UF Herbert Wertheim College of Engineering
UNIVERSITY of FLORIDA

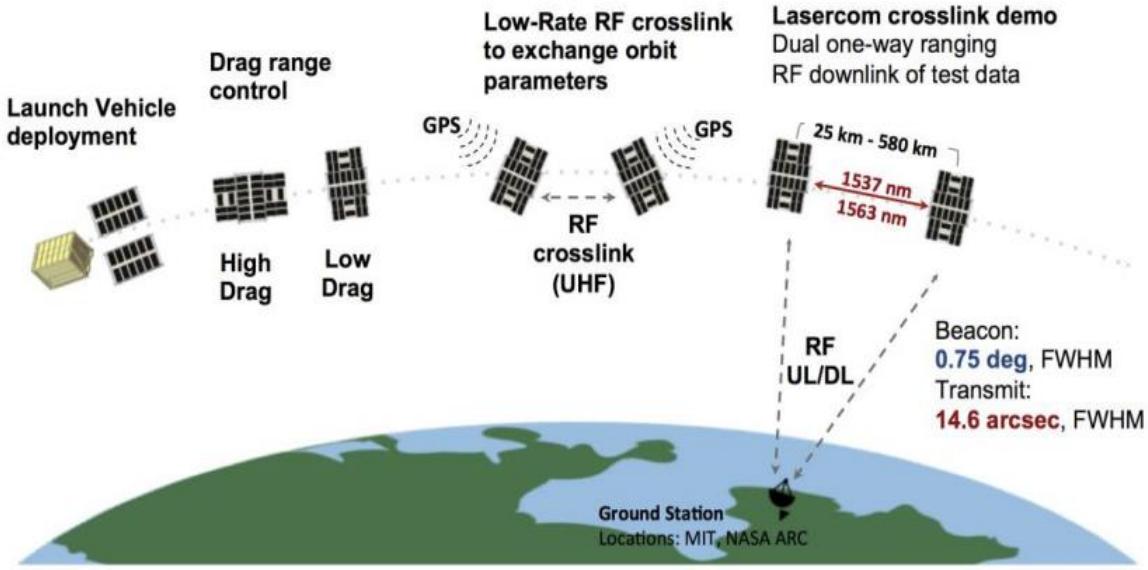
Comparison of evolutionary algorithms to derivative-based methods for optimizing laser-based ranging and communications

27 April 2023
California Polytechnic State University
CubeSat Developers' Workshop 2023

J. Arthur Conroy, Shreeyam Kacker, Myles Clark, Paul Serra, Kerri Cahoy, John Conklin

The Mission

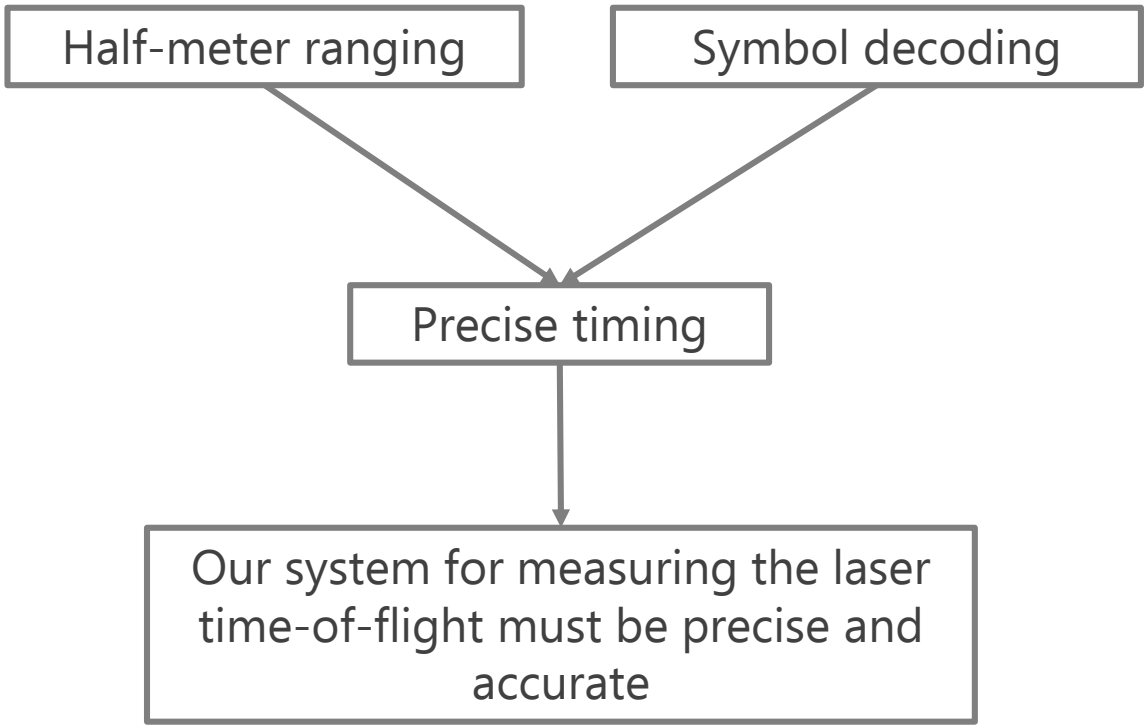
- What is the CLICK-BC mission?
  - Use CSACs to achieve precise laser ranging & communications
  - MIT STAR Labs, UF PSSL, & NASA Ames
- What are its goals?
  - Full duplex communications
  - Ranging down to 0.5 meters (1.6 nanosecond precision)
  - Pulse Position Modulation at 2 to 7 bits per symbol
- Why do this mission's goals matter?
  - Lower SWaP compared to radio
  - Enhanced security



