SHMTools Dataset Management

Overview

This notebook provides comprehensive documentation and management utilities for the SHMTools example datasets. It covers:

1. **Dataset Overview**: Physical descriptions and experimental setups
2. **Data Loading**: Standardized loading procedures and validation
3. **Dataset Exploration**: Structure analysis and visualization
4. **Usage Examples**: Common data access patterns for different examples
5. **Integrity Validation**: Automated checking of dataset completeness

This notebook serves as both documentation and a practical utility for working with SHMTools data.

Setup

Import required modules and setup the environment.

In [1]:
```python
# Setup path to find shmtools
import sys
from pathlib import Path

# Handle different execution contexts
current_dir = Path.cwd()
possible_paths = [
    current_dir,  # From project root
    current_dir.parent,  # From examples/
    current_dir.parent.parent,  # From examples/notebooks/
    current_dir.parent.parent.parent,  # From examples/notebooks/utilities/
]

for path in possible_paths:
    if (path / 'shmtools').exists():
        if str(path) not in sys.path:
            sys.path.insert(0, str(path))
        print(f"Found shmtools at: {path}")
        break
else:
    print("Warning: Could not find shmtools module")

# Import libraries
import numpy as np
import matplotlib.pyplot as plt
from typing import Dict, Any
```

```python
# Import SHMTools data utilities
from shmtools.utils.data_loading import (
    load_3story_data,
    load_cbm_data,
    load_active_sensing_data,
    load_sensor_diagnostic_data,
    load_modal_osp_data,
    get_available_datasets,
    check_data_availability,
    validate_dataset_integrity,
    print_dataset_summary,
    load_example_data,
    get_data_dir
)

# Setup plotting defaults
plt.style.use('default')
plt.rcParams['figure.figsize'] = (12, 8)
plt.rcParams['font.size'] = 10

print("Setup complete.")
```

```
Found shmtools at: /Users/eric/repo/shm/shmtools-python
Setup complete.
```

```
/Users/eric/repo/shm/shmtools-python/shmtools/classification/nlpca.py:27: Us
erWarning: TensorFlow not available. NLPCA functions will not work. Install
TensorFlow: pip install tensorflow
  warnings.warn(
```

Dataset Availability Check

Check which datasets are currently available and validate their integrity.

In [2]:
```python
# Check dataset availability
print("Dataset Availability:")
print("=" * 60)
check_data_availability()
```

```
Dataset Availability:
============================================================
Data directory: /Users/eric/repo/shm/shmtools-python/examples/data
Directory exists: True

data3SS.mat                    ( 25.0 MB) - ✓ Available
data_CBM.mat                   ( 54.0 MB) - ✓ Available
data_example_ActiveSense.mat   ( 32.0 MB) - ✓ Available
dataSensorDiagnostic.mat       (  0.1 MB) - ✓ Available
data_OSPExampleModal.mat       (  0.1 MB) - ✓ Available

Available: 5/5 datasets (111.1 MB total)
```

In [3]:
```python
# Comprehensive dataset summary
print_dataset_summary()
```

```
============================================================================
====
SHMTools Dataset Summary
============================================================================
====
```

```
data3SS.mat (25 MB) - ✓ VALID
------------------------------------------------------------
Description: 3-story structure base excitation (primary dataset)
Examples: PCA, Mahalanobis, SVD, NLPCA, Factor Analysis
Structure:
  shape: (8192, 5, 170)
  fs: 2000.0
  num_states: 17
  channels: ['Force', 'Ch2', 'Ch3', 'Ch4', 'Ch5']

data_CBM.mat (54 MB) - ✓ VALID
------------------------------------------------------------
Description: Condition-based monitoring (rotating machinery)
Examples: CBM Bearing Analysis, CBM Gearbox Analysis
Structure:
  shape: (10240, 4, 384)
  fs: 2048.0
  fault_states: 6
  channels: ['Tachometer', 'Gearbox_Accel', 'Top_Bearing_Accel', 'Side_Beari
ng_Accel']

data_example_ActiveSense.mat (32 MB) - ✓ VALID
------------------------------------------------------------
Description: Guided wave ultrasonic measurements
Examples: Active Sensing Feature Extraction
Structure:
  variables: ['__header__', '__version__', '__globals__', 'waveformBase', 'w
aveformTest', 'borderStruct', 'sampleRate', 'sensorLayout', 'pairList', 'act
uationWaveform', 'damageLocation']
  estimated_size: 39686248

dataSensorDiagnostic.mat (0.06 MB) - ✓ VALID
------------------------------------------------------------
Description: Piezoelectric sensor impedance measurements
Examples: Sensor Diagnostics
Structure:
  variables: ['__header__', '__version__', '__globals__', 'sd_ex_broken', 's
d_ex']
  estimated_size: 73692

data_OSPExampleModal.mat (0.05 MB) - ✓ VALID
------------------------------------------------------------
Description: Modal analysis and optimal sensor placement
Examples: Modal Features, Optimal Sensor Placement
Structure:
  variables: ['__header__', '__version__', '__globals__', 'modeShapes', 'res
pDOF', 'nodeLayout', 'elements']
  estimated_size: 90096

============================================================================
====
Summary: 5/5 valid, 5/5 available
```

