Available online at www.sciencedirect.com

SCIENCE DIRECT®

Journal of Computational Physics 213 (2006) 451–457

JOURNAL OF COMPUTATIONAL PHYSICS

www.elsevier.com/locate/jcp

Short note

A kernel independent fast multipole algorithm for radial basis functions

Lexing Ying *

Applied and Computational Mathematics, California Institute of Technology, MC 217-50, Caltech, Pasadena, CA 91125, United States

Received 18 June 2005; accepted 14 September 2005
Available online 2 November 2005

Abstract

We present a fast multipole algorithm for the evaluation of pairwise interaction through the radial basis functions such as $1/r^a$, $\sqrt{r^2+a^2}$ and $1/\sqrt{r^2+a^2}$ (with $a > 0$) in both two and three dimensions. Our algorithm is an extension of the kernel independent fast multipole method presented in Ying et al. [L. Ying, G. Biros, D. Zorin, A kernel-independent adaptive fast multipole algorithm in two and three dimensions, J. Comput. Phys. 196(2) (2004) 591–626]. Numerical results are provided to illustrate the accuracy and complexity properties of the algorithm.
© 2005 Elsevier Inc. All rights reserved.

Keywords: Radial basis functions; Fast multipole method; Kernel independency; Multiscale method

1. Introduction

Radial basis functions (RBFs) are widely used in various fields of computational sciences. Applications include image processing, scattered data interpolation, physical-interaction modeling and so on. A common component of these applications is the evaluation of pairwise interaction through RBFs for a set of points: given a set of source densities $\{\phi_i\}$ located at points $\{x_i\}$ in two or three dimensions and a RBF kernel G, we want to compute for every x_i the potential $u_i = \sum_j G(|x_i - x_j|)\phi_j$. Some commonly used RBFs are power functions ($1/r^a$ with $a > 0$), multiquadrics ($\sqrt{r^2+a^2}$ and $1/\sqrt{r^2+a^2}$) and Gaussian kernels (e^{-ar^2}). While having infinite support is essential to many useful properties of these RBFs, it also makes the evaluation process prohibitive for large data sets. For a problem with N points, the direct calculation is of order $O(N^2)$.

In the past two decades, a number of efficient approximation algorithms have been proposed. The original fast multipole method (FMM) [6] by Greengard and Rokhlin was developed for the fast evaluation of the interaction through the fundamental solutions of the Laplace equation. Gaussian kernels were addressed in [7,8]. A series of papers [2,3,1,4] by Beatson and coauthors developed FMM type algorithms for polyharmonic spline and multiquadrics. [5] proposed an algorithm for the kernel $1/r$ in 2D.

* Tel.: +1 626 395 5760.
E-mail address: lexing@acm.caltech.edu.

0021-9991/$ - see front matter © 2005 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2005.09.010

A recent paper [9] proposes a kernel independent FMM type algorithm for a wide class of non-oscillatory kernels originated from second-order elliptic partial differential equations (PDEs). This algorithm relies on the results from the potential theory, such as the existence and uniqueness theorems for the boundary value problem of these PDEs. In the current paper, we extend this kernel independent algorithm to the kernels $1/r^a$, $\sqrt{r^2+a^2}$ and $1/\sqrt{r^2+a^2}$ (with $a > 0$), which do not satisfy any second-order elliptic PDE.

2. Algorithm description

Given a set of N source densities $\{\phi_i\}$ located at points $\{x_i\}$ in R^d ($d = 2, 3$), and a RBF kernel G, the goal is to compute efficiently and accurately the potentials $\{u_i\}$ defined by the following formula:

$$u_i = \sum_{j=1}^{N} G(|x_i - x_j|)\phi_j.$$

In this section, we first review the kernel independent FMM algorithm of [9] for G being an elliptic PDE kernel. Then we describe how to modify this algorithm to accommodate the situation where G is a RBF kernel.

