Adaptive interference inference

View On GitHub

Download PDF

This project is maintained by AndrewGYork in the York lab, and was funded by Calico Life Sciences LLC

Micro-publication

What is the signal-to-noise ratio of phase-contrast imaging? Can it be infinite?

Andrew G. York[1*]

[1]*Calico Life Sciences LLC, South San Francisco, CA 94080, USA*

[*]*Permanent email: andrew.g.york+inference@gmail.com*

[*]*Institutional email: agy+inference@calicolabs.com*

Abstract

A mathematical optimization problem, inspired by the physics of interferometry. How much can tuning the intensity and phase of an interfering reference beam improve our ability to infer the absorption and phase shift due to an unknown object?

Intended audience

Anyone who likes math, coding, or physics. Click the appropriate link below to jump to the relevant section:

Click here if you like math	Click here if you like coding	Click here if you like physics

Peer review status

Pre-print published XXX XX, 20XX (This article is not yet peer-reviewed)

Cite as: doi:10.5281/zenodo.XXXX

Note that this is a limited PDF or print version; animated and interactive figures are disabled. For the full version of this article, please visit https://andrewgyork.github.io/adaptive_interference_inference

A math puzzle

Let's play a game. I know a complex number x. Your job is to infer x. You know that $|x| \leq 1$. *

You may choose any two complex numbers, a and b. I will calculate $y = |ax + b|^2$. Then I'll draw randomly from a Poisson distribution with expectation value y, and tell you the (stochastic) result r.

We repeat this process as many times as you'd like: on round i, you choose a_i and b_i, I'll calculate y_i and I'll tell you r_i. Your "cost" for round i is the quantity $|a_i|^2$, and you have a "budget" of A. If you go over budget (meaning, if you choose a value a_N such that $A < \sum_i^N |a_i|^2$), I won't answer, and the game is over.

What algorithm should you follow to infer x as accurately** as possible, for a given value of A? Ideally, express this algorithm as a short Python/Numpy script, ideally in a Github repo that I can link to here. Feel free to invite anyone who's interested to play.

Let the best algorithm win!

- Can you do any better than the algorithm $a_0 = A$, $b_0 = 0$?
- Can you do any better than an algorithm where a and b are chosen in advance, independently of r_k ? For example: $a_k = \sqrt{A/N}$ and $b_k = \sqrt{A/N}e^{2\pi ik/N}$

Notes:

* For Bayesians: I'll choose the magnitude $|x|$ from a uniform distribution between 0 and 1, and I'll choose the phase $\angle x$ from a uniform distribution between 0 and 2π. Bonus points if your algorithm works robustly (or can be easily adapted) if x is drawn from a different distribution.

** Note I didn't define "accuracy"! I'll leave this intentionally flexible. I suspect the "ideal" algorithm will depend on the choice of definition, but I'm indifferent as long as your definition of accuracy seems sane to me.

A coding challenge

I've implemented an 'oracle' that will play a game with you. You can find a copy of the Python 3 code in our `adaptive_interference_inference` GitHub repository: ./code/game.py

```
import numpy as np

class Oracle:
    def __init__(self, budget):
        assert budget > 0
        self.budget = budget
        self.total_cost = 0
        self.round = 0
        self.a_history = []
        self.b_history = []
        self.r_history = []
        self.game_over = False

        # Generate some secret numbers.
        # Don't peek!
        self._magnitude = np.random.random(1)
        self._phase = 2*np.pi*np.random.random(1)
        self._x = self._magnitude * np.exp(1j*self._phase)
        return None

    def ask(self, a, b):
        if self.game_over:
            return None
        self.total_cost += np.abs(a)**2
        if self.total_cost > self.budget:
            self.game_over = True
            return None
        y = np.abs(a*self._x + b)**2
        r = np.random.poisson(y)
        self.round += 1
        self.a_history.append(a)
        self.b_history.append(b)
        self.r_history.append(r)
        return r
```

To play this game, import `game.py`, choose a value for `budget`, and create an instance of the `Oracle` object. Try to infer the value of `Oracle._magnitude` and/or `Oracle._phase` as accurately as you can, using only the return values of calls to the `Oracle.ask(a, b)` method. You are allowed to inspect the code of `game.py`, but after you instantiate an `Oracle` object, you are *not* allowed to inspect its `Oracle._x`, `Oracle._magnitude`, or `Oracle._phase` attributes; that's cheating!***