[1] Kerri Cahoy et al (2018) *The CubeSat Laser Infrared CrosslinK Mission (CLICK)*

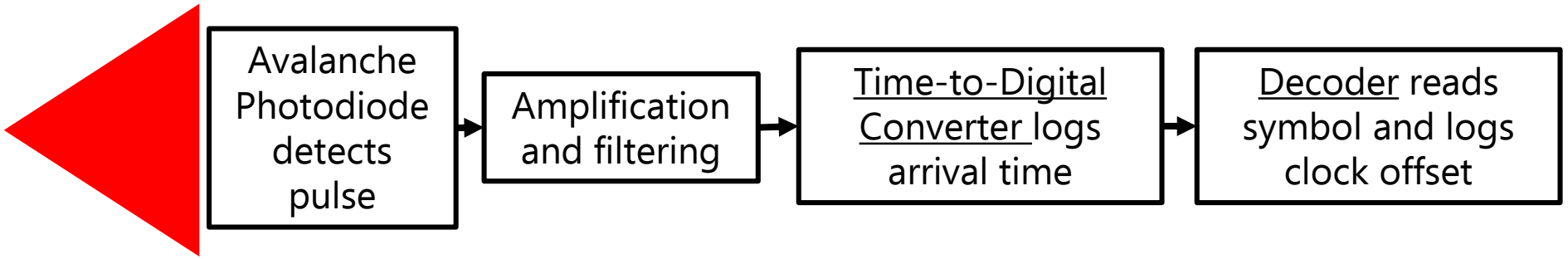
The Motivation

- How do the goals of the mission motivate my work?



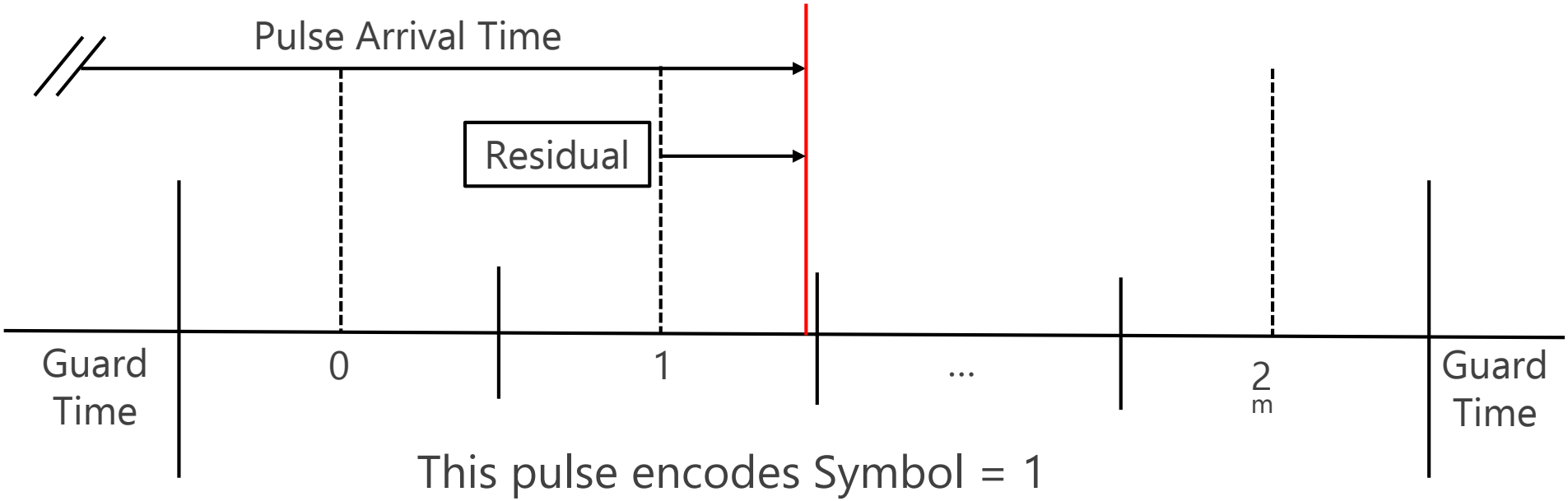
We want to optimize the way we measure time-of-flight

