A Modular Open-Source Python Framework for Small Satellite Attitude Determination and Control

Patrick McKeen[1] Niclas Scheuer[2] Kerri Cahoy[1]

[1]MIT Aeronautics and Astronautics [2]ETH Zurich

5 minutes from install to working ADCS simulation

Same code in simulation, HIL, and flight

The Problem

New control law = weeks of reimplementation

Existing tools are heavyweight (C++, messages)

Teams pick one approach and never compare

The Solution

Configuration replaces reimplementation

Pure Python — pip install
No build toolchain needed

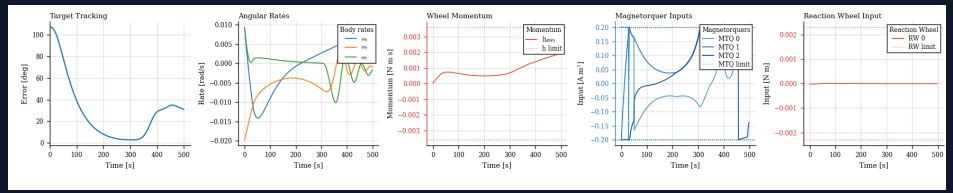
Automatic allocation handles any actuator set

What's In the Box

Component	Capabilities
Dynamics	6-DOF attitude + RW coupling, J2 orbit propagation
Sensors	Magnetometer, gyro, sun sensor, star tracker, EHS, GPS
Actuators	Reaction wheels, thrusters, magnetorquers — inc. bounds
Estimation	E/UKF, SRUKF, augmented-state bias/disturbance est.
Control	PD, Lovera, Wisniewski, B-dot, hybrid MTQ-RW, ALTRO
Allocation	LP, QP, weighted/constrained — bound-respecting
Goals	Nadir, sun, ground track, ECI, reduced-attitude, GoalList
Infrastructure	Satellite factory, Monte Carlo, HIL

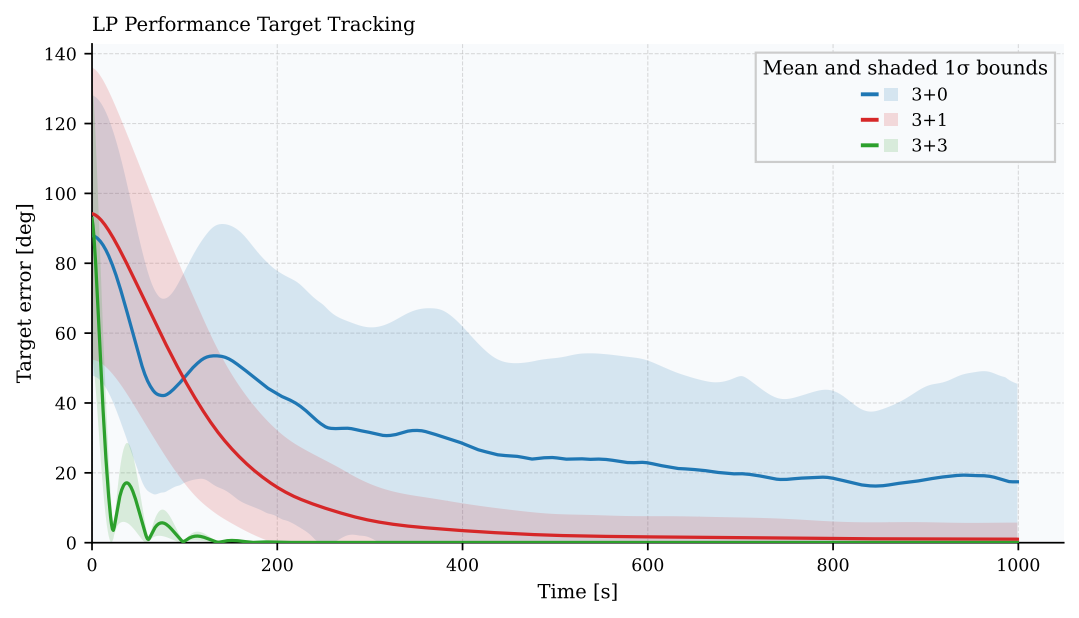
15 Lines → Working ADCS Simulation

```
# Define hardware
acts = [ADCS.MTQ(ax, max_torque=0.2) for ax in np.eye(3)]
acts += [ADCS.RW(axis=[0,0,1], max_torque=0.23e-3)]
sat  = ADCS.Satellite(mass=4, J_0=J, actuators=acts, sensors=sens)

# Configure ADCS
ctrl = ADCS.controller.MTQ_w_RW_LP(est_sat=sat)
goal = ADCS.goals.Coordinate_Goal(lat=42.36, lon=-71.06)

# Run
results = ADCS.simulate(x=x_0, satellite=sat,
    controller=ctrl, goal=goal, os0=os0, dt=1, tf=500)
```



pip install → 15 lines → run

Rapid Architecture Trade

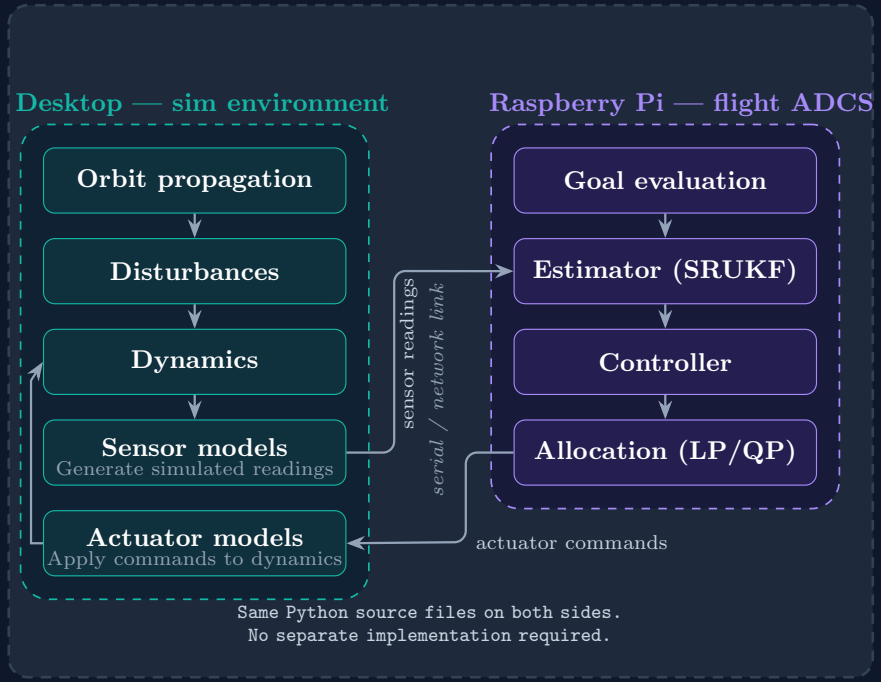


Config	Error	$< 1°$
3MTQ	**17.4°**	19%
3+1RW	**1.0°**	95%
3+3	**0.067°**	100%

3 lines of config change

100-trial Monte Carlo, <1 hour

Hardware-in-the-Loop



Same source files

Identical Python on desktop and RPi

No translation step

No C++ rewrite, no code generation

10 ms control loop

100× real-time margin on RPi 4

Flight heritage

BeaverCube 2 — NASA CSLI mission

Comparison with Basilisk

Identical scenario: 3U, 3+1, nadir, ISS orbit

Metric	This Work	Basilisk
Config lines	**45**	213
Message wiring	**0**	30
Runtime	~38 s	**~0.4 s**
Add estimation	**~20 lines**	**N/A***

**TAM+CSS+gyro estimation requires custom C++ in Basilisk*

Basilisk excels at speed and scale. This framework excels at ADCS-focused rapid iteration.

Get Started

```
pip install -e .
```

github.com/nscheuer/Generalized_ADCS

pmckeen@mit.edu · nscheuer@ethz.ch