Publl. RIMS, Kyoto Univ.
39 (2003), 785–796

A Remark on the Fast Gauss Transform

By

Kenta Kobayashi*

Abstract

We propose an improvement on the Fast Gauss Transform which was presented by Greengard and Sun [Documenta Mathematica, Extra volume ICM 1998, **III**, pp.575-584(1998)]. In their method, plane waves are used to approximate the Gauss kernel. Plane waves they used were generated by the Fourier integral and the trapezoidal rule. We propose different plane waves, which enables us to calculate the Fast Gauss Transform more efficiently.

§1. Introduction

The Gauss transform

$$G_\delta f(x) = (\pi\delta)^{-d/2} \int_\Gamma e^{-|x-y|^2/\delta} f(y)dy \qquad (\delta > 0\ ),$$

where δ is a given parameter and Γ is a prescribed subset of $\mathbb{R}^d$, appears in many problems of mathematics and applied mathematics, such as initial/boundary value problems for the heat equation[1]. Accordingly its discrete versions are of importance in numerical analysis.

In this paper, we consider the following discrete Gauss transform

$$G(x) = \sum_{j=1}^{N} q_j e^{-|x-s_j|^2/\delta},$$

Communicated by H. Okamoto. Received August 13, 2002. Revised February 24, 2003.
2000 Mathematics Subject Classification(s): 65R10, 33B10, 35K05
Key words: Fast Gauss Transform, fast algorithms
Partially supported by Japan Society for Promotion of Science #3314
*RIMS, Kyoto University, Kyoto 606-8502, Japan.

where coefficients $q_j \in \mathbb{R}$ and source location $s_j \in \mathbb{R}^d$ are given. The problem is, then, to compute $G(x)$ at prescribed points $x_j \in \mathbb{R}^d$ $(j = 1, 2, \ldots, M)$.

If we are required to evaluate $G(x)$ at M target points, the number of operations to calculate them directly is $O(NM)$. If both M and N are large, the cost of computation can be very expensive. To deal with this difficulty, Greengard and Strain invented a Fast Gauss Transform using the Hermite polynomials. This algorithm can calculate the discrete Gauss transform with $O(N+M)$ work by admitting certain acceptable errors [2]. In 1998, Greengard and Sun [3] proposed a new version of the Fast Gauss Transform, which was significantly faster than the previous version. The method of [2] is based on the approximation of the Gauss Kernel by the Hermite expansion. The algorithm of [3], however, approximates the Gauss Kernel by the plane waves, which enabled them to have quite an important.

In this paper, we propose a further improvement on the Fast Gauss Transform in [3]. We outline in section 2 the Fast Gauss Transform in [3]. Our improvement is presented in section 3. Effectiveness of our method is illustrated in section 4 by numerical example. Conclusions are given in section 5.

§2. An Outline of the Fast Gauss Transform by Greengard and Sun

In this section, we outline their new version of the Fast Gauss Transform. The reader should consult [3] for details.

The fast Gauss transform is an analysis-based fast algorithm like the fast multipole methods for the Laplace and Helmholtz equations [4]. The difference between analysis-based fast algorithms and others like the Fast Fourier Transform is that the former accompanies a truncation error, while the latter is exact. Although an error exists, it can be made arbitrarily small. Further, analysis-based algorithms can be used in a variety of problems in which FFT alone is of limited use.

Their algorithm can be used in arbitrary dimensions. However, for the sake of brevity, we deal with only the one-dimensional case.

They divide the interval containing all the source and target points into subintervals of length $c\sqrt{\delta}$ (they use $\sqrt{\delta}$ in their paper but we added c for generalization) denoted by $I_1, I_2, \ldots, I_L$ with center $u_1, u_2, \ldots, u_L$, respectively.

Their method is based on an approximation of the Gauss kernel by plane waves. The Gauss kernel is approximated by a linear combination of plane waves as follows:

$$E_j(x) = \begin{cases} \displaystyle\sum_{k=1}^{P} a_k e^{ib_k(x-s_j)/\sqrt{\delta}} & (\ |l-w| \le r \ \text{ where } \ s_j \in I_l,\ x \in I_w\ ) \\ 0 & (\ |l-w| > r \ \text{ where } \ s_j \in I_l,\ x \in I_w\ ), \end{cases}$$

where r is a given integer and specifies the range of approximation by plane waves. The accuracy of this approximation is based on how to choose parameters P, a_k and b_k. We describe it later.