2.1. Review of kernel independent FMM for PDE kernels

In the case where G is a kernel from the Laplace equation, the fast multipole method (FMM) [6] is an algorithm which computes u_i within a prescribed accuracy in linear time. To simplify the exposition, we assume that the points x_i are uniformly distributed inside the unit box centered at the origin of R^d (for the non-uniform case, see [6,9]). The FMM starts by subdividing the unit box dyadically into an octree tree such that the number of points in every leaf box is less than a fixed constant s. Since the point distribution is assumed to be uniform, the resulting octree is full. For a box B, the *near field* $N(B)$ is defined to be the region containing B and its adjacent neighbors. The rest is the *far field* $F(B)$. The *interaction list* of B, denoted as $I(B)$, consists of the boxes which are in the same level as B and, at the same time, belong to the region $F(B) - F(P)$ where P is the parent box of B.

The basic idea of FMM is to construct two representations for every box B: (1) the multipole expansion m_B which accurately approximates in the far field region $F(B)$ the potential generated by the densities in B; (2) the local expansion l_B which accurately approximates in B the potential generated by the densities in $F(B)$. Additionally, for a fixed error tolerance, the sizes of both expansions are independent of the number of densities contained in the box. In the original FMM algorithm [6] for the Laplace equation, the multipole expansion is based on the Laurent series in 2D and on the spherical harmonics in 3D. The algorithm proceeds in several stages. First, the multipole expansions are constructed during a bottom-up traversal of the octree. The multipole expansion of a leaf box is constructed directly from the densities in this box using analytic expansion of the kernel G while the multipole expansion of a non-leaf box is constructed from the multipole expansion of its children box using the multipole-to-multipole (M2M) translation. Next, the local expansions are generated during a top-down traversal of the octree. For every box B, the local expansion is computed from the local expansion of its parent box using the local-to-local (L2L) translation, and the multipole expansion of the boxes in its interaction list $I(B)$ using the multipole-to-local translation (M2L) translation. It is important to notice that the sum of these two parts gives the potential in B induced by all the densities in $F(B)$. All three translations (M2M, M2L and L2L) are derived from the analytic expansion of the kernel G and have constant complexity. Finally, for every leaf box B, we compute the potential at each of its point as the sum of the potential from the local expansion l_B and the potential generated by the original densities in $N(B)$. For a fixed error tolerance, the linear complexity of this algorithm follows from the facts that all expansions and translations have constant complexity and the number of densities in $N(B)$ is bounded by a multiple of the constant s.

In the classical FMM described above, the expansions and translations all depend on the kernel G in an analytic way. For some kernels, these analytic expansions and translations can be quite difficult to obtain. The main advantage of the kernel independent algorithm of [9] is that it removes such kernel dependency. This algorithm has the same structure as the classical FMM. The differences between them concern which representations are used and how the translations are carried out. First, the kernel dependent multipole expansion of a box B is replaced with an *upwards equivalent density*, denoted by ϕ_B^{u}, distributed on ∂B. This is justified by the fact that this boundary density is capable of reproducing the potential in $F(B)$ induced by the source densities inside B. In order to compute it, we only need to match, on $\partial F(B)$, the potentials generated by ϕ_B^{u} and by the source densities inside B. The uniqueness theorem of the boundary value problems of the elliptic PDEs guarantees that as long as the two potentials match on $\partial F(B)$ they match everywhere inside $F(B)$. This matched potential is called the *upwards check potential*, and denoted by u_B^{u}. In actual numerical implementation, the *discretized* upwards equivalent density is supported on an evenly sampled Cartesian grid of ∂B, while the check potential is matched at an evenly sampled Cartesian grid of $\partial F(B)$ (see Fig. 1(a)). The matching process to solve for the upwards equivalent density ϕ_B^{u} is done by first evaluating the check potential u_B^{u} using the densities inside B, and then multiplying it with the (regularized) inverse of the matrix which relates the ϕ_B^{u} to u_B^{u}.

Similarly, the kernel dependent local expansion of a box B is replaced with an *downwards equivalent density* ϕ_B^{d} distributed on a Cartesian grid of $\partial F(B)$. To compute ϕ_B^{d}, we match on a Cartesian grid of ∂B the potentials generated by ϕ_B^{d} itself and by the source densities in the far field $F(B)$ (see Fig. 1(b)). This matched potential is called the *downwards check potential* and denoted by u_B^{d}.

