I Solved FizzBuzz With Tensors and Accidentally Did Signal Processing

FizzBuzz is that classic coding interview problem: print numbers 1 to 100, replacing multiples of 3 with "Fizz", multiples of 5 with "Buzz", and multiples of both with "FizzBuzz". Most people solve it with if-statements in about 30 seconds and move on.

I came across Susam Pal's article "Fizz Buzz With Cosines", where he solved FizzBuzz using trigonometric functions instead of conditionals. The key insight: divisibility is periodic. Multiples of 3 repeat every 3 numbers, multiples of 5 repeat every 5 numbers. This led me to wonder: if FizzBuzz is fundamentally a periodic function, why not represent it as a tensor?

Repository: https://github.com/aaronsb/fizzbuzztensor

The Pattern Emerges

Here's what I discovered: FizzBuzz repeats every 15 numbers. This happens because 15 is the least common multiple (LCM) of 3 and 5. Once you see this, the entire infinite sequence collapses into a single 15-element pattern vector:

```
PATTERN = [0, 0, 1, 0, 2, 1, 0, 0, 1, 2, 0, 1, 0, 0, 3]
```

The encoding is simple: 0 means print the number, 1 means "Fizz", 2 means "Buzz", and 3 means "FizzBuzz". Every number from 1 to infinity maps to one of these 15 positions via modular arithmetic. For any position n, the FizzBuzz category is just `PATTERN[(n-1) % 15]`. That's the complete solution in constant space with O(1) lookup time.

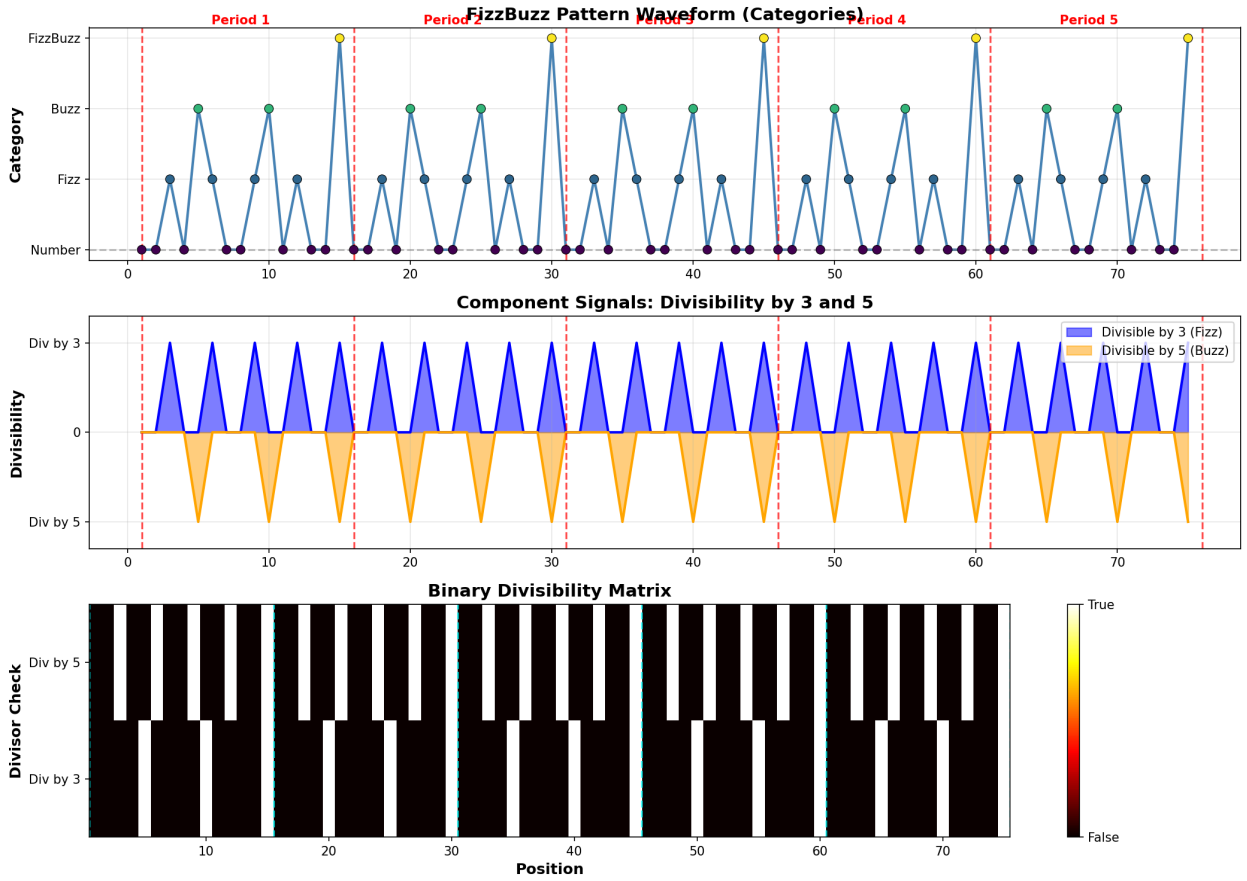
The pattern vector can be computed directly from divisibility checks using NumPy broadcasting:

```
nums = np.arange(1, 16)[:, None]        # (15, 1)
divisors = np.array([3, 5])[None, :]    # (1, 2)
div_matrix = (nums % divisors == 0).astype(int)  # (15, 2)
PATTERN = div_matrix @ np.array([1, 2])  # Encode as single
vector
```

