Auterion

Controllers auto-tuning for multirotors and fixed-wing vehicles

Mathieu Bresciani
Auterion

September 2021

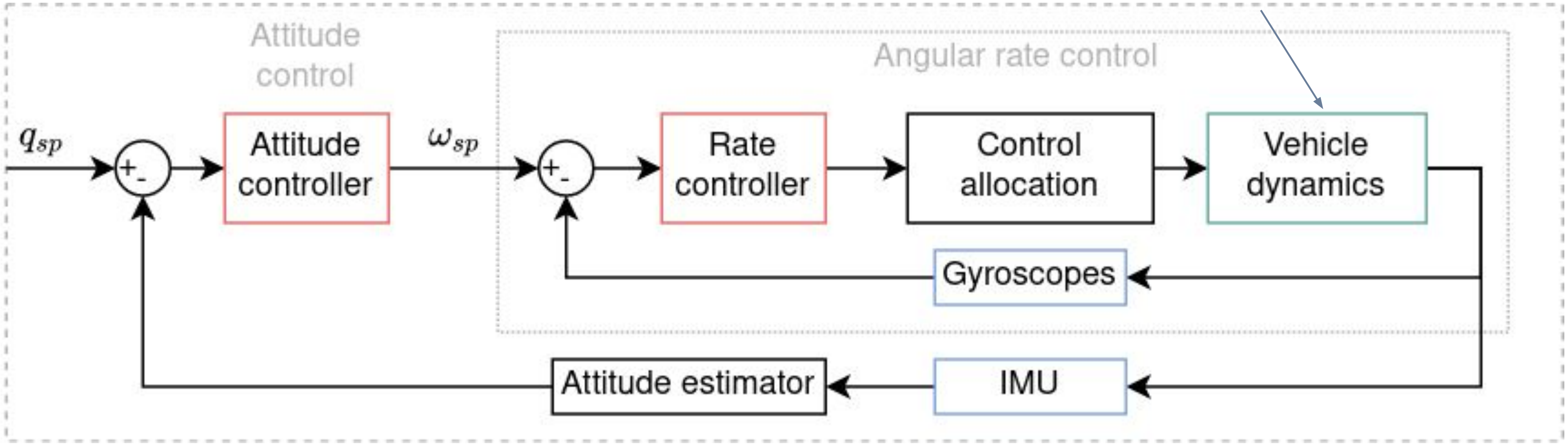
Outline

1. **(Basic) Control theory**
    a. Block diagrams
    b. Controller tuning techniques
2. **Why do we need auto-tuning?**
3. **Auto-tuning and adaptive control**
    a. Direct and indirect adaptive control
    b. Adaptive control → auto-tuning
4. **The strategy selected for PX4**
5. **Python prototyping**
6. **Onboard implementation (in PX4)**
7. **Flight test results**
8. **Conclusion**

1.a. Control theory - Block diagrams

Example: Cascaded attitude/rate controller

Set of (non)linear differential equations
"Converting" force → motion

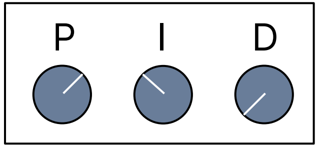


1.b. Control theory - Controller tuning

No model required:

- manual PID tuning
- Ziegler-Nichols → too dangerous for aircrafts
- Auto-tuning (~adaptive control)

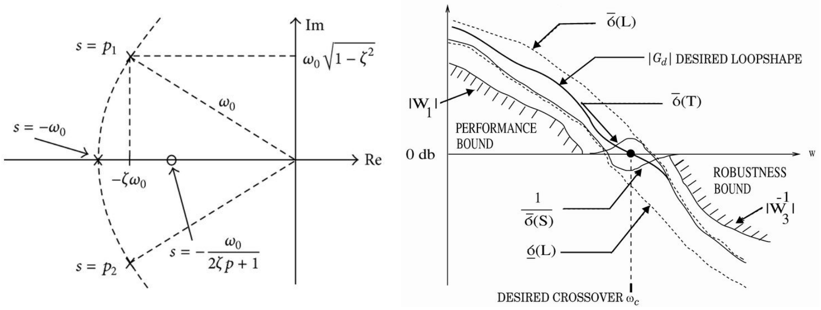
Widely used in industrial projects



Model required:

- Loop-shaping
- H-infinity
- Pole placement
  - Internal model control
  - Minimum variance control
  - Model reference control
- ...

