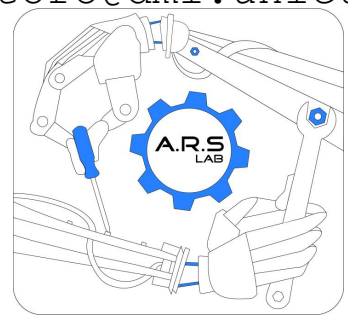
Cascading Controllers

Corrado Santoro

ARSLAB - Autonomous and Robotic Systems Laboratory
Dipartimento di Matematica e Informatica - Università di Catania, Italy
santoro@dmi.unict.it

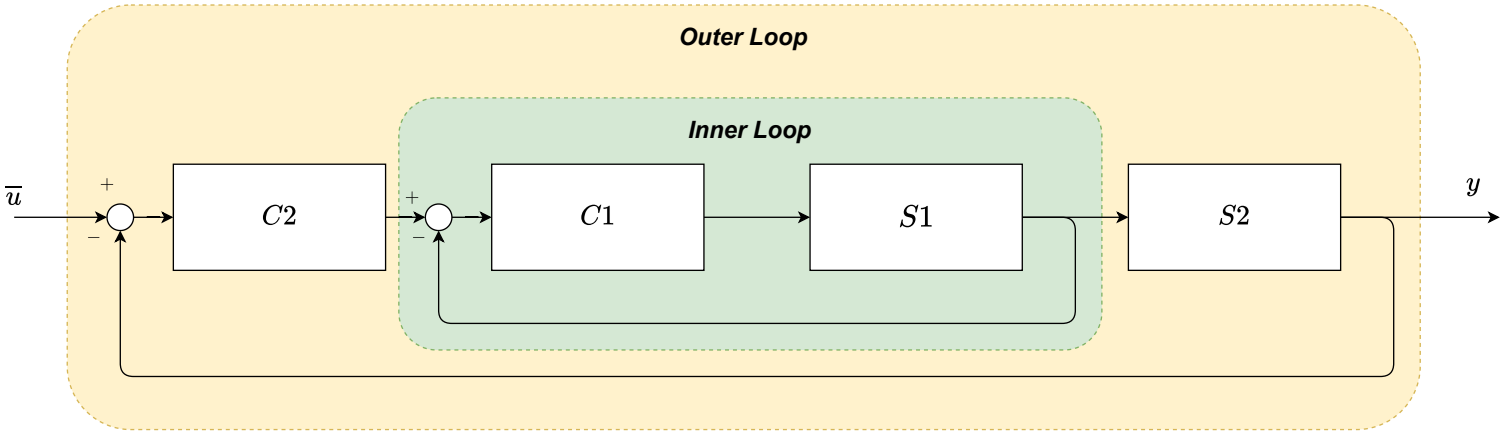


Robotic Systems

Systems and Control Loops

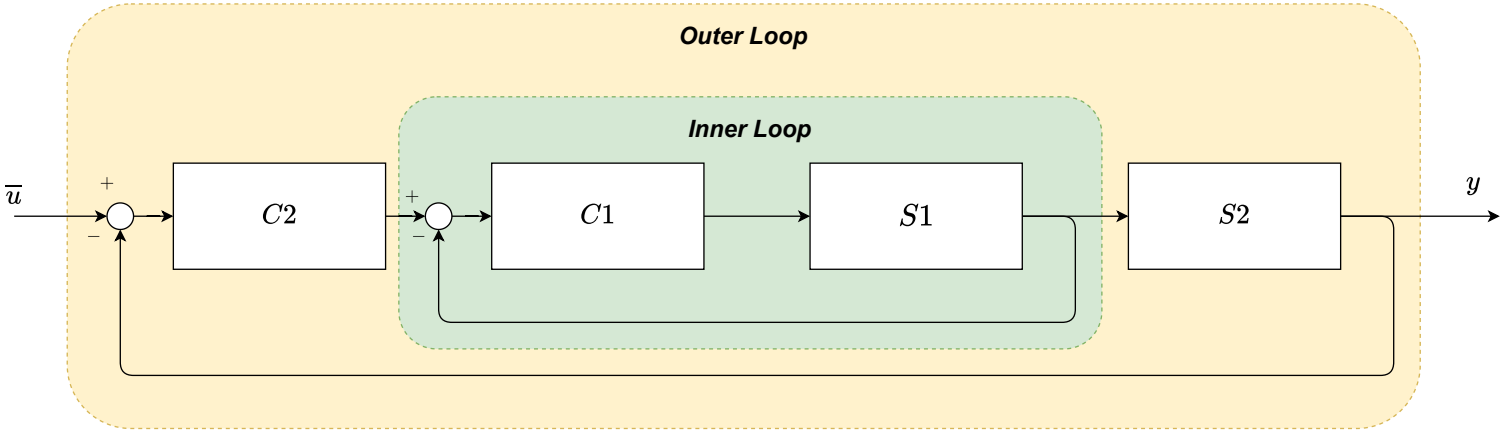
1. A complete robotic system is made of many “control loops”, about one for each variable to be controlled
2. These control loops may interact with each other or be independent
3. In some cases, control loops may be “nested”, i.e. a control loop drives another control loop
4. Therefore, the overall schema can be also quite complex, but using some hints, it can be easily understood and implemented

Nested Controllers or Cascading Controllers



- In many cases, a robotic system is featured by such schemas
- Here, we have system $S1$ that is controlled by $C1$ ...
- ... the output then passes through $S2$
