

# **Order Pattern Matching for ADHD analysis using EEG waves Maintenance Guide**

**25-1-R-1**

Lecturers Guide:

Dr. Anat Dahan

Dr. Samah Idrees Ghazawi

Students Submit:

Tamir Nahari

Yaakov Shitrit

# Maintenance Guide - Approach 1

## System Overview

### Maintenance Guide – Approach 1

The **Order Pattern Matching (OPM)** system is a computational framework developed to support the discovery of temporal patterns in EEG waveforms for the purpose of ADHD diagnosis and analysis. This guide focuses on **Approach 1**, which emphasizes the **structural properties of the raw EEG signal itself**, using a combination of filtering, decomposition, and symbolic encoding to identify recurring motifs in brain activity.

### Purpose

The primary objective of this system is to detect repeating waveform structures—such as trend reversals and rhythmic cycles—within independent components (ICs) derived from raw EEG recordings. These motifs are analyzed and compared across groups (ADHD vs. Control) to identify meaningful diagnostic features that may correlate with neurophysiological characteristics of ADHD.

### Architecture Summary

Approach 1 relies on the following core components:

- **Preprocessing Pipeline:** Includes filtering steps such as CAR (Common Average Referencing), harmonic notch removal, and bandpass filtering (e.g., theta band). These ensure noise removal while preserving structural features of interest.
- **AMICA ICA Decomposition:** Separates EEG signals into statistically independent components, improving the ability to localize distinct neural sources.
- **Cycle Segmentation:** Applies Hilbert transform and trend analysis to segment the EEG signal into cycles with identifiable start/end points.
- **Motif Discovery (OPM):** Converts each segmented cycle into a symbolic representation based on its temporal structure and frequency characteristics. Similar cycles are grouped and stored as motifs.
- **Pattern Analysis:** Each motif is annotated with its distribution across ADHD and control subjects, allowing researchers to study group-specific differences.

### Data Flow

1. **Input:** Clean EEG recordings in `.mat` or `.fdt` format.
2. **Processing:** Sequential application of AMICA, signal filtering, cycle detection, and pattern cataloging.

3. **Output:** Parquet and CSV files containing motif metrics, per-subject motif appearances, and group-level statistical summaries.

