Digit Occurrence of an Ordinal's Factorial in Euler's Number

Parker Emmerson

July 2024

Abstract: We analyze the frequency of digit occurrences in that digit's factorial expression in different bases. I write programs for visualizing it.

1 Introduction

Mathematical Notation of the Program

1. Factorial Calculation

The factorial of a number n is calculated as:

$$n! = \prod_{i=1}^{n} i$$

For example, for $n = 5$:

$$5! = 5 \times 4 \times 3 \times 2 \times 1 = 120$$

2. Conversion to a Given Base

To convert a number x to its representation in a specified base b:

$$x = \sum_{i=0}^{k} d_i \cdot b^i$$

where d_i are the digits of x in base b. The digits are calculated as:

$$d_i = x \mod b, \quad x = \left\lfloor \frac{x}{b} \right\rfloor$$

3. Counting Digit Occurrences

To count the occurrences of a specific digit d in the base-b representation of $n!$:

$$n! = \sum_{i=0}^{k} d_{i,b} \cdot b^i$$

$$\text{count}(d, n!, b) = \sum_{i=0}^{k} \delta(d, d_{i,b})$$

where δ is the Kronecker delta function:

$$\delta(a, b) = \begin{cases} 1 & \text{if } a = b, \\ 0 & \text{otherwise.} \end{cases}$$

4. Main Program Loop

For each i from 1 to 99, the program performs the following steps:

1. Compute the factorial $i!$. 2. Convert $i!$ to base 100. 3. Count the occurrences of the digit i in this base-100 representation.

Formally, for $i \in \{1, 2, \ldots, 99\}$, the program computes:

$$\text{result}(i) = \text{count}(i, i!, 100)$$

5. Relation to Euler's Number

Euler's number e is defined as:

$$e = \sum_{n=0}^{\infty} \frac{1}{n!}$$

This definition shows that e can be represented as an infinite series involving reciprocals of factorials. By analyzing the digit occurrences in factorials, we study deeper properties of these numbers which are fundamental to the series representation of Euler's number.

Mathematical Notation of the Program with Radix e

1. Factorial Calculation

The factorial of a number n is calculated as:

$$n! = \prod_{i=1}^{n} i$$

2. Conversion to Base e (Approximation)

To convert a number x to its representation in base e:

$$x = \sum_{i=0}^{\infty} d_i \cdot e^i$$

where d_i are the coefficients. Given the irrational nature of e, these coefficients are approximated as:

$$d_i = \left\lfloor \frac{x}{e^i} \right\rfloor, \quad x = x - d_i \cdot e^i$$

3. Counting Approximate Digit Occurrences

To count the occurrences of integer parts of coefficients in the approximate base-e representation of $n!$:

$$\text{count}_e(d, n!) = \sum_{i=0}^{\infty} \delta \left(\lfloor d_i \rfloor, d \right)$$

4. Main Program Loop (Approximation)

For each i from 1 to 99, the program performs the following steps:

1. Compute the factorial $i!$. 2. Convert $i!$ to approximate base e. 3. Count the occurrences of the approximate digit i.

Formally, for $i \in \{1, 2, \ldots, 99\}$, the program computes:

$$\text{result}(i) = \text{count}_e(i, i!)$$

Understanding Euler's Number e in Its Own Base and Factorial Digit Frequencies

1. Representation of e in Its Own Base

To express Euler's number e in its own base e:

$$e = \sum_{i=-\infty}^{\infty} d_i \cdot e^i$$

where d_i are the coefficients in this unique base e representation.

2. Digit Frequencies in Factorials

Define the frequency of digit d in the base-e representation of $n!$:

$$F_d(n) = \frac{\text{count of digit } d \text{ in } n! \text{ (base } e\text{)}}{\text{total digits in } n! \text{ (base } e\text{)}}$$

For example, if the digit d appears $c_d(n)$ times in the base-e representation of $n!$, then:

$$F_d(n) = \frac{c_d(n)}{\sum_{d'} c_{d'}(n)}$$

3. Series Representation of Euler's Number e

Euler's number e can be represented by the series:

$$e = \sum_{n=0}^{\infty} \frac{1}{n!}$$

Considering the contribution of digit frequencies in factorials, we propose a weighted sum representation:

$$e = \sum_{n=0}^{\infty} \left(\frac{1}{n!} \cdot \sum_{d} d \cdot F_d(n) \right)$$

where the inner sum accounts for the influence of digit d frequencies $F_d(n)$.

2 Python Program

Below is the Python program that calculates the occurrences of a digit in the factorial of a number, represented in a specified base.

