Filament-Simulation - Computing Performance Test

March 11, 2025

[1]:
```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.ndimage import gaussian_filter
import time
import pandas as pd

class FiberImageGenerator:
    def __init__(self, image_size=1024, n_fibers=25, steps=100, fiber_width=10):
        self.image_size = image_size
        self.n_fibers = n_fibers
        self.steps = steps
        self.fiber_width = fiber_width

    # [Previous methods remain unchanged]
    def generate_random_walk(self):
        x = np.zeros(self.steps)
        y = np.zeros(self.steps)
        x[0] = 512
        y[0] = 512
        angle = np.random.uniform(0, 2*np.pi)

        for i in range(1, self.steps):
            dx = 0.7 * np.cos(angle)
            dy = 0.7 * np.sin(angle)
            random_angle = np.random.uniform(0, 2*np.pi)
            dx += 0.3 * np.cos(random_angle)
            dy += 0.3 * np.sin(random_angle)
            x[i] = x[i-1] + dx * 5
            y[i] = y[i-1] + dy * 5
            angle += np.random.normal(0, 0.1)
            x[i] = np.clip(x[i], 0, self.image_size-1)
            y[i] = np.clip(y[i], 0, self.image_size-1)
        return x, y

    def draw_fiber(self, image, x, y, step_num):
        brightness = 255 - (205 * step_num / self.steps)
        for dx in range(-self.fiber_width//2, self.fiber_width//2 + 1):
```

```
            for dy in range(-self.fiber_width//2, self.fiber_width//2 + 1):
                px = int(x + dx)
                py = int(y + dy)
                if 0 <= px < self.image_size and 0 <= py < self.image_size:
                    dist = np.sqrt(dx**2 + dy**2)
                    intensity = brightness * (1 - 0.3 * dist/self.fiber_width)
                    image[py, px] = min(255, max(image[py, px], intensity))

    def add_noise(self, image):
        noise_sigma = 10
        gaussian_noise = np.random.normal(0, noise_sigma, image.shape)
        image = image + gaussian_noise
        pixel_noise = np.random.uniform(-20, 20, image.shape)
        image = image + pixel_noise
        image = gaussian_filter(image, sigma=0.5)
        return np.clip(image, 0, 255)

    def generate_image(self):
        image = np.zeros((self.image_size, self.image_size))
        for _ in range(self.n_fibers):
            x, y = self.generate_random_walk()
            for step in range(self.steps):
                self.draw_fiber(image, x[step], y[step], step)
        image = self.add_noise(image)
        return image.astype(np.uint8)

    def generate_dataset(self, n_images=10):
        images = []
        for _ in range(n_images):
            img = self.generate_image()
            images.append(img)
        return images

def run_simulation(runs=1, scaling_factor=2, n_images=10, base_steps=200):
    """
    Run multiple simulations with scaling applied only from second run onwards
    and sequential image display
    """
    all_stats = []
    all_figures = []  # Store all figures for later display

    for run in range(runs):
        # Modified step calculation: base_steps for first run, scaled for␣
 ↪subsequent runs
        if run == 0:
            current_steps = base_steps  # First run uses base steps
        else:
```

```
            current_steps = base_steps * (scaling_factor ** (run))  # Scaling␣
↪starts from second run

        print(f"\nSimulation Run {run + 1}/{runs}")
        print(f"Current Steps: {current_steps}")
        print("-" * 60)

        # Start timing
        start_time = time.time()

        # Create generator with scaled steps
        generator = FiberImageGenerator(steps=current_steps)
        images = generator.generate_dataset(n_images)

        # Calculate execution time and statistics
        execution_time = time.time() - start_time
        total_elements = generator.n_fibers * current_steps * n_images
        elements_per_second = total_elements / execution_time

        # Store statistics
        stats = {
            'Run': run + 1,
            'Steps': current_steps,
            'Total Elements': total_elements,
            'Execution Time': execution_time,
            'Elements/Second': elements_per_second
        }
        all_stats.append(stats)

        # Print current run statistics
        print(f"Elements per fiber: {current_steps}")
        print(f"Total elements: {total_elements:,}")
        print(f"Execution time: {execution_time:.2f} seconds")
        print(f"Processing speed: {elements_per_second:.0f} elements/second")

        # Create figure for current run
        fig = plt.figure(figsize=(20, 8))
        plt.suptitle(f'Simulation Run {run + 1} - {current_steps} steps',␣
↪fontsize=16, y=1.02)

        for i, img in enumerate(images):
            plt.subplot(2, 5, i+1)
            plt.imshow(img, cmap='gray')
            plt.axis('off')
            plt.title(f'Image {i+1}')

        plt.tight_layout()
```