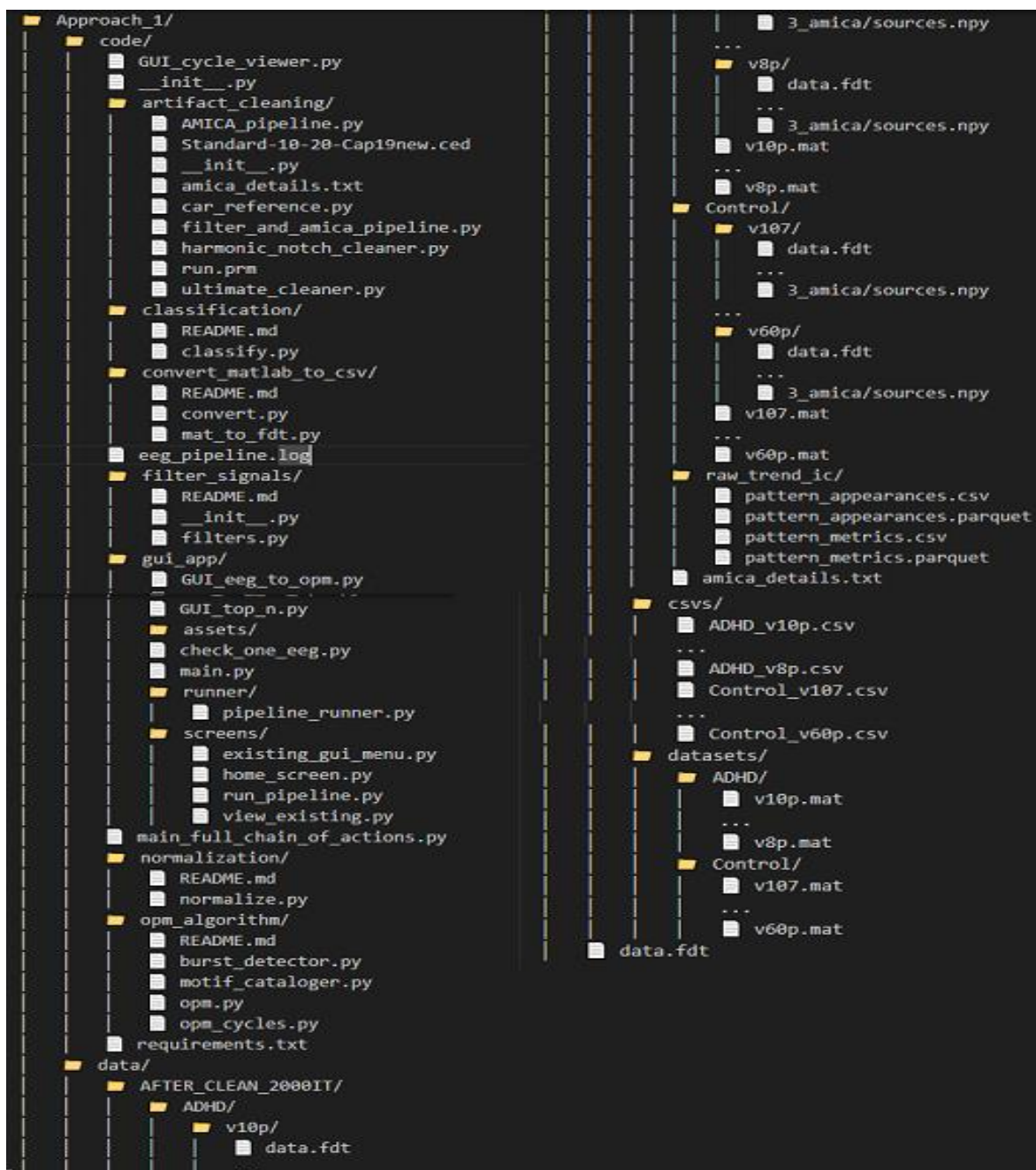
## Use Case

This system is intended for EEG researchers and developers working on diagnostic tools for ADHD. By matching waveform shapes rather than traditional spectral metrics alone, it offers a novel path for symbolic pattern analysis, potentially revealing overlooked biomarkers.

## Target Audience

This Maintenance Guide is intended for **developers, signal processing engineers, and data scientists** maintaining or extending the OPM pipeline. It assumes familiarity with Python, signal processing concepts (e.g., ICA, bandpass filters), and EEG data handling.

## Directory & File Structure for: \OPM4EEG



# Installation Requirements

Below is a suggested **Installation Requirements** section you can drop into your Maintenance Guide—written in English to match your existing document—and four brief lead-ins you can place immediately before each screenshot.

---

## Installation Requirements

Before you can run Approach 1, make sure you have:

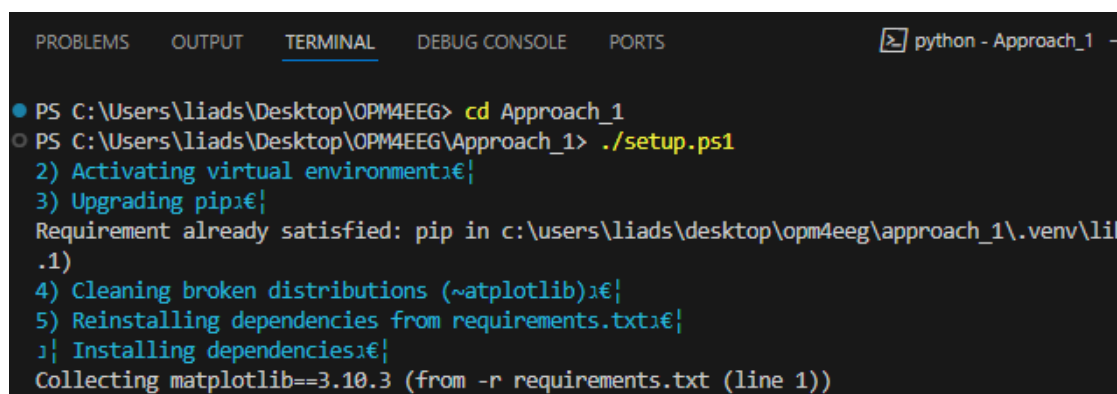
- **Operating System:** Windows 10 or 11.
- **PowerShell:** with an execution policy that allows local scripts (e.g. `RemoteSigned`).
- **Python:** version 3.8 or above.
- **Visual Studio Code:** installed for development and debugging.
- **External tools:**
  - A copy of the AMICA binary (`amica15mk1.exe`) under `Approach_1\amica`.
  - The Octave-portable ZIP (`octave-10.2.0-w64.zip`) under `tools\octave-portable`.

