Sparse Bessel Decomposition for fast 2D non-uniform convolution

Martin Averseng

September 6, 2017

Abstract

We describe an efficient algorithm for computing matrix vector products involved in the numerical resolution of boundary integral equations in 2 dimensions of space. This work is an extension of the so-called Sparse Cardinal Sine Decomposition algorithm decribed in [1], which is the three-dimensional analogous.

1 Introduction

In most modern codes, the numerical resolution of boundary integral equations is performed by piecewise polynomial approximation and requires the resolution of a dense linear system $Au = b$. When the number of unknowns is large, it becomes prohibitively costly to invert the matrix A, and instead, the solution is found using Krylov subspace based methods. The latter requires very fast evaluations of matrix vector products, which, in the context of boundary integral formulations, take the form of discrete convolutions:

$$q_k = \sum_{l=1}^{N_x} G(|\boldsymbol{x}_k - \boldsymbol{x}_l|)\phi(\boldsymbol{x}_l), \quad k \in [1, N_x]. \tag{1.1}$$

Here, G is the Green's kernel of the partial derivative equation being solved, ϕ is some basis function of the discrete functional scheme, and $\boldsymbol{x} = (\boldsymbol{x}_k)_{k\in[1,N_x]}$ is any set of points in $\mathbb{R}^2$. For example, the resolution of the Laplace equation with Dirichlet boundary conditions leads to (1.1) with $G(\boldsymbol{x}) = \log|\boldsymbol{x}|$ (the kernel of the single layer potential).

Several efficient algorithms have emerged to compute a fast approximation of (1.1), among which the celebrated Fast Multipole Method [] (Lequel Citer ? Greengard 1997 ? Coifman ?), and the Hierarchical Matrix idem, à citer. Many of them are based on local variable separation

$$G(|\boldsymbol{u} - \boldsymbol{v}|) \approx \sum_k \lambda_k G_1^k(\boldsymbol{u}) G_2^k(\boldsymbol{v}),$$

valid for $\boldsymbol{u}$ and $\boldsymbol{v}$ distant from each other. A geometrical splitting (octree) of the mesh is performed to define regions in which the coefficients λ_k are constants. This eventually results in compressed matrix representations and matrix-vector products with quasi-linear complexity.

Here we present an alternative compression and acceleration method, which we call the Sparse Bessel Decomposition (SBD), an extension of the Sparse Cardinal Sine Decomposition (SCSD) decribed in [1] adapted to 2-dimensional problems. SBD and SCSD achieve performances almost comparable to the aforementioned algorithms, they are quite general with respect to the kernel G, and do not rely on the construction of octrees, which makes them easier to implement. In addition, they materialise in an elegant way the intuition according to which a discrete convolution should translate into a product of Fourier spectra.

The method mainly relies heavily on the Non-uniform Fast Fourier Transforms [2] (of type III), which is a fast procedure:

$$\mathrm{NuFFT}_{\pm}[\boldsymbol{x}, \boldsymbol{\xi}](\alpha)$$

that returns the vector q defined by:

$$q_k = \sum_{\nu=1}^{N_\xi} e^{\pm i \boldsymbol{\xi}_\nu \cdot \boldsymbol{x}_k} \alpha_\nu,$$

with an quasi-linear cost in both N_x and N_ξ for any (non equispaced) set of points $\boldsymbol{x}, \boldsymbol{\xi}$ in $\mathbb{R}^2$ and complex vector α. The SBD method first seeks a discrete and sparse approximation of the spectrum of G,

$$G(x) \approx \sum_{\nu=1}^{N_\xi} e^{i\boldsymbol{x}_k \cdot \boldsymbol{\xi}_\nu} \hat{\omega}_\nu, \tag{1.2}$$

