply a step on the PID output (up or down)
    5. Wait for stable situation (can be long)
    6. Identify your process dynamics (curve analysis)

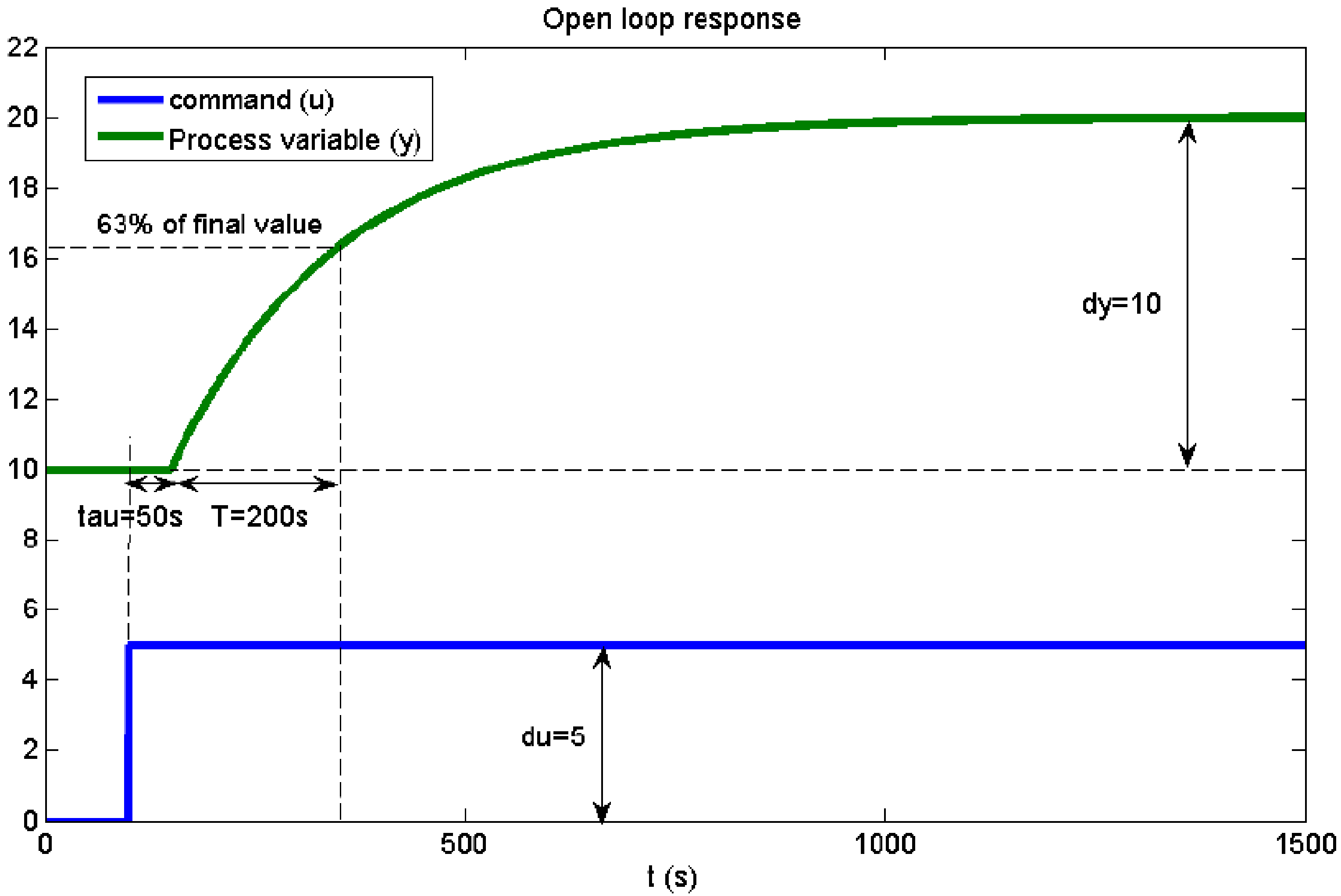
2. Compute PID parameters (1 minute)

3. Close the loop (Controller in regulation mode) to test

Open loop identification for stable system

1. Apply a step "du" on the PID output
2. When stable value, measure:
   - ✓ Gain: K=dy/du
   - ✓ Delay: tau
   - ✓ Time constant: T = time to reach 63% of final value

$$P(s) = \frac{K}{T \cdot s + 1} \cdot e^{-\tau \cdot s}$$



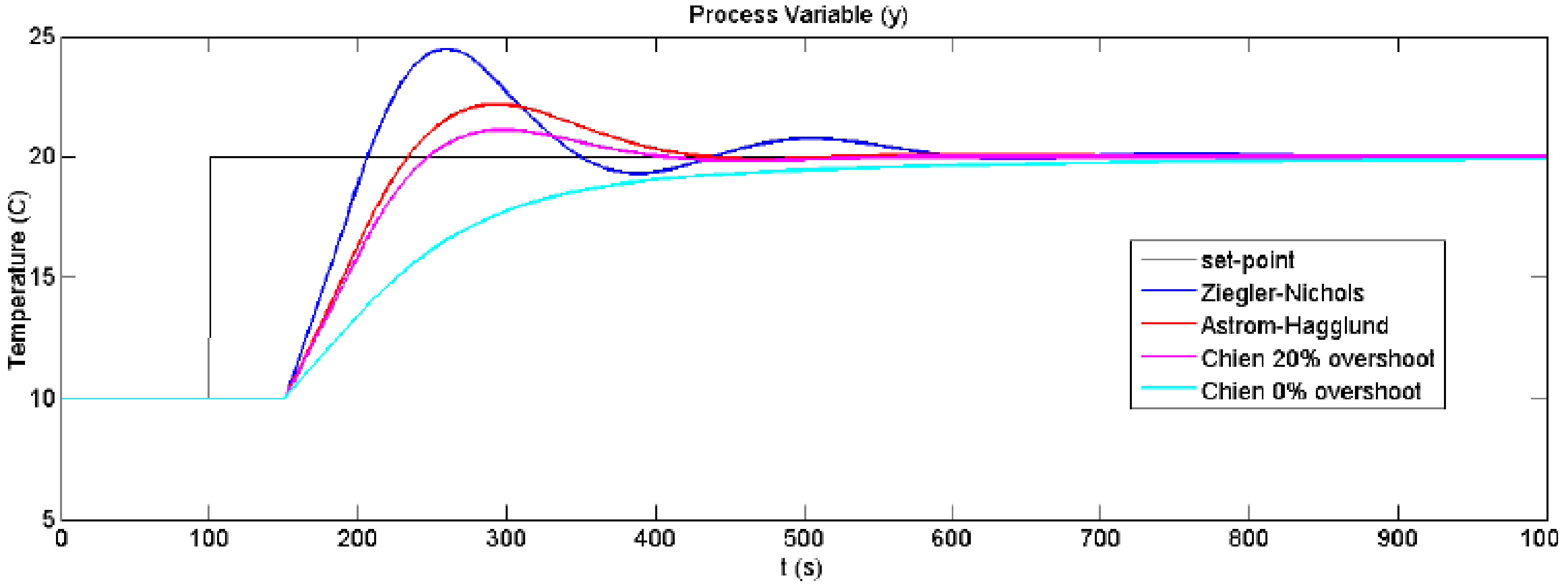
PI tuning for stable systems

There are about thousands of tuning rules for a 1st order system. Here are 4 examples:

TABLE 2.1 – Different PI tunning rules for stable systems

Tuning	Ziegler-Nichols	Astrom-Hagglund	Chien (20% overshoot)	Chien(0% overshoot)
Kc	$\frac{0.9 \cdot T}{K \cdot \tau}$	$\frac{0.63 \cdot T}{K \cdot \tau}$	$\frac{0.6 \cdot T}{K \cdot \tau}$	$\frac{0.35 \cdot T}{K \cdot \tau}$
Ti	$3.33 \cdot \tau$	$3.2 \cdot \tau$	T	$1.17 \cdot T$

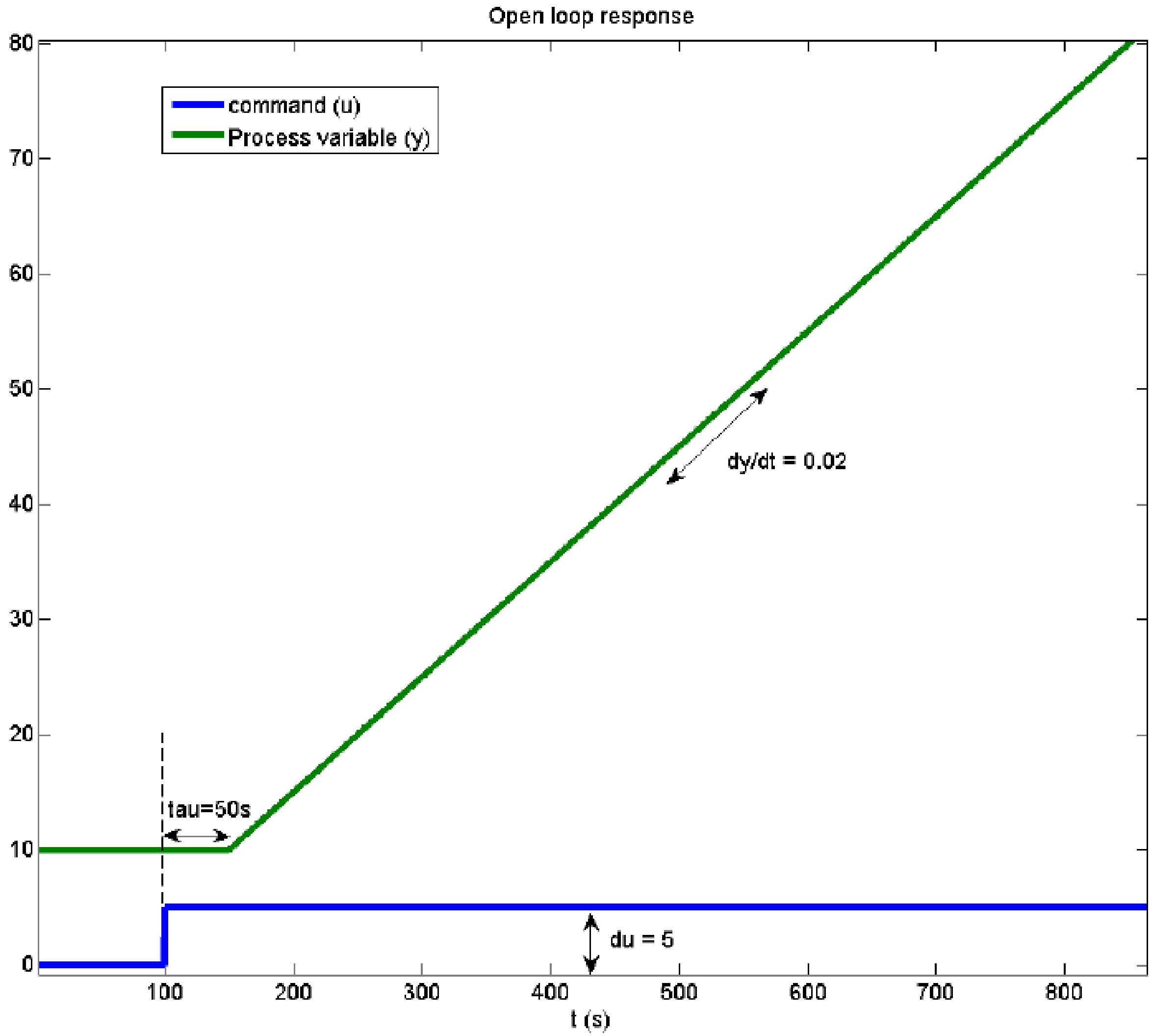
**This table is valid for a mixed PI structure (compatible with UNICOS)*



