Table of Contents I

notes

- this is the table of contents of this document; each section corresponds to a specific part of the course

Filtering Lab: Noise Reduction and Outlier Rejection

notes

Contents map

developed content units	taxonomy levels
Filter design and performance evaluation	u1, e1
Practical trade-offs (noise vs. outliers)	u2, e2
Python implementation of filters	u3, e3

prerequisite content units	taxonomy levels
Sinusoidal fidelity, LTI systems	u1, e1
Basic Python/NumPy/Matplotlib	u2, e2

notes

Main ILO of sub-module "Filtering Lab: Noise Reduction and Outlier Rejection"

Choose the right filter for a given signal (noise vs. outliers)

Quantify filter performance (e.g., frequency distortion, outlier rejection)

Understand practical implications of impulse/frequency responses

notes

- Focus on decision-making, not just implementation

Todays Plan

- **Task 1**: Tweak Butterworth parameters (order, cutoff) and observe effects
- **Task 2**: Compare frequency content (original vs. filtered)
- **Task 3**: Visualize impulse/frequency responses of LTI filters
- **Task 4**: Smooth signals using moving average, median, or Savitzky-Golay

notes

- Hands-on: Use Python to explore, not just watch
- Signals: EMG, noisy sine/square waves, quantized data

Signals Youll Work With

Real-world:

- EMG (biceps movement)
- Heavily quantized time series

Synthetic:

- Sine/square waves + noise
- Same + outliers (spikes)

Goal: Match the filter to the problem!

notes

- Outliers noise: Different filters needed.

Lab time!

`FilteringLab.ipynb`

notes

- download the file in STEM, open it, and we'll continue from there!

Summarizing

Choose the right filter for a given signal (noise vs. outliers)

Quantify filter performance (e.g., frequency distortion, outlier rejection)

Understand practical implications of impulse/frequency responses

Main intuitions:

- **Butterworth**: Smooth noise, but distorts edges
- **Median**: Crush outliers, preserve sharp transitions
- **Savitzky-Golay**: Smooth while preserving peaks

notes

- Trade-offs: Latency, distortion, computational cost.

Most important python code for this sub-module

notes

Key Python Functions

Butterworth Filter

```
from scipy.signal import butter, filtfilt
b, a = butter(N, cutoff, btype='low')
filtered_signal = filtfilt(b, a, signal)
```

Savitzky-Golay

```
from scipy.signal import savgol_filter
filtered_signal = savgol_filter(signal, window_length, polyorder)
```


notes

- Use `filtfilt` for zero-phase distortion!

Self-assessment material

notes

Question 1

Which filter is **least** suitable for removing sparse, large-amplitude outliers?

Potential answers:

I: **(wrong)** Median filter
II: **(wrong)** Butterworth filter
III: **(correct)** Moving average
IV: **(wrong)** Savitzky-Golay filter

Solution 1:

Butterworth filters smooth noise but are linearoutliers "pull" the output. Median filters excel at outlier rejection.

notes

- see the associated solution(s), if compiled with that ones :)

Recap of sub-module
“Filtering Lab: Noise Reduction and Outlier Rejection”

- Filter choice depends on signal features (noise band, outliers, quantization).
- Always check frequency/impulse responses for unintended effects.
- Non-linear filters (median) handle outliers; linear filters (Butterworth) handle noise.

notes

- Key idea: Theres no "best" filteronly the right tool for the job.