Department: Head
Editor: Name, xxxx@email

PyExaFMM: Designing a highly-performant particle fast multipole solver in Python with Numba

S. Kailasa
Department of Mathematics, University College London

T. Betcke
Department of Mathematics, University College London

T. Wang
Department of Mechanical and Aerospace Engineering, The George Washington University

L. A. Barba
Department of Mechanical and Aerospace Engineering, The George Washington University

***Abstract*—PyExaFMM is a pythonic kernel-independent particle fast multipole method (FMM) implementation, built on the success of the ExaFMM project, to answer the question: can we develop a highly-performant scientific code without resorting to a lower level language? The FMM is a good case study to understand the maturity of Python in the development of high-performance software, due its reliance on a complex heirarchical octree data structure. In this paper we offer an overview the kernel-independent FMM algorithm and the mathematical and software development techniques we used in order to develop a performant practical implementation. We proceed to benchmark the software's accuracy, speed, and memory footprint with respect to the state of the art C++ implementation from the ExaFMM project. We report that we are able to achieve runtimes within $\mathcal{O}(10)$ of the state of the art, with comparable accuracy.**

THE INTRODUCTION What is the appeal of developing HPC codes in Python? Why is it a useful case-study, what does it mean. What is the FMM approximately, where is it used, why is it important? What will we talk about here, and what do we conclude? What will the remaining sections say?

 Published by the IEEE Computer Society

THE KERNEL-INDEPENDENT FAST MULTIPOLE METHOD

Algorithm

First introduced by Ying [1], the kernel-independent fast multipole method (KIFMM) ...

TECHNIQUES FOR ACHIEVING PERFORMANCE

What is Numba?

What is does, how is it useful? Where do we use it, and why there. How much difference can it make in an idealized routine. What doesn't work, why doesn't it work. Where to be careful. Programming to an (invisible) framework ...

Where is numba used heavily? Tree construction routines on Morton coordinates. Multithreading of tree construction, as well as P2M evaluation. Experiment to demonstrate the speed of kernel evaluation, with caveat that data must be pre-organised.

Precomputing Operators

Transfer vectors, hashing, HDF5, loading into memory.

Compressing the M2L step with a randomised SVD

Introduce rSVD [4]. Numerical bounds on error out of scope, but show how experiments demonstrate that the FMM error dominates the SVD error.

Software Architecture

The key is separating routines to be accelerated, and organising data ready to run. The data organisation part (and it's slowness) should be demonstrated as a bottleneck using experiment.

PERFORMANCE COMPARISON WITH STATE OF THE ART

Description of main experiments, and how they are conducted. What are the main results? Are they expected from theory? What conclusions can be drawn about developing HPC codes entirely in Python, is it worth it?

CONCLUSION

What have we learned, what will we be working on next?

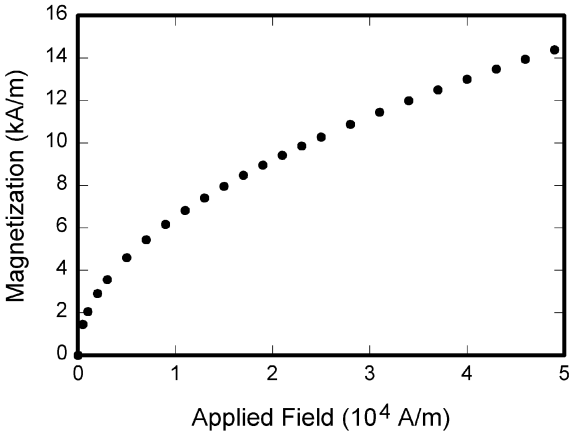

Figure 1. Note that "Figure" is spelled out. There is a period after the figure number, followed by one space. It is good practice to briefly explain the significance of the figure in the caption. (Used, with permission, from [4].)

ACKNOWLEDGMENT

SK is supported by EPSRC Studentship 2417009.

REFERENCES

1. L. Ying, G. Biros, and D. Zorin, "A kernel-independent adaptive fast multipole algorithm in two and three dimensions," J. Comput. Phys., vol. 196, no. 2, pp. 591–626, 2004.
2. I. Lashuk et al.,"A massively parallel adaptive fast multipole method on heterogeneous architectures," Commun. ACM, vol. 55, no. 5, pp. 101–109, May 2012.
3. D. Malhotra and G. Biros, "A Distributed Memory Fast Multipole Method for Volume Potentials," ACM Trans. Math. Softw., vol. 43, no. 2, pp. 1–27, Sep. 2016.
4. N. Halko, P. G. Martinsson, and J. A. Tropp, "Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions *," vol. 53, no. 2, pp. 217–288.