In this case,

$$\left| e^{-|x-s_j|^2/\delta} - E_j(x) \right| \le \max(\varepsilon_1, \varepsilon_2)$$

where

$$\varepsilon_1 = \max_{|x| \le (r+1)c} \left| e^{-x^2} - \sum_{k=1}^{P} a_k e^{ib_k x} \right|, \quad \varepsilon_2 = e^{-(cr)^2}. \tag{2.1}$$

Accordingly, $G(x)$ is approximated by $\sum_{j=1}^{N} q_j E_j(x) = \widetilde{G}(x)$ as

$$\left| G(x) - \widetilde{G}(x) \right| \le Q \max(\varepsilon_1, \varepsilon_2) \quad \text{where} \quad Q = \sum_{j=1}^{N} |q_j|.$$

At first, we calculate the partial sum of $E_j(x)$ at every intervals

$$\sum_{s_j \in I_l} q_j E_j(x) = \begin{cases} \displaystyle\sum_{k=1}^{P} A_{lk} e^{ib_k(x-u_l)/\sqrt{\delta}} & (|l-w| \le r \ \text{ where } \ s_j \in I_l,\ x \in I_w) \\ 0 & (|l-w| > r \ \text{ where } \ s_j \in I_l,\ x \in I_w), \end{cases}$$

where coefficients A_{lk} are given by

$$A_{lk} = \sum_{s_j \in I_l} q_j a_k e^{ib_k(u_l - s_j)/\sqrt{\delta}}.$$

Then, we can calculate $\widetilde{G}(x)$ as

$$\widetilde{G}(x) = \sum_{\substack{|l-w| \le r \\ s_j \in I_l}} q_j E_j(x) = \sum_{k=1}^{P} B_{wk} e^{ib_k(x-u_w)/\sqrt{\delta}} \quad (x \in I_w),$$

where coefficients B_{wk} is given by

$$B_{wk} = \sum_{|l-w|\le r} A_{lk} e^{ib_k(u_w-u_l)/\sqrt{\delta}}.$$

The amount of work required for this Fast Gauss Transform is

$$O(S\sqrt{\delta}^{-1}c^{-1}P) + O(PN) + O(PM).$$

$O(PN)$ is to calculate $\{A_{lk}\}$ from $\{q_j\}$, $O(S\sqrt{\delta}^{-1}c^{-1}P)$ is to calculate $\{B_{wk}\}$ form $\{A_{lk}\}$, and $O(PM)$ is to evaluate $G(x_j)$ from $\{B_{lk}\}$, where S is the length of the smallest interval containing all the source and target points (See [3] about the method of calculating B_{lk} not by $O(rS\sqrt{\delta}^{-1}c^{-1}P)$ but by $O(S\sqrt{\delta}^{-1}c^{-1}P)$).

From (2.1), it is necessary to make product cr large according to the accuracy we assume and it turns out that r and P have the relation of a trade off.

Greengard and Sun determined a_k and b_k by means of the Fourier relation

$$e^{-x^2} = \frac{1}{2\sqrt{\pi}} \int_{-\infty}^{\infty} e^{-\xi^2/4} e^{-ix\xi} d\xi$$

in the following way: since the integrand is rapidly decaying when $|\xi|$ grows, the trapezoidal rule gives a good approximation. Applying the trapezoidal rule of an interval $2K/(P-1)$ and truncate to P terms, we have

$$e^{-x^2} \approx \frac{K}{\sqrt{\pi}(P-1)} \sum_{k=1}^{P} e^{-\xi_k^2/4} e^{-i\xi_k x} \quad \text{where} \quad \xi_k = \frac{K}{P-1}(2k-P-1).$$

Accuracy will become good if K and P becomes large.

Namely this definition implies that

$$a_k = \frac{K}{\sqrt{\pi}(P-1)} e^{-\xi_k^2/4}, \quad b_k = -\xi_k.$$

If the dimension d is greater than one, the Gauss kernel

$$e^{-|x|^2/\delta} = e^{-(x_1^2+x_2^2+\cdots+x_d^2)/\delta} = e^{-x_1^2/\delta}\, e^{-x_2^2/\delta} \,\ldots\, e^{-x_d^2/\delta}$$

is approximated by

$$\left(\sum_{k=1}^{P} a_k e^{ib_k x_1/\sqrt{\delta}}\right) \left(\sum_{k=1}^{P} a_k e^{ib_k x_2/\sqrt{\delta}}\right) \cdots \left(\sum_{k=1}^{P} a_k e^{ib_k x_d/\sqrt{\delta}}\right).$$

In this case, the amount of work required for this Fast Gauss Transform is

$$O(dS\sqrt{\delta}^{-d}c^{-d}P^d) + O(P^dN) + O(P^dM).$$

where S is the Lebesgue measure of rectangles which contain all source points and target points.

