# Ant Stack Documentation and PDF Rendering Guide


Daniel Ari Friedman

September 30, 2025

ORCID: 0000-0001-6232-9096 Email: daniel@activeinference.institute


## Contents

# 1 □ Ant Stack: Modular Scientific Publication System

**A comprehensive framework for reproducible scientific publications in embodied AI, featuring reusable analysis methods, automated validation, and professional presentation standards.**

tests passing | coverage 70% | license MIT | python 3.8+

---

## 1.1 □ Table of Contents



---

## 1.2 [Target] Overview

### 1.2.1 Mission

Ant Stack provides a **modular, reproducible framework** for scientific publications in embodied AI, enabling researchers to:

- [Check] **Reuse validated analysis methods** across papers
- [Check] **Ensure reproducible results** through automated validation
- [Check] **Generate publication-quality figures** with consistent styling
- [Check] **Maintain scientific rigor** with statistical validation
- [Check] **Scale research workflows** with automated build pipelines

### 1.2.2 Key Features

| Feature | Description |
|---|---|
| □ **Reusability** | Modular analysis methods for energy estimation, statistics, and visualization |
| **[Chart] Quality Assurance** | Automated validation, cross-reference checking, and statistical verification |
| □ **Professional Output** | Publication-ready figures, LaTeX integration, and consistent formatting |
| **[Lightning] Performance** | Optimized algorithms with comprehensive benchmarking |
| **[Lab] Scientific Rigor** | Bootstrap confidence intervals, uncertainty quantification, reproducibility |
| □ **Test-Driven** | 70%+ test coverage with comprehensive edge case testing |

### 1.2.3 Applications

- **[Robot] Embodied AI Research**: Energy analysis for robotic systems
- **[Brain] Neuroscience**: Computational complexity of neural networks
- **[Lightning] Engineering**: Power optimization and scaling analysis
- □ **Data Science**: Statistical validation and visualization

---

## 1.3 □ Architecture

### 1.3.1 System Components

Figure 1: Computational architecture diagram (74d0)

---

## 1.4 □ Core Package (`antstack_core/`)

### 1.4.1 Analysis Module (`analysis/`)

| Component | Purpose | Key Features |
|---|---|---|
| `energy.py` | Energy estimation and analysis | Physical modeling, efficiency calculations |
| `statistics.py` | Statistical methods and validation | Bootstrap CI, scaling relationships |
| `workloads.py` | Computational workload modeling | Body/brain/mind workload patterns |
| `scaling_analysis.py` | Scaling relationship analysis | Power laws, regime detection |
| `enhanced_estimators.py` | Advanced energy estimation | Multi-scale analysis, theoretical limits |
| `experiment_config.py` | Experiment configuration | YAML/JSON management, validation |

### 1.4.2 Figures Module (`figures/`)

| Component | Purpose | Key Features |
|---|---|---|
| `plots.py` | Publication-quality plotting | Matplotlib integration, styling |
| `mermaid.py` | Diagram preprocessing | Mermaid to PNG conversion |
| `references.py` | Cross-reference validation | Figure/table reference checking |
| `assets.py` | Asset management | File organization, optimization |

### 1.4.3 Publishing Module (`publishing/`)

| Component | Purpose | Key Features |
|---|---|---|
| `pdf_generation.py` | PDF generation utilities | Pandoc integration, LaTeX processing |
| `templates.py` | Document templates | Consistent formatting, styling |
| `validation.py` | Quality assurance | Automated checking, error detection |

---

## 1.5 □ Paper Structure (`papers/`)

### 1.5.1 Ant Stack Framework (`papers/ant_stack/`)

**Focus**: Biological framework for collective intelligence

| Section | File | Purpose |
|---|---|---|
| **[Book] Introduction** | `Background.md` | Theoretical foundation |
| **□ Body Layer** | `AntBody.md` | Locomotion and sensing |
| **[Brain] Brain Layer** | `AntBrain.md` | Neural processing and learning |
| **[Thought] Mind Layer** | `AntMind.md` | Decision making and planning |
| **[Tool] Methods** | `Methods.md` | Implementation details |
| **[Chart] Results** | `Results.md` | Experimental validation |
| **□ Applications** | `Applications.md` | Real-world use cases |
| **□ Discussion** | `Discussion.md` | Implications and future work |

