Parker Melling's Exit Presentation

ARMSTRONG

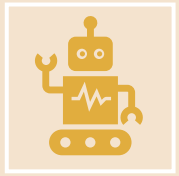
FLIGHT RESEARCH CENTER

Introduction

- Code 550 Intern
- Mentor: Sarkis Mikaelian
- Supervisor: John Baca
- Originally from Indiana
- Go to school at Georgia Tech
- Majoring in computer science and aerospace engineering
- Started at NASA in January at Marshall Space Flight Center under Flight Software Design and Development for SLS
- Started at Armstrong in May under Code 550

Projects

ARMSTRONG
FLIGHT RESEARCH CENTER



Hybrid Physics and NN-Based Runtime Assurance for Integrated Vehicle Health Management on HPSC

- Control Surface Actuator Real-time Onboard Digital Twin

Open Avionics Testbed for Flight Control Research, Rapid Integration, Validation and Technology Maturation

- Next-generation unmanned Research Flight Control System (NuRFCS)

High Performance Spaceflight Computer (HPSC)

ARMSTRONG
FLIGHT RESEARCH CENTER

Radiation-hardened flight computer for NASA missions through 2040+

Enables onboard AI/ML and real-time autonomy for deep-space missions

≥100× legacy processor performance for science and crewed ops

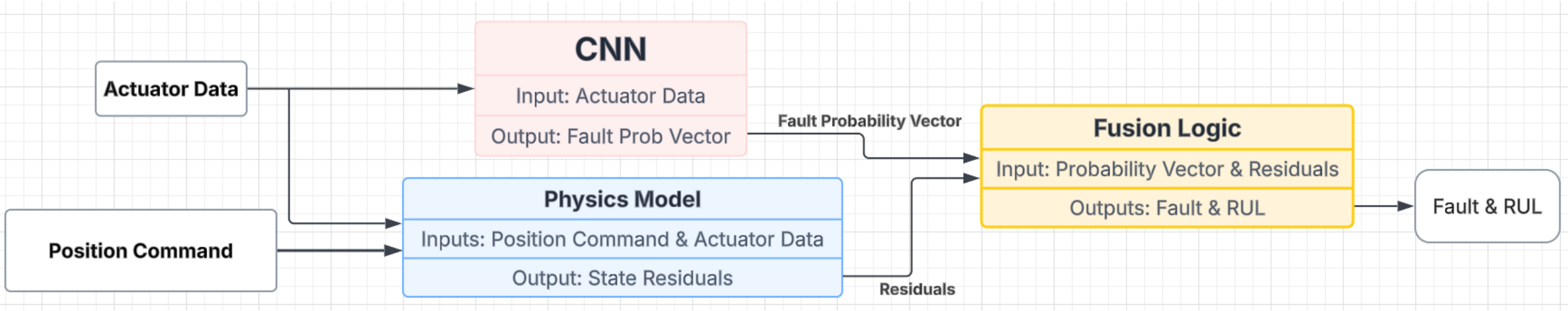
Scalable power management with robust fault tolerance

High-speed networking for distributed spacecraft systems

My Team's Work

- Developed fault identification and isolation algorithm
- Currently targets actuator faults; architecture designed to be system-agnostic
- Fuses CNN outputs with physics-model residuals to determine fault type and RUL
- Results to be submitted/presented at IEEE and AIAA

System Architecture



My Work

- Develop digital twin of actuator
  - Dynamics
  - Torque
  - Current
  - Voltage
  - Temperature
  - PWM
  - Fault injection

How to Create a Digital Twin

ARMSTRONG
FLIGHT RESEARCH CENTER

Digital twin: simulated version of a real system

Useful for predicting how a system responds to a given input

Start by modeling something measurable

Use physics and measured relationships to expand the model

Validate by comparing simulation outputs to the real system

ARMSTRONG
FLIGHT RESEARCH CENTER

Input-Output Model Space

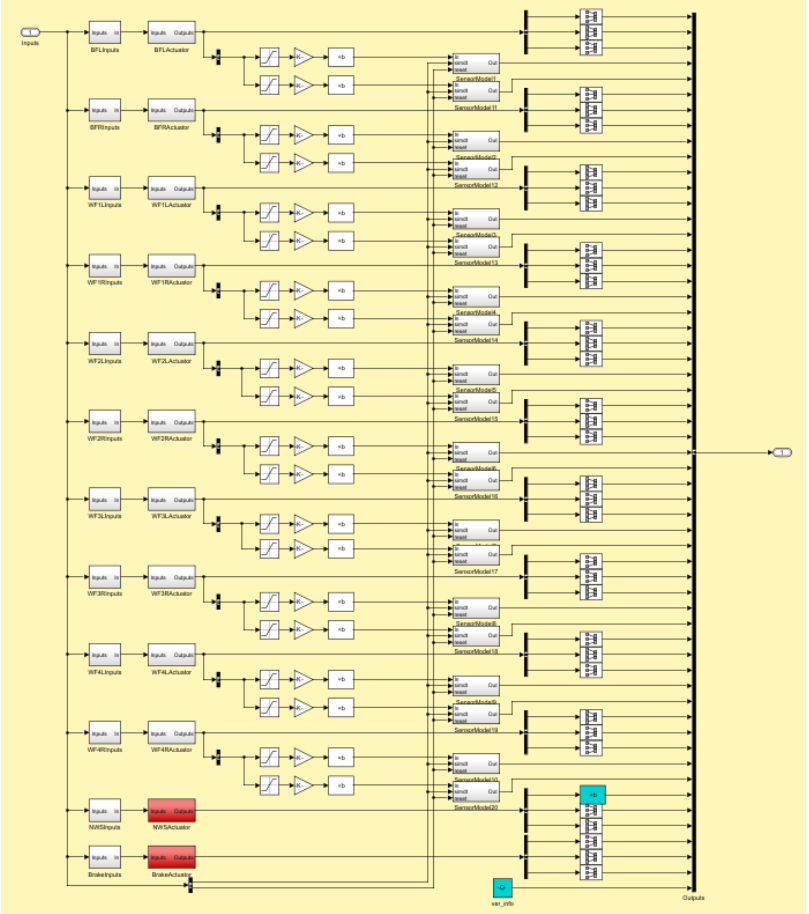
14 inputs, 24 outputs

High Level

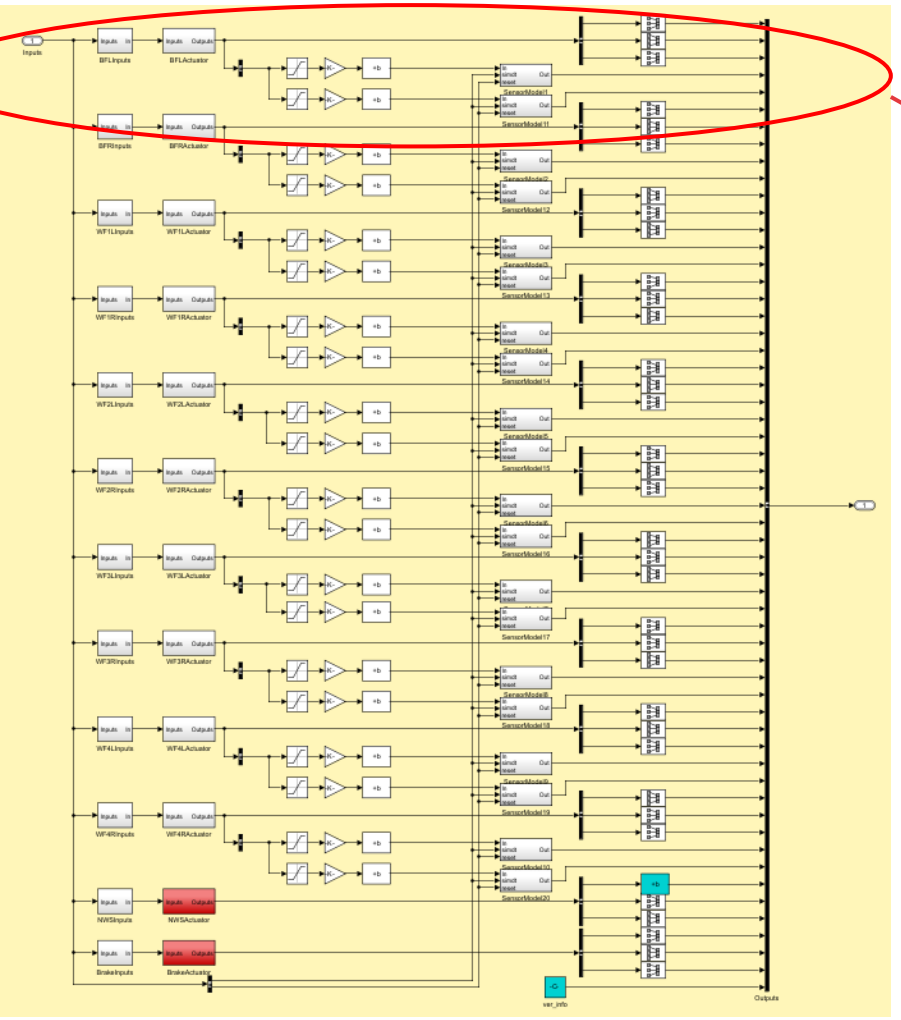
- Key input: actuator command (where do we want it to go?)
- Key outputs: actuator data (what's happening inside – torque, current, voltage, temperature)

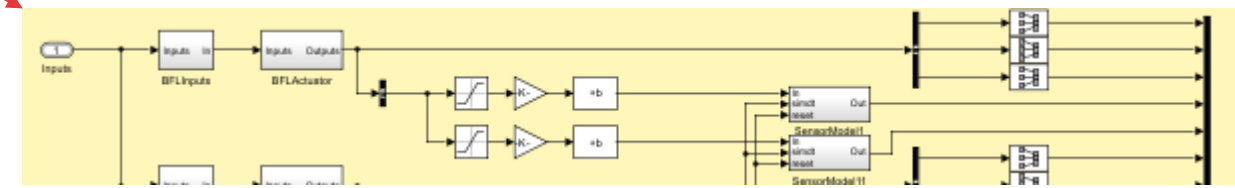
Starting Point

- Given a Simulink model of 12 X-56 actuators
- Key inputs: commanded positions
- Outputs: motion data (position, velocity, acceleration) & sensor voltages