The Receive Channel



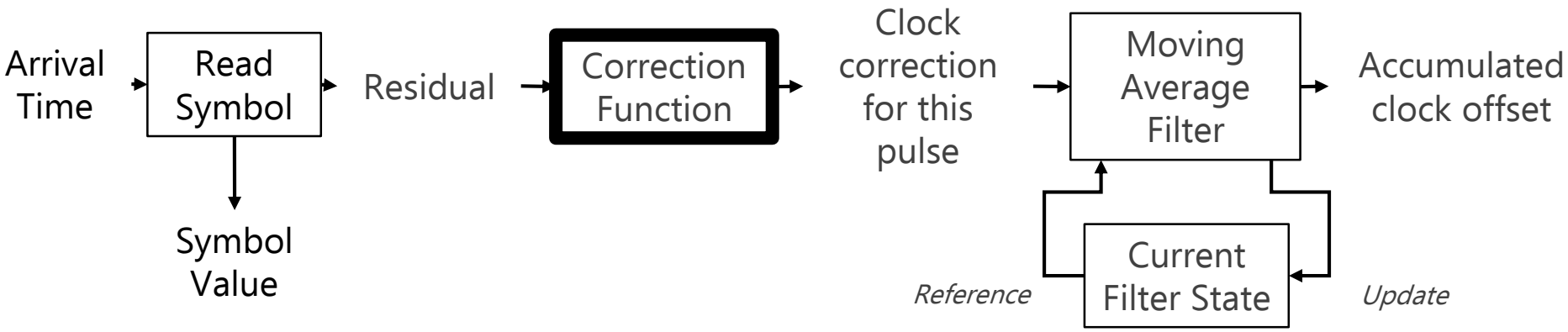
The Decoder

- When the receiver detects a pulse, a Time-to-Digital Converter records the Pulse Arrival Time
- The decoder determines what symbol this arrival time encodes
- The Residual is the undershoot or overshoot of the arrival time compared to the center of its intended time slot