Dataset 1: 3-Story Structure (data3SS.mat)

Physical Description

The primary dataset contains measurements from a 3-story aluminum frame structure designed for structural health monitoring research at Los Alamos National Laboratory.

Physical Structure:

- Aluminum columns (17.7×2.5×0.6 cm) and plates (30.5×30.5×2.5 cm)
- 4-column frame design per floor (essentially 4-DOF system)
- Sliding rails constraining motion to x-direction only
- Suspended center column with adjustable bumper for damage simulation
- Base isolation using rigid foam

Instrumentation:

- Electrodynamic shaker for base excitation (band-limited random 20-150 Hz)
- Load cell measuring input force (2.2 mV/N sensitivity)
- 4 accelerometers at floor centerlines (1000 mV/g sensitivity)
- National Instruments PXI data acquisition system

In [4]:
```python
# Load and examine 3-story structure data
try:
    data_3story = load_3story_data()

    print("3-Story Structure Dataset:")
    print("=" * 50)
    print(f"Dataset shape: {data_3story['dataset'].shape}")
    print(f"Sampling frequency: {data_3story['fs']} Hz")
    print(f"Channels: {data_3story['channels']}")
    print(f"Total conditions: {len(data_3story['conditions'])}")
    print(f"Damage states: {len(data_3story['state_descriptions'])}")
    print(f"Description: {data_3story['description']}")

    # Show damage state descriptions
    print("\nDamage State Descriptions:")
    print("-" * 50)
    for state, desc in data_3story['state_descriptions'].items():
        print(f"State {state:2d}: {desc}")

except FileNotFoundError as e:
    print(f"3-Story dataset not available: {e}")
    print("Please download data3SS.mat and place it in the data directory.")
```

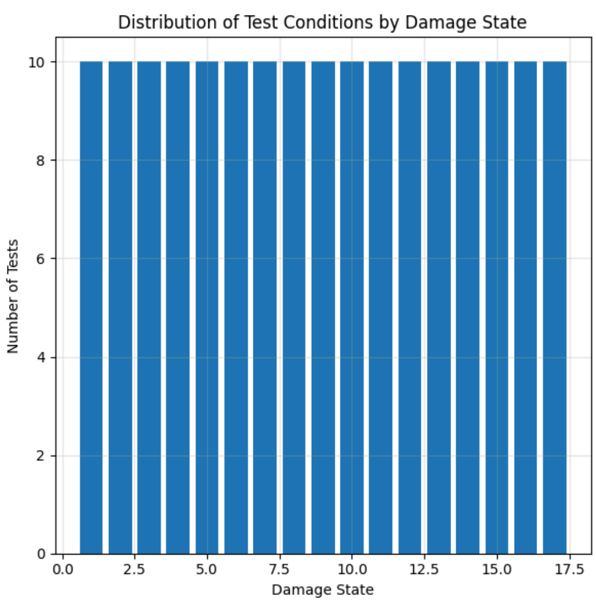
```
3-Story Structure Dataset:
==================================================
Dataset shape: (8192, 5, 170)
Sampling frequency: 2000.0 Hz
Channels: ['Force', 'Ch2', 'Ch3', 'Ch4', 'Ch5']
Total conditions: 170
Damage states: 17
Description: 3-story structure base excitation data (LANL)

Damage State Descriptions:
--------------------------------------------------
State  1: Undamaged - Baseline condition
State  2: Undamaged - Mass = 1.2 kg at the base
State  3: Undamaged - Mass = 1.2 kg on the 1st floor
State  4: Undamaged - 87.5% stiffness reduction in column 1BD
State  5: Undamaged - 87.5% stiffness reduction in columns 1AD and 1BD
State  6: Undamaged - 87.5% stiffness reduction in column 2BD
State  7: Undamaged - 87.5% stiffness reduction in columns 2AD and 2BD
State  8: Undamaged - 87.5% stiffness reduction in column 3BD
State  9: Undamaged - 87.5% stiffness reduction in columns 3AD and 3BD
State 10: Damaged - Gap = 0.20 mm
State 11: Damaged - Gap = 0.15 mm
State 12: Damaged - Gap = 0.13 mm
State 13: Damaged - Gap = 0.10 mm
State 14: Damaged - Gap = 0.05 mm
State 15: Damaged - Gap = 0.20 mm and mass = 1.2 kg at the base
State 16: Damaged - Gap = 0.20 mm and mass = 1.2 kg on the 1st floor
State 17: Damaged - Gap = 0.10 mm and mass = 1.2 kg on the 1st floor
```

