Wave of Ratio of Change for Consecutive Decimal Values of

April 26, 2025

[1]:

```
import plotly.graph_objects as go
from mpmath import mp

# Set precision for
mp.dps = 50000  # 100 decimal places of

# Extract the first 100 decimal places of   as integers
def get_pi_decimals():
    pi_str = str(mp.pi)[2:]  # Get   as a string and exclude "3."
    return [int(digit) for digit in pi_str[:100]]  # Convert first 100 decimals␣
 ↪to integers

# Compute the ratio of change between consecutive digits
def compute_ratio_of_change(digits):
    changes = [abs(digits[i] - digits[i - 1]) for i in range(1, len(digits))]
    ratios = [changes[i] / changes[i - 1] if changes[i - 1] != 0 else 0 for i␣
 ↪in range(1, len(changes))]
    return ratios

# Get   decimals and compute ratios
decimals = get_pi_decimals()
ratios = compute_ratio_of_change(decimals)

# Plot using Plotly
fig = go.Figure()
fig.add_trace(go.Scatter(
    y=ratios,
    mode='lines+markers',
    name='Ratio of Change'
))

# Add labels and title
fig.update_layout(
    title='Wave of Ratio of Change for Consecutive Decimal Values of  ',
    xaxis_title='Index',
    yaxis_title='Ratio of Change',
    template='plotly_dark'
```

```
)

# Show the plot
fig.show()
```

[2]:
```
import plotly.graph_objects as go
from mpmath import mp

# Set precision for
mp.dps = 1000  # Adjust to 1000 decimal places of   or more

# Extract the first N decimal places of   as integers
def get_pi_decimals(n):
    pi_str = str(mp.pi)[2:]  # Get   as a string and exclude "3."
    return [int(digit) for digit in pi_str[:n]]  # Convert first N decimals to␣
 ↪integers

# Compute the ratio of change between consecutive digits
def compute_ratio_of_change(digits):
    changes = [abs(digits[i] - digits[i - 1]) for i in range(1, len(digits))]
    ratios = [changes[i] / changes[i - 1] if changes[i - 1] != 0 else 0 for i␣
 ↪in range(1, len(changes))]
    return ratios

# Get   decimals and compute ratios
num_decimals = 1000  # Define the number of decimals to process
decimals = get_pi_decimals(num_decimals)
ratios = compute_ratio_of_change(decimals)

# Plot using Plotly
fig = go.Figure()
fig.add_trace(go.Scatter(
    x=list(range(len(ratios))),  # Use indices for the x-axis
    y=ratios,
    mode='lines+markers',
    name='Ratio of Change'
))

# Add labels and title
fig.update_layout(
    title=f'Wave of Ratio of Change for First {num_decimals} Decimal Values of␣
 ↪ ',
    xaxis_title='Index',
    yaxis_title='Ratio of Change',
    template='plotly_white'
)
```

```
# Show the plot
fig.show()
```

[3]:

```
import plotly.graph_objects as go
from mpmath import mp

# Step 1: Generate 1,000,000 Decimal Places of
mp.dps = 1_000_000  # Set precision to 1,000,000
pi_str = str(mp.pi)[2:]  # Get   as a string and exclude "3."

# Convert to integers
def get_pi_decimals(precision):
    return [int(digit) for digit in pi_str[:precision]]

# Compute the ratio of change between consecutive digits
def compute_ratio_of_change(digits):
    changes = [abs(digits[i] - digits[i - 1]) for i in range(1, len(digits))]
    ratios = [changes[i] / changes[i - 1] if changes[i - 1] != 0 else 0 for i␣
 ↪in range(1, len(changes))]
    return ratios

# Step 2: Apply Nexus 2 Framework Harmonization
def harmonize_data(data, harmonic_constant=0.35):
    return [(value - harmonic_constant) ** 2 for value in data]

# Generate data and compute ratios
decimals = get_pi_decimals(1_000_000)
ratios = compute_ratio_of_change(decimals)

# Harmonize the ratios
harmonized_ratios = harmonize_data(ratios)

# Step 3: Visualization
fig = go.Figure()

# Plot original ratios
fig.add_trace(go.Scatter(
    x=list(range(len(ratios[:10_000]))),  # Limit visualization to 10,000␣
 ↪points for clarity
    y=ratios[:10_000],
    mode='lines',
    name='Original Ratios of Change'
))

# Plot harmonized ratios
fig.add_trace(go.Scatter(
```

```
    x=list(range(len(harmonized_ratios[:10_000]))),  # Limit visualization to␣
↪10,000 points
    y=harmonized_ratios[:10_000],
    mode='lines',
    name='Harmonized Ratios of Change (H=0.35)'
))

# Update layout
fig.update_layout(
    title="Harmonized vs. Original Ratios of Change (First 10,000 Points)",
    xaxis_title="Index",
    yaxis_title="Ratio Value",
    template="plotly_dark",
    showlegend=True
)

# Show the plot
fig.show()
```

