The Journal of Open Source Software

FLAM: Fast Linear Algebra in MATLAB - Algorithms for Hierarchical Matrices

Kenneth L. Ho[1]

1 Center for Computational Mathematics, Flatiron Institute

DOI: 10.21105/joss.01906

Software

- Review
- Repository
- Archive

Editor: Melissa Weber Mendonça

Reviewers:

- @sivaramambikasaran
- @pitsianis
- @vijaysm

Submitted: 07 November 2019
Published: 04 July 2020

Summary

Many large matrices in science and engineering possess a special hierarchical low-rank structure that enables fast multiplication and inversion, among other fundamental operations. Such matrices commonly occur in physical problems, including the classical integral (IE) and differential equations (DE) of potential theory, as well as, for example, in kernel methods for high-dimensional statistics and machine learning. Owing to their ubiquity and practical importance, a wide variety of techniques for exploiting this structure have been developed, appearing in the literature under an assortment of related names such as the fast multipole method (FMM) (Greengard & Rokhlin, 1987), $\mathcal{H}$- and $\mathcal{H}^2$-matrices (Hackbusch, 1999), hierarchically semiseparable matrices (Xia, Chandrasekaran, Gu, & Li, 2010), hierarchically block separable matrices (Gillman, Young, & Martinsson, 2012), and hierarchical off-diagonal low-rank matrices (Ambikasaran & Darve, 2013). Many of these now also have accompanying open-source software packages: `FMMLIB` (Gimbutas & Greengard, 2015), `H2Lib` (http://www.h2lib.org), `STRUMPACK` (Ghysels, Li, Rouet, Williams, & Napov, 2016; Rouet, Li, Ghysels, & Napov, 2016), `HODLRlib` (Ambikasaran, Singh, & Sankaran, 2019), `hm-toolbox` (Massei, Robol, & Kressner, 2020), etc.

`FLAM` is a MATLAB (and Octave-compatible) library in this same vein, implementing methods mostly in the "recursive skeletonization" style. Briefly, the core algorithms take as input a matrix implicitly defined by a function to generate arbitrary entries and whose rows/columns are associated with points in $\mathbb{R}^d$ (while we nominally target $d \leq 3$, there is no such explicit limitation). The induced geometry exposes the low-rank matrix blocks, which are then compressed or sparsified using the interpolative decomposition (ID) in a multilevel manner. Being written in MATLAB, the code is quite readable and easy to extend, especially for research purposes, though it may not be the most performant. Still, it is reasonably feature-complete; currently provided algorithms include, for dense IE-like matrices with dimensions $M \geq N$:

- `ifmm`: ID-based kernel-independent FMM for $O(M+N)$ multiplication (Martinsson & Rokhlin, 2007);
- `rskel`: recursive skeletonization for inversion (Ho & Greengard, 2012) and least squares (Ho & Greengard, 2014) via extended sparse embedding; typical complexity of $O(M+N)$ for $d=1$ and $O(M+N^{3(1-1/d)})$ for $d>1$;
- `rskelf`: recursive skeletonization factorization (RSF) (Ho & Ying, 2016a) for generalized LU/Cholesky decomposition; same complexity as above but restricted to $M=N$; allows multiply/solve with matrix or Cholesky factors, determinant computation, selected inversion, etc.; and
- `hifie`: hierarchical interpolative factorization (HIF) for IEs (Ho & Ying, 2016a); like RSF but with quasilinear complexity for all d.

Similarly, for sparse DE-like matrices, we have:

- `mf`: multifrontal factorization (MF); basically the sparse equivalent of RSF; and
- `hifde`: HIF for DEs (Ho & Ying, 2016b); like MF but with quasilinear complexity for all d.

Most of these have previously been published though some are perhaps new (if but straightforward modifications or extensions of existing ones).

Each algorithm comes with extensive tests demonstrating its usage and performance. For instance, `rskelf/tests/ie_circle.m` solves a second-kind Laplace boundary IE on the unit circle. At default settings, the problem is characterized by a *fully dense* square matrix of size $N = 16384$, which takes about 60 s to factor using MATLAB's `lu` at a storage cost of 2 GB. In contrast, `rskelf` requires just 0.7 s to compute an LU-like approximation to 10^{-12} precision, a speedup of about $80\times$; once computed, the factorization needs only 9 MB to store and can be used to execute solves in 0.04 s. Another exemplary test case is `hifde/tests/fd_square2.m`, which considers a Poisson DE on a very rough and high-contrast background field. Standard preconditioners such as `ichol` do not deal well with such severe ill-conditioning, but `hifde` remains highly effective, producing preconditioners that converge in just a handful of iterations. For more detailed performance analyses, we refer the reader to the cited literature above.