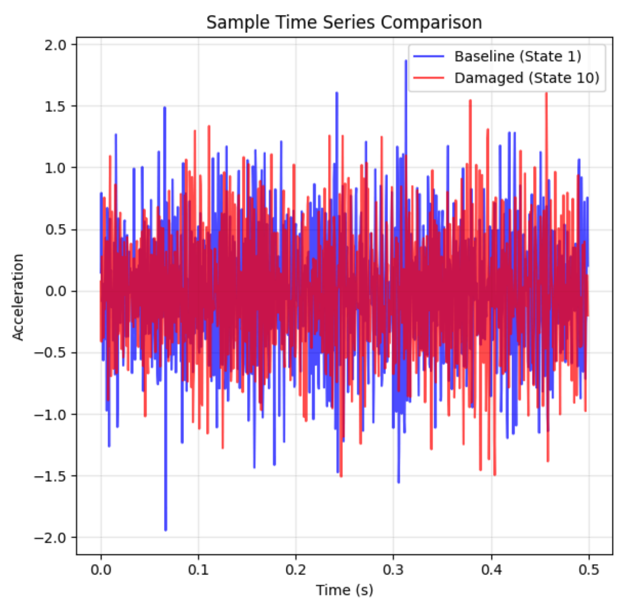
Data Structure Analysis

In [5]:

```python
# Analyze data structure if available
if 'data_3story' in locals():
    dataset = data_3story['dataset']
    damage_states = data_3story['damage_states']

    # Plot damage state distribution
    plt.figure(figsize=(12, 6))

    plt.subplot(1, 2, 1)
    unique_states, counts = np.unique(damage_states, return_counts=True)
    plt.bar(unique_states, counts)
    plt.xlabel('Damage State')
    plt.ylabel('Number of Tests')
    plt.title('Distribution of Test Conditions by Damage State')
    plt.grid(True, alpha=0.3)

    # Plot example time series from different states
    plt.subplot(1, 2, 2)
    t = np.arange(dataset.shape[0]) / data_3story['fs']

    # Plot baseline condition (state 1, test 1) - channel 2
    baseline_signal = dataset[:1000, 1, 0]  # First 1000 points, channel 2,
    plt.plot(t[:1000], baseline_signal, 'b-', label='Baseline (State 1)', al
```

```python
    # Plot damaged condition (state 10, test 1) - channel 2
    damage_signal = dataset[:1000, 1, 90]  # First 1000 points, channel 2, c
    plt.plot(t[:1000], damage_signal, 'r-', label='Damaged (State 10)', alph

    plt.xlabel('Time (s)')
    plt.ylabel('Acceleration')
    plt.title('Sample Time Series Comparison')
    plt.legend()
    plt.grid(True, alpha=0.3)

    plt.tight_layout()
    plt.show()

    # Statistical summary
    print("\nStatistical Summary (Channel 2):")
    print("-" * 40)

    # Compare baseline vs damaged conditions
    baseline_data = dataset[:, 1, :90].flatten()  # All baseline conditions,
    damaged_data = dataset[:, 1, 90:].flatten()   # All damaged conditions,

    print(f"Baseline - Mean: {np.mean(baseline_data):.4f}, Std: {np.std(base
    print(f"Damaged  - Mean: {np.mean(damaged_data):.4f}, Std: {np.std(damag
    print(f"RMS Ratio (Damaged/Baseline): {np.std(damaged_data)/np.std(basel
```






```
Statistical Summary (Channel 2):
----------------------------------------
Baseline - Mean: -0.0039, Std: 0.5019
Damaged  - Mean: -0.0039, Std: 0.4938
RMS Ratio (Damaged/Baseline): 0.984
```

Dataset 2: Condition-Based Monitoring (data_CBM.mat)

Physical Description

Rotating machinery vibration data collected from the SpectraQuest Magnum Machinery Fault Simulator for bearing and gearbox fault analysis.