The M2M, M2L and L2L translations are replaced with procedures using the same *matching* idea. For example, in the M2L translation from box A to box B, we first evaluate u_B^{d} (the downwards checking potential) induced by ϕ_A^{u} (the upwards equivalent density of A) and then solve for ϕ_B^{d} (the downwards equivalent density of B) by matching its induced potential with u_B^{d} (see Fig. 1(c)). All the matching processes use only the kernel evaluation, thus freeing us from the tedious analytic expansions. We notice that both the evaluation step and the inversion step of the matching processes can be represented as small matrices which are independent of the specific locations of the boxes since the kernel is translation invariant. Therefore, these matrices can be precomputed. Moreover, by choosing the support of the upwards equivalent density and the downwards check potential to be the same Cartesian grid of the box boundary, the fast Fourier transform (FFT) can be used to speed up the M2L translation which is usually the most expensive step of the algorithm (see [9] for details).

2.2. New algorithm for RBFs

In general, the kernel independent FMM in [9] works well only for the kernels derived from the second-order elliptic PDEs, since it relies on the results from the potential theory of these PDEs. However, the radial basis functions, such as $1/r^a$ (for arbitrary $a > 0$) and multiquadrics (such as $\sqrt{r^2+a^2}$ and $1/\sqrt{r^2+a^2}$) in general do not satisfy any elliptic PDE. Therefore, for these kernels, the *surface supported* equivalent densities are not capable of representing the true densities, and checking the potentials on a *surface* cannot guarantee the validness of the equivalent density approximation. As a result, we propose the following changes for the equivalent densities and translations to accommodate these RBF kernels.

Upwards equivalent density. For a box B, the upwards equivalent density ϕ_B^{u} in the new algorithm for RBFs is supported on an evenly spaced Cartesian grid of the box B (instead of a grid of ∂B in the case of the PDE

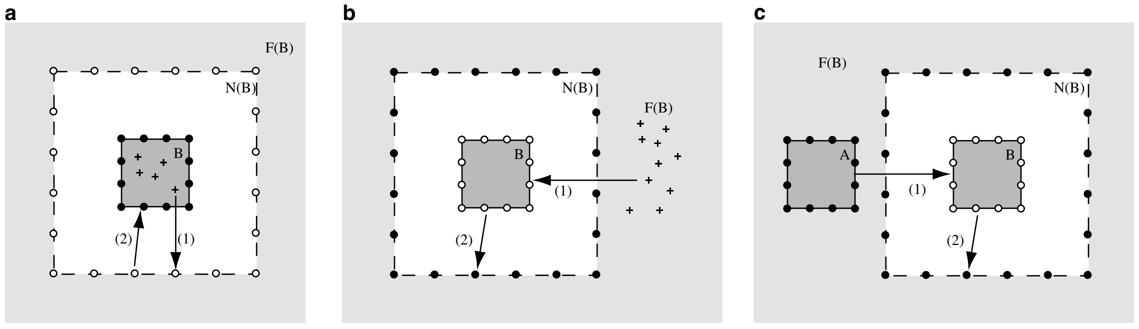

Fig. 1. The kernel independent FMM algorithm for PDE kernels in [9]. "+", "●" and "○" denote the locations of the source densities, the equivalent density and the check potential, respectively. (a) The upwards equivalent density of B and its construction. (b) The downwards equivalent density of B and its construction. (c) M2L translation from box A to box B. In every case, the computation of the equivalent densities has two steps: (1) the evaluation step; (2) the inversion step.

kernels). By increasing the size of the grid, we are able to achieve better approximations to the true densities in box B. Since all RBFs under consideration are real analytic away from its center, the potential in $F(B)$ induced by the densities in B is a real analytic function as well. We know that if two real analytic functions are matched on a set of non-zero measure, they are identical. This leads us to the following scheme to compute ϕ_B^{u}. Instead of matching the potentials on $\partial F(B)$, we require the potentials to agree on a corona in $F(B)$ adjacent to $\partial F(B)$. In our implementation, this corona is sampled regularly by a Cartesian grid, and the potentials are matched on these sampling points (see Fig. 2(a)). The numerical results in Section 3 show that this scheme computes the equivalent density in an accurate and stable way.