Step 1: Isolate Single Actuator & Define Buses

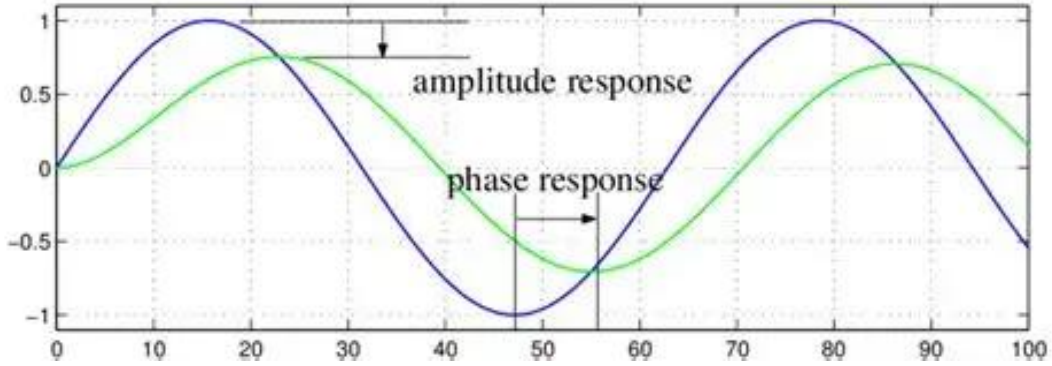




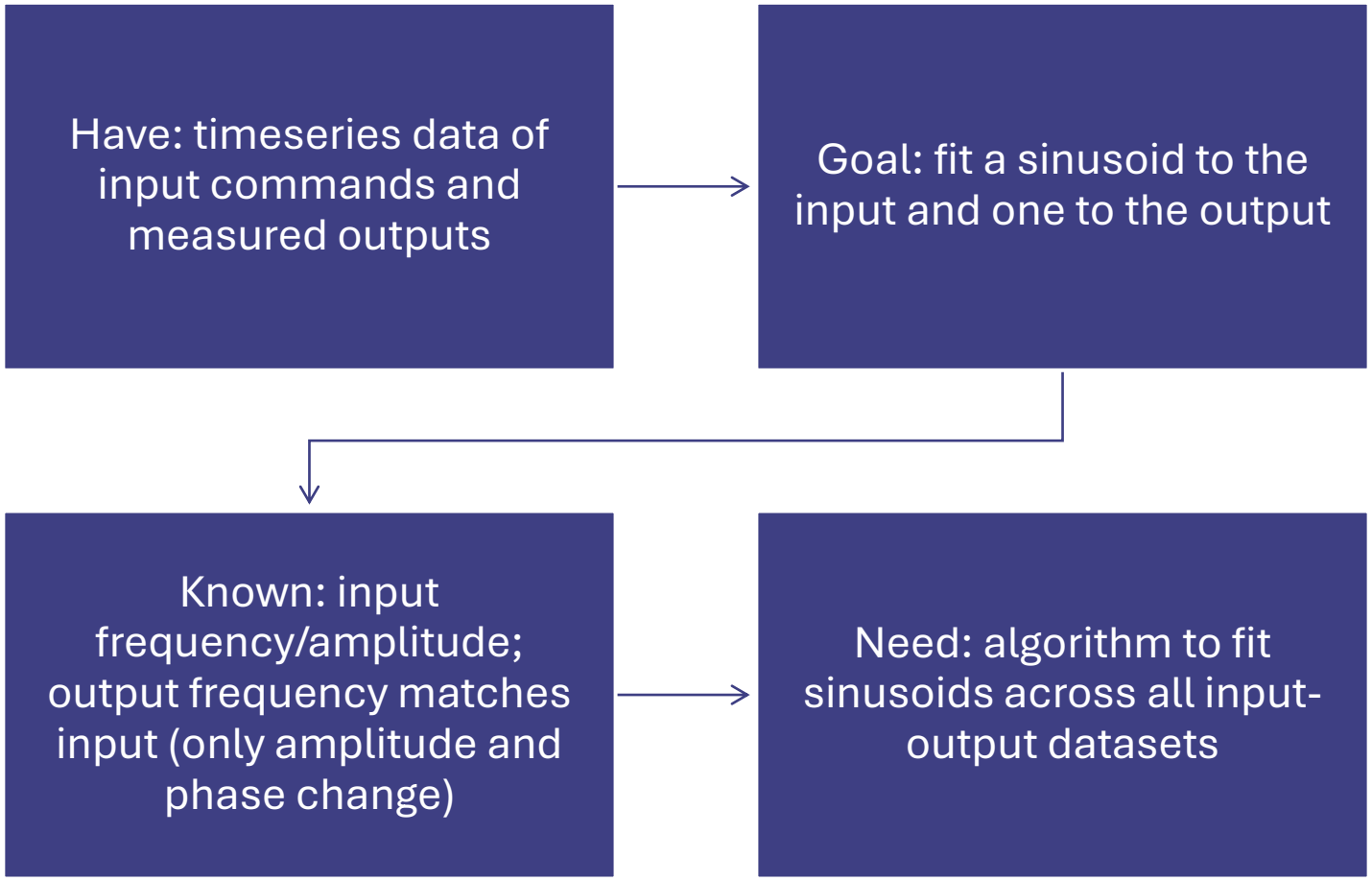
- Extracted a single actuator from the 12-actuator model into its own standalone model

Step 2: Collect Frequency Sweep Data

- Frequency sweep: low-amplitude sinusoids at various frequencies input to derive input-output relationship
- Transfer function: shows how much output is amplified/attenuated and shifted relative to input
- Data collected at 2.0° amplitude across 17 frequencies (0.1–30 Hz)



Step 3: Fit Frequency Sweep Data



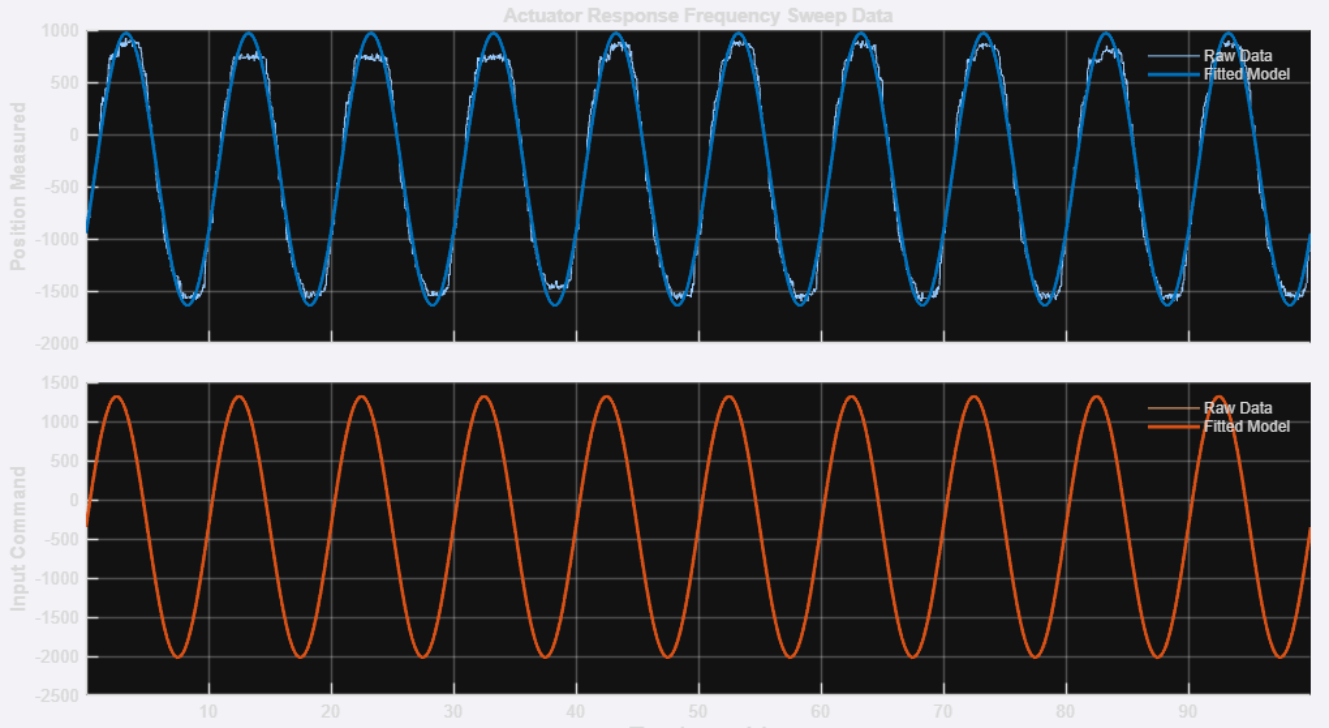
Step 3: Fit Frequency Sweep Data Approach

ARMSTRONG
FLIGHT RESEARCH CENTER

1. Model each timeseries as $y(t) \approx \alpha \sin(\omega t) + \beta \cos(\omega t) + c$
2. Linear least squares: $\begin{bmatrix} \sin(\omega t_1) & \cos(\omega t_1) & 1 \\ \vdots & \vdots & \vdots \\ \sin(\omega t_n) & \cos(\omega t_n) & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ c \end{bmatrix} = \begin{bmatrix} y(t_1) \\ \vdots \\ y(t_n) \end{bmatrix}$
3. Fit parameters: $\begin{bmatrix} \alpha \\ \beta \\ c \end{bmatrix} = (A^T A)^{-1} A^T b$
4. Convert to single sinusoid: $B = \sqrt{\alpha^2 + \beta^2}$ & $\varphi = \tan^{-1}\left(\frac{\beta}{\alpha}\right) \rightarrow B \sin(\omega t + \varphi) + c$
5. Compute Gain & Phase: $gain = \frac{B_{out}}{B_{in}}$ & $\varphi_{lag} = \varphi_{out} - \varphi_{in}$
6. Compute R^2: $1 - \frac{\sum_{i=1}^{n} (y(t_i) - B \sin(\omega t_i + \varphi) + c)^2}{\sum_{i=1}^{n} \left(y(t_i) - \overline{y(t_i)}\right)^2}$

Step 3: Fit Frequency Sweep Data Approach

- Wanted a way to visually assess fit quality beyond the R^2 value
- Built an interactive tool to compare sinusoid fits against measured data



Step 4: Create Bode Plot & Fit Transfer Function

ARMSTRONG
FLIGHT RESEARCH CENTER

- Goal: create a Bode plot and fit a transfer function to the frequency sweep data

- Started with an assumed third-order dynamic model:

$$G(s) = \frac{\tau\omega_n^2}{(s+\tau)\left(s^2+2\zeta\omega_n s+\omega_n^2\right)}, \text{ where } s = j\omega$$

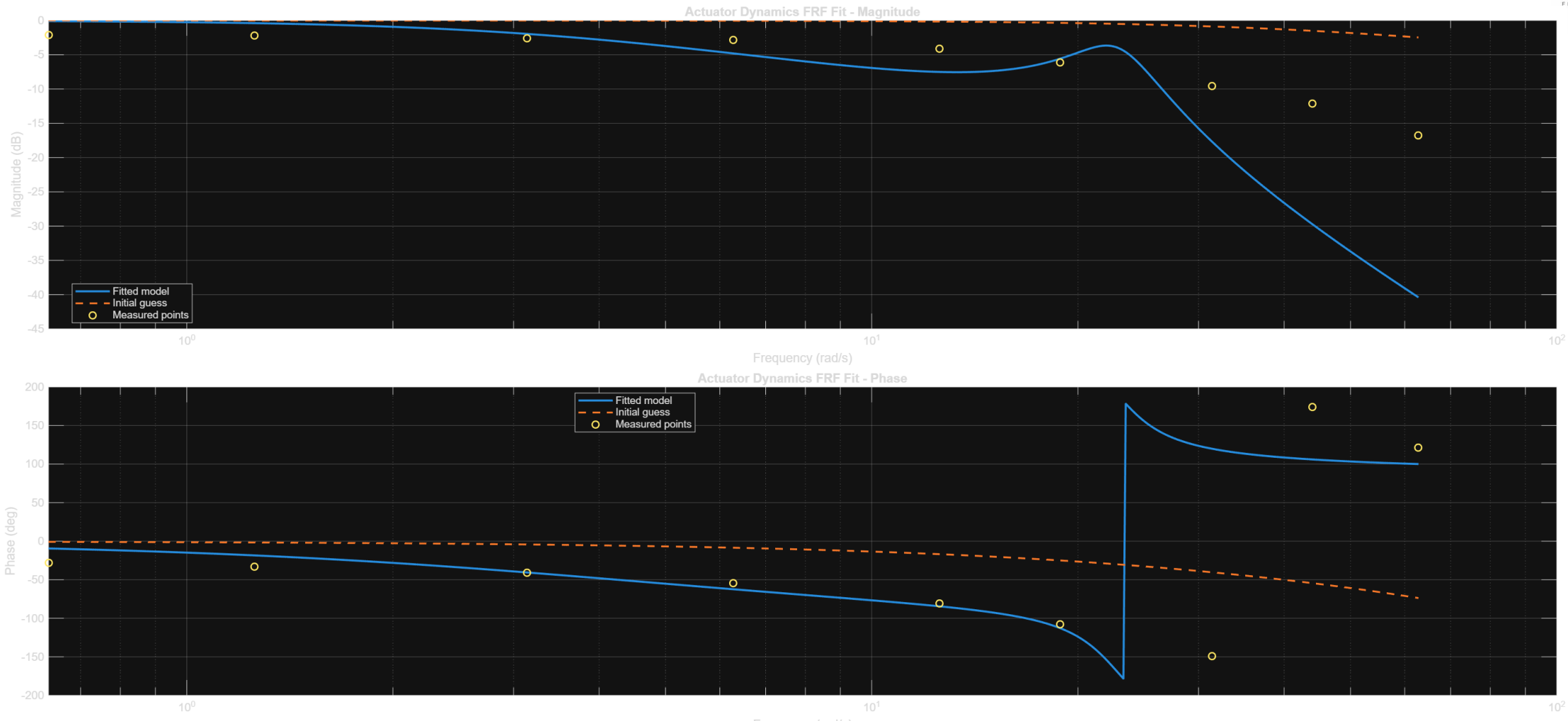
- Built an algorithm to fit τ, ω_n, ζ using the gain and phase results from Step 3

