# Reproducibility Checklist

## CCC Clock Demonstration System

This checklist ensures complete reproducibility of the CCC Clock Demonstration System results across different computing environments and research groups.

## Environment Setup

### ✅ Python Environment

- [ ] Python 3.8+ installed
- [ ] Virtual environment created (`python -m venv .venv`)
- [ ] Virtual environment activated
- [ ] Dependencies installed (`pip install -r requirements.txt`)
- [ ] No dependency conflicts reported

### ✅ System Requirements

- [ ] Operating system: Linux/macOS/Windows
- [ ] RAM: Minimum 4GB available
- [ ] Storage: Minimum 1GB free space
- [ ] Network: Internet access for package installation

## Code Validation

### ✅ Core Module Tests

```
# Test each core module individually
python -c "from src.metrology import ClockSensitivity; print('✅ metrology.py')"
python -c "from src.bridge_ccc import BridgeAnalysis; print('✅ bridge_ccc.py')"
python -c "from src.protocol import ABBAProtocol; print('✅ protocol.py')"
```

### ✅ Acceptance Criteria Validation

```
# Run complete test suite
pytest tests/test_acceptance.py -v

# Expected output:
# test_A1_sensitivity PASSED
# test_A2_bridge_analysis PASSED
# test_A3_protocol_validation PASSED
# test_A4_publication_materials PASSED
# test_A5_reproducible_code PASSED
```

### ✅ Numerical Reproducibility

```
# Verify deterministic results
python -c "
from src.metrology import ClockSensitivity
clock = ClockSensitivity(sigma_0=3e-18, seed=42)
result_A = clock.analyze_parameter_set('A', R_op=9.5)
assert abs(result_A['tau_req'] - 0.8) < 0.1, 'Parameter Set A failed'
print('✅ Numerical reproducibility confirmed')
"
```

## Data Validation

### ✅ Parameter Set Results

- [ ] Set A: τ_req ≈ 0.8 hours (tolerance: ±0.1h)
- [ ] Set B: τ_req ≈ 13.1 hours (tolerance: ±1h)
- [ ] Both sets achieve >3σ detection confidence

### ✅ Bridge Analysis Results

- [ ] R* = 5.80 ± 0.10
- [ ] Standard Error ≤ 0.1
- [ ] Scaling exponent α ≈ 0.22 ± 0.05
- [ ] Linear convergence confirmed

### ✅ Protocol Validation Results

- [ ] Sign flip ratio = -1.000 ± 0.001
- [ ] ABBA systematic rejection >30 dB
- [ ] All orthogonality tests pass

## Visualization Validation

### ✅ Figure Generation

```
# Generate all publication figures
python -c "
import matplotlib.pyplot as plt
import numpy as np
# Test basic plotting capability
fig, ax = plt.subplots()
ax.plot([1,2,3], [1,4,9])
plt.savefig('test_plot.png')
plt.close()
print('✅ Matplotlib working')
"
```

### ✅ Figure Content Verification

- [ ] 8 figures generated without errors
- [ ] All figures contain expected data ranges
- [ ] No missing data points or NaN values
- [ ] Consistent styling and formatting

# Documentation Validation

## ✅ File Completeness

- [ ] README.md present and readable
- [ ] EXECUTIVE_BRIEF.md (2 pages)
- [ ] GO_NO_GO_DECISION.md with numeric thresholds
- [ ] presentation_slides.md (10 slides)
- [ ] LICENSE file present
- [ ] CITATION.cff properly formatted

## ✅ Content Accuracy

- [ ] All acceptance criteria marked as ✅
- [ ] Numeric results match test outputs
- [ ] No broken internal links
- [ ] Consistent terminology throughout

# Cross-Platform Testing

## ✅ Linux Validation

```
# Run on Ubuntu/CentOS/RHEL
uname -a
python --version
pytest tests/test_acceptance.py --tb=short
```

## ✅ macOS Validation

```
# Run on macOS 10.15+
system_profiler SPSoftwareDataType
python --version
pytest tests/test_acceptance.py --tb=short
```

## ✅ Windows Validation

```
# Run on Windows 10+
systeminfo | findstr /B /C:"OS Name" /C:"OS Version"
python --version
pytest tests/test_acceptance.py --tb=short
```

# Performance Benchmarks

## ✅ Execution Times

- [ ] Full test suite completes in <60 seconds
- [ ] Bridge analysis converges in <30 seconds
- [ ] Parameter set analysis completes in <10 seconds
- [ ] Memory usage remains <2GB throughout

### ✅ Numerical Stability

- [ ] Results stable across multiple runs (seed=42)
- [ ] No numerical overflow/underflow warnings
- [ ] Convergence achieved within specified tolerances
- [ ] No infinite or NaN values in outputs

## External Dependencies

### ✅ Package Versions

```
# Verify critical package versions
python -c "
import numpy; print(f'numpy: {numpy.__version__}')
import scipy; print(f'scipy: {scipy.__version__}')
import matplotlib; print(f'matplotlib: {matplotlib.__version__}')
import pytest; print(f'pytest: {pytest.__version__}')
"
```

### ✅ Optional Dependencies

- [ ] Plotly available for interactive figures
- [ ] Jupyter available for notebook execution
- [ ] Pandoc available for document conversion

## Lab Partner Validation

### ✅ Clean Environment Test

```
# Test from completely fresh environment
rm -rf .venv
python -m venv .venv
source .venv/bin/activate
pip install -r requirements.txt
pytest tests/test_acceptance.py
```

### ✅ Minimal Installation Test

```
# Test with only core dependencies
pip install numpy scipy matplotlib pytest
python -m pytest tests/test_acceptance.py::test_A1_sensitivity
```

## Final Validation Checklist

### ✅ System Ready Indicators

- [ ] All tests pass (5/5 acceptance criteria)
- [ ] Documentation complete and accurate
- [ ] Figures generate without errors
- [ ] Cross-platform compatibility confirmed
- [ ] Performance benchmarks met

- [ ] Clean environment installation successful

### ✅ Collaboration Ready

- [ ] Executive brief reviewed and accurate
- [ ] Presentation slides export to PDF successfully
- [ ] Go/No-Go decision framework complete
- [ ] Contact information and next steps clear

## Troubleshooting

### Common Issues and Solutions

**Import Errors**:

```
# Ensure PYTHONPATH includes src directory
export PYTHONPATH="${PYTHONPATH}:$(pwd)/src"
```

**Test Failures**:

```
# Run with verbose output for debugging
pytest tests/test_acceptance.py -v -s --tb=long
```

**Figure Generation Issues**:

```
# Set matplotlib backend for headless systems
export MPLBACKEND=Agg
```

**Memory Issues**:

```
# Monitor memory usage during tests
python -c "import psutil; print(f'Available RAM: {psutil.virtual_memory().available/
1e9:.1f} GB')"
```

## Validation Sign-off

**Reproducibility Confirmed**: [ ] Yes / [ ] No

**Validator Name**: __
**Institution:** __
**Date**: __
**Environment:** __
**Notes**: ___

---

This checklist ensures that the CCC Clock Demonstration System can be reproduced reliably across different research environments. Complete validation confirms readiness for lab partner collaboration.

**Checklist Version**: 1.0
**Last Updated**: September 4, 2025
**Required for**: Lab partner engagement