Assignment 8 Report

High Performance Computing

James Smith

Pratik Mitra

1.

Euler ODE

Name	Time (s)	Integral Value	Error
Pure Python	12.39480042	0.8390718	2.73E-07
Numba1	8.463911533	0.8390718	2.73E-07
Numba2	0.353434324	0.8390718	2.73E-07

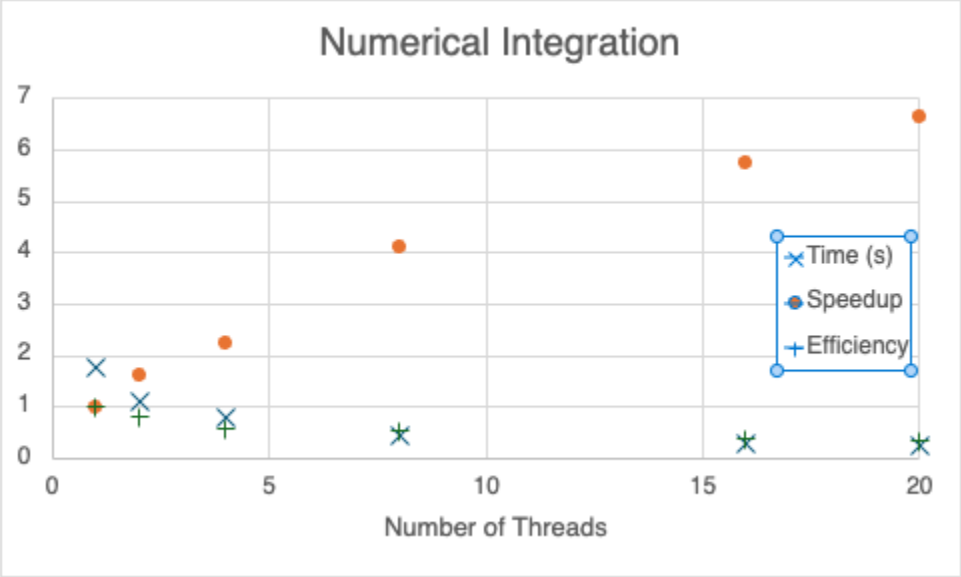
As shown by these results, incorporating jit from the Numba library significantly speeds up function execution time. Just-in-time execution further speeds up this execution in the Numba2 python script.

Execution Time Line-by-line results:

```
Line #      Hits         Time  Per Hit   % Time  Line Contents
==============================================================
    52         1          1.8      1.8      0.0      nevals = int((tmax-t0)/dt)
    55         1          9.6      9.6      0.0      y = np.zeros(nevals+1)
    56         1          5.0      5.0      0.0      t = np.zeros(nevals+1)
    59         1          1.4      1.4      0.0      y[0] = y0
    60         1          0.3      0.3      0.0      t[0] = t0
    64  10000001    1986056.9      0.2      9.3      for i in range(nevals):
    65  10000000   15588099.4      1.6     73.1          y[i+1] = dt*int_funct(y[i],t[i]) + y[i]
    66  10000000    3749574.0      0.4     17.6          t[i+1] = t[i] + dt
    70         1          2.2      2.2      0.0      return  y, t
```

The lines that take the most time are the lines that initialize the arrays for y and t in the euler's method function. These lines likely take the most time because they involve rewriting the largest amount of memory to make space for the array.

2.



Numerical Integration

Name or # of Threads	Time(s)	Integral Value	Speedup	Efficiency
Pure Python	78.410316	1.804776	N/A	N/A
1	1.764196	1.804776	1	1
2	1.104661	1.804776	1.59704742	0.79852371
4	0.792431	1.804776	2.226308663	0.556577166
8	0.428604	1.804776	4.116144506	0.514518063
16	0.306155	1.804776	5.762427529	0.360151721
20	0.265872	1.804776	6.635508816	0.331775441

3. Bonus: Cython – Matrix-matrix multiplication

Matrix	3 x 3 (seconds)	10 x 10 (seconds)	100 x 100 (seconds)	1000 x 1000 (seconds)
Cython	0.000003	0.000003	0.001232	1.212368
NumPy	0.000667	0.000108	0.000590	0.005499

The table illustrates a comparison between Cython and NumPy for matrix-matrix multiplication. Cython is faster for smaller matrices but becomes less efficient as matrix size increases, where NumPy's optimized algorithms take the lead.