CMSN process

The provided code implements a mathematical analysis of trajectories in the Collatz-Matthews process. Below is its mathematical abstraction:

---

1. Collatz-Matthews Process

For a starting integer n, iteratively apply the transformation:

$$a_{k+1} = \begin{cases} \frac{a_k}{2} & \text{if } a_k \text{ is even} \\ 3a_k + 1 & \text{if } a_k \text{ is odd} \end{cases}$$

until $a_k = 1$. During this process, track the following metrics for n:

- $b_x(n)$: Total steps (even + odd).
- $b_z(n)$: Count of odd steps.
- $b_y(n)$: Cumulative sum of $\log_2(a_k)$ over odd steps.
- $G(n)$: Cumulative sum of $\log_2\left(3 + \frac{1}{a_k}\right)$ over odd steps.
- $\text{max\_a}(n)$: Maximum a_k encountered.

2. Metric Definitions

For a trajectory starting at n:

$$\begin{aligned} b_x(n) &= \text{Total iterations until } a_k = 1, \\ b_z(n) &= \sum_{k \text{ odd step}} 1, \\ b_y(n) &= \sum_{k \text{ odd step}} \log_2(a_k), \\ G(n) &= \sum_{k \text{ odd step}} \log_2\left(3 + \frac{1}{a_k}\right), \\ \text{max\_a}(n) &= \max\{a_1, a_2, \dots, a_{b_x(n)}\}. \end{aligned}$$

3. Derived Ratios and Conditions

For analysis, define:

$$
\begin{aligned}
\text{bz\_bx\_ratio}(n) &= \frac{b_z(n)}{b_x(n)}, \\
\text{by\_bx\_ratio}(n) &= \frac{b_y(n)}{b_x(n)}, \\
\text{bx\_minus\_bz}(n) &= b_x(n) - b_z(n), \\
\text{net\_log\_balance}(n) &= (b_x(n) - b_z(n)) - G(n) - \log_2(n).
\end{aligned}
$$

Key conditions tested:

$$
b_y(n) < b_x(n) - b_z(n), \\
b_x(n) - b_z(n) > G(n).
$$

4. Statistical Analysis

For $n = 1, 2, \ldots, N$, compute:

1. **Descriptive Statistics**: Mean, standard deviation, quantiles for $b_x, b_y, b_z, G, \text{max\_a}$.
2. **Correlations**: Matrix of Pearson correlations between metrics.
3. **Linear Regression**: Fit $b_z(n)$ vs. $b_x(n)$:

$$
b_z = \beta \cdot b_x + \epsilon, \quad \text{with slope } \beta \text{ and } R^2.
$$

4. **Distributions**: Histograms of metrics and ratios (e.g., b_z/b_x, max_a in log scale).

5. Key Equations

- **Growth Relationship**:

$$
\text{net\_log\_balance}(n) \approx 0 \quad \text{(empirical observation)}.
$$

- **Regression**:

$$
b_z(n) \propto b_x(n) \quad (\text{slope } \beta \approx 0.388 \text{ observed}).
$$

