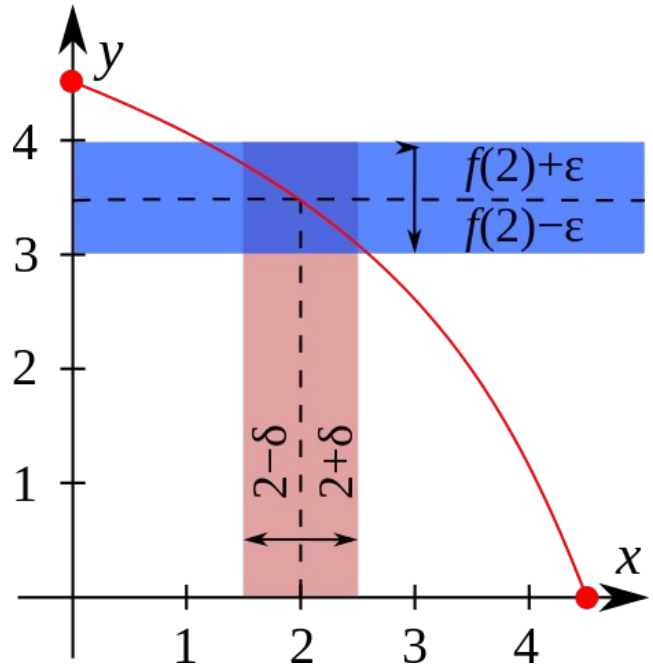
Nearby-**o**perand **C**ontinuous **Ap**proximation (NO❌CAP🧢 bruh.)

Group 10
Neel Shah, Owen Goebel, Zhixiang Teoh, Peter Ly
https://github.com/neel-one/nocap

Application+Motivation

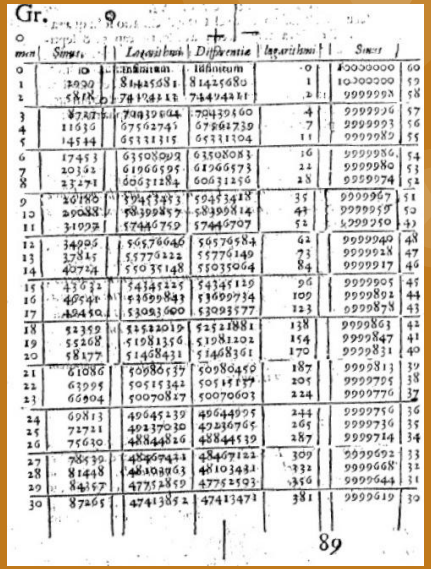
Video game graphics: trig functions

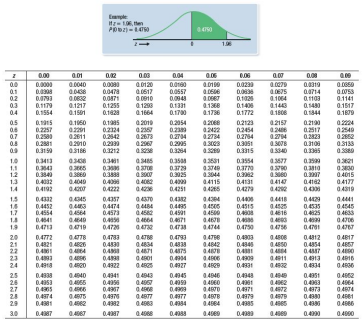
Neural networks: sigmoid function

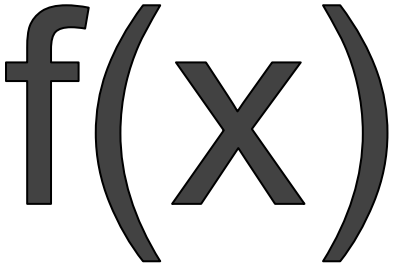


“[Logarithms,] by shortening the labors, doubled the life of an astronomer”

– Laplace (maybe)