[4]:

```
import plotly.graph_objects as go
from mpmath import mp

# Step 1: Generate 1,000,000 Decimal Places of
mp.dps = 100  # Set precision to 1,000,000
pi_str = str(mp.pi)[2:]  # Get   as a string and exclude "3."

# Convert to integers
def get_pi_decimals(precision):
    return [int(digit) for digit in pi_str[:precision]]

# Compute the ratio of change between consecutive digits
def compute_ratio_of_change(digits):
    changes = [abs(digits[i] - digits[i - 1]) for i in range(1, len(digits))]
    ratios = [changes[i] / changes[i - 1] if changes[i - 1] != 0 else 0 for i␣
↪in range(1, len(changes))]
    return ratios

# Step 2: Apply Nexus 2 Framework Harmonization
def harmonize_data(data, harmonic_constant=0.35):
    return [(value - harmonic_constant) ** 2 for value in data]

# Generate data and compute ratios
decimals = get_pi_decimals(1_000_000)
ratios = compute_ratio_of_change(decimals)

# Harmonize the ratios
harmonized_ratios = harmonize_data(ratios)
```

```
# Visualization: Original Ratios
fig_original = go.Figure()
fig_original.add_trace(go.Scatter(
    x=list(range(len(ratios[:10_000]))),  # Limit visualization to 10,000␣
↪points for clarity
    y=ratios[:10_000],
    mode='lines',
    name='Original Ratios of Change'
))
fig_original.update_layout(
    title="Original Ratios of Change (First 10,000 Points)",
    xaxis_title="Index",
    yaxis_title="Ratio Value",
    template="plotly_white"
)

# Visualization: Harmonized Ratios
fig_harmonized = go.Figure()
fig_harmonized.add_trace(go.Scatter(
    x=list(range(len(harmonized_ratios[:10_000]))),  # Limit visualization to␣
↪10,000 points
    y=harmonized_ratios[:10_000],
    mode='lines',
    name='Harmonized Ratios of Change (H=0.35)'
))
fig_harmonized.update_layout(
    title="Harmonized Ratios of Change (First 10,000 Points, H=0.35)",
    xaxis_title="Index",
    yaxis_title="Harmonized Ratio Value",
    template="plotly_white"
)

# Show the plots
fig_original.show()
fig_harmonized.show()
```

