FAST ALGORITHM FOR COMPUTING NONLOCAL OPERATORS WITH FINITE INTERACTION DISTANCE[*]

XIAOCHUAN TIAN[†] AND BJÖRN ENGQUIST[‡]

Abstract. Developments of nonlocal operators for modeling processes that traditionally have been described by local differential operators have been increasingly active during the last few years. One example is peridynamics for brittle materials and another is nonstandard diffusion including the use of fractional derivatives. A major obstacle for application of these methods is the high computational cost from the numerical implementation of the nonlocal operators. It is natural to consider fast methods of fast multipole or hierarchical matrix type to overcome this challenge. Unfortunately the relevant kernels do not satisfy the standard necessary conditions. In this work a new class of fast algorithms is developed and analyzed, which in some cases reduces the computational complexity of applying nonlocal operators to essentially the same order of magnitude as the complexity of standard local numerical methods.

Keywords. Nonlocal operator; fast algorithm; nonlocal diffusion; peridynamics; heterogeneous material; fast multipole method; finite interaction.

AMS subject classifications. 65R20; 82C21; 65F05; 45K05.

1. Introduction

Nonlocal operators as alternatives to the local differential operators have been widely applied to the modeling of a number of physical and biological processes over the last few years. One example is given by the peridynamics models for brittle materials [3, 10, 14, 15, 25, 26], and another is nonstandard diffusion including the use of fractional derivatives [1, 4, 6, 9, 20, 21]. Moreover, even in the case of solving differential equations, nonlocal integral relaxations to local differentiations are sometimes introduced as a numerical technique, see [11, 19, 22, 23].

While the nonlocal models can provide more accurate descriptions of a physical system, the nonlocality also increases the computational cost compared to classical local models based on partial differential equations (PDEs). Thus it is imperative to develop fast algorithms for the computation of nonlocal models. In this work, we focus on nonlocal models with finite nonlocal interaction distances (see [28]). Each point in the domain interacts with points within certain "horizon" of it. That is to say the interaction kernel is compactly supported in certain finite region and may also be singular at zero distance. When the interaction region is spherical, we call the radius of the region the horizon radius or just horizon in short. Such kernels make most fast algorithms, that utilize the low rank property of the far field interactions, fail to work, because the kernel is typically not regular around the boundary of its support. In the work of [31, 33], FFT-based fast algorithms are developed for solving peridynamic models, in which Toeplitz structures of matrices are greatly exploited. However, such methods fail to work for the more general cases of inhomogeneuous media and variable horizons resulting in non-translation invariant kernels, which is the concern of this work.

The peridynamics (PD) theory of solid mechanics takes into account large scale deformations and reformulates the internal force of a deformed body by introducing the

[†]Department of Mathematics, University of Texas at Austin, Austin, TX 78712, USA (xtian@math.utexas.edu).

[‡]Department of Mathematics and the Oden Institute, University of Texas at Austin, Austin, TX 78712, USA (engquist@ices.utexas.edu).

nonlocal way of interactions of materials points. Let $\mathbf{u}$ denote the displacement field, the force balance equation in the bond-based peridynamic theory is given by

$$-\int_{\mathcal{H}_{\boldsymbol{x}}} \mathbf{f}(\mathbf{u}(\boldsymbol{y}) - \mathbf{u}(\boldsymbol{x}), \boldsymbol{y}, \boldsymbol{x}) d\boldsymbol{y} = \mathbf{b}(\boldsymbol{x}), \tag{1.1}$$

where the integration is limited to the finite region $\mathcal{H}_{\boldsymbol{x}}$. $\mathbf{b} = \mathbf{b}(\boldsymbol{x})$ denotes the body force and $\mathbf{f}$ is the force density function, which contains all the constitutive properties. For a prototype micro-elastic material, the force vector $\mathbf{f}$ points to the direction of the deformed bond with its magnitude being a function of the bond relative elongation (stretch). In general, $\mathbf{f}$ depends nonlinearly on the bond relative elongation, which gives the model the flexibility to describe changes of material properties. In this work, we will only consider the linear model obtained from the assumption of small deformation, namely,

$$\mathbf{f}(\mathbf{u}(\boldsymbol{y}) - \mathbf{u}(\boldsymbol{x}), \boldsymbol{y}, \boldsymbol{x}) = \omega(\boldsymbol{x}, \boldsymbol{y})[\mathbf{e} \cdot (\mathbf{u}(\boldsymbol{y}) - \mathbf{u}(\boldsymbol{x}))]\mathbf{e}, \tag{1.2}$$

where $\mathbf{e}$ is the unit vector in the bond direction $\boldsymbol{y} - \boldsymbol{x}$, and $\omega(\boldsymbol{x}, \boldsymbol{y})$ is the micromodulus function satisfying

$$\int_{\mathcal{H}_{\boldsymbol{x}}} |\boldsymbol{y} - \boldsymbol{x}|^2 \omega(\boldsymbol{x}, \boldsymbol{y}) d\boldsymbol{y} < \infty.$$

For each $\boldsymbol{x}$, $\omega(\boldsymbol{x}, \cdot)$ is a function with support on $\mathcal{H}_{\boldsymbol{x}}$. In practice, this is achieved by choosing a fractional-type kernel multiplied with the characteristic function or the conical function on $\mathcal{H}_{\boldsymbol{x}}$ (see [14]), where both choices create non-smooth transition to zero at the boundary $\partial\mathcal{H}_{\boldsymbol{x}}$.

As we can see, the discretization of the nonlocal operator in (1.1) results in a matrix A with high density, for which fast algorithms must be considered in order to lower the cost of matrix multiplication and inversion. There are several classical algorithms for the treatment of non-sparse matrices, including the fast multipole methods (FMMs) [12, 13, 32] and the hierarchical matrix techniques [16, 17]. More recent developments can be found in [18]. We will first consider the existing fast solvers to see whether they are effective in the computation of nonlocal models like (1.1). In Section 2 we will show that the performance of the fast solvers depends largely on the regularity of the kernel across the boundary $\partial\mathcal{H}_{\boldsymbol{x}}$. This sends the clear message of the preference of the far field behavior of the kernel function in terms of computation efficiency. Popular choices of kernels in the peridynamics simulation will cause serious problems in the interest of fast computation. We then propose in Section 3 to split the two types of singularities – one at origin and the other at the boundary $\partial\mathcal{H}_{\boldsymbol{x}}$– of the kernel function, and treat them separately. Once the kernel is decomposed into two, one that is smooth away from origin and the other that is smooth away from the truncation, we then use different fast algorithms that exploit the different types of smoothness of the two kernels. The classical fast algorithms such as FMMs and hierarchical matrix techniques only resolve the first type of singularity, namely that the singularity set is one point (the origin of the kernel in most cases). Notice that the second type of singularity is on the boundary set $\partial\mathcal{H}_{\boldsymbol{x}}$, and it is a set of codimension 1. In 2D the sets of singularity are boundary curves, and in 3D they are boundary surfaces. Our algorithm that deals with second type of singularity is basically a new FMM-type method for kernels that exhibit singularities on codimension 1 sets. There is a reduction of computational complexity in all dimensions with optimal rate $O(N \log N)$ for N unknowns in 1D. Other potential applications of this work include computing with the retarded potentials raised in time domain boundary integral equations [24], where the potentials are discontinuous functions defined in space-time.

