Michelson Interferometer & Precision Wavelength Measurement --- Week 2

1 Where We Are in the Sequence

Week 2 of 2: Precision Wavelength Measurement by Fringe Counting

Last week you built and aligned a Michelson interferometer, derived a theoretical model for its behavior, and explored how non-ideal effects change the interference signal. This week you'll add a motorized translation stage and continuous data acquisition to measure the HeNe laser wavelength by counting interference fringes. You'll perform measurements at multiple distances to investigate how precision depends on scan distance, and build a quantitative error budget.

In Week 1 you estimated how small a displacement your interferometer could detect. This week you use the same physics in reverse: instead of detecting unknown displacements, you'll count fringes over a *known* displacement to measure wavelength.

Last week: Built interferometer, derived $P(d_2)$, explored misalignment and polarization effects, estimated position resolution

This week: Add motorized stage → Count fringes during mirror motion → Measure λ_{HeNe} → Test whether observed precision follows the predicted $1/N$ counting floor

2 Learning Goals

After completing this week's lab, you will be able to:

1. Derive $\lambda = 2\Delta L/N$ from the interferometer model and explain the factor of 2.
2. Use and adapt provided Python code to acquire photodetector data during a motor move and count fringes.
3. Examine time-domain signals to distinguish real fringes from spurious counts.
4. Design and execute a multi-distance experiment, compare observed scatter to the predicted $1/N$ counting floor, and investigate departures from the prediction.
5. Identify and diagnose a systematic error through the measure-compare-investigate cycle.
6. Construct your own error budget based on errors you discover.
7. Perform statistical analysis of repeated measurements and compare observed scatter to predicted uncertainty.

3 Overview of Your Work

This week culminates in a precision measurement of the HeNe laser wavelength. Your work proceeds in four phases:

Phase 1 — Theory (Prelab, ~30-45 min): Derive the fringe counting formula from your Week 1 model, calculate expected fringe counts and uncertainties at several distances, and verify with a synthetic simulation.

Phase 2 — Hardware setup (~30 min): Replace the manual micrometer with a motorized actuator, connect the DAQ, and verify that the system produces clean fringes during motorized motion.

Phase 3 — Measurement and investigation (~90 min): Measure the HeNe wavelength, discover and diagnose a systematic bias, and develop a fix. Build a dataset that shows how the bias behaves.

Phase 4 — Error budget (~30 min): Construct your own error budget based on errors you discovered, and report your best wavelength measurement.

See the detailed deliverables checklist at the end of this guide.

4 Python Preparation

You already have experience with the DAQ and motor from the Gaussian Beams lab: single voltage reads with `nidaqmx` (GB Week 2), motor connection, homing, and `MoveTo()` (GB Week 3), and combined motor + DAQ automation (GB Week 3-4). This week builds directly on those skills, but the *mode of operation* is fundamentally different.

Key difference from Gaussian Beams: In the beam profiling lab, you moved the motor to a position, stopped, took a measurement, then moved again. For fringe counting, the motor moves *continuously* at constant velocity while the DAQ records *streaming* data. The motor and DAQ must operate simultaneously.

What you already know	What's new this week	Resource
Single DAQ reads (`task.read()`)	Streaming acquisition (`cfg_samp_clk_timing` + `AcquisitionType.FINITE`)	NI-DAQmx
Move-to-position (`MoveTo()`)	Constant-velocity motion (`GetVelocityParams` / `SetVelocityParams`)	Thorlabs Motors
Sequential step → stop → read	DAQ and motor running simultaneously (threading)	See reference scripts below

Reference scripts: Working Python code for this lab is provided below. Each script builds on the previous one — download them to your lab computer and use them as starting points.