[5]:
```
import numpy as np
from mpmath import mp
import plotly.graph_objects as go

# Set precision for
mp.dps = 1000  # Set precision to 1000 decimal places

# Extract the first N decimal places of  as integers
def get_pi_decimals(num_decimals):
    pi_str = str(mp.pi)[2:]  # Get  as a string and exclude "3."
```

```
    return [int(digit) for digit in pi_str[:num_decimals]]  # Convert first num_decimals to integers

# Compute the ratio of change between consecutive digits
def compute_ratio_of_change(digits):
    changes = [abs(digits[i] - digits[i - 1]) for i in range(1, len(digits))]
    ratios = [changes[i] / changes[i - 1] if changes[i - 1] != 0 else 0 for i in range(1, len(changes))]
    return ratios

# Apply harmonic stabilization to the ratios
def harmonize_ratios(ratios, harmonic_constant):
    return [ratio / (1 + harmonic_constant * abs(ratio - harmonic_constant)) for ratio in ratios]

# Number of   decimals to process
num_decimals = 1000

# Get   decimals and compute ratios
decimals = get_pi_decimals(num_decimals)
ratios = compute_ratio_of_change(decimals)

# Apply harmonic stabilization
harmonic_constant = 0.35
harmonized_ratios = harmonize_ratios(ratios, harmonic_constant)

# Convert harmonized values to a long string
harmonized_values_string = ''.join(f'{value:.8f}' for value in harmonized_ratios)

# Print the harmonized values as a single long string
print("Harmonized Values as String:")
print(harmonized_values_string)

# Plot original and harmonized ratios
fig = go.Figure()

# Original ratios
fig.add_trace(go.Scatter(
    y=ratios[:100],  # Plot first 100 points for visualization
    mode='lines',
    name='Original Ratios'
))

# Harmonized ratios
fig.add_trace(go.Scatter(
    y=harmonized_ratios[:100],  # Plot first 100 points for visualization
```

```
    mode='lines',
    name='Harmonized Ratios'
))

# Add labels and title
fig.update_layout(
    title='Comparison of Original and Harmonized Ratios of Change',
    xaxis_title='Index',
    yaxis_title='Ratio Value',
    template='plotly_white'
)

# Show the plot
fig.show()
```

```
Harmonized Values as String:
0.814663950.991940480.814663951.174496640.530328140.241545891.267828840.81466395
1.069518720.331400171.267828840.814663951.556420230.156617070.814663951.90294957
0.691144710.475059381.267828840.814663950.475059380.475059380.000000000.00000000
0.814663950.190023751.902949570.393120391.267828840.950570340.393120391.55642023
0.000000000.000000000.657894741.472618500.241545891.556420230.712758370.55172414
1.267828840.814663950.000000000.000000000.600150040.475059381.267828840.24154589
1.902949570.551724141.267828840.331400171.834862390.212709391.069518721.14090131
0.814663950.000000000.000000001.756311751.174496640.133200131.556420231.37727496
0.133200132.103681440.633512830.393120391.267828841.069518720.600150041.06951872
0.331400171.267828840.475059381.756311750.115874860.000000000.000000001.26782884
1.267828841.069518720.991940480.371747210.331400171.756311751.069518720.47505938
0.600150040.475059381.267828840.475059380.000000000.000000000.907323400.72793449
0.156617071.267828840.475059382.015113350.156617071.556420230.991940481.26782884
0.814663950.241545890.475059381.756311750.475059380.475059382.015113350.81466395
0.156617071.556420231.267828840.000000000.000000001.069518721.377274960.96852300
0.600150040.712758370.691144710.000000000.000000001.556420231.069518720.43022318
0.000000000.000000000.814663950.551724141.267828840.000000000.000000001.26782884
1.556420230.712758370.551724140.600150040.814663951.267828840.950570340.69114471
1.267828840.739176350.133200132.015113350.600150040.814663951.174496640.00000000
0.000000000.000000000.475059380.331400171.902949570.393120391.556420230.60015004
0.657894740.331400171.267828841.426533521.023766000.133200132.175390890.65789474
0.712758370.551724140.814663950.331400170.000000000.000000001.902949570.00000000
0.000000000.000000000.657894740.600150040.000000000.000000001.267828840.00000000
0.000000000.633512830.691144710.241545891.756311750.241545891.902949570.92485549
0.475059380.814663951.140901311.023766000.894604420.371747210.600150040.00000000
0.000000001.556420230.000000000.000000000.133200132.234636870.212709390.81466395
0.475059380.000000000.000000001.756311751.069518720.000000000.000000000.00000000
0.000000001.426533520.190023752.015113350.600150040.657894740.600150040.47505938
1.267828841.267828841.069518720.156617070.000000000.000000000.241545891.26782884
0.475059380.814663951.902949570.393120391.426533520.190023751.556420231.14090131
0.924855490.156617071.267828840.475059382.175390890.884955750.814663950.74783828
0.371747210.331400170.814663951.267828841.267828840.657894740.331400170.00000000
```