Open loop identification for unstable system

1. Apply a step “du” on the PID output
2. When stable value, measure:
   - ✓ Gain: K=dy/dt/du
   - ✓ Delay: tau

$$P(s) = \frac{K}{s} \cdot e^{-\tau \cdot s}$$



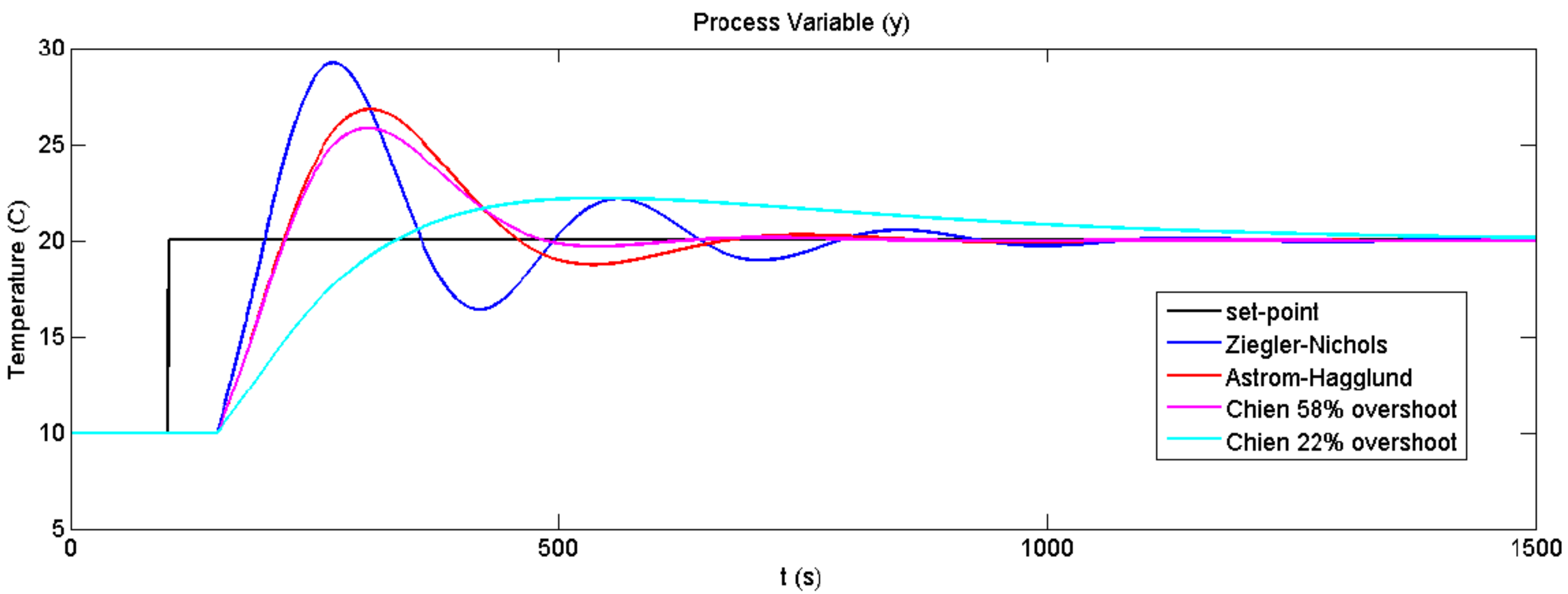
PI tuning for unstable systems

There are about thousands of tuning rules for a 1st order system. Here are 4 examples:

TABLE 2.2 – Different PI tunning rules for unstable systems

Tuning	Ziegler-Nichols	Astrom-Hagglund	Chien (58% overshoot)	Chien(22% overshoot)
Kc	$\frac{0.9}{K \cdot \tau}$	$\frac{0.63}{K \cdot \tau}$	$\frac{4 \cdot \tau}{6.25 \cdot K \cdot \tau^2}$	$\frac{10 \cdot \tau}{30.25 \cdot K \cdot \tau^2}$
Ti	$3.33 \cdot \tau$	$3.2 \cdot \tau$	$4 \cdot \tau$	$10 \cdot \tau$

**This table is valid for a mixed PI structure (compatible with UNICOS)*



PID validity domains

Domains	Regulation type
T/tau > 20	On/Off Regulation
10 < T/tau < 20	P regulation
5 < T/tau < 10	PI regulation
1 < T/tau < 5	PID regulation
T/tau < 1	PID not able to regulate correctly

UNICOS PID specificities

- For all regulation loops:
  - There is a **ramp on the set-point** to limit the error when the regulation is starting. This ramp has to be compatible with your tuning.
  - There is a **ramp on the actuator** to limit actuator jumps if PID output is too fast. This ramp has to be compatible with your tuning.

- PID algorithm uses of a mixed structure (ISA standard)

$$PID = K_c\left(1 + \frac{1}{T_i \cdot s} + \frac{T_d \cdot s}{1 + \frac{T_d}{T_{ds}} \cdot s}\right)$$

- PID regulation loops can be scaled or not:
  - No Scaling (CV): Engineering values are used
  - Input Scaling (CRYO): the set-point and the process value are scaled between [0-100 %] according to the sensor range. So, the proportional gain “Kc” should be scaled as: Kc (UNICOS) = Kc* (PMaxRan-PMinRan)/100

PID auto-tuning in UNICOS

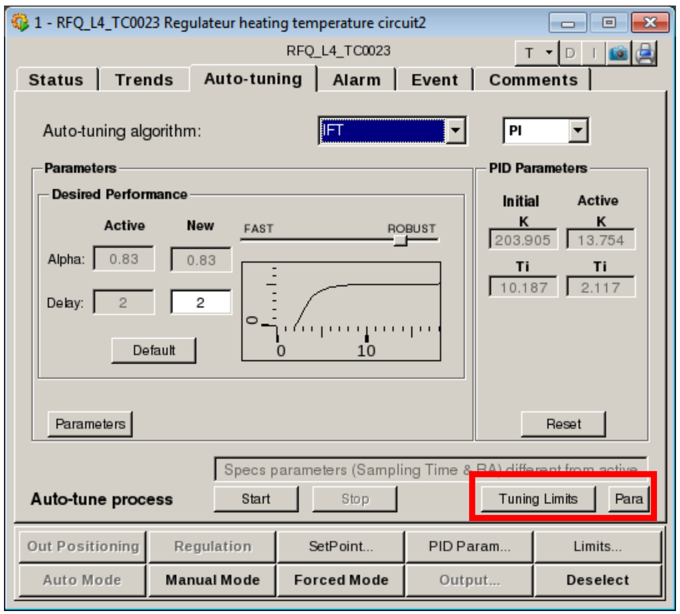
- Automatic tool to excite the process and find PID parameters online
  - Define limits to be respected on process value and on actuator
  - Select “PI” or “PID”
  - Auto-Tune

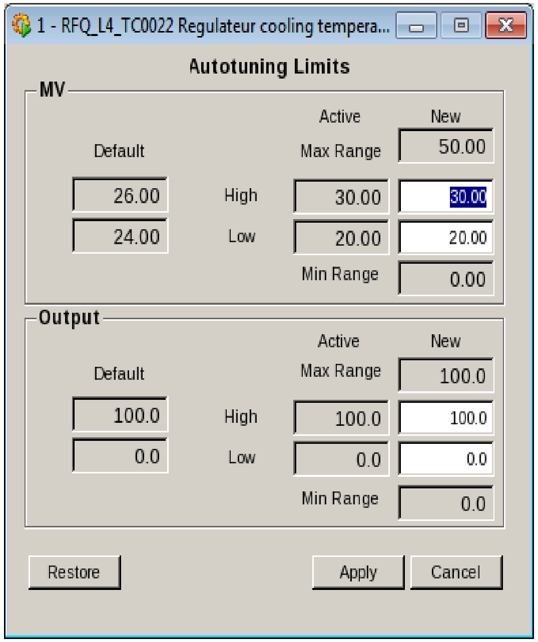
- 3 methods available
  - Relay method: Open-loop auto-tuning.
  - SIMC method: Open-loop manual tuning (identification + tuning),
  - IFT method: Close-loop auto-tuning (need an existing stable tuning).

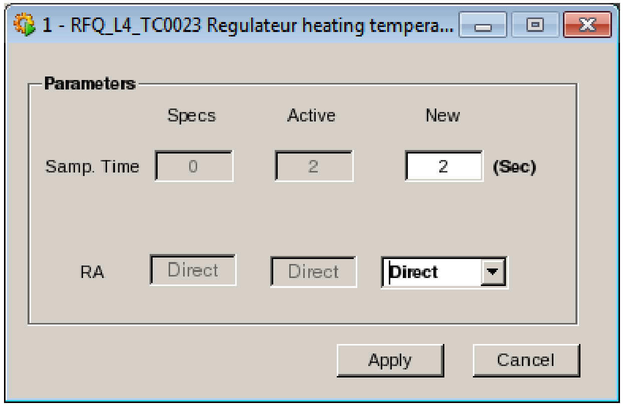
✓ *Auto-tuning stable version is available from WinCC OA package unCPC 6.3.6*

PID auto-tuning: General Parameters

- General Auto-tuning parameters:
  - Tuning Limits, MV and Output maximum and minimum values.
  - Sampling Time and Direct/Inverse Action.