Here's an example of how you might use this oracle to play the game:

```
import numpy as np
import game
```

```
print("An example game")
oracle = game.Oracle(budget=1000)
print("Budget:", oracle.budget, '\n')
num_phases = 5
for phase_angle in np.arange(0, 2*np.pi, 2*np.pi/num_phases):
    print('Round', oracle.round)
    a = np.sqrt(oracle.budget / num_phases) * (1 - 1e-12)
    b = a * np.exp(1j*phase_angle)
    print(' a: %07s    (intensity)\n'%('%0.2f'%(np.abs(a)**2)),
          '   %07s*pi (phase)'%('%0.2f'%(np.angle(a)/np.pi)))
    print(' b: %07s    (intensity)\n'%('%0.2f'%(np.abs(b)**2)),
          '   %07s*pi (phase)'%('%0.2f'%(np.angle(b)/np.pi)))
    response = oracle.ask(a, b)
    print('oracle.ask(a, b):', response)
    print('Total cost: %0.2f / %0.2f\n'%(oracle.total_cost,
                                         oracle.budget))
print("Game over.")
print("From these responses, how well can you infer 'x'?")
print("\nIf you'd made smarter choices of 'a' and 'b' on each round,\n",
      "could you do a better job inferring 'x'?", sep='')
```

Note that this is just an example of how to ask the oracle questions. I didn't specify a method to infer `Oracle._x` based on the oracle's responses to `Oracle.ask(a, b)`. That's the game!

What's the best algorithm you can implement to infer the value of `Oracle._x` as accurately** as possible, for a given value of `Oracle.budget`? Ideally, express this algorithm as a short Python/Numpy script, ideally in a Github repo that I can link to here. Feel free to invite anyone who's interested to play.

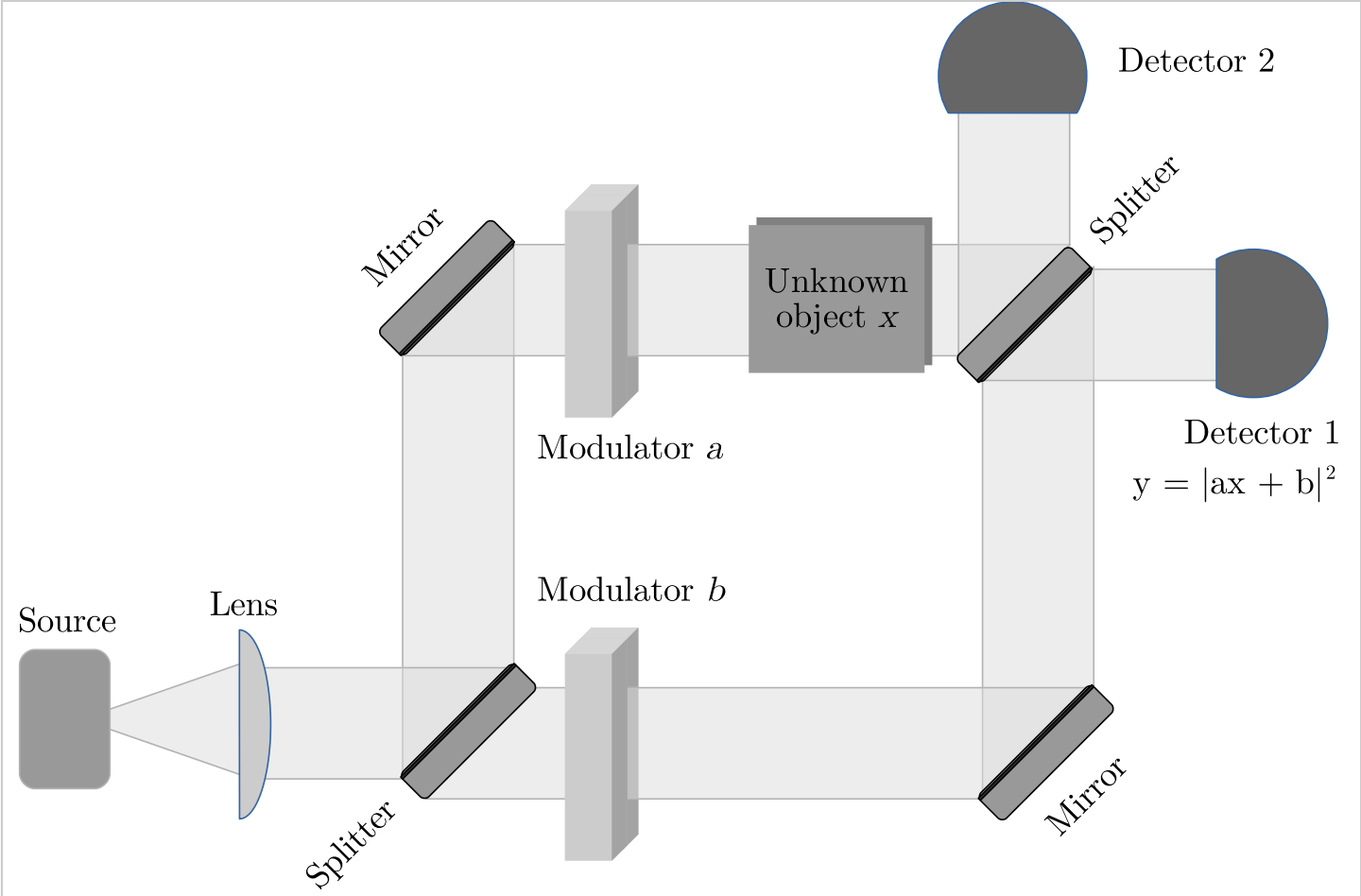
Notes:

*** For machine-learners: feel free to inspect `Oracle._x`, `Oracle._magnitude`, and/or `Oracle._phase` while you're training and validating a model, just be careful that your *model* doesn't consult these values.

A physics question

Suppose we want to measure the complex optical transmission coefficient x of a thin uniform flat-faced slab of partially transparent unknown material†. When we transmit a collimated beam of monochromatic light through the slab, the slab reduces the beam's amplitude by $|x|$, and shifts the beam's phase by $\angle x$. How accurately can we measure this change in intensity and phase?

Consider the ideal interferometric measurement shown in Figure 1 below:



Measurement type:	Interferometric measurement
(Click the radio buttons to change the figure)	Object-free interferometric measurement ($x = 1$)
	Background intensity measurement ($a = 0$)
	Transmitted intensity measurement ($b = 0$)

Figure 1: An unknown material sits in one arm of an ideal optical interferometer. The interfering beams are collimated, monochromatic, perfectly coherent, and overlap perfectly at the outputs. The average number of photoelectrons y measured per trial at Detector 1 is $|ax + b|^2$, where a, b, and x are complex numbers. Modulators a and b let us tune the amplitude and phase of a and b respectively. The absorption and phase shift due to the material determines the amplitude and phase of x.

We make a measurement by choosing the amplitude and phase for each modulator and counting photoelectron "clicks" at Detector 1 for some time interval. Suppose that our source and our interferometer are perfectly stable, and our detector never gives false detections[††]. Even in this ideal case, the number of "clicks" we count is inherently stochastic. The number of clicks per trial at Detector 1 is drawn from a Poisson distribution with expected value $y = |ax + b|^2$. How shall we infer x?

With the probe beam blocked (Fig. 1 with `Measurement type: Background intensity measurement`), we can calibrate the reference beam signal $|b|^2$ due to Modulator b. Since the mean y of a Poisson process grows faster than its standard deviation $\sqrt{y}$, we can calibrate the output amplitude of Modulator b with arbitrarily high signal-to-noise ratio $y/\sqrt{y}$, by chasing the limit $|b|^2 \to \infty$. Next, we can remove the object (Fig. 1 with `Measurement type: Object-free interferometric measurement`) and use a similar procedure to calibrate the output amplitude of Modulator a, and also to calibrate how the modulators tune the relative phase $\angle(a/b)$ between the probe and reference beams.

