Chapter 13: Real-Time Signal Processing using MATLAB

Introduction

Real-time signal processing is a critical component in modern computing systems where immediate processing and response to signals is required, such as in communication systems, biomedical instruments, audio and video processing, and control systems. MATLAB provides an efficient environment for developing and simulating real-time signal processing applications. With the help of MATLAB's extensive toolbox support and Simulink integration, it becomes possible to model, simulate, and test real-time systems rapidly and effectively. This chapter introduces the concepts of real-time signal processing and provides a detailed hands-on guide to implementing them using MATLAB.

13.1 Basics of Signal Processing

13.1.1 Definition of a Signal

A signal is a function that conveys information about a phenomenon. It could be time-varying, such as an audio waveform, or spatially-varying like an image.

13.1.2 Classification of Signals

- **Continuous-Time Signals**
- **Discrete-Time Signals**
- **Deterministic vs Random Signals**
- **Periodic vs Aperiodic Signals**
- **Energy and Power Signals**

13.1.3 Signal Processing Systems

- **Analog Signal Processing**
- **Digital Signal Processing (DSP)**
- **Real-Time vs Offline Processing**

13.2 Real-Time Signal Processing Concepts

13.2.1 Real-Time Constraints

- **Latency**
- **Sampling Rate**

- **Throughput and Response Time**
- **Buffering Techniques**

13.2.2 Sampling and Aliasing

- **Nyquist Theorem**
- **Undersampling and Oversampling**
- **Anti-Aliasing Filters**

13.2.3 Quantization and Bit Depth

- **Quantization Noise**
- **Fixed-point vs Floating-point Representation**

13.3 MATLAB for Real-Time Signal Processing

13.3.1 MATLAB Toolboxes for Signal Processing

- **Signal Processing Toolbox**
- **DSP System Toolbox**
- **Audio Toolbox**
- **Simulink Real-Time**

13.3.2 Real-Time Simulation Environment

- **Real-Time Execution using Simulink**
- **External Mode Simulation**
- **Host-Target Communication**
- **MATLAB Coder for Real-Time Deployment**

13.4 Signal Acquisition in MATLAB

13.4.1 Audio Input using MATLAB

```
recObj = audiorecorder(44100, 16, 1);
recordblocking(recObj, 5);
myRecording = getaudiodata(recObj);
```

13.4.2 Real-Time Plotting

```
plot(myRecording);
title('Real-Time Audio Signal');
xlabel('Sample Index');
ylabel('Amplitude');
```

13.4.3 Using `dsp.AudioRecorder`

```
audioIn = dsp.AudioRecorder('SamplesPerFrame',1024);
audioOut = dsp.AudioPlayer('SampleRate',44100);
while true
audio = step(audioIn);
step(audioOut, audio);
end
```

13.5 Real-Time Signal Filtering

13.5.1 FIR and IIR Filters

- **Filter Design in MATLAB**

```
d = designfilt('lowpassfir','FilterOrder',20, ...
'CutoffFrequency',0.25,'DesignMethod','window');
```

13.5.2 Applying Filters in Real-Time

```
filt = dsp.FIRFilter('Numerator',d.Coefficients);
audioIn = dsp.AudioRecorder('SamplesPerFrame',1024);
audioOut = dsp.AudioPlayer('SampleRate',44100);
while true
x = step(audioIn);
y = step(filt, x);
step(audioOut, y);
end
```

13.6 Real-Time Fourier Analysis

13.6.1 Fast Fourier Transform (FFT)

```
Y = fft(myRecording);
f = (0:length(Y)-1)*44100/length(Y);
plot(f,abs(Y));
title('Frequency Domain');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
```

13.6.2 Real-Time Spectrogram

```
spectrogram(myRecording,256,200,256,44100,'yaxis');
```

13.7 Noise Removal and Enhancement

13.7.1 Noise Identification

- **Gaussian Noise**
- **Impulse Noise**

13.7.2 Real-Time Denoising Techniques

- **Median Filtering**
- **Wiener Filtering**

```
denoised = wiener2(myRecording,[5 5]);
plot(denoised);
```

13.8 Real-Time Audio Effects Implementation

13.8.1 Echo Effect

```
delay = dsp.Delay('Length',4410); % 0.1 sec delay at 44100 Hz
gain = 0.5;
audioIn = dsp.AudioRecorder('SamplesPerFrame',1024);
audioOut = dsp.AudioPlayer('SampleRate',44100);
while true
x = step(audioIn);
y = x + gain*step(delay,x);
step(audioOut,y);
end
```

13.8.2 Volume Control

```
volume = 0.8;
output = volume * inputSignal;
```

13.9 Real-Time Signal Visualization

13.9.1 Time-Domain Visualization

- Real-time updating plots using `animatedline`

13.9.2 Frequency-Domain Visualization

- Live spectrogram using `dsp.SpectrumAnalyzer`

13.10 Simulink for Real-Time Systems

13.10.1 Real-Time Blocks in Simulink

- **Signal Generator**
- **Scope**
- **Audio Device Reader/Writer**

13.10.2 External Mode Execution

- Execute on host or target systems with data monitoring

13.10.3 Code Generation for Embedded Real-Time

- Using **Simulink Coder** or **MATLAB Coder**
- Deployable to ARM Cortex, Arduino, Raspberry Pi

13.11 Case Study: Real-Time Voice Recorder and Playback

Objective

To create a real-time voice recording and playback system with filtering and noise suppression.

Steps Involved

1. **Capture audio using** `audiorecorder`
2. **Filter audio in real-time with DSP toolbox**
3. **Add live gain control and frequency monitoring**
4. **Play the processed signal back using** `audioplayer`
5. **Optionally log and save the data**

13.12 Challenges in Real-Time Processing

13.12.1 Computational Delays

- Caused by heavy filter designs or transform operations.

13.12.2 Memory Management

- Real-time systems must manage buffers efficiently to avoid overflow/underflow.

13.12.3 Integration with Hardware

- Requires understanding of I/O latency, drivers, and real-time OS.