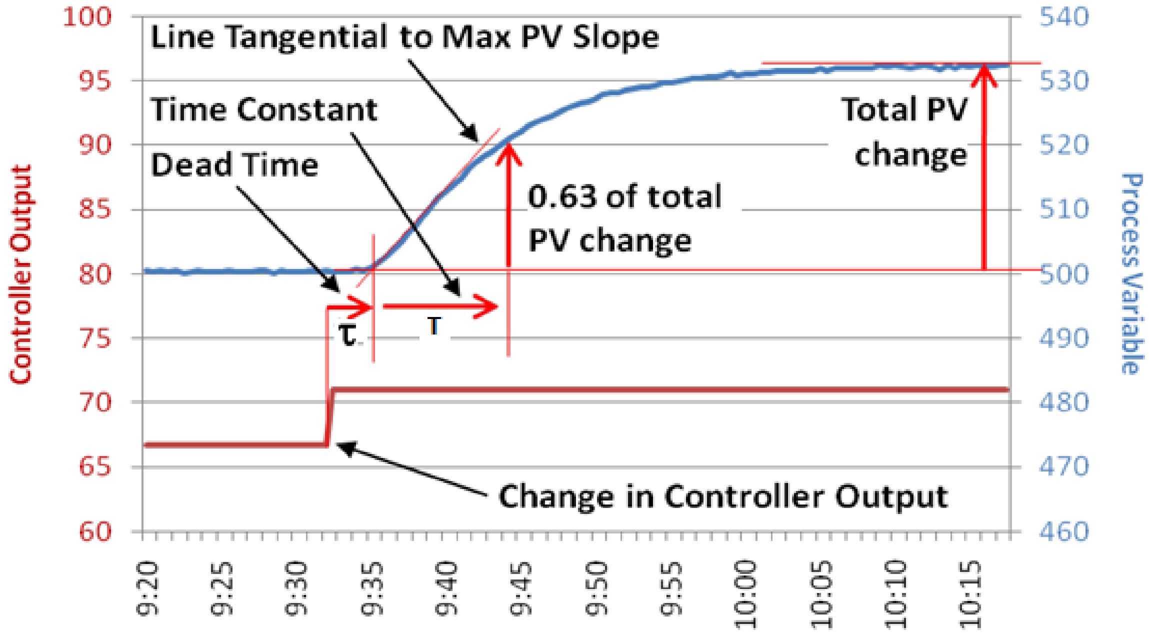






PID auto-tuning: General Parameters

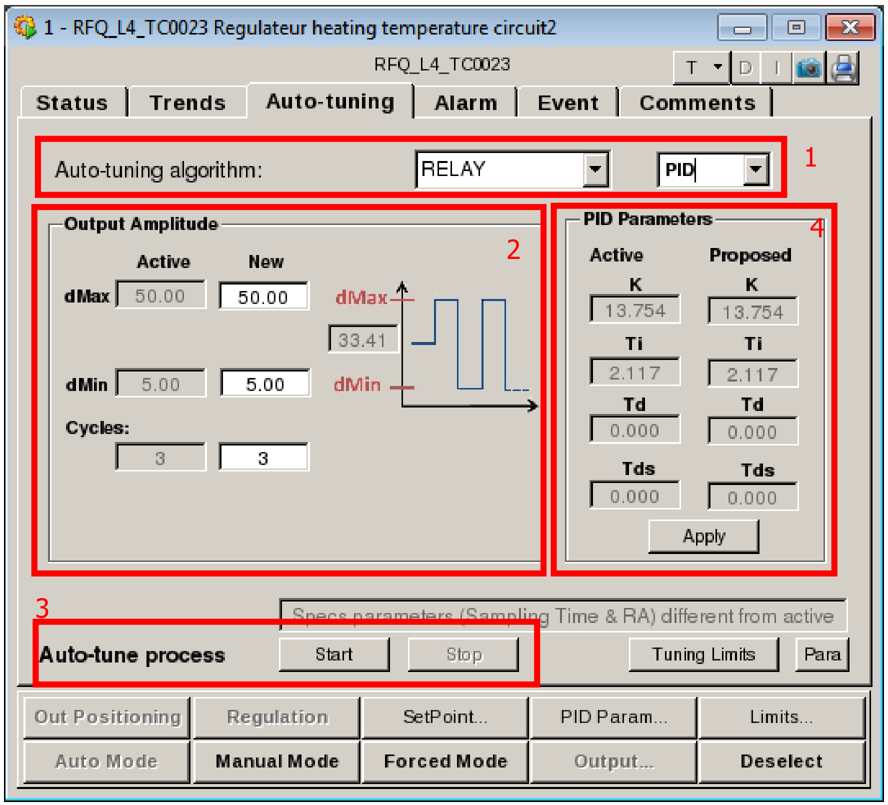
- Sampling Time



- Best practice: $T_{Sampling} \sim \mathrm{T}/10$
- Example: time constant of 5 min needs a sampling time around 30sec
- Must not be too small or too high !

Relay method

RELAY: Open loop auto tuning



1. Open PID faceplate and select "Auto-tuning" tab. Then the user has to choose the "RELAY" method and select the architecture of the controller to tune (PI-PID).

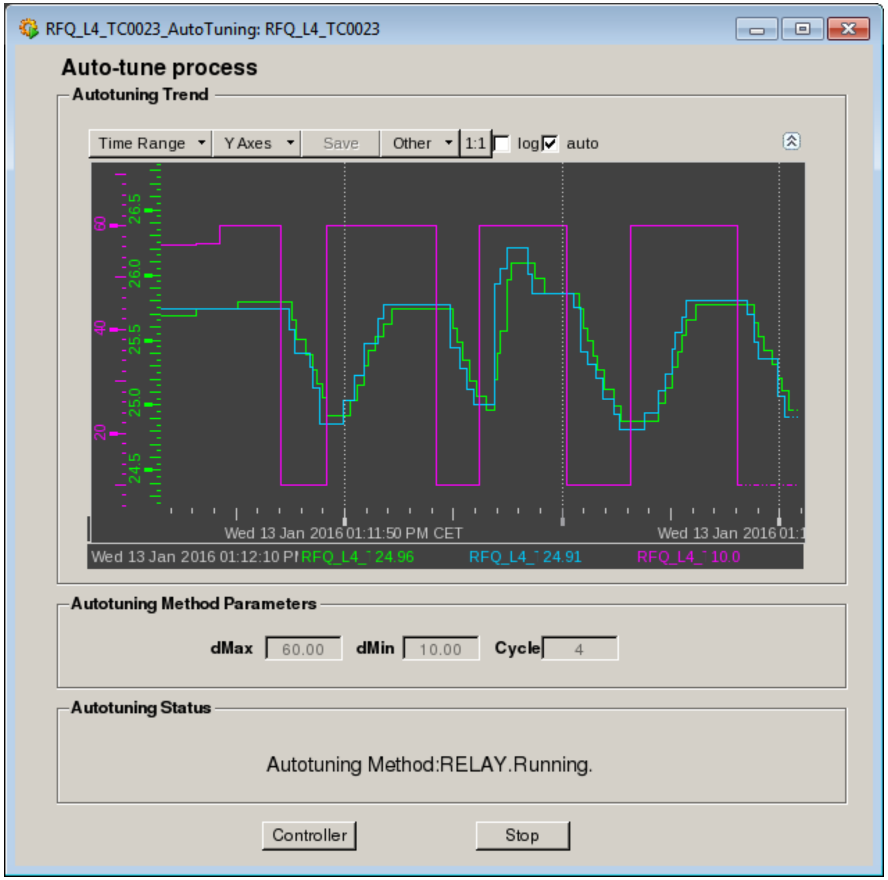
2. Setup Relay parameters
   - dMax: maximum value MV.
   - dMin: minimum value MV.
   - Cycles: how many iterations will execute for the tuning.

