**PI Controller Tuning and Stability Assessment App**

**Developed by Masoud Bakhshi**  
**Contact:** masoud.bakhshi@volvo.com

**Overview**

This app is a web-based tool designed for tuning and stability analysis of PI controllers in automotive electric machine control. Primarily focused on **d-axis and q-axis current control**, this app helps users achieve optimal gain settings for current controllers, either by initial tuning with Ziegler-Nichols or by re-tuning based on imported log data. Using stability analysis through Bode and Nyquist plots, it provides insights into controller stability, phase, and gain margins.

**Purpose and Use Cases**

The app is intended for professionals in automotive control systems, specifically those working with electric drives and current controllers in electric vehicles. It simplifies the process of tuning controllers on the **d and q axes**, which is critical for electric machine stability and performance. With this tool, users can:

1. **Optimize Initial Tuning**: For first-time controller setup using Ziegler-Nichols tuning.
2. **Refine Existing Settings**: Re-tune gains based on real-world data to improve controller response.
3. **Analyze Stability**: Assess controller stability using Nyquist and Bode plots, ensuring performance and safety.

This app is well-suited for engineers working on current controllers for Permanent Magnet Synchronous Machines (PMSMs), Induction Machines (IMs), and other electric machine types in the automotive industry.

**Key Features**

* **Dual-Axis Control**: Supports tuning for both d-axis and q-axis current controllers.
* **Two Tuning Modes**:
  + *First-Time Tuning*: Uses the Ziegler-Nichols method based on critical gains.
  + *Re-Tuning*: Optimizes gain values based on previously recorded data.
* **Detailed Stability Analysis**: Provides phase and gain margins with real-time Bode and Nyquist plots.
* **User-Friendly Interface**: Streamlit-based UI with accessible file upload, settings, and live plot updates.

**Libraries and Their Usage**

This app utilizes several key libraries to deliver its functionality:

**1. Streamlit**

* *Purpose*: Provides an interactive and web-based interface.
* *Usage*: Handles layout design, sidebar inputs, and dynamic plot rendering.
* *Example*: st.file\_uploader enables users to upload .mat files for imported data, and st.plotly\_chart displays real-time Bode and Nyquist plots.

**2. Pandas**

* *Purpose*: Data manipulation and storage.
* *Usage*: Stores extracted variables from .mat files and organizes them for plotting.
* *Example*: pd.DataFrame is used to create a structured table with reference and measured currents for easy analysis.

**3. NumPy**

* *Purpose*: Performs mathematical operations and array handling.
* *Usage*: Handles array operations, numerical calculations, and signal processing.
* *Example*: np.cumsum is used in calculating cumulative integral errors for controller tuning.

**4. Plotly**

* *Purpose*: Generates interactive plots (Bode, Nyquist, and current response plots).
* *Usage*: Plots stability analysis results, providing dynamic visualization of Bode and Nyquist plots.
* *Example*: plotly.graph\_objs enables flexible, interactive charts with customized formatting for visual clarity.

**5. SciPy (signal and optimize modules)**

* *Purpose*: Provides signal processing, optimization, and stability analysis.
* *Usage*: Defines PI controller transfer functions, finds peaks for Ziegler-Nichols method, and optimizes gains.
* *Key Functions*:
  + TransferFunction: Constructs the continuous-time transfer function for the PI controller.
  + bode: Computes the Bode plot for stability analysis.
  + find\_peaks: Finds peaks in oscillation data to determine the ultimate period (P\_u) in the Ziegler-Nichols method.
  + minimize: Optimizes Kp and Ki gains by minimizing RMS error, enhancing controller accuracy.

**How the App Works**

**Step 1: User Interface and Inputs**

1. **Logo and Title**: Displayed on the main page with contact details for easy identification.
2. **Instructions and Settings**: The sidebar provides instructions for tuning modes, sample time input, and controller critical gains.
   * **Sampling Time**: Accepts sampling time in seconds with high precision.
   * **Critical Gains**: Inputs for q-axis and d-axis gains for Ziegler-Nichols method.

**Step 2: Choosing Tuning Mode**

* **First-Time Tuning (Ziegler-Nichols)**:
  + Calculates Kp and Ki values based on critical gains (K\_u) and ultimate periods (P\_u) derived from oscillation data.
  + Determines phase margin and gain margin using Bode plots and assesses stability through Nyquist plots.
* **Re-Tuning with Imported Data**:
  + Users upload .mat files with real log data.
  + The app calculates optimal Kp and Ki values based on RMS error minimization, allowing refinement of controller response without re-entering initial gain values.

**Step 3: Data Upload and Plotting**

1. **Data Upload**: Users upload .mat files with required variable names for reference and measured currents on q and d axes.
2. **Plotting**:
   * **Current Response Plots**: Shows reference and measured currents on both q and d axes for visual comparison.
   * **Bode Plot**: Displays gain and phase response, providing phase and gain margins.
   * **Nyquist Plot**: Shows stability conclusion (Stable/Unstable) based on Nyquist criterion.

**Step 4: Stability Assessment**

* The app uses continuous transfer functions (TransferFunction) to generate the Bode and Nyquist plots.
* **Phase Margin and Gain Margin**:
  + Calculated from the Bode plot to determine controller responsiveness and robustness.
* **Stability Conclusion**:
  + Assessed via Nyquist plot using the Nyquist stability criterion, which checks the number of encirclements around the critical point (−1,0)(-1,0)(−1,0).