All remaining dependencies and configurations are handled automatically by our bootstrap script.

---

### 1) Run the bootstrap script

```
.\setup.ps1
```



```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE  PORTS  python - Approach_1
PS C:\Users\liads\Desktop\OPM4EEG> cd Approach_1
PS C:\Users\liads\Desktop\OPM4EEG\Approach_1> ./setup.ps1
2) Activating virtual environment!
3) Upgrading pip!
Requirement already satisfied: pip in c:\users\liads\desktop\opm4eeg\approach_1\.venv\li
.1)
4) Cleaning broken distributions (~matplotlib)!
5) Reinstalling dependencies from requirements.txt!
! Installing dependencies!
Collecting matplotlib==3.10.3 (from -r requirements.txt (line 1))
```

This will:

- Create & activate a `.venv` virtual environment
- Upgrade `pip`
- Reinstall every package from `requirements.txt` (Matplotlib, NumPy, Pandas, SciPy, tqdm, PyArrow, PyQt6, PyQtGraph, MNE, etc.)

- Extract and prune the Octave portable distribution
- Copy and verify AMICA binaries
- Set environment variables for AMICA and Octave
- Configure VSCode's Python interpreter to point at  
`.venv\Scripts\python.exe`

*Place screenshot here: PowerShell running `setup.ps1` with progress messages.*

---

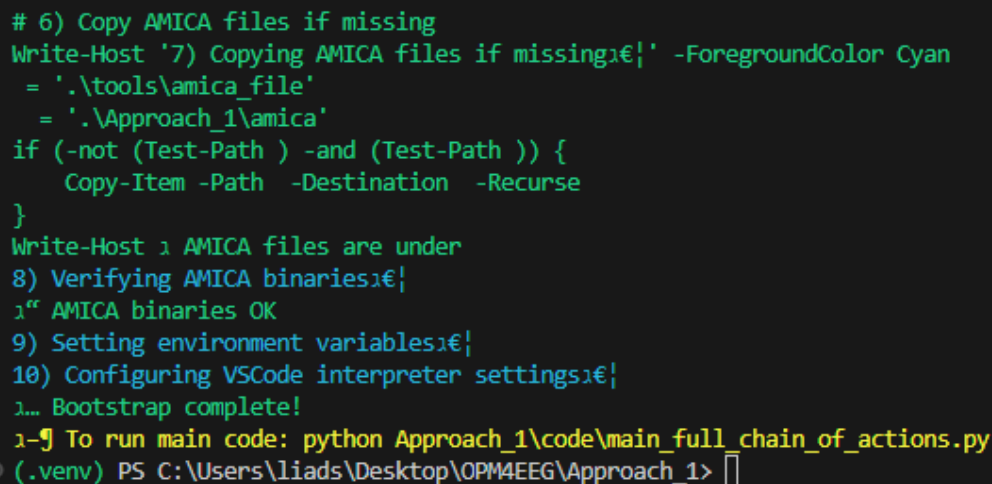
## 2) Confirm bootstrap completion

When everything finishes without errors, you'll see:

✓ Bootstrap complete!

► To run main code: `python`

`Approach_1\code\main_full_chain_of_actions.py`



```
# 6) Copy AMICA files if missing
Write-Host '7) Copying AMICA files if missing' -ForegroundColor Cyan
$amica_file = '.\tools\amica_file'
$amica_dir = '.\Approach_1\amica'
if (-not (Test-Path $amica_dir) -and (Test-Path $amica_file)) {
    Copy-Item -Path $amica_file -Destination $amica_dir -Recurse
}
Write-Host '7) AMICA files are under'
8) Verifying AMICA binaries
$amica_binaries = 'AMICA binaries OK'
9) Setting environment variables
10) Configuring VSCode interpreter settings
... Bootstrap complete!
1-9 To run main code: python Approach_1\code\main_full_chain_of_actions.py
(.venv) PS C:\Users\liads\Desktop\OPM4EEG\Approach_1>
```

---

## 4) Launch the main pipeline

Finally, start the OPM workflow by running:

```
python Approach_1\code\main_full_chain_of_actions.py
```

You'll be presented with the menu of OPM methods (`raw_full_ic`, `hilbert_trend_ch`, etc.).

