The Journal of Open Source Software

kifmm-rs: A Kernel-Independent Fast Multipole Framework in Rust

Srinath Kailasa [1]

1 Department of Mathematics, University College London, UK

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Open Journals

Reviewers:

- @openjournals

Submitted: 01 January 1970
Published: unpublished

Summary

`kifmm-rs` is an open-source framework for implementing Fast Multipole Methods (FMMs) in shared and distributed memory. FMMs accelerate the calculation of N-body potential evaluation problems that arise in computational science and engineering of the form,

$$\phi(x_i) = \sum_{j=1}^{N} K(x_i, y_j) q(y_j) \tag{1}$$

where the potential ϕ is evaluated at a set of target particles, $\{x_i\}_{i=1}^M$, due to a set of densities, $\{q_j\}_{j=1}^N$ collocated at a set of source particles $\{y_j\}_{j=1}^N$. This evaluation can be interpreted as a matrix-vector multiplication,

$$\boldsymbol{\phi} = \mathbf{K}\mathbf{q} \tag{2}$$

where $\boldsymbol{\phi}$ is an M dimensional vector, $\mathbf{K}$ is a dense $M \times N$ matrix and $\mathbf{q}$ is an N dimensional vector. As $\mathbf{K}$ is dense, the naive evaluation of (2) is of $O(M \cdot N)$ complexity. FMMs provide a way of evaluating (2) in $O(P(M+N))$ where P is a user defined parameter $P \ll M, N$ called the *expansion order*, for a range of interaction kernels $K(\cdot)$ that commonly arise from elliptic partial differential equations (PDEs). The prototypical example for which the FMM was first presented is the Laplace kernel (Greengard & Rokhlin, 1987), which describes particles interacting electrostatically or gravitationally,

$$K(x,y) = \begin{cases} \frac{1}{2\pi}\log(\frac{1}{\|x-y\|}), & \text{2D} \\ \frac{1}{4\pi\|x-y\|}, & \text{3D} \end{cases} \tag{3}$$

Since their initial introduction FMMs have been developed for a wide variety of kernel functions such as the Helmholtz kernel which arises from the time-independent wave equation (Rokhlin, 1993),

$$K(x,y) = \begin{cases} \frac{i}{4}H_0^{(1)}(k|\mathbf{x}-\mathbf{y}|), \text{2D} \\ \frac{e^{ik|\mathbf{x}-\mathbf{y}|}}{4\pi|\mathbf{x}-\mathbf{y}|}, \text{3D} \end{cases} \tag{4}$$

As well as the Stokes kernel describing fluid flow (Tornberg & Greengard, 2008), and the Navier kernel used in elastostatics (Fu et al., 1998) among others. The major application of FMMs is found in the accelerated application of dense operator matrices that arise from the integral formulation of partial differential equations, and as a result FMM algorithms are a crucial component across simulations in many domains, from medical imaging to geophysics.

FMMs accelerate the evaluation of (2) by decomposing the problem domain using a hierarchical tree, a quadtree in 2D and an octree in 3D, the algorithm consisting of a series of operations collectively referred to as *field translations*. Evaluations of the off-diagonal blocks of the matrix $\mathbf{K}$ in (2) correspond to the 'multipole to local' or M2L field translation which summarise the long-range interactions of a local cluster of target points, and the evaluation of the diagonal blocks correspond to unavoidable direct evaluations of the kernel function for evaluating near-range interactions.

Many parts of an implementation are common across FMM algorithms, such as the tree setup and the kernel function implementation. As a result, our framework is presented as a set of modular, re-usable, sub-components of each of the key algorithmic sub-components: (i) the tree in shared and distributed memory, (ii) the field translation operations and (iii) the kernel evaluation. We use our framework to develop an implementation of the so-called kernel independent Fast Multiple Method (kiFMM) (Ying et al., 2004), compatible with a wide variety of elliptic PDE kernels with an implementation for the Laplace and Helmholtz kernels provided currently. The kiFMM has favourable implementation properties due to its formulation in terms of high operational-intensity BLAS operations.

Statement of need

Other notable software efforts for the FMM include PVFMM (Malhotra & Biros, 2015), ExaFMM (Wang et al., 2021), ScalFMM (Blanchard et al., 2015) and TBFMM (Bramas, 2020). PVFMM, ScalFMM and TBFMM are fully distributed implementations of the black box FMM (Fong & Darve, 2009) and kiFMM respectively. The former two are notable for being distributed using a task-based runtime, with the latter using a more traditional MPI based approach. ExaFMM offers a shared memory implementation, with Python interfaces and a simple template based design. A commonality of previous implementations is the coupling of algorithmic optimisation with implementation optimisation. For example, ExaFMM and PVFMM both offer field translations that are highly optimised for x64 architectures and lack ARM support, with ScalFMM and TBFMM being tailored to the runtime systems they are designed for.