Step 4: Bode Plot & Fit Transfer Function Approach

ARMSTRONG
FLIGHT RESEARCH CENTER

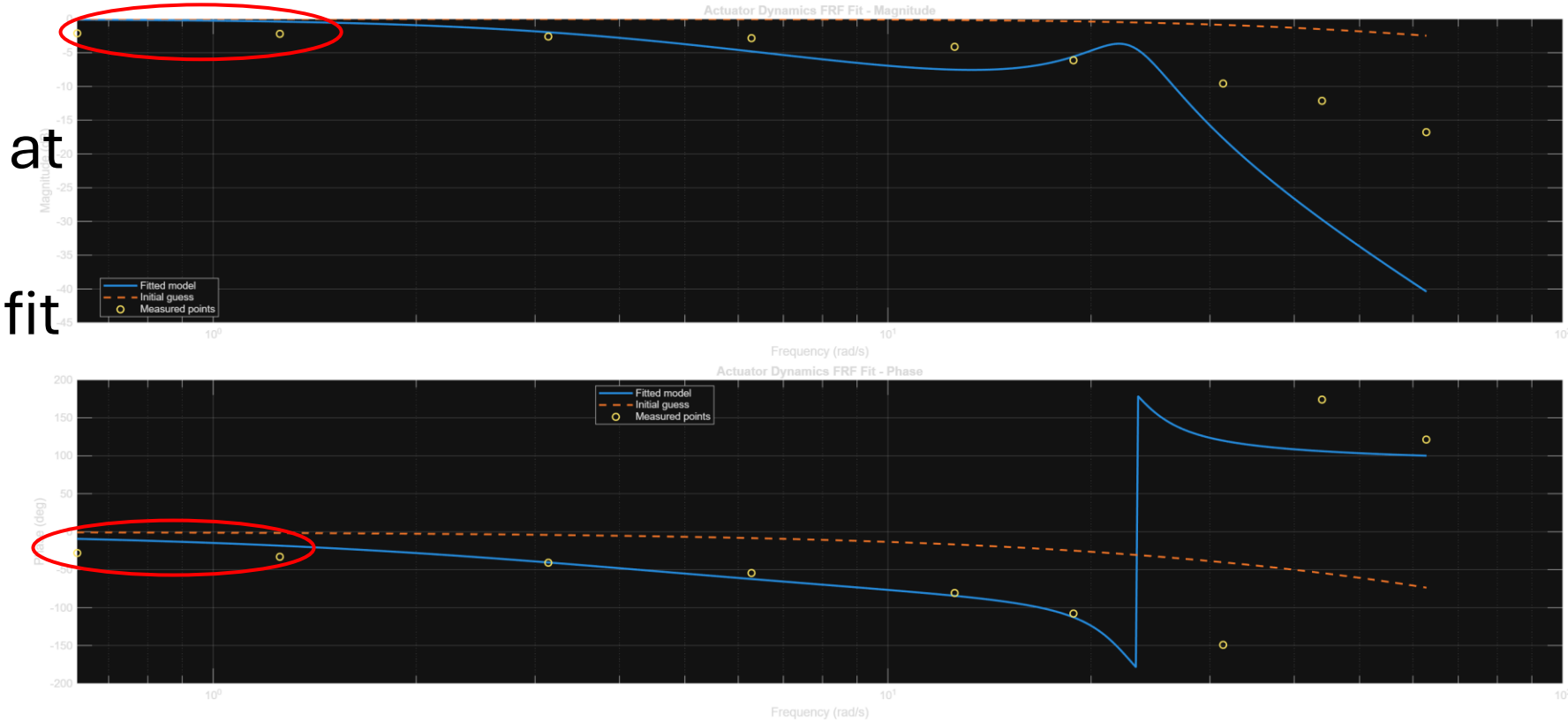
1. Read phase & gain from frequency fit
2. Construct complex frequency response: $H_{meas} = gain \cdot e^{j\phi}$
3. Define transfer function: $G(s) = \frac{\tau {\omega_n}^2}{(s+\tau)(s^2+2\zeta\omega_n s+{\omega_n}^2)}$
4. Build objective function: $J = \sum_k \left((\Delta \log|H|)^2 + (\Delta\phi)^2 \right)$
5. Optimize parameters using nonlinear least squares (same "family" as gradient descent and SPSA)

Step 4: Bode Plot & Fit Transfer Function Plot



Step 5: Bode Plot Analysis

- ~25% anomalous attenuation at 0.1 Hz
- Severe 27° phase shift at 0.1 Hz
- Poor overall dynamics fit
- Actuator appears abnormally sluggish



Step 6: Investigation

ARMSTRONG
FLIGHT RESEARCH CENTER

Tested varying amplitudes at low frequencies to check for amplitude-dependence

- No significant change in attenuation or phase delay

Investigated internal actuator PID control loop gains

- PID: P controls response speed, I corrects steady-state error, D acts as a damper

Step 7: Adjust Gains

Increased P gain: more aggressive response, but underdamped

Increased I gain: eliminated steady-state error

Result: actuator responds much quicker with slight overshoot, no steady-state error

ARMSTRONG
FLIGHT RESEARCH CENTER

Step 8: Re-do Sweep, Fit, & Bode Plot

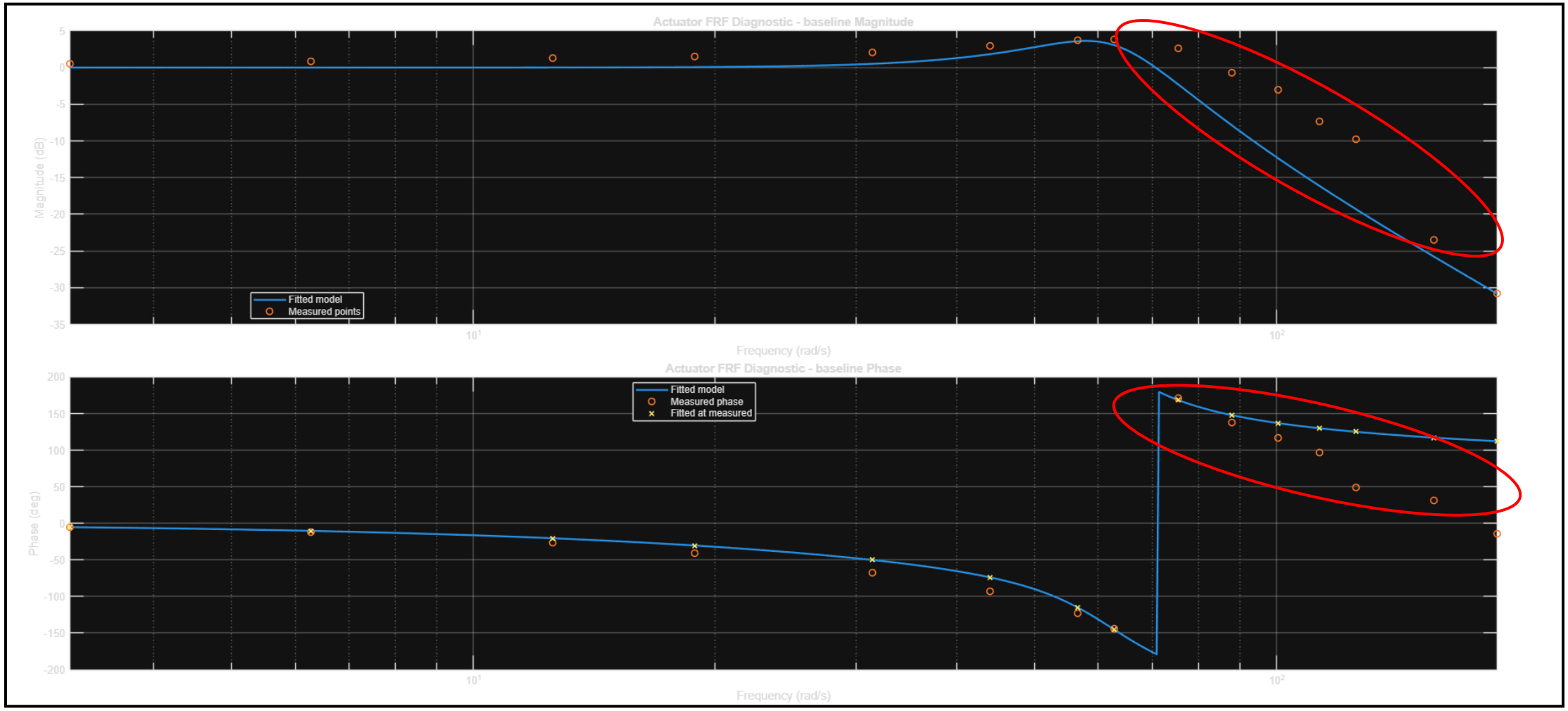
ARMSTRONG
FLIGHT RESEARCH CENTER

PID gain change → dynamics change

Collected new frequency sweep data

Refit transfer function with new Bode plot

Step 8: Redone Bode Plot



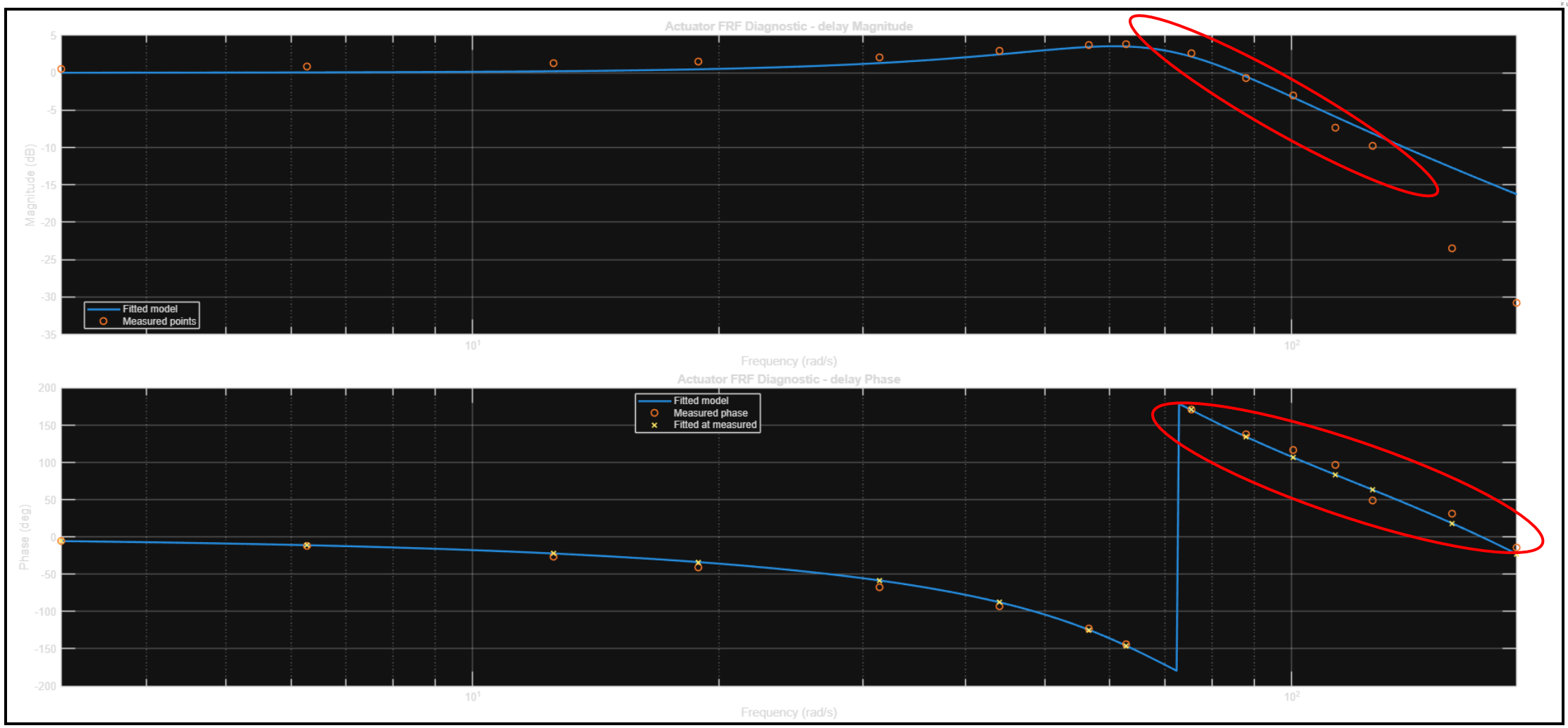
Step 9: Transfer Function Modification

- Original transfer function poorly fit higher-frequency data
- A 3-pole model can only produce up to 270° of phase lag
- Measured phase lag exceeded 270° → suggests missing dynamics or an unaccounted delay
- Hypothesis: input-output delay is the cause
- Added tunable delay term to the model:

$$G(s) = e^{-sT_d} \frac{\tau\omega_n^2}{(s+\tau)(s^2 + 2\zeta\omega_n s + \omega_n^2)}$$