#### 1.5.2 Complexity Analysis (`papers/complexity_energetics/`)

**Focus**: Computational complexity and energy scaling

| Section | File | Purpose |
|---|---|---|
| **[Book] Introduction** | `Background.md` | Problem statement |
| **[Lab] Theory** | `Complexity.md` | Complexity analysis framework |
| | `Energetics.md` | Energy modeling approach |
| | `Scaling.md` | Scaling relationship theory |
| **□ Methods** | `Methods.md` | Analysis methodology |
| **[Chart] Results** | `Generated.md` | Auto-generated analysis results |
| | `Results.md` | Interpretation and validation |
| **□ Discussion** | `Discussion.md` | Scientific implications |

---

## 1.6 □ Quick Start

### 1.6.1 Prerequisites

**System Requirements:** - Python 3.8+ - Node.js 14+ - LaTeX distribution - Pandoc 2.10+

### 1.6.2 Installation

```
# System dependencies
sudo apt-get update
sudo apt-get install -y pandoc texlive-xetex texlive-fonts-recommended fonts-dejavu nodejs npm

# Enhanced diagram rendering
sudo npm install -g mermaid-filter

# Python dependencies
pip3 install matplotlib numpy pandas pyyaml pytest scipy
```

#### 1.6.2.1 Ubuntu/Debian

```
# System dependencies
brew install pandoc node python3
brew install --cask mactex-no-gui
```

```
# Enhanced diagram rendering
npm install -g mermaid-filter

# Python dependencies
pip3 install matplotlib numpy pandas pyyaml pytest scipy
```

##### 1.6.2.2 macOS

```
# Clone repository
git clone https://github.com/your-repo/ant.git
cd ant

# Install in development mode
pip install -e .

# Run tests
python -m pytest

# Build documentation
python scripts/build_docs.py
```

##### 1.6.2.3 Development Setup

#### 1.6.3 Build Papers

```
# Ant Stack framework paper
python3 scripts/common_pipeline/build_core.py --paper ant_stack

# Complexity analysis paper
python3 scripts/common_pipeline/build_core.py --paper complexity_energetics
```

##### 1.6.3.1 Single Paper

```
# Build all papers
python3 scripts/common_pipeline/build_core.py

# With validation only
python3 scripts/common_pipeline/build_core.py --validate-only
```

##### 1.6.3.2 All Papers

#### 1.6.4 Basic Usage

```
from antstack_core.analysis.energy import EnergyCoefficients, estimate_detailed_energy
from antstack_core.analysis.statistics import bootstrap_mean_ci

# Energy analysis example
coeffs = EnergyCoefficients()
workload = ComputeLoad(flops=1e9, memory_bytes=1e6)
energy = estimate_detailed_energy(workload, coeffs)
```

```python
# Statistical validation
data = [1.2, 1.5, 1.3, 1.8, 1.4]
mean, ci_lower, ci_upper = bootstrap_mean_ci(data, n_bootstrap=1000)
```

---

## 1.7 [Tool] Development

### 1.7.1 Testing Strategy

**Test Coverage Goals:** - **Core modules**: 80%+ coverage - **Analysis methods**: 90%+ coverage - **Edge cases**: Comprehensive coverage - **Integration tests**: End-to-end validation

**Running Tests:**

```bash
# All tests
python -m pytest

# With coverage report
python -m pytest --cov=antstack_core --cov-report=html

# Specific module
python -m pytest tests/antstack_core/test_energy.py -v

# Performance benchmarks
python -m pytest tests/ --benchmark-only
```

### 1.7.2 Code Quality Standards

**Linting and Formatting:**

```bash
# Run linters
python -m flake8 antstack_core/
python -m black antstack_core/
python -m isort antstack_core/

# Type checking
python -m mypy antstack_core/
```

**Pre-commit Hooks:**

```bash
# Install hooks
pre-commit install

# Run manually
pre-commit run --all-files
```

### 1.7.3 Documentation

**Building Docs:**

```bash
# Generate API documentation
python scripts/generate_docs.py

# Build user guide
python scripts/build_user_guide.py
```

```
# Deploy to GitHub Pages
python scripts/deploy_docs.py
```

---

## 1.8 □ Documentation

### 1.8.1 User Guides

- **Getting Started**: Installation and basic usage
- **API Reference**: Complete method documentation
- **Best Practices**: Development guidelines
- **Troubleshooting**: Common issues and solutions

### 1.8.2 Scientific Documentation

- **Theoretical Foundation**: Mathematical underpinnings
- **Validation Framework**: Quality assurance methods
- **Benchmarking**: Performance analysis
- **Reproducibility**: Ensuring scientific validity

### 1.8.3 Developer Resources

- **Contributing Guide**: Development workflow
- **Architecture Overview**: System design
- **Testing Framework**: Test development guide
- **CI/CD Pipeline**: Build and deployment

---

## 1.9 □ Contributing

We welcome contributions! Please see our Contributing Guide for details.