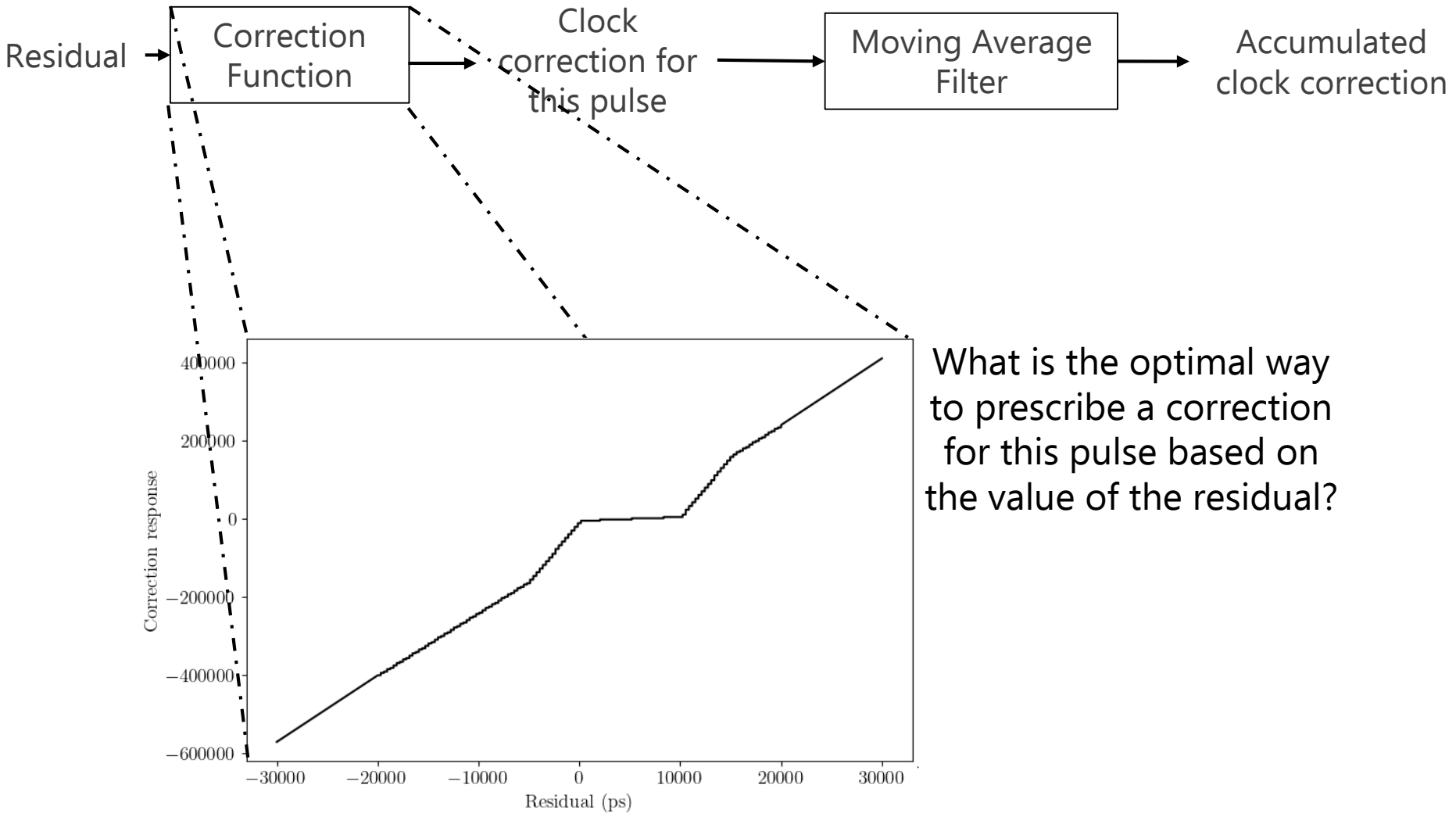


[2] Danielle E. Coogan et al (2022) *Development of CubeSat Spacecraft-to-Spacecraft Optical Link Detection Chain for the CLICK-B/C Mission*

The Decoder


Arrival Time
Read Symbol
Symbol Value
Residual
Correction Function
Clock correction for this pulse
Moving Average Filter
Accumulated clock offset
Reference
Current Filter State
Update

The Correction Function



What is the optimal way to prescribe a correction for this pulse based on the value of the residual?

The Lookup Table

- We want to:
  - Store the correction function on-board an FPGA
  - Accomplish all correction function calculations within two to eight clock cycles
  - Correct for accumulated drift due to frequency noise
  - Make the correction function amenable to numerical optimization
- We can achieve this with a Lookup Table
  - An array of corrections corresponding to set values of residuals
  - Continuous -> Discrete function
  - We can interpolate between points in the table if we have sufficient clock cycles

$$f_{correction}: R \rightarrow C$$

f(r1)	f(r2)	f(r3)	f(r4)	...	f(rn)
r1	r2	r3	r4	...	rn

The Point

- How do we optimize this Lookup Table version of the correction function?
  - Choose the correction entries which minimize the summed square symbol error
- How do we find these optimal correction entries?
  - Follow the gradient to the minimum?
  - Combine the gradient with curvature information to find the minimum?
- Why won't these conventional methods work?
  - Clock phase and frequency noise are normal random variables, so the shape of the performance index surface changes on each test.
  - Differencing on this surface results in suboptimal steps or precision loss
- Why not just assess with many more cases?
  - Are these methods worth the additional computation cost?
- Could we use optimization methods which do not require a derivative to find a solution?
  - As it turns out, yes. That's why I am presenting.