- `01_fringe_simulation.py` — Simulate fringe counting at multiple distances (no hardware needed). Use this to check your prelab work.
- `02_daq_streaming.py` — Streaming DAQ acquisition. Use this to verify your DAQ works in streaming mode before adding the motor.
- `03_motor_velocity.py` — Constant-velocity motor control. Use this to verify the motor works in velocity mode before combining with the DAQ.
- `04_fringe_counting.py` — Complete fringe counting measurement coordinating motor and DAQ. This is the reference implementation for the full measurement, similar to `04_beam_profiler.py` in Gaussian Beams Week 4.

You do not need to write all of this code from scratch. Use these scripts as starting points and adapt them for your measurements. However, you are expected to *understand* the code you run. Before running each script, read through it and identify: (a) what physical parameters are set and where, (b) where fringes are counted and how the wavelength is calculated, and (c) what assumptions the code makes about your experimental setup.

Before lab, complete the prelab (below) and ensure you can:

- Run `01_fringe_simulation.py` and verify the wavelength calculation matches your hand calculations

- Review the NI-DAQmx resource page section on timed acquisition (`cfg_samp_clk_timing`)
- Review the Thorlabs Motors resource page section on setting velocity parameters

5 Prelab

Complete this prelab before your lab session (~30-45 min). All exercises use Python only—no hardware is required. Before starting, read through the Deliverables and Assessment section at the end of this guide so you know what is expected for the week.

Before you begin — connect to Week 1. In Week 1 you estimated that your interferometer could resolve displacements of roughly a few nanometers (based on $d(d_2)/dP$ and your noise floor $\delta P_{\min}$). This week you will move the mirror by 1–10 mm — roughly 10^3 to 10^4 times farther. In your notebook, briefly answer: why is the Week 2 measurement strategy fundamentally different from the Week 1 position resolution estimate, even though both use the same interferometer model and the same equation $P(d_2)$? What changed about how you use that model? Keep your answer in mind as you work through the derivation below.

5.1 Part 1: Derive the fringe counting formula

From your Week 1 analysis, you know the power on the photodetector depends on mirror position:

$$P(d_2) = P_0\left[1 + \cos\left(\tfrac{4\pi d_2}{\lambda}\right)\right] \tag{1}$$

When the mirror moves a distance ΔL, the argument of the cosine changes by $4\pi\Delta L/\lambda$. Each complete fringe corresponds to a 2π change, so the number of fringes is:

$$N = \tfrac{2\Delta L}{\lambda} \tag{2}$$

Solving for wavelength gives the measurement equation:

$$\lambda = \tfrac{2\Delta L}{N} \tag{3}$$

a. **Derive** Equation 2 from Equation 1. Explain in your own words why the factor of 2 appears (hint: it comes from the round-trip path difference).

b. **Calculate** the expected number of fringes for a HeNe laser ($\lambda = 632.8$ nm) at the following travel distances:

Travel distance ΔL	Expected fringes N	Uncertainty $\delta\lambda = \lambda/N$
1 mm		
2 mm		
5 mm		
10 mm		

c. **Explain** why the uncertainty $\delta\lambda = \lambda/N$ makes physical sense. What does it mean to have an uncertainty of ± 1 fringe? Why does precision improve with longer scans?

5.2 Part 2: Fringe counting simulation

The reference script `01_fringe_simulation.py` simulates fringe counting at multiple distances. Before running it, work through the following predict-observe-explore cycle.

5.2.a Predict

a. Open `01_fringe_simulation.py` and read the `generate_fringes()` function. Find the line that computes `fringe_freq`. Why is the factor of 2 there? (Connect this to the round-trip path difference you derived in Part 1.)

b. Read the `count_zero_crossings()` function. What assumption does it make about the signal mean? Why does dividing zero-crossings by 2 give the fringe count?

c. **Before running the script**, predict the fringe count and calculated wavelength for a 5 mm scan. Does your prediction match your Part 1 table?

5.2.b Observe

d. Run `01_fringe_simulation.py`. Compare the Part 1 output table to your hand calculations. Do the fringe counts and wavelengths agree? If there are small discrepancies, where do they come from? (Hint: the signal has a finite number of samples — what happens at the edges?)

