

Nearby-operand Continuous Approximation (NO ~~X~~ CAP 🧢 bruh.)

Group 10

Neel Shah, Owen Goebel, Zhixiang Teoh, Peter Ly

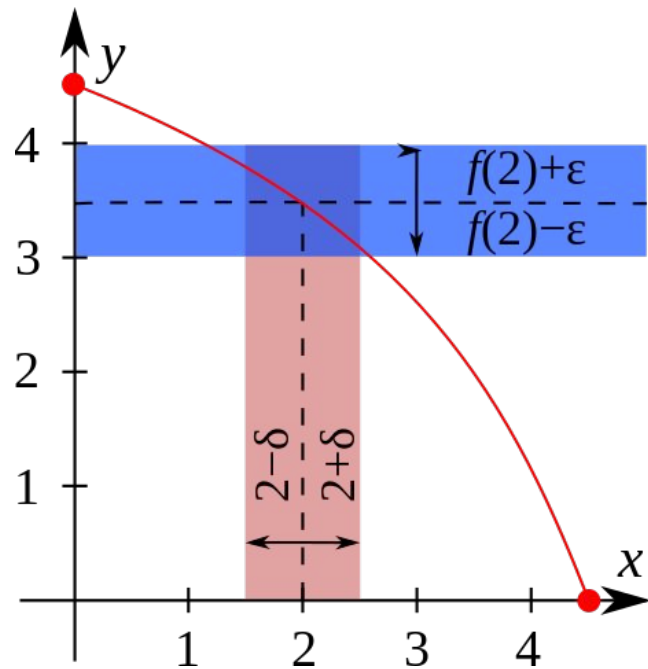
<https://github.com/neel-one/nocap>



Application+Motivation

Video game graphics: trig functions

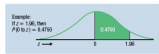
Neural networks: sigmoid function



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

– Laplace (maybe)

Gr.	0	1	2	3	4	5	6	7	8	9
mon	Summ.	Logarithm	Differentia	Arithmetica	Sines					
0	10	10000000	10000000	0	10000000	60				
1	1000	81445031	81445680	1	10000000	59				
2	10000	79183112	79183441	2	9999999	58				
3	100000	76820644	76820960	3	9999998	57				
4	1000000	74458176	74458479	4	9999997	56				
5	10000000	72095708	72095994	5	9999996	55				
6	17451	61000000	61000081	16	9999986	54				
7	20131	61966596	61966573	21	9999980	53				
8	21271	60651284	60651256	28	9999974	52				
9	20186	59453451	59453418	35	9999967	51				
10	20028	58309857	58309814	41	9999959	50				
11	31092	57246759	57246707	51	9999950	49				
12	34995	56276646	56276584	64	9999940	48				
13	37845	55379122	55379049	81	9999928	47				
14	40754	54551148	54551064	102	9999917	46				
15	43731	53781112	53781019	127	9999905	45				
16	46776	53069841	53069734	157	9999891	44				
17	49890	52409400	52409277	192	9999878	43				
18	53073	51792019	51791881	231	9999861	42				
19	56326	51210356	51210202	274	9999847	41				
20	59649	50664811	50664661	321	9999831	40				
21	63042	50155657	50155496	372	9999811	39				
22	66505	50673144	50672977	427	9999795	38				
23	70038	50217672	50217501	486	9999776	37				
24	73641	49789639	49789467	549	9999760	36				
25	77314	49388526	49388354	616	9999746	35				
26	81057	48913812	48913640	687	9999734	34				
27	84870	48465098	48464926	762	9999722	33				
28	88753	48042884	48042712	840	9999711	32				
29	92706	47646670	47646498	921	9999700	31				
30	96729	47275956	47275784	1005	9999690	30				