0.000000001.377274960.133200130.814663951.267828841.556420230.475059380.33140017
1.756311750.475059381.426533520.190023751.756311750.241545890.814663950.81466395
0.814663951.756311750.000000000.000000001.267828841.069518720.156617071.26782884
0.000000000.000000000.814663950.475059381.267828840.907323400.633512830.69114471
1.069518720.331400170.814663951.426533521.113043480.371747210.814663950.33140017
1.902949570.691144710.814663950.950570340.393120390.475059381.556420230.33140017
2.103681440.000000000.000000000.000000000.000000000.814663950.475059380.60015004
1.267828840.000000000.000000000.000000000.000000000.727934490.475059380.99194048
0.000000000.000000000.530328140.657894740.600150041.834862390.676491910.27942717
1.834862390.331400170.991940481.069518720.814663950.907323400.814663950.41710115
0.331400171.756311751.351351350.747838280.657894740.814663950.600150040.47505938
1.069518720.600150040.475059381.556420230.331400170.814663950.814663952.10368144
0.417101150.991940481.351351350.331400170.814663951.267828840.000000000.00000000
0.000000000.000000000.000000000.000000001.140901310.393120391.069518721.14090131
0.190023751.756311750.000000000.000000000.331400171.267828840.475059380.00000000
0.000000001.556420230.331400171.267828841.426533520.691144710.657894740.81466395
0.600150041.069518720.991940480.814663951.267828840.570125430.551724140.99194048
0.814663950.000000000.000000000.924855491.069518720.518268980.190023750.00000000
0.000000001.140901310.393120391.426533520.814663951.023766000.417101150.81466395
0.331400171.267828840.814663951.267828840.814663950.814663951.267828840.81466395
0.475059380.475059381.069518721.556420230.676491910.133200132.103681440.27942717
1.426533520.000000000.000000000.600150040.950570341.023766000.894604420.65789474
0.156617071.756311750.950570340.000000000.000000000.331400171.556420230.33140017
0.475059381.902949570.691144710.000000000.000000000.417101150.331400171.75631175
1.267828840.739176350.417101150.000000000.000000001.267828840.241545891.75631175
0.475059380.000000000.000000000.991940480.950570340.551724141.140901310.69114471
0.241545890.814663951.756311750.475059381.267828841.174496640.000000000.00000000
0.600150040.814663951.069518721.140901310.814663950.393120391.267828840.24154589
2.175390890.115874860.000000000.000000000.393120391.556420230.712758371.02376600
0.814663950.633512830.551724140.331400170.000000000.000000000.570125430.81466395
1.113043480.115874862.103681440.133200131.902949570.000000000.000000000.33140017
1.756311750.000000000.000000000.991940480.475059380.000000000.000000001.06951872
0.156617070.814663950.000000000.000000000.475059381.556420231.472618500.24154589
1.556420230.331400170.475059381.267828841.556420230.712758370.814663950.81466395
0.393120391.069518721.267828840.600150040.657894740.000000000.000000001.06951872
0.991940480.814663951.069518720.991940480.884955750.676491910.814663950.27942717
0.475059382.015113350.331400170.475059381.267828841.556420231.069518720.51826898
0.190023751.756311751.174496640.279427171.426533520.000000000.000000000.63351283
0.393120391.556420230.907323400.133200132.015113350.814663950.814663950.15661707
1.756311751.174496640.727934490.331400171.556420230.156617070.814663951.26782884
1.069518720.331400171.902949570.691144710.475059380.475059381.556420230.99194048
1.174496640.814663950.530328140.475059380.475059380.814663950.000000000.00000000
1.140901310.691144710.950570340.691144710.475059380.475059381.267828840.00000000
0.000000000.000000000.190023751.267828841.664684900.133200131.902949570.92485549
0.475059380.331400171.556420230.991940480.657894740.600150040.475059382.01511335
0.991940480.657894740.712758370.000000000.000000001.556420230.600150040.00000000
0.000000000.331400170.475059381.267828841.426533520.393120390.814663950.00000000
0.000000000.814663951.556420231.267828841.069518720.747838280.657894740.60015004

