Particle Filter Speed Up:
A Comparison of Different Python Resampling Implementations

Ashton Palacios | Brigham Young University

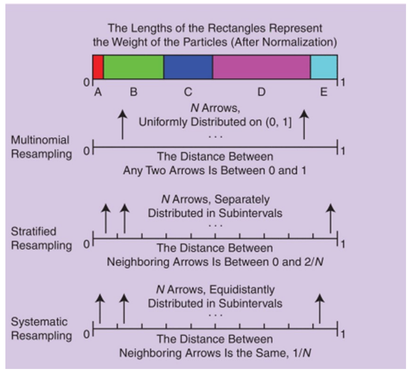
CS 677
Bayesian
Methods

ABSTRACT

- Particle filters are widely used as a localizer in many robotic systems.
- Particle filters work by resampling particles to estimate the true posterior of the location of the system.
- Resampling is computationally expensive, but there are newer libraries to speed up the computation.
- Numpy, Numba [1], Jax [2], and C++ implementations are explored and compared with respect to their runtimes for differing number of particles.

Introduction

- There are many ways to resample the particles [3]. The most widely used, and one of the slowest, is called multinomial resampling.
- Another popular and considerably faster way to resample is called systematic resampling.



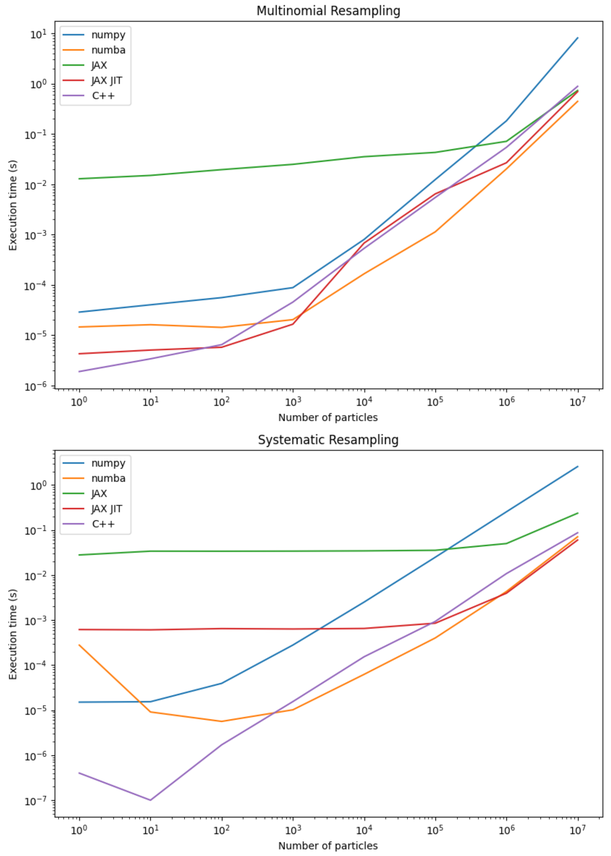
- I present the runtime comparisons for these two techniques when they are implemented in Numpy, Numba, JAX, JAX with JIT, and single core C++.

Methods

- Each algorithm is implemented in each of the frameworks. The code for each framework is approximately the same with different function decorators to ensure fair comparison.
- Each implementation is timed 10 times for differing number of particles to be resampled (1, 10, 100, 1000, 10000, 100000, 1000000, 10000000).

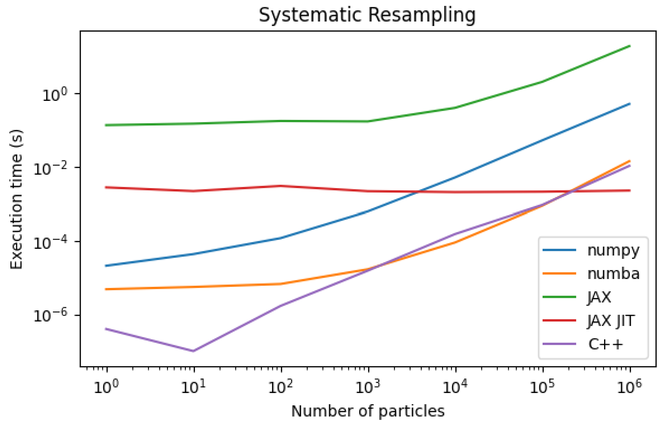
Results

- 11th Gen Intel i9-11900 8 core processor with 64G of Ram



Results Cont.

- Google Colab GPU



- GPU did not cause a considerable speed up in comparison to the CPU runtimes. GPU ran out of memory for the 10M particle experiment.

Conclusion

1. State of the art Python solutions can provide speed ups to make particle filter resampling fast enough to be used in real time systems.
2. Care needs to be taken when using these libraries to avoid exponential memory consumption.
3. C++ implementations are still the fastest, and with tools like ChatGPT[4] and Github Copilot[5], are easier than ever to implement and use.

References

1. Numba. (2023, April). Numba: A High Performance Python Compiler. https://numba.pydata.org
2. Jax. (2023, April). JAX: Autograd and XLA. GitHub Repository. https://github.com/google/jax
3. T. Li, M. Bolic and P. M. Djuric, "Resampling Methods for Particle Filtering: Classification, implementation, and strategies," in IEEE Signal Processing Magazine, vol. 32, no. 3, pp. 70-86
4. ChatGPT. (2023, April). ChatGPT: An AI language model trained by OpenAI. GitHub Repository. https://github.com/kasvith/chat-gpt3
5. Github Copilot. (2023, April). Github Copilot: An AI pair programmer. https://docs.github.com/en/copilot/quickstart