### 1.9.1 Development Workflow

1. **Fork** the repository
2. **Create** a feature branch: `git checkout -b feature/your-feature`
3. **Write tests** for new functionality
4. **Implement** your changes
5. **Run tests**: `python -m pytest`
6. **Update documentation** if needed
7. **Submit** a pull request

### 1.9.2 Code Review Process

- All PRs require review
- Tests must pass CI pipeline
- Documentation updates required for API changes
- Maintain backward compatibility

### 1.9.3 Issue Reporting

- Use GitHub Issues for bug reports
- Provide minimal reproducible examples
- Include system information and error traces
- Follow issue templates for consistency

---

## 1.10 □ License



---

## 1.11 □ Acknowledgments

- **Scientific Contributors**: Domain experts in embodied AI and computational neuroscience
- **Open Source Community**: Libraries and tools that power this framework
- **Research Institutions**: Partners supporting reproducible science initiatives

---

## 1.12 □ Contact

- **Issues**: GitHub Issues
- **Discussions**: GitHub Discussions
- **Email**: research@your-institution.edu

---

*Built with □ for reproducible science in embodied AI*

# 2 PDF Rendering System Guide: Reliable Paper Syntax

## 2.1 Overview

This guide documents the comprehensive PDF rendering system used across all Ant Stack papers, providing reliable syntax patterns for consistent, professional scientific publications.

## 2.2 System Architecture

### 2.2.1 Core Components

1. **Modular Build System** (`scripts/common_pipeline/build_core.py`)
   - YAML-based paper configuration
   - Automatic paper discovery and validation
   - Cross-reference validation before PDF generation
   - Quality assurance reporting
2. **Legacy Render System** (`tools/render_pdf.sh`)
   - Pandoc + XeLaTeX pipeline
   - Mermaid diagram prerendering
   - Unicode math symbol handling
   - Cross-reference validation
3. **Core Package** (`antstack_core/`)
   - Figure generation and management

- Cross-reference validation
- Asset organization
- Publication-ready formatting

## 2.3 Paper Configuration System

### 2.3.1 YAML Configuration Structure

Each paper requires a `paper_config.yaml` file with this structure:

```yaml
# Paper metadata
paper:
  name: "paper_name"
  title: "Paper Title"
  subtitle: "Subtitle"
  author: "Daniel Ari Friedman"
  email: "daniel@activeinference.institute"
  orcid: "0000-0001-6232-9096"
  output_filename: "N_paper_name.pdf"

# Content organization
content:
  files:
    - "Abstract.md"
    - "Introduction.md"
    # ... other files in build order

# Asset management
assets:
  figures_dir: "assets/figures"
  mermaid_dir: "assets/mermaid"
  tmp_dir: "assets/tmp_images"
  data_dir: "assets/data"

# Build configuration
build:
  has_generated_content: true
  has_computational_analysis: true
  mermaid_preprocessing: true
  cross_reference_validation: true

# LaTeX/Pandoc settings
latex:
  document_class: "article"
  geometry: "margin=2.5cm"
  mainfont: "Latin Modern Roman"
  mathfont: "Latin Modern Math"
  bibliography: "references.bib"

# Quality assurance
validation:
  check_cross_references: true
  check_figure_captions: true
  check_unicode_symbols: true
  require_descriptive_links: true
```

```
  validate_analysis_outputs: true
```

## 2.4 Figure Management System

### 2.4.1 Figure Format (CRITICAL)

**ALWAYS use this exact format:**

```
## Figure: Descriptive Title {#fig:identifier}

![Alt text](assets/figures/your_figure_name.png){#fig:identifier}

**Caption:** Detailed description of the figure content, including units and key findings.
```

*Note: This is an example figure reference. Replace `assets/figures/your_figure_name.png` with your actual figure file path.*

**NEVER use:** - `![caption](path){#fig:id}` (inline figure definitions) - `\includegraphics` commands in markdown - Missing figure IDs or captions