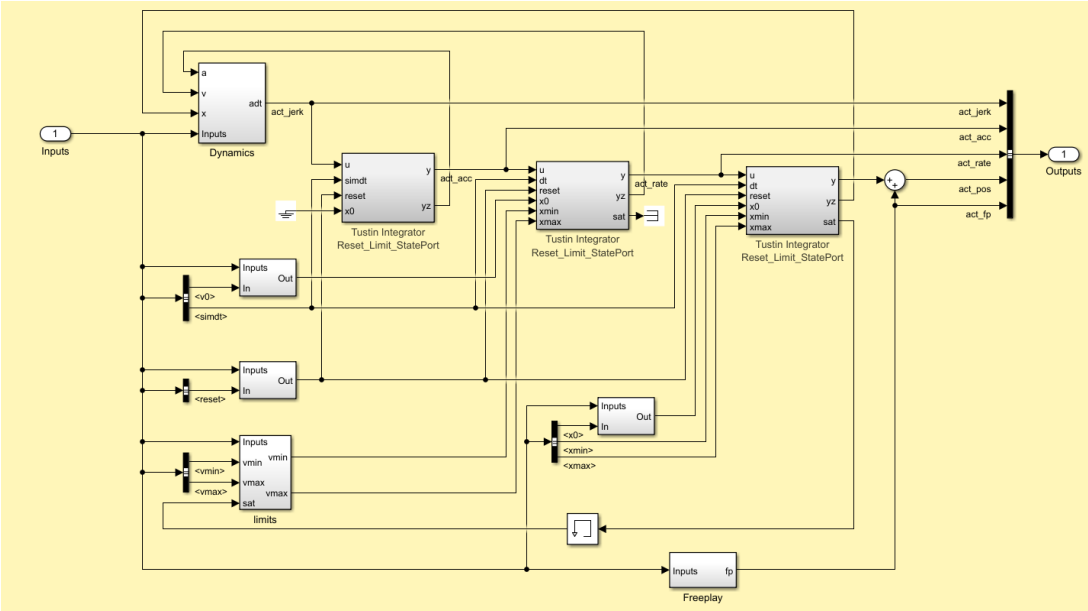
- Updated fitting script to solve for this new delay parameter

Step 9: Transfer Function Modification



Step 10: Compute Parameters & Update Model

- Using the nonlinear least squares approach, computed:
  - $\omega_n = 70.1753$
  - $\zeta = 0.355294$
  - $\tau = 9105.14$
  - $T_d = 0.0202589$
- Updated model with new transfer function structure and parameters



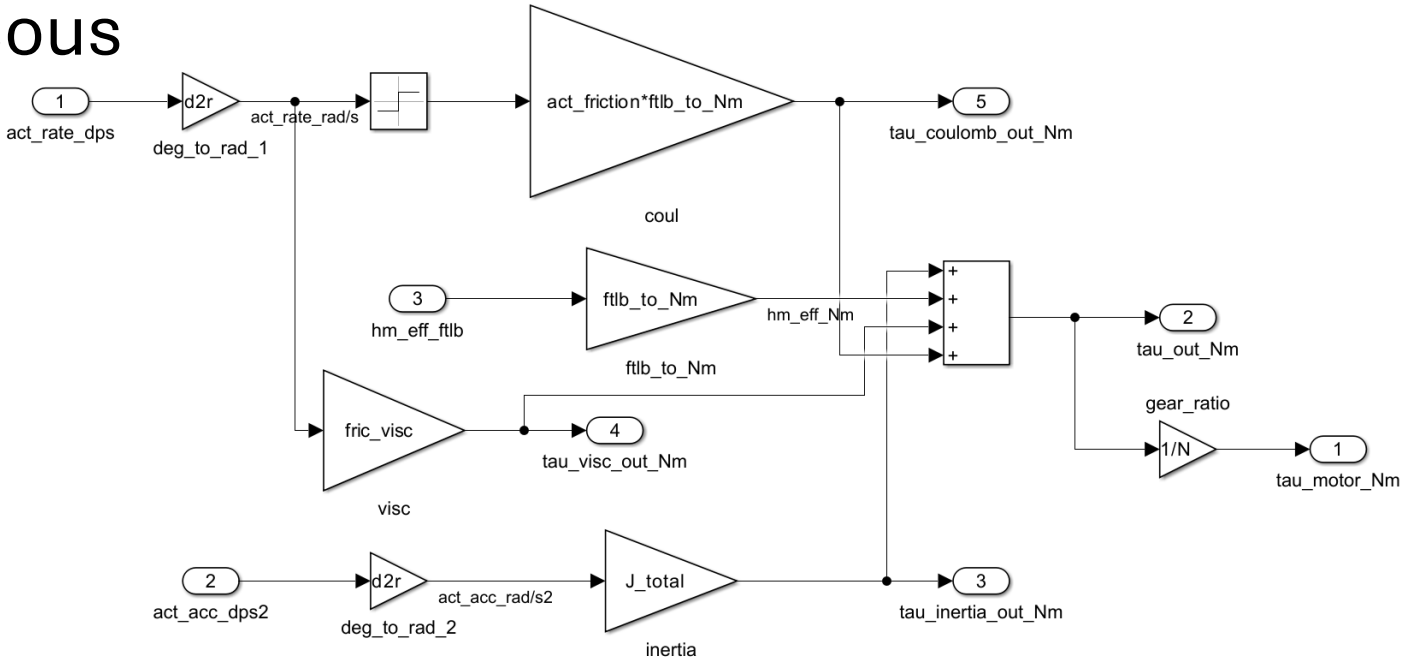
Step 10: Torque Model

- Dynamics are now fully encapsulated in the model
- Next: compute the torque the actuator must generate
- Forces impeding rotation: gravity, Coulomb friction, viscous friction
- With dynamics modeled, we can calculate net torque and derive the exact motor torque needed for the current motion:

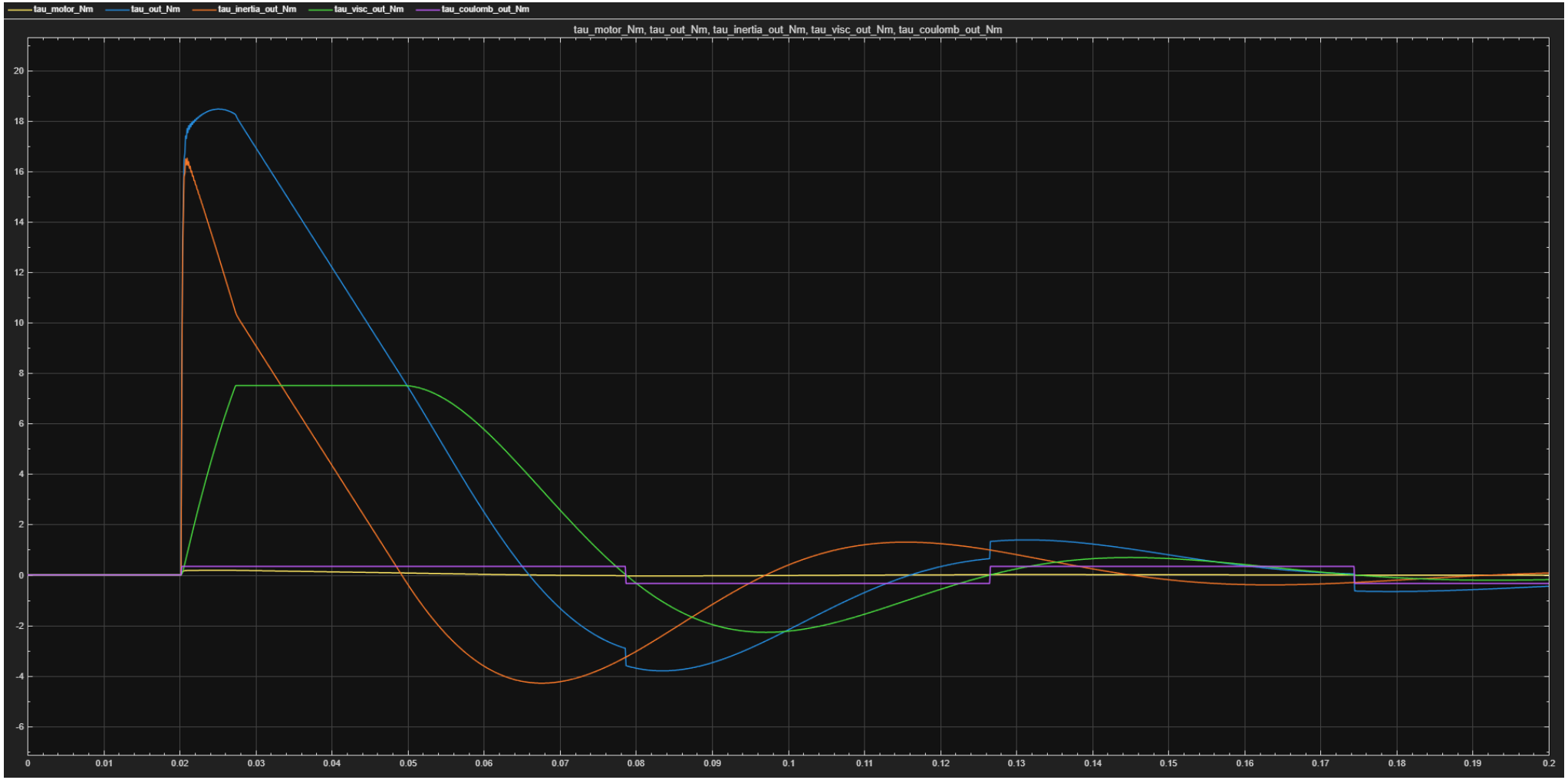
$$\tau_{motor} = \frac{\tau_{coulomb} + \tau_{hm} + B\omega_{act} + J_{total}\alpha_{act}}{N}$$

Step 10: Torque Model Architecture

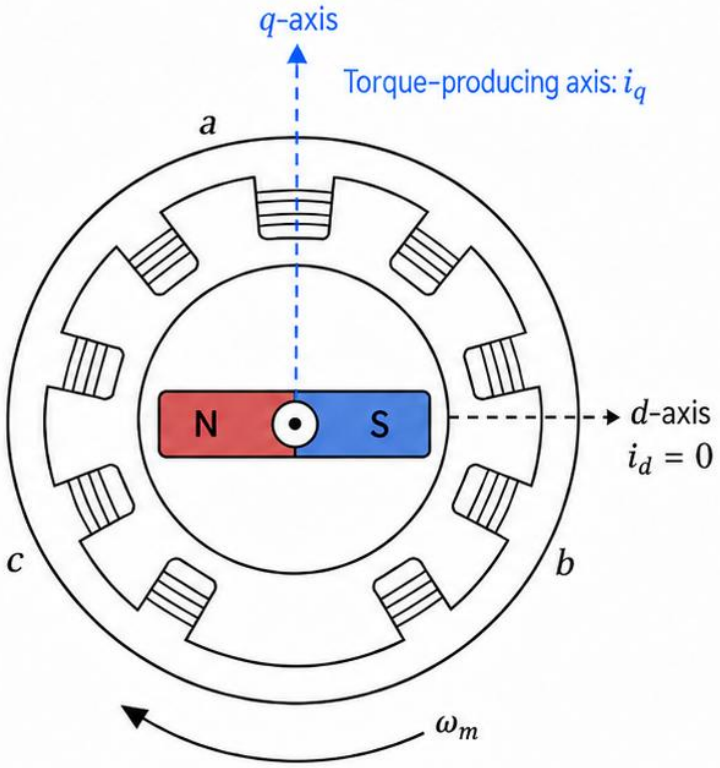
- Inputs: actuator rate, actuator acceleration, & gravitational hinge moment
- Outputs: motor torque, output torque, inertial torque, viscous torque, coulomb torque