Mainly used in academia or in research projects



Left: pole placement, Right: loop shaping (Mathworks.com)

2. Why do we need auto-tuning?

Manual tuning

- Need good pilot (and a 2nd operator)
- Understand the control parameters
- Tuning experience
- A lot of flight time (iterations)
- Everyone has a different standard
- Limited to simple controllers (e.g.: PID)
- ...

Total: expensive for every bringup

Auto-tuning

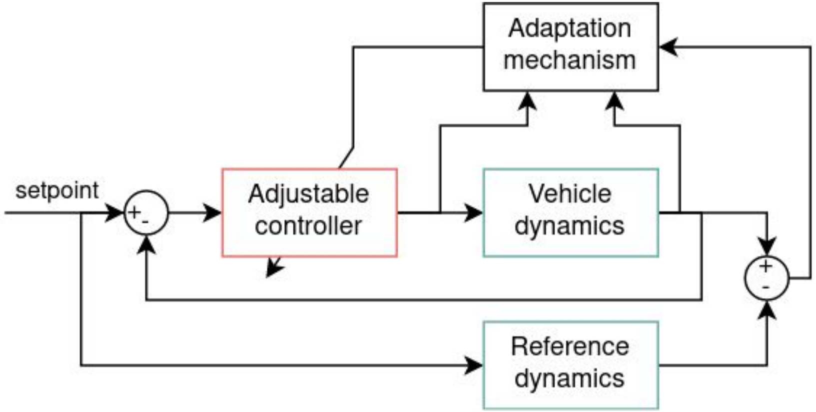
- Can be used by everyone
- High-level understandable criteria
- No tuning experience required
- Fast
- Generates a good base tuning on all drones
- Can be used to tune simple and advanced controllers

Total: expensive to develop but then almost free

3. Auto-tuning and adaptive control

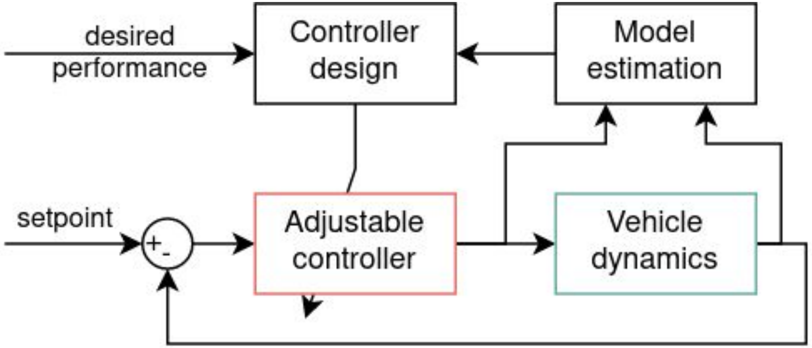
Direct adaptive control

Adjust the controller to track the desired reference dynamics (e.g.: MRAC)



Indirect adaptive control

1. Estimate the vehicle dynamic model
2. Adjust the controller using model-based tuning techniques (e.g.: loop shaping, pole placement, ...)