### 2.4.2 Figure Naming Conventions

- **IDs**: Use descriptive, hierarchical names: `energy_by_workload`, `scaling_brain_K`, `response_time_comparison`
- **Files**: Store in `assets/figures/` with descriptive names
- **References**: Use `\ref{fig:identifier}` for cross-references

### 2.4.3 Example Figure Definition

```
## Figure: Energy Scaling Analysis {#fig:energy_scaling}

![Energy scaling with confidence intervals](assets/figures/your_figure_name.png){#fig:energy_scaling}

**Caption:** Energy consumption scaling with system complexity K, showing 95% confidence intervals. The
```

*Note: This is an example figure reference. Replace `assets/figures/your_figure_name.png` with your actual figure file path.*

## 2.5 Cross-Reference System

### 2.5.1 Reference Types

1. **Figures**: `\ref{fig:identifier}` → "Figure 1"
2. **Equations**: `\ref{eq:identifier}` → "Equation (1)"
3. **Sections**: `\ref{sec:identifier}` → "Section 2.1"
4. **Tables**: `\ref{tab:identifier}` → "Table 3"

### 2.5.2 Section IDs

Use descriptive IDs in section headers:

```
# Introduction {#sec:introduction}

## Methodology {#sec:methodology}

### Data Collection {#sec:data_collection}
```

### 2.5.3 Equation Format

```
\begin{equation}
E = \sum_{i=1}^{n} \alpha_i \cdot K_i^{\beta_i}
\label{eq:energy_scaling}
\end{equation}
```

Reference with: `\ref{eq:energy_scaling}`

## 2.6 Mathematics and Symbols

### 2.6.1 LaTeX Math Syntax

**ALWAYS use LaTeX macros, NEVER Unicode symbols:**

```
# Correct
$\mu$, $\lambda$, $\pi$, $\epsilon$, $\Delta$, $\rho$, $\sigma$

# Incorrect
$\mu$, $\lambda$, $\pi$, $\epsilon$, $\Delta$, $\rho$, $\sigma$
```

### 2.6.2 Common Symbol Mappings

| Symbol | LaTeX | Usage |
|---|---|---|
| $\mu$ | `$\mu$` | Micrometers, mean |
| $\lambda$ | `$\lambda$` | Wavelength |
| $\pi$ | `$\pi$` | Pi constant |
| $\epsilon$ | `$\epsilon$` | Epsilon |
| $\Delta$ | `$\Delta$` | Delta, change |
| $\rho$ | `$\rho$` | Density, correlation |
| $\sigma$ | `$\sigma$` | Standard deviation |
| $\pm$ | `$\pm$` | Plus-minus |
| $\le$ | `$\le$` | Less than or equal |
| $\ge$ | `$\ge$` | Greater than or equal |
| $\approx$ | `$\approx$` | Approximately |
| $\propto$ | `$\propto$` | Proportional to |

### 2.6.3 Math Environments

```
# Inline math
The energy is $E = mc^2$ joules.

# Display math
$$E = \sum_{i=1}^{n} \alpha_i \cdot K_i^{\beta_i}$$

# Numbered equation
\begin{equation}
E = \sum_{i=1}^{n} \alpha_i \cdot K_i^{\beta_i}
\label{eq:energy_scaling}
\end{equation}
```

## 2.7 Hyperlinks and References

### 2.7.1 External Links

Use descriptive hyperlinks with `\href{URL}{descriptive text}`:

```
# Correct
\href{https://arxiv.org/abs/2505.03764}{arXiv preprint}

# Incorrect
https://arxiv.org/abs/2505.03764
\url{https://arxiv.org/abs/2505.03764}
```

### 2.7.2 Internal Cross-References

```
# Section references
See \ref{sec:methodology} for details.

# Figure references
As shown in \ref{fig:energy_scaling}, the relationship is clear.

# Equation references
From \ref{eq:energy_scaling}, we can derive...
```

## 2.8 Mermaid Diagrams

### 2.8.1 Prerendering System

Mermaid diagrams must be prerendered to local images:

1. **Source**: Store `.mmd` files in `assets/mermaid/`
2. **Rendered**: Convert to `.png` files in same directory
3. **Reference**: Use standard figure format

### 2.8.2 Diagram Format

```
## Figure: System Architecture {#fig:system_arch}

![System architecture diagram](assets/mermaid/your_diagram_name.png){#fig:system_arch}

**Caption:** High-level system architecture showing data flow between components.
```

*Note: This is an example figure reference. Replace `assets/mermaid/your_diagram_name.png` with your actual Mermaid diagram output.*