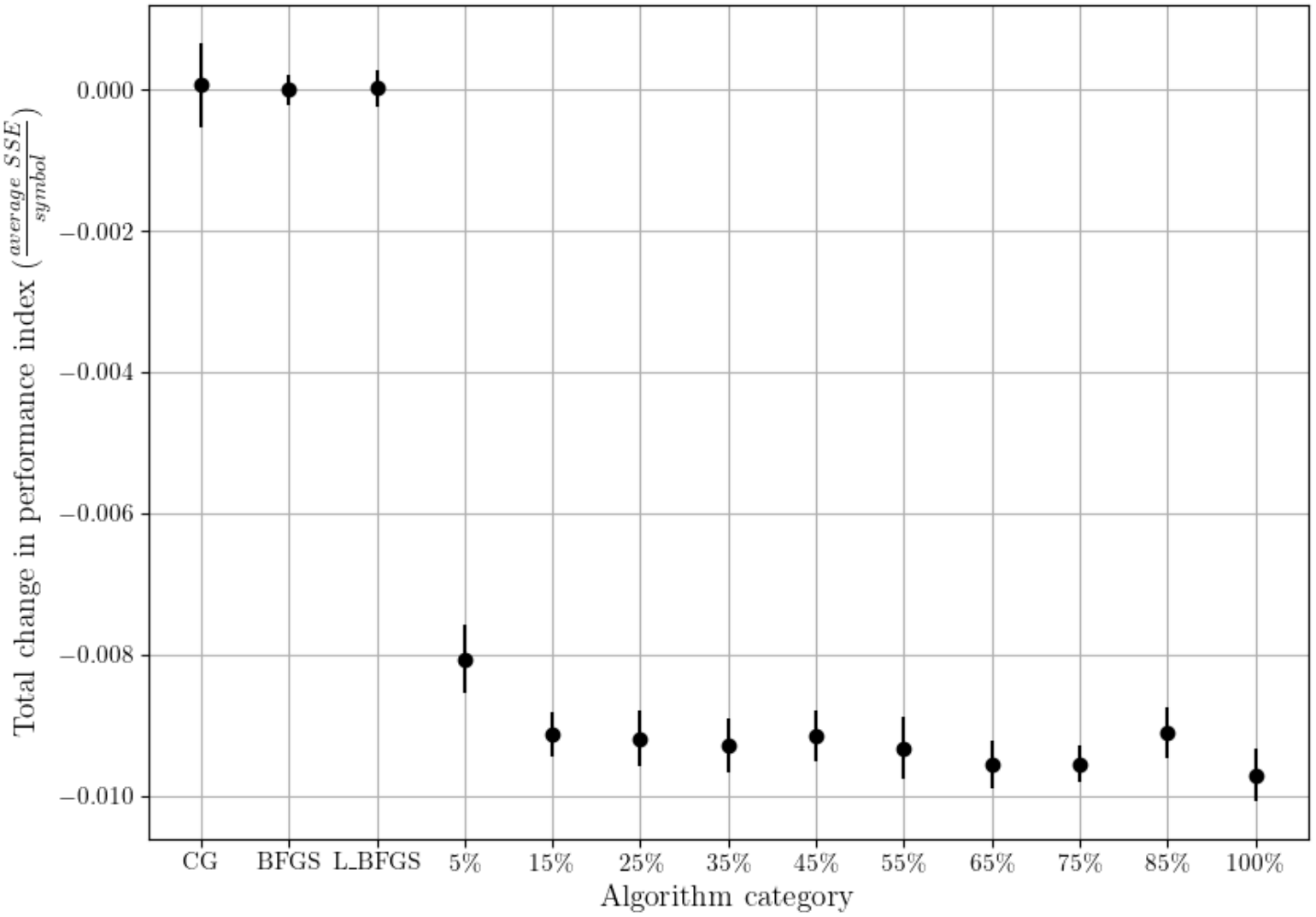
How each algorithm broadly works

- Zeroth-order: Evolutionary Algorithm
  1. Select best members of a population of Lookup Tables
  2. Randomly add a number to a small fraction of the entries in the chosen arrays
  3. Splice the arrays together
  4. Re-assess the performance of the arrays
  5. Repeat
- First-order: Conjugate Gradient Descent
  1. Obtain a performance index as a quadratic function of the independent variable
  2. Descend in the optimal gradient direction at the initial point
  3. Register the previous direction
  4. Find a new direction which is conjugate to all previous step directions
- Second-order: Broyden-Fletcher-Goldfarb-Shanno
  1. Approximate the Hessian at the current point
  2. Calculate the gradient at the point
  3. Solve for the search direction with gradient and approximate Hessian
  4. Step along search direction
  5. Update Hessian approximation using the gradient at the current and previous points