2. The role of far field behaviors of kernels

We will use the nonlocal diffusion operator to illustrate our methodology in this section. The nonlocal diffusion operator is given by

$$\mathcal{L}u(\boldsymbol{x}) = \int_{\mathcal{H}_{\boldsymbol{x}}} \omega(\boldsymbol{x}, \boldsymbol{y})(u(\boldsymbol{y}) - u(\boldsymbol{x}))d\boldsymbol{y}. \tag{2.1}$$

See more discussions of details in [6]. Here we assume that the kernel ω is given by

$$\omega(\boldsymbol{x}, \boldsymbol{y}) = \frac{C(\boldsymbol{x}, \boldsymbol{y})}{\delta(\boldsymbol{x}, \boldsymbol{y})^{d+2}} \gamma\left(\frac{|\boldsymbol{y} - \boldsymbol{x}|}{\delta(\boldsymbol{x}, \boldsymbol{y})}\right), \tag{2.2}$$

where the kernel γ is a nonnegative function supported on $[-1, 1]$ with $\int_{-1}^{1} s^2 \gamma(|s|)ds = 1$. $C(\boldsymbol{x}, \boldsymbol{y})$ is the diffusion coefficient and $\delta(\boldsymbol{x}, \boldsymbol{y})$ is the horizon function that satisfy

$$C_0 \leq C(\boldsymbol{x}, \boldsymbol{y}) \leq C_1, \text{ and } 0 < \delta(\boldsymbol{x}, \boldsymbol{y}) \leq \delta_1. \tag{2.3}$$

Here we keep the dependence of C and δ on $\boldsymbol{x}$, $\boldsymbol{y}$ so that it could model the inhomogeneous medium, see related works [7, 27, 30].

The common practice in the peridynamics simulation (see [14]) takes γ to be

$$\gamma(|s|) = \frac{c_1}{|s|}\chi(|s| < 1), \text{ or } \gamma(|s|) = \frac{c_2}{|s|}(1 - |s|)\chi(|s| < 1). \tag{2.4}$$

We distinguish between the two cases:

(a) If C and δ only depend on x, namely $C(\boldsymbol{x}, \boldsymbol{y}) = C(\boldsymbol{x})$ and $\delta(\boldsymbol{x}, \boldsymbol{y}) = \delta(\boldsymbol{x})$, then (2.1) is the nonlocal diffusion operator of non-divergence type. As $\delta_1 \to 0$, we have

$$\mathcal{L}u(\boldsymbol{x}) \to C(\boldsymbol{x})\Delta u(\boldsymbol{x}).$$

(b) If C and δ are symmetric, namely $C(\boldsymbol{x}, \boldsymbol{y}) = C(\boldsymbol{y}, \boldsymbol{x})$ and $\delta(\boldsymbol{x}, \boldsymbol{y}) = \delta(\boldsymbol{y}, \boldsymbol{x})$, then (2.1) is the nonlocal diffusion operator of divergence type. As $\delta_1 \to 0$, we have

$$\mathcal{L}u(\boldsymbol{x}) \to \nabla(\sigma(\boldsymbol{x})\nabla u(\boldsymbol{x})), \text{ where } \sigma(\boldsymbol{x}) = C(\boldsymbol{x}, \boldsymbol{x}).$$

The FFT-based methods mentioned earlier are restrictive in the application to heterogeneous media. Here we seek algorithms that can be applied to the more general cases given by (2.2). We use the recently proposed hierarchical interpolative factorization (HIF) [18] method to factorize the dense matrix A obtained from discretizing the nonlocal operator $\mathcal{L}$. The factorization of the matrix can be used to rapidly apply both A and A^{-1}. Therefore it can serve as a direct solver or be stored and reused for iterative methods or time-dependent problems. The HIF takes advantage of the low-rank behavior of the off-diagonal entries of the dense matrix, so the far field behavior of the kernel ω is the central factor in the performance of the algorithm. We will illustrate in this section that the performance of the HIF algorithm improves greatly as the smoothness of the kernel away from zero enhances. All computations are performed in MATLAB R2014b on a single core of a 1.1GHz Intel Core M CPU on a 64-bit Mac laptop.

2.1. Nonlocal diffusion in homogeneous media. In this section, we solve the nonlocal problem on the interval $\Omega = (0, 1)$ with $C(x, y) \equiv 1$ and $\delta(x, y) \equiv \delta$ with δ being the horizon radius of the homogeneous medium. The matrix A is obtained from the discretization of the nonlocal operator $\mathcal{L}$, which is based on the asymptotically compatible schemes in [29], which ensures uniform discretization error independent of the

horizon function δ. In Table 2.1, the storage and the time of matrix-vector multiplication are listed for the original matrix A as well as its compression after using the HIF method. The original matrix A is stored using the sparse matrix representation with the MATLAB built-in matrix-vector multiplication applied to the test. The kernel $\gamma(s)$ has $O(1/|s|)$ singularity at 0 and is assumed to be smooth expect at origin and ± 1 (ω has singularity at 0 and $\pm\delta$). The number of discretization nodes N is fixed to be 2048 in Table 2.1. The relative precision parameter for the HIF method is set to be $1E-8$.

kernel	δ	HIF		Original	
		Memory (MB)	$T_{\mathrm{multi}}(\sec)$	Memory (MB)	$T_{\mathrm{multi}}(\sec)$
C^{-1}	1	1.93	$3.54E-3$	67.13	$8.49E-3$
C^0	1	1.93	$3.52E-3$	67.13	$8.61E-3$
C^1	1	1.93	$3.53E-3$	67.13	$8.21E-3$
C^2	1	1.93	$3.43E-3$	67.13	$8.21E-3$
C^3	1	1.93	$3.38E-3$	67.13	$8.25E-3$
C^{-1}	1/4	33.58	$6.23E-3$	29.40	$3.34E-3$
C^0	1/4	33.37	$6.34E-3$	29.40	$3.36E-3$
C^1	1/4	4.44	$4.34E-3$	29.40	$3.39E-3$
C^2	1/4	2.10	$3.61E-3$	29.40	$3.43E-3$
C^3	1/4	1.99	$3.43E-3$	29.40	$3.41E-3$
C^{-1}	h	2.11	$3.30E-3$	0.11	$5.04E-5$
C^0	h	2.11	$3.21E-3$	0.11	$4.57E-5$
C^1	h	2.11	$3.27E-3$	0.11	$4.57E-5$
C^2	h	2.11	$3.21E-3$	0.11	$4.76E-5$
C^3	h	2.11	$3.31E-3$	0.11	$4.66E-5$

TABLE 2.1. *The storage and time of matrix-vector multiplication for the 1D nonlocal diffusion in homogeneous medium. The domain $\Omega=(0,1)$ is discretized using $N=2048$ points. The mesh size $h=1/2048$. The relative precision of HIF approximation is set to be $\epsilon=10^{-8}$.*

The left column of Table 2.1 lists the different types of regularity (at ± 1) of the kernel $\gamma(s)$ and the different choices of the horizon parameter δ. The discontinuous kernel (denoted as C^{-1}) is the one that is most often used in peridynamics simulations and is given by

$$\gamma(|s|) = \frac{1}{|s|}\chi(|s|<1).$$

The more regular C^k $(k=0,1,2,3)$ kernels are obtained by subtracting γ with polynomials p^{2k} $(k=0,1,2,3)$ of degree $2k$. The details of constructing these polynomials will be given in Section 3. When $\delta \geq 1$, the matrix A is fully dense and it does not see the tail of the kernel. As a result, the HIF works well for all the types of kernels. When δ gets smaller, the regularity of the kernel $\gamma(s)$ at ± 1 plays a decisive role on the performance of the HIF. We see that when $\gamma(s)$ is discontinuous at ± 1, no compression of the matrix A is observed using the HIF. The storage taken and computation time decrease with increasing regularity of $\gamma(s)$ from discontinuous (C^{-1}) all the way to three times continuously differentiable (C^3). In the case of $\delta=1/4$, the critical improvement of the computational efficiency happens between C^0 kernel functions and C^1 kernel functions as shown in Table 2.1. We remark that the location of critical regularity changes with

the precision parameter ϵ. When the HIF algorithm runs with higher precision, higher critical regularity is observed; and when the HIF algorithm runs with lower precision, lower critical regularity is observed. Similar patterns are also observed in the experiments that will be discussed in Sections 2.2 and 2.3. When δ keeps decreasing, the effect of different regularity of the kernel $\gamma(s)$ will diminish. The extreme case is that δ equals the mesh size h when the matrix A is in fact tridiagonal. The case of $\delta = h$ is listed in Table 2.1 to compare the overhead of using the HIF algorithm with the sparse matrix. The overhead of using the HIF algorithm in this case is due to the fact that HIF does not build into itself the ability to detect the horizon and ignore the zeros outside. It will be a future work to explore this possibility depending on application.