z	0.00	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09
0.0	0.5000	0.5040	0.5080	0.5120	0.5160	0.5199	0.5238	0.5277	0.5315	0.5353
0.1	0.5398	0.5438	0.5478	0.5517	0.5557	0.5596	0.5636	0.5675	0.5714	0.5753
0.2	0.5793	0.5832	0.5871	0.5910	0.5948	0.5987	0.6026	0.6064	0.6103	0.6141
0.3	0.6179	0.6217	0.6255	0.6293	0.6331	0.6368	0.6406	0.6443	0.6480	0.6517
0.4	0.6554	0.6591	0.6628	0.6664	0.6700	0.6736	0.6772	0.6808	0.6844	0.6879
0.5	0.6915	0.6950	0.6985	0.7019	0.7054	0.7088	0.7123	0.7157	0.7190	0.7224
0.6	0.7257	0.7291	0.7324	0.7357	0.7389	0.7421	0.7453	0.7484	0.7515	0.7546
0.7	0.7577	0.7608	0.7638	0.7668	0.7697	0.7726	0.7755	0.7783	0.7811	0.7839
0.8	0.7867	0.7894	0.7921	0.7948	0.7974	0.7999	0.8025	0.8050	0.8075	0.8100
0.9	0.8124	0.8148	0.8171	0.8193	0.8215	0.8236	0.8257	0.8278	0.8298	0.8318
1.0	0.8338	0.8358	0.8377	0.8395	0.8413	0.8431	0.8448	0.8465	0.8481	0.8498
1.1	0.8514	0.8530	0.8547	0.8562	0.8577	0.8591	0.8605	0.8619	0.8633	0.8646
1.2	0.8659	0.8673	0.8687	0.8699	0.8713	0.8726	0.8738	0.8750	0.8762	0.8774
1.3	0.8786	0.8798	0.8810	0.8821	0.8832	0.8843	0.8854	0.8864	0.8875	0.8885
1.4	0.8895	0.8905	0.8915	0.8925	0.8935	0.8944	0.8953	0.8962	0.8971	0.8980
1.5	0.8989	0.8997	0.9006	0.9015	0.9023	0.9031	0.9039	0.9047	0.9055	0.9063
1.6	0.9071	0.9078	0.9086	0.9093	0.9099	0.9106	0.9112	0.9119	0.9125	0.9132
1.7	0.9138	0.9144	0.9150	0.9156	0.9161	0.9167	0.9172	0.9177	0.9182	0.9187
1.8	0.9191	0.9196	0.9201	0.9206	0.9211	0.9216	0.9221	0.9226	0.9230	0.9235
1.9	0.9239	0.9244	0.9248	0.9252	0.9256	0.9260	0.9264	0.9268	0.9271	0.9275
2.0	0.9279	0.9282	0.9285	0.9288	0.9291	0.9294	0.9297	0.9299	0.9302	0.9304
2.1	0.9306	0.9308	0.9310	0.9312	0.9314	0.9316	0.9317	0.9319	0.9320	0.9321
2.2	0.9322	0.9323	0.9324	0.9325	0.9326	0.9327	0.9328	0.9328	0.9329	0.9329
2.3	0.9330	0.9330	0.9331	0.9331	0.9332	0.9332	0.9332	0.9332	0.9332	0.9332
2.4	0.9332	0.9332	0.9332	0.9332	0.9332	0.9332	0.9332	0.9332	0.9332	0.9332
2.5	0.9332	0.9332	0.9332	0.9332	0.9332	0.9332	0.9332	0.9332	0.9332	0.9332
2.6	0.9332	0.9332	0.9332	0.9332	0.9332	0.9332	0.9332	0.9332	0.9332	0.9332
2.7	0.9332	0.9332	0.9332	0.9332	0.9332	0.9332	0.9332	0.9332	0.9332	0.9332
2.8	0.9332	0.9332	0.9332	0.9332	0.9332	0.9332	0.9332	0.9332	0.9332	0.9332
2.9	0.9332	0.9332	0.9332	0.9332	0.9332	0.9332	0.9332	0.9332	0.9332	0.9332
3.0	0.9332	0.9332	0.9332	0.9332	0.9332	0.9332	0.9332	0.9332	0.9332	0.9332

f(x)