Our design is data oriented, with complex behaviour composed over simple linear data structures using Rust's trait system. Traits offer a way of specifying behaviour defining contracts between types that are enforced by Rust's compiler. This enables the exposure of performance critical sections in a manner that is easy to optimise in isolation. In contrast to previous software efforts, our design enables a decoupling of the underlying algorithmic implementation and the software optimisation. This has enabled the comparative analysis of the implementation of critical algorithmic sub-components (Kailasa et al., 2024).

Our principle contributions with `kifmm-rs` can therefore be summarised as:

- *A modular data-oriented design* which enables field translations to be implementd over simple linear data structures, allowing us to easily examine the impact of different algorithmic approaches and computational backends, such as BLAS libraries, for critical algorithmic sub-components.
- *Optimisations for ARM and x86 targets*. ARM targets are increasingly common in high-performance computing systems, with portability enabled by Rust's LLVM based compiler.
- *Competitive shared and distributed memory performance*. With state of the art M2L performance (Kailasa et al., 2024), as well as a communication-optimal distributed memory implementation inspired by (Ibeid et al., 2016).
- *A C API*, using Rust's C ABI compatibility allowing for the construction of bindings into other languages, with full Python bindings for non-specialist users.
- *A moderate frequency Helmholtz FMM*. Helmholtz FMMs are often presented for the low-frequency case Malhotra & Biros (2015), due to the challenging data sizes involved.

> We present an extension of the kiFMM to the Helmholtz problem which has proven so far to be effective in single precision for high wave numbers of up to $k \sim 100$.

`kifmm-rs` is currently a core library used as a part of the BemPP boundary element project (Betcke & Scroggs, 2024). A full API description is available as a part of published documentation. Both Python and Rust examples can be found in the repository.

Acknowledgements

Srinath Kailasa is supported by EPSRC Studentship 2417009

References

Betcke, T., & Scroggs, M. (2024). *Bempp-rs: Boundary element methods in rust*. https://github.com/bempp/bempp-rs

Blanchard, P., Bramas, B., Coulaud, O., Darve, E., Dupuy, L., Etcheverry, A., & Sylvand, G. (2015). ScalFMM: A generic parallel fast multipole library. *SIAM Conference on Computational Science and Engineering (SIAM CSE 2015)*.

Bramas, B. (2020). TBFMM: A c++ generic and parallel fast multipole method library. *Journal of Open Source Software*, *5*(56), 2444.

Fong, W., & Darve, E. (2009). The black-box fast multipole method. *Journal of Computational Physics*, *228*(23), 8712–8725.

Fu, Y., Klimkowski, K. J., Rodin, G. J., Berger, E., Browne, J. C., Singer, J. K., Van De Geijn, R. A., & Vemaganti, K. S. (1998). A fast solution method for three-dimensional many-particle problems of linear elasticity. *International Journal for Numerical Methods in Engineering*, *42*(7), 1215–1229.

Greengard, L., & Rokhlin, V. (1987). A fast algorithm for particle simulations. *Journal of Computational Physics*, *73*(2), 325–348.

Ibeid, H., Yokota, R., & Keyes, D. (2016). A performance model for the communication in fast multipole methods on high-performance computing platforms. *International Journal of High Performance Computing Applications*, *30*(4), 423–437. https://doi.org/10.1177/1094342016634819

Kailasa, S., Betcke, T., & Kazdadi, S. E. (2024). M2L translation operators for kernel independent fast multipole methods on modern architectures. *arXiv Preprint arXiv:2408.07436*.

Malhotra, D., & Biros, G. (2015). PVFMM: A parallel kernel independent FMM for particle and volume potentials. *Communications in Computational Physics*, *18*(3), 808–830.

Rokhlin, V. (1993). Diagonal forms of translation operators for the helmholtz equation in three dimensions. *Applied and Computational Harmonic Analysis*, *1*(1), 82–93.

Tornberg, A.-K., & Greengard, L. (2008). A fast multipole method for the three-dimensional stokes equations. *Journal of Computational Physics*, *227*(3), 1613–1619.

Wang, T., Yokota, R., & Barba, L. A. (2021). ExaFMM: A high-performance fast multipole method library with c++ and python interfaces. *Journal of Open Source Software*, *6*(61), 3145.

Ying, L., Biros, G., & Zorin, D. (2004). A kernel-independent adaptive fast multipole algorithm in two and three dimensions. *Journal of Computational Physics*, *196*(2), 591–626.