5.2.c Explore: noise sensitivity

e. In `generate_fringes()`, the `noise_amplitude` parameter adds Gaussian noise. The script's Part 2 tests several noise levels, but only at 5 mm. **Modify the script** to explore how noise sensitivity depends on distance:

- Run the noise sweep at 1 mm instead of 5 mm. At what noise amplitude does fringe counting break down? Is this threshold higher or lower than at 5 mm?
- Try changing the sample rate (e.g., 5000 Hz vs. 20000 Hz). How does this affect noise tolerance? Why?

Record the noise level where the wavelength error exceeds 1 nm for each condition you test.

5.3 Part 2b: What happens with buffer noise?

In practice, the DAQ starts recording *before* the motor begins moving and continues *after* it stops — typically ~0.5–1 s of "buffer" on each end. During these quiet periods the motor is stationary, but environmental vibrations (building vibration, air currents, table bumps) cause the interferometer mirrors to drift slightly, producing a small fluctuating signal.

5.3.a Predict before running

a. The script's Part 3 simulates this by prepending and appending buffer periods with low-frequency vibration noise to the clean fringe signal. **Before running Part 3**, answer:

- Will the extra zero-crossings from buffer noise make the fringe count too high or too low?
- Will the calculated wavelength therefore be biased high or low?
- If the buffer noise always produces roughly the same number of extra fringes regardless of motor travel distance, how does the resulting wavelength *error* scale with distance? Fill in a prediction table:

Distance	True N	Extra fringes	Biased N	λ_{biased}	Error
1 mm					
5 mm					
10 mm					

5.3.b Check and explore

b. Run the script and compare Part 3 output to your predictions. How many extra fringes come from the buffer periods? Does the distance dependence match what you predicted?

c. **Modify the buffer parameters** to explore how this effect changes:
   - What happens if you increase the vibration frequency from 15 Hz to 50 Hz? To 120 Hz (fluorescent light flicker)?
   - What happens if you increase the buffer duration from 0.5 s to 1.0 s?
   - What happens if you increase the vibration amplitude from 0.3 to 0.8?

   For each change, note whether the number of extra fringes increases or decreases, and whether this matches your intuition.

> Keep these results in mind
>
> As you work through the hardware measurements, revisit the simulation results you recorded here. The patterns you observe in the lab may connect to what you found in this simulation.

5.4 Part 3: Error budget preview

Before taking data, think about what could limit your measurement accuracy.

a. **Brainstorm error sources.** List at least three potential sources of error in the fringe counting measurement. For each one:
   - Estimate its magnitude (in nm of wavelength error, or as a fraction of λ)
   - Classify it as **systematic** (always biases the result in one direction) or **random** (causes scatter between runs)
   - Predict whether it depends on travel distance, and if so, how

   Hint: Think about what goes into the formula $\lambda = 2\Delta L/N$. What could make ΔL wrong? What could make N wrong?

b. **Pick your most important source.** Which error source do you predict will dominate at short distances (1 mm)? At long distances (10 mm)?

> Your error budget will evolve
>
> This is a preliminary list based on your understanding *before* taking data. During the lab, you will discover which sources actually matter — and you may find sources you didn't predict. The final error budget should reflect what you *learned*, not just what you guessed.

6 Converting the Interferometer to a Fringe Counter

6.1 Hardware setup

a. Replace the micrometer dial on the translation stage of the interferometer with the ZST225B electronic actuator. Connect the actuator to the KST101 controller and connect the power supply. You set up this same motor in Gaussian Beams Week 3 — refer to the Thorlabs Motors resource page if you need a refresher on the connection and homing procedure.