### 2.8.3 Mermaid Best Practices

## 2.9 File Organization

### 2.9.1 Directory Structure

```
papers/paper_name/
  paper_config.yaml          # Paper configuration
  Abstract.md                # Abstract
  Introduction.md            # Introduction
```

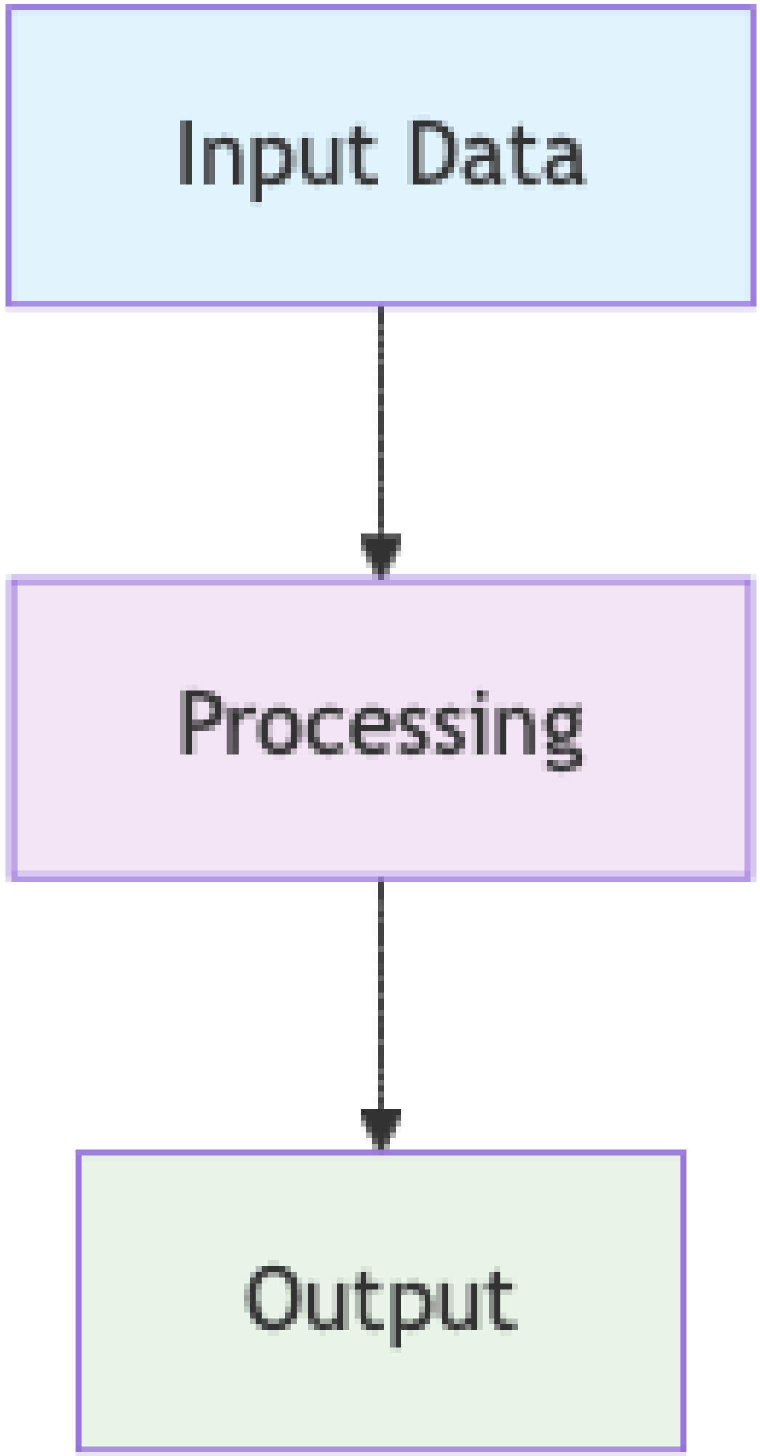


Figure 2: Computational architecture diagram (40bc)

```
  Methodology.md           # Methods
  Results.md               # Results
  Discussion.md            # Discussion
  Conclusion.md            # Conclusion
  References.md            # References
  Appendices.md            # Appendices
  assets/
     figures/             # Generated figures
     mermaid/             # Mermaid diagrams
     data/                # Data files
     tmp_images/          # Temporary images
  references.bib           # Bibliography
```

