Journal of Computational Mathematics
Vol.41, No.6, 2023, 1093–1116.

http://www.global-sci.org/jcm
doi:10.4208/jcm.2202-m2021-0324

A FAST FREE MEMORY METHOD FOR AN EFFICIENT COMPUTATION OF CONVOLUTION KERNELS*

Matthieu Aussal and Marc Bakry[1)]
Ecole Polytechnique (CMAP), INRIA, Institut Polytechnique Paris, Route de Saclay 91128, Palaiseau, France
Email: matthieu.aussal@polytechnique.edu, marc.bakry@polytechnique.edu

Abstract

We introduce the Fast Free Memory method (FFM), a new implementation of the Fast Multipole Method (FMM) for the evaluation of convolution products. The FFM aims at being easier to implement while maintaining a high level of performance, capable of handling industrially-sized problems. The FFM avoids the implementation of a recursive tree and is a kernel independent algorithm. We give the algorithm and the relevant complexity estimates. The quasi-linear complexity enables the evaluation of convolution products with up to one billion entries. We illustrate numerically the capacities of the FFM by solving Boundary Integral Equations problems featuring dozen of millions of unknowns. Our implementation is made freely available under the GPL 3.0 license within the Gypsilab framework.

Mathematics subject classification: 65T50, 65Z05, 65R20.
Key words: Convolution product, Fast multipole method, Boundary integral equations, Open-source.

1. Introduction

The numerical computation of convolution products is a crucial issue arising in many domains like the filtering, the computation of boundary integral operators, optimal control, etc. In a continuous framework, a convolution product is of the form

$$v(x) = \int_\Omega G(x,y)\, u(y)\, d\Omega_y, \tag{1.1}$$

where Ω is some domain of integration in $\mathbb{R}^d, d \in \mathbb{N}^\star$, u some function. The bivariate function $G(.,.)$ is some convolution kernel of the form

$$G(x,y) = G(x-y, |x-y|), \tag{1.2}$$

where $|\,.\,|$ is the euclidean distance. Of course, Eq. (1.1) does not admit an analytical expression in the general case and the integral is computed numerically using, for instance, a quadrature rule. Assuming we want to evaluate v on a finite set of nodes $X = (x_i)_{i\in[\![1,N_X]\!]}$, we have

$$v(x_i) \approx \sum_{j=1}^{N_Y} \omega_j\, G(x_i,y_j)\, u(y_j), \tag{1.3}$$

* Received November 16, 2021 / Accepted February 16, 2022 / Published online July 26, 2022 /
1) Corresponding author

where $(\omega_j)_{j\in[\![1,N_Y]\!]}$ and $Y = (y_j)_{j\in[\![1,N_Y]\!]}$ are respectively the weights and nodes of such a quadrature. The *discrete convolution* may be recast as a simple matrix-vector product

$$\mathbf{v} = \mathbf{G}\cdot\mathbf{W}\cdot\mathbf{u}, \tag{1.4}$$

where $\mathbf{u} = (u_j)_j = (u(y_j))_j$, $\mathbf{v} = (v_i)_i = (v(x_i))_i$, $\mathbf{G} = G(x_i,y_j))_{\{i,j\}}$ and $\mathbf{W} = \mathrm{diag}((\omega_j)_j)$ (we omit $\mathbf{W}$ in the following). Obviously, the matrix $\mathbf{G}$ is *dense*. Therefore, the memory footprint and computational cost grow quadratically. The computation of (1.4) is constrained to smaller problems ($N_X, N_Y \approx$ a few thousand) on personal computers and smaller servers, and to $N_X, N_Y \approx 10^6$ for industrial servers.

The current approach is to perform the computation approximately up to a given tolerance ε (accuracy). In the past thirty years, multiple so-called acceleration methods have been proposed. The entries of $\mathbf{G}$ can be seen as the description of an interaction between a source set of nodes Y and a target set of nodes X. Thus all blocks of $\mathbf{G}$ describe the interaction between a source subset of Y and a target subset of X. Theses interactions may be compressible, i.e. it admits a low-rank representation. This is the case, for example, when two subsets are far enough following an admissibility criterion. The methods mentioned in the following propose different alternatives on the way the interactions are characterized and computed. The standard way is probably the Fast Multipole Method (FMM) developed by L. Greengard and V. Rokhlin (see [13]), initially introduced for the computation of the gravitational potential of a cloud of particles. Later versions feature the support of oscillatory kernels like the Helmholtz Green kernel. One major drawback is that the implementations are mostly kernel-specific despite recent advances in the domain. We refer to [14] for more details. In 1999, a new approach named Hierarchical matrices ($\mathcal{H}$-matrices) was introduced by S. Börm, L. Grasedyck and W. Hackbusch. This method is based on the representation of the matrix by a quadtree whose leaves are low-rank or full-rank submatrices. A strong advantage in favor of hierarchical matrices is that a complete algebra has been created: addition, multiplication, LU-decomposition, etc. Unfortunately, $\mathcal{H}$-matrices become less effective for strongly oscillating kernels because the rank of the compressible blocks increases with the frequency of the oscillations. For more details and a complete mathematical analysis, we refer to [1,6]. A recent compression method is the Sparse Cardinal Sine Decomposition (SCSD) proposed by F. Alouges and M. Aussal in 2015 [15]. It is based on a representation of the Green kernel in the Fourier domain using the integral representation of the cardinal sine function. One major advantage is that the matrix-vector product is performed without partitioning of the space. However, there is no corresponding algebra. More recently, we find skeletonization techniques like the Hierarchical Interpolative Factorization [30]. All the aforementioned methods aim at having a quasi-linear memory footprint and computational complexity.

In this paper, we present a simplified implementation of the well-known FMM algorithm which we call the Fast Free Memory method (FFM). The FFM aims at being easier to implement and enabling a high level of versatility without compromising too much on the performance. The purpose of this method is not to compete with the high-performance industrial FMM but rather to provide a good and well-performing academic tool still capable of dealing with very large problems with dozen of millions of entries. Unlike the FMM, there are no communications between levels of the octree (there are no L2L or M2M operators): only a downward "implicit" tree traversal is performed. It makes use of the Non Uniform Fast Fourier Transform algorithm when the convolution kernel is oscillating, and the Lagrange interpolation in the other case. Consequently, it is easy to support a new kernel in an already-existing implementation. The