Step 10: Torque Model Data (step input)



PMSM Voltage / Current Subsystem (FOC, $i_d = 0$)



Electrical speed scales with pole pairs:

$$\omega_e = N_{pp}\,\omega_m$$

① $i_d = 0$

↓

② $i_q = \dfrac{\tau}{K_t}$

↓

③ $I_{\text{phase,rms}} = \dfrac{|i_q|}{\sqrt{2}}$

↓

④ $\omega_e = N_{pp}\,\omega_m$

↓

⑤ $e_{\text{bemf}} = K_e\,\omega_e$

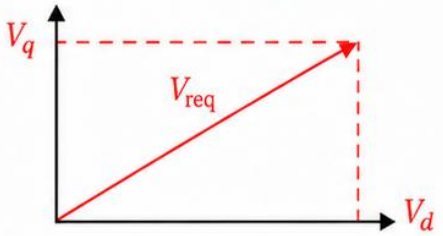
↓

⑥ $V_q = \underbrace{R i_q}_{\text{resistive drop}} + \underbrace{L\dfrac{di_q}{dt}}_{\text{dynamic / inductive term}} + \underbrace{e_{\text{bemf}}}_{\text{back-EMF term}}$

↓

⑦ $V_d = -L\omega_e i_q$

$$V_{\text{req}} = \sqrt{V_d^2 + V_q^2}$$



→ main current axis (torque-producing current)

Ⓐ phase rms current

electrical speed

back-EMF increases with electrical speed

V_q q-axis voltage

V_d d-axis voltage (cross-coupling inductive term)

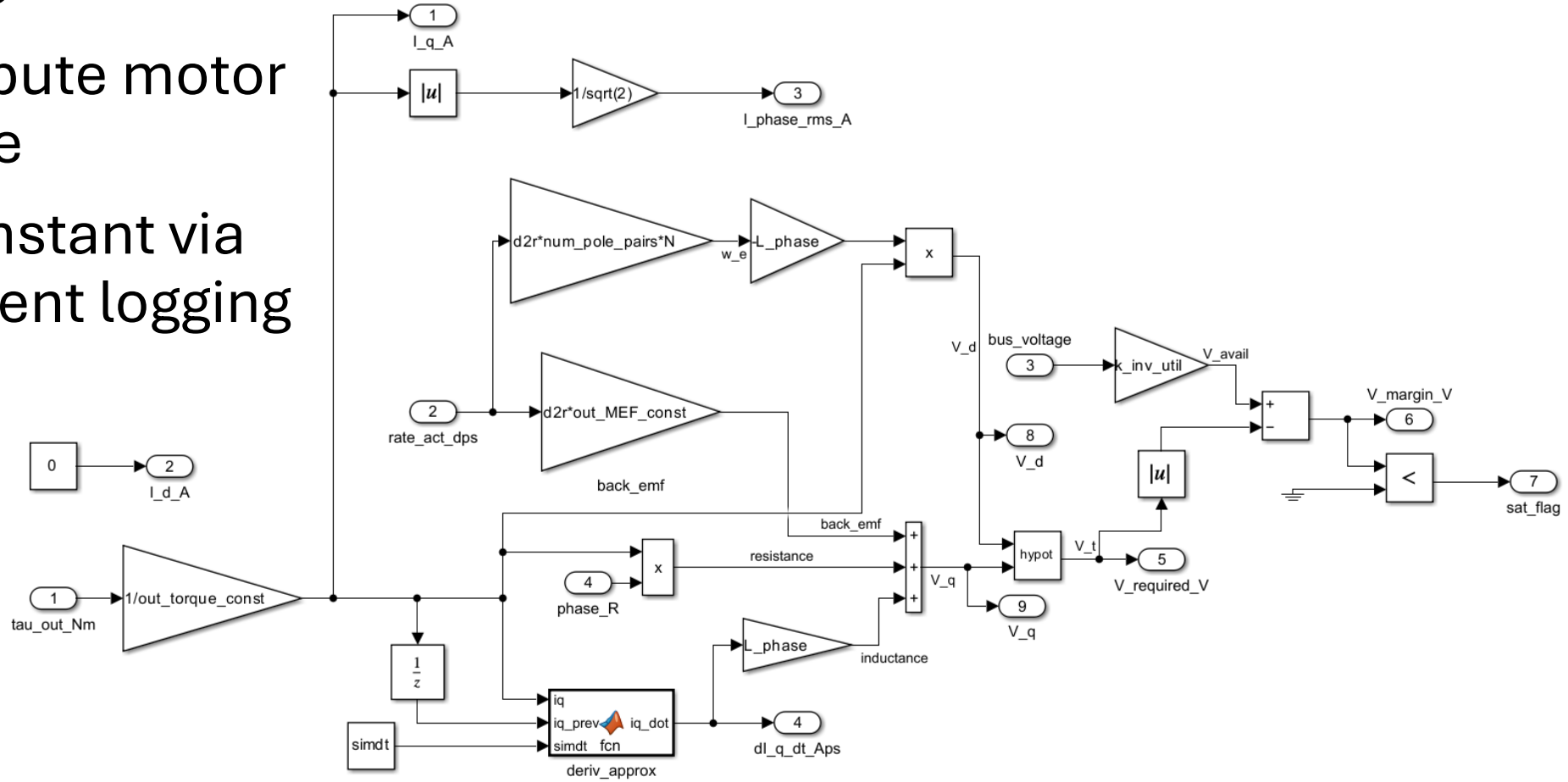
V_{req} required voltage magnitude

Saturation occurs when available inverter voltage is less than required voltage.

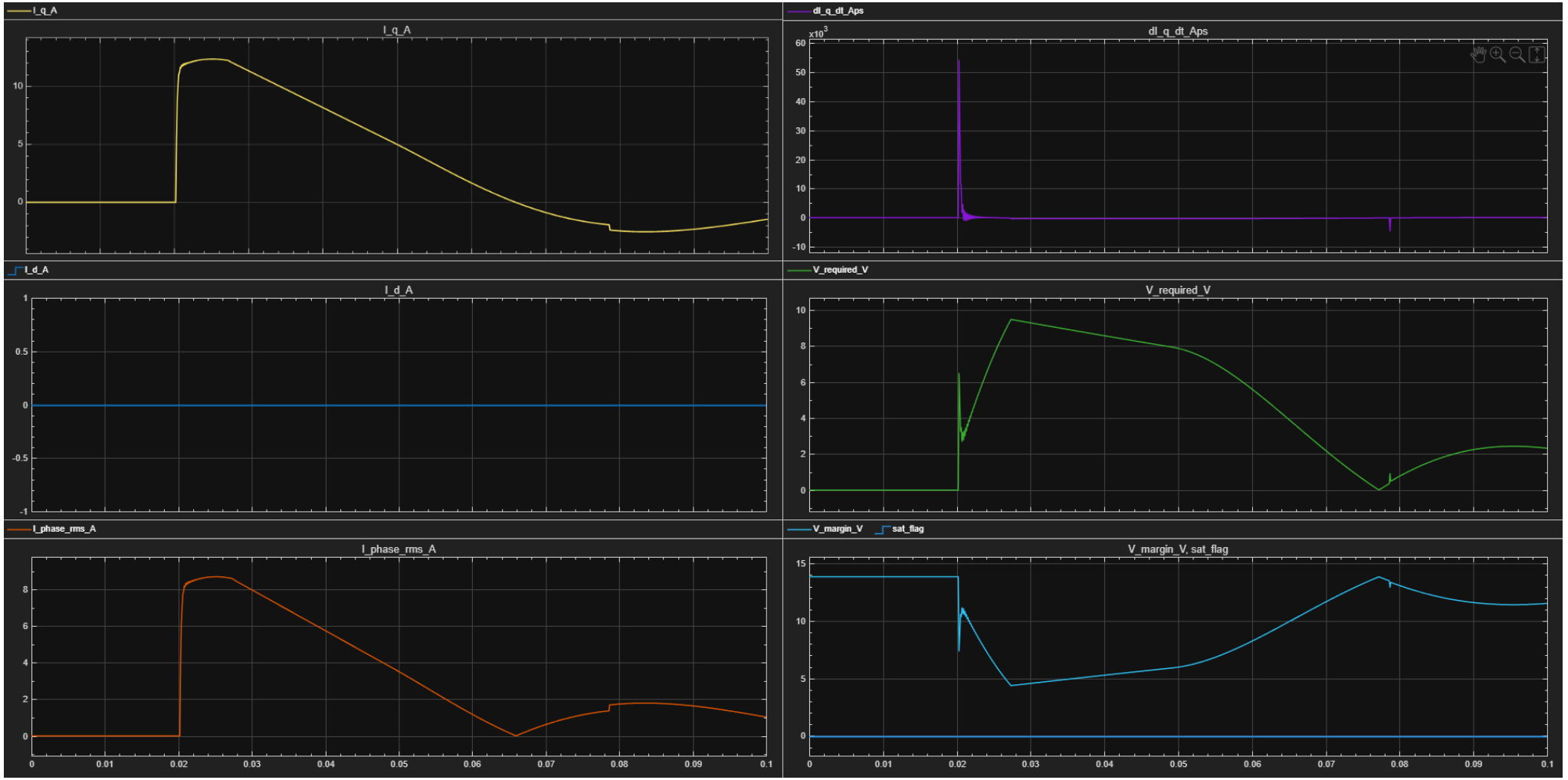
ARMSTRONG
FLIGHT RESEARCH CENTER

Step 11: Current & Voltage Model

- Model now simulates dynamics & torque
- From torque, compute motor current and voltage
- Derived torque constant via constant load current logging

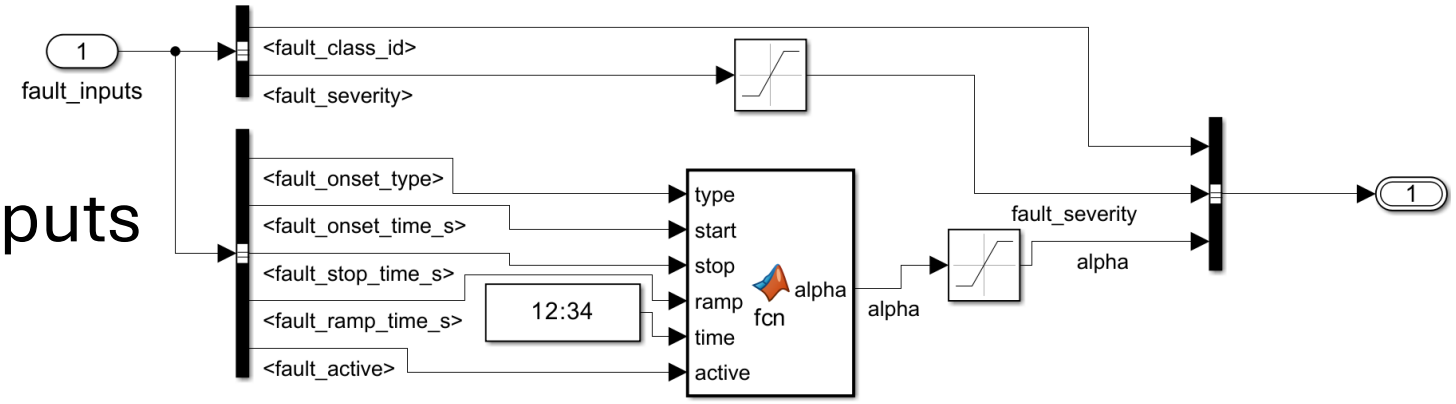


Step 11: Current & Voltage Model Data (step input)



