PyCubed Mini: An Open-Source, Computational Platform for Education and Research in the Pocketqube Form Factor

Andrew Gatherer and Max Holiday

gatherer@Stanford.edu

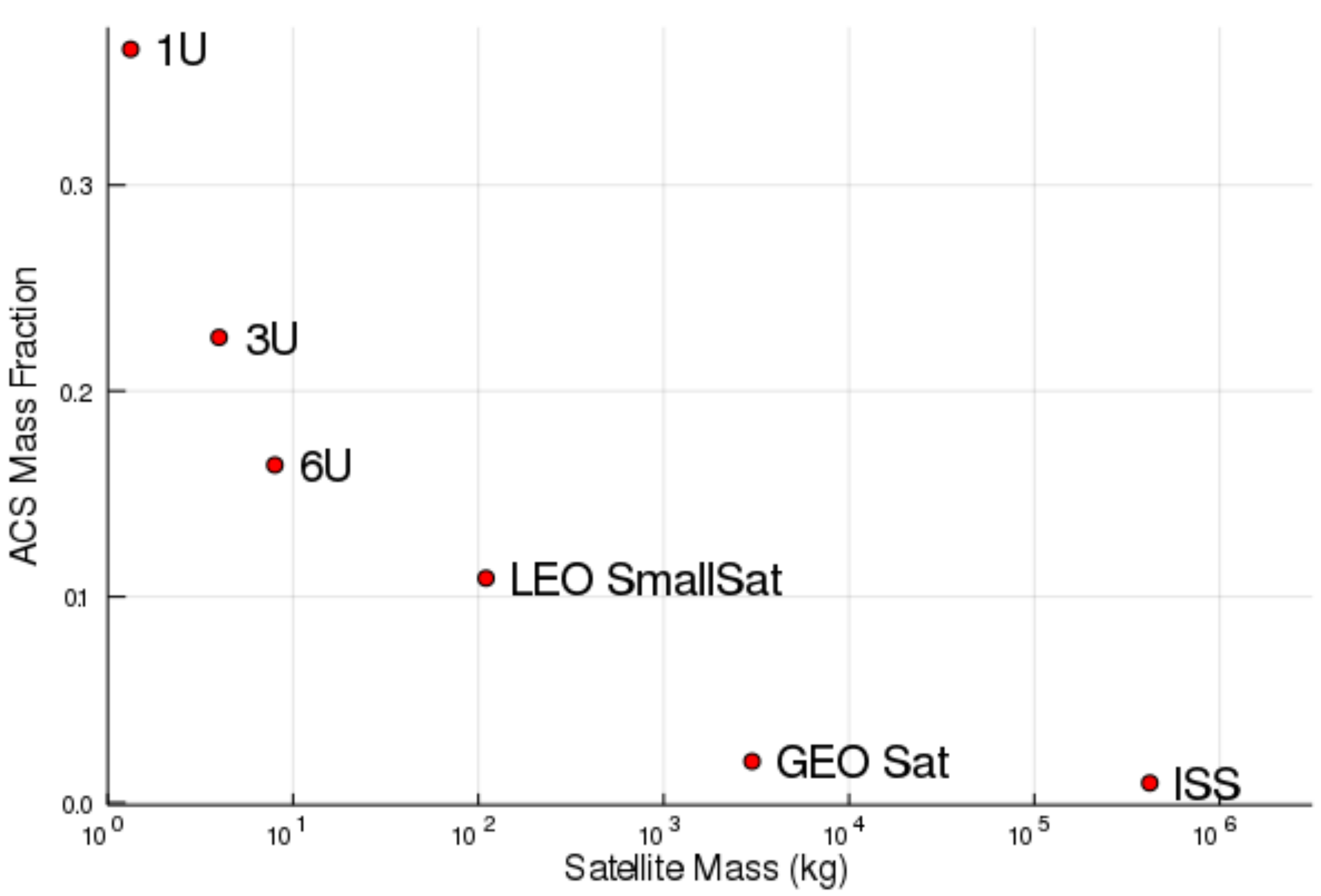
The Big Picture:

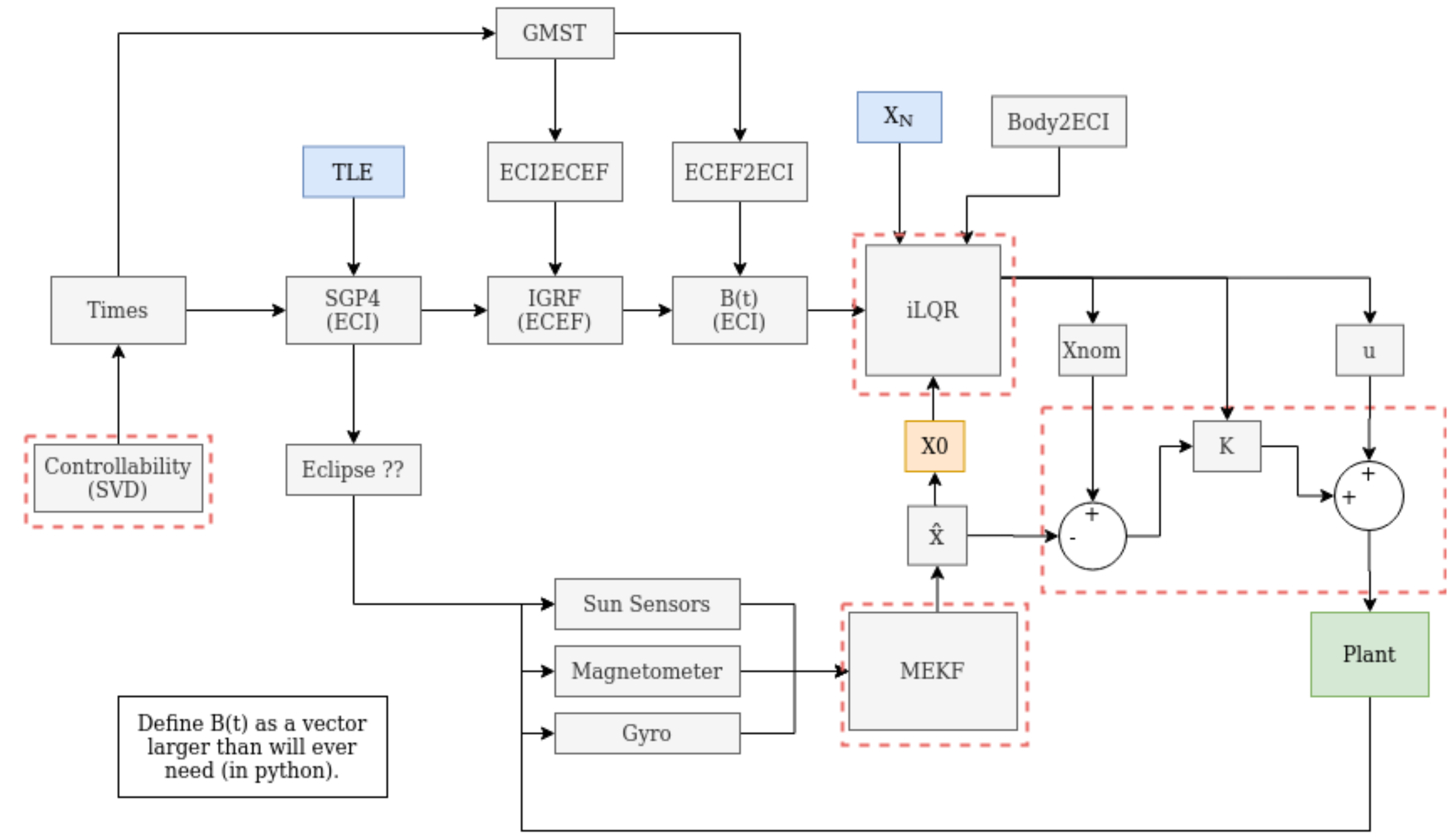
As more CubeSat missions incorporate the hardware and software framework standardized by PyCubed, constraints driving future mission deliverables will become increasingly dependent upon on-board computation. We have developed new embedded tools for PyCubed that facilitate increased computational performance in satellite microcontrollers, provide advanced attitude control capability, and maintain the project's original emphasis on usability and educational approachability. The "PyCubed Mini" project incorporates the refined hardware and software of PyCubed into a PocketQube spacecraft that is scheduled to launch in Q4 2020. These advancements in spacecraft performance are accomplished while developing the mission as part of a year-long Stanford undergraduate capstone course.