Listing 1: Python code for factorial digit occurrence

```
from collections import Counter

def factorial(n):
    if n == 0 or n == 1:
        return 1
    fact = 1
    for i in range(2, n + 1):
        fact *= i
    return fact

def convert_to_base(n, base):
    if n == 0:
        return [0]
    digits = []
    while n:
        digits.append(n % base)
        n = n // base
    return digits[::-1]  # Reverse to get the correct order

def count_digit_occurrences(n, base):
    fact_n = factorial(n)
    digits = convert_to_base(fact_n, base)
    counter = Counter(digits)
    return counter
```

```
# Example usage:
base = 100
for i in range(1, 100):
    occurrence = count_digit_occurrences(i, base)
    subscript_digit = i
    result = occurrence.get(subscript_digit, 0)
    print(f"The digit {subscript_digit} appears {result} times
    in the factorial {i}! in base {base}")
```

3 Python Program for Interactive Visualization

This program provides an interactive visualization of the occurrences of digits in the factorials of numbers, represented in a specified base.

Explanation of the Program

- **Factorial Calculation**: The `factorial` function computes the factorial of an integer n using an iterative approach.
- **Base Conversion**: The `convert_to_base` function converts a number n to its representation in a given base and returns a list of digits.
- **Digit Counting**: The `count_digit_occurrences` function calculates the factorial of n, converts itto the given base, and counts the occurrences of each digit using a 'Counter'.
- **Occurrences Computation**: The `compute_occurrences` function calculates digit occurrences for numbers from 1 to `max_range` and stores these in a matrix.
- **3D Plotting**: The `plot_occurrences` function uses Matplotlib's 3D plotting capabilities to visualize the digit occurrences in a 3D bar chart.
- **Interactive Visualization**: The `interactive_plot` function uses `ipywidgets` to create an interactive plot, allowing users to adjust `max_range` and `base` using sliders.

Python Code

Listing 2: Python code for interactive visualization of digit occurrences in factorials

```
from ipywidgets import interact, IntSlider
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from collections import Counter
```

```
def factorial(n):
    if n == 0 or n == 1:
        return 1
    fact = 1
    for i in range(2, n + 1):
        fact *= i
    return fact

def convert_to_base(n, base):
    if n == 0:
        return [0]
    digits = []
    while n:
        digits.append(n % base)
        n = n // base
    return digits[::-1]

def count_digit_occurrences(n, base):
    fact_n = factorial(n)
    digits = convert_to_base(fact_n, base)
    counter = Counter(digits)
    return counter

def compute_occurrences(max_range, base):
    occurrences_matrix = np.zeros((max_range, base))
    for i in range(1, max_range + 1):
        occurrence = count_digit_occurrences(i, base)
        for digit in range(base):
            occurrences_matrix[i - 1, digit] = occurrence.get(digit, 0)
    return occurrences_matrix

def plot_occurrences(max_range, base):
    occurrences_matrix = compute_occurrences(max_range, base)
    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')
    x_data, y_data = np.meshgrid(range(1, max_range + 1), range(base))
    x_data = x_data.flatten()
    y_data = y_data.flatten()
    z_data = occurrences_matrix.flatten()

    ax.bar3d(x_data, y_data, np.zeros_like(z_data), 1, 1, z_data, shade=True)

    ax.set_xlabel('Number (n)')
    ax.set_ylabel(f'Base-{base} Digit')
    ax.set_zlabel('Occurrences')
    plt.show()

def interactive_plot():
    interact(
        plot_occurrences,
        max_range=IntSlider(min=2, max=500, step=1, value=10, description=
        'Max-Range'),
        base=IntSlider(min=2, max=500, step=1, value=10, description='Base')
    )

# Run the interactive plot
interactive_plot()
```

Mathematical Notation of the Program

4 Python Program for Prime Factorial Digit Occurrence Visualization

This program provides an interactive visualization of the occurrences of digits in the factorials of prime numbers, represented in a specified base.

Explanation of the Program

- **Factorial Calculation**: The `factorial` function computes the factorial of an integer n using an iterative approach.
- **Base Conversion**: The `convert_to_base` function converts a number n to its representation in a given base and returns a list of digits.
- **Digit Counting**: The `count_digit_occurrences` function calculates the factorial of n, converts it to the given base, and counts the occurrences of each digit using a `Counter`.
- **Prime Occurrences Computation**: The `compute_prime_occurrences` function calculates digit occurrences for the factorials of prime numbers up to a specified index and stores these occurrences in a matrix.
- **3D Plotting**: The `plot_prime_occurrences` function uses Matplotlib's 3D plotting capabilities to visualize the digit occurrences in a 3D bar chart.
- **Interactive Visualization**: The `interactive_prime_plot` function uses `ipywidgets` to create an interactive plot, allowing users to adjust `max_prime_idx` and `base` using sliders.

