Programmer's Manual

Table of Contents

1. Introduction

This manual describes a comprehensive MATLAB-based audio processing and signal analysis suite comprising three integrated applications developed using MATLAB App Designer:

- **Interactive Audio Equalizer**: Allows users to load audio files, apply filters (Bell, Shelf, Band, HighCut, LowCut), visualize frequency responses and Fourier transforms, play filtered audio, and export results or save filter configurations.
- **Signal Measurement System**: Enables generation, acquisition, analysis, and visualization of synthetic or recorded signals, computing metrics like power, gain, and Total Harmonic Distortion (THD).
- **D Class Amplifier Simulator**: Simulates a D Class amplifier with sinusoidal or audio file inputs, adjustable parameters, and visualization of processing and output signals.

These applications are designed for audio processing, signal analysis, and simulation, supporting both mono and stereo signals, with intuitive GUIs and flexible configurations.

2. System Requirements

2.1 Hardware

- **Audio Output Device**: Speakers or headphones required for playback across all applications.

- **Microphone**: Required for recording mode in the Signal Measurement System.

- **General**: Sufficient system memory for processing large audio files and simulations.

2.2 Software

- MATLAB: R2021a or newer is recommended.

  Reasons:
  - App Designer: Required to run .mlapp-based applications.
  - Audio Toolbox: Provides audio processing functions such as audioplayer, audioread, etc., essential for audio handling.
  - Improved FFT performance: Useful for frequency domain processing and visualization.
  - Graphics rendering: Enhanced rendering of UIAxes and dynamic plots since R2021a.
  - File I/O compatibility: Functions like audiowrite and audioread are essential for importing/exporting files.

- No additional toolboxes are required.

3. Files

3.1 Main Application File

- **Equalizer**:
    - `EQapp.mlapp:` Main application file.
    - `PanellBanda.mlapp`: Controls band shelf filters.
    - `PanellBell.mlapp:` Controls bell filters.
    - `PanellPA.mlapp:` Controls low-cut filters.
    - `PanellPB.mlapp`: Controls high-cut filters.
    - `PanellShelf.mlapp:` Controls low/high shelf filters.
- **Signal Measurement System**:
    - `MScomentat.mlapp:` Main application file containing all logic, GUI, and signal processing functions.
- **D Class Amplifier Simulator**:
    - `DClassAmplifier.mlapp`: Main application file for the D Class amplifier simulation.

3.2 Application Structure

The application is structured as a main MATLAB class `(classdef)` with the following key components:

UI Components

Equalizer

- **UI Components**:
    - Toolbar with buttons for filter toggle, add/edit/delete filters, save/export, and filter selection dropdown.
    - Axes for filter and signal visualization.
    - Play/Pause buttons, audio slider, and panels for adding filters and exporting/saving.
- **Public Properties**:
    - `filtres:` Matrix of filter frequency responses.
    - `filtresParam:` Matrix of filter parameters.
    - `H:` Total equalizer frequency response.
    - `player:` audioplayer object for the current signal.
    - `isChanging:` Indicates if a filter is being modified.
- **Private Properties**:
    - `fs`: Sampling frequency.
    - `nextID:` ID for the next filter.

        - `audio, audioName, audioSegsL/R, audioEQL/R, audioWindow, audioFiltrat, filtersOn, ExportMode, N, Panell.`
- **Public Methods**:
    - `actualitza_eq():` Updates and plots the total frequency response.
    - `actualitza_ft()`: Updates and plots the Fourier transform of the current audio segment.
- **Private Methods**:
    - `hamming_custom(), segmentar_hamming(), ajuntar_hamming(), filtrar()` for audio processing.
- **App Callbacks**: Handle UI interactions (e.g., `startupFcn, ToggleToolOn/Off, addButtonClicked`).

Signal Measurement System

- **UI Components**:
    - Mode selector (`synthetic/recorded`), signal type dropdown, duration/frequency controls, play/record/import/export buttons, display fields for power, gain, THD, and axes for waveform, spectrogram, and frequency response.
- **Key Methods**:
    - `generateSignal()`: Generates synthetic signals.
    - `analyzeAndPlot()`: Computes power, gain, THD, and visualizations.
    - `RecordButtonPushed()`: Records audio.
    - Import/export and playback functions.
- **App Callbacks**: Handle UI interactions (e.g., mode selection, button presses).

D Class Amplifier Simulator

- **UI Components**:
    - UIAxes for input and output/processing signals.
    - Edit fields for sinusoidal frequency, working frequency, triangle wave frequency, comparator amplitude, and low-pass filter cutoff.
    - Buttons for generating sine signals, selecting audio files, running simulations, and exporting outputs.
- **Public Properties**:
    - `isStereo, fs, x, x_stereo, y, y_stereo.`

- **Private Methods**:
  - `lpf(), manualResample(), showx(), triangular(), comparador(), ampli_graf().`
- **Public Methods**:
  - `ampli():` Runs simulation for external control.
- **Callbacks**:
  - `GenerateSineButtonPushed, SelectFileButtonPushed, AmplifyButtonPushed, ExportFileButtonPushed.`

4. Configurable Parameters

4.1 Sampling Frequency (FS)

- **Equalizer**: Fixed at 48,000 Hz (`app.FS = 48000`).
- **Signal Measurement**: Fixed at 44,100 Hz `(app.FS = 44100).`
- **D Class Amplifier**: Adjustable via GUI edit field.
- **Note**: Not configurable via GUI in Equalizer and Signal Measurement.