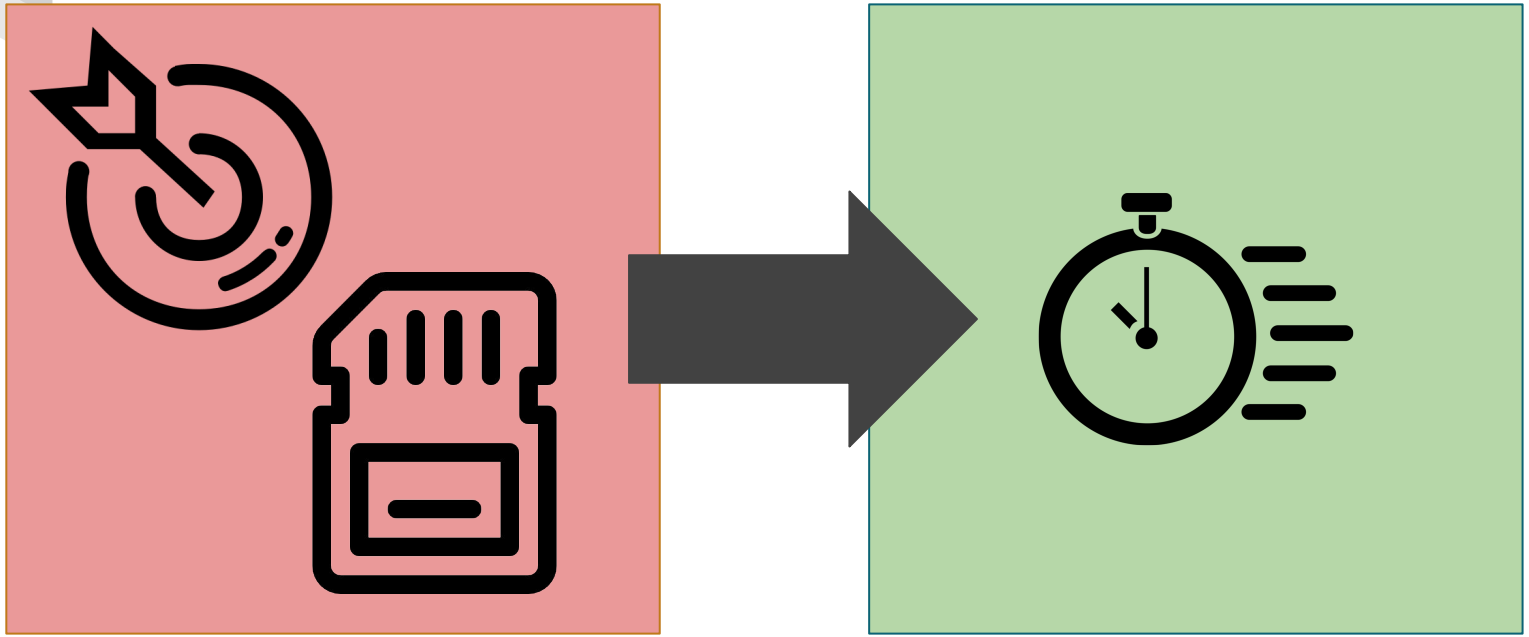




Input: **code**, **sample inputs**, and a C **function** f **: double → double**

```
nocap -bucketsFill -numBuckets 1000
      -testName blackscholes
      -func log
      -args "1 test/blackscholes/in_10M.txt /dev/null"
      build
```

Output: code with calls to f replaced with **lookup table** queries for f based on the inputs