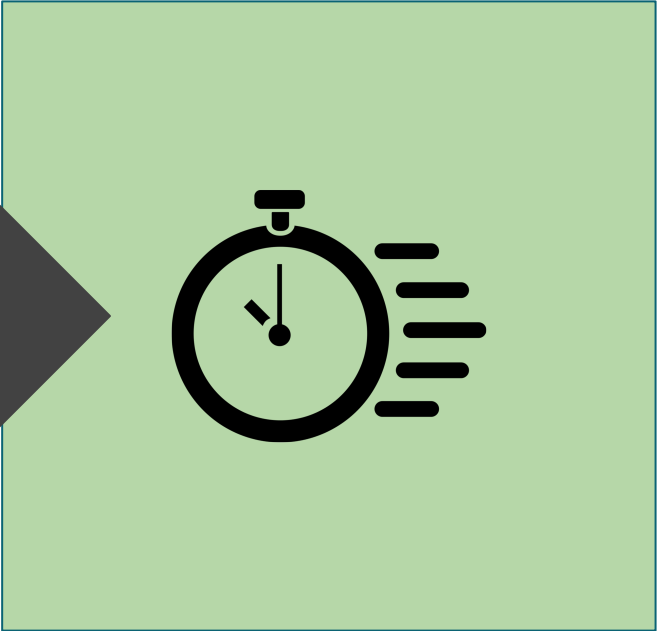
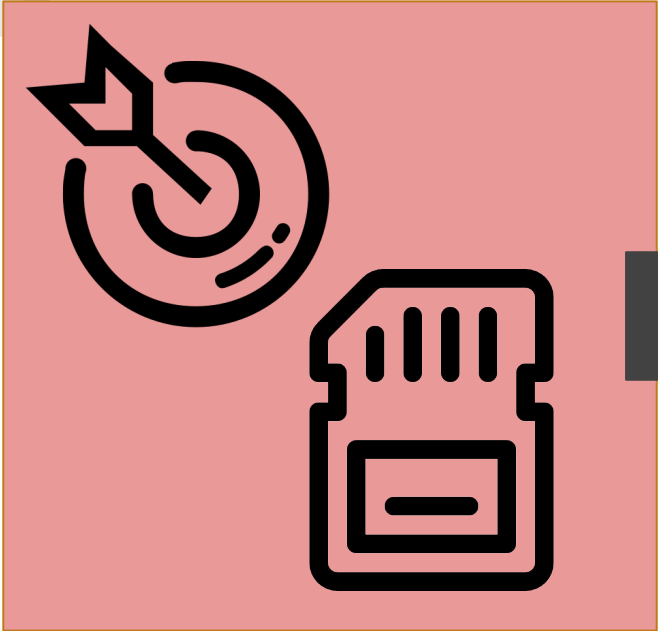
Input: code, sample inputs, and a C function f: double → double

```

nocab -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





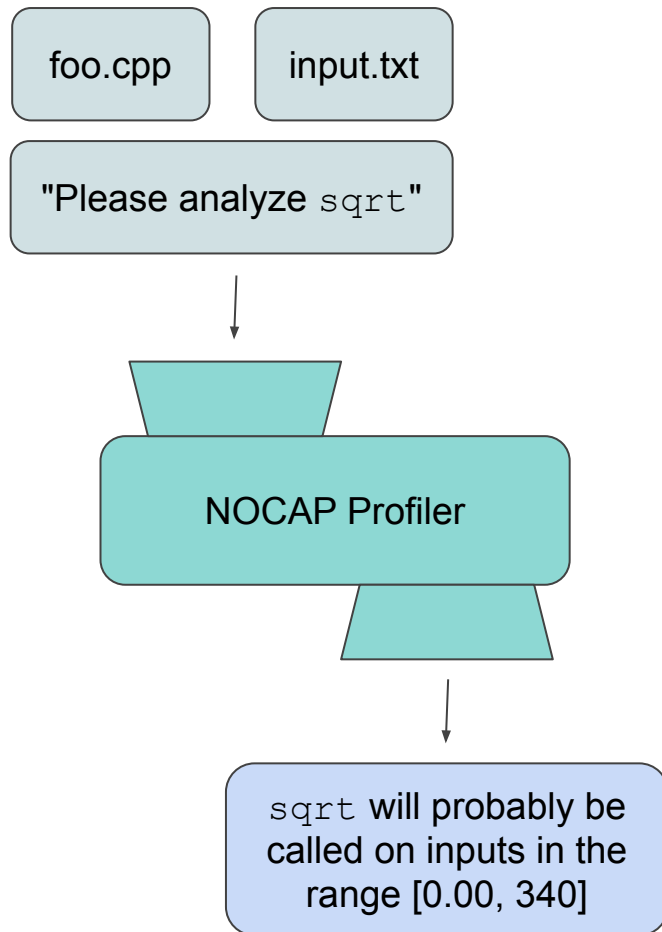
Description of the Process





Profiling Functions

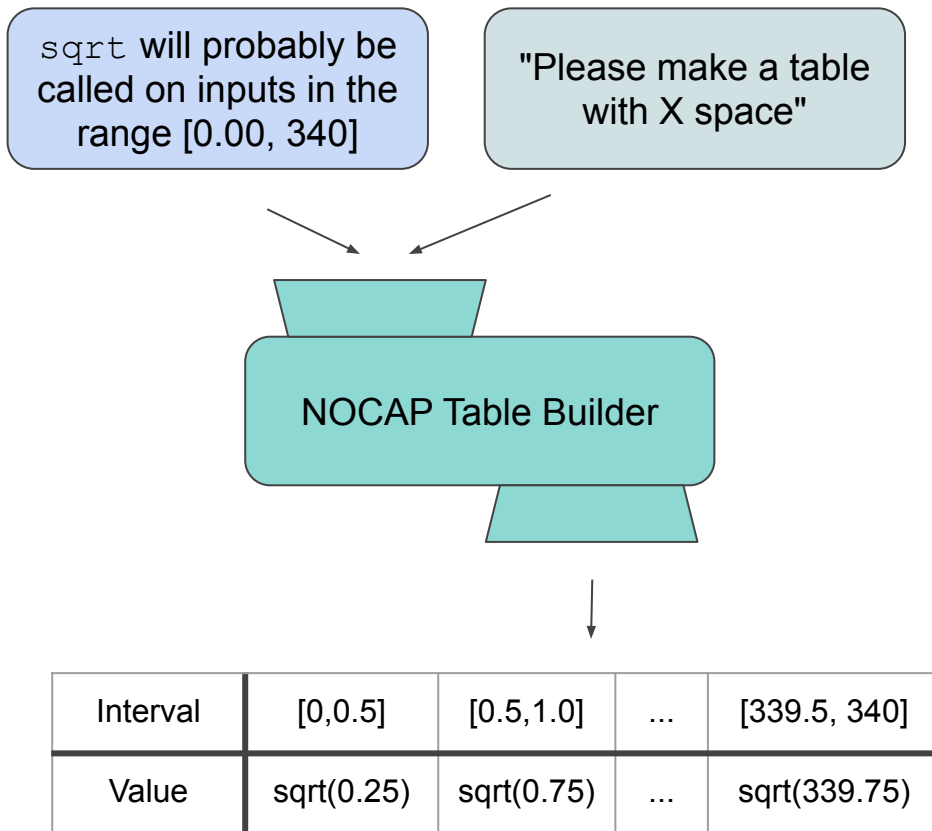
1. Programmer inputs **target function(s)** and **provides inputs** to profile the functions on
2. NOCAP uses an LLVM pass to profile the functions and **estimate the distribution of operands**





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





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

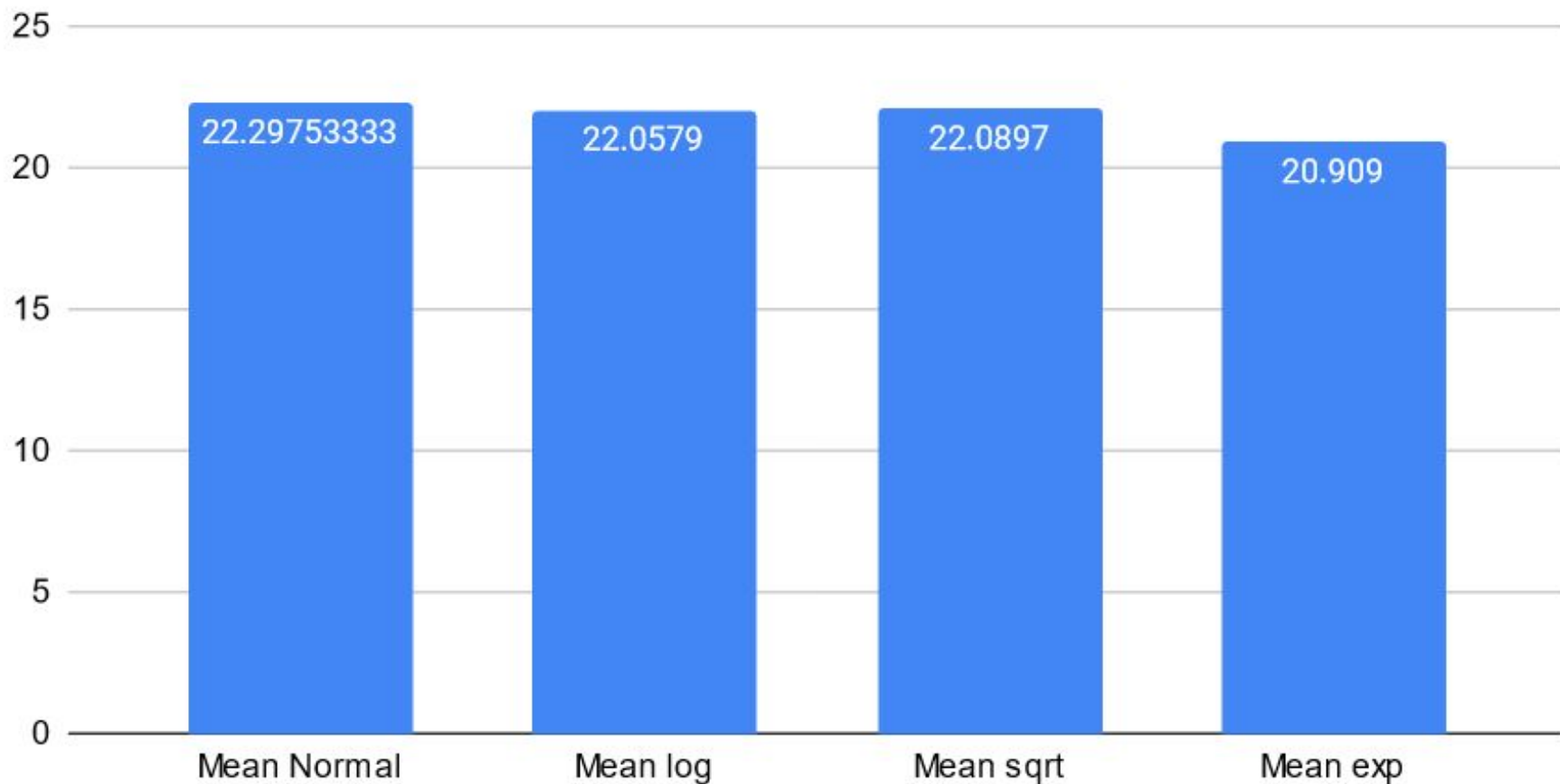
σ = volatility of the asset

Black-Scholes is a financial model to estimate options pricing

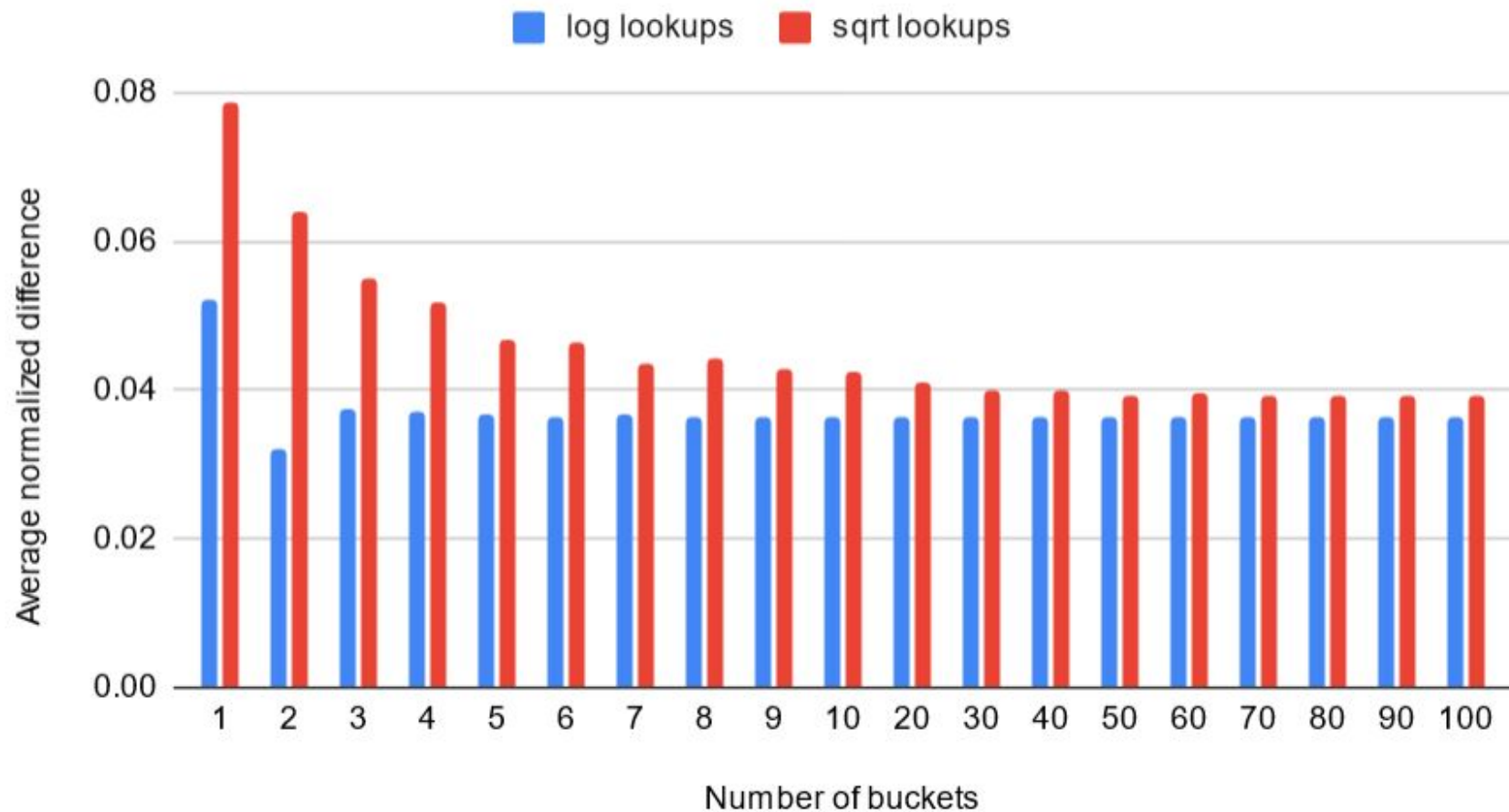
- uses `<math.h>` functions
`exp()`, `log()`, `sqrt()`
- also used as a benchmark by ACCEPT framework¹

¹<https://github.com/uwsampa/accept-apps>

Mean runtime for each program version



Average normalized difference vs. Number of buckets



.5-60%

speedup

tunable

memory

**Dominated by
profiling**

compilation time

0.036

avg normalized error



Q&A