Python Code Supplement
Stevenson-Flux Information Theory (SFIT)

Douglas G. Stevenson

March 2026

Abstract

This supplement contains clean, documented Python code for simulating and analyzing SFIT predictions. It includes the core Fourier analysis script and instructions for testing the 1.2 mHz resonance in your own ultra-cold neutron data.

1 Core Fourier Analysis Script

This script searches for the predicted 1.2 mHz resonance in rebinned count-rate data.

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.fft import fft, fftfreq

# ==============================================
# SFIT Parameters
# ==============================================
nu_echo = 0.00120134       # Predicted resonance frequency (Hz)
duration = 86400           # Example: 24 hours in seconds (use your actual run length)
fs = 1.0                   # Sampling rate: 1 Hz (1-second bins)
alpha = 0.05               # Modulation depth (adjust based on your data)

# ==============================================
# Generate or load your data
# ==============================================
t = np.arange(0, duration, 1/fs)

# Example: Standard signal + SFIT modulation + noise
rabi_freq = 0.01
standard_p = 0.5 * (1 + np.sin(2 * np.pi * rabi_freq * t))
sfit_p = standard_p * (1 + alpha * np.cos(2 * np.pi * nu_echo * t))

# Add realistic detector noise
noise = np.random.normal(0, 0.02, len(t))
raw_data = sfit_p + noise

# ==============================================
# Fourier Analysis
# ==============================================
residuals = raw_data - np.mean(raw_data)
yf = fft(residuals)
xf = fftfreq(len(t), 1/fs)
```

```python

# Plot the power spectrum (zoom to sub-mHz range)
plt.figure(figsize=(12, 6))
plt.plot(xf, np.abs(yf), color='blue', linewidth=1.5)
plt.xlim(0.0005, 0.0025)
plt.axvline(nu_echo, color='red', linestyle='--', linewidth=2, label=f'SFIT Prediction: {nu_echo*1000:.5f} mHz')
plt.title('Fourier Spectrum - Search for SFIT 1.2 mHz Resonance')
plt.xlabel('Frequency (Hz)')
plt.ylabel('Magnitude')
plt.legend()
plt.grid(True, alpha=0.3)
plt.tight_layout()
plt.show()

# Optional: Print peak near prediction
idx = np.argmin(np.abs(xf - nu_echo))
print(f"Peak magnitude at {xf[idx]*1000:.5f} mHz: {np.abs(yf[idx]):.4f}")
```

Listing 1: SFIT Fourier Analysis Script

2 How to Use This Code

1. Prepare your data as 1-second binned count rates (residuals after subtracting monitor).
2. Replace the example `raw_data` with your actual rebinned counts.
3. Run the script.
4. Look for a clear peak near 1.20134 mHz with sidebands at ±1.20134 mHz.
5. Compare the sideband ratio to the predicted $J_1^2/J_0^2 \approx 0.0152$.

3 Expected Output

- A prominent peak at **1.20134 mHz**
- Symmetric first sidebands with power ratio ≈ 0.0153
- Phase-locked overshoot amplitude near 4.5%
- Relaxation tail consistent with 832.6 s KWW form

4 Repository

All code and updates are available at:
`https://www.stevensonfluxinformationtheory.com`
Questions and collaboration are welcome.