# Product Requirements Document

## Sinusoidal Peak Detection using Convolution

**Author:** Yair Levi
**Date:** October 8, 2025
**Version:** 1.0
**Status:** Draft

---

# 1. Executive Summary

This document outlines the requirements for a signal processing system that detects peaks in sinusoidal signals using convolution-based template matching. The system generates synthetic sinusoidal data, creates a peak template, and uses convolution to identify all peak locations in the signal.

---

# 2. Product Overview

### 2.1 Purpose

Develop a Python-based proof-of-concept for peak detection in periodic signals using convolution, demonstrating the effectiveness of template matching for signal analysis.

### 2.2 Objectives

- Generate controllable synthetic sinusoidal signals for testing
- Implement template-based peak detection using convolution
- Provide clear visualization of signal analysis results
- Demonstrate reliable detection of periodic features in signals

### 2.3 Target Users

- Signal processing engineers
- Data scientists working with periodic signals
- Educational use for teaching convolution concepts
- Research teams developing detection algorithms

---

# 3. Functional Requirements

### 3.1 Signal Generation (FR-1)

**Priority:** High

The system shall generate a sinusoidal signal with the following specifications:

- **Cycles:** 10 complete sine wave cycles
- **Amplitude:** -1 to +1 (normalized)
- **Period:** Each cycle spans $2\pi$ radians
- **Sampling Rate:** 200 samples per cycle
- **Total Samples:** 2,000 samples (10 cycles × 200 samples)

**Acceptance Criteria:**

- Signal spans exactly 10 complete cycles
- Peak amplitude equals +1.0
- Trough amplitude equals -1.0
- Samples are evenly distributed across time domain

## 3.2 Template Extraction (FR-2)

**Priority:** High

The system shall extract a peak template from the generated signal:

- **Template Source:** First peak of the sinusoidal signal
- **Template Length:** 30 samples
- **Centering:** Peak value at center of template (15 samples before, peak, 14 samples after)
- **Purpose:** Serve as reference pattern for peak detection

**Acceptance Criteria:**

- Template contains exactly 30 samples
- Maximum value occurs at template center
- Template captures characteristic peak shape

## 3.3 Convolution-Based Detection (FR-3)

**Priority:** High

The system shall perform convolution between the signal and template:

- **Method:** 1D convolution with template flipping
- **Output Mode:** Same size as input signal
- **Peak Detection:** Identify local maxima in convolution output
- **Threshold:** Detect peaks exceeding 90% of maximum convolution response

**Acceptance Criteria:**

- Convolution output length matches input signal length
- All 10 peaks are detected
- No false positive detections
- Peak locations accurate within ±5 samples

## 3.4 Visualization (FR-4)

**Priority:** High

The system shall provide comprehensive visualization including:

**Plot 1: Original Signal**

- Display complete 10-cycle sinusoidal signal
- X-axis: Time in radians
- Y-axis: Amplitude
- Reference line at zero amplitude

**Plot 2: Template Display**

- Show extracted 30-sample template
- Highlight peak location
- Display sample points

**Plot 3: Detection Results**

- Plot convolution output
- Mark detected peak locations with distinct markers
- Show detection threshold line
- Include legend with detection count

**Acceptance Criteria:**

- All three plots clearly labeled and titled
- Detected peaks visually distinguishable
- Axes properly scaled and labeled
- Figure suitable for presentation/publication

### 3.5 Reporting (FR-5)

**Priority:** Medium

The system shall output summary statistics:

- Total number of detected peaks
- Peak locations (sample indices)
- Peak spacing intervals
- Template length and parameters

**Acceptance Criteria:**

- Console output clearly formatted
- All key metrics reported
- Peak spacings approximately 200 samples

---

## 4. Technical Requirements

### 4.1 Dependencies

- **Python:** 3.7 or higher
- **NumPy:** For numerical computations and signal generation
- **Matplotlib:** For visualization and plotting

### 4.2 Performance

- Execution time: < 2 seconds for standard configuration
- Memory usage: < 100 MB
- Support signals up to 100,000 samples

### 4.3 Code Quality

- Well-commented code explaining key operations
- Modular structure for easy modification
- Clear variable naming following conventions