Downwards equivalent density. For a box B, the new downwards equivalent density ϕ_B^{d} is supported on the same corona used for the computation of ϕ_B^{u}. In our implementation, this region is sampled again by a Cartesian grid, and all the grid points inside the corona comprise the support of the downwards equivalent density ϕ_B^{d}. To compute ϕ_B^{d}, the potentials induced by the true densities and by ϕ_B^{d} are matched on an evenly spaced Cartesian grid of box B (see Fig. 2(b)). In fact, we choose this grid to be the same as the grid on which ϕ_B^{u} is supported in order to accelerate the M2L translation.

Translations. The M2M, M2L and L2L translations remain essentially the same except the changes of the support of the equivalent densities and the location to match the potentials. For example, for the M2L translation from box A to box B, ϕ_A^{u} is now supported on a Cartesian grid of box A, while the check potential u_B^{d} is located at a Cartesian grid of box B (see Fig. 2(c)). Since A and B are in the same level of the octree, these two Cartesian grids have the same size. Therefore, we can still use FFT to accelerate the M2L translation as in [9].

3. Numerical results

The proposed algorithm is implemented in C++ for both 2D and 3D cases. All the tests are performed on a PC with 2 GHz CPU. The numerical results are summarized in Tables 1–4.

2D accuracy test. Table 1 shows the accuracy test results of our algorithm on seven 2D RBF kernels. We fix the number of source densities N to be 80,000. For each kernel, we perform three tests with the upwards equivalent density supported on Cartesian grids of size 4×4, 6×6 and 8×8, where denser grid gives better accuracy. The support of the downwards equivalent density is refined accordingly. The column T_{fmm} gives the running time of our FMM type algorithm in seconds, while T_{dir} shows the estimated running time if the direct evaluation were used. The error is computed as follows. We randomly choose k points $\mathbf{x}_{t_i}$ where $i = 1,\ldots,k$. Suppose $\tilde{u}_{t_i}$ are the exact potentials computed from the direct evaluation, and u_{t_i} are the ones from our FMM algorithm. We estimate the error by $E = \sqrt{\frac{\sum_{i=1}^{k}|u_{t_i} - \tilde{u}_{t_i}|^2}{\sum_{i=1}^{k}|\tilde{u}_{t_i}|^2}}$. In all experiments, k is equal to 200. We observe that our algorithm behaves well for all seven kernels with the accuracy improved by a factor of 20–80 between two adjacent grid sizes.

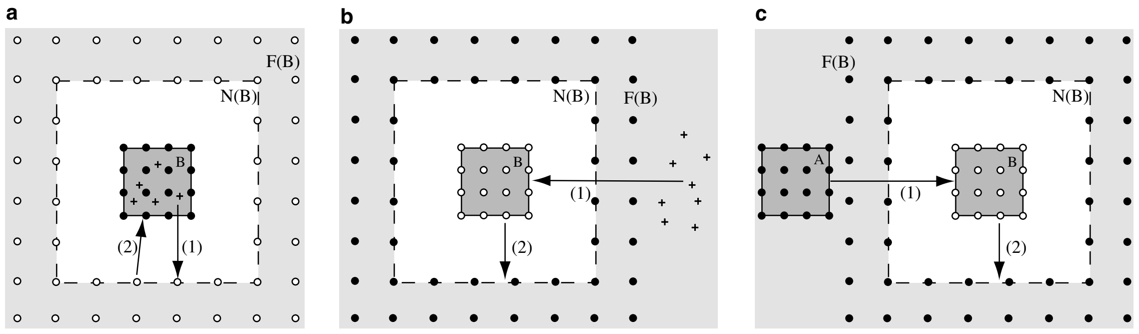

Fig. 2. The kernel independent FMM algorithm for RBFs. "+", "●" and "○" denote the locations of the source densities, the equivalent density and the check potential, respectively. (a) The upwards equivalent density of B and its construction. (b) The downwards equivalent density of B and its construction. (c) M2L translation from box A to box B.