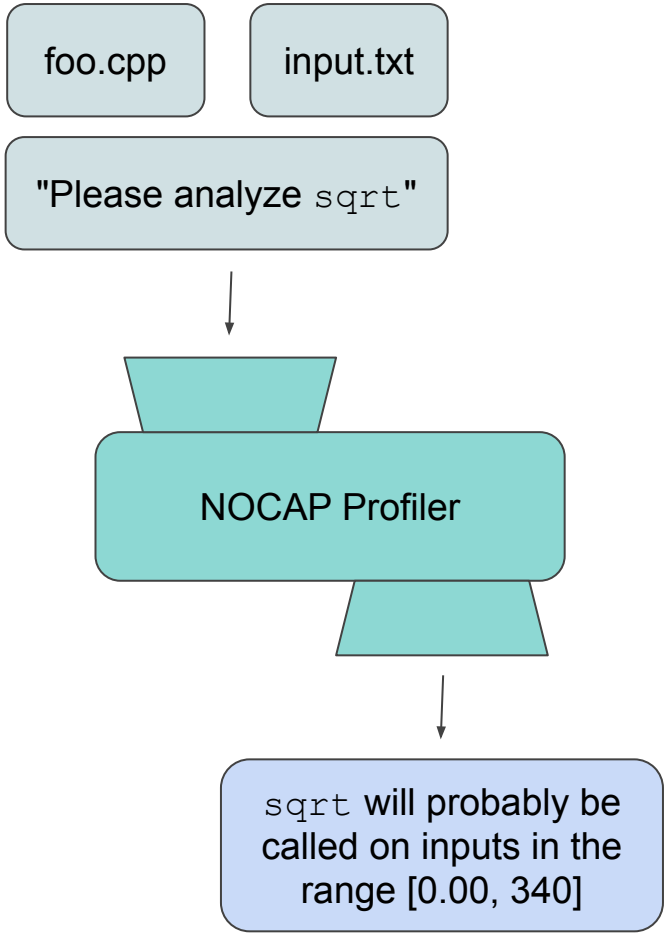


Description of the Process

Profiling Functions

1. Programmer inputs **target function(s)** and **provides inputs** to profile the functions on

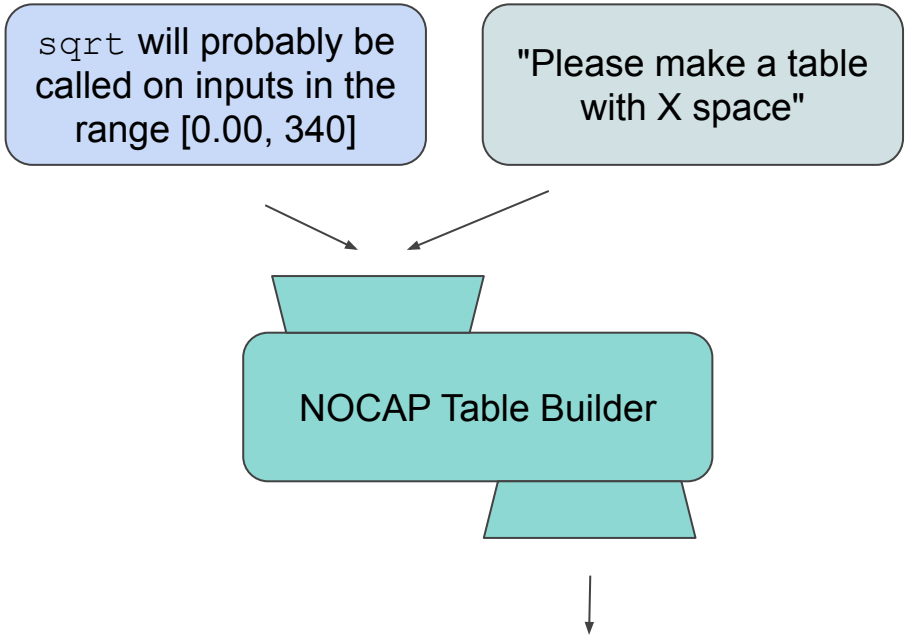
2. NOCAP uses an LLVM pass to profile the functions and **estimate the distribution of operands**



Building Tables

3. NOCAP **estimates good intervals** to include in the table based on the distribution of inputs

4. NOCAP **builds a table of function values** for the good intervals



Interval	[0,0.5]	[0.5,1.0]	...	[339.5, 340]
Value	sqrt(0.25)	sqrt(0.75)	...	sqrt(339.75)

Using Tables

5. NOCAP **modifies the source** to include the table of function values

6. NOCAP **modifies functions to perform table lookups** if the input value is within the table and resolve normally otherwise

```
// nocap_sqrt.c
double nocap_sqrt_table[] = { ... };

double nocap_sqrt(double x) {
    if (x in table range) {
        table_index = ...
        return nocap_sqrt_table[table_index];
    }
    return sqrt(x);
}
```

```
// target_program.c
#define sqrt(x) nocap_sqrt(x)
```

Demo

Benchmarks & Statistics

60%

Average speed up on a toy example

```
#include <math.h>
#include <stdio.h>

int main() {
        for (int j = 0; j < 1e7; j++) {
                for (int i = 0; i <= 20; i++) {
                double x = i;
                double y = exp(-x);
                }
        }
        return 0;
}
```