4.2 Filter Types and Parameters (Equalizer)

- **Bell**: Central frequency `(fc)`, Q factor, Gain `(G)`.
- **Shelf:** `fc,` Slope `(S), G,` Low/High switch.
- **Band**: `fc,` Bandwidth `(BW), Q, G.`
- **HighCut/LowCut**: `fc.`

4.3 Audio Segment Size (Equalizer)

- **Purpose**: Determines samples per audio processing block.
- **Default**: 4096 samples `(N = 4096).`

4.4 Signal Duration (Signal Measurement)

- **Purpose**: Sets signal duration in seconds.
- **Range**: 0.1 to 10 seconds.
- **Default**: 1.0 second.
- **Location**: Numeric input field `(Duració (s)).`

4.5 Base Frequency (Signal Measurement)

- **Purpose**: Sets synthetic signal base frequency.
- **Range**: 20 Hz to 20,000 Hz.
- **Default**: 440 Hz (A4 note).
- **Location**: Numeric input field `(Freq. (Hz)`), visible in synthetic mode.

4.6 Signal Type (Signal Measurement)

- **Purpose**: Chooses synthetic signal waveform.
- **Options**: `Sine, square, triangle, chirp, saturated.`
- **Location**: Dropdown `(SignalTypeDropDown)`, visible in synthetic mode.

4.7 Triangle Wave and Comparator Parameters (D Class Amplifier)

- **Working Frequency**: Simulator’s sampling frequency.
- **Triangle Wave Frequency**: Frequency of the triangle wave.
- **Comparator Amplitude**: Output amplitude scaling factor.
- **Low-Pass Filter Cutoff**: Cutoff frequency for filtering.

5. System Workflow

5.1 Audio Loading and Segmentation (Equalizer)

- Loads audio using `audioread,` segments into overlapping blocks with Hamming windows.

5.2 Filter Application (Equalizer)

- Designs filters based on user parameters, computes frequency responses, and stores in filtres. Total response H is the product of individual responses.

5.3 Filtering Process (Equalizer)

- Transforms segments to frequency domain (FFT), multiplies by H, applies IFFT, and reconstructs using overlap-add.

5.4 Visualization (Equalizer)

- Displays individual and combined filter responses and Fourier transform of the current segment.

5.5 Playback (Equalizer)

- Plays filtered audio using `audioplayer,` navigable via slider.

5.6 Signal Acquisition (Signal Measurement)

- **Synthetic**: Generates signals using `generateSignal().`
- **Recorded**: Records via `audiorecorder`, trims silence.

5.7 Computation (Signal Measurement)

- **Power**: *P = (y' * y) / length(y).*
- **Gain**: *10 * log10(var(amplifiedSignal) / var(y)).*
- **THD**: *10 * log10(sum(harmonics(2:5)) / P1).*

5.8 Visualization (Signal Measurement)

- **Waveform**: Colored by THD (blue to red).
- **Spectrogram**: Manual FFT with Hamming windows, plotted using surf.
- **Frequency Response**: Hann-windowed FFT, comparing original and amplified signals in dB.

5.9 Input Method (D Class Amplifier)

- Generates sinusoidal signal or loads audio file, stores samples in `app.x` (mono) or `app.x_stereo` (stereo), and plots on UIAxes.

5.10 Process of Simulation (D Class Amplifier)

- `Upsamples` input to working frequency, generates triangle wave, compares signals, applies low-pass filter, and `downsamples` output.

5.11 Result of Simulation (D Class Amplifier)

- Plots upsampled input, triangle wave, comparator output, filtered signal, and final output in time and frequency domains. Outputs can be saved as audio files.

6. Internal Functions

6.1 Equalizer Functions

- **actualitza_eq()**: Updates and plots total frequency response H.
- **actualitza_ft()**: Computes and plots FFT of current audio segment.
- **segmentar_hamming()**: Segments audio with Hamming windows.
- **filtrar()**: Applies frequency-domain filtering and reconstructs signal.
- **Filter Design Functions:**
  - `filtre_bell(fc, Q, G, fs, N):` Bell filter response.
  - `filtre_shelf(fc, S, G, lH, fs, N):` Shelf filter response.
  - `filtre_Banda(fc, BW, Q, G, fs, N):` Band filter response.
  - `butterworth_highpass_freq_response(fc, fs, N):` High-pass Butterworth filter.
  - `butterworth_lowpass_freq_response(fc, fs, N)`: Low-pass Butterworth filter.