Greengard and Sun took $K = 7.5, P = 23, r = 4, c = 1$ for single floating precision and $K = 12, P = 47, r = 6, c = 1$ for double floating precision.

§3. Improvement

As we have seen, good approximation of the Gauss kernel by plane waves is essential in their scheme. So, we can make this scheme more efficient if we can choose suitable a_k and b_k to smaller P and larger c. We construct such an approximation using Newton's Method as follows.

Given the approximate relation

$$e^{-x^2} \approx \sum_{k=1}^{P} a_k e^{ib_k x},$$

where we assume that $a_k = a_{P+1-k},\ b_k = -b_{P+1-k}$ $(k = 1, 2, \dots, P)$, we define

$$f(a, x) := e^{-x^2} - \sum_{k=1}^{P} a_k e^{ib_k x}$$

where

$$a = \begin{cases} (a_1, a_2, \dots, a_{P/2}, b_1, b_2, \dots, b_{P/2}) & (P : \text{even}) \\ (a_1, a_2, \dots, a_{(P+1)/2}, b_1, b_2, \dots, b_{(P-1)/2}) & (P : \text{odd}). \end{cases}$$

We look for those a which makes $f(a, \cdot)$ be sufficiently small. In order to find such an a with minimal labor, we look, in addition to a, for P points $t_0, t_1, \dots, t_{P-1}$, which are candidates for the extremal points of $f(a, \cdot)$. Taking Chebyshev's theorem (see, e.g., [5]) in mind, we then look for a and $t = (t_0, t_1, \dots, t_{P-1})$ simultaneously in the way that the absolute values

$$f(a, t_0) - \varepsilon, \quad f(a, t_1) + \varepsilon, \quad f(a, t_2) - \varepsilon, \ \dots$$

be minimal.

We first fix P and set

$$\varepsilon = 0.01, \quad K = 3.0,$$

$$a_k^{(0)} = \frac{K}{\sqrt{\pi}(P-1)} e^{-\xi_k^2/4}, \quad b_k^{(0)} = -\xi_k \quad \text{where} \quad \xi_k = \frac{K}{P-1}(2k-P-1),$$

$$t_k^{(0)} = \frac{\pi}{\xi_{P+1}} k \qquad (k = 0, 1, \dots, P-1),$$

$$a^{(n)} = (a_1^{(n)}, \dots, b_1^{(n)}, \dots),$$

$$t^{(n)} = \left(t_0^{(n)}, t_1^{(n)}, \dots, t_{P-1}^{(n)} \right).$$

And then, the following iterations are employed:

$$a^{(n+1)} = a^{(n)} - \lambda_1 J\left(a^{(n)}, t^{(n)}\right)^{-1} F\left(a^{(n)}, t^{(n)}\right),$$

$$t_k^{(n+1)} = t_k^{(n)} - \lambda_2 \frac{\dfrac{\partial}{\partial x} f(a^{(n+1)}, t_k^{(n)})}{\dfrac{\partial^2}{\partial x^2} f(a^{(n+1)}, t_k^{(n)})}.$$

where

$$F(a, t^{(n)}) = \left(f(a, t_0^{(n)}) - \varepsilon,\ f(a, t_1^{(n)}) + \varepsilon,\ \dots\ ,\ f(a, t_{P-1}^{(n)}) - (-1)^{P-1}\varepsilon \right)^t$$

and $J(a,t)$ is the Jacobi matrix of $F(a,t)$ about a. Positive constants λ_1 and λ_2 are smaller than 1, but we have to adjust them appropriately to have a good convergence. If $a^{(n)}$ and $t^{(n)}$ have converged, we let ε decrease a little and repeat the same iteration. When ε have decreased below an initially given tolerance, we stop iteration.

Initial condition $\varepsilon = 0.01, K = 3.0$ worked well with $P \leq 60$.