---

# 5. Non-Functional Requirements

## 5.1 Usability

- Single-file implementation for easy distribution
- No configuration files required
- Clear console output with progress information

## 5.2 Maintainability

- Code comments explaining convolution logic
- Configurable parameters at top of script
- Easy to modify cycle count, sampling rate, template size

## 5.3 Portability

- Cross-platform compatibility (Windows, macOS, Linux)
- Standard library dependencies only (NumPy, Matplotlib)
- No hardware-specific optimizations

---

# 6. Use Cases

## 6.1 Educational Demonstration

**Actor:** Instructor teaching signal processing
**Goal:** Demonstrate convolution-based template matching
**Steps:**

1. Run script to generate signal
2. Explain template extraction concept
3. Show convolution sliding window operation
4. Discuss peak detection results

## 6.2 Algorithm Validation

**Actor:** Signal processing engineer
**Goal:** Validate convolution-based detection approach
**Steps:**

1. Generate known test signal
2. Verify all peaks detected correctly
3. Measure detection accuracy
4. Compare with alternative methods

## 6.3 Research Prototype

**Actor:** Research scientist
**Goal:** Develop custom detection algorithm
**Steps:**

1. Use as baseline implementation
2. Modify template extraction logic
3. Test alternative detection thresholds
4. Benchmark performance improvements

---

# 7. Future Enhancements

## 7.1 Phase 2 Features (Post-MVP)

- Support for noisy signals (SNR variation)
- Multiple template matching (different frequencies)
- Real-time signal processing capability
- Adaptive threshold calculation
- Export results to CSV/JSON

## 7.2 Advanced Features (Future Consideration)

- GUI interface for parameter adjustment
- Batch processing of multiple signals
- Frequency domain analysis
- Machine learning-based peak detection comparison
- Integration with data acquisition hardware

---

# 8. Success Metrics

## 8.1 Key Performance Indicators

- **Detection Accuracy:** 100% of peaks correctly identified
- **False Positive Rate:** 0%
- **Execution Time:** < 2 seconds
- **Code Coverage:** > 80% (if tests implemented)

## 8.2 Quality Metrics

- Zero critical bugs
- Clear visualization output
- Positive user feedback from initial testing

---

# 9. Constraints and Assumptions

## 9.1 Constraints

- Python 3.7+ environment required
- Limited to 1D signal processing
- Template must come from same signal family

## 9.2 Assumptions

- Input signal is purely sinusoidal (no noise)
- Peaks are regularly spaced
- User has basic Python knowledge
- Display capable of showing plots

---

# 10. Risks and Mitigation

| Risk | Impact | Probability | Mitigation |
|---|---|---|---|
| Dependency version conflicts | Medium | Low | Document specific versions, use virtual environment |
| Poor visualization on different displays | Low | Medium | Use standard Matplotlib defaults, test on multiple systems |
| Template doesn't generalize | High | Low | Validate with different frequencies, document limitations |
| Performance with large datasets | Medium | Medium | Add performance notes, suggest downsampling for large signals |

---

# 11. Approval

| Role | Name | Signature | Date |
|---|---|---|---|
| Author | Yair Levi | | Oct 8, 2025 |
| Reviewer | | | |
| Approver | | | |

---

# 12. Revision History

| Version | Date | Author | Changes |
|---|---|---|---|
| 1.0 | Oct 8, 2025 | Yair Levi | Initial draft |

---

# Appendix A: Technical Specifications

## Signal Parameters

Frequency: $f = 1/(2\pi)$ Hz per cycle
Sampling Rate: $f_s = 200$ samples per cycle
Nyquist Frequency: $f_n = 100$ samples per cycle
Time Vector: $t = \text{linspace}(0, 20\pi, 2000)$

## Convolution Mathematics

$\text{output}[n] = \Sigma\, x[m] \cdot h[n-m]$
where h is the flipped template

## Detection Algorithm

Peak detected if:

- output[i] > output[i-1]
- output[i] > output[i+1]
- output[i] > 0.9 × max(output)
- |i - previous_peak| > 100 samples