Step 12: General Fault Injection Subsystem

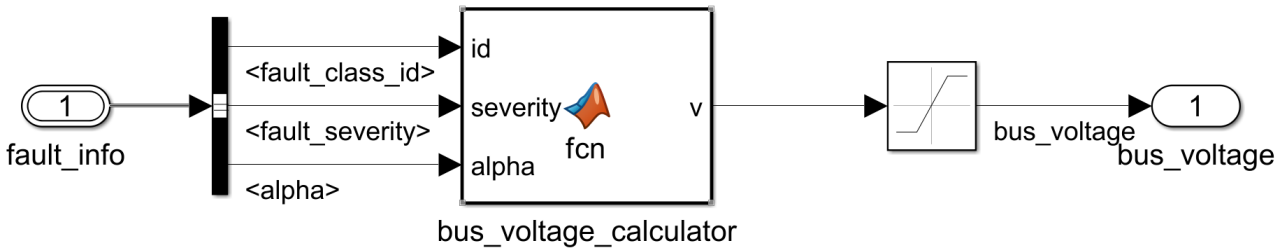
- Dynamics, torque, current, and voltage now modeled
- Wanted a generic, expandable way to inject faults into the simulation
- Built subsystem that takes fault configuration and outputs fault progression and ID
- Enables testing how the system responds to various faults

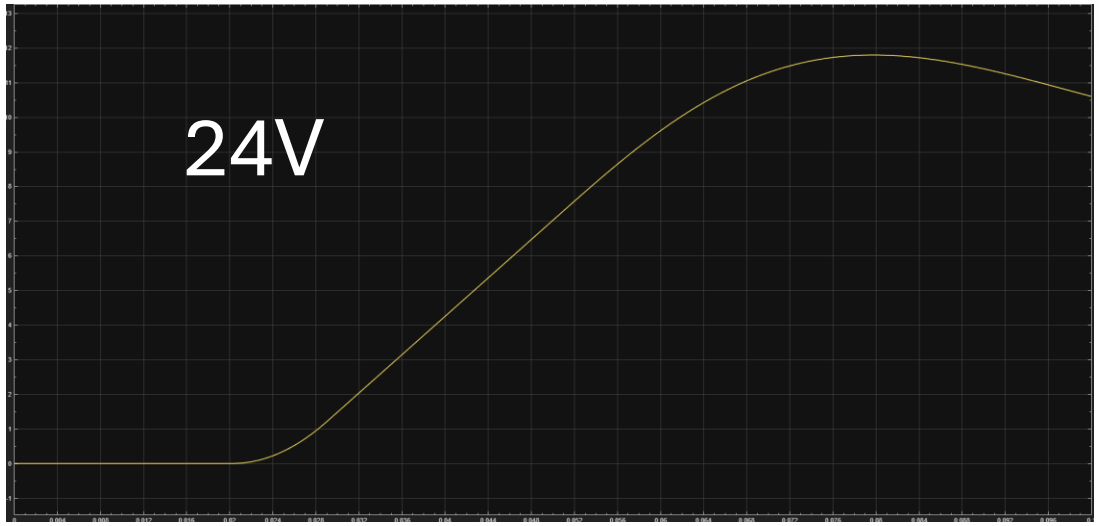


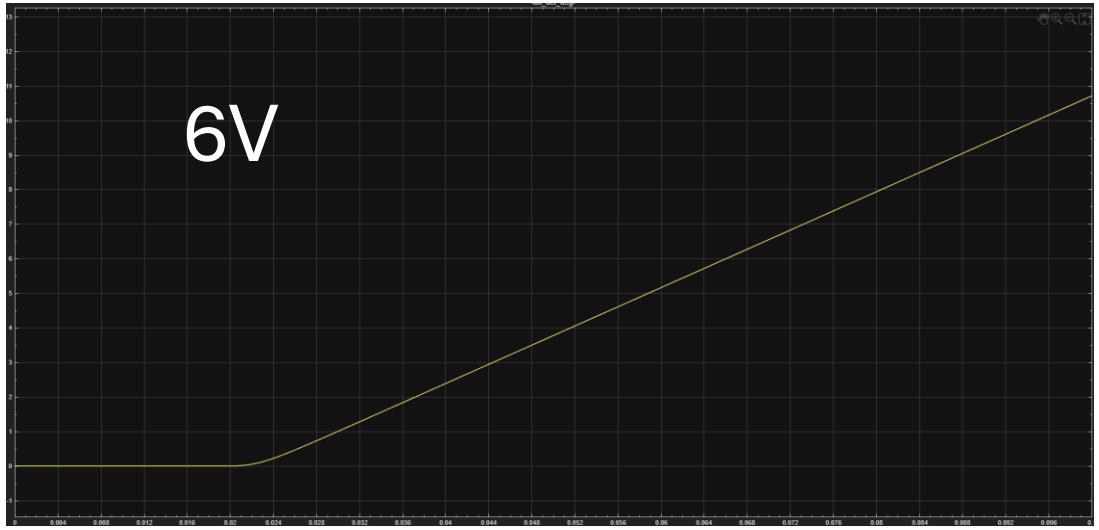
Step 13: Undervoltage Fault Subsystem

- Undervoltage fault: inadequate voltage supplied to actuator
- Causes sluggish behavior or complete inability to move
- Severity scales voltage between 0V (severity=1) and 24V (severity=0)

severity = 1 -> 0v
severity = 0 -> 24v

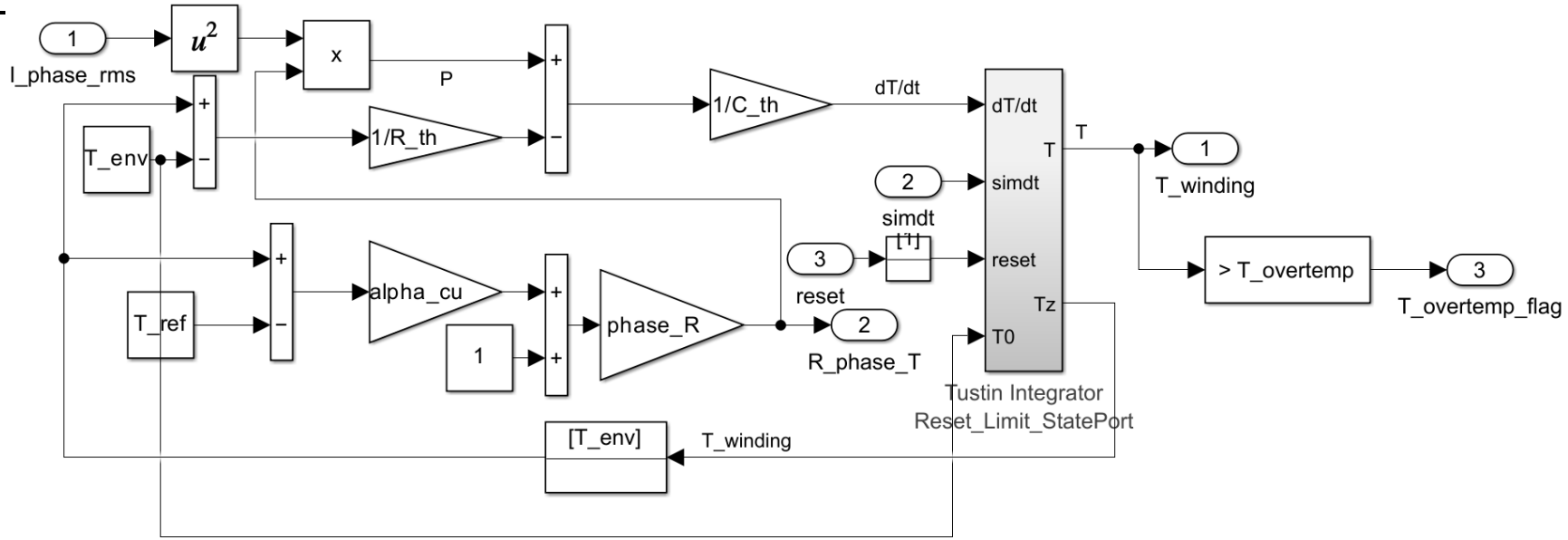






Step 14: Thermal Model

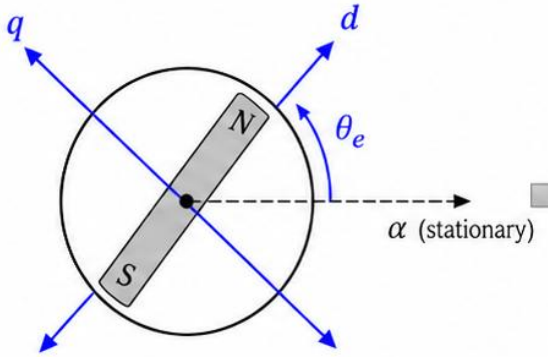
- An accurate thermal model helps detect thermal-related anomalies and faults
- Model captures how winding temperature evolves from resistive heating and heat dissipation:
  - $R_{phase\_T} = R_{phase,ref}\left(1 + \alpha_{cu}\left(T_{winding} - T_{ref}\right)\right)$
  - $P_{loss} = I_{phase,rms}^2 \, R_{phase\_T}$
  - $\dot{T}_{winding} = \dfrac{P_{loss} - \frac{T_{winding} - T_{env}}{R_{th}}}{C_{th}}$



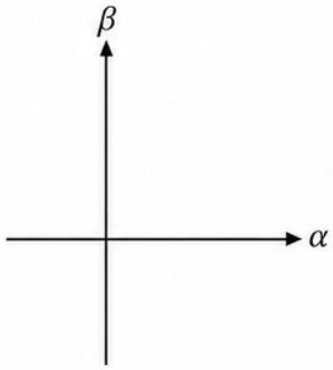
Inverse Park-Clarke Transformation (PMSM Controller)

$$\theta_e = N_{pp}\,\theta_m$$

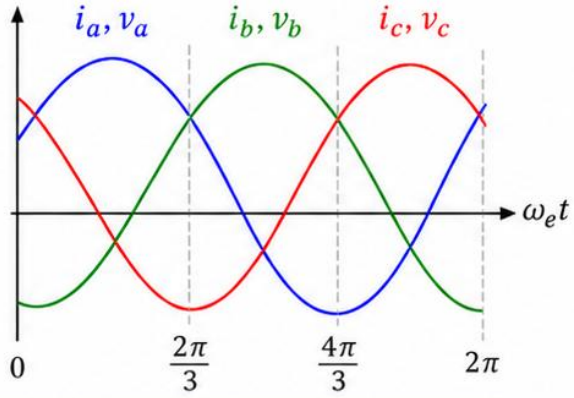
1) Rotating dq frame (attached to rotor)



2) Stationary $\alpha\beta$ frame (orthogonal)



3) Three-phase stator quantities (stator phase quantities)



Current transformation ($dq \rightarrow \alpha\beta \rightarrow$ abc)

$dq \rightarrow \alpha\beta$ (Inverse Park)

$$i_\alpha = i_d \cos\theta_e - i_q \sin\theta_e$$
$$i_\beta = i_d \sin\theta_e + i_q \cos\theta_e$$

$\alpha\beta \rightarrow abc$ (Inverse Clarke)

$$i_a = i_\alpha$$
$$i_b = -\frac{1}{2} i_\alpha + \frac{\sqrt{3}}{2} i_\beta$$
$$i_c = -\frac{1}{2} i_\alpha - \frac{\sqrt{3}}{2} i_\beta$$

Voltage transformation ($dq \rightarrow \alpha\beta \rightarrow$ abc)

$dq \rightarrow \alpha\beta$ (Inverse Park)

$$v_\alpha = v_d \cos\theta_e - v_q \sin\theta_e$$
$$v_\beta = v_d \sin\theta_e + v_q \cos\theta_e$$