This creates a rank-2 divisibility matrix that compresses to a rank-1 categorical vector through linear transformation. We've moved from algorithmic thinking (check conditions step-by-step) to declarative thinking (describe the structure, then look it up).

Viewing FizzBuzz as a Signal

Once you have a repeating numerical pattern, you can treat it as a discrete signal. I plotted FizzBuzz as a waveform over five periods:

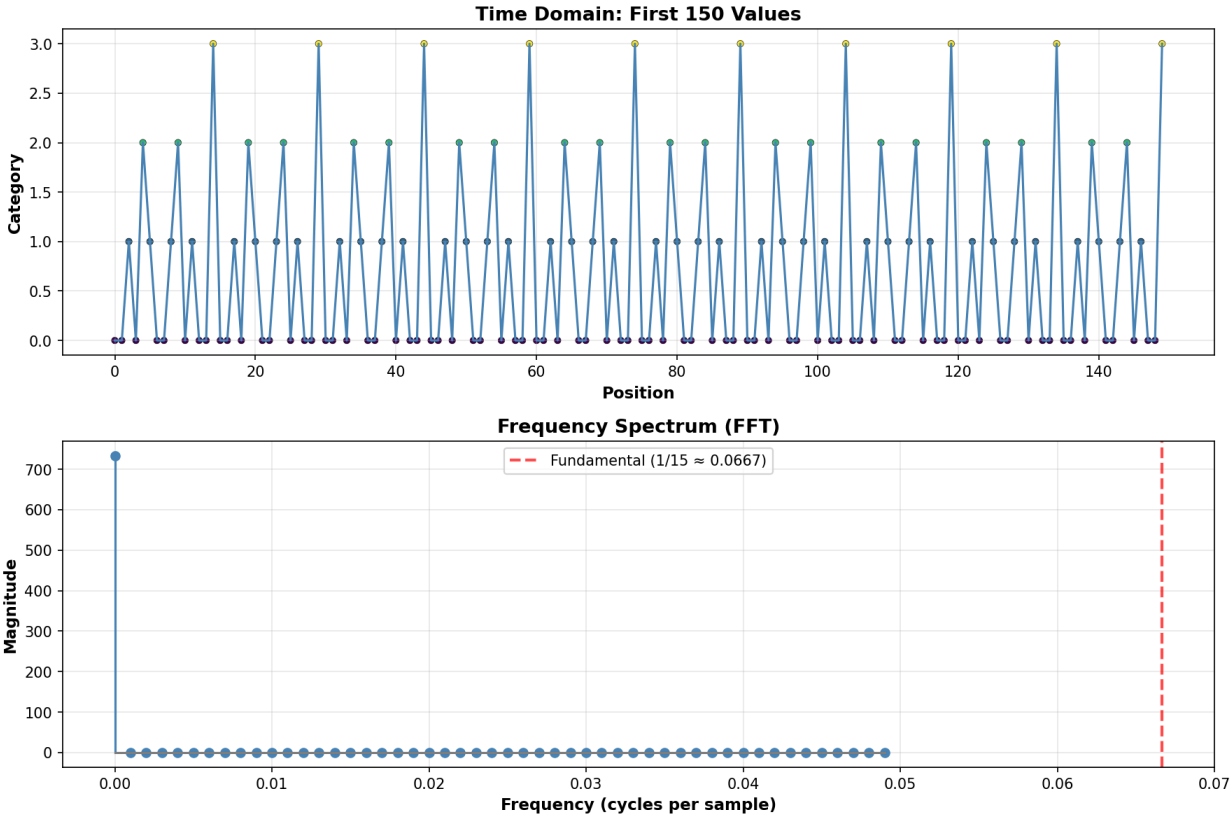


The top panel shows the category values (0, 1, 2, 3) as a repeating wave. The middle panel reveals something interesting: divisibility by 3 and divisibility by 5 are separate component frequencies that interfere to create the full FizzBuzz pattern. These two signals combine like waves, and where they overlap you get "FizzBuzz".