Table 1
2D accuracy test

Kernel	Grid	T_{fmm} (s)	T_{dir} (s)	Error	Error ratio
$1/r^2$	4×4	0.93	200	9.688e − 05	
	6×6	1.69		1.266e − 06	76.536
	8×8	2.15		2.660e − 08	47.582
$1/r$	4×4	1.29	300	5.099e − 05	
	6×6	2.78		8.862e − 07	57.535
	8×8	3.31		3.270e − 08	27.099
$1/r^{0.5}$	4×4	1.81	432	1.024e − 05	
	6×6	4.26		2.345e − 07	43.646
	8×8	4.86		5.943e − 09	39.467
$\sqrt{r^2+0.01^2}$	4×4	1.72	432	6.851e − 06	
	6×6	4.09		3.054e − 07	22.435
	8×8	4.66		8.669e − 09	35.226
$\sqrt{r^2+0.1^2}$	4×4	1.72	432	9.122e − 06	
	6×6	4.12		3.213e − 07	28.396
	8×8	4.64		1.238e − 08	25.949
$1/\sqrt{r^2+0.01^2}$	4×4	1.71	420	4.047e − 05	
	6×6	4.09		6.714e − 07	60.279
	8×8	4.66		2.720e − 08	24.679
$1/\sqrt{r^2+0.1^2}$	4×4	1.70	424	3.404e − 05	
	6×6	4.06		4.141e − 07	82.214
	8×8	4.69		1.154e − 08	35.885

Table 2
2D complexity test

Kernel	N	T_{fmm} (s)	T_{fmm} ratio	T_{dir} (s)	Error
$1/r^2$	5000	0.05		0.8	7.160e − 05
	40,000	0.43	8.60	50.0	6.058e − 05
	320,000	3.21	7.47	3200.0	9.643e − 05
	2,560,000	29.60	9.22	203,008.0	1.439e − 04
$1/r^{0.5}$	5000	0.10		1.5	8.966e − 06
	40,000	1.05	10.50	108.0	1.048e − 05
	320,000	8.16	7.77	6896.0	2.345e − 05
	2,560,000	69.69	8.54	442,240.0	3.418e − 05
$\sqrt{r^2+0.01^2}$	5000	0.10		1.5	8.085e − 06
	40,000	0.99	9.90	108.0	1.032e − 05
	320,000	8.76	8.85	6864.0	2.164e − 05
	2,560,000	66.67	7.61	437,504.0	2.201e − 05
$1/\sqrt{r^2+0.01^2}$	5000	0.11		1.8	5.019e − 05
	40,000	0.99	9.00	104.0	4.894e − 05
	320,000	8.67	8.76	6688.0	1.224e − 04
	2,560,000	66.06	7.62	426,112.0	1.221e − 04

2D complexity test. Table 2 gives the results on problems with increasing size (N from 5000 to 2,560,000) for four different 2D RBF kernels. We fix the Cartesian grid used for the upwards equivalent density to be 4×4. We observe that the complexity of our algorithm is linear with respect to the data size. Moreover, the errors remain at the same level of 10^{-4} for all runs.

3D accuracy test. Table 3 summarizes the accuracy test results of our algorithm on seven 3D kernels. Since the kernel $1/r$ used in 2D test is now the fundamental solution of the Laplace equation, we replace it with $1/r^{1.5}$. The achieved accuracy is controlled again by the grid of the upwards equivalent density. We use the

Table 3
3D accuracy test

Kernel	Grid	T_{fmm} (s)	T_{dir} (s)	Error	Error ratio
$1/r^2$	$4\times4\times4$	7.22	208	9.916e − 05	
	$6\times6\times6$	11.35		1.352e − 06	73.348
	$8\times8\times8$	19.45		2.477e − 08	54.589
$1/r^{1.5}$	$4\times4\times4$	15.61	480	1.990e − 04	
	$6\times6\times6$	25.79		2.289e − 06	86.911
	$8\times8\times8$	35.83		4.797e − 08	47.721
$1/r^{0.5}$	$4\times4\times4$	14.68	456	1.455e − 04	
	$6\times6\times6$	24.24		2.758e − 06	52.748
	$8\times8\times8$	34.13		5.357e − 08	51.485
$\sqrt{r^2+0.01^2}$	$4\times4\times4$	9.30	276	9.260e − 05	
	$6\times6\times6$	13.35		1.115e − 06	83.052
	$8\times8\times8$	17.84		1.261e − 08	88.386
$\sqrt{r^2+0.1^2}$	$4\times4\times4$	9.27	276	8.735e − 05	
	$6\times6\times6$	13.47		1.100e − 06	79.390
	$8\times8\times8$	17.83		2.131e − 08	51.630
$1/\sqrt{r^2+0.01^2}$	$4\times4\times4$	10.76	320	2.131e − 04	
	$6\times6\times6$	15.96		2.246e − 06	94.896
	$8\times8\times8$	20.50		4.968e − 08	45.198
$1/\sqrt{r^2+0.1^2}$	$4\times4\times4$	10.72	320	2.198e − 04	
	$6\times6\times6$	15.96		3.243e − 06	67.781
	$8\times8\times8$	20.53		6.321e − 08	51.313