3. Start: Launch the method

4. Apply the new PID parameters

Relay method

RELAY: Open loop auto tuning

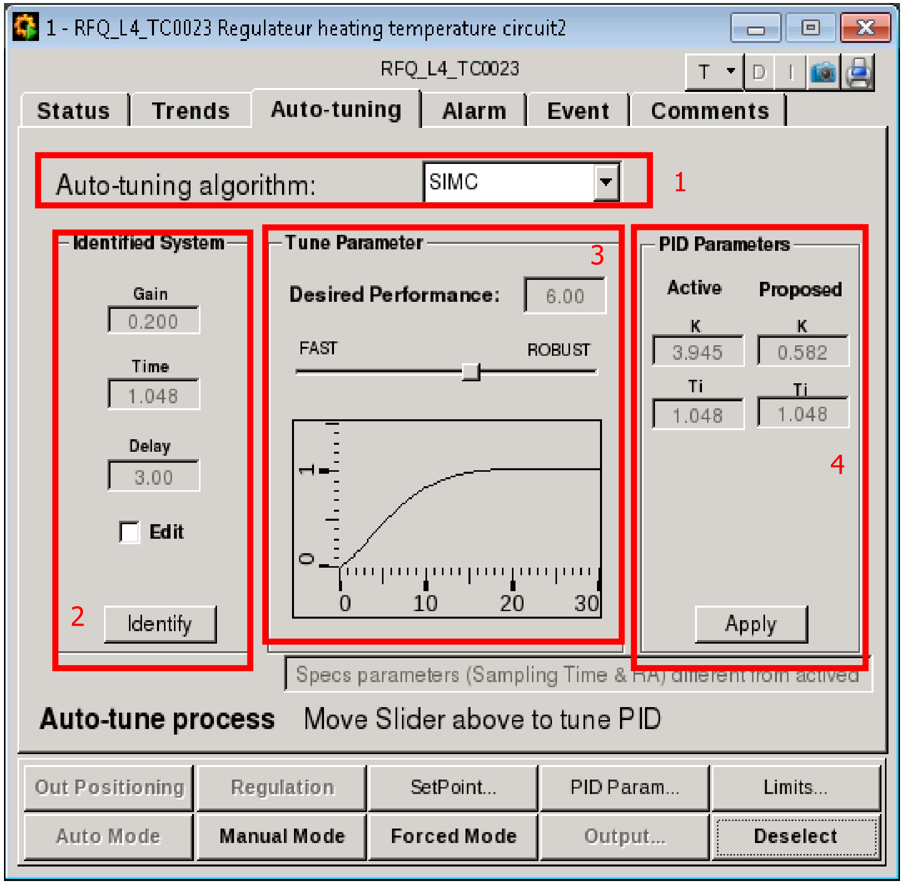


Pop-up panel for the auto-tuning process:

- Allow to follow the evolution of the tuning
- Must stay open
- You can continue to work on HMI

SIMC method

SIMC: Open loop manual tuning



1. Open PID faceplate and select "Auto-tuning" tab. Then the user has to choose the "SIMC" method.

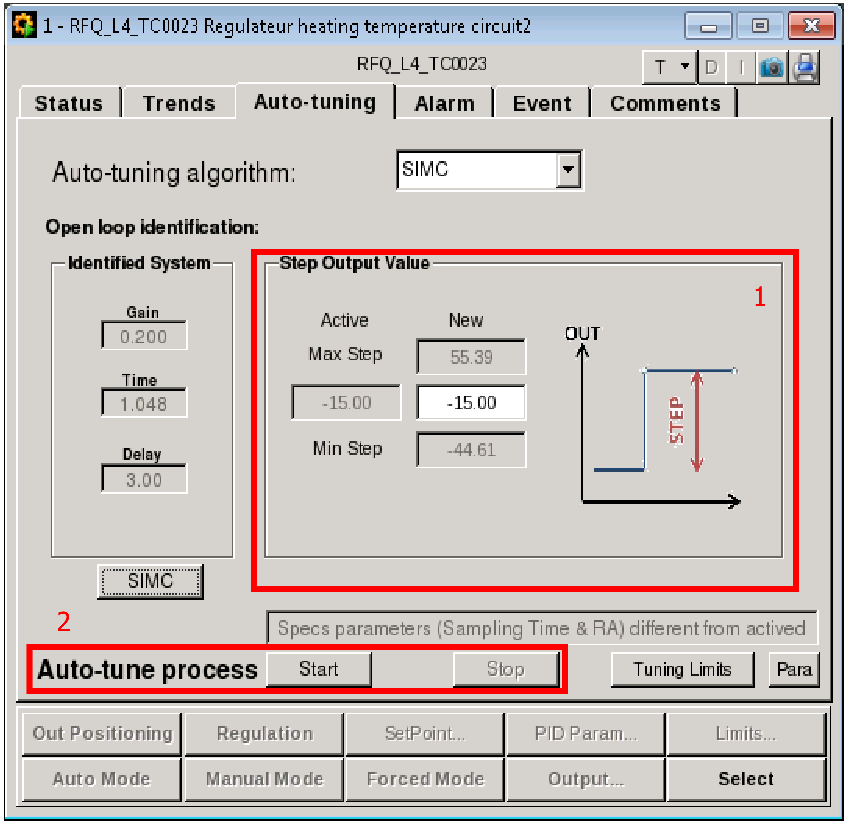
2. Identify a process model (see next slide)
   - First order transfer system
   - It can ben edited by the user.