The provided functionality is somewhat similar to that offered by other software packages, but `FLAM` implements certain advanced methods like `hifie` and `hifde`, which to our knowledge are not publicly available elsewhere (though a close relative of `hifde` can be found in `STRUMPACK`). Furthermore, `FLAM` leverages a different core framework that we believe is considerably simpler and particularly well-suited to MATLAB's concise and readable style.

`FLAM` was originally created to support the computations in Ho & Ying (2016a) and Ho & Ying (2016b), and has since been used in Liu (2016), Wang (2016), Corona, Rahimian, & Zorin (2017), Fang, Ho, Ristroph, & Shelley (2017), Jiang, Rachh, & Xiang (2017), V. Minden et al. (2017a), Fan, An, & Ying (2019), Li, Xu, & Shao (2019), Tian & Engquist (2019), Wang, Jiang, & Wang (2019), and Askham & Rachh (2020). It has also served as the starting point for various more sophisticated codes such as those described in Minden, Damle, Ho, & Ying (2016), Li & Ying (2017), V. L. Minden (2017), V. Minden et al. (2017b), and Feliu-Fabà, Ho, & Ying (2020).

We hope that `FLAM` will be a valuable research tool for the broader scientific community to explore the use of fast matrix methods in their applications of interest as well as to prototype new algorithmic ideas and implementations.

`FLAM` is released under the GPLv3 license and can be accessed at klho.github.io/FLAM.

Acknowledgements

We thank Lexing Ying, Victor Minden, and Anil Damle for many fruitful discussions. This work was originally initiated while the author was at Stanford University under the partial support of the National Science Foundation (grant DMS-1203554).

References

Ambikasaran, S., & Darve, E. (2013). An $\mathcal{O}(N \log N)$ fast direct solver for partial hierarchically semi-separable matrices. *J. Sci. Comput.*, *57*(3), 477–501. doi:10.1007/s10915-013-9714-z

Ambikasaran, S., Singh, K. R., & Sankaran, S. S. (2019). HODLRlib: a library for hierarchical matrices. *J. Open Source Softw.*, *4*(34), 1167. doi:10.21105/joss.01167

Askham, T., & Rachh, M. (2020). A boundary integral equation approach to computing eigenvalues of the Stokes operator. *Adv. Comput. Math.*, *46*(2), 20. doi:10.1007/s10444-020-09774-2

Corona, E., Rahimian, A., & Zorin, D. (2017). A tensor-train accelerated solver for integral equations in complex geometries. *J. Comput. Phys.*, *334*, 145–169. doi:10.1016/j.jcp.2016.12.051

Fan, Y., An, J., & Ying, L. (2019). Fast algorithms for integral formulations of steady-state radiative transfer equation. *J. Comput. Phys.*, *380*, 191–211. doi:10.1016/j.jcp.2018.12.014

Fang, F., Ho, K. L., Ristroph, L., & Shelley, M. J. (2017). A computational model of the flight dynamics and aerodynamics of a jellyfish-like flying machine. *J. Fluid Mech.*, *819*, 621–655. doi:10.1017/jfm.2017.150

Feliu-Fabà, J., Ho, K. L., & Ying, L. (2020). Recursively preconditioned hierarchical interpolative factorization for elliptic partial differential equations. *Commun. Math. Sci.*, *18*(1), 91–108. doi:10.4310/CMS.2020.v18.n1.a4

Ghysels, P., Li, X. S., Rouet, F.-H., Williams, S., & Napov, A. (2016). An efficient multicore implementation of a novel HSS-structured multifrontal solver using randomized sampling. *SIAM J. Sci. Comput.*, *38*(5), S358–S384. doi:10.1137/15M1010117

Gillman, A., Young, P. M., & Martinsson, P.-G. (2012). A direct solver with $O(N)$ complexity for integral equations on one-dimensional domains. *Front. Math. China*, *7*(2), 217–247. doi:10.1007/s11464-012-0188-3

Gimbutas, Z., & Greengard, L. (2015). Computational software: Simple FMM libraries for electrostatics, slow viscous flow, and frequency-domain wave propagation. *Commun. Comput. Phys*, *18*(2), 516–528. doi:10.4208/cicp.150215.260615sw

Greengard, L., & Rokhlin, V. (1987). A fast algorithm for particle simulations. *J. Comput. Phys.*, *73*(2), 325–348. doi:10.1016/0021-9991(87)90140-9