0.657894740.814663950.991940481.069518720.814663950.156617070.814663951.90294957
0.190023751.556420230.600150040.814663951.267828840.000000000.000000001.35135135
0.814663950.102537810.814663950.000000000.000000001.426533520.691144710.47505938
0.475059380.814663951.556420230.331400171.556420230.600150040.475059380.81466395
1.902949570.691144710.657894740.814663950.600150041.267828841.069518720.15661707
1.902949570.814663951.023766000.279427171.664684900.000000000.000000000.39312039
1.069518720.600150040.475059382.103681440.417101150.814663950.331400172.10368144
0.133200131.267828840.475059381.267828840.814663950.475059382.175390890.00000000
0.000000000.241545891.902949570.000000000.000000000.814663950.814663952.10368144
0.968523000.212709390.475059382.175390890.371747211.267828840.991940480.24154589
0.814663951.267828840.657894740.331400170.000000000.000000001.377274960.81466395
0.530328140.814663950.241545892.103681440.279427171.069518720.991940481.17449664
0.279427170.000000000.000000000.814663950.000000000.000000000.331400171.06951872
1.556420230.000000000.000000001.267828840.712758370.691144711.174496640.13320013
2.103681440.814663950.633512830.190023750.000000000.000000000.475059381.90294957
0.000000000.000000000.000000000.000000000.000000000.000000001.902949570.69114471
0.950570341.023766000.279427170.475059382.175390890.475059380.950570340.00000000
0.000000000.814663950.241545891.902949570.691144710.475059381.267828840.47505938
1.556420230.600150040.241545892.015113350.907323400.633512830.924855491.06951872
0.331400170.814663950.600150041.664684900.417101150.991940480.814663950.95057034
0.393120390.814663950.000000000.000000001.756311750.950570340.190023750.00000000
0.000000000.475059381.267828841.069518721.556420230.747838280.570125430.55172414
0.600150041.267828840.475059380.814663951.069518720.814663950.000000000.00000000
1.556420231.472618500.657894740.475059380.600150040.000000000.000000000.00000000
0.000000000.814663951.069518721.140901310.551724140.600150041.069518720.99194048
0.950570341.113043480.657894740.331400170.000000000.000000000.000000000.00000000
0.727934490.814663950.156617070.814663950.814663950.000000000.000000000.00000000
0.600150040.814663951.267828840.241545891.902949570.814663950.190023751.26782884
1.069518721.267828840.000000000.000000000.475059381.556420230.331400171.26782884
0.814663950.000000000.000000001.267828841.756311750.570125430.814663951.02376600
0.417101150.600150040.814663951.556420230.712758370.924855490.814663950.81466395
0.000000000.000000000.000000000.000000001.472618500.371747210.814663950.99194048
0.657894740.814663950.600150041.267828840.241545892.015113350.475059380.60015004
0.475059381.902949571.194029850.430223180.475059381.556420230.156617071.26782884
0.000000000.000000001.267828840.814663950.475059381.756311750.475059380.00000000
0.000000000.814663950.814663950.241545891.756311751.069518720.331400171.06951872
1.140901310.000000000.000000000.156617071.267828840.814663951.267828840.24154589
0.814663951.267828841.267828841.069518720.600150040.475059381.267828841.26782884
0.475059380.475059380.000000000.000000002.103681440.814663950.417101150.60015004
0.000000000.000000002.175390890.570125430.393120390.475059380.814663952.01511335
0.814663950.156617070.000000000.000000000.475059381.756311750.657894740.00000000
0.000000000.393120391.069518720.000000000.000000002.175390890.739176350.63351283
0.190023750.814663950.814663950.000000000.000000000.000000000.000000000.00000000
0.475059380.814663951.351351350.814663950.676491910.133200131.902949570.39312039
0.814663950.814663950.475059382.175390890.00000000