*Place screenshot here: terminal showing `python main_full_chain_of_actions.py` and menu prompt.*

---

## Environment Configuration

What's set for you by `setup.ps1`:

- `AMICA_DIR` & `AMICA_MATLAB_DIR` → `tools\amica_file`
- `OCTAVE_CMD` → path to the Octave-CLI inside `tools\octave-portable`

- **python.defaultInterpreterPath** → your `.venv\Scripts\python.exe` in VSCode

If you ever need to point to a different AMICA build, or use a system-wide Octave, override these in your shell (or in a custom PowerShell wrapper) before running the pipeline.

## Execution Methods

*Make sure you have activated the .venv (via `.\.venv\Scripts\Activate.ps1`) before running either option.*

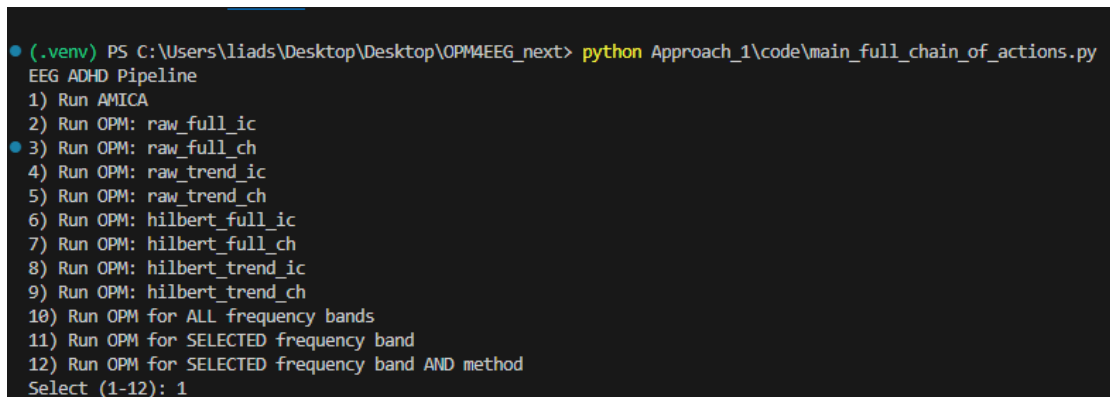
### 4.1 CLI (Recommended)

From the project root, launch the full pipeline menu:

```
powershell
CopyEdit
python Approach_1\code\main_full_chain_of_actions.py
```

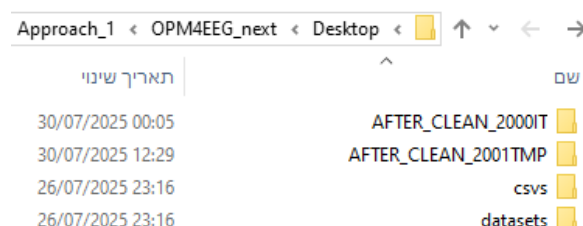
**What happens:**

- You'll see a numbered list of 12 OPM methods (raw\_full\_ic, hilbert\_trend\_ch, etc.)



```
(.venv) PS C:\Users\liads\Desktop\Desktop\OPM4EEG_next> python Approach_1\code\main_full_chain_of_actions.py
EEG ADHD Pipeline
1) Run AMICA
2) Run OPM: raw_full_ic
3) Run OPM: raw_full_ch
4) Run OPM: raw_trend_ic
5) Run OPM: raw_trend_ch
6) Run OPM: hilbert_full_ic
7) Run OPM: hilbert_full_ch
8) Run OPM: hilbert_trend_ic
9) Run OPM: hilbert_trend_ch
10) Run OPM for ALL frequency bands
11) Run OPM for SELECTED frequency band
12) Run OPM for SELECTED frequency band AND method
Select (1-12): 1
```

- The script auto-detects or resumes the existing clean data folder (`data\AFTER_CLEAN_<n>IT`)



