Auto-track Model with Error and Noise for Dishes (AMEND) Machine Learning Team

Undergraduate Team: Mikias Balkew, Adrija Bhattacharya, Tyler Cole, Shreyas Mhasawade, Chris Rothmann
Faculty Advisor: Dr. Xiaoli Ma

Background

- The number of Low Earth Orbit (LEO) satellites are increasing, which requires significant investment in ground stations for communications

Goal: Create a machine learning (ML) algorithm capable of generating reliable PID controller parameters for the Auto-tracking Control System designed to keep a ground station's parabolic dish pointed at a LEO satellite at all times.

Fig 1. Viasat 7m dish

Why: Minimize the amount of time between the acquisition of satellite position data and the execution of control system protocols for corrective actions to the ground station's servo motors, accounting for dynamic satellite data through automatic initialization of control parameters.

Normally controls are done with traditionally calibrated PID systems

Now implementing with machine learning techniques based on simulation databases

Method

- The existing model from last year will produce some representative data for chosen scenarios, and the data generated will create more accurate inputs for the PID controller by using machine learning techniques for analysis

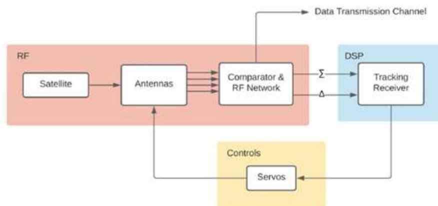

Fig 2. Block Diagram for the existing RF Model

- RMS error incorporates both signal overshoot and damping delay in its calculation and is thus optimal for ML training
- Classical control system design is used to generate control parameter data used to train the ML block

Create MATLAB database that will hold data points generated from the existing RF model

Develop method of extracting data from the RF model and transfer it into the ML algorithm

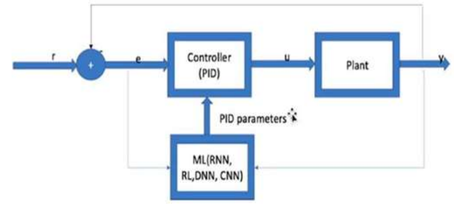

Fig 3. Block Diagram for proposed solution. The MATLAB code is used to generate data which feeds into the ML code which is run in Python

MATLAB Code

- The MATLAB script is utilized to generate different K1 and K2 values based on strategic eigenvalue placement
- Simulations are run with updated K1 & K2 parameters
- RMS of output angle and corresponding K1 & K2 are written to training data set
- RMS data serves as training data for the ML algorithm

Python Code

- The Python script trains the RNN algorithm by comparing the RMS error in the plant's output to that present in the training data
- An upper limit will be placed on the RMS error as a constraint on the output Kp and Ki values
- The Python code also graphs a predictive plot comparing the RMSE value of the training data against the test data

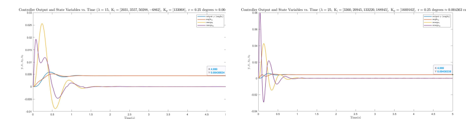

Fig 4. Graph of plant outputs as a function of time for two different lambda values

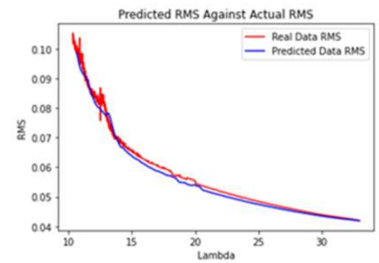

Fig 5. Graph showing predicted RMS plotted against the Lambda value to demonstrate improved RMS error through the ML algorithm

Future Work

- A derivative (D) block can be added in the Simulink to create a PID controller instead of the PI used in this project for higher accuracy
- A < 30% overshoot parameter can be added to the machine learning algorithm for better control

References

[1] Ziegler, J. G., and N. B. Nichols. "Optimum Settings for Automatic Controllers." Journal of Dynamic Systems, Measurement, and Control, vol. 115, no. 2B, 1993, pp. 220–222., https://doi.org/10.1115/1.2899060.

[2] Zulu, Andrew. "Towards Explicit PID Control Tuning Using Machine Learning." 2017 IEEE AFRICON, 2017, https://doi.org/10.1109/afrcon.2017.8095520.

[3] Davis, Zachary et al. "A Causal Model Approach to Dynamic Control." Cognitive Science (2018): 281-286.