Phys332W

February 2, 2026

[6]:
```python
import pandas as pd
import numpy as np
from scipy import stats
import sys
import os

def main():
    # Try to locate file relative to script, otherwise use CWD
    # Path Resolution (Environment Agnostic)
    target_file = 'SF - B2 Data (3).xlsx'

    # Places to look: Current dir, Script dir (if known)
    search_dirs = [os.getcwd()]
    if '__file__' in globals():
        search_dirs.append(os.path.dirname(os.path.abspath(__file__)))

    filename = None
    for d in search_dirs:
        candidate = os.path.join(d, target_file)
        if os.path.exists(candidate):
            filename = candidate
            break

    if filename is None:
        print(f"ERROR: Could not find '{target_file}'")
        print(f"Searched in: {', '.join(set(search_dirs))}")

        print(f"\nAvailable .xlsx files in {os.getcwd()}:")
        try:
            files = [f for f in os.listdir(os.getcwd()) if f.endswith('.xlsx')]
            if files:
                for f in files:
                    print(f"  - {f}")
            else:
                print("  (No .xlsx files found in current directory)")
        except Exception as e:
            print(f"  Error listing directory: {e}")
```

```python
        return

    print(f"Loading data from {filename}...")
    # Load data with header at row 2 (0-indexed), identifying correct columns
    try:
        df = pd.read_excel(filename, header=2)
    except Exception as e:
        print(f"Error loading file: {e}")
        return

    # Identify datasets
    # Look for pairings of "Column" and "Column - Model"
    datasets = []
    columns = df.columns

    # Common keys expected based on file content
    potential_keys = ['A', 'B', 'C', 'D', 'E', 'F', 'H', 'I', 'J', 'K3', 'K5']

    for key in potential_keys:
        # Check for model column
        # Matches formats like "A - Model" or "K5 (-5, ..., +5) - Model"
        model_col = None
        for col in columns:
            if isinstance(col, str) and col.startswith(f"{key} ") and "Model" in col:
                model_col = col
                break

        if key in columns and model_col:
            datasets.append((key, model_col))
        elif key == 'K5' and 'K5' in columns:
             # Specific handle for K5 if regex logic failed but we suspect it exists
             # In the file scan we saw 'K5 (-5, ..., +5) - Model'
             for col in columns:
                 if 'K5' in str(col) and 'Model' in str(col):
                     datasets.append(('K5', col))
                     break

    datasets = sorted(list(set(datasets)), key=lambda x: x[0]) # Deduplicate and sort

    if not datasets:
        print("No valid data/model column pairs found.")
        return
```

```
    print(f"Found {len(datasets)} datasets to analyze: {', '.join([d[0] for d in datasets])}")
    print("-" * 100)
    print(f"{'Dataset':<10} {'N':<6} {'RMS':<10} {'Chi^2':<12} {'dof':<6} {'Red. Chi^2':<12} {'p-value':<10} {'R^2':<10} {'Quality'}")
    print("-" * 100)

    results_list = []

    # Load Period from metadata (Row 1, Column 0)
    try:
        # Read just the first few rows to get period
        meta_df = pd.read_excel(filename, header=None, nrows=5)
        # Look for "Period (px)" in the first few cells
        period_val = 58.6 # Default fallback
        for r in range(3):
            for c in range(3):
                val = str(meta_df.iat[r,c])
                if "Period" in val and "(px)" in val:
                    # Value is likely in the cell below
                    period_val = float(meta_df.iat[r+1, c])
                    break
    except Exception as e:
        print(f"Warning: Could not read period from file ({e}). Using default 58.6 px.")
        period_val = 58.6

    period_px = int(round(period_val))
    print(f"Using Period: {period_val:.2f} pixels (window size: {period_px})")

    # Load data with header at row 2 (0-indexed)
    try:
        df = pd.read_excel(filename, header=2)
    except Exception as e:
        print(f"Error loading data: {e}")
        return

    # Identify datasets
    datasets = []
    columns = df.columns
    potential_keys = ['A', 'B', 'C', 'D', 'E', 'F', 'H', 'I', 'J', 'K3', 'K5']

    for key in potential_keys:
        model_col = None
        for col in columns:
            if isinstance(col, str) and col.startswith(f"{key} ") and "Model" in col:
```

```
                model_col = col
                break

        if key in columns and model_col:
            datasets.append((key, model_col))
        elif key == 'K5' and 'K5' in columns:
             for col in columns:
                 if 'K5' in str(col) and 'Model' in str(col):
                     datasets.append(('K5', col))
                     break

    datasets = sorted(list(set(datasets)), key=lambda x: x[0])

    if not datasets:
        print("No valid data/model column pairs found.")
        return

    print(f"Found {len(datasets)} datasets. Analyzing BEST matching single␣
↪period for each...")
    print("-" * 110)
    print(f"{'Dataset':<10} {'StartPx':<8} {'RMS':<10} {'Chi^2':<12} {'dof':<6}␣
↪{'Red. Chi^2':<12} {'p-value':<10} {'Quality'}")
    print("-" * 110)

    for data_col, model_col in datasets:
        data_raw = df[data_col]
        model_raw = df[model_col]

        mask = ~(pd.isna(data_raw) | pd.isna(model_raw))
        data_full = data_raw[mask].astype(float).values
        model_full = model_raw[mask].astype(float).values

        n_full = len(data_full)
        if n_full < period_px:
            continue

        # Sliding window to find best period
        best_chi2 = np.inf
        best_idx = 0
        best_res = None

        # Optimization: Convolution or just simple loop (loop is fine for␣
↪N=1400)
        # We want to minimize Sum of Squared Residuals for a window of size␣
↪period_px

        residuals_full = data_full - model_full
```

```
        sq_res_full = residuals_full**2

        # Calculate moving sum of squared residuals
        # Convolve is efficient: window sum
        window = np.ones(period_px)
        # mode='valid' returns only where window fully overlaps
        moving_ssr = np.convolve(sq_res_full, window, mode='valid')

        # Find index of minimum SSR
        best_idx = np.argmin(moving_ssr)

        # Extract best window
        data = data_full[best_idx : best_idx + period_px]
        model = model_full[best_idx : best_idx + period_px]

        # Recalculate stats for this specific window
        residuals = data - model
        rms = np.sqrt(np.mean(residuals**2))

        # Error Model / Uncertainty (sigma)
        # -------------------------------------------------------------
        # Camera: FLIR Blackfly S (Sony IMX273)
        # Bit Depth: 8-bit (0-255) typical for basic TIFFs
        # Sources of Noise:
        # 1. Shot Noise (Poisson): sigma_shot = sqrt(N_electrons) / Gain
        # 2. Read Noise: ~10 electrons rms (~0.5-1 DN depending on gain)
        # 3. Quantization Error: 1/sqrt(12) ~ 0.29 DN

        # We will approximate the total sigma in DN (Digital Numbers):
        # sigma_total = sqrt( Intensity + sigma_read^2 )
        # We enforce a generic minimum uncertainty of 1.0 - 2.0 DN to account␣
↪for
        # quantization and read noise floor, which avoids overweighting dark␣
↪pixels.

        sigma_read_noise = 2.0  # Conservative estimate (DN) for read +␣
↪quantization

        # If intensity < 0 (background subtraction artifacts), treat as 0 for␣
↪noise calc
        safe_intensity = np.maximum(data, 0)

        # Sigma^2 = Shot Noise variance + Read Noise variance
        # (Assuming gain roughly 1 e-/DN for simplicity, or just proportional␣
↪scaling)
        sigma_sq = safe_intensity + (sigma_read_noise ** 2)
```

```
        chi2 = np.sum((residuals**2) / sigma_sq)
        # --------------------------------------------------------------

        # Determine Parameters (p) based on dataset properties
        # Standard: 3 parameters (Translation aka Shift, Amplitude aka Scale,␣