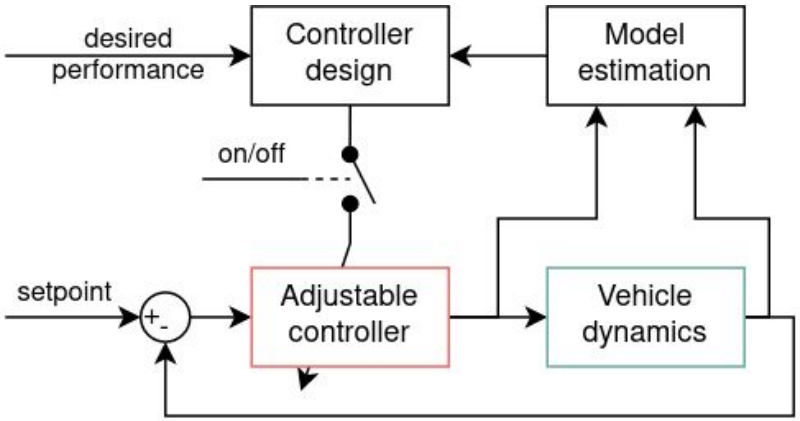


Ref: A.Karimi, Robust and Adaptive control, epfl

3. Auto-tuning and adaptive control

Adaptive control: Continuously adjust the controller as the parameters change

Auto-tuning: Temporarily enable the control adaptation algorithm to tune the controller



4. The strategy selected for PX4

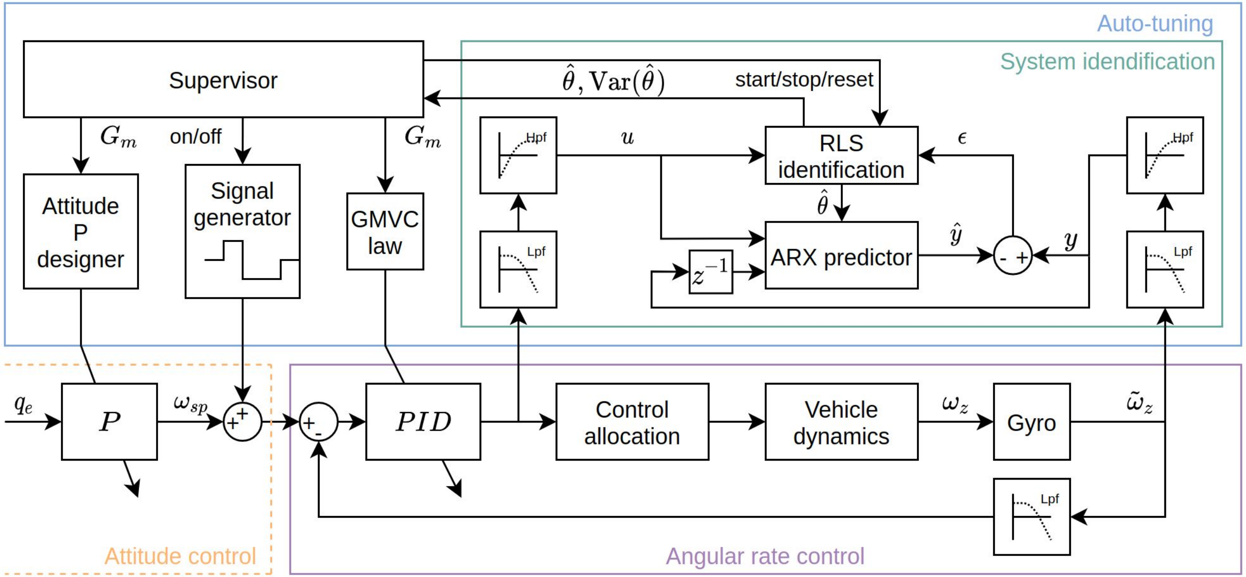
Indirect adaptive control

System identification:

- Efficient Recursive Least-Squares (RLS) algorithm (no matrix inversion)
- Linear optimization
- Fixed structure

Attitude and rate control tuning:

- Closed-form solution



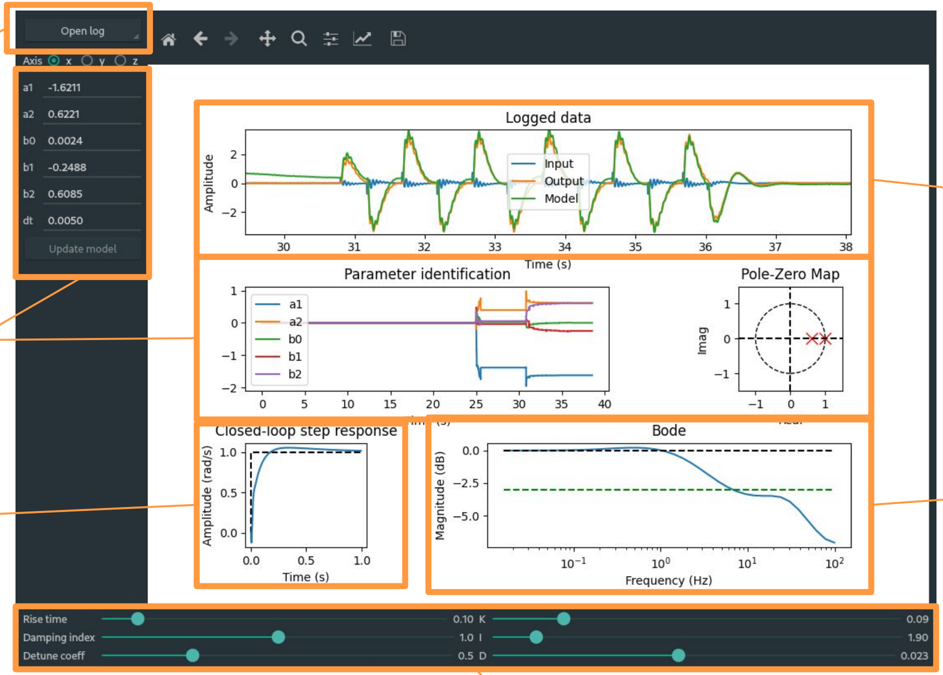
Auterion

5. Python prototyping

Input: high-rate *.ulg* flight logs

Discrete-time model identification

Closed-loop (controller + identified model) step response



Simulates model output based on identified model and logged inputs

Closed-loop bode plot (frequency response)

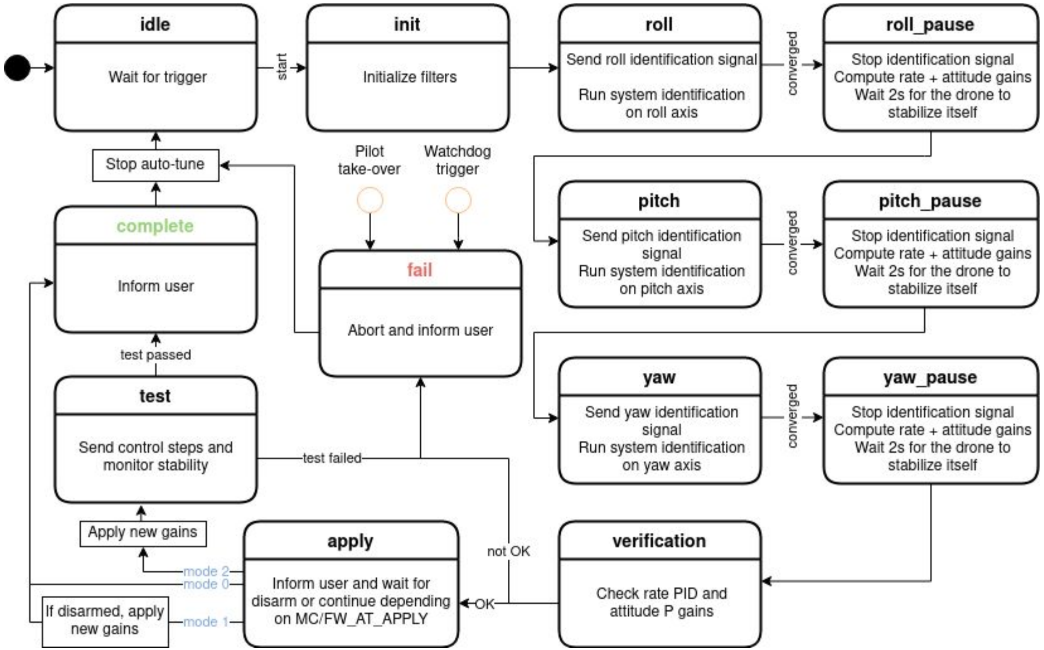
Sliders for auto-tuning meta-parameters and PID gains