The bottom panel shows the binary divisibility matrix as a heatmap. The pattern structure becomes visually obvious in a way that's invisible when you're just writing if-statements.

The Frequency Domain

Since FizzBuzz is periodic, it has a well-defined frequency spectrum. Applying the Discrete Fourier Transform:

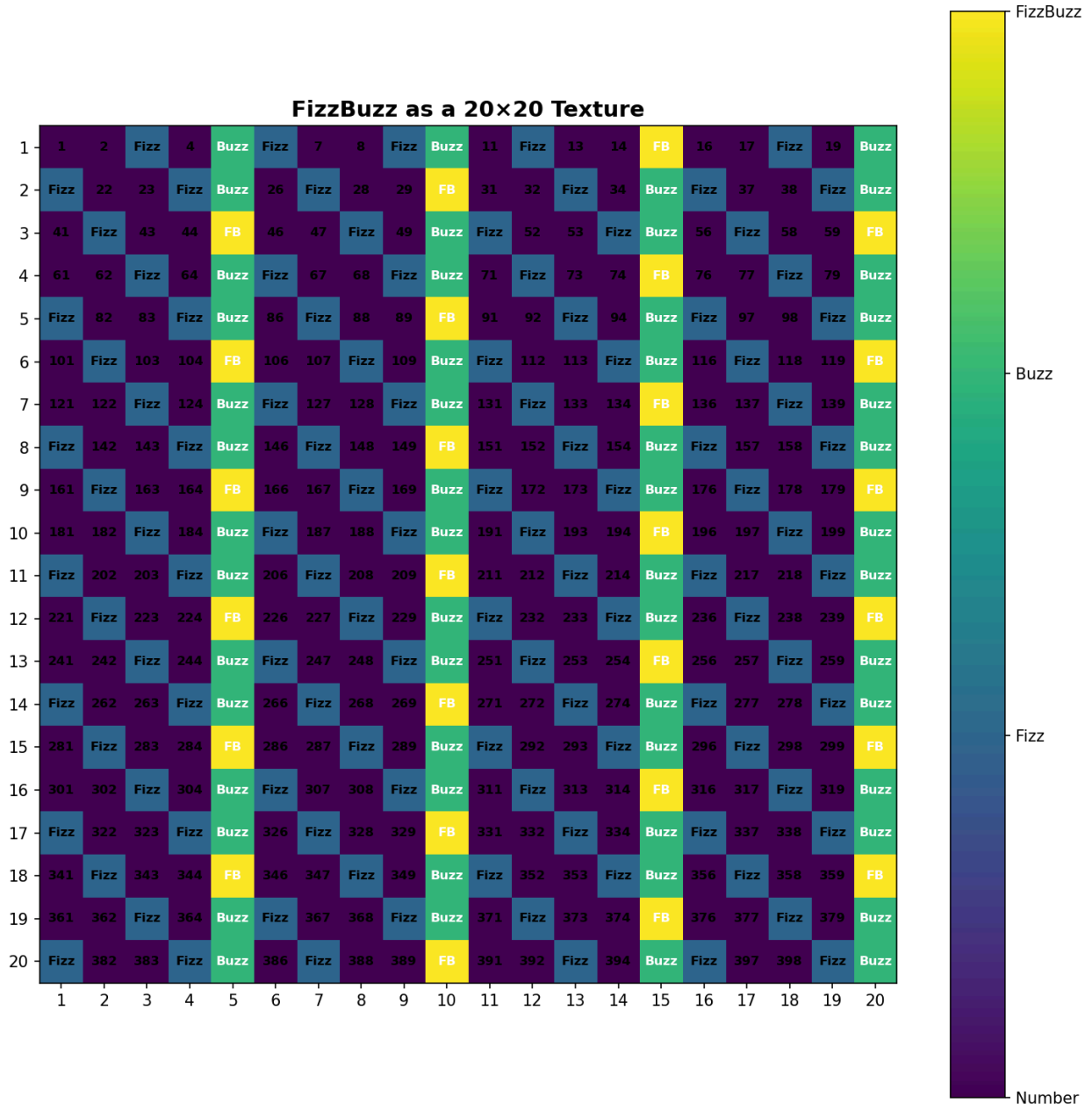


The spectrum shows a strong fundamental frequency at 1/15 ≈ 0.0667 cycles per sample, exactly matching our period of 15. This is what you'd expect from a perfectly periodic signal. The large DC component represents the average value across all categories.

