MTPA/MTPV Extraction of PMSM Dyno test data

The Python script is organised into **logical blocks** that correspond to the core processing steps in the MTPA/MTPV analysis pipeline. Below is an overview of each part of the code:

A. Imports (cell 1)

Library	Usage in the Script
`pandas` (`pd`)	Used for loading, cleaning, manipulating, grouping, and saving data in DataFrames.
`os`	Handles file paths, directory listings, and navigation across date folders.
`numpy` (`np`)	Performs numerical operations like computing power, efficiency, torque-to-current/voltage ratios, and generating ranges for grouping.
`nptdms.TdmsFile`	Reads `.tdms` files (LabVIEW format) and converts them into pandas DataFrames.
`sklearn.cluster.DBSCAN`	Applies DBSCAN clustering to filter out noisy `Id` and `Iq` current values per torque group.
`sklearn.preprocessing.StandardScaler`	Standardizes `Id` and `Iq` features before clustering, ensuring proper DBSCAN behavior.
`sklearn.neighbors.NearestNeighbors`	Used to compute optimal epsilon (`eps`) for DBSCAN by analyzing k-distance graph.
`plotly.express` (`px`)	Creates scatter plots and interactive visualizations such as `Torque vs Speed`, `Iq vs Id`, etc.
`plotly.graph_objects` (`go`)	Used for adding markers, annotations, and contour plots with custom styling and trace-level control.
`plotly.io` (`pio`)	Saves Plotly plots as HTML files without opening them immediately.
`scipy.interpolate.griddata`	Interpolates discrete efficiency values to create a smooth contour map over torque-speed space.

B. Configuration and Group Assignment functions (Cell 2)

Paths and flags like `USE_DBSCAN_FILTER` and `USE_MTPV_ANALYSIS` are declared.

Grouping thresholds for **torque** and **speed** are set.

Voltage thresholds are computed from DC bus voltage (`Vdc`) for filtering.

Two helper functions:

- `assign_torque_group(torque)`: Assigns each row to a torque bin
  `assign_speed_group(speed)`: Similar, for speed bins (used only for MTPV).

These allow downstream analysis to operate on grouped data.

Key configuration parameters:

Parameter	Description
`USE_MTPV_ANALYSIS`	Set to `True` for MTPV, `False` for MTPA
`USE_DBSCAN_FILTER`	Enable or disable noise filtering via clustering
`torque_centers`	Defines bin centers for torque grouping (e.g., 0 to 45 Nm in steps of 5)
`speed_centers`	Defines bin centers for speed grouping (e.g., 3000 to 6500 RPM in steps of 500)
`Vdc`	DC link voltage used to define stator voltage threshold

C. TDMS File Processing, Filtering, and Grouping (Cell 3)

1. **Initial Setup**:
   - An empty list `final_grouped` is initialised to store filtered data from each date.
2. **Iterate Through Date Folders**:
   - For each selected folder (based on test dates), all `.tdms` files are scanned and processed.
3. **Read & Convert TDMS to DataFrame**:
   - Each `.tdms` file is read using `TdmsFile.read()` and converted to a pandas DataFrame.
   - Column names are cleaned and formatting adjusted for consistency.
4. **Check for Required Columns**:
   - The script proceeds only if all required sensor channels and computed signals are present in the file.
5. **Filter Raw Data**:
   - This is the **primary location for all data filtering**.
   - Filters include:
     - Removing zero or invalid reference commands.
     - Ignoring unphysical operating points (e.g., negative or reversed currents).
     - Speed stability across rows.
     - Matching specific motor parameters (Ld, Lq).
     - Ensuring consistency between reference and actual current values.

- **Note**: Any new condition for filtering should be added here.

6. **Group Data**:
    - After filtering, data is grouped by **torque levels**.
    - If MTPV mode is enabled, data is also grouped by **speed ranges**.
7. **Mode-Specific Filtering**:
    - Based on the mode (MTPA or MTPV), filtering on stator voltage is performed using a threshold.
8. **Compute Key Metrics**:
    - Power output (`Pout`) and input (`Pin`) are calculated.
    - Efficiency is computed and invalid values are removed.
    - Depending on mode, either **torque per amp** or **torque per volt** is calculated.
9. **Save Per-Date Processed CSV**:
    - The cleaned and enriched dataset is saved for each date for traceability.
10. **Append for Global Use**:

- Each date's processed dataset is added to the `final_grouped` list for global analysis in later cells.

D. Merging, Outlier Removal (DBSCAN), and Optimal Point Extraction (Cell 4)

This part consolidates all processed data into a single master dataset and applies **adaptive DBSCAN filtering** to remove outliers based on current values (`Id_in`, `Iq_in`) within each torque group.

Key operations:

1. **Combining Data**:
   All per-date grouped DataFrames from the previous cell are concatenated into a single DataFrame `all_grouped_df`.
2. **Outlier Filtering Using DBSCAN**:
   If enabled via the `USE_DBSCAN_FILTER` flag, each torque group is clustered using the DBSCAN algorithm with adaptive `eps` values (based on the 90th percentile of nearest-neighbor distances).
    - Only clusters (non-outliers) are retained.
    - This step helps discard noisy measurements that could skew the final analysis.

3. **Save Master Dataset**:
   The resulting filtered dataset is saved to a CSV file for reference and reuse.

4. **Optimal Point Selection – MTPA or MTPV**:
   Depending on the analysis mode:

   - **MTPV**: Maximum torque per volt (`Torque / Vstator`) is selected **per torque and speed group**.
   - **MTPA**: Maximum torque per ampere (`Torque / Istator`) is selected **per torque group**.
   - The selected optimal operating points are stored in a separate CSV file (`MTPA.csv` or `MTPV.csv`).

This cell marks the completion of the data cleaning and optimisation phase. The output is now ready for visualisation or further analysis.

E. Visualization and Plot Generation (Cell 5)

This cell creates interactive **visualisations** to help analyse and interpret the filtered and grouped dataset. All plots are exported as standalone HTML files for ease of sharing and viewing.

📊 Key Visualisations:

1. **Torque vs Speed Scatter Plot**
   - Shows the entire operational range of the motor.
   - Hover reveals the corresponding `Id_in` and `Iq_in` values at each operating point.
   - Useful to see clustering of data across the speed-torque envelope.
2. **Iq_in vs Id_in Scatter Plot**
   - A simple current vector map to visualise the input current distribution.
3. **Iq_in vs Id_in Colored by Torque Group with Optimal Points**
   - Each torque group is colour-coded.
   - Optimal MTPA or MTPV points are overlaid as 'X' markers.
   - An annotation box summarises optimal `T/A` or `T/V` ratios for each group.
4. **(Optional)**: Iq vs Id for Specific Torque Group
   - Commented section that can be enabled for focused analysis on a single torque group (e.g., 5 Nm).
5. **Efficiency Contour Plot: Torque vs Speed**
   - Interpolated contour of efficiency across the torque-speed plane.

    - Helps visualise how efficiency varies under different operating conditions.

📁 **Output:**

All plots are saved as HTML files locally. These files can be opened in any browser without the need for additional software or a Python environment.