Test Setup:

- Main shaft: 3/4" diameter steel, 28.5" center-to-center bearing support
- Gearbox: Hub City M2, 1.5:1 ratio, 18/27 teeth (pinion/gear)
- Belt drive: ~1:3.71 ratio, 13" span, 3.7 lbs tension
- Magnetic brake: 1.9 lbs-in torsional load
- Shaft speed: ~1000 rpm nominally constant

Fault Conditions:

- Ball bearing faults (roller spin)
- Gearbox worn tooth faults
- Baseline conditions with ball and fluid bearings

In [6]:

```python
# Load and examine CBM data
try:
    data_cbm = load_cbm_data()

    print("Condition-Based Monitoring Dataset:")
    print("=" * 50)

    # Show available variables
    print("Available variables:")
    for key, value in data_cbm.items():
        if isinstance(value, np.ndarray):
            print(f"  {key}: {value.shape} ({value.dtype})")
        else:
            print(f"  {key}: {value}")

    # Show fault state descriptions
    if 'fault_states' in data_cbm:
        print("\nFault State Descriptions:")
        print("-" * 50)
        for state, desc in data_cbm['fault_states'].items():
            print(f"State {state}: {desc}")

    # Show bearing fault frequencies if available
    shaft_freq = data_cbm['shaft_speed_rpm'] / 60.0  # Convert RPM to Hz
    print(f"\nBearing Fault Frequencies (Shaft = {shaft_freq:.1f} Hz):")
    print("-" * 50)
    print(f"Cage Speed: {3.048 * shaft_freq:.1f} Hz")
    print(f"Outer Race: {3.048 * shaft_freq:.1f} Hz")
    print(f"Inner Race: {4.95 * shaft_freq:.1f} Hz")
    print(f"Ball Spin: {1.992 * shaft_freq:.1f} Hz")

except FileNotFoundError as e:
    print(f"CBM dataset not available: {e}")
    print("Please download data_CBM.mat and place it in the data directory."
```

```
Condition-Based Monitoring Dataset:
==================================================
Available variables:
  __header__: b'MATLAB 5.0 MAT-file, Platform: PCWIN64, Created on: Fri Jul
26 13:27:52 2013'
  __version__: 1.0
  __globals__: []
  Fs: (1, 1) (uint16)
  damageStates: (384, 1) (uint8)
  dataset: (10240, 4, 384) (float32)
  stateList: (384, 1) (uint8)
  fs: 2048.0
  duration: 5.0
  shaft_speed_rpm: 1000.0
  channels: ['Tachometer', 'Gearbox_Accel', 'Top_Bearing_Accel', 'Side_Beari
ng_Accel']
  fault_states: {1: 'Baseline 1 (ball bearings, healthy)', 2: 'Baseline 2 (b
all bearings, healthy)', 3: 'Ball roller spin fault', 4: 'Baseline 1 (fluid
bearings, healthy)', 5: 'Baseline 2 (fluid bearings, healthy)', 6: 'Gearbox
worn tooth fault'}
  description: SpectraQuest Magnum fault simulator CBM data (LANL)

Fault State Descriptions:
--------------------------------------------------
State 1: Baseline 1 (ball bearings, healthy)
State 2: Baseline 2 (ball bearings, healthy)
State 3: Ball roller spin fault
State 4: Baseline 1 (fluid bearings, healthy)
State 5: Baseline 2 (fluid bearings, healthy)
State 6: Gearbox worn tooth fault

Bearing Fault Frequencies (Shaft = 16.7 Hz):
--------------------------------------------------
Cage Speed: 50.8 Hz
Outer Race: 50.8 Hz
Inner Race: 82.5 Hz
Ball Spin: 33.2 Hz
```

