Ten Standard Tasks for Evaluating Signal Analysis Code Generation

Vladislav Bulgakov

April 13, 2025

Abstract

This document outlines ten standard tasks commonly used in digital signal processing (DSP) and signal analysis. They serve as benchmarks for evaluating whether a large language model can reliably generate code suitable for practical scenarios.

1 Introduction

Signal analysis often involves transforming or filtering data in order to glean insight about underlying structures or behaviors. When assessing a code-generating model for signal analysis, it is valuable to test a broad range of tasks. Below are ten tasks that collectively cover frequency-domain analysis, time-domain filtering, multi-rate operations, time-frequency representations, and more.

2 Tasks Overview

1. **Fast Fourier Transform (FFT) of a Given Signal**
   - *Rationale*: Frequency-domain representation is one of the most fundamental operations in DSP.
   - *Key Points*:
     - Use of libraries such as `numpy.fft` or `scipy.fft`.
     - Handling real vs. complex signals, windowing, zero-padding, etc.
     - Ensuring correct frequency bin mapping.

2. **Inverse FFT (IFFT) of a Frequency Spectrum**
   - *Rationale*: Complements the FFT, allowing signals to be reconstructed in the time domain.
   - *Key Points*:
     - Handling phase and amplitude.
     - Working with real vs. complex signals.
     - Managing array shapes, ensuring correct dimensionality.
3. **Design and Application of a Digital Low-Pass Filter**
   - *Rationale*: Filters are a fundamental building block for noise reduction and band-limited analysis.
   - *Key Points*:
     - IIR designs like Butterworth, Chebyshev, or Elliptic filters.
     - FIR design using windows (Hamming, Blackman, etc.).
     - Implementation details such as cut-off frequency, filter order, and passband ripple.
4. **Design of a Bandpass Filter**
   - *Rationale*: Allows signals in a specified frequency band, rejecting others.
   - *Key Points*:
     - Low and high cutoff frequencies.
     - Transition bands and stability considerations for IIR.
     - Zero-phase filtering vs. causal filtering (e.g. `filtfilt` vs. `lfilter`).
5. **Resampling a Signal**
   - *Rationale*: Multi-rate DSP tasks (e.g., changing sampling rates for different system requirements).
   - *Key Points*:
     - Upsampling vs. downsampling.
     - Polyphase filters or straightforward interpolation.
     - Handling boundary conditions to avoid aliasing.
6. **Window-Based Smoothing or Moving Average**
   - *Rationale*: Reduces noise or fluctuations in the signal.

- *Key Points*:
  - Different window shapes (e.g. boxcar, triangular, Gaussian).
  - Managing boundary effects (padding vs. ignoring edges).
  - Implementation via convolution or direct sliding methods.

7. **Computation of a Spectrogram**
   - *Rationale*: Essential time-frequency representation for non-stationary signals.
   - *Key Points*:
     - Repeated short-time FFT with overlapping segments.
     - Windowing and step sizes (hop length).
     - Visualization in decibels or amplitude.
8. **Peak Detection**
   - *Rationale*: Identifying local maxima for event detection or spectral peaks.
   - *Key Points*:
     - Setting thresholds, minimum peak distances.
     - Handling noise and smoothing prior to detection.
     - Returning indices or amplitude values of identified peaks.
9. **Cross-Correlation (Time or Frequency Domain)**
   - *Rationale*: Used for signal alignment, similarity, or system identification.
   - *Key Points*:
     - Direct time-domain convolution vs. FFT-based multiplication.
     - Normalizing correlation or leaving it unnormalized.
     - Handling large data efficiently.
10. **Wavelet Transform (e.g., DWT, CWT)**
    - *Rationale*: More advanced time-frequency analysis than STFT, excellent for non-stationary signals with transient features.
    - *Key Points*:
      - Choice of wavelet families (Daubechies, Haar, Morlet, etc.).

- Discrete Wavelet Transform (DWT) vs. Continuous Wavelet Transform (CWT).
- Interpreting coefficients for denoising or feature extraction.

3 Conclusion

These ten tasks span a broad spectrum of typical signal-processing techniques, from fundamental frequency-domain transforms to advanced multirate and wavelet analyses. By testing large language models on each of these, one can gain insight into the model's ability to:

- Generate correct, functional, and efficient code.
- Make appropriate use of well-known libraries (e.g., `numpy`, `scipy.signal`).
- Handle real-world edge cases in complex DSP scenarios.

A model that performs consistently well across these tasks demonstrates good potential for general-purpose coding in the signal analysis domain.