To account for the complexity of the HIF, we provide the computation time for different discretization nodes N in Table 2.2. The horizon parameter δ in this table is fixed to be $1/4$. The scaling results for the choice of the C^0 kernel are compared with the choice of the C^3 kernel. The complexity of the matrix-vector multiplication for the original matrix A without compression is approximately $O(N^2)$ in both cases. On the other hand, the HIF code behaves quite differently for the C^0 kernel and the C^3 kernel. The observed complexity of HIF with the C^0 kernel is still $O(N^2)$, while the complexity of HIF with the C^3 kernel is just around $O(N)$.

The simulations show the effectiveness of the HIF fast algorithm with the improving far field regularity of nonlocal interaction kernels. Finally, we remark that the cost of constructing the factorization is in general larger than the cost of matrix-vector multiplication. This is acceptable since one only needs to construct the factorization once and reuse it for an iterative method or in time-dependent problems.

kernel	N	HIF		Original	
		Memory (MB)	T_{multi}(sec)	Memory (MB)	T_{multi}(sec)
C^0	512	2.10	$4.87E\!-\!4$	1.85	$1.68E\!-\!4$
C^0	1024	8.36	$1.56E\!-\!3$	7.36	$8.76E\!-\!4$
C^0	2048	33.35	$5.31E\!-\!3$	29.40	$2.96E\!-\!3$
C^0	4096	122.99	$2.11E\!-\!2$	117.52	$1.39E\!-\!2$
C^3	512	0.41	$1.30E\!-\!3$	1.85	$1.63E\!-\!4$
C^3	1024	1.04	$1.55E\!-\!3$	7.36	$8.95E\!-\!4$
C^3	2048	1.99	$3.07E\!-\!3$	29.40	$3.35E\!-\!3$
C^3	4096	3.93	$5.97E\!-\!3$	117.52	$1.48E\!-\!2$

TABLE 2.2. *The storage and time of matrix-vector multiplication for the 1D nonlocal diffusion in homogeneous medium. $\delta = 1/4$. The relative precision of HIF approximation is set to be $\epsilon = 10^{-8}$.*

2.2. Nonlocal diffusion in heterogeneous media.   We test the effectiveness of the HIF algorithm for nonlocal diffusion in heterogeneous media in this section. The kernel ω is defined as (2.2), where $C(x,y) \equiv 1$ and $\delta(x,y) = \delta(x)$. Each point x has its own horizon radius $\delta(x)$. In this case, $\mathcal{L}$ is a nonlocal diffusion operator of non-divergence type. The matrix A obtained from the discretization of the nonlocal operator $\mathcal{L}$ is now non-symmetric, in which case the FFT-based fast algorithm will not work. In Table 2.3, the storage and the time of matrix-vector multiplication are listed for the original matrix A as well as its compression after using the HIF algorithm.

In the numerical experiments we use $\delta(x) = \delta_0(1 + e^{-20(x-0.5)^2})$ for $x \in \Omega = (0,1)$ so

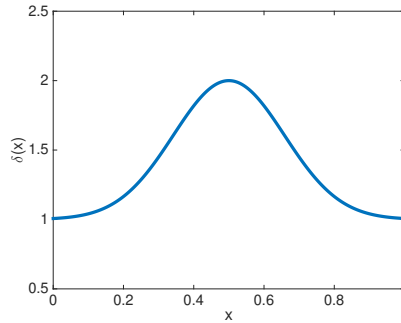that $\delta(x)$ is a function between δ_0 and $2\delta_0$, as shown in Figure 2.1.

FIG. 2.1. *Plot of the horizon function* $\delta(x) = \delta_0(1 + e^{-20(x-0.5)^2})$ *with* $\delta_0 = 1$.

The kernel $\gamma(s)$ has $O(1/|s|)$ singularity at 0 and is assumed to be smooth expect at origin and ± 1. The left column of Table 2.3 lists the types of regularity (at 0 and ± 1) of the kernel γ and the horizon parameter δ_0. The HIF algorithm for nonlocal diffusion in heterogeneous media works similarly as the homogeneous cases. In particular, we tailored Table 2.3 to the case where $\delta_0 = 1/4$ because for the extreme cases where $\delta_0 = 1$ or $\delta_0 = h$, the variation of regularity of the kernel does not result in any change in the computational cost, which is exactly the same behavior observed in Table 2.1.

kernel	δ_0	HIF		Original	
		Memory (MB)	$T_{\mathrm{multi}}(\mathrm{sec})$	Memory (MB)	$T_{\mathrm{multi}}(\mathrm{sec})$
C^{-1}	1/4	55.90	$6.18E-3$	40.24	$4.80E-3$
C^0	1/4	55.80	$6.24E-3$	40.24	$4.72E-3$
C^1	1/4	5.10	$3.92E-3$	40.24	$4.77E-3$
C^2	1/4	2.44	$3.27E-3$	40.24	$4.79E-3$
C^3	1/4	2.31	$3.13E-3$	40.24	$4.67E-3$

TABLE 2.3. *The storage and time of matrix-vector multiplication for the 1D nonlocal diffusion in heterogeneous medium. The domain* $\Omega = (0,1)$ *is discretized using* $N = 2048$ *points. The mesh size* $h = 1/2048$. *The relative precision of HIF approximation is set to be* $\epsilon = 10^{-8}$.

To account for the complexity, we provide the computation time for different numbers of discretization nodes N in Table 2.4. The horizon parameter δ_0 is fixed to be $1/4$. Two types of kernels – the C^0 kernel and the C^3 kernel – are used in the simulation. The complexity of the matrix-vector multiplication for the original matrix A without compression is $O(N^2)$ as expected. Similar to the homogeneous case, the observed complexity of HIF in the heterogeneous case is again $O(N^2)$ for the C^0 kernel and $O(N)$ for the C^3 kernel. The numerical examples in Sections 2.1 and 2.2 indicate that the HIF has the same effectiveness for models in heterogeneous media as it has for models in homogeneous media if the kernel functions are sufficiently smooth away from origin.

2.3. Two-dimensional test. In this section, we perform 2D tests on the domain $\Omega = (0,1) \times (0,1)$ to show the effectiveness of the HIF code. We choose $C(x,y) \equiv 1$ and

kernel	N	HIF		Original	
		Memory (MB)	T_{multi}(sec)	Memory (MB)	T_{multi}(sec)
C^0	512	3.50	$4.48E-4$	2.52	$2.29E-4$
C^0	1024	13.96	$1.75E-3$	10.07	$1.12E-3$
C^0	2048	55.80	$5.56E-3$	40.24	$4.13E-3$
C^0	4096	221.35	$1.98E-2$	160.87	$1.61E-2$
C^3	512	0.53	$1.34E-3$	2.52	$2.42E-4$
C^3	1024	1.22	$1.42E-3$	10.07	$1.12E-3$
C^3	2048	2.32	$2.61E-3$	40.24	$4.24E-3$
C^3	4096	4.56	$4.99E-3$	160.87	$1.66E-2$

TABLE 2.4. *The storage and time of matrix-vector multiplication for 1D nonlocal diffusion in heterogeneous medium. Kernel $\gamma(s)$ is C^3 away from origin. $\delta_0 = 1/4$. The relative precision of HIF approximation is set to be $\epsilon = 10^{-8}$.*

kernel	δ	HIF		Original	
		Memory (MB)	T_{multi}(sec)	Memory (MB)	T_{multi}(sec)
C^{-1}	1	161.71	$2.12E-2$	248.24	$2.50E-2$
C^0	1	33.01	$8.93E-3$	248.24	$2.45E-2$
C^1	1	32.08	$8.83E-3$	248.24	$2.37E-2$
C^2	1	31.50	$8.59E-3$	248.24	$2.46E-2$
C^3	1	31.16	$8.76E-3$	248.24	$2.42E-2$
C^{-1}	1/2	264.63	$3.20E-2$	89.51	$9.41E-3$
C^0	1/2	41.25	$1.04E-2$	89.51	$9.67E-2$
C^1	1/2	30.98	$8.98E-3$	89.51	$1.00E-2$
C^2	1/2	28.62	$8.86E-3$	89.51	$1.01E-2$
C^3	1/2	27.16	$8.59E-3$	89.51	$1.03E-2$
C^{-1}	h	7.85	$5.86E-3$	0.49	$8.17E-5$
C^0	h	7.85	$5.73E-3$	0.49	$8.05E-5$
C^1	h	7.85	$5.93E-3$	0.49	$8.78E-5$
C^2	h	7.85	$6.01E-3$	0.49	$8.68E-5$
C^3	h	7.85	$7.51E-3$	0.49	$9.40E-5$