This abstraction captures the dynamical system, metric definitions, and statistical relationships analyzed in the code. The Collatz conjecture's unresolved nature implies these relationships remain empirical observations rather than proven theorems.

```

import sys
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```python
from scipy import stats
import seaborn as sns
from multiprocessing import Pool

# CMSN Generation
def collatz_matthews_step(a, bx, by, bz, g, max_a):
    if a % 2 == 0:
        return a // 2, bx + 1, by, bz, g, max_a
    new_a = 3 * a + 1
    return new_a, bx + 1, by + np.log2(a), bz + 1, g + np.log2(3 + 1/a), max(new_a, max_a)

def generate_cmsn(start):
    a, bx, by, bz, g, max_a = start, 0, 0, 0, 0, start
    while a != 1:
        a, bx, by, bz, g, max_a = collatz_matthews_step(a, bx, by, bz, g, max_a)
    return (start, bx, by, bz, g, max_a)

def generate_chunk(start_end):
    start, end = start_end
    return [generate_cmsn(n) for n in range(start, end + 1)]

def generate_data(max_n, num_processes=16):
    print(f"Generating CMSN data for n=1 to {max_n} with {num_processes} processes...")
    chunk_size = max_n // num_processes
    ranges = [(i * chunk_size + 1, (i + 1) * chunk_size if i < num_processes - 1 else max_n)
              for i in range(num_processes)]

    with Pool(num_processes) as pool:
        results = pool.map(generate_chunk, ranges)

    flat_results = [item for sublist in results for item in sublist]
    df = pd.DataFrame(flat_results, columns=['n', 'bx', 'by', 'bz', 'g', 'max_a'])
    df.to_csv("cmsn_data_30M_with_max_a.csv.gz", compression='gzip', index=False)
    print(f"Data saved to 'cmsn_data_30M_with_max_a.csv.gz'")
    return df

# Analysis Functions (unchanged from prior, but adjusted file names)
def analyze_data(df):
    df['bz_bx_ratio'] = df['bz'] / df['bx']
    df['by_bx_ratio'] = df['by'] / df['bx']
    df['by_bz_ratio'] = df['by'] / df['bz'].replace(0, np.nan)
    df['g_bz_ratio'] = df['g'] / df['bz'].replace(0, np.nan)
    df['bx_minus_bz'] = df['bx'] - df['bz']
    df['by_lt_bx_minus_bz'] = df['by'] < df['bx_minus_bz']
    df['bx_minus_bz_gt_g'] = df['bx_minus_bz'] > df['g']
    df['log_n'] = np.log2(df['n'])
    df['net_log_balance'] = df['bx_minus_bz'] - df['g'] - df['log_n']
```

```
    print("\nBasic Statistics:")
    stats_summary = df[['bx', 'by', 'bz', 'g', 'max_a']].describe()
    print(stats_summary.round(2))
    stats_summary.to_csv("cmsn_stats_summary_30M.csv")

    print("\nKey Conditions:")
    print(f"Mean b_z / b_x: {df['bz_bx_ratio'].mean():.3f}, Max: {df['bz_bx_ratio'].max():.3f}")
    print(f"Sequences where b_y < b_x - b_z: {df['by_lt_bx_minus_bz'].mean()*100:.2f}%")
    print(f"Sequences where b_x - b_z > G: {df['bx_minus_bz_gt_g'].mean()*100:.2f}%")
    print(f"Mean net log balance (should ≈ 0 without b_y): {df['net_log_balance'].mean():.3f}")

    for col, label in [('bx', 'Steps'), ('by', 'Log Odd Sum'), ('bz', 'Odd Count'),
                       ('g', 'Growth G'), ('max_a', 'Max Odd Value')]:
        top5 = df.nlargest(5, col)[['n', col]]
        print(f"\nTop 5 by {label}:")
        print(top5)
        top5.to_csv(f"cmsn_top5_{col}_30M.csv", index=False)

    plt.figure(figsize=(12, 12))
    for i, (col, label) in enumerate([
        ('bx', 'Steps (b_x)'), ('by', 'Log Odd Sum (b_y)'),
        ('bz', 'Odd Count (b_z)'), ('g', 'Odd Growth (G)'), ('max_a', 'Max Odd Value (log scale)')
    ], 1):
        plt.subplot(5, 1, i)
        sns.histplot(df[col], bins=50, kde=True, log_scale=(col == 'max_a', True))
        plt.title(f'Distribution of {label}')
        plt.xlabel(label)
    plt.tight_layout()
    plt.savefig('cmsn_distributions_30M.png')
    plt.close()

    plt.figure(figsize=(10, 6))
    sns.histplot(df['bz_bx_ratio'].dropna(), bins=50, kde=True)
    plt.axvline(0.388, color='r', linestyle='--', label='Threshold 0.388')
    plt.title('Distribution of b_z / b_x')
    plt.xlabel('b_z / b_x')
    plt.legend()
    plt.savefig('bz_bx_distribution_30M.png')
    plt.close()

    sample = df.sample(min(10000, len(df)), random_state=42)
    plt.figure(figsize=(10, 6))
    plt.scatter(sample['bx_minus_bz'], sample['g'], s=1, alpha=0.5)
    plt.plot([0, max(sample['bx_minus_bz'])], [0, max(sample['bx_minus_bz'])], 'r--', label='y = x')
```

````python
    plt.xlabel('b_x - b_z (Even Steps)')
    plt.ylabel('G (Odd Growth)')
    plt.title('b_x - b_z vs G')
    plt.legend()
    plt.savefig('bx_minus_bz_vs_g_30M.png')
    plt.close()

    corr_matrix = df[['bx', 'by', 'bz', 'g', 'max_a']].corr()
    print("\nCorrelation Matrix:")
    print(corr_matrix.round(3))
    plt.figure(figsize=(8, 6))
    sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', vmin=-1, vmax=1)
    plt.savefig('cmsn_correlation_heatmap_30M.png')
    plt.close()

    slope, _, r_value, _, _ = stats.linregress(df['bx'], df['bz'])
    print(f"\nLinear Regression (b_z vs b_x): slope={slope:.3f}, R²={r_value**2:.3f}")

def main(max_n):
    df = generate_data(max_n)
    analyze_data(df)
    print(f"\nAnalysis complete. Data saved to 'cmsn_data_30M_with_max_a.csv.gz' and related files.")

if __name__ == "__main__":
    if len(sys.argv) != 2:
        print("Usage: python cmsn_full_30M.py <max_n>")
        sys.exit(1)
    try:
        max_n = int(sys.argv[1])
        if max_n < 1:
            raise ValueError("max_n must be positive")
    except ValueError as e:
        print(f"Error: {e}")
        sys.exit(1)
    main(max_n)
    ```
````
````