- The reference for $C1$ is however not given as an external input, but is the output of $C2$ that **controls** the variable output of the overall system

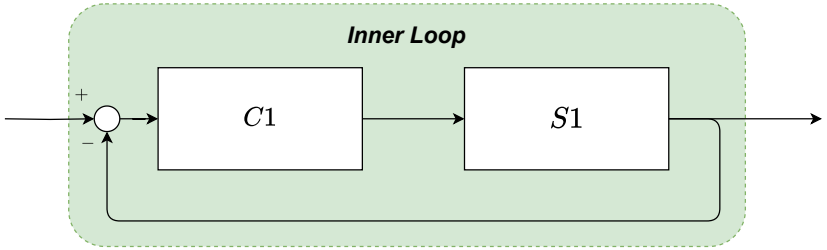
Nested Controllers or Cascading Controllers



Implementation

- Given that we can implement the overall system in a single function or method, a better way is to use a bottom-up approach
- We first deal with the **inner loop** (and tune the parameters of $C1$)
- Then we replace, in the schema, the inner loop with a “black box” (implementing the inner loop) and deal with the **outer loop**

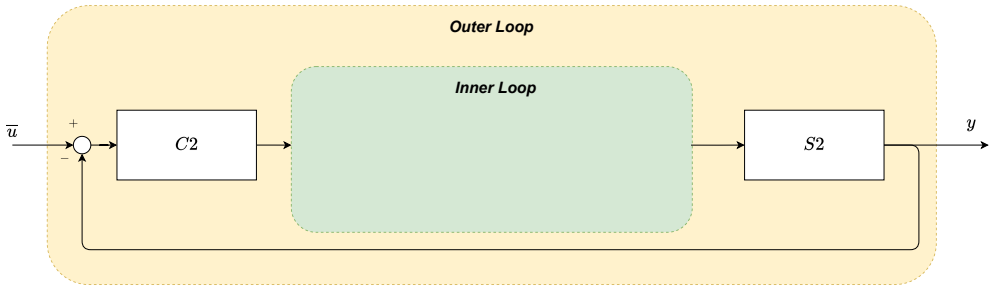
Nested Controllers or Cascading Controllers



The Inner Loop

```
class S1:
    ....

class C1:
    ....

class InnerLoop:

    def __init__(self):
        self.s1 = S1()
        self.c1 = C1()
        self.y = 0

    def evaluate(self, delta_t, _input):
        error = _input - self.y
        out_c1 = self.c1.evaluate(delta_t, error)
        self.y = self.s1.evaluate(delta_t, out_c1)
        return self.y
```