TABLE 2.5. *The storage and time of matrix-vector multiplication for 2D nonlocal diffusion in homogeneous medium. The domain $\Omega = (0,1)^2$ is discretized using $N = 64 \times 64$ points. The mesh size $h = 1/64$. The relative precision of HIF approximation is set to be $\epsilon = 10^{-3}$.*

$\delta(x,y) \equiv \delta$. The homogeneous medium has the same horizon radius δ for every point. The kernel $\gamma(s)$ is the same as before and has $O(1/|s|)$ singularity at 0 and is assumed to be smooth expect at origin and ± 1.

In Table 2.5, the storage and the time of matrix-vector multiplication are listed for the original matrix A as well as its compression after using the HIF method. The original matrix A is again stored using the sparse matrix representation and the MATLAB built-in matrix-vector multiplication is used to record the computation time in the right column of Table 2.5. The relative precision is chosen to be $1E-3$ in the 2D test.

		HIF		Original	
kernel	N	Memory (MB)	$T_{\mathrm{multi}}(\mathrm{sec})$	Memory (MB)	$T_{\mathrm{multi}}(\mathrm{sec})$
C^{-1}	8^2	0.07	$1.18E{-}4$	0.03	$3.15E{-}5$
C^{-1}	16^2	1.05	$2.29E{-}4$	0.43	$7.43E{-}5$
C^{-1}	32^2	16.79	$2.71E{-}3$	6.01	$8.19E{-}4$
C^{-1}	64^2	264.28	$3.29E{-}2$	89.51	$1.11E{-}2$
C^3	8^2	0.07	$1.70E{-}4$	0.03	$3.24E{-}5$
C^3	16^2	0.62	$3.91E{-}4$	0.43	$7.49E{-}5$
C^3	32^2	4.19	$1.73E{-}3$	6.01	$7.70E{-}4$
C^3	64^2	27.21	$9.05E{-}3$	89.51	$1.09E{-}2$

TABLE 2.6. *The storage and time of matrix-vector multiplication for 2D nonlocal diffusion in homogeneous medium. $\delta = 1/2$. The relative precision of HIF approximation is set to be $\epsilon = 10^{-3}$.*

We remark that in the case of $\delta = 1/2$, the critical improvement of the computational efficiency happens between discontinuous kernel functions and C^0 kernel functions as a result of the choice of the precision parameter ϵ. Similarly as in 1D, higher critical regularity is observed if ϵ is getting smaller.

Table 2.6 shows the storage and time for different values of N with horizon parameter fixed to be $1/2$. The discontinuous (C^{-1}) kernel and the C^3 kernel are used in the simulation. Here we remark that we choose the C^{-1} kernel to be compared with the C^3 kernel because C^{-1} is right before the critical improvement of efficiency happens with the increase of regularity as seen in Table 2.5. This is a result of the particular choice of relative precision $\epsilon = 1E{-}3$. As ϵ becomes smaller, higher regularity is needed in order for the critical improvement of efficiency to happen. We observe that the complexity of the matrix-vector multiplication for the original matrix A without compression is approximately $O(N^2)$. HIF also gives $O(N^2)$ computation time for matrix-vector multiplication with the discontinuous kernel being used, while in the case of C^3 kernel, HIF performs much better and the observed complexity for matrix-vector multiplication is about $O(N \log(N))$.

To sum up, the simulations in Sections 2.1, 2.2 and 2.3 show that the popular choice of the kernel γ in (2.4) is problematic in the interest of fast computation since it has non-smooth truncation at ± 1. On the other hand, when the regularity of the kernel gets better the HIF fast algorithm becomes more effective. Those observations motivate us to propose the splitting method that deals kernels with non-smooth truncation in the simulations of the peridynamic model.

3. The splitting of singularities

Popular choices of kernels in the peridynamic simulations contain two types of singularities: the singularity at origin and the non-smooth truncation at the boundary $\partial \mathcal{H}_x$. The first type of singularity is handled by the HIF algorithm through the idea of far-field compression. In 1D, the compression is performed through a block matrix with small block sizes near the diagonal. The algorithm is thus effective when the kernel is sufficiently smooth away from origin, as we have seen in the previous numerical examples. Next, we propose to split the two types of singularities and treat them respectively. In Figure 3.1, the left plot is the kernel $\gamma(|s|) = \frac{1}{|s|} \chi(|s| < 1)$. The kernel splits into two parts, the middle plot which has singularity at zero but C^3 away from
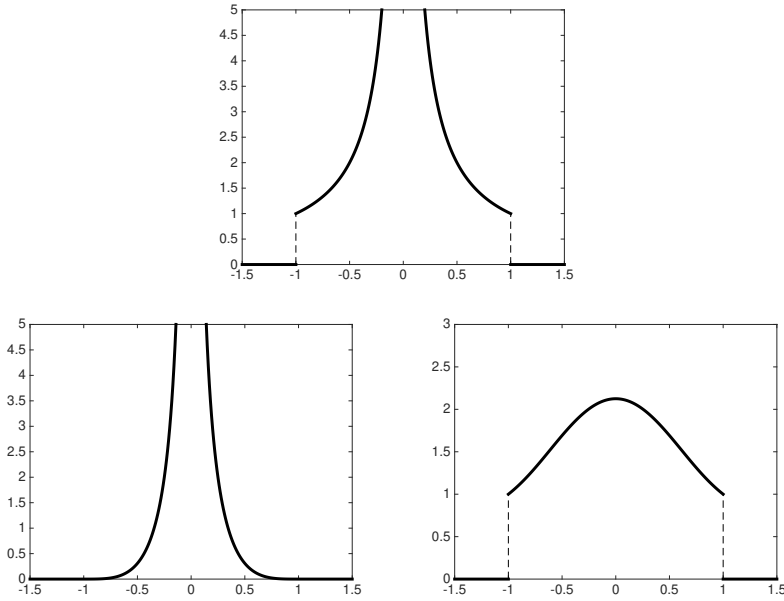
FIG. 3.1. *The kernel $\gamma(s)$ (top) splits into $\kappa(s)$ (bottom left) and $p^3(s)\chi(|s|<1)$ (bottom right).*

zero, and the right plot which is discontinuous at $s=\pm 1$ but smooth in the interior. In practice, the kernel $\gamma(|s|)$ is written into

$$\gamma(|s|) = \kappa(|s|) + p^{2K}(s)\chi(|s|<1),$$

where $\kappa(|s|)$ is C^K regular away from zero, and p^{2K} is an even polynomial of degree $2K$ that matches γ and its derivatives at $|s|=1$ up to the K-th order. For the choice of $\gamma(|s|) = \frac{1}{|s|}\chi(|s|<1)$, p^{2K} is explicitly given by the following formulas for $K=0,1,2,3$.

$$p^{2K}(s) = \begin{cases} 1 & K=0 \\ \dfrac{3}{2} - \dfrac{1}{2}s^2 & K=1 \\ \dfrac{15}{8} - \dfrac{5}{4}s^2 + \dfrac{3}{8}s^4 & K=2 \\ \dfrac{17}{8} - 2s^2 + \dfrac{9}{8}s^4 - \dfrac{1}{4}s^6 & K=3 \end{cases}$$

