The University of Texas at Austin
Operations Research and
Industrial Engineering

Introduction to Code Optimization

Sudesh K Agrawal

INFORMS Student Chapter Workshop

The University of Texas at Austin

May 3, 2021

Outline

Introduction

- Components of code optimization
  - Data Structure & Algorithm
  - Code Implementation
  - Hardware Optimization
- Primary components of a computer—CPU, Memory, Storage, etc.
- Registers vs. Cache
- Clock cycle—smallest unit of time to perform a task
- Addition (3 cycles), multiplication (6 cycles), division (30–60 cycles)

Basic Operation

- $x = y + z$
- Fetch instruction from memory and decode it
- Hard disk $\rightarrow$ RAM $\rightarrow$ CPU (Cache and Registers)
- Fetch data y and z from memory. How long does it take? (300 cycles.)
- Perform addition. How long does this take? (3 cycles.)
- Retrieve result, and put it in x.

How long does the actual addition take?

- Each pipeline stage happens sequentially.

Pipelines and Data Streams

- $x(i) = y(i) + z(i), \quad i \in \{1, 2, \ldots n\}$
- Can we do better?
- Use data stream
- Reduce latency: fetch next instruction and data while the previous instruction is executing

Can we do even better?

Vectorization

- No data dependency, unroll the loop
  - $i \in \{1, 5, 9, \ldots\}$
  - $x(i) = y(i) + z(i)$
  - $x(i+1) = y(i+1) + z(i+1)$
  - $x(i+2) = y(i+2) + z(i+2)$
  - $x(i+3) = y(i+3) + z(i+3)$
- Vector instructions: single message multiple data
- Vector registers comprising multiple cache lines

Python Activities

Python Loops

- Python loops are slower compared to Java, C, etc. (Why?)

What are the alternatives available to looping constructs?

- List Comprehension and Set Comprehension
- Vectorization using numpy library
  - Broadcasting for more complex scenarios
- Just-in-time compilation (JIT)

What if you are stuck with a python for-loop?

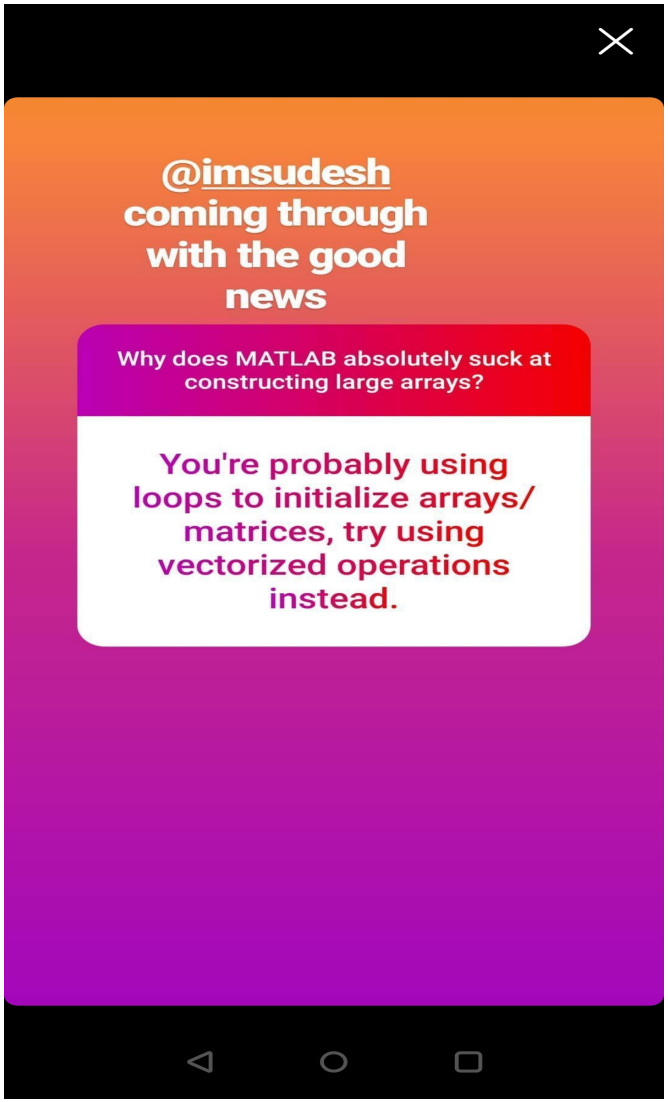
Activity 1: Simple Exponential Smoothing

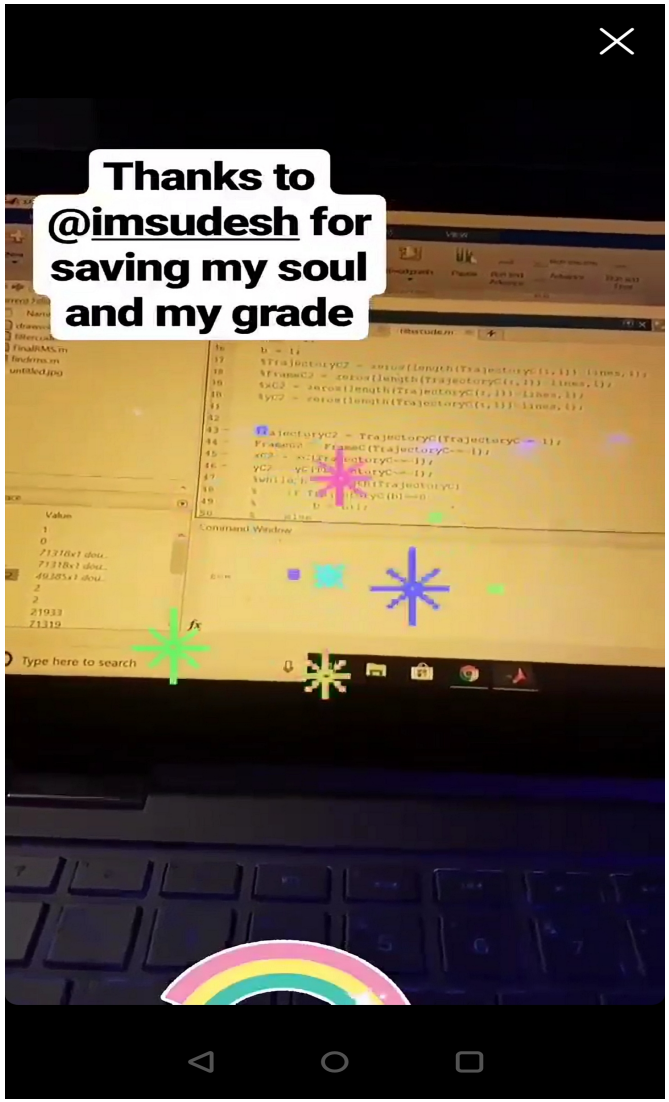
Write a python function to generate the exponential smoothing prediction of a series given as a numpy array. Your function should take the series and the α value as input. Time your function call using the "timeit" python function for the series: $0, 0, 4.58, 0, 5.30, 3.19, 1.3$, and $\alpha = 0.1$. You are allowed to use python libraries, even for directly calculating the output. (Check that your function returns 1.1468938.)

SES formula

$$x_{\text{forecast}} = \alpha \cdot x_{\text{current observation}} + (1 - \alpha) \cdot x_{\text{previous forecast}}.$$

Anecdote: Instagram Story (2018)





Tips for Optimizing for-loops

- Minimize dots
- Prefer local variables over global variables
- Multiplication/Division are relatively expensive operations.

Activity 2: Optimize for-loops

- Optimize the following code:

```
for(int i=2; i<=n-1; i++)
{
    z[i] = x[i]/a + y[i]/a;
}
```

Any caveats to your optimized code?

Activity 2: My solution

```
b = 1.0/a
for(int i=2; i<=n-1; i++)
{
    z[i] = (x[i] + y[i])*b;
}
```

Final Remarks

- Optimize only when needed.
- Identify bottle-necks in your code using profilers.

Thank You!

Questions?