The Simulation

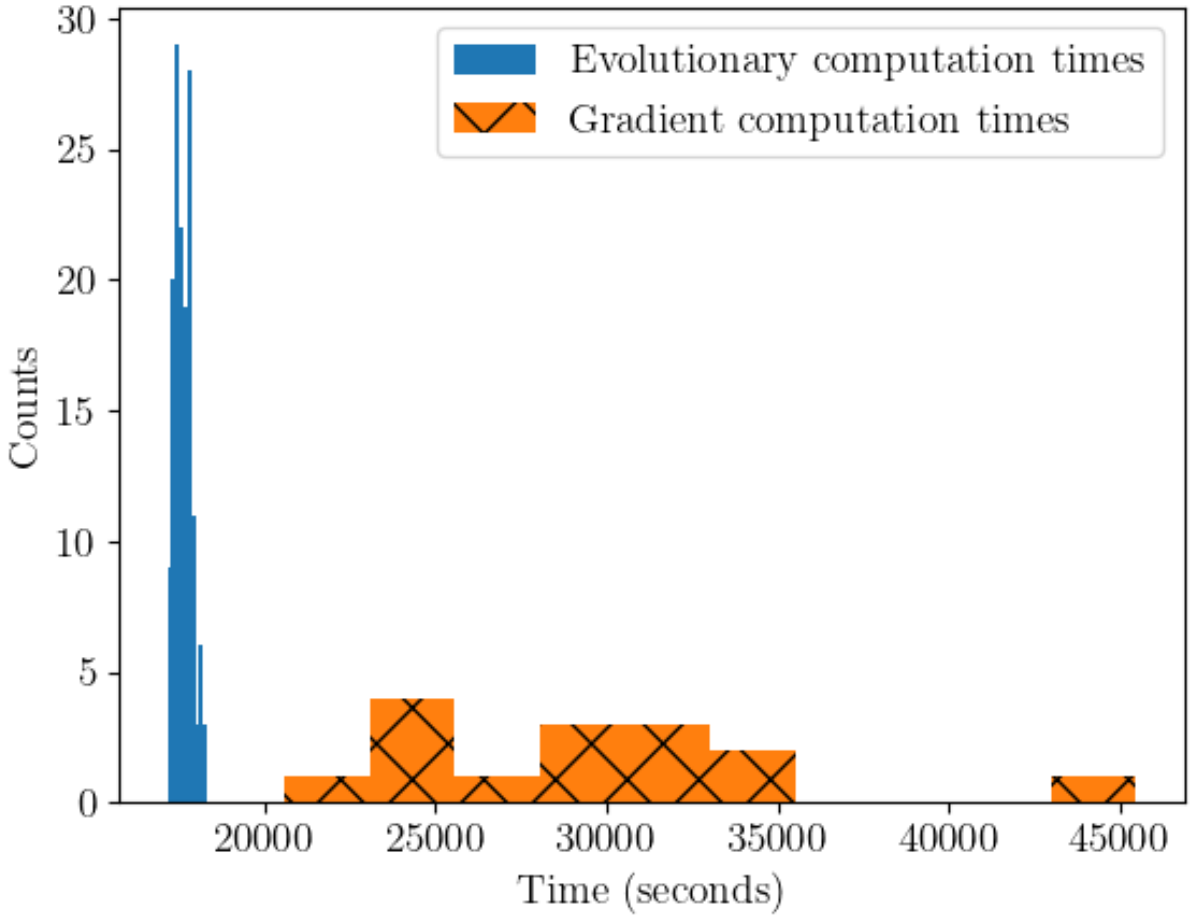
- Resources
  - 64 gigabytes of RAM
  - 32 CPUs
  - Run all calculations on a single machine with no other processes
- Methodology
  - Initial and final performance indices are calculated over 100,000 cases
  - Each optimization step is made after averaging over 10,000 cases
  - Average the results over 5 to 15 runs
  - Obtain confidence intervals for the true performance metrics
- Parameters
  - 128 entries
  - Each entry is an 8-bit float
  - 2500 Picosecond phase noise
  - 3e-9 picosecond/second frequency noise