- AMICA decomposition is resumed with a progress bar (iterations, IC/s) and runs OPM on raw or Hilbert-transformed signals at IC or channel level

```
(.venv) PS C:\Users\liads\Desktop\Desktop\OPM4EEG_next> python Approach_1\code\main_full_chain_of_actions.py
EEG ADHD Pipeline
1) Run AMICA
2) Run OPM: raw_full_ic
3) Run OPM: raw_full_ch
4) Run OPM: raw_trend_ic
5) Run OPM: raw_trend_ch
6) Run OPM: hilbert_full_ic
7) Run OPM: hilbert_full_ch
8) Run OPM: hilbert_trend_ic
9) Run OPM: hilbert_trend_ch
10) Run OPM for ALL frequency bands
11) Run OPM for SELECTED frequency band
12) Run OPM for SELECTED frequency band AND method
Select (1-12): 1
[INFO] Using cleaned data: AFTER_CLEAN_2000IT
Iters [2000]: 2001
Copying previous version C:\Users\liads\Desktop\Desktop\OPM4EEG_next\Approach_1\data\AFTER_CLEAN_2000IT → C:\
Users\liads\Desktop\Desktop\OPM4EEG_next\Approach_1\data\AFTER_CLEAN_2001TMP
Skipping filtering, reusing cleaned data.

Resuming AMICA (+1 iters) on C:\Users\liads\Desktop\Desktop\OPM4EEG_next\Approach_1\data\AFTER_CLEAN_2001TMP.
..

Resuming AMICA subjects: 26% | 31/121 [00:42<01:02, 1.45it/s]
```

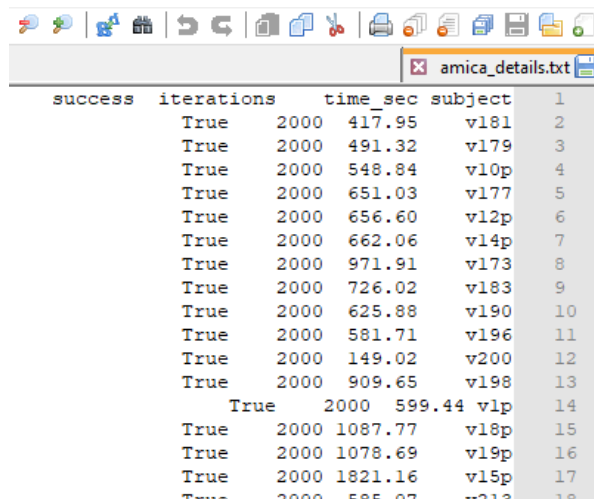
- If requested iters  $\leq$  existing, filtering is skipped to save time

```
(.venv) PS C:\Users\liads\Desktop\Desktop\OPM4EEG_next> python Approach_1\code\main_full_chain_of_actions.py
EEG ADHD Pipeline
1) Run AMICA
2) Run OPM: raw_full_ic
3) Run OPM: raw_full_ch
4) Run OPM: raw_trend_ic
5) Run OPM: raw_trend_ch
6) Run OPM: hilbert_full_ic
7) Run OPM: hilbert_full_ch
8) Run OPM: hilbert_trend_ic
9) Run OPM: hilbert_trend_ch
10) Run OPM for ALL frequency bands
11) Run OPM for SELECTED frequency band
12) Run OPM for SELECTED frequency band AND method
Select (1-12): 1
[INFO] Using cleaned data: AFTER_CLEAN_2000IT
Iters [2000]: 2000
Requested (2000)  $\leq$  existing (2000), skipping everything.
All done. CSV files are alongside Parquet outputs.
(.venv) PS C:\Users\liads\Desktop\Desktop\OPM4EEG_next>
```

- Writes both Parquet and CSV outputs for pattern\_metrics & pattern\_appearances

גודל	סוג	תאריך שינוי	שם
128,122 KB	Microsoft Excel C...	28/07/2025 21:50	pattern_appearances
7,866 KB	קובץ PARQUET	28/07/2025 21:51	pattern_appearances.parquet
15 KB	Microsoft Excel C...	28/07/2025 21:50	pattern_metrics
14 KB	קובץ PARQUET	28/07/2025 21:50	pattern_metrics.parquet