We considered three cases: Case 1 $\sim$ Case 3. Table 1 shows the conditions of these cases. We decided P, c and r so that ε_1 and ε_2 are the same or less than Greengard and Sun's method.

Graphs of $f(a,x)$ are drawn in Fig 1. The graph on the bottom is the one used in [3]. Note that the number of the extrema is much smaller than P. In view of the theory of the best approximation(Chebyshev's theorem[5]), this explains why their method can further be improved. In fact, our $f(a,x)$ shows a better quality in view of the number of extrema and P.

Remark. We described the method for single floating precision. The same thing is possible also for double floating precision. However, in order to prevent

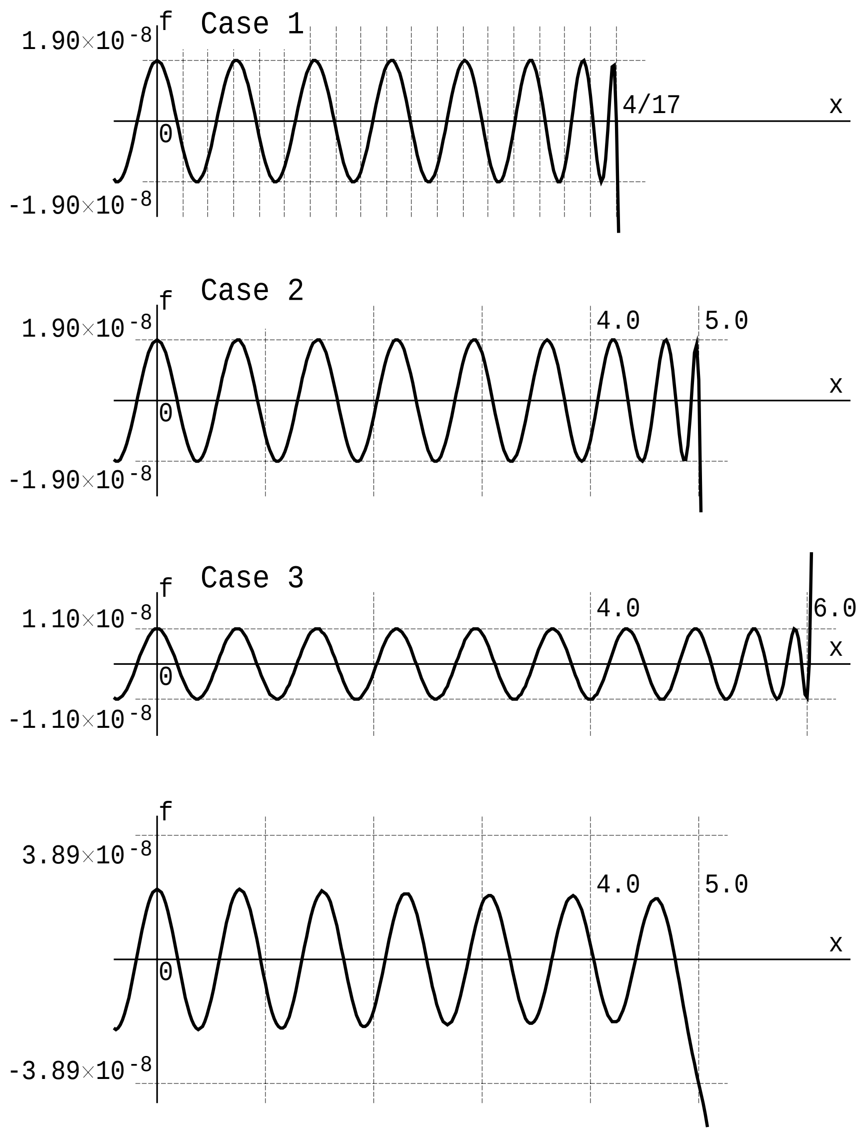

Figure 1. The graphs of $f(a, x)$. $(P, r, c) = (23, 4, 1.0)$ proposed by Greengard and Sun (bottom).

cancellation error in that case, high accuracy, using quad floating precision etc. is required for calculation of $f(a, \cdot)$.

§4. Numerical Results

We carried out the discrete Gauss transform in these cases and compared CPU time and precision. $d = 2$, $S = [0, 10] \times [0, 10]$, $M = N = 30000$ were applied.

We take s_j and x_j at random in S and also take q_j at random in $[-1, 1]$ by uniform distribution, where random numbers are generated by c++ function rand() on Solaris8(Ultra Spark 60).