With perfect knowledge and control of a and b, we're ready to measure the complex transmission x of our object. Let's add one additional constraint: for this measurement, we can't use an infinite amount of light. Perhaps we're time-limited; our source has finite brightness, we have many objects to measure today, and we can't spend as much time on one measurement as we did on calibration. Perhaps we're dose-limited; too much light might melt or burn our unknown object. Either way, if the cumulative "light dose" $\sum_i^N |a_i|^2$ at trial N exceeds our "dose budget" A, we must stop measuring, and Poisson noise limits our signal-to-noise ratio. What is our signal, and what is our noise?

Let's start with a simple measurement of $|x|$ (how much the object reduces the probe beam's amplitude; Fig. 1 with `Measurement type: Transmitted intensity measurement`). Each trial, Detector 1 yields a number of "clicks" drawn stochastically from a Poisson distribution with expectation value $|ax|^2$. Our noise is the standard deviation of this distribution, $|ax|$. Given our perfect knowledge of a, our mean signal is the *change* in intensity $|a|^2 - |ax|^2$, yielding a signal-to-noise ratio (SNR) of $|\frac{a}{x}| - |ax|$. Note that in the $|x| \to 1$ limit of perfect transmission, the SNR tends to zero. Similarly, in the $x \to 0$ limit of perfect opacity, the SNR tends to infinity! This makes sense: an object with sufficiently high transmission becomes invisible to a transmission measurement, and a perfectly opaque object yields a detectable and fluctuation-free signal. Perhaps most interestingly, if you count even a single "click", you know $|x| \neq 0$ and the object was not perfectly opaque.

If we want to measure phase $\angle x$ in addition to amplitude $|x|$, we must use interference (Fig. 1 with `Measurement type: Interferometric measurement`). Our noise in this case is $|ax + b|$, and our signal is still the change in intensity due to the object, $|a + b|^2 - |ax + b|^2$, yielding an SNR of $\frac{|a+b|^2}{|ax+b|} - |ax + b|$. Like transmission measurements, our SNR can be infinite, but since we control a and b, we can achieve infinite SNR regardless of the value of x! All we have to do is choose $\frac{b}{a} = x$, and we're guaranteed to get a detectable, fluctuation-free signal with infinite SNR: zero "clicks". This is presumably insane, but it raises some fun questions.

In order to achieve "infinite SNR", we must have chosen $\frac{b}{a} = x$. But if we don't know x yet, how are we supposed to choose a and b? Also, observing zero clicks doesn't guarantee $x = \frac{b}{a}$; maybe we just didn't wait long enough. Infinite SNR is therefore quite different from infinite precision. However, if we observe even a single click on Detector 1, we know with certainty $x \neq \frac{b}{a}$. This suggests that if we view measurement as asking a question to the unknown object, the question "What are you?" may be very different from the question "Are you *this*?". There are only two possibly answers: "maybe" (zero clicks after using up your entire dose budget) and "no" (the first click, at which point you should probably halt the measurement and re-tune your modulators). Clearly there are important clues encoded in how quickly the "no" is delivered.

So, what algorithm should we use for tuning our modulators? My suspicion is the optimal measurement will use an extremely small fixed value for a (virtually guaranteeing zero or one "clicks" per trial), and choose a new $|b|$ and $\angle b$ after every time the detector clicks, hunting for the informative special case $b = ax$. I also suspect that the new choice of b should be informed by how long each previous measurement took to produce its first click.

But I'm not going to solve this myself; I'd rather play the game with you. What algorithm would you use? Ideally, express this algorithm as a short Python/Numpy script, ideally in a Github repo that I can link to here. Feel free to invite anyone who's interested to play.

What does this teach us about measurement?

- Is "infinite SNR" actually indicative of high precision?
- Do we actually benefit from "hunting" for perfect destructive interference at Detector 1, or would we be better off searching for a diverse set of a, b combinations?
- How much better could you do if you also had access to "clicks" from Detector 2?

Notes:

[†]This is an extremely common, extremely general problem. For example, this is the one-voxel limit of nearly the entire field of transmitted light microscopy.

[††] This is consistent with the laws of physics, but an awful lot of work to achieve in practice.

Hosted on
GitHub Pages