Auterion

6. Onboard implementation (in PX4)

Same algorithms used in python prototype script (same results are confirmed by unit tests)

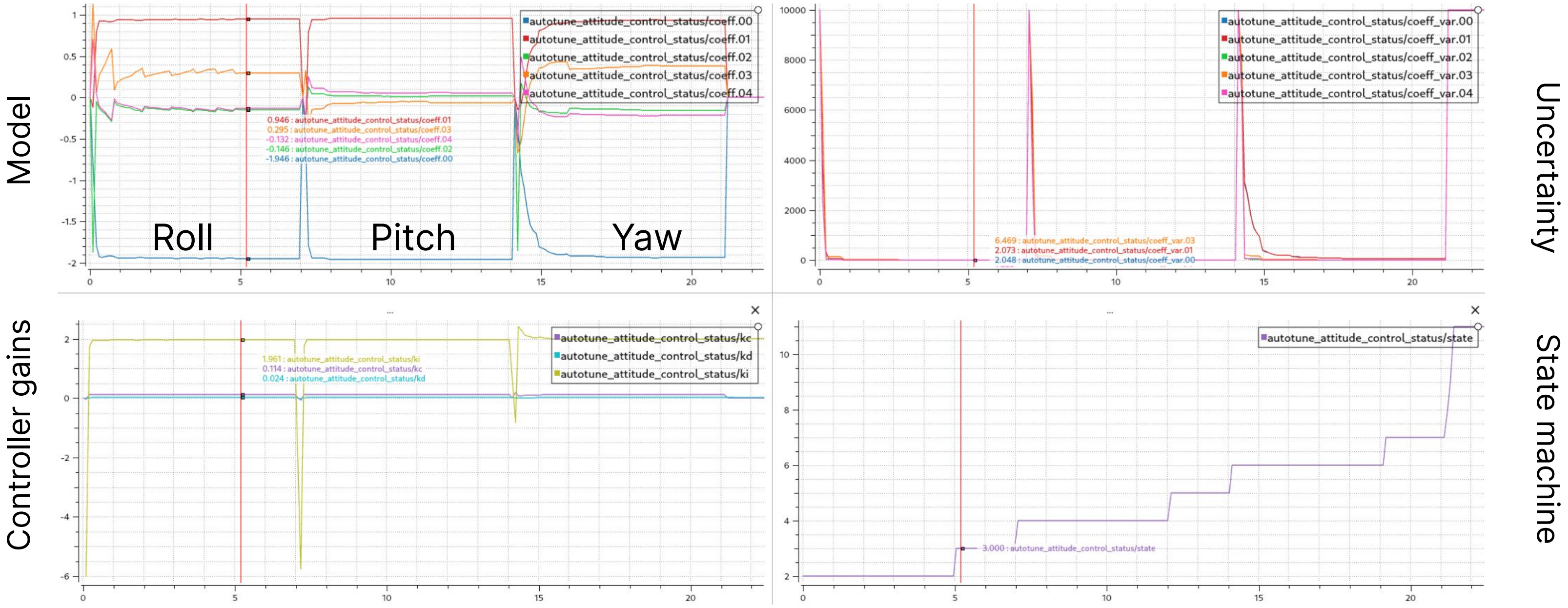
Additional supervisor state machine and error handling logic



Auterion

7. Flight test results

Example: X500 Quadrotor



8. Conclusion

Benefits

Speeds-up initial bringup

Complicated trial and error tuning → single click

No control theory knowledge required

Provides a dynamic model (can be used in sim of offline controller design)

Limitations

Cannot tune all drones with same meta-parameters

The drone needs to fly decently before starting the auto-tune algorithm

Struggles to tune UAVs with flexible body or large actuator delays (unmodeled dynamics)

Does not fix the hardware problems

Auterion

Questions?

Thank you!