3. Once the model has been obtained, using the slider the user chooses the desired performance.

4. Apply the new PID parameters

SIMC method: Identification

SIMC: Open loop manual tuning

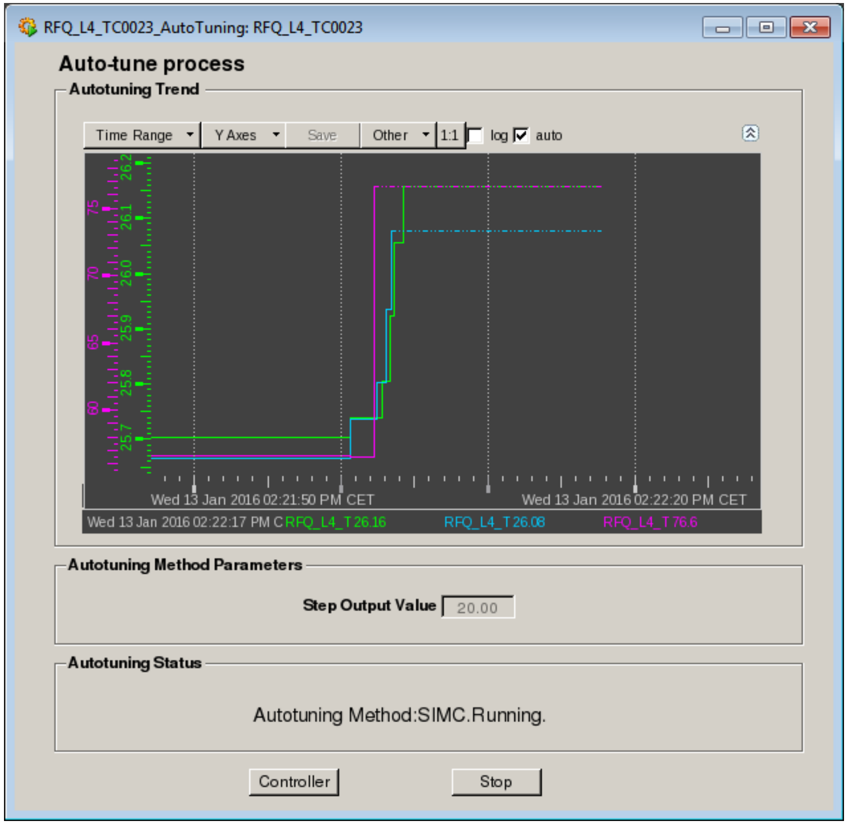


- Identify a process model
  1. The user has to define the desired step-change in the MV. The panel shows user the limits of the MV.
  2. Pushing "Start" button the identification is launched and a pop-panel, which summarizes the main information about the process.

  ➢ Only valid for stable 1st order transfer system.

SIMC method: Identification

SIMC: Open loop manual tuning

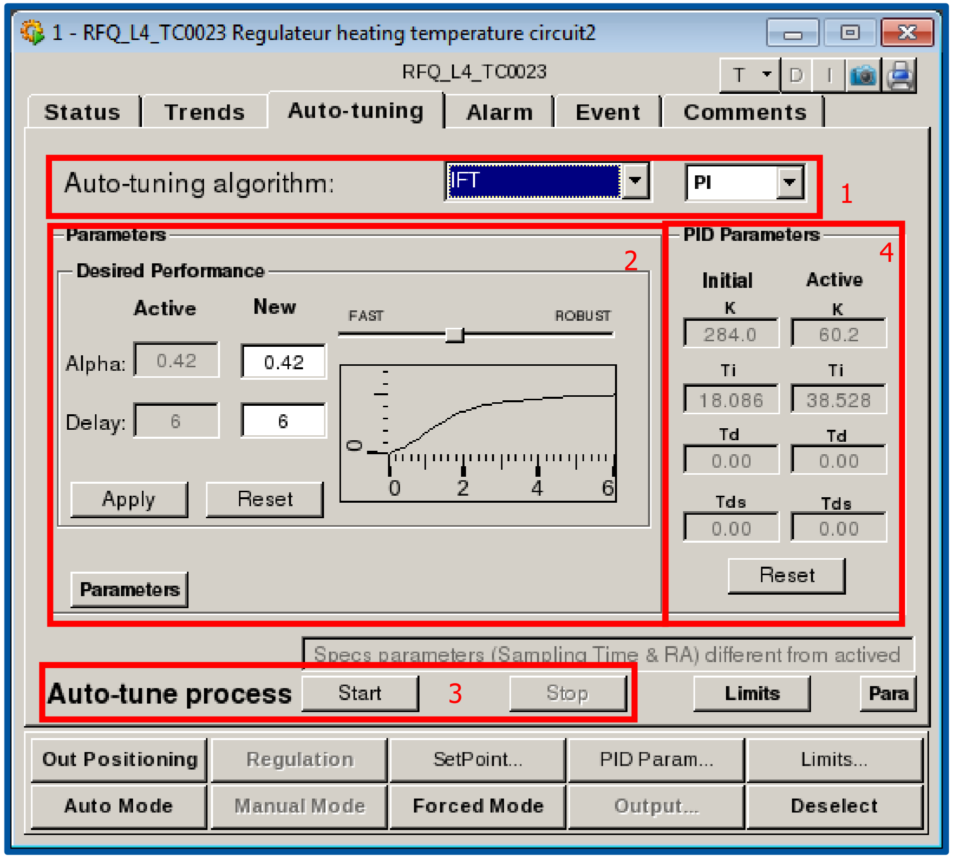


Pop-up panel for the auto-tuning process:

- Allow to follow the evolution of the identification.
- Must stay open
- You can continue to work on HMI

IFT method

IFT: Close loop auto tuning

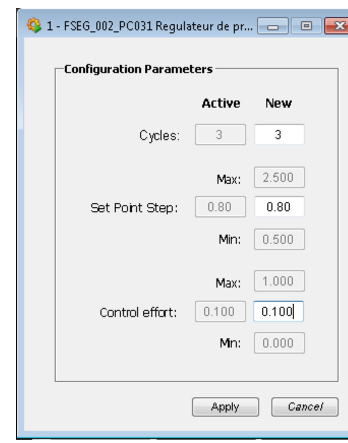


➢ *Cannot work if Set-point is calculated in PLC*

1. Open PID faceplate and select “Auto-tuning” tab. Then the user has to choose the “IFT” method and select the architecture of the controller to tune.