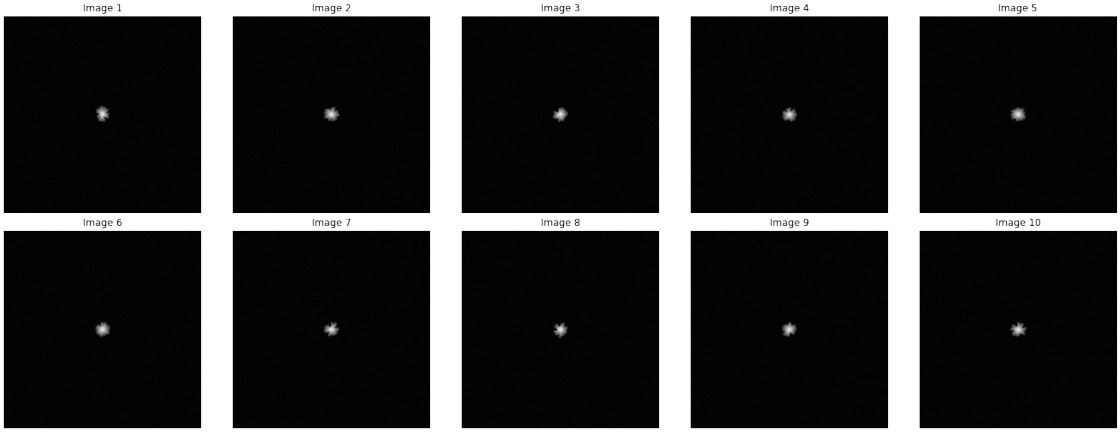
```
        all_figures.append(fig)  # Store figure for later display

    # Create summary statistics
    df_stats = pd.DataFrame(all_stats)
    print("\nSummary Statistics")
    print("=" * 80)
    print("\nDetailed Statistics per Run:")
    print(df_stats.to_string(index=False))

    print("\nPerformance Analysis:")
    print("-" * 80)
    print(f"Total execution time: {df_stats['Execution Time'].sum():.2f}␣
↪seconds")
    print(f"Average elements/second: {df_stats['Elements/Second'].mean():.0f}")
    print(f"Maximum elements/second: {df_stats['Elements/Second'].max():.0f}")
    print(f"Total elements processed: {df_stats['Total Elements'].sum():,}")

    # Create and store performance visualization
    fig_perf = plt.figure(figsize=(12, 6))
    plt.plot(df_stats['Steps'], df_stats['Elements/Second'], 'bo-')
    plt.xlabel('Steps per Fiber')
    plt.ylabel('Processing Speed (Elements/Second)')
    plt.title('Performance Analysis Across Runs')
    plt.grid(True)
    all_figures.append(fig_perf)

    # Display all figures at once
    plt.show()

    return df_stats

# Example usage:
stats = run_simulation(runs=10, scaling_factor=2, n_images=10, base_steps=10)
```

```
Simulation Run 1/10
Current Steps: 10
------------------------------------------------------------
Elements per fiber: 10
Total elements: 2,500
Execution time: 2.19 seconds
Processing speed: 1140 elements/second

Simulation Run 2/10
Current Steps: 20
------------------------------------------------------------
Elements per fiber: 20
```