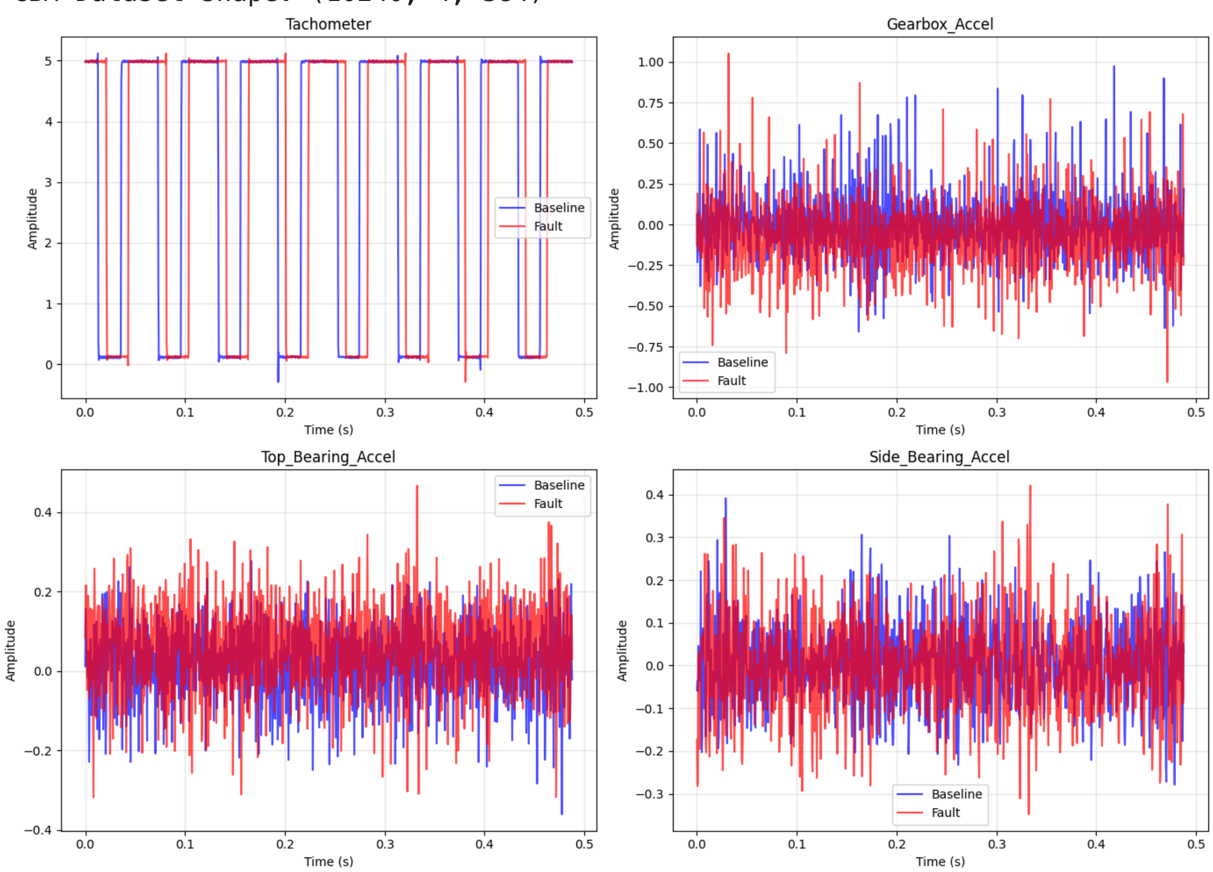
CBM Data Visualization

In [7]:

```
# Visualize CBM data if available
if 'data_cbm' in locals() and 'dataset' in data_cbm:
    cbm_dataset = data_cbm['dataset']
    fs = data_cbm['fs']
    channels = data_cbm['channels']

    print(f"CBM Dataset shape: {cbm_dataset.shape}")

    # Plot example signals from different channels and conditions
    plt.figure(figsize=(14, 10))

    # Time vector
    t = np.arange(1000) / fs  # First 1000 points for visualization

    # Plot signals from each channel
```

```
    for i, channel in enumerate(channels):
        plt.subplot(2, 2, i+1)

        # Plot baseline condition (assuming condition 0)
        if cbm_dataset.shape[2] > 0:
            baseline_signal = cbm_dataset[:1000, i, 0]
            plt.plot(t, baseline_signal, 'b-', label='Baseline', alpha=0.7)

        # Plot fault condition (assuming later condition exists)
        if cbm_dataset.shape[2] > 64:  # If we have fault conditions
            fault_signal = cbm_dataset[:1000, i, 64]
            plt.plot(t, fault_signal, 'r-', label='Fault', alpha=0.7)

        plt.xlabel('Time (s)')
        plt.ylabel('Amplitude')
        plt.title(f'{channel}')
        plt.legend()
        plt.grid(True, alpha=0.3)

    plt.tight_layout()
    plt.show()

elif 'data_cbm' in locals():
    print("CBM data loaded but 'dataset' variable not found.")
    print("Available variables:", list(data_cbm.keys()))
```

```
CBM Dataset shape: (10240, 4, 384)
```



Other Datasets

Brief exploration of the remaining datasets used in specialized examples.