2. Select the set-point is external object

3. Setup IFT parameters
   - ➢ Alpha: desired performance (slide bar).
   - ➢ Delay: Process delay (minimum 1).
   - ➢ Parameters:
     - ➢ Cycle number
     - ✓ Set-point step
     - ✓ Control effort

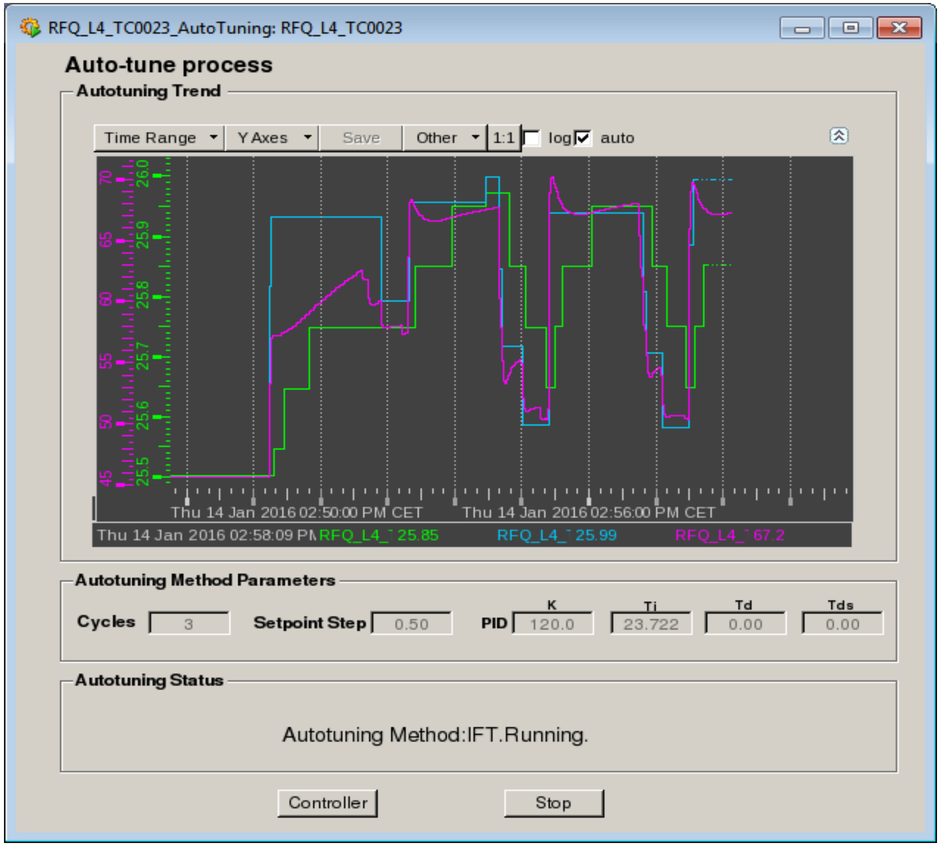


4. Start: Launch the method

5. Apply the new PID parameters

IFT method

IFT: Close loop auto tuning

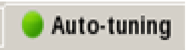


Pop-up panel for the auto-tuning process:

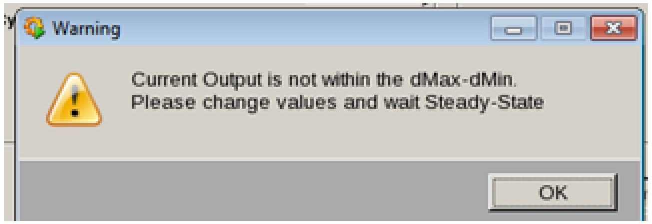
- Allow to follow the evolution of the tuning
- Must stay open
- You can continue to work on HMI

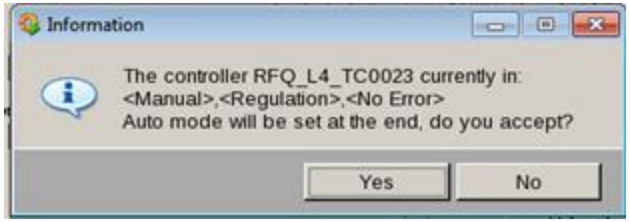
Further features

- Controller widget is blinking during auto-tuning
- Controller faceplate auto-tuning tab is Green during auto-tuning



- When press “Start”, several checks are performed
  - Relay: Maximum output > current output
  - Relay: Minimum output < current output
  - All: not allow auto-tuning if the controller is in “Tracking” or “Local”
  - All: Controller status and mode are checked and summarized in a pop-up message before the to launch the auto-tuning.
- At the end of the auto-tuning
  - Controller is set in auto mode
  - Original Set-point is restored





How to choose a method ?

- First tuning
  - **S-IMC**
    - ✓ Stable process only
    - ✓ Desired trajectory to be specified by user
    - ✓ One single step: fast
  - **Relay**
    - ✓ Stable/Unstable processes
    - ✓ No trajectory selection
    - ✓ Need around 4 cycles: can take time for a slow process

- Fine tuning
  - **IFT**
    - ✓ Need an existing tuning not too bad
    - ✓ Desired trajectory to be specified by user
    - ✓ Close loop tuning: better safety
    - ✓ Need around 4 cycles: can take time for a slow process
    - ✓ *Note: cannot be used if the set-point is calculated in PLC*

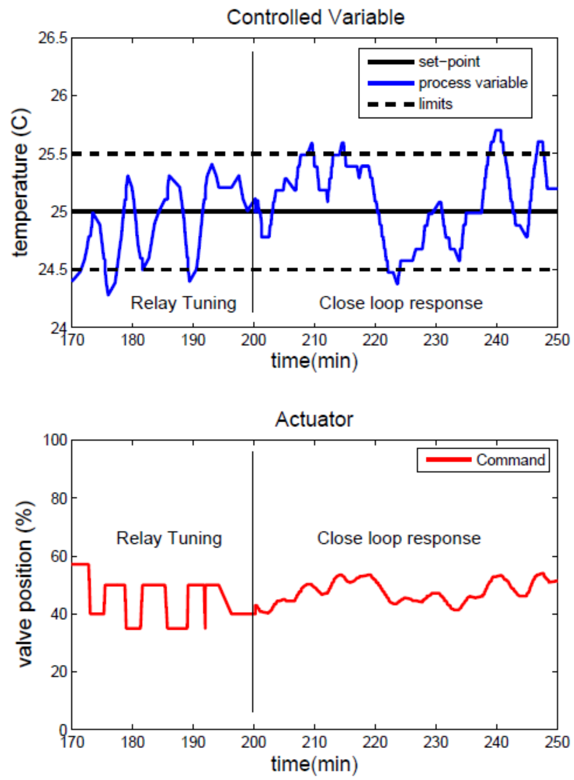
PID auto-tuning: real example

LHC chilled water production: condenser output temperature regulation

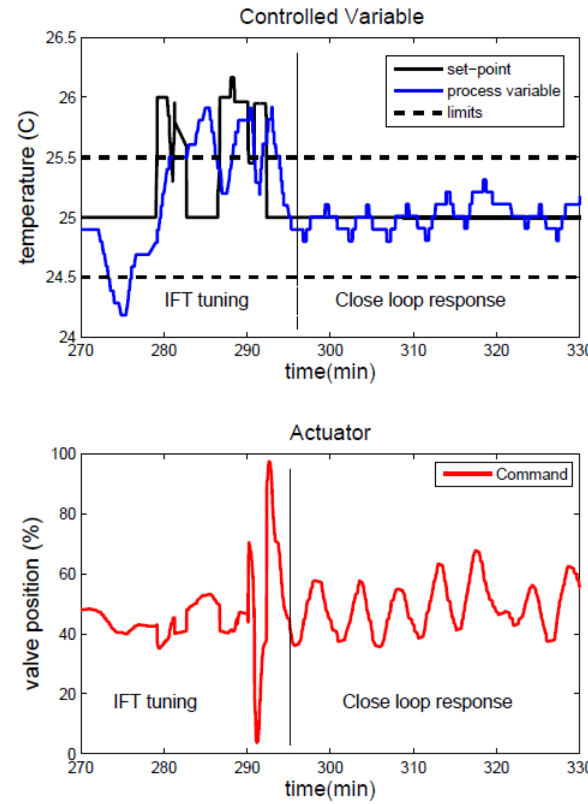
Initial: Impossible to find good tuning with try/error method.
Temperature oscillation = 2 C
Valve oscillation = 100 %