b. Connect the photodetector to the DAQ using **differential mode** (signal to AI0+ on pin 2, ground to AI0- on pin 3). Differential mode reduces ground loop noise, which matters because the fringe signal can be small. (Recall from the NI-DAQmx guide that differential mode measures the voltage *between* two pins rather than relative to ground.)
Before adding the motor, verify that the DAQ can record streaming data from the photodetector. Run `02_daq_streaming.py` to acquire a few seconds of data and check that you see a signal. If you gently tap the table, you should see fringes appear in the time-domain trace.
The key new pattern compared to Gaussian Beams is **timed acquisition** — instead of reading one sample at a time, you configure the DAQ to collect a fixed number of samples at a precise rate:

```
import nidaqmx
from nidaqmx.constants import AcquisitionType, TerminalConfiguration

sample_rate = 10000   # Hz
duration = 10.0       # seconds
num_samples = int(sample_rate * duration)

with nidaqmx.Task() as task:
    task.ai_channels.add_ai_voltage_chan(
        "Dev1/ai0",
        terminal_config=TerminalConfiguration.DIFF,
        min_val=-10.0, max_val=10.0
    )
    task.timing.cfg_samp_clk_timing(
        rate=sample_rate,
        sample_mode=AcquisitionType.FINITE,
        samps_per_chan=num_samples
    )
    # This blocks until all samples are collected
    data = task.read(
        number_of_samples_per_channel=num_samples,
        timeout=duration + 5.0
    )
```

Compare this to the single-read pattern from Gaussian Beams (`task.read()` with no timing configuration). Here, `cfg_samp_clk_timing` tells the DAQ hardware to sample at a precise rate using its internal clock, and `task.read()` blocks until all samples are collected. See the NI-DAQmx guide for more details.

c. Do a quick check to make sure the interferometer is well aligned. Use the Week 1 alignment techniques to realign if necessary. Does moving the stage throughout its full range of motion change the alignment? Why might this happen?

Practical tip: Turn off the overhead fluorescent lights during data collection. Fluorescent lights flicker at 100-120 Hz, which couples optically into the photodetector and can obscure your fringe signal.

6.2 Motor velocity control

In Gaussian Beams you used `MoveTo()` to position the motor and wait for it to arrive. For fringe counting, you instead set a *velocity* and command a long move while recording data simultaneously. The velocity control pattern uses `GetVelocityParams` / `SetVelocityParams` (documented on the Thorlabs Motors resource page under “Setting Velocity”):

```
from System import Decimal

# Set velocity parameters
vel_params = device.GetVelocityParams()
vel_params.MaxVelocity = Decimal(0.5)       # 0.5 mm/s
vel_params.Acceleration = Decimal(2.0)       # 2.0 mm/s^2
device.SetVelocityParams(vel_params)

# Start motor (this returns before motion completes)
end_position = start_position + travel_mm
device.MoveTo(Decimal(end_position), 0)  # timeout=0 for non-blocking
```

Run `03_motor_velocity.py` to verify the motor works in velocity mode. Check that the reported travel distance matches your commanded distance.

Important: For fringe counting, the motor's *endpoint accuracy* matters more than its velocity stability. Unlike FT spectroscopy (which requires precisely constant velocity), fringe counting only needs the motor to start and stop at known positions. Velocity fluctuations during the scan don't affect the measurement as long as you count all the fringes.

6.3 Hardware readiness checkpoint

Before proceeding to measurements, verify all three components work:

a. **DAQ streaming:** `02_daq_streaming.py` produces a clean time-domain trace; tapping the table produces visible fringes.
b. **Motor control:** `03_motor_velocity.py` completes a 5 mm move with correct encoder readout.
c. **Combined scan:** `04_fringe_counting.py` produces a fringe signal for a short (1 mm) test scan.

If any of these fail, debug the hardware before starting the measurement investigation — troubleshooting during the investigation will eat into your analysis time.

7 Measuring the HeNe Wavelength

This section is structured as a guided investigation. You will start with a simple measurement, notice something unexpected, and then systematically diagnose it. **Read each step and record your observations before moving on.**

Synchronization note: The DAQ must start sampling *before* the motor starts moving. The reference script `04_fringe_counting.py` demonstrates this threading pattern in the `run_scan()` function. By default, it uses simple zero-crossing counting.