Nested Controllers or Cascading Controllers

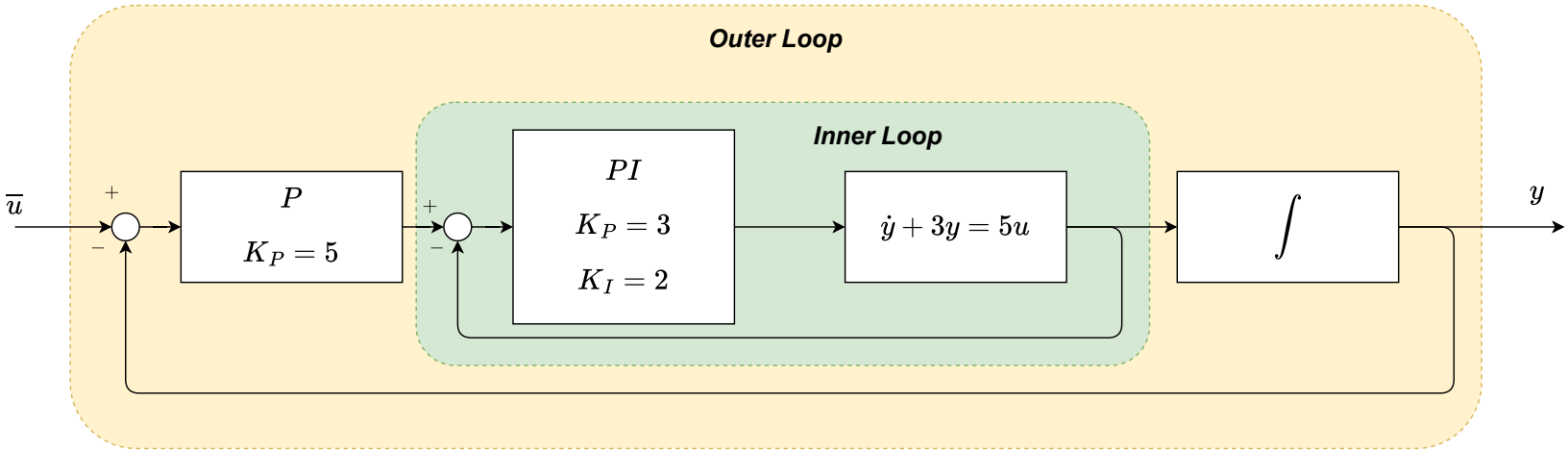


The Outer Loop

```
class S2:
    ....

class C2:
    ....

class OuterLoop:

    def __init__(self):
        self.s2 = S2()
        self.c2 = C2()
        self.innter = InnerLoop()
        self.y = 0

    def evaluate(self, delta_t, _input):
        error = _input - self.y
        out_c2 = self.c2.evaluate(delta_t, error)
        out_inner = self.inner.evaluate(delta_t, out_c2)
        self.y = self.s2.evaluate(delta_t, out_inner)
        return self.y
```

Nested Loops Example

Let's consider:



Implementation

We have:

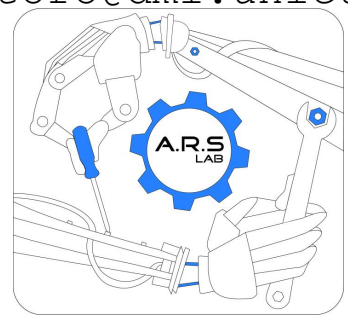
- A first order system
- An integrator
- A *PI* controller for the inner loop
- A *P* controller for the outer loop

(see examples/systems/double_control_loop.ipynb)

Cascading Controllers

Corrado Santoro

ARSLAB - Autonomous and Robotic Systems Laboratory
Dipartimento di Matematica e Informatica - Università di Catania, Italy
santoro@dmi.unict.it



Robotic Systems