[6]:
```
import plotly.graph_objects as go
from mpmath import mp

# Set precision for
mp.dps = 10000  # Adjust the precision as needed

# Extract the first N decimal places of
def get_pi_decimals(num_decimals):
    pi_str = str(mp.pi)[2:]  # Exclude "3."
    return [int(digit) for digit in pi_str[:num_decimals]]

# Perform arithmetic operations on  digits
def calculate_arithmetic_patterns(digits):
    pattern = []
    for i in range(1, len(digits) - 1, 2):  # Step by 2 to alternate operations
        sum_val = digits[i] + digits[i + 1]
        diff_val = digits[i + 1] - digits[i]
        pattern.append(sum_val)
        pattern.append(diff_val)
    return pattern

# Get  decimals and compute patterns
num_decimals = 1000  # Adjust the number of decimals
decimals = get_pi_decimals(num_decimals)
arithmetic_pattern = calculate_arithmetic_patterns(decimals)

# Plot the results
fig = go.Figure()
fig.add_trace(go.Scatter(
    y=arithmetic_pattern[:100],  # Plot the first 100 points
    mode='lines+markers',
    name='Arithmetic Pattern'
))

# Add labels and title
fig.update_layout(
    title='Arithmetic Patterns in Digits of  ',
    xaxis_title='Index',
    yaxis_title='Pattern Value',
    template='plotly_white'
)

# Show the plot
fig.show()
```

[7]:
```
import plotly.graph_objects as go
from mpmath import mp
```

```python
# Set precision for
mp.dps = 2000  # Set precision for sufficient range

# Extract the first N decimal places of  as integers
def get_pi_decimals(N):
    pi_str = str(mp.pi)[2:]  # Get  as a string and exclude "3."
    return [int(digit) for digit in pi_str[:N]]  # Convert first N decimals to integers

# Compute arithmetic patterns based on differences
def compute_arithmetic_patterns(digits):
    patterns = []
    for i in range(1, len(digits)):
        pattern = digits[i] - digits[i - 1]
        patterns.append(pattern)
    return patterns

# Compute forward and reverse patterns
N = 1000
decimals = get_pi_decimals(N)

# Forward pattern
forward_patterns = compute_arithmetic_patterns(decimals)

# Reverse pattern (from the end of the N range going backward)
reverse_patterns = compute_arithmetic_patterns(decimals[::-1])

# Plot using Plotly
fig = go.Figure()

# Forward patterns
fig.add_trace(go.Scatter(
    y=forward_patterns,
    mode='lines',
    name='Forward Arithmetic Patterns'
))

# Reverse patterns
fig.add_trace(go.Scatter(
    y=reverse_patterns,
    mode='lines',
    name='Reverse Arithmetic Patterns'
))

# Add labels and title
fig.update_layout(
```

```
    title='Overlay of Forward and Reverse Arithmetic Patterns in  ',
    xaxis_title='Index',
    yaxis_title='Arithmetic Pattern Value',
    template='plotly_white'
)

# Show the plot
fig.show()
```

[8]:

```
import plotly.graph_objects as go
from mpmath import mp

# Set precision for
mp.dps = 100  # 100 decimal places of

# Extract the first 100 decimal places of   as integers
def get_pi_decimals():
    pi_str = str(mp.pi)[2:]  # Get   as a string and exclude "3."
    return [int(digit) for digit in pi_str[:100]]  # Convert first 100 decimals␣
  ↪to integers

# Compute the ratio of change between consecutive digits
def compute_ratio_of_change(digits):
    changes = [abs(digits[i] - digits[i - 1]) for i in range(1, len(digits))]
    ratios = [changes[i] / changes[i - 1] if changes[i - 1] != 0 else 0 for i␣
  ↪in range(1, len(changes))]
    return ratios

# Get   decimals and compute ratios
decimals = get_pi_decimals()
ratios = compute_ratio_of_change(decimals)

# Plot using Plotly
fig = go.Figure()
fig.add_trace(go.Scatter(
    y=ratios,
    mode='lines+markers',
    name='Ratio of Change'
))

# Add labels and title
fig.update_layout(
    title='Wave of Ratio of Change for Consecutive Decimal Values of  ',
    xaxis_title='Index',
    yaxis_title='Ratio of Change',
    template='plotly_white'
)
```

```python
# Show the plot
fig.show()
```

```
[ ]:
```