- Saves detailed AMICA logs to  
...\\data\\AFTER\_CLEAN\_<n>IT\\amica\_details.txt



success	iterations	time_sec	subject	
				1
True	2000	417.95	v181	2
True	2000	491.32	v179	3
True	2000	548.84	v10p	4
True	2000	651.03	v177	5
True	2000	656.60	v12p	6
True	2000	662.06	v14p	7
True	2000	971.91	v173	8
True	2000	726.02	v183	9
True	2000	625.88	v190	10
True	2000	581.71	v196	11
True	2000	149.02	v200	12
True	2000	909.65	v198	13
True	2000	599.44	v1p	14
True	2000	1087.77	v18p	15
True	2000	1078.69	v19p	16
True	2000	1821.16	v15p	17
True	2000	585.07	v11p	18

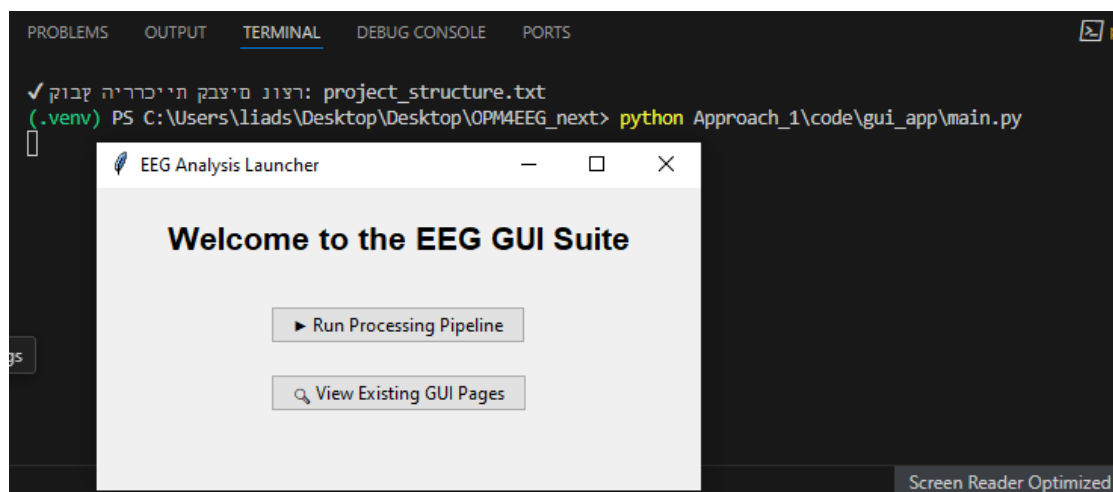
## 4.2 GUI (Optional)

If you prefer an interactive explorer, run the cycle-viewer GUI:

```
powershell
CopyEdit
python Approach_1\\code\\gui_app\\main.py
```

### Features:

- Pick a subject folder, then choose an IC or channel and a discovered motif
- See raw vs. filtered waveforms side-by-side
- Navigate forwards/backwards through all pattern occurrences
- Zoom, pan and export snapshots or CSV snippets for any cycle
- Built on PyQt6 + PyQtGraph for high-performance plotting





## Parameter Configuration

All key constants live in the “==== CONFIG ====” block at the top of `main_full_chain_of_actions.py`. You can tweak them there (or extend the CLI) to change default behavior.

### General pipeline settings

- **RAW\_DIR\_DEFAULT**: location of raw EEG files (`data/datasets`).
- **BASE\_OUT\_DEFAULT**: base output folder for cleaned data and OPM results (`data`).
- **DEFAULT\_ITERS**: number of AMICA iterations to run (default: 2000).
- **FS**: sampling frequency in Hz (default: 128.0).
- **FLUSH EVERY**: how many patterns to process before writing metrics back to disk (default: 1000).

### OPM-specific parameters (passed into `run_opm_generic`)

- **pre\_band**: initial filtering band, as (low, high) in Hz (default: (1.0, 40.0)).
- **theta\_band**: band for Hilbert-based burst detection (default: (4.0, 8.0)).
- **min\_cycles**: minimum number of theta cycles to qualify as a burst (default: 3).
- **pattern\_len**: length (in samples) of each sliding window used for OPM (default: 5).
- **flush\_every**: number of windows to process before flushing each pattern’s results (default: 1000).

## Maintenance Tasks

*These steps help you keep your Approach 1 pipeline running smoothly and up-to-date.*

### 1. Re-running the full analysis

- Before you start, delete or rename any existing clean-data folder under `data/AFTER_CLEAN_*IT/`
- Then launch

```
powershell
CopyEdit
python Approach_1\code\main_full_chain_of_actions.py
```

to perform ICA, filtering and OPM from scratch.