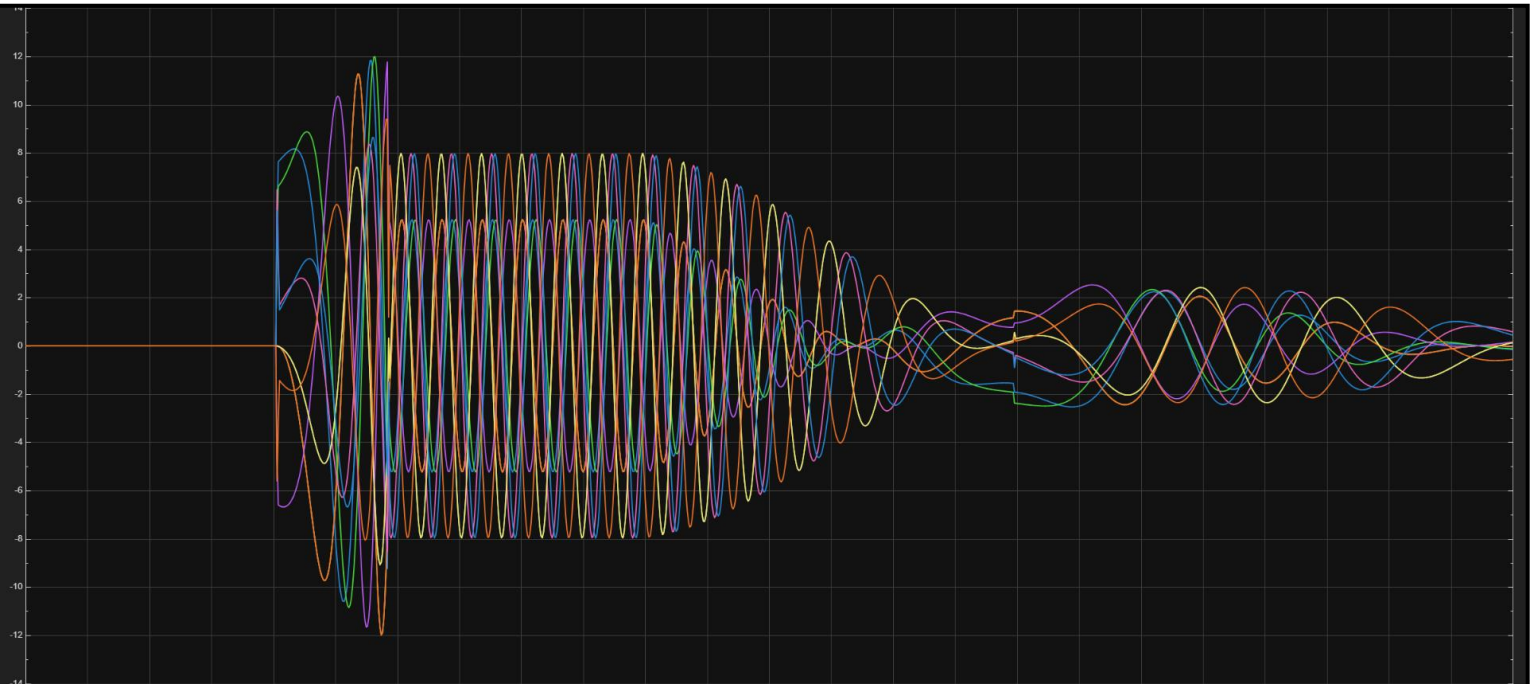
$\alpha\beta \rightarrow abc$ (Inverse Clarke)

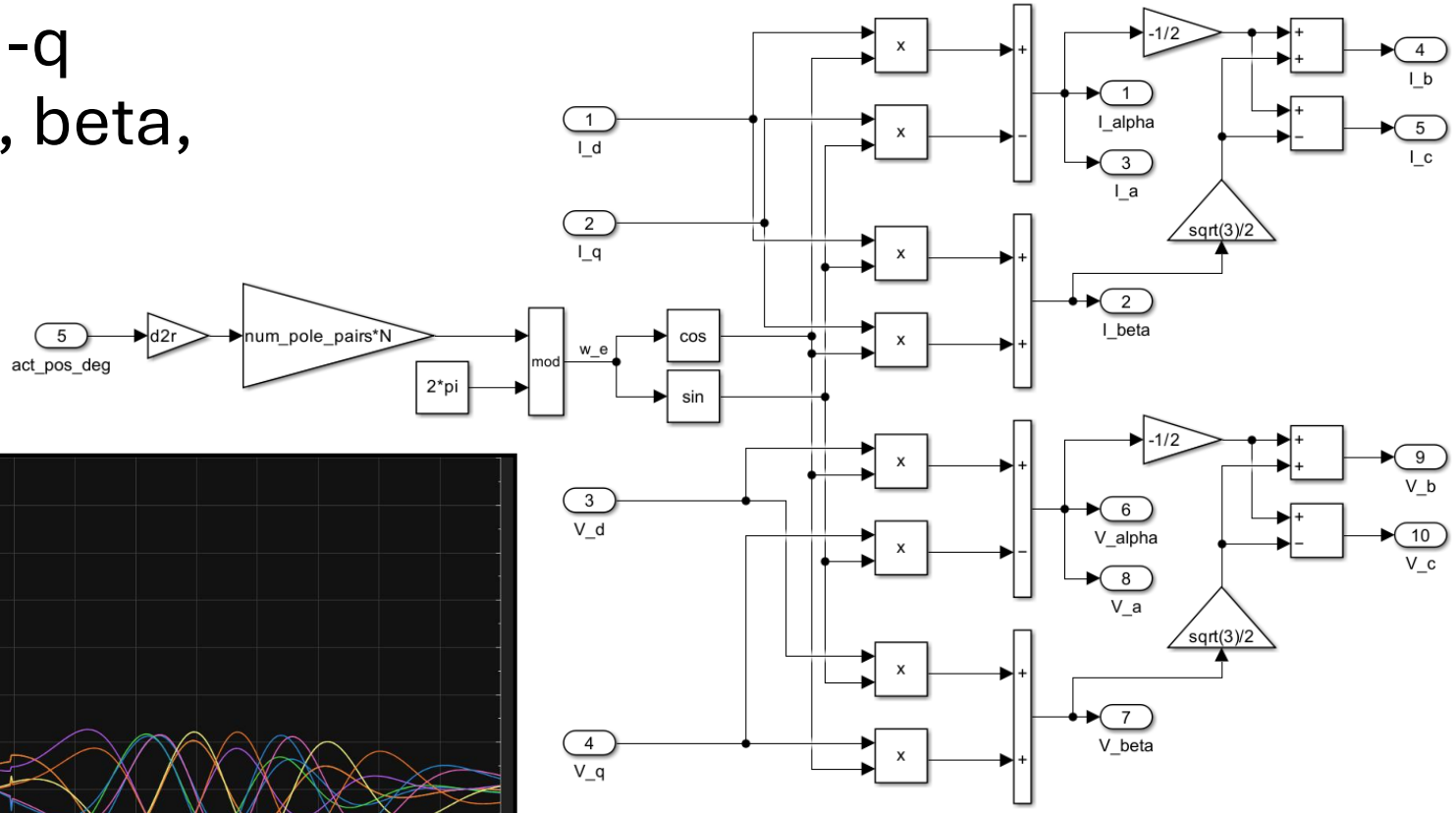
$$v_a = v_\alpha$$
$$v_b = -\frac{1}{2} v_\alpha + \frac{\sqrt{3}}{2} v_\beta$$
$$v_c = -\frac{1}{2} v_\alpha - \frac{\sqrt{3}}{2} v_\beta$$

dq = rotating control frame, $\alpha\beta$ = stationary orthogonal frame, abc = physical three-phase motor quantities.

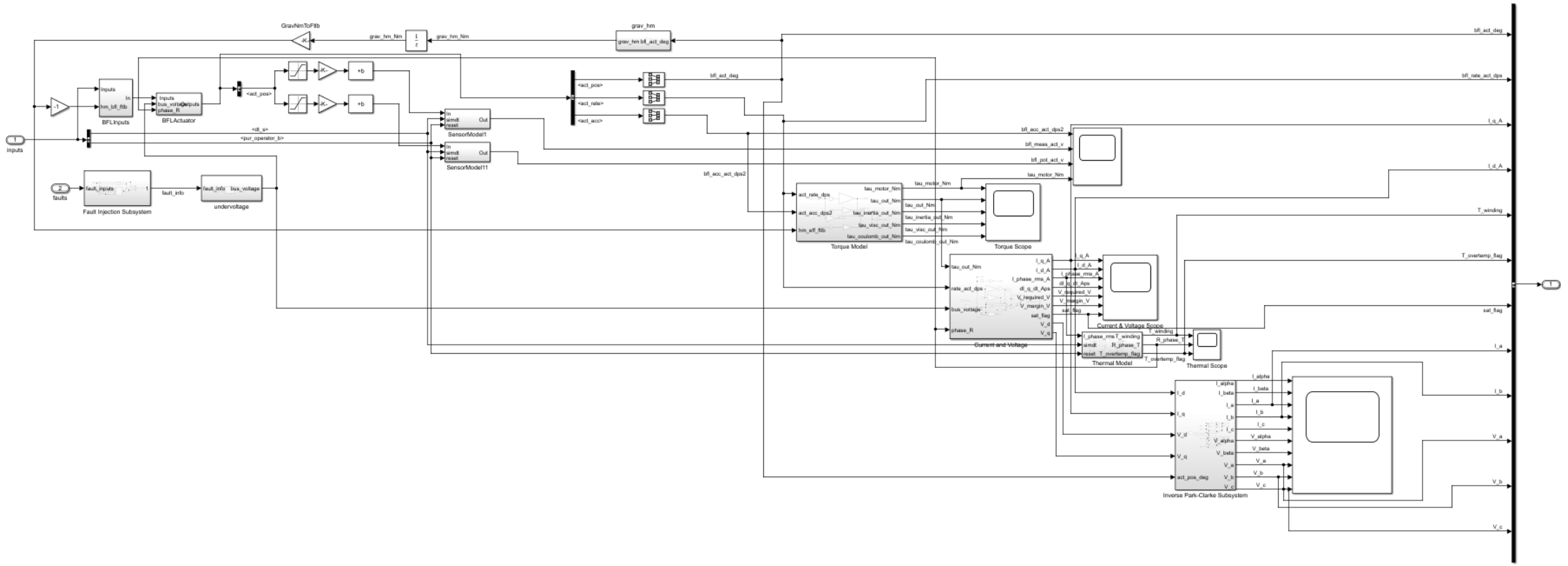
Step 15: Inverse Park-Clarke Transformation

- Built subsystem to convert d-q currents/voltages into alpha, beta, and three-phase quantities





Total Model


GravNmToFtlb
grav_hm_Nm
grav_hm
grav_hm bfl_act_deg
BFLInputs
BFLActuator
SensorModel1
SensorModel11
Fault Injection Subsystem
undervoltage
Torque Model
Torque Scope
Current and Voltage
Current & Voltage Scope
Thermal Model
Thermal Scope
Inverse Park-Clarke Subsystem
inputs
faults
bfl_act_deg
bfl_rate_act_dps
I_q_A
I_d_A
T_winding
T_overtemp_flag
sat_flag
I_a
I_b
V_a
V_b
V_c

Validation

ARMSTRONG
FLIGHT RESEARCH CENTER

Accuracy is critical for reliably detecting in-flight actuator faults

Quick dynamics check shows good agreement with the physical actuator

Currently validating the full digital twin:

Creating executables to command actuators (step, ramp, doublet) and log data	Building script to run logged inputs through the model and compare outputs	Will adjust model as needed based on results

Actuator Digital Twin Conclusion

ARMSTRONG
FLIGHT RESEARCH CENTER

- Created a digital twin combining physics theory and physical actuator data for fault identification and isolation
- Will run in parallel with the neural network fault classifier on the HPSC multi-core hardware
- Enables advanced fault-protection previously impossible with legacy processors
- Goal: complete validation, finalize model tweaks, and build real-time residual output script before I leave (last day 8/5)
- Will help write AIAA/IEEE paper on methodology and findings over the coming months

NuRFCS Project

ARMSTRONG
FLIGHT RESEARCH CENTER

NuRFCS: Next-generation unmanned Research Flight Control System

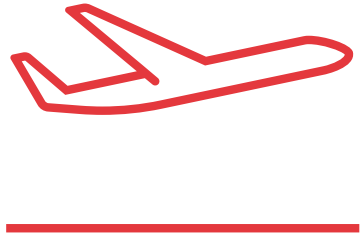
Safely flight-tests experimental control laws using envelope protections and runtime assurance

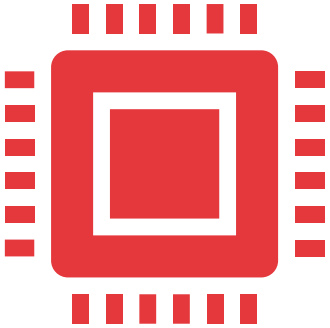
Blends baseline (proven) and experimental controllers, with rapid fallback to safe mode

My work is part of the Open Avionics Testbed for Flight Control Research, Rapid Integration, Validation and Technology Maturation project

Is the basis of establishing a HIL testbed to conduct advanced flight test research