Hackbusch, W. (1999). A sparse matrix arithmetic based on $\mathcal{H}$-matrices. Part I: Introduction to $\mathcal{H}$-matrices. *Computing*, *62*(2), 89–108. doi:10.1007/s006070050015

Ho, K. L., & Greengard, L. (2012). A fast direct solver for structured linear systems by recursive skeletonization. *SIAM J. Sci. Comput.*, *34*(5), A2507–2532. doi:10.1137/120866683

Ho, K. L., & Greengard, L. (2014). A fast semidirect least squares algorithm for hierarchically block separable matrices. *SIAM J. Matrix Anal. Appl.*, *35*(2), 725–748. doi:10.1137/120902677

Ho, K. L., & Ying, L. (2016a). Hierarchical interpolative factorization for elliptic operators: integral equations. *Comm. Pure Appl. Math*, *69*(7), 1314–1353. doi:10.1002/cpa.21577

Ho, K. L., & Ying, L. (2016b). Hierarchical interpolative factorization for elliptic operators: differential equations. *Comm. Pure Appl. Math.*, *69*(8), 1415–1451. doi:10.1002/cpa.21582

Jiang, S., Rachh, M., & Xiang, Y. (2017). An efficient high order method for dislocation climb in two dimensions. *Multiscale Model. Simul.*, *15*(1), 235–253. doi:10.1137/16M1081920

Li, W., Xu, S., & Shao, M. (2019). Simulation of two-dimensional steady-state heat conduction problems by a fast singular boundary method. *Eng. Anal. Bound. Elem.*, *108*, 149–157. doi:10.1016/j.enganabound.2019.06.020

Li, Y., & Ying, L. (2017). Distributed-memory hierarchical interpolative factorization. *Res. Math. Sci.*, *4*(1), 12. doi:10.1186/s40687-017-0100-6

Liu, Y. (2016). *The numerical solution of frequency-domain acoustic and electromagnetic periodic scattering problems* (PhD thesis). Dartmouth College.

Martinsson, P. G., & Rokhlin, V. (2007). An accelerated kernel-independent fast multipole method in one dimension. *SIAM J. Sci. Comput.*, *29*(3), 1160–1178. doi:10.1137/060662253

Massei, S., Robol, L., & Kressner, D. (2020). hm-toolbox: Matlab software for HODLR and HSS matrices. *SIAM J. Sci. Comput.*, *42*(2), C43–C68. doi:10.1137/19M1288048

Minden, V., Damle, A., Ho, K. L., & Ying, L. (2016). A technique for updating hierarchical skeletonization-based factorizations of integral operators. *Multiscale Model. Simul.*, *14*(1), 42–64. doi:10.1137/15M1024500

Minden, V., Damle, A., Ho, K. L., & Ying, L. (2017a). Fast spatial Gaussian process maximum likelihood estimation via skeletonization factorizations. *Multiscale Model. Simul.*, *15*(4), 1584–1611. doi:10.1137/17M1116477

Minden, V., Ho, K. L., Damle, A., & Ying, L. (2017b). A recursive skeletonization factorization based on strong admissibility. *Multiscale Model. Simul.*, *15*(2), 768–796. doi:10.1137/16M1095949

Minden, V. L. (2017). *Data-sparse algorithms for structured matrices* (PhD thesis). Stanford University.

Rouet, F.-H., Li, X. S., Ghysels, P., & Napov, A. (2016). A distributed-memory package for dense hierarchically semi-separable matrix computations using randomization. *ACM Trans. Math. Softw.*, *42*(4), 27. doi:10.1145/2930660

Tian, X., & Engquist, B. (2019). Fast algorithm for computing nonlocal operators with finite interaction distance. *Commun. Math. Sci.*, *17*(6), 1653–1670. doi:10.4310/CMS.2019.v17.n6.a7

Wang, S. (2016). *Efficient high-order integral equation methods for the heat equation* (PhD thesis). New Jersey Institute of Technology.

Wang, S., Jiang, S., & Wang, J. (2019). Fast high-order integral equation methods for solving boundary value problems of two dimensional heat equation in complex geometry. *J. Sci. Comput.*, *79*(2), 787–808. doi:10.1007/s10915-018-0872-x

Xia, J., Chandrasekaran, S., Gu, M., & Li, X. S. (2010). Fast algorithms for hierarchically semiseparable matrices. *Numer. Linear Algebra Appl.*, *17*(6), 953–976. doi:10.1002/nla.691