As suggested by Section 2, the nonlocal operator corresponding to the kernel $\kappa(s)$ can be treated by the HIF fast algorithm effectively. So we only need to deal with the second part corresponding to the kernel $p^{2K}(s)\chi(|s|<1)$. Taking $\Omega = [0,1]^d$ in the d-dimensional Euclidean space, we consider the following nonlocal diffusion problem with Dirichlet-type boundary

$$\begin{cases} -\mathcal{L}^P u(\boldsymbol{x}) := -\displaystyle\int_{\mathcal{H}_{\boldsymbol{x}}} P(\boldsymbol{x},\boldsymbol{y})(u(\boldsymbol{y}) - u(\boldsymbol{x}))d\boldsymbol{y} = f(\boldsymbol{x}) & x \in \Omega \\ u(\boldsymbol{x}) = 0 & x \in \Omega^c \end{cases} \tag{3.1}$$

where $\mathcal{H}_{\boldsymbol{x}}$ is some neighborhood of $\boldsymbol{x}$, and P is given by

$$P(\boldsymbol{x},\boldsymbol{y}) = \frac{C(\boldsymbol{x},\boldsymbol{y})}{\delta(\boldsymbol{x},\boldsymbol{y})^{d+2}} p^{2K} \left(\frac{|\boldsymbol{y}-\boldsymbol{x}|}{\delta(\boldsymbol{x},\boldsymbol{y})}\right). \tag{3.2}$$

For convenience of illustration, we take $C(\boldsymbol{x},\boldsymbol{y}) \equiv C(\boldsymbol{x})$, $\delta(\boldsymbol{x},\boldsymbol{y}) \equiv \delta(\boldsymbol{x})$, $K = 0$ and $p^0 \equiv 1$ in the next. The generalization to p^{2K} for $K > 0$ is straightforward. Taking into account the boundary condition in (3.1), the nonlocal operator is to be evaluated at each x:

$$\mathcal{L}^P u(\boldsymbol{x}) = \frac{C(\boldsymbol{x})}{\delta(\boldsymbol{x})^{d+2}} \left(\int_{\mathcal{H}_{\boldsymbol{x}} \cap \Omega} u(\boldsymbol{y}) d\boldsymbol{y} - |\mathcal{H}_{\boldsymbol{x}}| u(\boldsymbol{x}) \right). \tag{3.3}$$

We therefore only need to find a fast algorithm to compute $\int_{\mathcal{H}_{\boldsymbol{x}} \cap \Omega} u(\boldsymbol{y}) d\boldsymbol{y}$. Our algorithm starts by the hierarchical decomposition of the space Ω, which is similar to the one in standard FMM [12]. Assume that the domain Ω is uniformly discretized. Let $N = n^d$ be the total number of discretization nodes. Let $L = \log_2(n)$, then from level 1 to L, the computation domain $\Omega = [0,1]^d$ is hierarchically subdivided into panels. Each panel in the l-th level can be represented by one of the cubes $\prod_{j=1}^{d} [\frac{k_j}{2^l}, \frac{k_j+1}{2^l}]$, $k_j = 0, 1, \cdots, 2^l - 1$.

The hierarchical subdivision of Ω forms a tree structure. Figure 3.2 is the quadtree formed by the subdivision of the two-dimensional domain $[0,1]^2$. The blue dot in Figure 3.2 represents the original domain $[0,1]^2$, and it has four children $\{[\frac{i}{2}, \frac{i+1}{2}] \times [\frac{j}{2}, \frac{j+1}{2}]\}_{i,j=0}^{1}$ represented by the red dots in level 1. The subdivision of the 4 domains in level 1 results in 16 subdomains in level 2 represented by the black dots. The hierarchical subdivision is performed until reaching the L-th level. Similarly, in 1D a binary tree will be formed by the subdivision of $[0,1]$ and in 3D an octree will be formed by the subdivision of $[0,1]^3$.
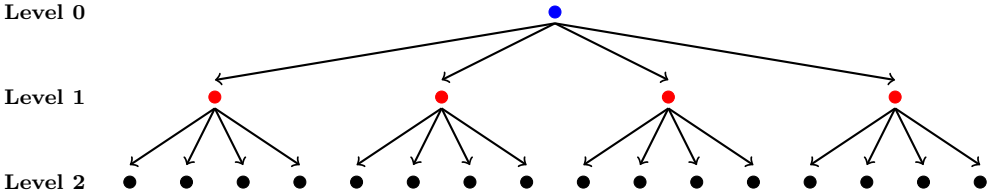

FIG. 3.2. *Tree structure induced by the hierarchical subdivision of* $[0,1]^2$.

We now present the algorithm. It contains the following three steps that will be explained in detail.

(1) *Initialization. Traverse the tree from top to bottom to obtain the decomposition of* $\mathcal{H}_{\boldsymbol{x}} \cap \Omega$ *for each grid point* $\boldsymbol{x}$.

(2) *Traverse the tree from bottom to top to obtain the partial sums* $\sum_{\boldsymbol{x}_k \in Q_l^j} u(\boldsymbol{x}_k)$ *for each panel* Q_l^j *in the l-th level.*

(3) *Approximate the integral* $\int_{\mathcal{H}_{\boldsymbol{x}} \cap \Omega} u(\boldsymbol{y}) d\boldsymbol{y}$ *by using (1) and (2).*

Step 1. *Initialization step.* For each $\boldsymbol{x}$ on the grid point, decompose the domain of integration $\mathcal{H}_{\boldsymbol{x}} \cap \Omega$ using the panels in L_l $(l = 0, 1, \cdots, L)$. The decomposition of $\mathcal{H}_{\boldsymbol{x}} \cap \Omega$ for each $\boldsymbol{x}$ is stored for reuse. More specifically, we traverse the tree from top to

bottom to determine whether a panel Q_l^j from level l is included or not, and then express $\mathcal{H}_{\boldsymbol{x}} \cap \Omega = \bigcup_{l=0}^{L} \bigcup_{j \in \mathcal{I}_l(\boldsymbol{x})} Q_l^j$, where $\mathcal{I}_l(\boldsymbol{x})$ is the set of indices at the l-th level to be included in the decomposition. We may do this recursively by calling the function recur(Ω), where recur is defined in Algorithm 1 below.

> function recur(Q)
> **if** *Q is fully contained in $\mathcal{H}_{\boldsymbol{x}} \cap \Omega$* **then**
> > Q is included;
> > **return** 1;
>
> **else if** *Q intersects with $(\mathcal{H}_{\boldsymbol{x}} \cap \Omega)$ nontrivially* **then**
> > **if** *Q is not a leaf* **then**
> > > run recur($Q[k]$) from $k=1$ to $k=2^d$, where $Q[k]$ is the k-th child of Q;
> >
> > **else**
> > > Q is included;
> > > **return** 1;
> > **end**
>
> **else**
> > **return** 0;
> **end**

Algorithm 1: The recursive function.

Let us now discuss on the complexity of performing the recur function when $\mathcal{H}_{\boldsymbol{x}}$ is a d-dimensional ball centered at $\boldsymbol{x}$. The main cost of running the recur function comes from the intersection test of balls and cubes. To test whether a cube Q is fully contained in a ball, we only need to test whether the total of 2^d corners of Q are all contained in the ball. This requires a total of $O(2^d)$ operations, where the constant in $O(2^d)$ is independent of the dimension d. Now to test whether a cube Q has nontrivial intersection with a ball, we use the algorithm given in [2], which is an $O(d)$ algorithm that determines whether an axis-aligned bounding box (AABB) intersects a ball. See the function defined in Algorithm 2, where the inputs are three vectors $\mathbf{Bmin}, \mathbf{Bmax}, \mathbf{C}$ and a positive number r. The vector $\mathbf{Bmin} \in \mathbb{R}^d$ stores the minima of the AABB for each axis, $\mathbf{Bmax} \in \mathbb{R}^d$ stores the maxima of the AABB for each axis, and $\mathbf{C} \in \mathbb{R}^d$ and r are the center and radius of the ball resepctively. Combining the above discussions, we know that each time calling the recur function requires $C \cdot 2^d$ operations, where C is independent of dimension.