Tables 2 through 5 show the condition when $\delta = 1.0$, $\delta = 0.1$, $\delta = 0.01$ and $\delta = 0.001$. When $\delta = 0.001$, we could not calculate the Case 1 because the number of mesh points was too large.

If δ is relatively large, computation time depends more on P than on c. Hence Case 1 gives the best result in this case. On the other hand, c is more dominant than P, if δ is small. In this case, Case 3 gives the best result.

Tables 6-9 are lists of a_k and b_k which was used in case 1 $\sim$ case 3 and the method that Greengard and Sun presented, respectively.

	P	c	r	ε_1	ε_2
Case 1	15	4/17	17	1.90×10^{-8}	1.13×10^{-7}
Case 2	17	1	4	1.90×10^{-8}	1.13×10^{-7}
Case 3	20	2	2	1.10×10^{-8}	1.13×10^{-7}
Greengard and Sun's method	23	1	4	3.89×10^{-8}	1.13×10^{-7}

Table 1. A list of conditions

$\delta = 1.0$	CPU time (sec)	maximum error
Case 1	32	2.24×10^{-6}
Case 2	50	2.17×10^{-6}
Case 3	77	1.43×10^{-6}
Greengard and Sun's method	93	3.16×10^{-6}
Direct calculation	671	

Table 2. Numerical result when $\delta = 1.0$

These results show that any of Case 1 to Case 3 can achieve calculation faster than Greengard and Sun's method with a smaller error.

Remark. There are considerable differences in direct calculation by various δ. It is not caused by the difference in the speed of a computer but on the influence of internal algorithm of the exponential function.

$\delta = 0.1$	CPU time (sec)	maximum error
Case 1	47	1.02×10^{-6}
Case 2	51	1.87×10^{-6}
Case 3	77	1.36×10^{-6}
Greengard and Sun's method	96	2.61×10^{-6}
Direct calculation	1149	

Table 3. Numerical result when $\delta = 0.1$

$\delta = 0.01$	CPU time (sec)	maximum error
Case 1	186	3.02×10^{-7}
Case 2	62	3.39×10^{-7}
Case 3	81	1.32×10^{-6}
Greengard and Sun's method	116	2.67×10^{-6}
Direct calculation	1427	

Table 4. Numerical result when $\delta = 0.01$

$\delta = 0.001$	CPU time (sec)	maximum error
Case 1	—	—
Case 2	163	1.08×10^{-7}
Case 3	117	1.38×10^{-6}
Greengard and Sun's method	300	2.60×10^{-6}
Direct calculation	1280	

Table 5. Numerical result when $\delta = 0.001$

$a_1 = 1.5024412350542115 \times 10^{-7}$	$b_1 = 7.5$
$a_2 = 1.7247900894047711 \times 10^{-6}$	$b_2 = 6.8181818181818182$
$a_3 = 1.5693812286707054 \times 10^{-5}$	$b_3 = 6.1363636363636364$
$a_4 = 1.1318115235748990 \times 10^{-4}$	$b_4 = 5.4545454545454541$
$a_5 = 6.4695373000284346 \times 10^{-4}$	$b_5 = 4.7727272727272727$
$a_6 = 2.9310672496119460 \times 10^{-3}$	$b_6 = 4.0909090909090909$
$a_7 = 1.0525234889691936 \times 10^{-2}$	$b_7 = 3.4090909090909091$
$a_8 = 2.9956513229246879 \times 10^{-2}$	$b_8 = 2.7272727272727273$
$a_9 = 6.7577820416272305 \times 10^{-2}$	$b_9 = 2.0454545454545455$
$a_{10} = 0.12082882446808624$	$b_{10} = 1.3636363636363636$
$a_{11} = 0.17123414603808165$	$b_{11} = 0.6818181818181818$
$a_{12} = 0.19233735802764421$	$b_{12} = 0.0$
$a_k = a_{24-k}$ $(13 \leq k \leq 23)$	$b_k = -b_{24-k}$ $(13 \leq k \leq 23)$