Table 4
3D complexity test

Kernel	N	T_{fmm} (s)	T_{fmm} ratio	T_{dir} (s)	Error
$1/r^2$	5000	0.28		0.9	5.316e − 05
	40,000	2.44	8.71	54.0	4.706e − 05
	320,000	21.41	8.77	3328.0	5.594e − 05
	2,560,000	179.04	8.36	212,736.0	6.213e − 05
$1/r^{0.5}$	5000	0.60		1.8	7.646e − 05
	40,000	5.67	9.45	114.0	2.774e − 05
	320,000	50.06	8.83	7280.0	9.531e − 05
	2,560,000	421.39	8.42	463,232.0	9.375e − 05
$\sqrt{r^2+0.01^2}$	5000	0.36		1.1	4.441e − 05
	40,000	3.34	9.28	72.0	1.978e − 05
	320,000	29.09	8.71	4400.0	5.418e − 05
	2,560,000	245.05	8.42	282,496.0	1.019e − 04
$1/\sqrt{r^2+0.01^2}$	5000	0.41		1.5	1.454e − 04
	40,000	3.97	9.68	80.0	9.611e − 05
	320,000	34.79	8.76	5088.0	1.954e − 04
	2,560,000	292.21	8.40	323,968.0	2.390e − 04

Cartesian grids of size $4\times4\times4$, $6\times6\times6$ and $8\times8\times8$. We observe that the error improves by a factor of 50–90 between adjacent grid sizes.

3D complexity test. Table 4 gives the results on 3D problems with increasing size (N again from 5000 to 2,560,000). We fix the Cartesian grid used for equivalent density to be $4\times4\times4$. The complexity of our algorithm is proportional to the size of the problem, and the errors remain almost independent of the data size.

Acknowledgments

The author thanks G. Biros, E. Candès, L. Demanet and D. Zorin for helpful discussions.

References

[1] R.K. Beatson, J.B. Cherrie, D.L. Ragozin, Fast evaluation of radial basis functions: methods for four-dimensional polyharmonic splines, SIAM J. Math. Anal. 32 (6) (2001) 1272–1310 (electronic).

[2] R.K. Beatson, W.A. Light, Fast evaluation of radial basis functions: methods for two-dimensional polyharmonic splines, IMA J. Numer. Anal. 17 (3) (1997) 343–372.

[3] R.K. Beatson, G.N. Newsam, Fast evaluation of radial basis functions: moment-based methods, SIAM J. Sci. Comput. 19 (5) (1998) 1428–1449 (electronic).

[4] J.B. Cherrie, R.K. Beatson, G.N. Newsam, Fast evaluation of radial basis functions: methods for generalized multiquadrics in $\mathbb{R}^n$, SIAM J. Sci. Comput. 23 (5) (2002) 1549–1571 (electronic).

[5] Z. Gimbutas, L. Greengard, M. Minion, Coulomb interactions on planar structures: inverting the square root of the Laplacian, SIAM J. Sci. Comput. 22 (6) (2000) 2093–2108 (electronic).

[6] L. Greengard, V. Rokhlin, A fast algorithm for particle simulations, J. Comput. Phys. 73 (2) (1987) 325–348.

[7] L. Greengard, J. Strain, A fast algorithm for the evaluation of heat potentials, Comm. Pure Appl. Math. 43 (8) (1990) 949–963.

[8] L. Greengard, J. Strain, The fast Gauss transform, SIAM J. Sci. Stat. Comput. 12 (1) (1991) 79–94.

[9] L. Ying, G. Biros, D. Zorin, A kernel-independent adaptive fast multipole algorithm in two and three dimensions, J. Comput. Phys. 196 (2) (2004) 591–626.