Python Code

Listing 3: Python code for interactive visualization of digit occurrences in prime factorials

```
import sympy as sp
from ipywidgets import interact, IntSlider
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from collections import Counter

def factorial(n):
    if n == 0 or n == 1:
        return 1
    fact = 1
    for i in range(2, n + 1):
        fact *= i
    return fact

def convert_to_base(n, base):
    if n == 0:
        return [0]
    digits = []
    while n:
        digits.append(n % base)
        n = n // base
    return digits[::-1]

def count_digit_occurrences(n, base):
    fact_n = factorial(n)
    digits = convert_to_base(fact_n, base)
    counter = Counter(digits)
    return counter

def compute_prime_occurrences(max_prime_idx, base):
    primes = list(sp.primerange(1, max_prime_idx + 1))
    occurrences_matrix = np.zeros((len(primes), base))
    for idx, prime in enumerate(primes):
        occurrence = count_digit_occurrences(prime, base)
        for digit in range(base):
            occurrences_matrix[idx, digit] = occurrence.get(digit, 0)
    return occurrences_matrix, primes

def plot_prime_occurrences(max_prime_idx, base):
    occurrences_matrix, primes = compute_prime_occurrences(max_prime_idx, base)
    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')
    x_data, y_data = np.meshgrid(range(0, len(primes)), range(base))
    x_data = x_data.flatten()
    y_data = y_data.flatten()
    z_data = occurrences_matrix.T.flatten()

    ax.bar3d(x_data, y_data, np.zeros_like(z_data), 1, 1, z_data, shade=True)

    ax.set_xlabel('Prime Index')
    ax.set_ylabel(f'Base-{base} Digit')
    ax.set_zlabel('Occurrences')

    plt.show()

def interactive_prime_plot():
    interact(
        plot_prime_occurrences,
        max_prime_idx=IntSlider(min=2, max=1000, step=1, value=50,
        description='Max Prime Index'),

        base=IntSlider(min=2, max=500, step=1, value=10, description='Base')
    )

# Run the interactive prime plot
interactive_prime_plot()
```

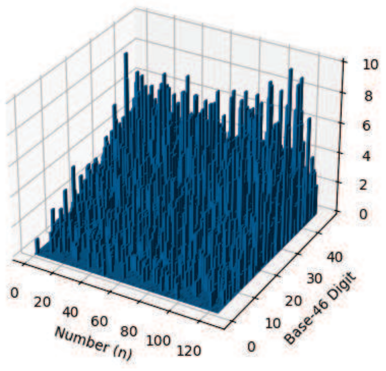

Figure 1: Image 1

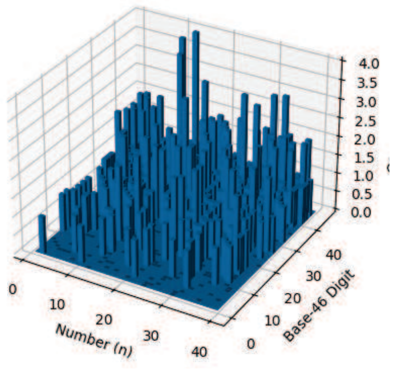

Figure 2: Image 2

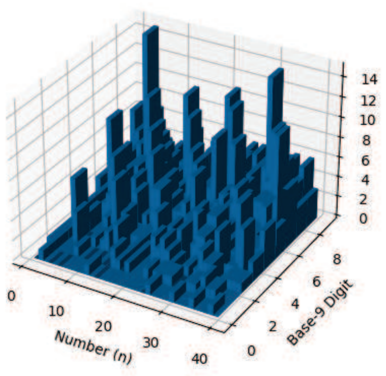

Figure 3: Image 3

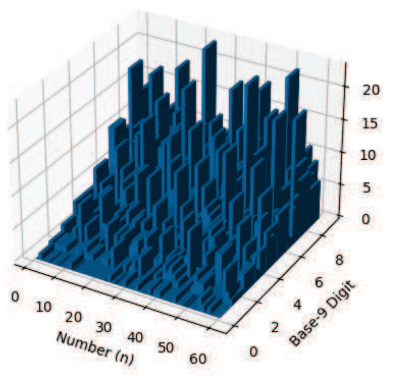

Figure 4: Image 4

Figure 5: Examples of Digit Occurences Data in Factorial Expressions of that Number.