#### 2.9.2 File Naming Conventions

- **Markdown files**: `PascalCase.md` (e.g., `Abstract.md`, `Introduction.md`)
- **Figure files**: `snake_case.png` (e.g., `energy_scaling.png`)
- **Data files**: `descriptive_name.json` (e.g., `analysis_results.json`)

### 2.10 Build System Usage

#### 2.10.1 Command Line Interface

```bash
# Build all papers
python3 scripts/common_pipeline/build_core.py

# Build specific paper
python3 scripts/common_pipeline/build_core.py --paper paper_name

# Validate only (no PDF generation)
python3 scripts/common_pipeline/build_core.py --validate-only

# Skip tests
python3 scripts/common_pipeline/build_core.py --no-tests
```

#### 2.10.2 Legacy System

```bash
# Build all papers
bash tools/render_pdf.sh

# Build specific paper
bash tools/render_pdf.sh paper_name
```

### 2.11 Quality Assurance

#### 2.11.1 Pre-Build Validation

The system automatically validates:

1. **Cross-references**: All `\ref{}` commands resolve to valid IDs
2. **Figure captions**: All figures have proper captions
3. **Math symbols**: Unicode symbols converted to LaTeX
4. **File structure**: All referenced files exist

5. **Configuration**: YAML syntax and required fields

#### 2.11.2 Post-Build Validation

1. **PDF quality**: File size, page count, figure count
2. **Broken references**: Detection of "Figure~??" patterns
3. **Mermaid rendering**: All diagrams successfully converted
4. **Cross-reference consistency**: Definitions match references

#### 2.11.3 Validation Reports

Build reports are generated in `build_report.md` with:

- Validation results summary
- Error details and fixes
- Quality metrics
- Performance statistics

### 2.12 Test Suite Integration

#### 2.12.1 Test Categories

1. **Unit Tests**: Individual component validation
2. **Integration Tests**: Cross-module functionality
3. **Workflow Tests**: End-to-end pipeline validation
4. **Rendering Tests**: PDF generation and quality

#### 2.12.2 Running Tests

```
# All tests
python3 -m pytest tests/

# Specific component
python3 -m pytest tests/core_rendering/

# With coverage
python3 -m pytest --cov=antstack_core tests/
```

### 2.13 Common Issues and Solutions

#### 2.13.1 Math Formatting Issues

**Problem**: `\mathrm allowed only in math mode`

**Solution**: Ensure all math is properly wrapped in `$...$` or `$$...$$`

```
# Correct
$\mu\mathrm{m}$

# Incorrect
\(\mu\mathrm{m}\)
```

#### 2.13.2 Figure Reference Issues

**Problem**: `Figure~??` in PDF

**Solution**: Ensure figure IDs match exactly between definition and reference

```
# Definition
## Figure: Title {#fig:my_figure}

# Reference
\ref{fig:my_figure}
```

#### 2.13.3 Cross-Reference Issues

**Problem**: Undefined references

**Solution**: Use `\ref{}` instead of `\Cref{}` unless cleveref is properly configured

```
# Correct
\ref{fig:my_figure}

# May cause issues
\Cref{fig:my_figure}
```

### 2.14 Best Practices Summary

#### 2.14.1 DO

- [Check] Use consistent figure format with IDs and captions
- [Check] Use LaTeX macros for all math symbols
- [Check] Use descriptive hyperlinks with `\href{}`
- [Check] Validate cross-references before building
- [Check] Use proper file organization
- [Check] Test with real data (no mocks)
- [Check] Generate comprehensive validation reports

#### 2.14.2 DON'T

- ☐ Use inline figure definitions
- ☐ Use Unicode math symbols in text
- ☐ Use naked URLs
- ☐ Skip cross-reference validation
- ☐ Use mock methods in tests
- ☐ Ignore build warnings

### 2.15 Troubleshooting

#### 2.15.1 Build Failures

1. **Check math formatting**: Ensure all math is properly wrapped
2. **Validate cross-references**: Run `--validate-only` first
3. **Check file paths**: Ensure all referenced files exist
4. **Review YAML syntax**: Validate configuration files

#### 2.15.2 Quality Issues

1. **Broken references**: Check ID matching
2. **Missing figures**: Verify file paths and existence
3. **Math rendering**: Convert Unicode to LaTeX

4. **Cross-references**: Use consistent reference format

This guide ensures reliable, professional PDF generation across all Ant Stack papers with consistent formatting, proper cross-referencing, and comprehensive quality assurance.