In [8]:
```
# Load other datasets if available
datasets_to_check = [
    ('Active Sensing', load_active_sensing_data),
    ('Sensor Diagnostic', load_sensor_diagnostic_data),
    ('Modal OSP', load_modal_osp_data)
]

loaded_datasets = {}

for name, loader_func in datasets_to_check:
    try:
        data = loader_func()
        loaded_datasets[name] = data

        print(f"\n{name} Dataset:")
        print("=" * (len(name) + 10))

        # Show dataset structure
        total_size = 0
        for key, value in data.items():
            if isinstance(value, np.ndarray):
                size_mb = value.nbytes / (1024**2)
                total_size += size_mb
                print(f"  {key}: {value.shape} ({value.dtype}) - {size_mb:.2
            elif isinstance(value, (list, dict)):
                print(f"  {key}: {type(value).__name__} (length: {len(value)
            else:
                print(f"  {key}: {value}")

        print(f"Total estimated size: {total_size:.2f} MB")

    except FileNotFoundError:
        print(f"\n{name} dataset not available.")
    except Exception as e:
        print(f"\nError loading {name} dataset: {e}")
```

```
Active Sensing Dataset:
========================
  __header__: b'MATLAB 5.0 MAT-file, Platform: PCWIN, Created on: Thu Jul 29
10:24:53 2010'
  __version__: 1.0
  __globals__: list (length: 0)
  waveformBase: (10000, 496) (float32) - 18.92 MB
  waveformTest: (10000, 496) (float32) - 18.92 MB
  borderStruct: (1, 1) ([('outside', 'O')]) - 0.00 MB
  sampleRate: (1, 1) (float32) - 0.00 MB
  sensorLayout: (3, 32) (float32) - 0.00 MB
  pairList: (2, 496) (float32) - 0.00 MB
  actuationWaveform: (469, 1) (float32) - 0.00 MB
  damageLocation: (2, 1) (float32) - 0.00 MB
Total estimated size: 37.85 MB

Sensor Diagnostic Dataset:
===========================
  __header__: b'MATLAB 5.0 MAT-file, Platform: PCWIN, Created on: Thu Jul 29
10:28:02 2010'
  __version__: 1.0
  __globals__: list (length: 0)
  sd_ex_broken: (801, 13) (float32) - 0.04 MB
  sd_ex: (801, 10) (float32) - 0.03 MB
Total estimated size: 0.07 MB

Modal OSP Dataset:
===================
  __header__: b'MATLAB 5.0 MAT-file, Platform: PCWIN, Created on: Thu Jul 29
10:26:27 2010'
  __version__: 1.0
  __globals__: list (length: 0)
  modeShapes: (1260, 13) (float32) - 0.06 MB
  respDOF: (1260, 2) (float32) - 0.01 MB
  nodeLayout: (4, 420) (float32) - 0.01 MB
  elements: (9, 216) (float32) - 0.01 MB
Total estimated size: 0.09 MB
```

Dataset Usage Examples

Demonstrate common data access patterns for different types of SHMTools examples.

In [9]:

```python
# Example 1: Loading data for outlier detection examples (PCA, Mahalanobis,
print("Example 1: Outlier Detection Data Loading")
print("=" * 50)

try:
    # This convenience function preprocesses the 3-story data for outlier de
    pca_data = load_example_data('pca')

    print(f"Preprocessed signals shape: {pca_data['signals'].shape}")
    print(f"Channels included: {pca_data['channels']}")
    print(f"Time points (t): {pca_data['t']}")
    print(f"Channels (m): {pca_data['m']}")
```

```python
    print(f"Conditions (n): {pca_data['n']}")

    # Show how to split into baseline and damaged conditions
    signals = pca_data['signals']
    damage_states = pca_data['damage_states']

    # Extract baseline conditions (states 1-9)
    baseline_indices = np.where(damage_states <= 9)[0]
    damaged_indices = np.where(damage_states >= 10)[0]

    baseline_signals = signals[:, :, baseline_indices]
    damaged_signals = signals[:, :, damaged_indices]

    print(f"Baseline conditions: {baseline_signals.shape[2]} tests")
    print(f"Damaged conditions: {damaged_signals.shape[2]} tests")

except FileNotFoundError:
    print("3-story dataset required for outlier detection examples not avail
```

```
Example 1: Outlier Detection Data Loading
==================================================
Preprocessed signals shape: (8192, 4, 170)
Channels included: ['Ch2', 'Ch3', 'Ch4', 'Ch5']
Time points (t): 8192
Channels (m): 4
Conditions (n): 170
Baseline conditions: 90 tests
Damaged conditions: 80 tests
```