Figure 3.3 shows an example of decomposing a circular region in 2D. Now the total complexity of Step 1 is obviously equal to $C \cdot 2^d$ times the total number of times the recur function is called. Now for each discretization node $\boldsymbol{x}_i$ ($i = 1, 2, \cdots, N$), let $\mathcal{N}_d(\boldsymbol{x}_i)$ denote the number of times the recur function is called in order to decompose the domain $\mathcal{H}_{\boldsymbol{x}_i} \cap \Omega$. The total number of times the recur function is called is then given by $\sum_{i=1}^{N} \mathcal{N}_d(\boldsymbol{x}_i)$. Now observe that $\mathcal{N}_d(\boldsymbol{x}_i)$ is also equal to the number of panels that intersect with the boundary $\partial \mathcal{H}_{\boldsymbol{x}_i}$ non-trivially, namely

$$\mathcal{N}_d(\boldsymbol{x}_i) = \# \bigcup_{l=0}^{L} \{Q_l^j \in L_l : (Q_l^j)^{\circ} \cap \partial \mathcal{H}_{\boldsymbol{x}_i} \neq \emptyset\} = \sum_{l=0}^{L} \mathcal{N}_d^l(\boldsymbol{x}_i),$$

where $\mathcal{N}_d^l(\boldsymbol{x}_i)$ denotes the number of panels in the l-th level that intersect with $\partial \mathcal{H}_{\boldsymbol{x}_i}$ non-trivially. Assume that for each point $\boldsymbol{x}_i$, the set $\partial \mathcal{H}_{\boldsymbol{x}_i}$ has uniform bounded mean

function doesCubeIntersectBall (**Bmin**, **Bmax**, **C**, r)
$dmin = 0;$
for $i = 1 : d$ **do**
    if **Bmin**$[i] > $**C**$[i]$ **then**
       | $dmin = dmin + ($**Bmin**$[i] - $**C**$[i])^2;$
    else if **Bmax**$[i] < $**C**$[i]$ **then**
       | $dmin = dmin + ($**Bmax**$[i] - $**C**$[i])^2;$
end
if $dmin < r^2$ **then**
  | **return** TRUE ;
else
  | **return** FALSE;
end

    Algorithm 2: The intersection test of d-dimensional cubes and balls.
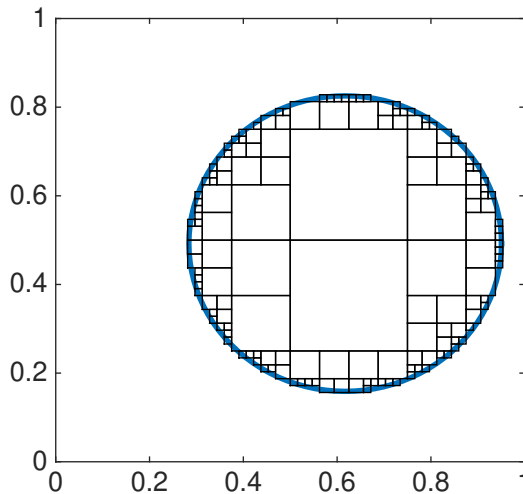
FIG. 3.3. *The decomposition of a circular region into panels in* $\{L_i\}$

curvature, then $\mathscr{N}_d^l(\boldsymbol{x}_i)$ can be estimated by the following formula

$$\mathscr{N}_d^l(\boldsymbol{x}_i) = \begin{cases} O(1) & \text{for } d = 1; \\ O\left(\dfrac{N}{(2^d)^l}\right)^{1-1/d} & \text{for } d \geq 2. \end{cases}$$

To sum up, the total complexity of Step 1, which equals $C \cdot 2^d \sum_{i=1}^{N} \sum_{l=0}^{L} \mathscr{N}_d^l(\boldsymbol{x}_i)$ is given by

$$\begin{cases} O(N \log N) & \text{for } d = 1; \\ C_d \cdot N^{2-1/d} & \text{for } d \geq 2, \end{cases} \tag{3.4}$$

where the dimension-dependent constant C_d is of order $O(2^d)$.

    Step 2. *Compute the the partial sums for each tree node.* For a given vector $\{u(\boldsymbol{x}_i)\}_{i=1}^{N}$, we assign the value $u(\boldsymbol{x}_i)$ to a leaf node. Then traversing the tree bottom to

top, we assign each parent node the value of the sum of all its children. The completion of this step gives us the partial sums $\sum_{\boldsymbol{x}_k \in Q_l^j} u(\boldsymbol{x}_k)$ for every panel Q_l^j. The partial sums obtained can then be used to approximate the integral $\int_{Q_l^j} u(\boldsymbol{y}) d\boldsymbol{y}$. We remark that if the kernel function given by (3.2) is generated from the polynomial p^{2K} with $K > 0$, we then need to compute the partial sums in the form of $\sum_{x_k \in Q_l^j} x_k^m u(x_k)$ for all $m = 0, 1, \cdots 2K$, so that they can be used to approximate the integral $\int_{Q_l^j} P(\boldsymbol{x}, \boldsymbol{y}) u(\boldsymbol{y}) d\boldsymbol{y}$.

Figure 3.4 exemplifies the way partial sums are computed for a two-dimensional problem. Each of the black dots on the left figure represents a leaf node that contains the partial sum of the function in the little black box that includes it. Then the partial sums are passed to the coarser level where each parent node represented by the red dots stores the sum of the values from its four children. The total sum of the function on the whole domain is stored in the top tree node represented by the blue dot in the right figure in Figure 3.4. Therefore, the computational cost of Step 2 depends only on the number of discretization nodes N and the degree $2K$ of the polynomial p^{2K}. To sum up, the complexity of Step 2 is given by $O(KN)$, where the constant in $O(KN)$ is independent of dimension.
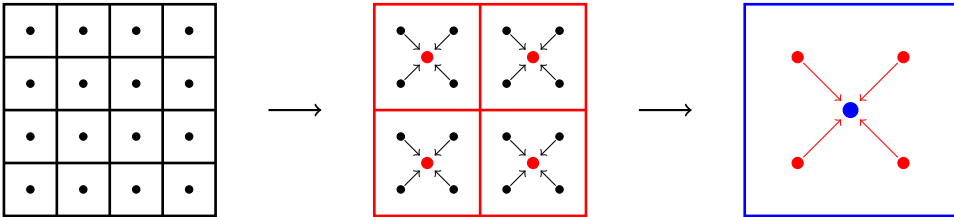

FIG. 3.4. *A 2D example of the process of computing partial sums.*

Step 3. For each discretization node $\boldsymbol{x}_i$ $(i = 1, 2 \cdots N)$, use the decomposition of $\mathcal{H}_{\boldsymbol{x}_i} \cap \Omega$ obtained in Step 1 and the partial sums obtained in Step 2 to approximate the integral

$$\int_{\mathcal{H}_{\boldsymbol{x}_i} \cap \Omega} u(\boldsymbol{y}) d\boldsymbol{y} = \sum_{l=0}^{L} \sum_{j \in \mathcal{I}_l(\boldsymbol{x}_i)} \int_{Q_l^j} u(\boldsymbol{y}) d\boldsymbol{y}. \tag{3.5}$$

Finally, the value of $\mathcal{L}_\delta^P u(\boldsymbol{x}_i)$ is obtained through the relation (3.3). We again remark that in the case of $K > 0$ for the polynomial p^{2K}, all the partial sums of the form $\sum_{x_k \in Q_l^j} x_k^m u(x_k) (m = 0, 1, \cdots, 2K)$ will be needed to approximate the desired integral. With the precomputed sums for the approximate integrals of the form $\int_{Q_l^j} u(\boldsymbol{y}) d\boldsymbol{y}$ in Step 2, the number of summations we need to take in (3.5) is essentially equal to $\mathcal{N}_d(\boldsymbol{x}_i)$ defined in Step 1. By the calculations in Step 1 and taking into the account the general case that p^{2K} being a polynomial of degree $2K > 0$, complexity of Step 3 is then given by the following formulas,

$$\begin{cases} O(KN \log N) & \text{for } d = 1; \\ O(KN^{2-1/d}) & \text{for } d \geq 2, \end{cases} \tag{3.6}$$

and the constants in the above estimates are independent of dimension.