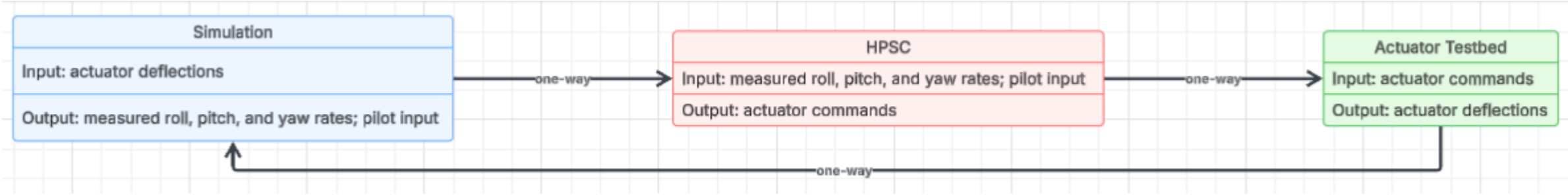
HIL Subproject





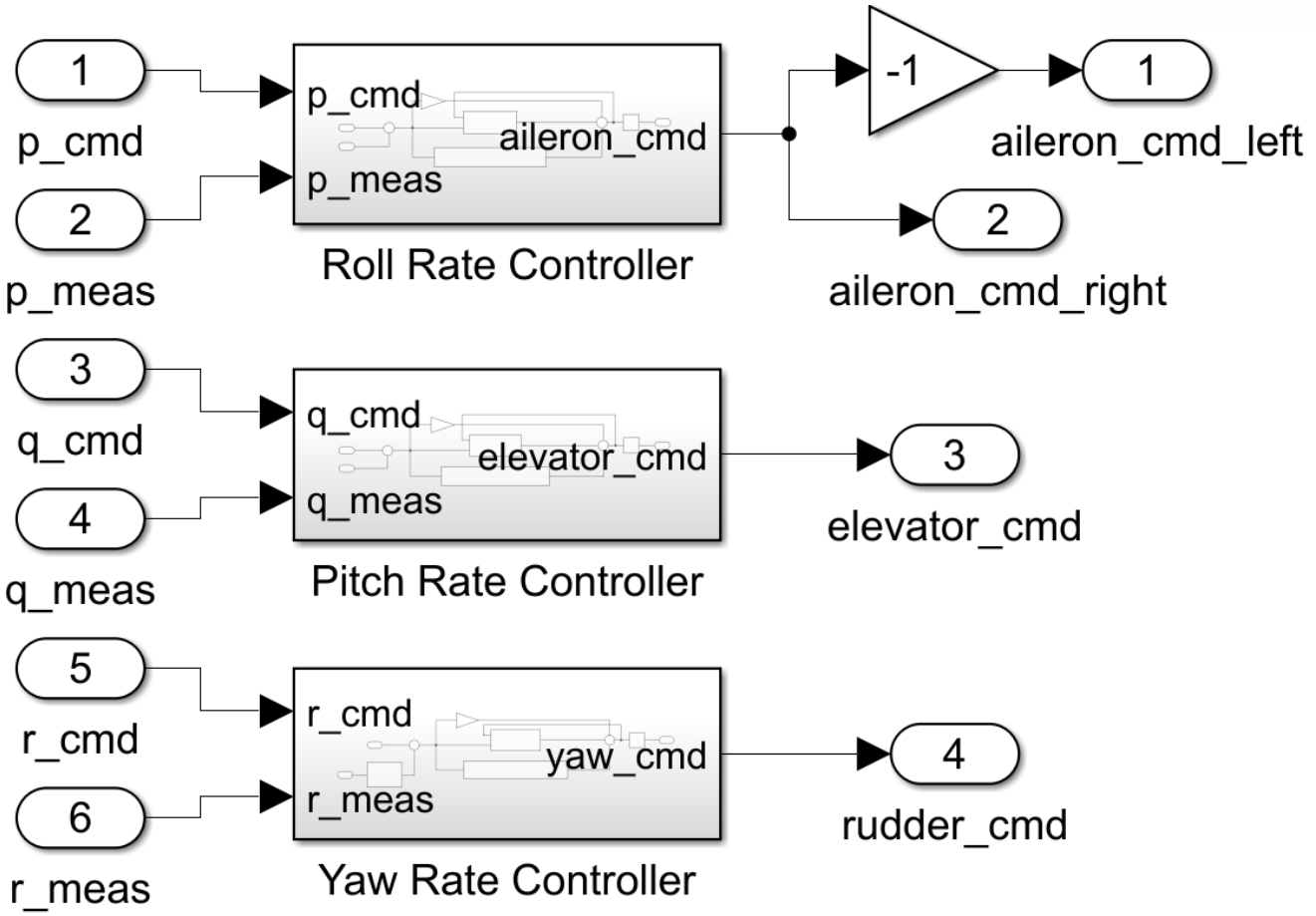
Before flight, test NuRFCS in simulation first

Plan: connect HPSC to a simulation, and connect a physical actuator testbed to HPSC



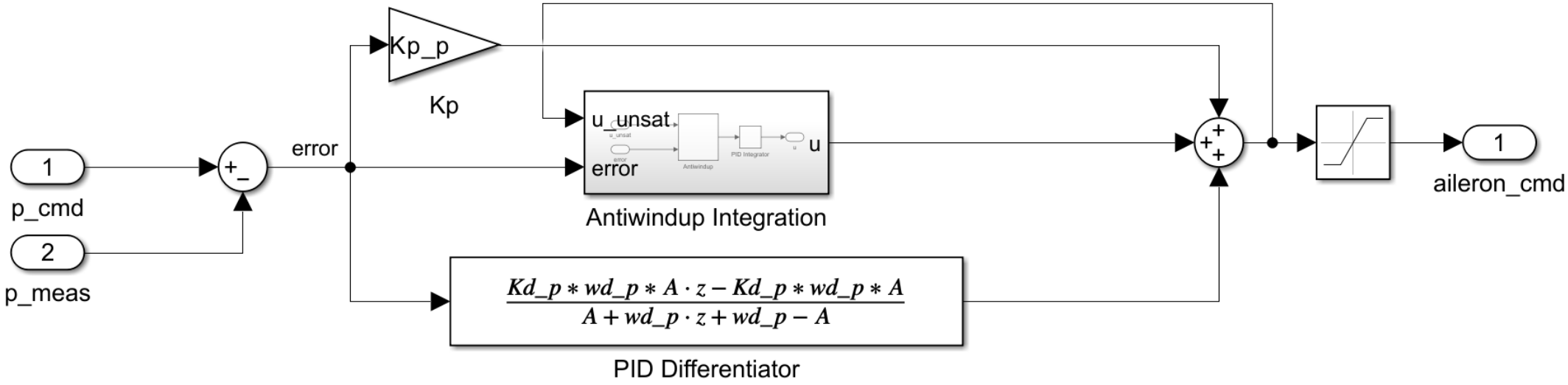
My Work

- Built a 4-actuator PID flight controller
- Input: measured and commanded roll, pitch, and yaw rates
- Output: aileron, elevator, and rudder deflections



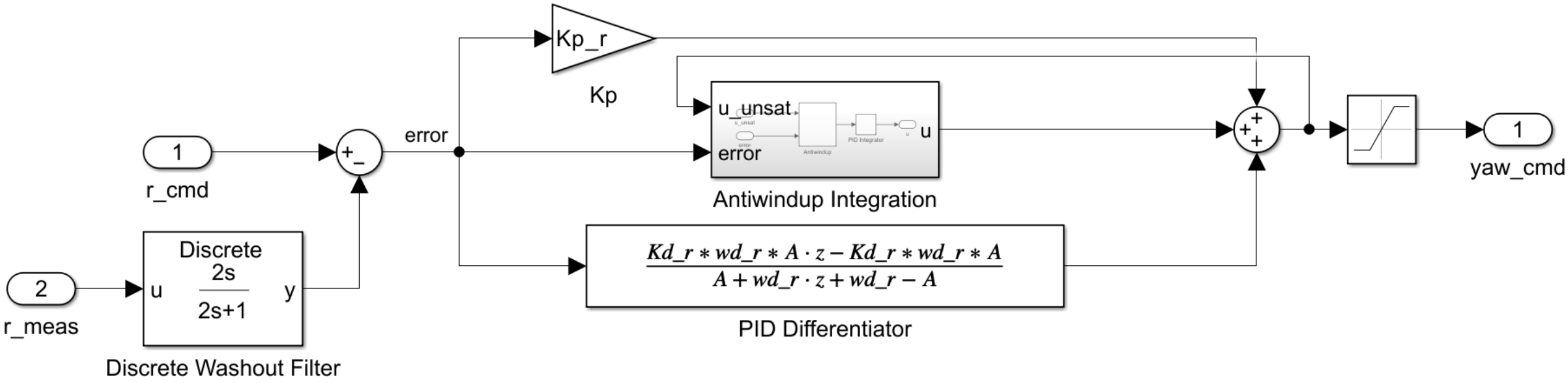
Roll (and Pitch) Rate Controllers

- Standard discrete PID controller
- Antiwindup: pauses integration when commanded to a rate limit, preventing overshoot from integral buildup
- Band-limited differentiator: extracts derivative info without amplifying high-frequency noise (D fades into P at high frequencies)
  - Continuous Form: $D(s) = K_d \frac{\omega_d s}{s+\omega_d}$
  - Discrete Form: $D(z) = K_d \frac{\omega_d A(z-1)}{(A+\omega_d)z+(\omega_d-A)}$



Yaw Rate Controller

- Same as pitch/roll controllers, plus a discrete washout filter on measured yaw
- Washout filter acts as a high-pass filter
  - Useful during banked turns: filters out low-frequency yaw from the bank while passing high-frequency disturbances to the controller



NuRFCS Summary

ARMSTRONG
FLIGHT RESEARCH CENTER

Built flight controller for initial HIL NuRFCS testing

Flight on a physical plane will require a tuned version – or a different architecture altogether

Current architecture suffices for validating the software pipeline

ARMSTRONG
FLIGHT RESEARCH CENTER

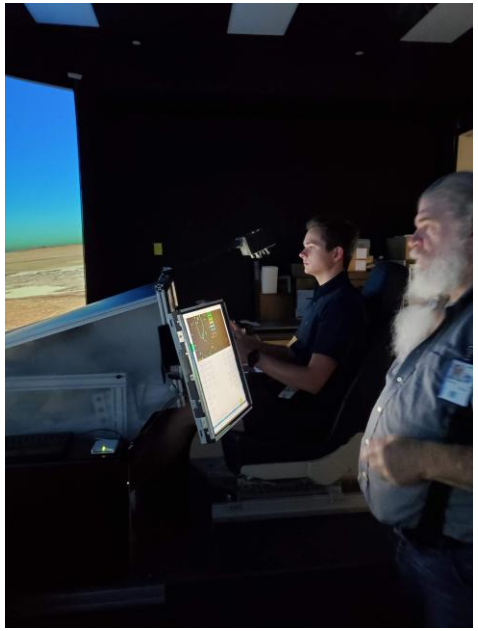
Overall Summary

- Created an actuator digital twin for fault detection and isolation
- Will assist in writing a paper on the methodology and findings
- Built a flight controller for initial NuRFCS HIL testing

ARMSTRONG
FLIGHT RESEARCH CENTER

Other Experiences

- Observed X-59 pre-flight and telemetry from TM Van
- Witnessed X-59 go supersonic
- Toured NASA JPL
- Toured Armstrong with family
- Flew the X-57 and F-18 simulators



Thank-You

ARMSTRONG
FLIGHT RESEARCH CENTER

John Baca – providing me the opportunity to experience Armstrong

Sarkis Mikaelian – being an excellent mentor and providing me with interesting work

Samuel Lemus – providing hardware expertise

Peter Suh – PMSM physics SME for digital twin help

Chris Miller – Controls SME for flight controller advice

Patrick Chan & Phillip Nguyen – Actuator Weights

Rebecca Porras – actuator testing rig

Cambria Parker – actuator arm MOI

Allen Parker – protractor

Joseph Piotrowski – hangar tour

Julio Trevino – global hawk tour

Manuel Castro – sims tour

Shideh Naderi – control room tour

Durand Nguyen – X-59 tour

Matthew Versteeg, Ernest Nwajagu, Ed Koshimoto, Jeffery Sutherland, Shedrick Bessent, David Spivey, Victor, & Angelo – TM Van & X-59 Meetings

Questions?