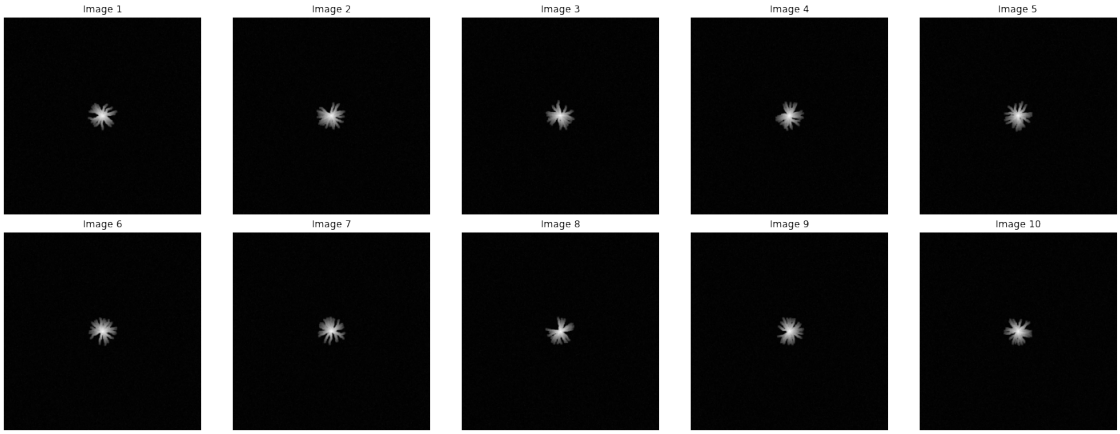
```
Total elements: 5,000
Execution time: 3.61 seconds
Processing speed: 1387 elements/second

Simulation Run 3/10
Current Steps: 40
------------------------------------------------------------
Elements per fiber: 40
Total elements: 10,000
Execution time: 8.48 seconds
Processing speed: 1179 elements/second

Simulation Run 4/10
Current Steps: 80
------------------------------------------------------------
Elements per fiber: 80
Total elements: 20,000
Execution time: 12.23 seconds
Processing speed: 1636 elements/second

Simulation Run 5/10
Current Steps: 160
------------------------------------------------------------
Elements per fiber: 160
Total elements: 40,000
Execution time: 22.93 seconds
Processing speed: 1745 elements/second

Simulation Run 6/10
Current Steps: 320
------------------------------------------------------------
Elements per fiber: 320
Total elements: 80,000
Execution time: 39.87 seconds
Processing speed: 2006 elements/second

Simulation Run 7/10
Current Steps: 640
------------------------------------------------------------
Elements per fiber: 640
Total elements: 160,000
Execution time: 75.02 seconds
Processing speed: 2133 elements/second

Simulation Run 8/10
Current Steps: 1280
------------------------------------------------------------
Elements per fiber: 1280
```

```
Total elements: 320,000
Execution time: 139.30 seconds
Processing speed: 2297 elements/second

Simulation Run 9/10
Current Steps: 2560
------------------------------------------------------------
Elements per fiber: 2560
Total elements: 640,000
Execution time: 275.62 seconds
Processing speed: 2322 elements/second

Simulation Run 10/10
Current Steps: 5120
------------------------------------------------------------
Elements per fiber: 5120
Total elements: 1,280,000
Execution time: 548.76 seconds
Processing speed: 2333 elements/second

Summary Statistics
================================================================================

Detailed Statistics per Run:
 Run  Steps  Total Elements  Execution Time  Elements/Second
   1     10            2500        2.193811      1139.569608
   2     20            5000        3.605950      1386.597015
   3     40           10000        8.478987      1179.386173
   4     80           20000       12.226321      1635.815061
   5    160           40000       22.927434      1744.634836
   6    320           80000       39.870718      2006.485036
   7    640          160000       75.017661      2132.831092
   8   1280          320000      139.303071      2297.149652
   9   2560          640000      275.617617      2322.057661
  10   5120         1280000      548.758659      2332.537225

Performance Analysis:
--------------------------------------------------------------------------------
Total execution time: 1128.00 seconds
Average elements/second: 1818
Maximum elements/second: 2333
Total elements processed: 2,557,500
```

