Accelerating MCMC-driven Gaussian Plumes with Numba

T. N. Stenborg1,2, S. C. Davis1,2

1*The University of Sydney, Sydney, New South Wales 2006, Australia;*

2*ARC Training Centre in Data Analytics for Resources and Environments (DARE);*

*travis.stenborg@sydney.edu.au*

**Abstract.** Data science can be limited by the computational intensity of numerical analysis. Technologies or methods that ameliorate this thus expand the range of tractable problems. Numba, a just-in-time compiler for Python and NumPy, is one such technology. This study quantifies the performance boost Numba provides to Python code. MCMC-driven Gaussian plume modelling of seawater droplets was used as a test case. Benchmarking showed an 88.8% speedup.

## Introduction

Data science problems can be computationally intensive (e.g., Tapiador et al. 2017, Polson and Sokolov 2020, Kenett et al. 2022). This problem can be naively addressed by making more computational resources available (e.g. faster processors, more RAM, multicore computing). Alternatively, it can be addressed by reducing the amount of computation needed to solve a given problem.

Python is common data science language (Ordonez 2021). It can suffer performance issues however, for certain computational problems due to being an interpreted, rather than a compiled language (Strout et al. 2019). Numba addresses this issue by producing compiled Python and NumPy code, at least for a subset of commands (Lam et al. 2015). Moreover, it's typically implementable via a modest level of refactoring and decorator addition (De Pra and Fontana 2020). For problems hampered by Python's computational limits, incorporating Numba can thus provide a performance boost for a smaller time investment than competing amelioration strategies, such as rewriting a program in C, C++ or Cython.

This study quantifies the performance enhancement Numba provides to a test program. The program tested was prototype MCMC-driven Gaussian plume modelling, essentially a fluid dynamics simulation of droplets moving through a carrier fluid. For an introduction to Gaussian Plumes, see e.g., De Visscher 2014.

## Methods

## MCMC-driven Gaussian Plumes

The prototype MCMC-driven Gaussian plume code developed here was used to model dispersion of seawater droplets in air. This modelling incorporated third-party 3D droplet concentration data (a 6997 row subset), collected via drone over the Great Barrier Reef near Townsville, Queensland. The test platform was Windows 11, version 22H2, on an Intel "Comet Lake" Core i7-10750H (with 16GB RAM) system. Python 3.9.13 and Numba 0.56.3 were invoked from Visual Studio Code 1.73.1.

Two versions of the Gaussian plume code were compared: one standard Python version, and another differing only by the addition of Numba decorators. Execution was configured to run for an arbitrary 10,000 samples. Each version of the code was tested 200 times, using the same random number generator seed each time. Numba was confirmed as running exclusively in *nopython* mode, the fastest of its two available operating modes. All foreground applications were closed, and networking connectivity terminated, to reduce benchmarking noise.

## Results

Performance testing results are given in Figure 1, below. The first iteration of the Numba version was noticeably slow, as the overhead of code compilation was incurred. Subsequent calls to that code invoke the compiled version.

Numba provided an average speedup of 88.8%.

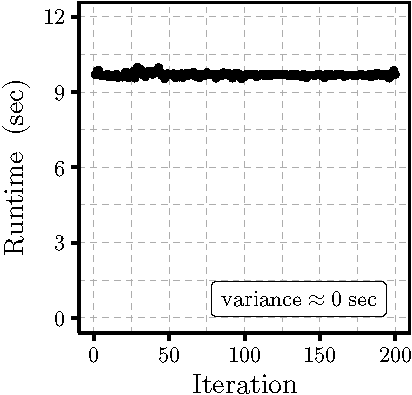
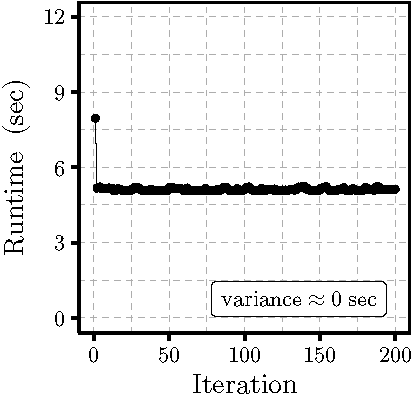
 

Figure 1. Runtime testing of unmodified (left) vs Numba-enhanced (right) Gaussian plume models. Each iteration performed 10,000 MCMC samples.

## Discussion

Application of Numba to the example application provided an unambiguous performance boost (average 88.8% over 200 iterations). Refactoring measures adopted in incorporating Numba included e.g., switching from Pandas to Numpy data structures, avoiding Python lists, and some changes to use of function types (viz. keyword, positional).

## Acknowledgements

This research was conducted by the Australian Research Council Training Centre in Data Analytics for Resources and Environments (project number ICI9010031).

**References**

De Pra, Y & Fontana, F, 2020, "Programming real-time sound in Python", Applied Sciences, vol. 10, no. 12, p. 4214.

De Visscher, A, 2014, "Air Dispersion Modeling: Foundations and Applications", Wiley, Somerset, New Jersey.

Lam, SK, Pitrou, A & Seibert, S, 2015, "Numba: A LLVM-based Python JIT Compiler", in Proc. Second Workshop on the LLVM Compiler Infrastructure in HPC, pp. 1-6, ACM.

Kenett, RS, Gotwalt, C, Freeman, L, & Deng, X, 2022, "Self-supervised cross validation using data generation structure", Applied Stochastic Models in Business and Industry, vol. 38, no. 5, pp. 750–765.

Ordonez, C, 2021, "A Comparison of Data Science Systems", in Big Data Analytics, vol. 12581, pp. 3–11, Springer International Publishing, Cham.

Polson, N & Sokolov, V, 2020, "Deep learning: Computational aspects", Wiley Interdisciplinary Reviews Computational Statistics, vol. 12, no. 5.

Strout, MM, Debray, S, Isaacs, K, Kreaseck, B, Cárdenas-Rodríguez, J, Hurwitz, B, Volk, K, Badger, S, Bartels, J, Bertolacci, I, Devkota, S, Encinas, A, Gaska, B, Neth, B, Sackos, T, Stephens, J, Willer, S, & Yadegari, B, 2019, "Language-Agnostic Optimization and Parallelization for Interpreted Languages", in Languages and Compilers for Parallel Computing, pp. 36–46, Springer International Publishing, Cham.

Tapiador, D, Berihuete, A, Sarro, LM, Julbe, F, & Huedo, E, 2017, "Enabling data science in the Gaia mission archive: The present-day mass function and age distribution", Astronomy and Computing, vol. 19, pp. 1–15.