### 2. Partial reruns / resuming AMICA

- If you only need to resume AMICA with more iterations, leave your current `AFTER_CLEAN_XXXXIT/` folder in place.
- At the menu prompt, choose **1) Run AMICA** again and enter a higher iteration count.
- The script will copy the previous folder, skip filtering, and resume ICA.

### 3. Adding new EEG recordings

- Place your new `.mat` files into:

```
bash
CopyEdit
data/datasets/ADHD/      or      data/datasets/Control/
```

- Rerun the pipeline (full or partial) to process only the new subjects.
- 4. **Validating and inspecting results**
  - **CLI:** after OPM finishes, open the Parquet/CSV outputs in `data/AFTER_CLEAN_<n>IT/raw_full_ic/` (or whichever method you ran).
  - **GUI:**

```
powershell
CopyEdit
python Approach_1\code\gui_app\main.py
```

– browse individual motifs, compare raw vs. filtered cycles, and navigate through appearances.

- 5. **Checking AMICA convergence logs**
  - In each clean-data folder you'll find

```
CopyEdit
amica_details.txt
```

– review it for per-subject iteration counts and any convergence warnings.

- 6. **Updating dependencies or tools**
  - If you change `requirements.txt` (new Python packages or versions), rerun:

```
powershell
CopyEdit
.\setup.ps1
```

to rebuild the virtual environment and reinstall everything.

- To update AMICA or Octave, replace the binaries under `tools\amica_file` or `tools\octave-portable` and rerun the bootstrap.
- 7. **Troubleshooting common issues**
  - **Stale outputs:** delete `AFTER_CLEAN_*IT` and rerun.
  - **Virtual-env corruption:** remove the `.venv` folder and run `.\setup.ps1` to recreate it.
  - **Missing files:** verify that
    - your `.mat` inputs are well-formed (2D arrays)
    - `amica15mkl.exe` exists under `tools\amica_file`
    - VSCode's interpreter is pointing to `.venv\Scripts\python.exe`

---

## Logging & Debugging

- `eeg_pipeline.log` in `code/` captures high-level pipeline steps.

- `amica_details.txt` in each `data/AFTER_CLEAN_<n>IT/` shows per-subject AMICA iterations & warnings.
- To increase verbosity, open `filter_and_amica_pipeline.py` and adjust the logger from `INFO` to `DEBUG`.

Together, these let you trace every filtering, ICA fallback and motif-discovery step when troubleshooting.

## Updating or Extending the System

- **Supporting new EEG groups**

Simply drop your new `.mat` files into

```
bash
CopyEdit
data/datasets/YourNewGroup/
```

and both the CLI menu and the GUI will pick it up automatically as another subject group.

- **Advanced pattern inspection (and ICA convergence logs)**

Launch the GUI:

```
powershell
CopyEdit
python Approach_1\code\gui_app\main.py
```

– Browse every detected motif, compare raw vs. filtered cycles, navigate forwards/backwards through all occurrences...

– **Plus:** from the same clean-data folder (`data/AFTER_CLEAN_<n>IT/`) you can open `amica_details.txt` to see per-subject AMICA iteration counts and convergence warnings—handy for debugging your ICA step.

# Maintenance Guide - Approach 2

**Version:** 1.0

**Last Updated:** July 2025

**Maintainers:** Tamir Nahari, Yaakov Shitrit

## 1. System Overview

This system is designed to detect and compare order-based EEG patterns between individuals with ADHD and a control group. The core algorithm used is Order-Preserving Matching (OPM), applied on dominant EEG frequency sequences.

The full pipeline includes:

- Data conversion (.mat → .npz)
- EEG cleaning (filtering, ICA, normalization)
- Window segmentation
- Frequency and band extraction
- Pattern matching using OPM
- Cross-group comparison
- Visual outputs and logging

Execution is supported via both GUI and CLI.

## 2. Directory & File Structure

```
├── src/           # Source code and pipeline logic
├── validation/    # Analysis and visual validation scripts
├── results/       # Output folders (auto-created)
│   ├── opm_runs/  # Raw OPM pattern outputs
│   ├── cross_group_comparison/ # ADHD vs Control comparison files
│   ├── summary_plots/ # CSV summaries per window size
│   └── single_run_plot/ # Top-10 pattern visualizations
├── converted_npy/ # EEG files converted from .mat
├── cleaned_npy/   # EEG files after ICA cleaning
├── windowed_npy_1s_50/ # Segmented EEG windows (1 sec, 50% overlap)
└── logs/          # Runtime logs
```

└─ requirements.txt      # Python packages  
└─ README.md            # General project overview

Intermediate folders are automatically generated per run. No manual creation is needed.