7.1 Step 1: First measurement

a. **Run a single scan** at 5 mm travel distance. The script records the motor's start and end positions from the encoder, counts zero-crossings in the recorded signal, and computes:
$\lambda = \frac{2\Delta L}{N} = \frac{2\times(\text{end pos} - \text{start pos})}{N_{\text{crossings}}/2}$
b. **Write down your result** before moving on. How does it compare to the known value of 632.8 nm? Is the error larger than you expected from $\delta\lambda = \lambda/N$?

7.2 Step 2: Is the error systematic or random?

c. **Repeat** the measurement 3–5 times at 5 mm. Record all values. Compute the mean and standard deviation. Is the mean consistently too high or too low? If the mean is biased in one direction, that points to a *systematic* error rather than random noise.

7.3 Step 3: How does the error depend on distance?

d. **Predict before measuring:** If the fringe count has a constant additive bias (some extra fringes that don't depend on distance), how would the wavelength error scale with distance? Would the bias be larger at short or long distances? Write down your prediction.
e. **Measure** at 1, 2, 5, and 10 mm, with 3 repeats at each distance. You can use the measurement series mode in `04_fringe_counting.py` to automate this.
f. **Compare** to your prediction. Does the pattern match? Is the error largest at short distances?

Analyzing distance dependence

Think carefully about how the error *scales* with distance. Does it decrease proportionally to $1/\text{ distance}$, or does it have a different dependence? The shape of the scaling is a clue about the *type* of error source at play.

7.4 Step 4: Diagnose the source

g. **Plot the raw signal** from one of your scans (the full time-domain trace). Look carefully at the beginning and end of the recording. What happens *before the motor starts* and *after it stops*? Do you see any fringes in those regions?
h. **Connect to your prelab.** In Part 2b, you simulated buffer noise and predicted extra fringes. Does the pattern you see in the real data match your simulation? How many extra crossings come from the buffer periods?

7.5 Step 5: Fix it

i. **Choose a fix.** You have (at least) two options:
   - **Analytical correction:** Estimate the number of extra fringes from the buffer periods and subtract them from your count. How accurately can you determine this correction?
   - **Algorithmic improvement:** The reference script provides `count_fringes_gated()` which uses the signal envelope to exclude buffer periods automatically. Run your measurement series again with the gated method (select option 2 at the counting method prompt in `04_fringe_counting.py`).

j. **Compare** corrected vs. uncorrected results at each distance. At which distances does the fix work well? At which distances does residual bias remain? What might explain why the fix is less effective at some distances than others?
k. **Reflect:** Could you redesign the *acquisition* (not just the analysis) to avoid this problem entirely? What would you change?

7.6 Step 6: Iterate

l. After applying your fix, do results at 1 mm now agree with 10 mm within uncertainty? If not, what *other* error source might you investigate next? Run additional measurements if time permits.

8 Analysis and Error Budget

8.1 Bias characterization

a. **Plot** your uncorrected wavelength vs. distance. Overlay the known value (632.8 nm). Does the bias pattern match what you predicted — largest at short distances, shrinking with distance?
b. **Plot** corrected (gated or analytically corrected) wavelength vs. distance on the same graph. Does the correction remove the systematic bias?

8.2 Precision check

c. **Precision scaling test.** At each distance, compute both the predicted counting floor ($\delta\lambda = \lambda/N$) and the observed standard deviation from your repeated measurements. Present both in a single table or plot.
Does the observed scatter agree with the counting floor? If the scatter exceeds the prediction, does the *excess* decrease with the same $1/N$ scaling as the counting floor, or does it have a weaker distance dependence? What does this tell you about the dominant source of run-to-run scatter — is it a counting effect, or something else?

When your data disagrees with a model

If your observed scatter exceeds the counting floor at all distances, the $1/N$ model is not wrong — it is *incomplete*. It correctly predicts the minimum possible scatter from counting alone. The excess scatter tells you that additional noise sources contribute to the measurement uncertainty. Identifying those sources is part of building a real error budget.