In [10]:

```python
# Example 2: Accessing specific damage states
print("\nExample 2: Accessing Specific Damage States")
print("=" * 50)

if 'pca_data' in locals():
    damage_states = pca_data['damage_states']
    state_descriptions = pca_data['state_descriptions']
    signals = pca_data['signals']

    # Access specific states
    target_states = [1, 10, 14]  # Baseline, first damage, severe damage

    for state in target_states:
        state_indices = np.where(damage_states == state)[0]
        state_signals = signals[:, :, state_indices]

        # Calculate RMS for each test in this state
        rms_values = np.sqrt(np.mean(state_signals**2, axis=0))  # RMS over
        mean_rms = np.mean(rms_values, axis=1)  # Mean RMS across tests

        print(f"State {state}: {state_descriptions[state]}")
        print(f"  Tests: {len(state_indices)}")
        print(f"  Mean RMS per channel: {mean_rms}")
        print()
```

```
Example 2: Accessing Specific Damage States
==================================================
State 1: Undamaged - Baseline condition
  Tests: 10
  Mean RMS per channel: [0.5081027  0.53241104 0.48763555 0.40463358]

State 10: Damaged - Gap = 0.20 mm
  Tests: 10
  Mean RMS per channel: [0.50694287 0.5337057  0.4894135  0.41107517]

State 14: Damaged - Gap = 0.05 mm
  Tests: 10
  Mean RMS per channel: [0.4825399  0.4678437  0.384145   0.38374212]
```

In [11]:
```python
# Example 3: Training/Testing splits commonly used in examples
print("Example 3: Common Training/Testing Splits")
print("=" * 50)

if 'pca_data' in locals():
    signals = pca_data['signals']
    damage_states = pca_data['damage_states']

    # Common split: Use baseline conditions for training
    baseline_indices = np.where(damage_states <= 9)[0]  # States 1-9
    damaged_indices = np.where(damage_states >= 10)[0]  # States 10-17

    training_signals = signals[:, :, baseline_indices]
    testing_signals = signals[:, :, np.concatenate([baseline_indices, damage

    # Create binary labels for testing (0=undamaged, 1=damaged)
    test_damage_states = damage_states[np.concatenate([baseline_indices, dam
    binary_labels = (test_damage_states >= 10).astype(int)

    print(f"Training set: {training_signals.shape[2]} undamaged conditions")
    print(f"Testing set: {testing_signals.shape[2]} total conditions")
    print(f"  - Undamaged: {np.sum(binary_labels == 0)} tests")
    print(f"  - Damaged: {np.sum(binary_labels == 1)} tests")

    # Alternative split: Use subset of each state for training
    print("\nAlternative split (subset training):")
    train_indices = []
    test_indices = []

    for state in range(1, 18):  # States 1-17
        state_indices = np.where(damage_states == state)[0]
        # Use first 7 tests for training, last 3 for testing
        train_indices.extend(state_indices[:7])
        test_indices.extend(state_indices[7:])

    train_indices = np.array(train_indices)
    test_indices = np.array(test_indices)

    print(f"Training set: {len(train_indices)} conditions from all states")
    print(f"Testing set: {len(test_indices)} conditions from all states")
```

```
Example 3: Common Training/Testing Splits
==================================================
Training set: 90 undamaged conditions
Testing set: 170 total conditions
  - Undamaged: 90 tests
  - Damaged: 80 tests

Alternative split (subset training):
Training set: 119 conditions from all states
Testing set: 51 conditions from all states
```

Dataset Integrity Validation

Automated validation of all datasets to ensure they're properly loaded and structured.

In [12]:
```python
# Run comprehensive dataset validation
print("Dataset Integrity Validation")
print("=" * 50)

validation_results = validate_dataset_integrity()

# Create summary table manually (without pandas)
print(f"{'Dataset':<30} {'Size (MB)':<10} {'Available':<10} {'Valid':<8} {'E
print("-" * 80)

for dataset_name, result in validation_results.items():
    dataset_info = get_available_datasets()[dataset_name]

    dataset_file = dataset_info['file']
    size_mb = dataset_info['size_mb']
    available = '✓' if result['available'] else '✗'
    valid = '✓' if result['valid'] else '✗'
    errors = len(result['errors'])
    warnings = len(result['warnings'])

    print(f"{dataset_file:<30} {size_mb:<10} {available:<10} {valid:<8} {err

# Show detailed errors/warnings if any
print("\nDetailed Issues:")
print("-" * 30)

issues_found = False
for dataset_name, result in validation_results.items():
    if result['errors'] or result['warnings']:
        issues_found = True
        dataset_info = get_available_datasets()[dataset_name]
        print(f"\n{dataset_info['file']}:")

        for error in result['errors']:
            print(f"  ERROR: {error}")
        for warning in result['warnings']:
            print(f"  WARNING: {warning}")
```

```
if not issues_found:
    print("No issues found. All available datasets are valid.")
```

```
Dataset Integrity Validation
==================================================
Dataset                        Size (MB)  Available  Valid    Errors   Warni
ngs
----------------------------------------------------------------------------
----
data3SS.mat                    25         ✓          ✓        0        0
data_CBM.mat                   54         ✓          ✓        0        0
data_example_ActiveSense.mat   32         ✓          ✓        0        0
dataSensorDiagnostic.mat       0.06       ✓          ✓        0        0
data_OSPExampleModal.mat       0.05       ✓          ✓        0        0

Detailed Issues:
------------------------------
No issues found. All available datasets are valid.
```