Tuning	Kc	T_i	$\lvert\Delta y\rvert$	$\lvert\Delta u\rvert$
Original	8	0.5	2 °C	100 %
Relay	8.3	311	1.3 °C	10 %
IFT	50.9	5666	0.5 °C	20 %

Relay Method as first tuning



IFT Method as fine tuning



➔ Here, PID shows some limitations due to large delay and many disturbances

Conclusion

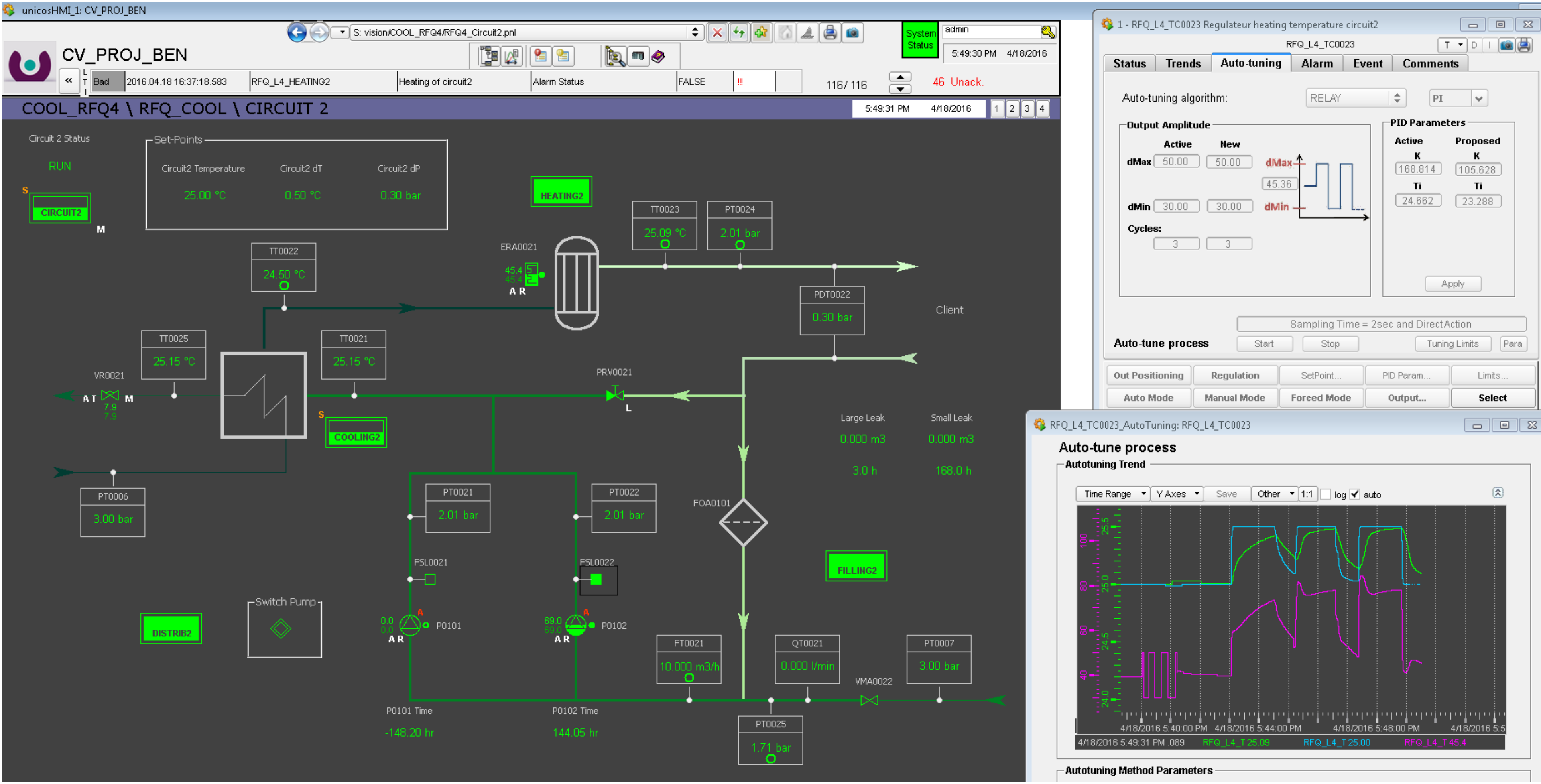
- PID tuning is easy to setup if :
  - The process can be excited enough.
  - The process is not too non-linear and not too disturbed.
  - Actuator dynamics and ranges are large enough.

- It is impossible to tune a regulation loop without exciting the process.
  - You need to move your actuator significantly to evaluate its effects in term of amplitude, time and frequency.
  - It is useless to try to change the PID parameters in closed loop when the loop is regulating at a constant set-point.

- It is impossible to compare controller gains between them if they are not normalized (no scaling in the regulation loop).

- The try and error method can work only if you are already experienced on identical regulation loops having the same kind of gains, time constants and delays.

PID auto-tuning: demo on RFQ4 cooling plant



References

PID tuning :

[1] K.J. Astrom and T. Hagglund. *PID Controllers : Theory, Design and Tuning.* Instrument Society of America,Research Triangle Park, North Carolina, 2nd Edition, 1995.

[2] I.L. Chien. Imc-pid controller design - an extension. In *IFAC Adaptive Control of Chemical Processes Conference*, pages 147–152, Copenhagen, Denmark, 1988.

[3] I.L. Chien, J.A. Hrones, and J.B. Reswick. An the automatic control of generalised passive systems. *Transactions of the ASME*, February :175–185, 1952.

[4] A. O Dwyer. Pi and pid controller tuning rules for time delay processes : a summary. *Technical Report AOD-00-01, Edition 1*, 2000.

[5] J.G. Ziegler and N.B. Nichols. Optimum settings for automatic controllers. *Transactions of the ASME*, November :759–768, 1942.

UNICOS PID auto-tuning:

- R. Martí, B. Bradu, E. Blanco Vinuela, L. Frutos, R. Mazaeda, C. Prada. *PID_TUNE: A PID autotuning software tool on UNICOS CPC*. In Proceedings of the 15th International Conference on Accelerator and Large Experimental Physics Control Systems, Melbourne, Australia, 2015.

- Instruction manual about UNICOS auto-tuning toolbox: **EDMS 1611581**

CERN