Primordial Photon–Dark Photon Entanglement: Full Repo Files

This includes fully populated example scripts, Jupyter notebooks, and `/docs/` content for immediate use. Everything is formatted for GitHub and ready to commit.

---

1. Example `/validation/` Scripts

null_tests.py

```
"""
Null signal generator for Photon-Dark Photon entanglement tests.
Generates randomized data to ensure pipeline is robust to no-signal input.
"""
import numpy as np

def generate_null_map(shape=(100,100), seed=42):
    np.random.seed(seed)
    return np.random.normal(0, 1, size=shape)

if __name__ == '__main__':
    null_map = generate_null_map()
    np.savetxt('validation/benchmark_results/null_map.csv', null_map,
delimiter=',')
    print('Null map saved to benchmark_results/null_map.csv')
```

injection_recovery.py

```
"""
Injection and recovery test for synthetic photon-dark photon entanglement
signals.
"""
import numpy as np
from null_tests import generate_null_map

def inject_signal(null_map, amplitude=1.0, location=(50,50)):
    map_copy = null_map.copy()
    map_copy[location] += amplitude
    return map_copy

if __name__ == '__main__':
```

```
    null_map = generate_null_map()
    test_map = inject_signal(null_map)
    np.savetxt('validation/benchmark_results/injection_map.csv', test_map,
delimiter=',')
    print('Injection map saved to benchmark_results/injection_map.csv')
```

noise_model.py

```
"""
Instrumental noise modeling for photon-dark photon analysis.
"""
import numpy as np

def add_gaussian_noise(data, sigma=0.05):
    noise = np.random.normal(0, sigma, size=data.shape)
    return data + noise
```

---

2. Example Jupyter Notebook `/notebooks/Photon_DarkPhoton_Cluster_Analysis.ipynb`

```
# Photon-Dark Photon Cluster Analysis Notebook
import numpy as np
import matplotlib.pyplot as plt
from validation.null_tests import generate_null_map
from validation.injection_recovery import inject_signal
from validation.noise_model import add_gaussian_noise

# Load / generate data
null_map = generate_null_map()
signal_map = inject_signal(null_map, amplitude=2.0, location=(50,50))
noisy_map = add_gaussian_noise(signal_map, sigma=0.1)

# Plot results
plt.imshow(noisy_map, cmap='viridis')
plt.colorbar(label='Signal Strength')
plt.title('Photon-Dark Photon Entanglement Map (Synthetic Example)')
plt.show()
```

---

3. `/docs/` Content Examples

theory.md

```
# Photon-Dark Photon Theory

This document summarizes the theoretical framework behind Photon-Dark Photon
entanglement.

## Kinetic Mixing
- Standard photon can mix with a dark photon via a small kinetic term.
- Coupling parameter: epsilon ~ 10^-12 to 10^-3 (example).

## Entanglement Hypothesis
- Photon emitted from early universe can maintain entanglement with dark photon.
- Observable via correlated polarization or intensity anomalies in astrophysical
sources.

## Observable Signatures
- Interstellar comet spectra anomalies
- Cluster lensing maps (e.g., Abell 1689)
- Polarization patterns beyond classical scattering predictions
```

data.md

```
# Data Sources & Provenance

- **HST / JWST Data:** Download from MAST (MAST Portal)
- **Example Cluster:** Abell 1689 (coordinates: RA 13:11:29, Dec -01:20:28)
- **Interstellar Comets:** 2I/Borisov, 1I/'Oumuamua

## FITS Structure
- Calibrated FITS images required
- WCS metadata needed for reprojection
```

workflow.md

```
# Pipeline Workflow

1. Download FITS data from MAST / JWST archive
2. Reproject all images to common WCS grid
3. Run physics validation tests
4. Generate entanglement maps
```

```
5. Null tests and injection recovery for verification
6. Save outputs and metadata
```

api.md

```
# API Reference

## `generate_null_map(shape=(100,100), seed=42)`
Generate randomized null map for validation.

## `inject_signal(null_map, amplitude=1.0, location=(50,50))`
Inject synthetic signal into null map.

## `add_gaussian_noise(data, sigma=0.05)`
Add Gaussian noise to map data.
```

---

4. Community & Contributor Tools Examples

CONTRIBUTING.md

```
# Contributing to Primordial Photon-Dark Photon Entanglement

Please follow the Contributor License Agreement (CLA) in `CLA.md`.

## Pull Request Process
- Fork the repo
- Make changes in a feature branch
- Submit a pull request with tests / validation examples
- Contributions added to `credits.json`
```

CODE_OF_CONDUCT.md

```
# Code of Conduct

All participants are expected to be professional, respectful, and constructive.
```

CLA.md

```
# Contributor License Agreement

By contributing, you agree that your contributions are under the project
```

```
license,
allowing the project owner to maintain IP and control licensing.
```

AUTHORS.md (example)

```
# Authors & Contributors

- Tony E. Ford (Lead Researcher)
- Contributor1 (Code, Weight 1.0)
- Contributor2 (Data, Weight 0.5)
```

credits.json (template)

```
[
  {"username": "TonyFord", "type": "lead", "weight": 1.0, "timestamp":
"2025-11-16"},
  {"username": "Contributor1", "type": "code", "weight": 1.0, "timestamp":
"2025-11-16"},
  {"username": "Contributor2", "type": "data", "weight": 0.5, "timestamp":
"2025-11-16"}
]
```

---

5. Citation / Funding Templates

CITATION.cff

```
cff-version: 1.2.0
title: "Primordial Photon-Dark Photon Entanglement"
authors:
  - family-names: Ford
    given-names: Tony E.
date-released: 2025-11-16
version: 1.0.0
url: "https://github.com/tlcagford/Primordial-Photon-Dark-Photon-Entanglement"
```

FUNDING.yml

```
sponsors:
  - name: "Independent Research"
    url: "https://tlcagford.com"
```

```
  - name: "Example Grant Agency"
    url: "https://example.org/grants"
```

This fully populates the repository with **ready-to-use scripts, notebooks, docs, and contributor system**, suitable for **GitHub publication, reproducibility, and collaboration**.