Srinath Kailasa is a graduate student at University College London. He is currently pursuing a PhD in Computational Mathematics, having received an MPhys in Physics (2017) and an MSc Scientific Computing (2020) from the University of Durham, and University College London respectively. His research interests are in high-performance and scientific computing. Contact him at srinath.kailasa.18@ucl.ac.uk.

Timo Betcke is a Professor of Computational Math-

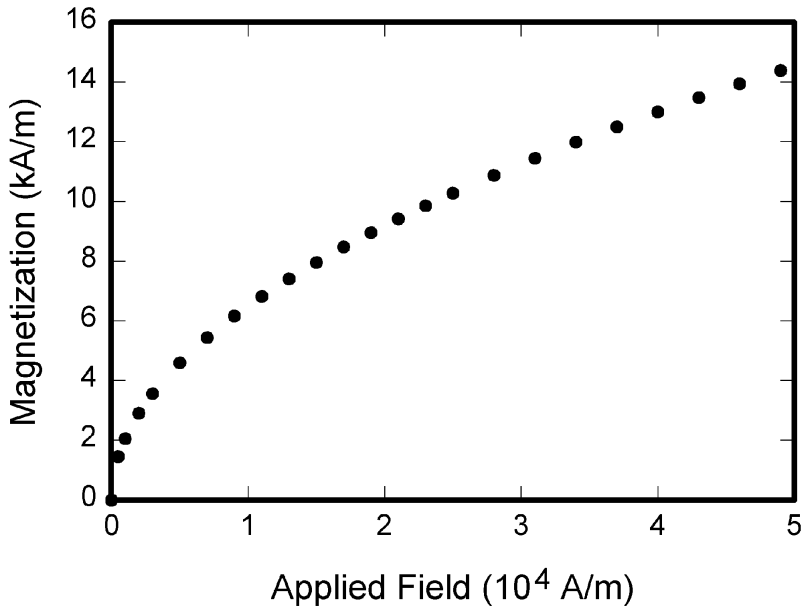

Figure 2. Note that “Figure” is spelled out. There is a period after the figure number, followed by one space. It is good practice to briefly explain the significance of the figure in the caption. (Used, with permission, from [4].)

ematics at University College London. Contact him at t.betcke@ucl.ac.uk.

Tingyu Wang is a PhD student in Mechanical Engineering at the George Washington University. Contact him at twang66@email.gwu.edu.

Lorena. A. Barba is a Professor of Mechanical and Aerospace Engineering at the George Washington University. Contact her at labarba@email.gwu.edu.

Table 1. Units for magnetic properties.

Symbol	Quantity	Conversion from Gaussian and CGS EMU to SI[a]
Φ	Magnetic flux	1 Mx $\rightarrow 10^{-8}$ Wb $= 10^{-8}$ V $\cdot$ s
B	Magnetic flux density, magnetic induction	1 G $\rightarrow 10^{-4}$ T $= 10^{-4}$ Wb/m^2
H	Magnetic field strength	1 Oe $\rightarrow 10^{-3}/(4\pi)$ A/m
m	Magnetic moment	1 erg/G $=$ 1 emu $\rightarrow 10^{-3}$ A $\cdot$ m^2 $= 10^{-3}$ J/T
M	Magnetization	1 erg/(G $\cdot$ cm^3) $=$ 1 emu/cm^3 $\rightarrow 10^{-3}$ A/m
$4\pi M$	Magnetization	1 G $\rightarrow 10^{-3}/(4\pi)$ A/m
σ	Specific magnetization	1 erg/(G $\cdot$ g) $=$ 1 emu/g $\rightarrow$ 1 A $\cdot$ m^2/kg
j	Magnetic dipole moment	1 erg/G $=$ 1 emu $\rightarrow 4\pi \times 10^{-10}$ Wb $\cdot$ m
J	Magnetic polarization	1 erg/(G $\cdot$ cm^3) $=$ 1 emu/cm^3 $\rightarrow 4\pi \times 10^{-4}$ T
χ, κ	Susceptibility	$1 \rightarrow 4\pi$
χ_ρ	Mass susceptibility	1 cm^3/g $\rightarrow 4\pi \times 10^{-3}$ m^3/kg
μ	Permeability	$1 \rightarrow 4\pi \times 10^{-7}$ H/m $= 4\pi \times 10^{-7}$ Wb/(A $\cdot$ m)
μ_r	Relative permeability	$\mu \rightarrow \mu_r$
w, W	Energy density	1 erg/cm^3 $\rightarrow 10^{-1}$ J/m^3
N, D	Demagnetizing factor	$1 \rightarrow 1/(4\pi)$

Vertical lines are optional in tables. Statements that serve as captions for the entire table do not need footnote letters.

[a]Gaussian units are the same as cg emu for magnetostatics; Mx = maxwell, G = gauss, Oe = oersted; Wb = weber, V = volt, s = second, T = tesla, m = meter, A = ampere, J = joule, kg = kilogram, H = henry.