Dataset File Information

Detailed file information and download guidance.

In [13]:

```
# Show data directory and file information
data_dir = get_data_dir()
print(f"Data Directory: {data_dir}")
print(f"Directory exists: {data_dir.exists()}")
print()

if data_dir.exists():
    print("Files in data directory:")
    print("-" * 40)

    # List all .mat files
    mat_files = list(data_dir.glob('*.mat'))

    if mat_files:
        for mat_file in sorted(mat_files):
            size_mb = mat_file.stat().st_size / (1024**2)
            print(f"  {mat_file.name:30} ({size_mb:6.2f} MB)")
    else:
        print("  No .mat files found")

    # List other files
    other_files = [f for f in data_dir.iterdir() if f.is_file() and not f.na
    if other_files:
        print("\nOther files:")
        for other_file in sorted(other_files):
            print(f"  {other_file.name}")
else:
    print(f"Data directory does not exist: {data_dir}")
    print("Please create the directory and download the dataset files.")

print("\nDataset Download Information:")
print("-" * 40)
```

```python
print("All datasets are from the original SHMTools library (LA-CC-14-046)")
print("developed by Los Alamos National Laboratory.")
print("")
print("To obtain the datasets:")
print("1. Download from the original MATLAB SHMTools distribution")
print("2. Extract the .mat files from the Examples/ExampleData/ directory")
print(f"3. Place them in: {data_dir}")
print("")
print("See the README.md file in the data directory for detailed instructior
```

```
Data Directory: /Users/eric/repo/shm/shmtools-python/examples/data
Directory exists: True

Files in data directory:
----------------------------------------
  data3SS.mat                    ( 24.74 MB)
  dataSensorDiagnostic.mat       (  0.06 MB)
  data_CBM.mat                   ( 54.03 MB)
  data_OSPExampleModal.mat       (  0.05 MB)
  data_example_ActiveSense.mat   ( 31.63 MB)

Other files:
  .gitignore
  README.md

Dataset Download Information:
----------------------------------------
All datasets are from the original SHMTools library (LA-CC-14-046)
developed by Los Alamos National Laboratory.

To obtain the datasets:
1. Download from the original MATLAB SHMTools distribution
2. Extract the .mat files from the Examples/ExampleData/ directory
3. Place them in: /Users/eric/repo/shm/shmtools-python/examples/data

See the README.md file in the data directory for detailed instructions.
```

Summary

This notebook provides comprehensive dataset management utilities for SHMTools Python. Key takeaways:

Available Datasets

1. **data3SS.mat**: Primary 3-story structure dataset (25 MB)
2. **data_CBM.mat**: Condition-based monitoring rotating machinery (54 MB)
3. **data_example_ActiveSense.mat**: Guided wave measurements (32 MB)
4. **dataSensorDiagnostic.mat**: Sensor health monitoring (63 KB)
5. **data_OSPExampleModal.mat**: Modal analysis and sensor placement (50 KB)

Key Functions

- `load_3story_data()` : Primary structural dataset with detailed metadata
- `load_cbm_data()` : Rotating machinery with fault information
- `load_example_data(type)` : Convenient preprocessing for specific examples
- `validate_dataset_integrity()` : Automated validation and checking
- `check_data_availability()` : Quick availability status

Usage Patterns

- **Outlier Detection**: Use `load_example_data('pca')` for preprocessed 3-story data
- **Training/Testing**: Split by damage states or use subset sampling
- **Validation**: Run integrity checks before starting analysis
- **Exploration**: Use metadata and descriptions for understanding structure

The enhanced data loading utilities provide comprehensive documentation, metadata, and validation capabilities that simplify working with SHMTools datasets while maintaining compatibility with the original MATLAB examples.