Bootstrap intervals for change in squared symbol error

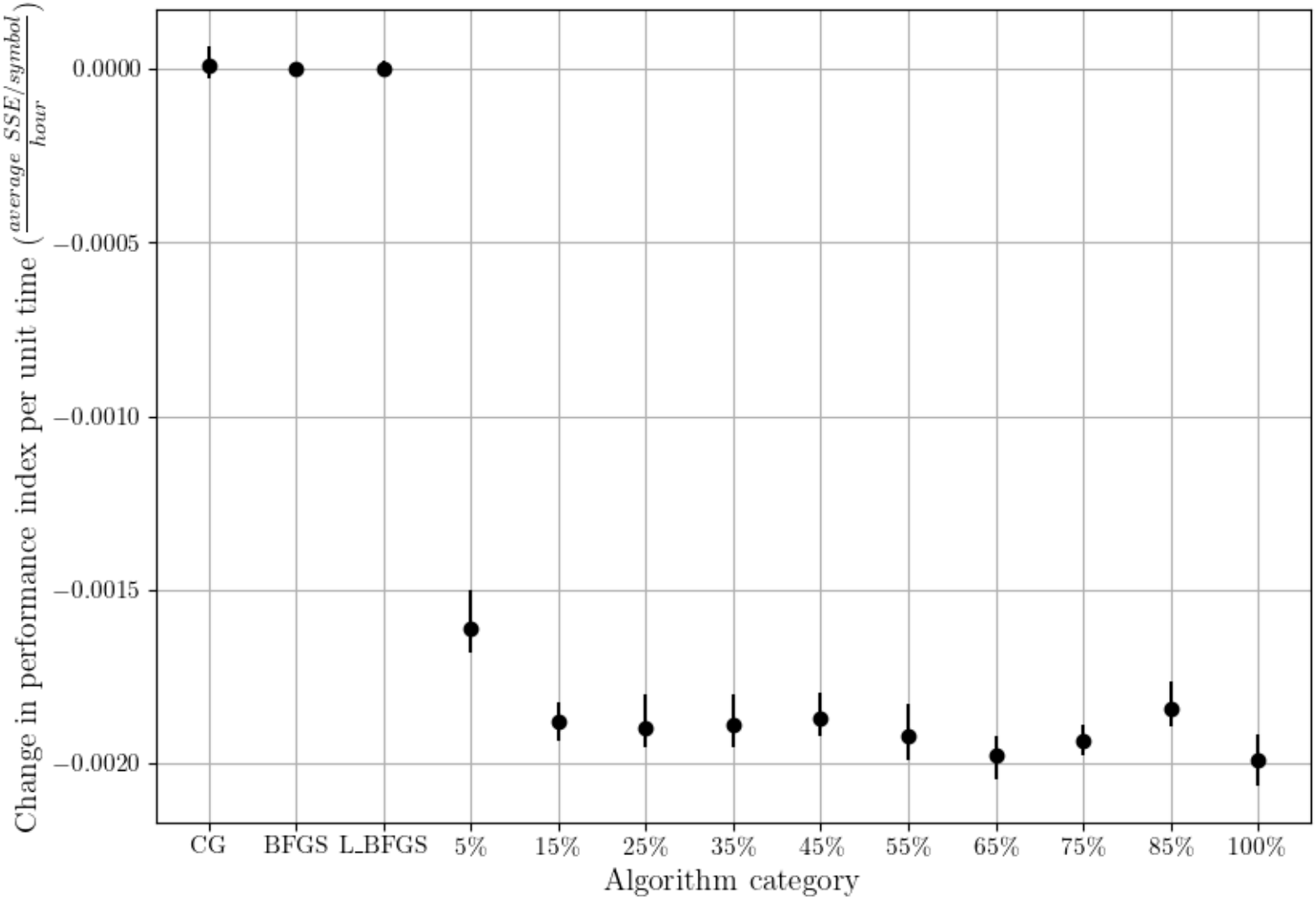


Histogram comparison for computation times

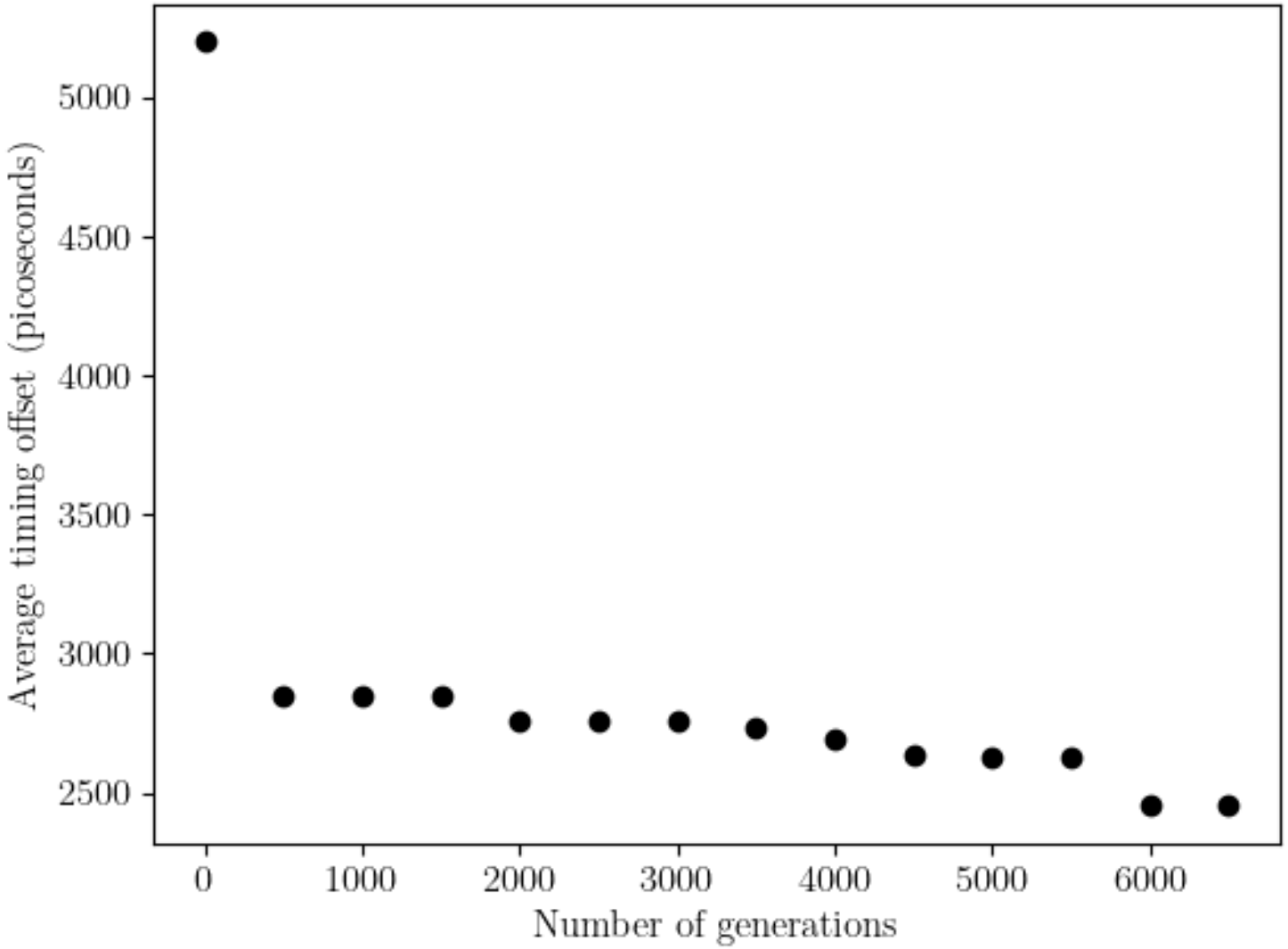
- Evolutionary algorithms
  - 10 different mutation fractions each run 15 times
  - 150 total runs
- Differencing-based algorithms
  - 3 different categories each run 5 times
  - 15 total runs



Bootstrap intervals for rate of change in squared symbol error



Average residual per symbol as a function of generations in an evolutionary algorithm



Discussion

- What does this suggest?
  - Zeroth-order optimization outperforms differencing optimization for problems with variable performance index surfaces.
- How can we apply this?
  - Optimizing the Lookup Table
  - Training RBF weights for the correction function
  - Training a Long Short-Term Memory network for the entire decoder