8.3 Error budget

d. **Construct your own error budget.** List ALL error sources you identified during the lab — not just the ones from the prelab. For each source, fill in a row:

Error source	$\delta\lambda$ at best distance	Systematic or random?	How you determined it
(your sources)			
Total			

Note that some sources may have both systematic and random character (e.g., buffer noise biases the result LOW but varies somewhat in magnitude between runs). Classify by the dominant character.

For your total, add systematic and random contributions separately. Report your final result with both:

$$\lambda_{\text{HeNe}} = (\text{value} \pm \text{random} \pm \text{systematic})\ \text{nm}$$

e. Which error source dominates? Is it one you predicted in the prelab, or one you discovered during the lab?

8.4 Reflection

f. Compare this measurement technique to other ways of measuring wavelength (e.g., diffraction grating, prism spectrometer). What are the advantages of interferometric fringe counting? What are its limitations?
g. What was the most important error source you identified during this lab, and how did you discover it? How did your understanding of the error evolve from your prelab predictions to your final error budget — did you find what you expected, or were you surprised?

9 Deliverables and Assessment

Your lab notebook should include the following for this week. This checklist represents minimum expectations — your notebook should also include any unexpected observations, failed approaches that informed your understanding, and questions that arose during the lab.

9.1 Prelab (completed before lab, ~30-45 min)

1. **Week 1 bridge**: Brief explanation of how Week 2 measurement strategy differs from Week 1 position resolution, even though both use $P(d_2)$
2. **Derivation**: $\lambda = 2\Delta L/N$ derived from $P(d_2)$, with explanation of the factor of 2
3. **Fringe count table**: Expected fringes and $\delta\lambda$ at 1, 2, 5, 10 mm
4. **Simulation predict-observe**: Predictions for fringe count at 5 mm, comparison to script output, explanation of any discrepancies
5. **Noise exploration**: How noise tolerance depends on distance and sample rate, with threshold values recorded
6. **Buffer noise predictions**: Predicted bias direction and distance dependence table, compared to script output
7. **Buffer parameter exploration**: Observations from modifying vibration frequency, buffer duration, and amplitude
8. **Error budget preview**: At least three potential error sources with magnitude estimates and classifications

9.2 In-Lab Documentation (recorded during lab)

1. **Hardware setup notes**: Any alignment changes needed, DAQ and motor configuration
2. **Signal check**: Time-domain trace from a test scan showing clean fringes during motor motion
3. **First measurement**: Wavelength result at 5 mm with comparison to 632.8 nm
4. **Systematic check**: Repeated measurements at 5 mm showing consistent bias direction
5. **Multi-distance investigation**: Measurements at 1, 2, 5, 10 mm (3× each), showing bias vs. distance pattern
6. **Diagnosis**: Time-domain plot of full signal showing buffer noise before/after motor motion
7. **Fix applied**: Comparison of uncorrected vs. corrected (gated or analytical) results

9.3 Analysis (can be completed after lab)

1. **Bias characterization plot**: Uncorrected and corrected wavelength vs. distance, overlaid with known value
2. **Precision comparison**: Observed standard deviation vs. predicted $\delta\lambda = \lambda/N$ at each distance, with analysis of whether the scaling matches the $1/N$ prediction
3. **Investigation narrative**: What you measured, what was wrong, how you diagnosed it, what you did about it
4. **Final result**: $\lambda_{\text{HeNe}} = (\text{value} \pm \text{random} \pm \text{systematic})$ nm
5. **Error budget table**: All sources *you identified*, with magnitudes, classifications, and how you determined each
6. **Dominant error identification**: Which source limits your measurement, and was it one you predicted?
7. **Reflection**: Comparison to other wavelength measurement techniques
8. **Discovery narrative**: What was the most important error source you identified, how did you find it, and how did your understanding evolve from prelab predictions to final error budget?