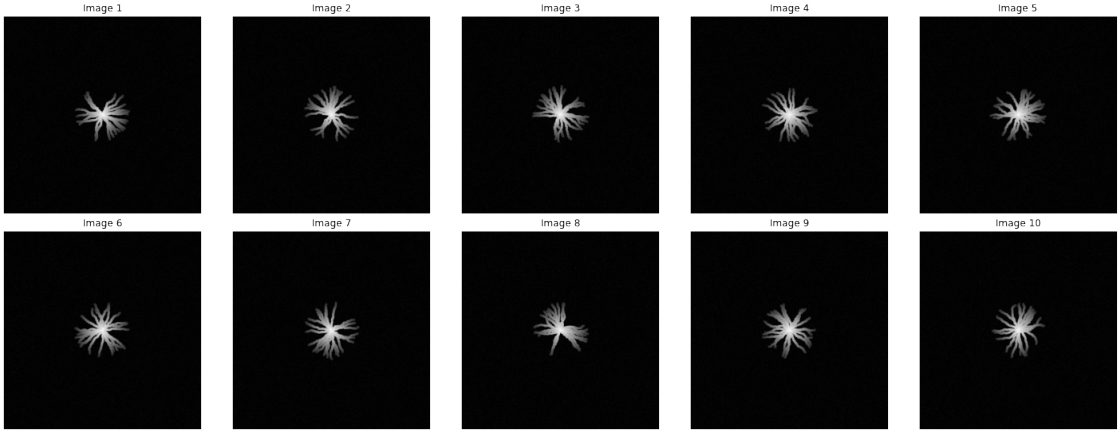
Simulation Run 1 - 10 steps



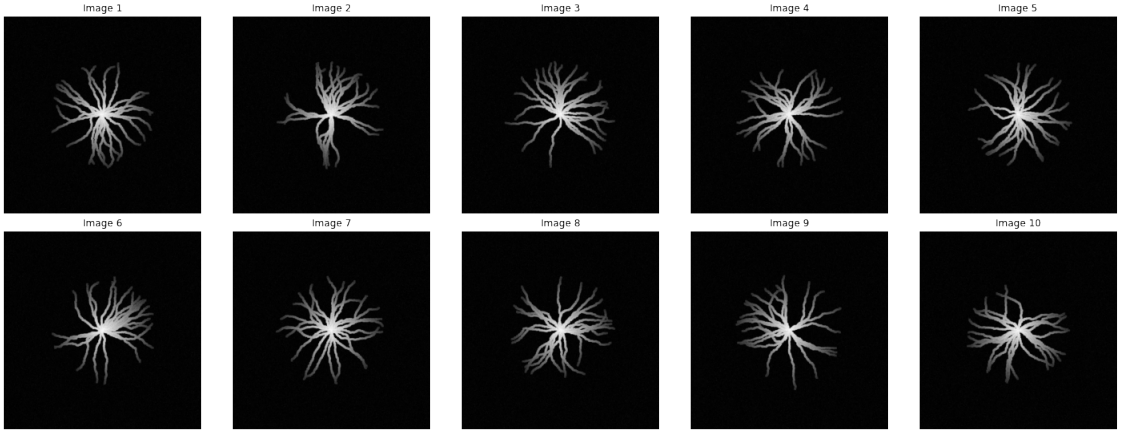
Simulation Run 2 - 20 steps



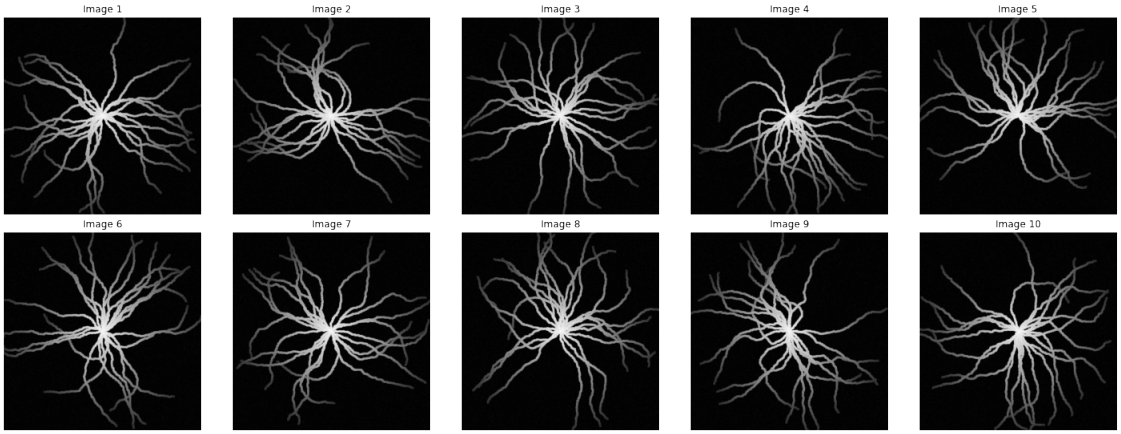
Simulation Run 3 - 40 steps