### 3. Installation Requirements

- **Python:** Version 3.8 or above
- **Packages:** Install via pip:

```
pip install -r requirements.txt
```

- **OS:** Tested on Windows 10/11.

### 4. Execution Methods

#### 4.1 GUI (Recommended)

```
python -m src.gui_launcher
```

Features:

- Full pipeline control
- Real-time progress
- Log monitoring
- Parameter input (window size, pattern size, thresholds)
- Validation tools (PSD, ICA, band-label verification)

#### 4.2 CLI (Advanced Users)

```
python -m src.main
```

Prompts for manual or default parameter entry.

### 5. Parameter Configuration

Key parameters:

- `window_sec`: EEG segment duration in seconds (default: 1)
- `overlap_percent`: % overlap between windows (default: 50)
- `pattern_size`: Number of elements in OPM pattern (default: 5)
- `match_threshold`: Pattern similarity required (default: 0.8)
- `subject_threshold`: % of subjects in base group (default: 0.8)

Parameters are set via GUI or passed to CLI. Also adjustable in `pipeline_runner.py`.

## 6. Maintenance Tasks

- **Re-running analysis:** Clear the results/ folder if reprocessing is required.
- **Adding new data:** Place .mat files into data/ADHD/ or data/CONTROL/.
- **Validating results:** Use GUI tools or scripts in validation/.
- **Partial reruns:** Use skip\_conversion=True in CLI or GUI to avoid repeating conversion/cleaning.

## 7. Error Handling & Troubleshooting

- **ICA Warnings:** Sometimes ICA does not fully separate components. This is not critical for OPM output.
- **Shape Mismatch:** Ensure all EEG files are structured as 2D NumPy arrays with shape (samples, channels).
- **Missing outputs:** Check results/ for expected files. If missing, rerun the stage manually.

## 8. Updating or Extending the System

- **Adding new validation scripts:** Place inside validation/ and add a button to gui\_launcher.py.
- **Supporting new EEG groups:** Add folder under data/ and update group name in GUI/CLI.
- **Changing pipeline logic:** Use pipeline\_runner.py for class-based modular expansion.
- **Enabling grid search for hyperparameters:**  
Inside src/run\_opm\_pattern\_search.py, there's a commented-out section at the end of the file which allows automatic execution of multiple OPM configurations across combinations of pattern\_size, match\_threshold, and subject\_threshold.  
This feature allows users to run multiple combinations in one long run, instead of waiting for each to complete individually. To enable this, simply uncomment the grid run block and adjust the parameter ranges as needed.
- To view which ICA components were removed, uncomment the disabled print line in the EEG cleaning file (preprocess\_clean\_eeg.py).

## 9. Appendix: Key Outputs Summary

Folder/File	Description	Source Script
opm_runs/	Patterns per group	run_opm_pattern_search.py
cross_group_comparison/	ADHD vs CONTROL comparison	cross_group_pattern_check.py
summary_plots/	Top patterns CSVs	extract_top_patterns.py
single_run_plot/	Top patterns bar chart	generate_single_run_plot.py
*_freqs.npy	Dominant frequencies	extract_dominant_frequencies.py
*_labels.npy	EEG band labels	extract_dominant_frequencies.py

## 10. Future Improvements

Several enhancements can be considered to further improve the system:

- Visual GUI Upgrade:
  - Improve the visual design of the GUI using a modern UI toolkit.
  - Add graphical icons and tooltips for better user experience.
- More Visualization Options:
  - Add buttons to visualize specific EEG channels over time.
  - Enable exporting visual outputs to PDF reports directly from the GUI.
- Advanced Pattern Inspection:
  - Add GUI functionality to inspect individual patterns detected by OPM.
  - Allow users to select and visualize where a specific pattern appears across subjects.
- Real-Time Analysis:
  - Explore the option to integrate real-time EEG data stream input for online pattern detection.

These features can significantly improve usability, insight, and adaptability of the system for future research or clinical applications.