3.1. Numerical tests.   We perform numerical tests based on the proposed algorithm.

Example 1. In this example, we take $\Omega = [0,1]^2$ in the 2-dimensional space. Assume $k = 0$, $p^0(x) \equiv 1$, $\delta(\boldsymbol{x}) = \frac{1}{2}$ and $C(\boldsymbol{x}) = 1$. The nonlocal operator to be evaluated is given by

$$\mathcal{L}_\delta^P u(\boldsymbol{x}) = \frac{1}{\delta^4} \int_{B_\delta(\boldsymbol{x})} (u(\boldsymbol{y}) - u(\boldsymbol{x})) d\boldsymbol{y} = \frac{1}{\delta^4} \int_{B_\delta(\boldsymbol{x}) \cap \Omega} u(\boldsymbol{y}) d\boldsymbol{y} - \frac{|B_1|}{\delta^2} u(\boldsymbol{x}).$$

Table 3.1 contains the experimental data of the new algorithm in comparison with two versions of matrix-vector multiplication for the original matrix. The scaling results are shown in Figure 3.5. Since the initialization step (Step 1) is only needed to be performed once in an iterative method or time-dependent problem, the computation time recorded for the new algorithm is given by the cost for Step 2 and Step 3 combined together. The computation time of the new algorithm grows at the rate $O(N^{1.5})$, which verifies the formula given in (3.4) for $d = 2$. In comparison, the two versions of matrix-vector multiplication for the original matrix shows $O(N^2)$ complexity. The first version (Original - 1) is performed with the MATLAB built-in matrix-vector multiplication. Because of the nature of MATLAB, the second version (Original -2), which goes through the evaluation of the nonlocal operator at each node $\boldsymbol{x}_i$ $(i = 1, 2, \cdots N)$ by using a for-loop, is also listed for a fair comparison with the new algorithm.

	New Algorithm		Original - 1		Original - 2	
N	T_{multi}(sec)	rate	T_{multi}(sec)	rate	T_{multi}(sec)	rate
8^2	$6.47E{-}4$	$--$	$3.58E{-}5$	$--$	$4.64E{-}4$	$--$
16^2	$3.94E{-}3$	1.30	$7.83E{-}5$	0.56	$5.56E{-}3$	1.79
32^3	$3.03E{-}2$	1.47	$6.94E{-}4$	1.57	$9.02E{-}2$	2.01
64^2	$2.32E{-}1$	1.47	$9.99E{-}3$	1.92	$2.85E{+}0$	2.50
128^2	$1.93E{+}0$	1.53	$1.66E{-}1$	2.03	$8.14E{+}1$	2.42

TABLE 3.1. *Computation time for evaluating the nonlocal operator in 2D. $k = 0$ and $p^0 = 1$. T_{multi} under the new algorithm is the time for Step 2 and Step 3 combined. Original -1 uses the MATLAB built-in sparse matrix-vector multiplication. Original -2 uses a for-loop to go through $\boldsymbol{x}_i (i = 1, 2, \cdots, N)$ together with the MATLAB built-in vector-vector multiplication applied in each loop.*

Example 2. To show that the algorithm can be applied to more generalized cases, we perform the numerical test on the 1D domain $\Omega = [0,1]$ and the kernel function $P(x,y)$ is given by (3.2) with $C(x,y) = 1$, $\delta(x,y) = \delta(x)$ as shown in Figure 2.1. We take $K = 3$ and p^6 a polynomial of degree 6. Now in Step 2 of the algorithm, we not only need to compute the partial sums $\sum_{x_k \in Q_l^j} u(x_k)$, but also $\sum_{x_k \in Q_l^j} x_k^m u(x_k)$ for all $m = 1, 2, \cdots 6$ so that they can be used to approximate $\int_{\mathcal{H}_x \cap \Omega} P(x,y) u(y) dy$.

The computation time for the new algorithm is given in Table 3.2 in comparison with two versions of matrix-vector multiplication for the original matrix, with scaling results shown in Figure 3.6. The computation time of the new algorithm grows at the rate which is slightly higher than $O(N)$, which relates well to our theoretical estimate of $O(N \log(N))$ for $d = 1$.
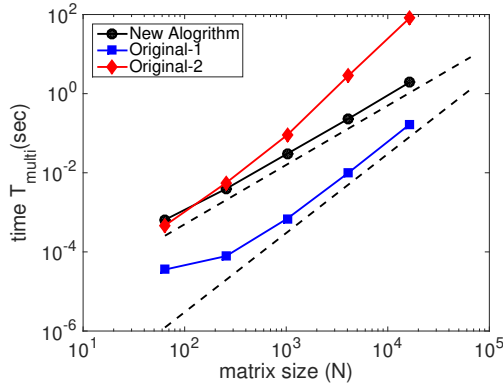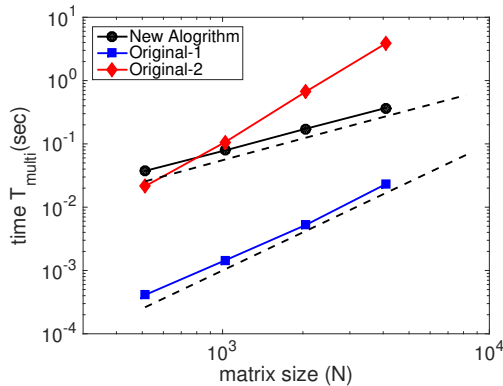
FIG. 3.5. *Scaling results for Example 1, comparing the new algorithm with the two versions of original matrix-vector multiplication. The reference scalings (dashed lines) of $O(N^2)$ and $O(N^{1.5})$ are also included.*

	New Algorithm		Original - 1		Original - 2	
N	T_{multi}(sec)	rate	T_{multi}(sec)	rate	T_{multi}(sec)	rate
512	$3.74E-2$	$--$	$4.10E-4$	$--$	$2.14E-2$	$--$
1024	$7.90E-2$	1.08	$1.45E-3$	1.82	$1.07E-1$	2.32
2048	$1.73E-1$	1.13	$5.22E-3$	1.85	$6.65E-1$	2.64
4096	$3.69E-1$	1.10	$2.32E-2$	2.15	$3.87E+0$	2.54

TABLE 3.2. *Computation time for evaluating the nonlocal operator in 1D. $k=3$ and p^3 is an even polynomial of degree 6. $\delta(x)$ is given by Figure 2.1 with $\delta_0 = 1/4$. T_{multi} under the new algorithm is the time for Step 2 and Step 3 combined. Original -1 uses the MATLAB built-in sparse matrix-vector multiplication. Original -2 uses a for-loop to go through $\boldsymbol{x}_i (i=1,2,\cdots,N)$ together with the MATLAB built-in vector-vector multiplication applied in each loop.*

FIG. 3.6. *Scaling results for Example 2, comparing the new algorithm with the two versions of original matrix-vector multiplication. The reference scalings (dashed lines) of $O(N^2)$ and $O(N\log(N))$ are also included.*

4. Conclusion

In this work, we develop and investigate fast algorithms for a class of nonlocal models featured by finite domain of nonlocal interactions. Important applications include the computation of the peridynamics model of solid mechanics and models of nonstandard diffusion. The main message is that the far field behavior of the nonlocal interaction kernel is decisive in order for the fast computation of nonlocal models. We use numerical tests for the nonlocal diffusion model as a proof of concept, and demonstrate that better regularity of the kernel away from zero is better in the interest of fast computation. This is reasonable since most fast algorithms that can be applied to non-translation invariant operators and general meshes are based on low rank approximation of the far field interactions. Based on this observation, the kernels that are most often used in the peridynamics simulations fail to work under existing fast algorithms since they all have non-smooth truncations at a finite distance. We propose a remedy for such kernels, and that is the splitting of the singularities at origin and at the finite distance and treating them separately. The splitting is done by constructing polynomials that match the derivatives of the original kernels at finite distances. The second type of singularity after splitting is on larger sets than just point discontinuity. We propose a new FMM-like fast algorithm to deal with such singularities which exploits the smoothness of the kernels away from the truncation. Such an idea that uses low rank approximations of kernels for different regions is also presented in [8] for a completely different problem. Note that we only focus on the fast matrix-vector multiplication in this work, and it can be used in a Krylov subspace method if computing the inverse of a matrix is the interest. Notice that the kernel function that is used to regularize the original kernel typically is not small, as indicate in the bottom right picture in Figure 3.1. If the original kernel is a small perturbation of a kernel that has smooth decay in the far field, then the inverse of the corresponding matrix could be simply computed using the series expansion under the smallest assumption. The goal of this paper is to show that fast algorithms for nonlocal operators with finite interaction distances are possible. Even if the techniques developed here substantially reduce the computational cost, there is room for improvements from future research. It would be natural to pursue further research as nonlocal modeling becomes more common. The essentially optimal computational complexity of $O(N \log N)$ is for 1D and in higher dimensions there is algebraic complexity higher than linear. For further complexity reduction in higher dimensions a better representation of precomputed quantities near the horizon would be required and it would be natural to use different geometries than boxes, for example, curvelets [5] in 2D deal efficiently with line discontinuities.