6.2 Signal Measurement Functions

- **generateSignal(app):** Generates synthetic signals (sine, square, triangle, chirp, saturated).
- **analyzeAndPlot(app)**: Computes power, gain, THD, and updates plots.
- **RecordButtonPushed(app, event):** Records audio, trims silence, and triggers analysis.
- **Playback and File I/O:** Handles audioplayer, audiowrite, audioread, etc.
- **UI Interaction Callbacks:** Manage mode selection, amplification, and visualization.

6.3 D Class Amplifier Functions

- **lpf(app, fc, x, fs)**: Applies low-pass filter.
- **manualResample(app, x, Fs_out, Fs_in)**: Resamples signal.
- **showx(app):** Plots input signal.
- **triangular(app, f_tri, Fs_Amp, N):** Generates triangle wave.
- **comparador(app, x, tri, G):** Simulates comparator.
- **ampli_graf(app):** Runs simulation and plots signals.

8. Public Methods

8.1 Equalizer Public Methods

- **actualitza_eq():** Updates filter responses and plots.
- **actualitza_ft():** Updates Fourier transform and plots.
- **valors()** (Filter Panels): Saves filter parameters to main app.

8.2 Signal Measurement Public Methods

- None explicitly defined, but `generateSignal()` and `analyzeAndPlot()` are core to external interaction.

8.3 D Class Amplifier Public Methods

- **ampli(app, x, fs):** Runs simulation for mono/stereo input, updates properties, and returns amplified signal.

9. Callbacks

9.1 Equalizer Callbacks

- **startupFcn():** Initializes app parameters and loads files.
- **UIFigureCloseRequest()**: Ends tasks and closes app.
- **ToggleToolOn/Off():** Activates/deactivates filters.
- **addButtonClicked(), editButtonClicked(), deleteButtonClicked():** Manage filter panels.
- **saveButtonClicked(), exportButtonClicked():** Handle save/export.
- **FiltreDropDownValueChanged():** Updates selected filter.
- **PlayButtonPushed(), PauseButtonPushed(), AudioSliderValueChanging():** Control playback and visualization.
- **Filter Panel Callbacks:** Manage fc, Q, G, BW, IH updates.

9.2 Signal Measurement Callbacks

- **ChangeAmpModeButtonPushed():** Opens amplification mode panel.
- **ButtonPushedFcn():** Confirms amplification mode selection.
- **RecordButtonPushed():** Triggers recording and analysis.
- **Other Callbacks:** Handle signal type, duration, frequency, and file I/O.

9.3 D Class Amplifier Callbacks

- **GenerateSineButtonPushed():** Generates 1-second sinusoidal signal at 48 kHz.
- **SelectFileButtonPushed():** Loads audio file and plots.
- **AmplifyButtonPushed():** Runs simulation for mono/stereo signals.
- **ExportFileButtonPushed():** Exports output signal as audio file.

10. Troubleshooting

10.1 Equalizer Issues

Issue	**Possible Cause(s)**	**Solution**
Audio file not loading	- Unsupported file format (e.g., not .wav or .mp3). - File not in project folder. - Incorrect file path.	- Use supported formats (.wav, .mp3). - Place file in the project folder - Provide full file path if not in project folder.
GUI not responding or displaying incorrectly	- Outdated MATLAB version. - Graphics driver issues.	- Use MATLAB R2018 or newer. - Update graphics drivers.
Audio not playing or playback issues	- Wrong audio device selected. - Device in use by another app. - audioplayer not initialized.	- Check MATLAB audio preferences. - Close other apps using the device. - Reload audio file.
Errors when saving or exporting files	- No write permissions. - Invalid file name. - File in use.	- Check folder permissions. - Use valid file names. - Close apps using the file.
Performance issues with large files	- Low system memory. - Small segment size (N).	- Increase N to reduce blocks. - Ensure sufficient memory.
Filter panels not working	- Multiple panels open. - Callback errors.	- Close open panels first. - Check the command window for errors.

10.2 Signal Measurement Issues

Issue	Possible Cause(s)	Solution
No audio recorded	Microphone not available, muted, or wrong device	Check system input device and permissions. Use `audiodevinfo` to verify
Audio not playing	`app.y` is empty or `sound()` fails	Ensure a signal is generated/recorded before playback
File import/export fails	File dialog canceled or invalid file selected	Make sure `.wav` or `.mat` files are properly selected
Crash during FFT	Input signal is empty or contains NaN/Inf	Verify signal content before analysis
UI components don't react	App not fully initialized or error occurred	Restart app or check console for messages

10.3 D Class Amplifier Issues

Issue	**Possible Cause(s)**	**Solution**
Simulation fails	Invalid parameters, mismatched signal lengths	Verify f_tri, Fs_Amp, ensure x and tri lengths match
No output signal	Input not loaded, simulation not run	Load input, run simulation
Export fails	Invalid filename, no permissions	Use valid names, check permissions
Crash during FFT	Input signal is empty or contains NaN/Inf	Verify signal content before analysis
Poor performance	High working frequency, large signal	Reduce Fs_Amp, use shorter signals