This connects directly to the trigonometric approach that inspired this work. The cosine-based solution exploits these same periodic properties, encoding divisibility checks as sinusoidal functions. Both approaches recognize that FizzBuzz has spectral structure, whether you represent it explicitly as a pattern vector or implicitly through trig functions.

Spatial Patterns

Arranging FizzBuzz values in a 2D grid reveals spatial structure:



Notice the diagonal yellow stripe where multiples of 15 align. The pattern tiles perfectly because the fundamental period (15) determines all the structure. What looks like random variation in one dimension becomes organized geometry in two dimensions.

This visualization technique applies broadly in data science. When time series data or sequences don't make sense linearly, reshaping them into higher-dimensional arrays can reveal hidden patterns. It's the same principle behind spectrograms, confusion matrices, and many other visualization tools.

Representation Learning

This project demonstrates a concept central to machine learning: how you represent data determines what patterns you can extract. As if-statements, FizzBuzz is procedural code. As a pattern vector, it's a mathematical object with well-defined periodic and spectral properties.

The divisibility matrix approach generalizes naturally. For divisors {3, 5, 7}, the period becomes LCM(3,5,7) = 105, and you get $2^3 = 8$ possible categories (each subset of active divisors). The pattern vector grows to 105 elements, but the lookup remains O(1). The same broadcasting technique works for any divisor set:

```
def create_pattern(divisors):
    div_values = [d for d, _ in divisors]
    period = np.lcm.reduce(div_values)
    nums = np.arange(1, period + 1)[:, None]
    div_array = np.array(div_values)[None, :]
    div_matrix = (nums % div_array == 0).astype(int)
    powers = 2 ** np.arange(len(divisors))
    return div_matrix @ powers
```

This is what transformers do with language, what CNNs do with images, and what embeddings do with categorical data: they find representations where patterns become explicit rather than implicit.

What This Teaches

Taking FizzBuzz seriously as a signal processing problem reveals connections between domains that aren't obvious when you stop at the if-statement solution. Recognizing the period-15 structure is the same skill as noticing that a database query repeats the same joins, seeing that an algorithm recomputes the same values, or identifying that user behavior follows daily cycles.

The traditional solution works perfectly well for FizzBuzz. Using tensors and FFT is massive overkill for printing numbers. But the exercise of exploring unusual approaches teaches more than stopping at "good enough." You learn about NumPy broadcasting, Fourier analysis, tensor operations, and pattern visualization. You see how algorithmic and mathematical perspectives offer different insights into the same problem.

When evaluating developers, look for people who ask why a pattern exists rather than just implementing the pattern. Someone who looks at FizzBuzz and thinks "this is a periodic signal" is someone who will find optimization opportunities and architectural improvements in your codebase. The ability to see problems from multiple angles separates competent solutions from elegant ones.

The Technical Details

For the full mathematical treatment, I wrote a complete paper with formal proofs of periodicity and complexity, Fourier analysis, trigonometric representations, and generalization to arbitrary divisor sets.

Full paper: TensorFizzBuzz: A Signal Processing Approach

Code repository: https://github.com/aaronsb/fizzbuzztensor

All code is open source with visualization tools included.

Your Turn

What's a problem you've solved where changing the representation completely changed the solution? What's something in your field that looks complex but turns out to be a pattern repeating? What interview question should I over-engineer next?

Stats:

- Pattern period: 15
- Space complexity: O(15) = O(1)
- Time per lookup: O(1)
- Fun had: Immeasurable

Links:

- GitHub: https://github.com/aaronsb/fizzbuzztensor
- Inspiration: https://susam.net/fizz-buzz-with-cosines.html
- Full paper: tensor-fizzbuzz-paper.md

"The most elemental solution to FizzBuzz is not an algorithm, but a number: 15."

#SoftwareEngineering #DataScience #MachineLearning #AI #ArtificialIntelligence #Programming #Python #TensorFlow #NumPy #SignalProcessing #TechInnovation #DeveloperLife #CodingLife #SoftwareDevelopment #TechCareers #HiringDevelopers #TechRecruiting #LearnToCode #DataVisualization #ComputerScience #AlgorithmDesign #TechEducation #DevCommunity #EngineeringExcellence #ProblemSolving #ThinkDifferent #TechTrends #FutureOfWork #Innovation #STEM #TechLeadership

---

P.S. Yes, I know this is complete overkill for FizzBuzz. That was the point.