Benchmarks

$$C = N(d_1)S_t - N(d_2)Ke^{-rt}$$

$$\text{where } d_1 = \frac{\ln \frac{S_t}{K} + (r + \frac{\sigma^2}{2})t}{\sigma\sqrt{t}}$$

$$\text{and } d_2 = d_1 - \sigma\sqrt{t}$$

C = call option price

N = CDF of the normal distribution

S_t = spot price of an asset

K = strike price

r = risk-free interest rate

t = time to maturity

σ = volatility of the asset

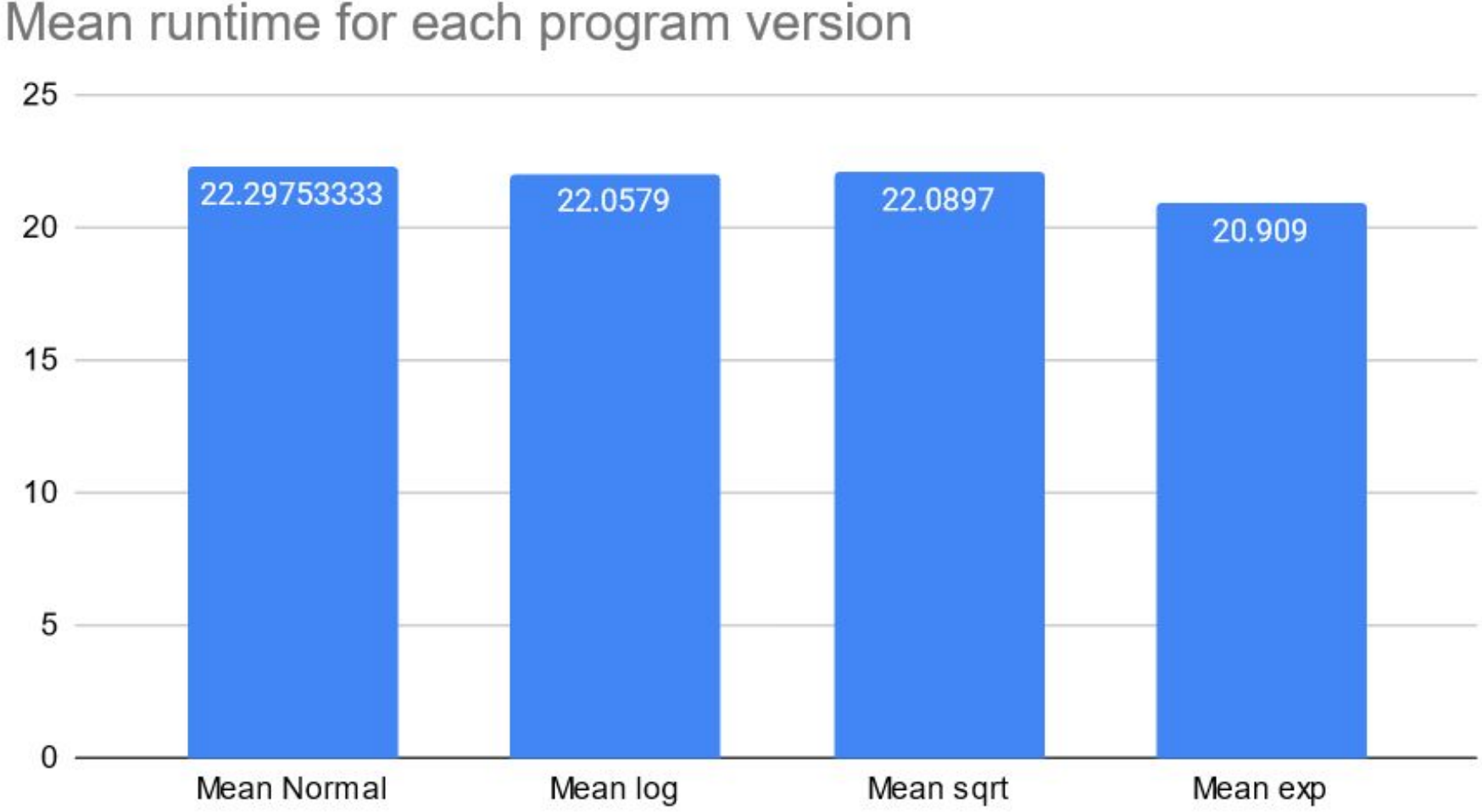
Black-Scholes is a financial model to estimate options pricing