Such a representation is computed "offline" and can be recycled for any choice of function ϕ in (1.1). The expansion (1.2) is then injected in (1.1) to yield

$$\begin{aligned} q_{\text{far}} &\approx \left(\sum_{\nu=1}^{N_\xi} e^{+i\boldsymbol{x}_k \cdot \boldsymbol{\xi}_\nu} \left[\hat{\omega}_\nu \sum_{l=1}^{N_x} e^{-i\boldsymbol{x}_l \cdot \boldsymbol{\xi}_\nu} \phi(\boldsymbol{x}_l) \right] \right)_{k \in [1, N_x]} \\ &= \text{NuFFT}_+[\boldsymbol{x}, \boldsymbol{\xi}] \left(\hat{\omega} \odot \text{NuFFT}_-[\boldsymbol{x}, \boldsymbol{\xi}](\phi(\boldsymbol{x})) \right). \end{aligned} \tag{1.3}$$

Here, $\odot$ denotes elementwise product between vectors. This approximation reduces the complexity from $O(N_x^2)$ to $O(M \log^a(M))$ for some integer a and where M is the greatest value among N_x and N_ξ.

In most cases, the kernel G is singular near the origin, so (1.2) will be only valid for $|x|$ above some threshold $\delta_{\min}$. Part of the SBD algorithm is thus dedicated to computing a local correction using a sparse matrix product to account for the exact close interactions. The threshold must be chosen so that the far and close contributions of the product (that is the time spent computing the NuFFT and the sparse matrix product respectively) be balanced. We denote by $a \in (0, 1)$ the quotient $\delta_{\min}/\delta_{\max}$, where $\delta_{\max}$ is the diameter of the cloud of points $\boldsymbol{x}$.

The object of this work is to describe precisely the algorithm, provide a Matlab (à Mettre en forme) code and estimate the complexity in the special case where the singular behavior of G is $O(\log(|\boldsymbol{x}|))$, which will be achieved by giving an asymptotic bound on N_ξ in (1.2). The complexity is summarized in the next theorem:

Theorem 1.1. *Let $\varepsilon > 0$ the desired accuracy of the method. For any constant $\lambda \geq 1$, if the parameter a is chosen as $a = \lambda \frac{\log(\varepsilon)^{2/3}}{N_x^{2/3}}$, then*

(i) The off-line complexity (for the computation of representation (1.2)) scales as

$$C_{\text{off}}(N_x, \varepsilon, \lambda) = O\left(|\log \varepsilon|^{3/4} \lambda^{-3} N_x^2 \right)$$

(ii) Once the off-line computations have been done, (1.1) is evaluated for any choice of ϕ at a precision at least $\varepsilon \sum_l |\phi(x_l)|$ with a complexity scaling as

$$C_{\text{on}}(N_x, \varepsilon, \lambda) = O\left(|\log \varepsilon|^{3/4} C_{\text{NuFFT}}(\varepsilon) \lambda N_x^{4/3} \log(N_x) \right)$$

where $C_{\text{NuFFT}}(\varepsilon)$ represents the complexity factor of the Type III NuFFT *related to the target tolerance.*

The choice $\lambda = 1$ in the previous theorem yields the minimal complexity for on-line computations. This is the appropriate choice when (1.1) needs to be evaluated a lot of times for different ϕ. In the case where (1.1) only has to be computed a few times, λ can be chosen so as to minimize the total computation time (off-line + on-line), that is $\lambda = \frac{N_x^{1/6}}{C_{\text{NuFFT}(\varepsilon)}^{1/4}}$, yielding a complexity of $O\left(|\log(\varepsilon)|^{3/4} C_{\text{NuFFT}}(\varepsilon)^{3/4} N_x^{3/2} \log(N_x) \right)$.

References

[1] François Alouges and Matthieu Aussal. The sparse cardinal sine decomposition and its application for fast numerical convolution. *Numerical Algorithms*, 70(2):427–448, 2015.

[2] A. Dutt and V. Rokhlin. Fast fourier transforms for nonequispaced data. *SIAM J. Sci. Comput.*, 14(6):1368–1393, November 1993.