Table 6. Lists of a_k and b_k which is used in Greengard and Sun's method

$a_1 = 3.1602158548065147 \times 10^{-6}$	$b_1 = 6.8323332716494347$
$a_2 = 1.0774164008553449 \times 10^{-4}$	$b_2 = 5.6445838356737017$
$a_3 = 1.4414987705394192 \times 10^{-3}$	$b_3 = 4.5966981140329235$
$a_4 = 1.0150979166018934 \times 10^{-2}$	$b_4 = 3.6198358194646820$
$a_5 = 4.2736758775148202 \times 10^{-2}$	$b_5 = 2.6854056286373478$
$a_6 = 0.11507861467068439$	$b_6 = 1.7773883441184946$
$a_7 = 0.20581846798610981$	$b_7 = 0.88502778008267924$
$a_8 = 0.24932553855111778$	$b_8 = 0.0$
$a_k = a_{16-k}$ $(9 \leq k \leq 15)$	$b_k = -b_{16-k}$ $(9 \leq k \leq 15)$

Table 7. Lists of a_k and b_k which is used in Case 1

§5. Conclusions

We found plane waves that furnish better approximation for the Fast Gauss transform. As we can see from numerical results in previous section, we could make the Fast Gauss Transform, in its best, several times as faster as the method of [3].

We listed up coefficients of the plane waves at table 6 $\sim$ table 9, so anybody can use the faster Fast Gauss Transform immediately.

$a_1 = 1.4480712327712938 \times 10^{-6}$	$b_1 = 7.0254411771533647$
$a_2 = 4.0179072932667429 \times 10^{-5}$	$b_2 = 5.9521675353096670$
$a_3 = 5.0584010513166024 \times 10^{-4}$	$b_3 = 4.9960582231137938$
$a_4 = 3.6652357944413679 \times 10^{-3}$	$b_4 = 4.1019533388132361$
$a_5 = 1.7025703894623791 \times 10^{-2}$	$b_5 = 3.2463293077293978$
$a_6 = 5.3812559765011703 \times 10^{-2}$	$b_6 = 2.4160036679818364$
$a_7 = 0.11992005473576491$	$b_7 = 1.6022885299976992$
$a_8 = 0.19248201399667833$	$b_8 = 0.79872749553235256$
$a_9 = 0.22509391012836563$	$b_9 = 0.0$
$a_k = a_{18-k}$ $(10 \leq k \leq 17)$	$b_k = -b_{18-k}$ $(10 \leq k \leq 17)$

Table 8. Lists of a_k and b_k which is used in Case 2

$a_1 = 4.2408195477813319 \times 10^{-7}$	$b_1 = 7.3296389251749705$
$a_2 = 9.4942307284900443 \times 10^{-6}$	$b_2 = 6.3821524074400342$
$a_3 = 1.1167657306883446 \times 10^{-4}$	$b_3 = 5.5282750548220214$
$a_4 = 8.2920394539577461 \times 10^{-4}$	$b_4 = 4.7264208710780498$
$a_5 = 4.2416043852381907 \times 10^{-3}$	$b_5 = 3.9581421510087544$
$a_6 = 1.5718392795466322 \times 10^{-2}$	$b_6 = 3.2130243227184327$
$a_7 = 4.3555801883621612 \times 10^{-2}$	$b_7 = 2.4842727384019905$
$a_8 = 9.2138495635379181 \times 10^{-2}$	$b_8 = 1.7669779404439534$
$a_9 = 0.15081565250910717$	$b_9 = 1.0572888477677547$
$a_{10} = 0.19257924846003965$	$b_{10} = 0.35195633263387421$
$a_k = a_{21-k}$ $(11 \leq k \leq 20)$	$b_k = -b_{21-k}$ $(11 \leq k \leq 20)$

Table 9. Lists of a_k and b_k which is used in Case 3

Acknowledgement

I would like to express my deep gratitude to Prof. H. Okamoto in carefully reading the manuscript and giving many useful suggestions to improve the paper.

References

[1] Friedman, A., *Partial Differential Equations of Parabolic Type*, Prentice-Hall, New Jersey, 1964.

[2] Greengard, L. and Strain, J., The fast Gauss transform, *SIAM J. Sci. Comput.*, **12**(1991), 79-94.

[3] Greengard, L., and Sun, X., A new version of the fast Gauss transform, Doc. Math., Extra volume ICM 1998, **III** (1998), 575-584.
[4] Greengard, L. and Rokhlin, V., A new version of the fast multiple method for the Laplace equation in three dimensions, *Acta Numer.*, **6**(1987), 229-269.
[5] Rice, J. R., *The approximation of functions*, I, II, Addison-Wesley, 1964-69.