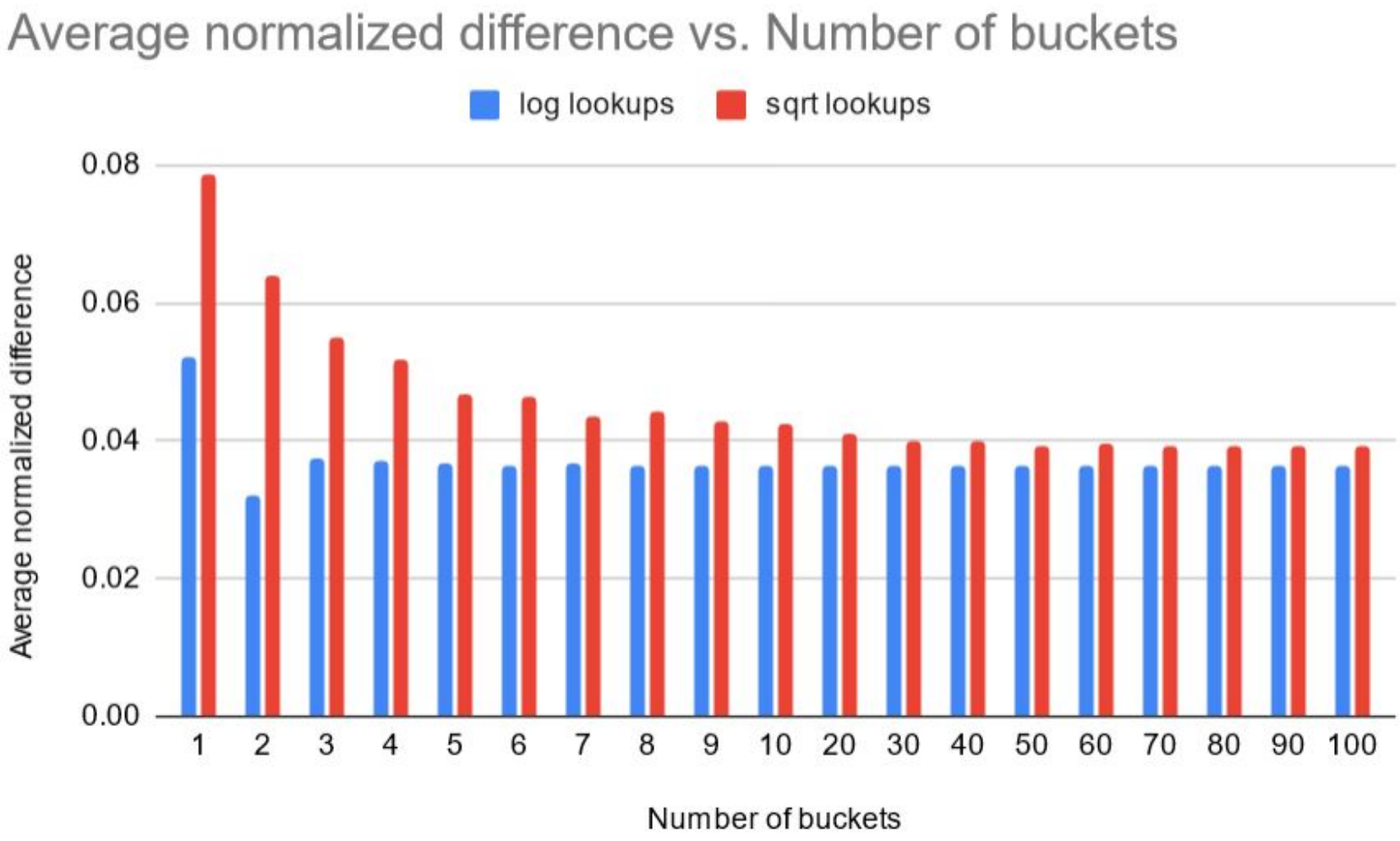
- uses <math.h> functions

  `exp()`, `log()`, `sqrt()`

- also used as a benchmark by ACCEPT framework[1]

[1] https://github.com/uwsampa/accept-apps


Mean runtime for each program version
25
20
15
10
5
0
22.29753333
22.0579
22.0897
20.909
Mean Normal
Mean log
Mean sqrt
Mean exp


Average normalized difference vs. Number of buckets
log lookups
sqrt lookups
0.08
0.06
0.04
0.02
0.00
Average normalized difference
1
2
3
4
5
6
7
8
9
10
20
30
40
50
60
70
80
90
100
Number of buckets

.5-60%
speedup

tunable
memory

Dominated by profiling
compilation time

0.036
avg normalized error

Q&A