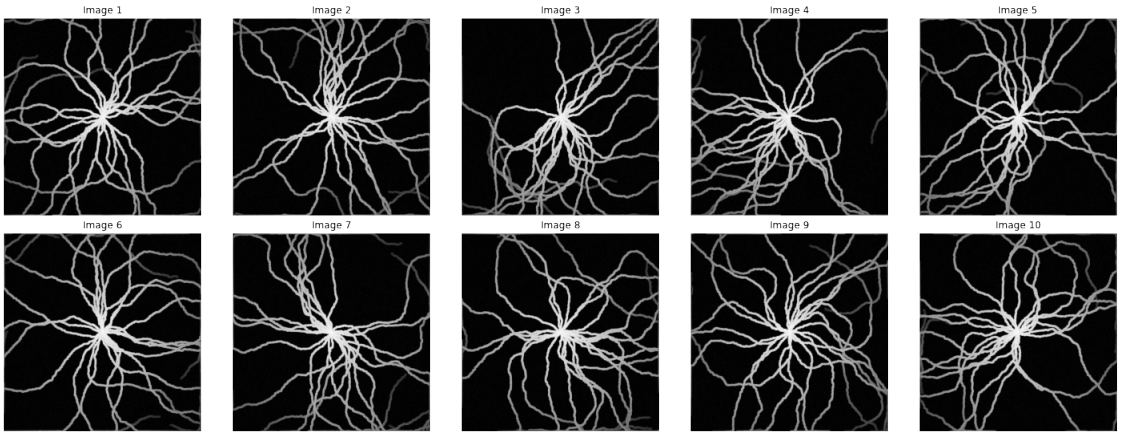
Simulation Run 4 - 80 steps



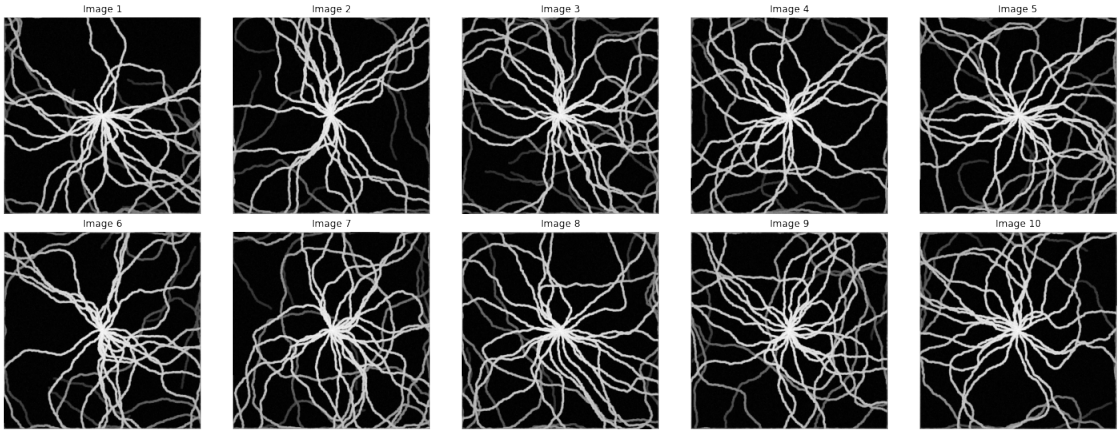
Simulation Run 5 - 160 steps



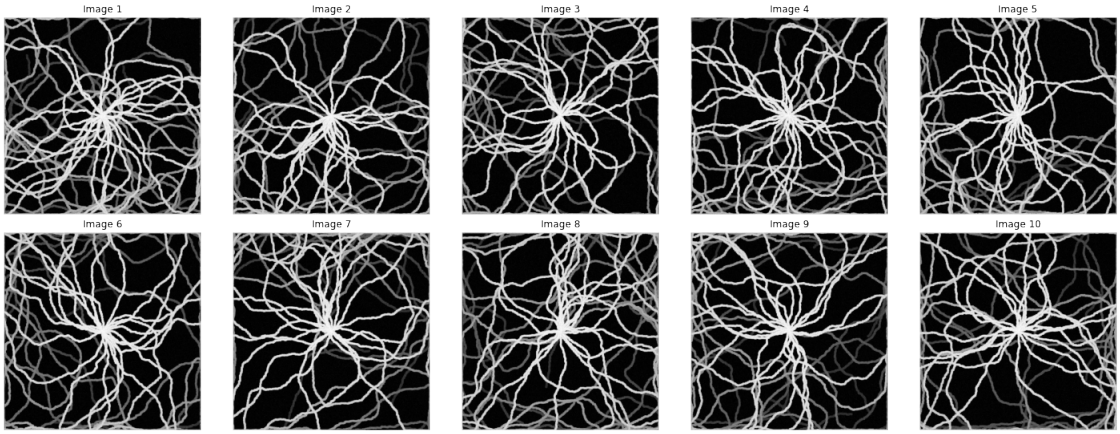
Simulation Run 6 - 320 steps



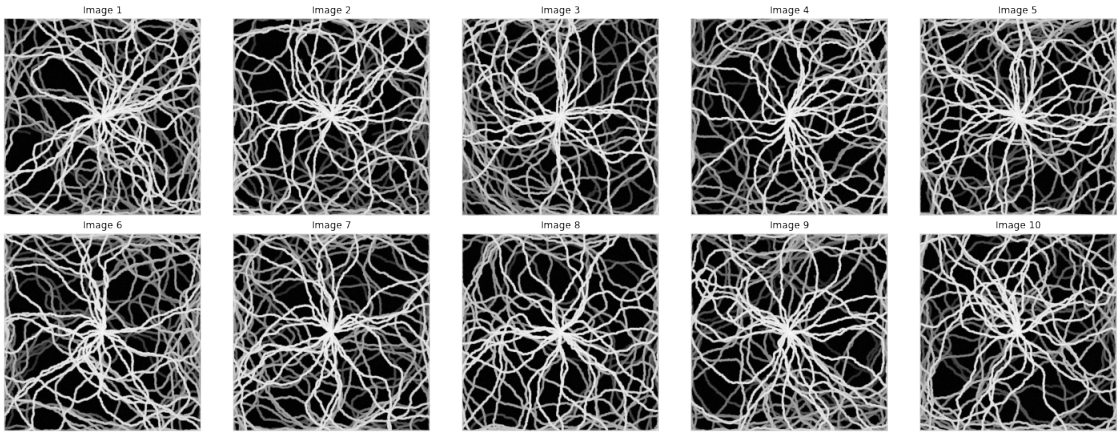