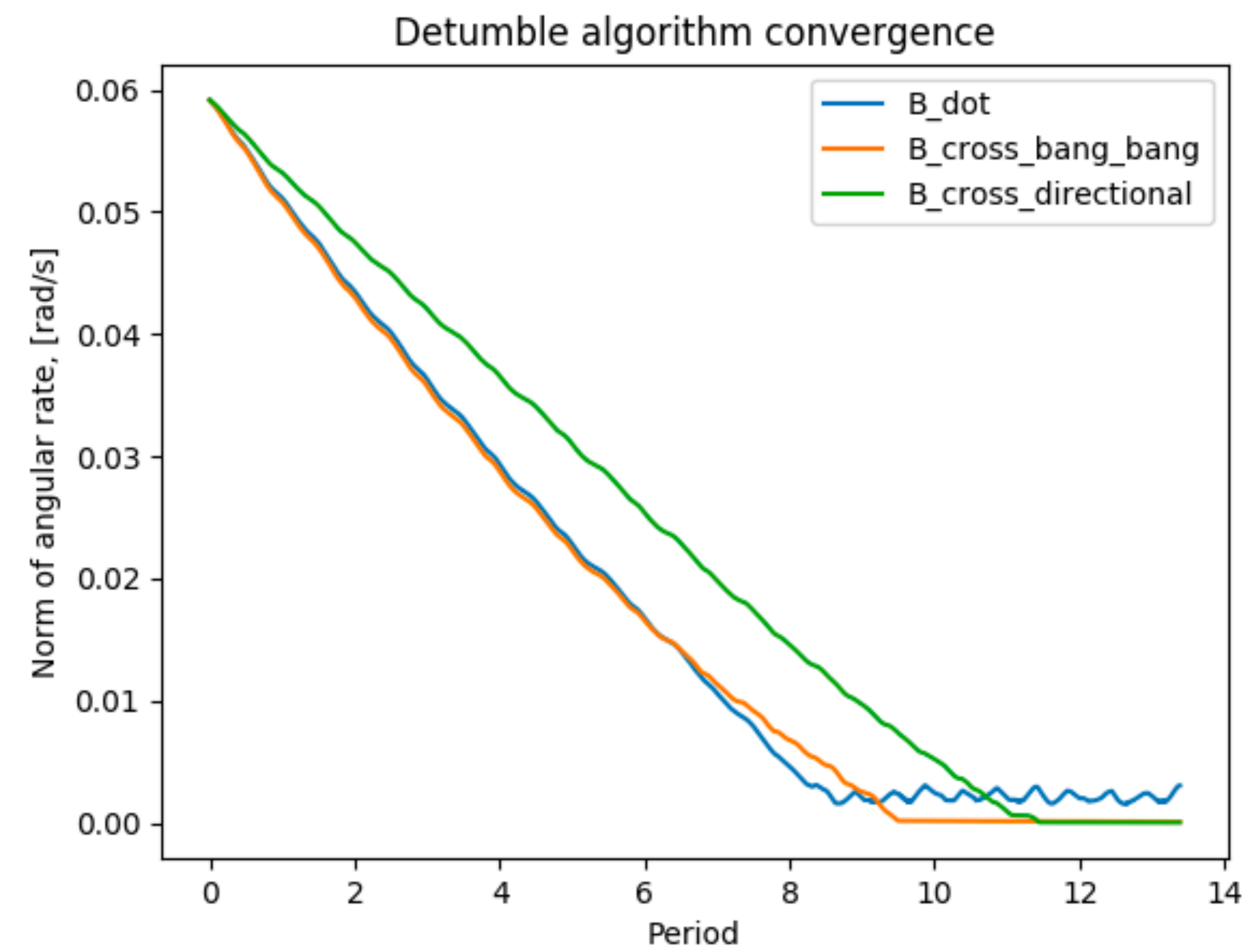
Magnetorquer-only Control

- As satellites shrink, the percentage of volume occupied by traditional attitude control systems increase significantly
- With significant onboard computational power, small satellites now have the ability to perform trajectory optimization techniques
- Additionally, the control authority of magnetorquers get larger as the satellite gets smaller

Mass Cost of Traditional ACS:







Software-in-the-Loop:

- Implemented groundtruth simulator with rigorously tested utility functions and holistic orbital simulator
- Includes linear error sensor model and gyro error model for appropriate randomization
- Perturbation forces and torques generated include: J2, gravity gradient, two-term exponential atmospheric model, and spherical harmonic magnetic field model
- Validated detumble algorithm and MEKF

Hardware-in-the-loop:

- Using groundtruth simulator as foundation and serial port as communication channel
- Rework algorithms into CircuitPython for easy testing

CircuitPython:

- Embedded language with Python syntax generated from CPython foundation
- Magnitudes faster and space-efficient than Arduino or Linux operating systems
- Allows students to implement state machine and flight software without embedded C experience
- Integrated linear algebra and linear solver package
- Flight heritage on KickSat-2 mission

Example CircuitPython:

```
import time, board, busio
import cpc
import adafruit_lsm9ds1 as IMU

# I2C connection:
i2c = busio.I2C(board.SCL, board.SDA)
sensor1 = IMU.LSM9DS1_I2C(i2c, address=0x40)
sensor2 = IMU.LSM9DS1_I2C(i2c, address=0x45)

# SPI connection:
spi = busio.SPI(board.SCK,
                MOSI=board.MOSI,
                MISO=board.MISO)
radio = CC1101(spi, freq=434400000, "666A")

radio.setupRX()
radio.setupCheck()

# Loop
while True:
 if 'take measurement' in radio.reciveData():
   [print(x, y, z) for x,y,z in sensor1.accel
 time.sleep(1)
```