↪Noise/Background aka Offset)
        p_params = 3

        # Exception 1: Constant Models (e.g., Mask B, 0-order only)
        # If the model is constant, Shift is irrelevant, and Scale/Offset are␣
↪degenerate (collapsed to 1 mean level)
        if np.std(model) < 1e-9 * (np.mean(np.abs(model)) + 1e-9):
            p_params = 1
        # Exception 2: Dataset A was used for Period Calibration
        # The period (pixel scale) was derived from A, so A consumed 1 extra␣
↪degree of freedom.
        elif data_col == 'A':
            p_params = 4

        dof = period_px - p_params
        if dof <= 0: dof = 1

        red_chi2 = chi2 / dof
        p_value = 1 - stats.chi2.cdf(chi2, dof)

        # Quality
        if red_chi2 < 0.5: quality = "Overfit"
        elif red_chi2 <= 2.0: quality = "Good"
        elif red_chi2 <= 5.0: quality = "Fair"
        else: quality = "Poor"

        print(f"{data_col:<10} {best_idx:<8} {rms:<10.2f} {chi2:<12.1f} {dof:
↪<6} {red_chi2:<12.3f} {p_value:<10.2e} {quality}")

    print("-" * 110)
    print("Analysis performed on the SINGLE period (approx 59 px) with the␣
↪lowest sum of squared residuals.")
    print("This method isolates the best-fitting region, ignoring global drift␣
↪or edge artifacts.")

    # Save to Excel
    try:
        results_df = pd.DataFrame(results_list)
        # Reorder columns for clarity
        cols = ['Dataset', 'StartPx', 'Chi2', 'DoF', 'Reduced_Chi2', 'Prob',␣
↪'RMS', 'Quality']
```

```
        # Filter for cols that exist
        cols = [c for c in cols if c in results_df.columns]
        results_df = results_df[cols]

        output_name = 'Chi_Squared_Results.xlsx'
        results_df.to_excel(output_name, index=False)
        print(f"\nResults saved to {output_name}")
    except Exception as e:
        print(f"\nError saving Excel file: {e}")

if __name__ == "__main__":
    main()
```

```
Loading data from /home/jupyter/course-material-phys233.git/SF - B2 Data
(3).xlsx…
Found 11 datasets to analyze: A, B, C, D, E, F, H, I, J, K3, K5
--------------------------------------------------------------------------------
--------------------
Dataset    N      RMS        Chi^2        dof     Red. Chi^2   p-value    R^2
Quality
--------------------------------------------------------------------------------
--------------------
Using Period: 58.61 pixels (window size: 59)
Found 11 datasets. Analyzing BEST matching single period for each…
--------------------------------------------------------------------------------
------------------------------
Dataset    StartPx  RMS        Chi^2        dof     Red. Chi^2   p-value
Quality
--------------------------------------------------------------------------------
------------------------------
A          852      14.61      85.8         55      1.559        4.98e-03   Good
B          643      1.21       1.4          58      0.025        1.00e+00
Overfit
C          821      0.99       1.6          58      0.027        1.00e+00
Overfit
D          671      1.21       1.0          56      0.019        1.00e+00
Overfit
E          676      2.75       7.8          56      0.138        1.00e+00
Overfit
F          844      7.70       76.1         56      1.359        3.83e-02   Good
H          749      27.15      1026.0       56      18.322       0.00e+00   Poor
I          732      33.25      518.6        56      9.260        0.00e+00   Poor
J          778      13.55      184.0        56      3.286        1.44e-15   Fair
K3         635      6.27       25.7         56      0.459        1.00e+00
Overfit
K5         703      8.06       37.8         56      0.674        9.71e-01   Good
--------------------------------------------------------------------------------
------------------------------
```

```
Analysis performed on the SINGLE period (approx 59 px) with the lowest sum of
squared residuals.
This method isolates the best-fitting region, ignoring global drift or edge
artifacts.

Results saved to Chi_Squared_Results.xlsx
```

[ ]:

```

```

[ ]:

```

```