Based on our findings, several more points have the value of further study. First, our work offers important messages to nonlocal modeling from the perspective of scientific computing. It is worthwhile to explore in the future whether the kernels with smooth decays at far field can be used in place of non-smooth kernels in the nonlocal models without compromising the physics. Second, in order to understand the properties of the algorithms, we separated the global HIF technique from the handling of discontinuities at the horizon by using a simple splitting. The more interesting question on the optimal way to split the singularities could be asked in the future. Moreover, it will be more desirable to build a tighter coupling of the singularities into the original HIF technique, and the question of detecting the zero values of the kernel could be asked as mentioned earlier. Finally, we note that our case study is restricted to nonlocal diffusion-type problem. It is reasonable that for the peridynamics system of equations similar studies can be done.

REFERENCES

[1] F. Andreu-Vaillo, J. Mazn, J. Rossi, and J. Toledo-Melero, *Nonlocal Diffusion Problems*, Mathematical Surveys and Monographs, Providence, RI, 2010. 1

[2] J. Arvo, *Graphics Gems II*, Elsevier, 2013. 3

[3] E. Askari, F. Bobaru, R.B. Lehoucq, M.L. Parks, S.A. Silling, and O. Weckner, *Peridynamics for multiscale materials modeling*, J. Phys. Conf. Ser., 125(1):012708, 2008. 1

[4] P. Bates and A. Chmaj, *An integrodifferential model for phase transitions: Stationary solutions in higher space dimensions*, J. Stat. Phys., 95:1119–1139, 1999. 1

[5] E.J. Candes and D.L. Donoho, *Curvelets: A surprisingly effective nonadaptive representation for objects with edges*, Technical Report, Stanford Univ. Ca. Dept. of Stat., 2000. 4

[6] Q. Du, M. Gunzburger, R. Lehoucq, and K. Zhou, *Analysis and approximation of nonlocal diffusion problems with volume constraints*, SIAM Rev., 56:676–696, 2012. 1, 2

[7] Q. Du, J. Zhang, and C. Zheng, *Nonlocal wave propagation in unbounded multi-scale media*, Comm. Comput. Phys., 24:1049–1072, 2018. 2

[8] B. Engquist and L. Ying, *Fast directional multilevel algorithms for oscillatory kernels*, SIAM J. Sci. Comput., 29:1710–1737, 2007. 4

[9] P. Fife, *Some nonclassical trends in parabolic and parabolic-like evolutions*, in M. Kirkilionis, S Krömker, R Rannacher and F. Tomi (eds.), Trends in Nonlinear Analysis, Springer, 153–191, 2003. 1

[10] J.T. Foster, S.A. Silling, and W.W. Chen, *Viscoplasticity using peridynamics*, Int. J. Numer. Meth. Eng., 81:1242–1258, 2010. 1

[11] R.A. Gingold and J.J. Monaghan, *Smoothed particle hydrodynamics: theory and application to non-spherical stars*, Mon. Not. R. Astron. Soc., 181:375–389, 1977. 1

[12] L. Greengard and V. Rokhlin, *A fast algorithm for particle simulations*, J. Comput. Phys., 73:325–348, 1987. 1, 3

[13] L. Greengard and V. Rokhlin, *A new version of the fast multipole method for the Laplace equation in three dimensions*, Acta Numer., 6:229–269, 1997. 1

[14] Y.D. Ha and F. Bobaru, *Studies of dynamic crack propagation and crack branching with peridynamics*, Int. J. Fracture, 162:229–244, 2010. 1, 1, 2

[15] Y.D. Ha and F. Bobaru, *Characteristics of dynamic brittle fracture captured with peridynamics*, Eng. Fract. Mech., 78:1156–1168, 2011. 1

[16] W. Hackbusch, *A sparse matrix arithmetic based on $\mathcal{H}$-matrices. Part I: Introduction to $\mathcal{H}$-matrices*, Computing, 62:89–108, 1999. 1

[17] W. Hackbusch and S. Börm, *Data-sparse approximation by adaptive $\mathcal{H}^2$-matrices*, Computing, 69:1–35, 2002. 1

[18] K.L. Ho and L. Ying, *Hierarchical interpolative factorization for elliptic operators: integral equations*, Comm. Pure Appl. Math., 69:1314–1353, 2016. 1, 2

[19] Z. Li and Z. Shi, *A convergent point integral method for isotropic elliptic equations on a point cloud*, Multiscale Model. Sim., 14:874–905, 2016. 1

[20] A. Massaccesi and E. Valdinoci, *Is a nonlocal diffusion strategy convenient for biological populations in competition?* J. Math. Bio., 74:113–147, 2017. 1

[21] R. Metzler and J. Klafter, *The random walk's guide to anomalous diffusion: a fractional dynamics approach*, Phys. Rep., 339:1–77, 2000. 1

[22] R. Nochetto, D. Ntogkas, and W. Zhang, *Two-scale method for the Monge-Ampère equation: Convergence to the viscosity solution*, Math. Comput., 88:637–664, 2019. 1

[23] R.H. Nochetto and W. Zhang, *Discrete ABP estimate and convergence rates for linear elliptic equations in non-divergence form*, Found. Comput. Math., 18:537–593, 2018. 1

[24] F.-J. Sayas, *Retarded Potentials and Time Domain Boundary Integral Equations: A Road Map*, Springer, 50, 2016. 1

[25] S. Silling, *Reformulation of elasticity theory for discontinuities and long-range forces*, J. Mech. Phys. Solids, 48:175–209, 2000. 1

[26] S. Silling, M. Epton, O. Weckner, J. Xu, and E. Askari, *Peridynamic states and constitutive modeling*, J. Elasticity, 88:151–184, 2007. 1

[27] S. Silling, D.J. Littlewood, and P. Seleson, *Variable horizon in a peridynamic medium*, J. Mech. Mater. Struct., 10(5):591–612, 2015. 2

[28] X. Tian, *Nonlocal models with a finite range of nonlocal interactions*, PhD thesis, Columbia University, 2017. 1

[29] X. Tian and Q. Du, *Analysis and comparison of different approximations to nonlocal diffusion and linear peridynamic equations*, SIAM J. Numer. Anal., 51:3458–3482, 2013. 2.1

[30] X. Tian and Q. Du, *Trace theorems for some nonlocal function spaces with heterogeneous localization*, SIAM J. Math. Anal., 49:1621–1644, 2017. 2

[31] C. Wang and H. Wang, *A fast collocation method for a variable-coefficient nonlocal diffusion model*, J. Comput. Phys., 330:114–126, 2017. 1

[32] L. Ying, G. Biros, and D. Zorin, *A kernel-independent adaptive fast multipole algorithm in two and three dimensions*, J. Comput. Phys., 196:591–626, 2004. 1

[33] X. Zhang and H. Wang, *A fast collocation method for a static bond-based linear peridynamic model*, Comput. Meth. Appl. Mech. Eng., 311:280–303, 2016. 1