Simulation Run 7 - 640 steps

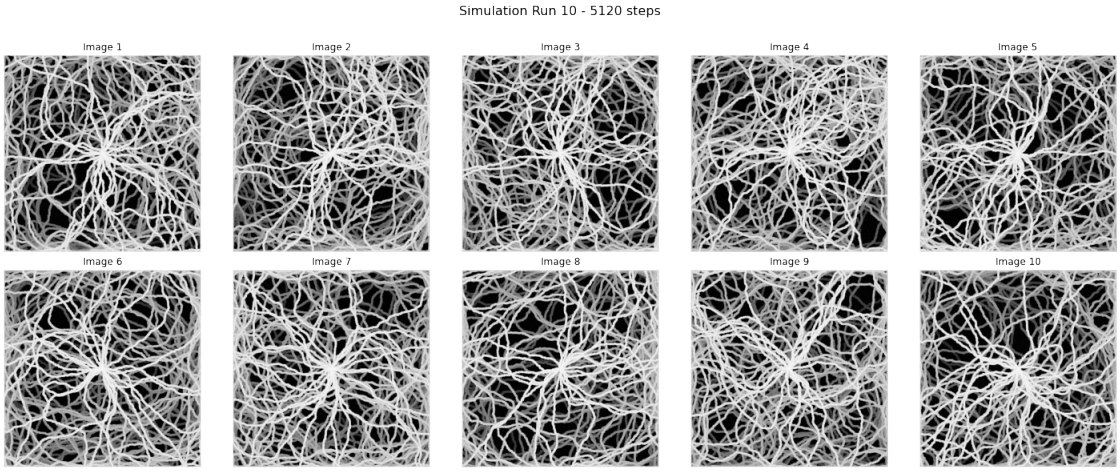


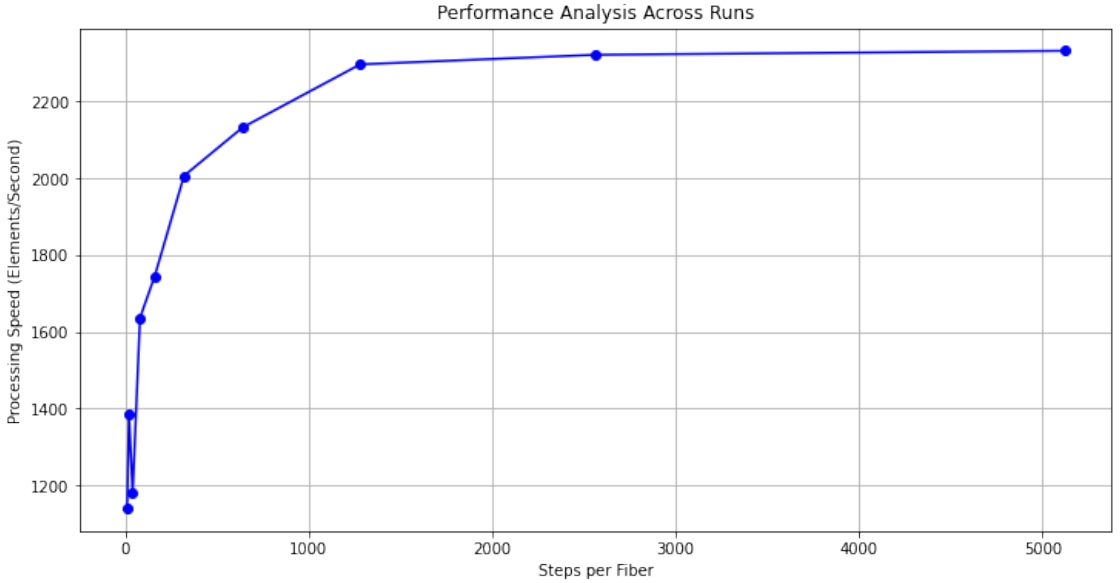
Simulation Run 8 - 1280 steps



Simulation Run 9 - 2560 steps




Simulation Run 10 - 5120 steps
Image 1
Image 2
Image 3
Image 4
Image 5
Image 6
Image 7
Image 8
Image 9
Image 10


Performance Analysis Across Runs
Processing Speed (Elements/Second)
Steps per Fiber
1200
1400
1600
1800
2000
2200
0
1000
2000
3000
4000
5000

[ ]: