Multilevel Matrix Multiplication and Fast Solution of Integral Equations[1]

A. Brandt, A.A. Lubrecht
Department of Applied Mathematics and Computer Science
The Weizmann Institute of Science
Rehovot, 76100, Israel

Abstract

A fast multigrid approach is described for the task of calculating $\int_\Omega K(x,y)u(y)dy$ for each $x \in \Omega \subseteq I\!R^d$. Discretizing Ω by an equidistant grid with n points and meshsize h, and approximating the integrations to $O(h^{2s})$ accuracy, it is shown that the complexity of this calculation can be reduced from $O(n^2)$ to $O(sn)$, provided the kernel K is sufficiently smooth. For potential-type kernels, the complexity is reduced to $O(sn \log n)$. Corresponding integral equations can be solved to a similar accuracy in basically the same amount of work, using a special kind of distributed relaxation in a multigrid algorithm. One and two dimensional numerical tests, and theoretical derivations of optimal strategies, are reported. The method is applicable to the task of multiplying by any matrix with appropriate smoothness properties, including most types of many body interactions.

Table of Content

[1] This research was sponsored by the Air-Force Office of Scientific Research, United States Air Force under grant numbers AFOSR-86-0126 and AFOSR-86-0127, by the Dutch 'Niels Stensen Stichting' and by the National Science Foundation under grant number NSF DMS-8704169.

1. Introduction

In this work we will describe an approach to reduce the complexity of *multi-integration*. By multi-integration (with kernel K, over domain Ω) we mean the task of calculating the function

$$w(x) = \int_\Omega K(x,y)u(y)dy, \qquad x \in \Omega \subseteq I\!R^d \tag{1}$$

given the function u. The discrete analog of this task is the multiplication of a vector by a dense (not sparse) matrix having certain smoothness properties. Such numerical tasks arise in many important problems in mathematics, physics and engineering, including: integro-differential equations, integral equations, panel methods, boundary element methods, plasma physics, problems in elasticity, gravitating masses, vortex schemes, coulombic molecular interactions, and other many-body long range interactions.

When the domain Ω is discretized by a grid with meshsize h and $n = O(h^{-d})$ points, the calculation of a single integral will cost $O(n)$ operations, thus the calculation of $w(x)$ for each of the n gridpoints will require $O(n^2)$ operations. Since for various problems a large number of points n is essential, it is our aim to reduce the complexity of this multi-integration, in order to avoid excessive computing times. This can be obtained by performing part of the integration on coarser grids, in a way that keeps the added error smaller than the original fine grid discretization error, by exploiting the smoothness of the kernel K. Specifically, if the kernel K has $4s$ bounded derivatives, we will show that the multi-integration can be calculated to accuracy $\varepsilon = O(h^{2s}) = O(n^{-2s/d})$ in $O(sn)$ operations. For a wider class of kernels, including the potential type, the number of required operations will be shown to be $O(sn \log n) = O\big(n \log \frac{1}{\varepsilon}\big)$.

The basic idea of the method was outlined before in [4, §8.6] and a brief description was given in [6]. Similar attempts to reduce the complexity of multi-integration have been reported by Rokhlin [13], Nowak and Hackbusch [10] and other related approaches existed earlier (see survey in [1]). Relevant references also include [2], [3], [7] and [9], which use hierarchical solvers for many-body simulations, and [11] and [12], which exhibit FFT-based schemes for the solution of integral equations. All these approaches, however, are either of limited accuracy or restricted to potential-type kernels, for which far field expansions are used in order to obtain the desired reduction in complexity. In the method presented below, the smoothness of K is exploited generally and directly, by replacing some of its values by interpolations from coarser grids. High-order accuracy is obtained by high order interpolations, requiring no potential theory (which is indeed unavailable in various physical systems).

The multigrid solver for integral equations described here (Sec. 5) introduces several additional algorithmic innovations, such as distributed relaxation schemes of arbitrary "order", and ways to use multi-integrations very sparingly. In fact, the solution, to the accuracy of the discretization error, can be obtained in an amount of work only a fraction more than that of *one* multi-integration. For potential-type kernels, a solution to accuracy $\varepsilon = O(h^{2s}) = O(n^{-2s/d})$ requires $O\big(sn(s+\log n)\big) = O\big(n \log \frac{1}{\varepsilon}\big)$ operations, compared to $O\big(n\big(\log \frac{1}{\varepsilon}\big)^3\big)$ operations in [13]. (There is usually no point in solving the discrete equations to any accuracy ε substantially below the $O(h^{2s})$ discretization error; but if one still wants to do so, the approach presented here would require $O\big(n\big(\log \frac{1}{\varepsilon}\big)^2\big)$ operations, while the complexity of [13] would remain $O\big(n\big(\log \frac{1}{\varepsilon}\big)^3\big)$. Thus, if the desired precision ε is *fixed* — at the machine precision, for example — both methods have $O(n)$ complexity. An extra $\log n$ factor appears in the complexity of the present method, and a $(\log n)^3$ factor would similarly appear for the method of [3], if the relation between the *needed* precision and the discretization error is accounted for.)

A forthcoming work, on implementing the present approach for the fast calculation of many body interactions and their steady states, is briefly described in Sec. 6.2.

The range of applicability of the methods proposed in this article, and the range of problems for which the above cited efficiency is obtained, are discussed in Sec. 6.3.

2. Discretization

Let $x_i^h = x_0 + ih$ be equidistant gridpoints in Ω, where $i = (i_1, i_2, \ldots, i_d)$ is a vector of integers and h is the meshsize. The discrete functions approximating u and w on this grid will be denoted by $u_i^h = u^h(x_i^h)$ and $w_i^h = w^h(x_i^h)$.

Approximating the function u by a piecewise polynomial function $\hat{u}^h$, of degree $2s-1$, interpolated from the grid values $\hat{u}^h(x_j^h) = u_j^h$, coefficients $K_{i,j}^{hh}$ can be calculated such that

$$w_i^h \underset{\text{def}}{=} \int_\Omega K(x,y)\hat{u}^h(y)dy = h^d \sum_j K_{i,j}^{hh} u_j^h. \tag{2}$$

In case $u_j^h = u(x_j^h)$ and u is sufficiently smooth, it follows from (1) and the theory of polynomial interpolations that $w(x_i) = w_i^h + O(h^{2s}|u|_{2s})$, where $|u|_{2s}$ is an upper bound for the $2s$-order derivatives of u. The values of the coefficients $K_{i,j}^{hh}$, which approximate $K(x_i^h, x_j^h)$, can often be calculated by analytical integrations (product integration, see for instance Young [15]); this is especially important near kernel singularities. Just computing all the coefficients $K_{i,j}^{hh}$ would require $O(sn^2)$ operations for a general kernel K, but, as we will see below, only few coefficients ($O(sn \log n)$ in case of a potential-type kernel) will be needed on the finest level.

2.1 Notation

In the algorithms below, we will use a coarser grid with meshsize $H = 2h$. (Other values of H/h could also be used, but are less effective and/or less convenient.) The running index on that grid will generally be denoted by capital letters; e.g., $u_J^H = u^H(x_J^H)$ is the value of the coarse grid function u^H at the coarse grid point $x_J^H = x_0 + JH$. The points x_J^H are thus, for simplicity, chosen to coincide with fine grid points, satisfying $x_J^H = x_{2J}^h$. A notation like $K_{i,J}^{hH}$ will stand for a discrete kernel whose first index is in the fine grid and the second in the coarse grid, approximating $K(x_i^h, x_J^H)$.

We will use $I\!I_H^h$ to denote an interpolation operator from the coarse grid (H) to the find grid (h): If v^H is a coarse-grid function, then $I\!I_H^h v^H$ is a fine grid function obtained from it by multi-polynomial interpolation of some specified order. For example, if the chosen order is 2 then $I\!I_H^h$ is the multi-linear interpolation; i.e., linear interpolation if $d=1$, bilinear if $d=2$, etc. This is the usual notation used in the multigrid literature, the use of $I\!I_H^h$ instead of I_H^h serving to hint that the interpolation will often be of higher (than second) order. The index on which the operator works is denoted, when needed, by a dot. For example, $I\!I_H^h K_{i,\cdot}^{hH}$ denotes, for each index i, a fine-grid function obtained by interpolating from the coarse-grid function $K_{i,\cdot}^{hH}$, the latter being the function whose value at x_J^H is $K_{i,J}^{hH}$. The value of the interpolated function at the fine grid point x_j^h is denoted by $[I\!I_H^h K_{i,\cdot}^{hH}]_j$.

We will denote by $(I\!I_H^h)^T$ the *adjoint* of $I\!I_H^h$. This means that if $I\!I_H^h$ is written as an $n \times n^c$ matrix (where n and n_c are the number of points on the fine and on the coarse grids respectively), then $(I\!I_H^h)^T$ is the $n^c \times n$ *transpose* of $I\!I_H^h$. Note that

$(\mathbb{I}_H^h)^T$ describes a fine-to-coarse transfer ("reduction") operator, and indeed it will be used for that purpose. In case $\mathbb{I}_H^h$ denotes linear interpolation, for example, $2^{-d}(\mathbb{I}_H^h)^T$ is the familiar "full weighting" operator, extensively used in multigrid algorithms.

Later on we will use more than two grids. It will therefore be convenient to refer to them as levels and number them, starting with the coarsest grid that will be called level 1, the next finer grid being level 2, etc.

3. Smooth Kernels

3.1 General description

Whenever the kernel $K(x,y)$ is sufficiently smooth with respect to the variable y, we can approximate $K_{i,j}^{hh}$ by

$$\tilde{K}_{i,j}^{hh} = [\mathbb{I}_H^h K_{i,.}^{hH}]_j, \tag{3}$$

where the interpolation $\mathbb{I}_H^h$ has sufficiently high order and $K_{i,.}^{hH}$ is "injected" from $K_{i,.}^{hh}$; i.e., $K_{i,J}^{hH} \underset{\text{def}}{=} K_{i,2J}^{hh}$ (for a more general situation see Sec. 6.2). Hence equation (2) can be approximated by

$$\begin{aligned} w_i^h \simeq \tilde{w}_i^h &\underset{\text{def}}{=} h^d \sum_j \tilde{K}_{i,j}^{hh} u_j^h = h^d \sum_j [\mathbb{I}_H^h K_{i,.}^{hH}]_j u_j^h \\ &= h^d \sum_J K_{i,J}^{hH} [(\mathbb{I}_H^h)^T u_{.}^h]_J = H^d \sum_J K_{i,J}^{hH} u_J^H , \end{aligned} \tag{4}$$

where

$$u^H \underset{\text{def}}{=} 2^{-d}(\mathbb{I}_H^h)^T u^h. \tag{5}$$

Note that u^H is comparable to u^h; in case u^h is smooth $u_J^H \simeq u_{2J}^h$.

Whenever $K(x,y)$ is also sufficiently smooth as a function of x (very often K has the same smoothness properties in both x and y), the value of w_i^h can be calculated only for coarse grid points $i = 2I$, using interpolation $\hat{\mathbb{I}}_H^h$ to obtain the other values on the fine grid (very often $\hat{\mathbb{I}}_H^h = \mathbb{I}_H^h$ can be used). Namely

$$w^h \simeq \hat{\mathbb{I}}_H^h w^H \tag{6}$$

where

$$w_I^H \underset{\text{def}}{=} \tilde{w}_{2I}^h = H^d \sum_J K_{I,J}^{HH} u_J^H \tag{7}$$

and where $K^{HH}_{\cdot,J}$ is "injected" from $K^{hH}_{\cdot,J}$, i.e., $K^{HH}_{I,J} \underset{\text{def}}{=} K^{hH}_{2I,J} = K^{hh}_{2I,2J}$.

The multi-summation (2) has thus been reduced to the analogous coarse grid multi-summation (7). The latter problem can be coarsened in a similar way, using a coarser grid with meshsize $\bar{H} = 4h$. This process of coarsening is repeated until the number of gridpoints is proportional to $n^{1/2}$. On that grid the multi-summation is actually performed (requiring $O(n)$ operations) since further coarsening would not reduce the overall complexity (e.g. the work involved in the transfer of u^h to the coarse grid (5) is already of $O(n)$). Note that, for a fixed number of gridpoints n, the number of coarser levels required to reach a grid with $n^{1/2}$ points is inversely proportional to d, the dimension of Ω.

3.2 <u>One dimensional test</u>

In the following one dimensional example the discretization error is $O(h^2)$ $(2s = 2)$, while the interpolation error of K, and therefore the coarse grid integration error, is $O(h^4)$ (since we use fourth order transfers). Therefore it should be possible to carry out the integration on a coarse grid with meshsize $\tilde{H} = h^{1/2}$, while the total error will be only slightly larger than the fine grid discretization error. The overall computing time is $O(n)$. The treatment of the integrals near the boundary of the domain will be explained in detail in Section 4.1.

The integral is given by

$$w(x_i) = \int_0^\pi K(y - x_i)u(y)dy \qquad x_i \in [0, \pi] \tag{8}$$

where $K(y - x)$ is defined by:

$$K(y - x) = \cos(y - x), \tag{9}$$

and u is given by:

$$u(y) = \sin^2(y). \tag{10}$$

The integration over K is carried out in such a way that the integral is exact for a linear function $(s = 1)$. The functions u and w are transferred using fourth order operators (see end of Section 4.1). K on the coarse grid is given by $K^{HH}_{I,J} = K^{hh}_{2I,2J}$. The coarsest grid (level 1) had $(8 + 1)$ points, the second coarsest $(16 + 1)$ etc.

To be able to monitor the error in the multilevel multi-integration we will measure the error E^l_k, defined as the average absolute error of the integrals on level l, when the integration itself is carried out on level k $(k \leq l)$.

$$E^l_k = \frac{1}{n+1}\sum_{i=0}^{n} |w^{k,l}_i - w(x_i)|, \tag{11}$$

Table 1. Average error E_k^ℓ (for a grid with $2^{\ell+2}+1$ points, employing a coarsest auxiliary grid with $2^{k+2}+1$ points) in calculating the one-dimensional smooth-kernel multi-integration (9).

l	$k=l$	$k=l-1$	$k=l-2$	$k=l-3$	$k=l-4$	$k=l-5$	$k=l-6$
2	8.48e-3	9.94e-3	-	-	-	-	-
3	2.14e-3	2.23e-3*	3.70e-3	–	–	–	–
4	5.35e-4	5.41e-4	6.34e-4		–	–	–
5	1.34e-4	1.34e-4	1.40e-4*	2.33e-4		–	–
6	3.35e-5	3.35e-5	3.39e-5	3.97e-5			–
7	8.37e-6	8.37e-6	8.39e-6	8.75e-6*	1.46e-5		
8	2.09e-6	2.09e-6	2.09e-6	2.12e-6	2.48e-6	8.28e-6	
9	5.23e-7	5.23e-7	5.23e-7	5.24e-7	5.47e-7*	9.10e-7	
10	$\sim$ 1.3e-7	1.31e-7	1.31e-7	1.31e-7	1.32e-7	1.55e-7	5.18e-7
11	$\sim$ 3.3e-8		3.25e-8	3.25e-8	3.26e-8	3.40e-8*	5.65e-8
12	$\sim$ 8.e-9			8.35e-9	8.06e-9	8.54e-9	1.04e-8

Table 2. CPU time in sec. for Table 1

l	$k=l$	$k=l-1$	$k=l-2$	$k=l-3$	$k=l-4$	$k=l-5$	$k=l-6$
2	0.007	0.005	–	–	–	–	–
3	0.016	0.008*	0.006	–	–	–	–
4	0.057	0.022	0.013		–	–	–
5	0.197	0.066	0.029*	0.021		–	–
6	0.753	0.211	0.079	0.045			–
7	2.94	0.785	0.241	0.108*	0.076		
8	11.4	2.95	0.848	0.312	0.170	0.142	
9	45.7	11.6	3.17	0.975	0.435*	0.300	
10	$\sim$ 200.	46.0	11.8	3.38	1.22	0.691	0.568
11	$\sim$ 800.		46.5	12.6	3.87	1.73*	1.21
12	$\sim$ 3200.			47.4	13.6	4.91	2.70

where in this section $w_i^{k,l}$ is given by $w_i^{k,l} = [I\!I_k^l w_\cdot^k]_i$. Note that E_l^l is the L_1 norm of the discretization error on grid l. We wish to see for which k $E_k^l \simeq E_l^l$ holds.

The results in Table 1 were obtained using fourth order operators for both $I\!I_H^h$ and $(I\!I_H^h)^T$. The starred results clearly show that the grid can be coarsened to $H = O(h^{1/2})$, whereas the total error remains very close to the discretization error of the fine grid integral. The work involved in transferring u^h to the coarse grid and in interpolating w^H to the fine grid is obviously $O(n)$. Since the integration is carried out on a coarse grid with a number of points proportional to $n^{1/2}$, the total work should be $O(n)$, and this complexity was indeed obtained by the algorithm (see the starred results in Table 2).

4. Singular-Smooth Kernels

4.1 General description

So far, the kernel K was assumed to be much smoother than the function u over the entire domain of integration. In many problems of practical importance the kernel $K(x,y)$ is what we will call 'singular-smooth', that is, it has some singular points and the smoothness increases rapidly with increasing distance from these points. One example is potential-type kernels, such as $K(x,y) = \log|x-y|$ or $K(x,y) = |x-y|^{-1}$. For simplicity we will assume that, as in these examples, the only singular points are the points $x = y$. (For further remarks about the kernels for which the present methods are applicable — see Sec. 6.3.)

We start with the case that $i = 2I$ and derive an exact expression relating $w_I^H = \tilde{w}_i^h$ and w_i^h, replacing the approximate equation (4).

$$\begin{aligned} w_i^h &= h^d \sum_j K_{i,j}^{hh} u_j^h = h^d \sum_j \tilde{K}_{i,j}^{hh} u_j^h + h^d \sum_j (K_{i,j}^{hh} - \tilde{K}_{i,j}^{hh}) u_j^h \\ &= w_I^H + h^d \sum_j (K_{i,j}^{hh} - \tilde{K}_{i,j}^{hh}) u_j^h \end{aligned} \tag{12}$$

Since $\tilde{K}_{i,j}^{hh}$ is an interpolation (3) of $K_{i,j}^{hh}$ itself using only coarse grid points, the operator $(K_{i,j}^{hh} - \tilde{K}_{i,j}^{hh})$ is given by

$$(K_{i,j}^{hh} - \tilde{K}_{i,j}^{hh}) = \begin{cases} 0 & \text{j=2J} \\ O(h^{2p} K^{(2p)}(\xi)) & \text{otherwise} \end{cases} \tag{13}$$

where $2p$ is the interpolation order and $K^{(2p)}(\xi)$ is a $2p$'th derivative of K at some intermediate point ξ.

Thus, whenever the $2p$'th difference of K is suitably small $(h^{2p}|K^{(2p)}| \ll h^{2s}|u|_{2s})$, the coarse grid approximation w_I^H to w_i^h in (12) will be an accurate one.

Clearly this is no longer uniformly true when the kernel K is singular at certain points. However, far from the singularity ($\|y-x\| \gg h$ or $\|j-i\| \gg 1$ for the discrete case), the $2p$'th differences of K will again be small. Using this knowledge of K we can split the correction term in (12) into two parts and write

$$w_i^h = w_I^H + h^d \sum_{\|j-i\| \leq m} (K_{i,j}^{hh} - \tilde{K}_{i,j}^{hh}) u_j^h + h^d \sum_{\|j-i\| > m} (K_{i,j}^{hh} - \tilde{K}_{i,j}^{hh}) u_j^h \tag{14}$$

where in one dimension $\|j-i\| = |j-i|$. The meaning of this norm in higher dimensions is more involved (it depends on the direction of interpolation) and will be discussed below.

The remaining problem is to find a value of m for which we can neglect the last term in (14). This means that for the case of singular-smooth kernels, the multi-integration is again performed on the coarse grid, but part of the integral, near the singularity, is corrected. The corrected value is injected to the fine grid, so that

$$w_i^h \simeq \bar{w}_I^H = w_I^H + h^d \sum_{\|j-i\| \leq m} (K_{i,j}^{hh} - \tilde{K}_{i,j}^{hh}) u_j^h. \tag{15}$$

If the point i is not a coarse grid point ($i = 2I+1$), we define another coarse grid approximation $\hat{K}$ to the fine grid kernel K (16), similar to $\tilde{K}$ defined by (3) but now interpolating with respect to the index i,

$$\hat{K}_{i,j}^{hh} = [\hat{I\!I}_H^h K_{\cdot,j}^{Hh}]_i, \tag{16}$$

where $K_{I,j}^{Hh} = K_{2I,j}^{hh}$. In terms of $\hat{K}$ we can write

$$\begin{aligned} w_i^h &= h^d \sum_j K_{i,j}^{hh} u_j^h = h^d \sum_j \hat{K}_{i,j}^{hh} u_j^h + h^d \sum_j (K_{i,j}^{hh} - \hat{K}_{i,j}^{hh}) u_j^h \\ &\simeq [\hat{I\!I}_H^h \bar{w}_\cdot^H]_i + h^d \sum_j (K_{i,j}^{hh} - \hat{K}_{i,j}^{hh}) u_j^h. \end{aligned} \tag{17}$$

Assuming that $K(x,y)$ has similar smoothness properties in x and y, and that therefore identical interpolation operators are used in (3) and (16), the equation for the correction term here will be similar to (13):

$$(K_{i,j}^{hh} - \hat{K}_{i,j}^{hh}) = O(h^{2p} K^{(2p)}(\xi)) \qquad (\forall i = 2I+1, \forall j) \tag{18}$$

Again the correction term in (17) is split into two parts, and the part for $\|j-i\| > m$ is neglected, defining the approximation to the fine grid integral, when i is not a coarse grid point, by:

$$w_i^h \simeq [\hat{I\!I}_H^h \bar{w}_\cdot^H]_i + h^d \sum_{\|j-i\| \leq m} (K_{i,j}^{hh} - \hat{K}_{i,j}^{hh}) u_j^h. \tag{19}$$

The equations (15, 19) define the coarse grid approximations to the fine grid integrals for all points i; first the coarse grid integrals are calculated, then these integrals are corrected and injected to the fine grid (15) and finally the fine grid integrals are interpolated to the points i that are not part of the coarse grid and corrected again (19).

An alternative way for computing (19) would be to interpolate $K^{HH}_{I,J}$ with respect to both variables I, J in (16): $\bar{K}^{hh}_{i,j} = [\hat{\mathbb{I}}^h_H \tilde{K}^{Hh}_{\cdot,j}]_i = [\hat{\mathbb{I}}^h_H \mathbb{I}^h_H K^{HH}_{\cdot,\cdot}]_{i,j}$. This would result in a somewhat (about one third) smaller error in the integrals w^h_i $(i = 2I+1)$, at the expense of introducing a second correction function, instead of (18). However, because of the additional storage needed and the more complicated calculation of $\bar{K}$, the actual computations were carried out as described by (19).

For convolution-type kernels, the correction terms $(K^{hh}_{i,j} - \tilde{K}^{hh}_{i,j})$ and $(K^{hh}_{i,j} - \hat{K}^{hh}_{i,j})$ can of course be pre-computed in $O(n)$ operations. Therefore this calculation does not change the overall $O(np)$ complexity. However, for general type kernels, the application of the corrections in this manner would consume $n(2p)(2m+1)$ operations on the finest grid. *This correction work can again be reduced to* $O(np)$ by grouping together cp points (cp values of i) at a time and by carrying out the corrections for all these cp points over a *fixed* (independent of i) interval (of length $2m + cp = O(p)$), instead of the varying interval ($\|j - i\| \le m$) used in (19). The calculation of the contributions of u^h_j, for each j in the fixed interval, to the coarse grid function u^H, and, through it, to w^H costs $O(p^2)$ operations, but it is performed once per cp points, hence costing only $O(p)$ operations per point. These contributions to w^H are then interpolated and subtracted from the integral at each of the cp points, and the exact contributions $\sum_j K^{hh}_{i,j} u^h_j$ are added instead (i being each of the cp points, $\sum_j$ being a summation over the fixed interval).

The restriction operators $\mathbb{I}^H_h = (\mathbb{I}^h_H)^T$ used in (5) for the one dimensional case are given below. These operators can also be used in problems of a higher dimension, if the grid is coarsened with respect to one dimension at a time (see Sec. 4.4). The operators of order 2, 4 and 6 are respectively:

$$\mathbb{I}^H_h = \frac{1}{2}[1, 2, 1] \tag{20a}$$

$$\mathbb{I}^H_h = \frac{1}{16}[-1, 0, 9, 16, 9, 0, -1] \tag{20b}$$

and

$$\mathbb{I}^H_h = \frac{1}{256}[3, 0, -25, 0, 150, 256, 150, 0, -25, 0, 3]. \tag{20c}$$

Near the boundary of the domain Ω non-central transfer operators for u and w should be applied. The number of additional operators that should be programmed is directly proportional to the order $2p$ of the transfer operator. *One can avoid this additional programming*, and use the central transfer operators also near the boundary, by smoothly extending $K(x,y)$ outside Ω (usually $K(x,y)$ is well-defined there in the first place), while defining on the finest grid $u(y)=0$ for $y \notin \Omega$. In practice this means adding external points near the boundary of each coarser grid, for which non-vanishing values of u^H, as well as needed-for-interpolation values of w^H, are calculated. The number of such external points is at most $2p-2$ in each direction on each line of interpolation.

4.2 Minimum work

Taking the one dimensional kernel $K = \ln|x-y|$ as an example, we will now derive the optimal values of the order $2p$ of the transfer operators (we will only consider even orders) and of the radius m of the fine grid correction region, so as to minimize the computational work, under the constraint that the added error, due to the use of coarser grids, should be smaller than the original fine grid discretization error. The interpolation error, resulting from the use of the interpolated kernel $\tilde{K}$ or $\hat{K}$ instead of the full kernel K is given by (13, 18). However, since part of this error is corrected (15, 19), the first uncorrected error term in w will occur at a distance $(m+1)h$ from the singularity. The error resulting from the use of $\tilde{K}_{i,j}$, instead of $K_{i,j}$, at that point $(j = i \pm (m+1))$ is:

$$\begin{aligned} &h^{2p}\frac{\partial^{2p}}{\partial y^{2p}}K(x_i,\xi)\times \\ &\times\frac{(2p-1)\times(2p-3)\times\ldots\times(1)\times(-1)\times\ldots\times(3-2p)\times(1-2p)}{(2p)!} \end{aligned} \qquad (21)$$

(see for instance [8, p. 279]), where a calculation for the logarithmical kernel shows that, provided $m \gtrsim 2.5p$ (so that no interpolation point is too close to the singularity at i), one can approximately take ξ at $x_j = x_i + (m+1)h$ and hence

$$\left|\frac{\partial^{2p}}{\partial y^{2p}}K(x_i,\xi)\right| \approx \frac{(2p-1)!}{(m+1)^{2p}h^{2p}}. \qquad (22)$$

The error in w resulting from *all* uncorrected points, at distances $(m+1)h, (m+2)h, \ldots$, both left and right of the singularity, can similarly be calculated, and, when added together, yields approximately

$$3.16\frac{\{1\times3\times5\times\cdots\times(2p-1)\}^2}{2p(m+1)^{2p}} \simeq \left(\frac{0.7p}{m+1}\right)^{2p}. \qquad (23)$$

Notice that this interpolation error is independent of the meshsize, hence the same error will approximately be introduced also at each interpolation in each of the subsequent coarsening steps.

The discretization error, the error of approximating (1) by (2), is $O(h^{2s}u^{(2s)}(\xi))$, where $2s$ is the approximation order. We will assume u to be smooth on the scale of the entire domain; so that $h^{2s}u^{(2s)} \approx n^{-2s}$, where n is the number of gridpoints on the domain. (More generally, this relation can be used to *define* the 'effective' n, so that n is roughly the number of meshsizes in the length scale on which u is smooth.) The condition that the coarse grid integration error (23) should be smaller than the fine grid discretization error is therefore

$$m = 0.7pn^{s/p} - 1. \tag{24}$$

As a second equation we calculate the amount of work per fine grid point as a function of $2p$ and m. Taking an 'operation' to mean a combination of one multiplication and one addition, the number of operations in transferring the function u^h to the coarse grid is p per fine grid point (since for half of the values of u^h the transfer is trivial). Similarly, p is also the number of operations per fine gridpoint in interpolating w^H to the fine grid. The number of operations per point in correcting the coarse grid integral on the fine grid is $2m+1$. This gives $(2m+1+2p)$ as the total fine grid work per fine grid point, and similar figures also hold in higher dimensions (as we will see below). For a d-dimensional problem (or in higher dimensions, when coarsening with respect to d dimensions at a time), since the coarse grid has approximately 2^{-d} the number of the fine grid points, the number of coarse grid operations is proportional to $2^{-d}(2m+1+2p)$; and so on for still coarser grids. The work of the actual integration on the coarsest grid can be neglected, assuming this grid has less than $n^{1/2}$ points. Adding all the work on all the levels we obtain the total work per fine grid point

$$W \simeq (1-2^{-d})^{-1}(2m+1+2p). \tag{25}$$

Substituting m from (24) into (25) and then minimizing W by setting $\partial W/\partial p = 0$ gives the equation $\alpha(\ln\alpha - 1) = 1.43$, where $\alpha = n^{s/p}$. It follows that $\alpha = 3.9$ and hence

$$2p = 1.4s\ln n, \tag{26a}$$

and by (24)

$$m = 1.4(2p) - 1 = 2s\ln n - 1. \tag{26b}$$

Hence, by (25), for $d=1$,

$$W = 2(2m+1+2p) \approx 11s\ln n. \tag{26c}$$

In Table 3 we for example take $s=1$ (piecewise linear integration rule) and compute the optimal values of $2p$ (by (26), rounded to an integer) and the corresponding m and W (derived from (24) and (25), respectively) as functions of the

Table 3. Optimal values for m and $2p$ and corresponding W for $d = 1$, $s = 1$

l	n	$2p$	m	W	$2p$	m	W
3	33	6	6	38	6	6	38
4	65	6	8	46	6	8	46
5	129	8	9	54	6	11	58
6	257	8	11	62	6	14	70
7	513	10	11	66	6	17	82
8	1025	10	13	74	6	22	102
9	2049	10	15	82	6	28	126
10	4097	12	16	90	6	36	158
11	8193	12	18	98	6	45	194
12	16385	12	20	106	6	57	242

level l. Furthermore, we show the optimal value of m (24), and the corresponding work W (25), when restricting the order of transfers to $2p = 6$. In this way we can see how the work with a restricted order of transfer is related to the minimal work.

Note that the work per point for the 'classical' integration is given by $W = n$. The main conclusion to be drawn from this table is that, even with unrestricted transfer orders, the optimal order of transfer is reasonable, while when restraining the order of transfer to 6, the amount of work increases only by a factor of 2 (for $n \simeq 10.000$). Thus, transfers of impractically high orders are not required to obtain computing times close to the theoretically best. For problems in higher dimensions the situation will be even more favourable, as we will see in Section 4.4.

4.3 One dimensional test

As an example of a multi-integral with a non-periodic, singular-smooth kernel, we tested the one dimensional case discussed above

$$w(x) = \int_{-1}^{1} \ln|x-y|(1-y^2)dy, \tag{27}$$

with piecewise linear (second order accurate: $s = 1$) discretization. Equations (15, 19) were used for the fast integration with $m = 3 + 2\ln n$ (found to give reasonable results at moderate values of n). In Tables 4 and 5 average errors (11) are given, using fourth and sixth order transfers respectively, where now $w_i^{k,l}$ are the values

Table 4. Average error E_k^ℓ (for a grid with $2^{\ell+2}+1$ points, employing a coarsest auxiliary grid with $2^{k+2}+1$ points) in calculating the one-dimensional logarithmic-kernel multi-integration (27), using fourth order transfers

l	$k=l$	$k=l-1$	$k=l-2$	$k=l-3$	$k=l-4$	$k=l-5$
2	8.332e-3	8.332e-3	–	–	–	–
3	2.094e-3	2.140e-3	2.141e-3	–	–	–
4	5.245e-4	5.477e-4	5.667e-4	5.662e-4	–	–
5	1.312e-4	1.403e-4	1.536e-4	1.620e-4	1.620e-4	–
6	3.282e-5	3.610e-5	4.202e-5	5.026e-5	5.451e-5	5.451e-5
7	8.207e-6	9.894e-6	1.317e-5	1.909e-5	2.780e-5	3.274e-5
8	2.052e-6	2.654e-6	3.846e-6	6.148e-6	1.029e-5	
9	5.130e-7		1.177e-6	2.051e-6		

Table 5. Same as Table 4, but using sixth order transfers

l	$k=l$	$k=l-1$	$k=l-2$	$k=l-3$	$k=l-4$	$k=l-5$	$k=l-6$
2	8.332e-3	8.332e-3	–	–	–	–	–
3	2.094e-3	2.078e-3	2.054e-3	–	–	–	–
4	5.245e-4	5.222e-4	5.170e-4	5.154e-4	–	–	–
5	1.312e-4	1.307e-4	1.297e-4	1.278e-4	1.278e-4	–	–
6	3.282e-5	3.269e-5	3.244e-5	3.194e-5	3.120e-5	3.120e-5	–
7	8.207e-6	8.141e-6	8.012e-6	7.756e-6	7.327e-6	7.315e-6	7.315e-6
8	2.052e-6	2.034e-6	1.998e-6	1.926e-6	1.786e-6	1.637e-6	1.735e-6
9	5.130e-7		4.968e-7	4.754e-7	4.331e-7	3.599e-7	3.015e-7
10	~1.2e-7			1.095e-7	8.809e-8	4.712e-8	4.793e-8
11	~3.0e-8				1.966e-8	7.668e-9	2.135e-8

obtained for w_i^h on the finest grid through the corrections (15) and (19), assuming similar corrections have also been used for obtaining w_I^H, and so on recursively to level k, for which the values of w are calculated by direct summation. The coarsest grid ($l=1$) has $8+1$ points including the boundaries, the second coarsest $16+1$, etc (see also Table 3). Additional points were used on coarse grids to cope with the necessity of a larger domain, as outlined in Section 4.1.

When one allows the additional error introduced by the coarse grid integration to be as large as the discretization error on the finest grid, the fourth order transfers

Table 6. CPU time in sec. for Table 5

l	$k = l$	$k = l - 1$	$k = l - 2$	$k = l - 3$	$k = l - 4$	$k = l - 5$	$k = l - 6$
2	0.006	0.014	–	–	–	–	–
3	0.015	0.023	0.036	–	–	–	–
4	0.057	0.057	0.059	0.064	–	–	–
5	0.199	0.124	0.124	0.128	0.133	–	–
6	0.766	0.349	0.260	0.257	0.260	0.269	–
7	2.95	1.02	0.612	0.514	0.526	0.537	0.534
8	11.8	3.53	1.60	1.18	1.11	1.11	1.12
9	∼45.		4.74	2.81	2.45	2.36	2.37
10	∼180.			7.14	5.23	4.80	4.75
11	∼720.				12.4	10.6	10.1

(20b) give good results for $2l - k \leq 10$. The sixth order scheme (20c) gives satisfactory results for $2l - k \leq 17$, while the amount of additional computing time needed by the sixth order transfers is small compared to the overall computing time (generally 20%).

To see whether the fast multi-integration efficiency depends on the smooth character of $u(y)$ in (27), the same calculations were carried out with a more oscillatory function u. It turns out that, since the discretization error of the fine grid integration is larger, the coarse grid integration becomes relatively more accurate; in other words, the *effective* n (see Sec. 4.2) is smaller. Thus, *the fast integration of a more oscillatory function is an easier task.*

In Table 6 the computing time for Table 5, in seconds on an IBM 3081, is given. Since the important information is the relative reduction in computing time, the results should be approximately machine independent. The multilevel multi-integration gives a significant reduction in computing time over the 'classical' one-level integration ($k = l$), from level 6 (257 points) onwards, so the approach is only worthwhile for multi-integration over many points. Using second order transfers, the multilevel multi-integration gives a significant gain in computing time from level 4 (65 points) onwards. When comparing the gain in computing time with the predictions from Table 3, it can be seen that the gain lies somewhere between the two predicted gains (with unrestricted p and with $2p$ restricted to 6), since the number of points m to be corrected on the fine grid, which is taken as $m = 3 + 2\ln n$, does not grow as fast as in the seventh column of Table 3. This results of course in somewhat less accurate integrals, but in the tested cases the error was still smaller than the truncation error.

All reported results were obtained from programs written in PASCAL. Since the compiler used was not very efficient, a FORTRAN code was written, for which a very efficient compiler was available, to check if the obtained reductions in computing time were compiler dependent. Whereas the FORTRAN code was much faster (CPU times were nearly one tenth of the PASCAL computing time), the relative reductions obtained by the multilevel integration were very similar (differences were less than 10%). Also the PASCAL code was run on a different machine (VAX11/750) giving similar results. Thus it can be safely concluded that the reported reductions in computing time can be generally obtained.

4.4 Two dimensional test

The above algorithm can easily be extended to two dimensions by coarsening alternately in the x dimension and the y dimension. In this way, the transfer of the kernel K and the function u and the interpolation and correction of w are essentially the same as for the one dimensional case (Section 4.1). This approach, which may seem cumbersome at first glance, proved to be very effective and simple to generalize to even higher dimensions. Its slight disadvantage is the additional (50%) storage required for the 'half-coarsened' grids. Its main strength is its simplicity and the way it decouples both variables. Furthermore, it ensures that the total work will continue to be proportional to $O(n \log n)$, since all the components of the algorithm are at most of this complexity.

As a test problem the following multi-integration was chosen:

$$w(x,y) = \int_\Omega K(x,y,x',y')u(x',y')dx'dy', \qquad (x,y) \in \Omega \tag{28}$$

where

$$K(x,y,x',y') = [(x-x')^2 + (y-y')^2]^{-1/2}$$

$$u(x',y') = \begin{cases} (1 - x'^2 - y'^2)^{1/2} & (x',y') \in \Omega \\ 0 & \text{elsewhere} \end{cases}$$

and Ω is the disc $x^2 + y^2 \leq 1$. The fine grid integration is second order accurate ($s = 1$).

As stated in Section 4.1, the number of points on each grid is extended according to the order of transfer, resulting in a straightforward integration near the boundary. In two dimensions the cost of this additional storage is considerable, except for very fine grids — which is our aim anyway, since the multilevel approach pays only when using a large number of points.

In the next six tables (Tables 7–12) results are reported for tests in which the corrections (15, 19) to the coarse grid integrals were carried out over rectangles of $(2m_1+1)\times(2m_2+1)$ points around the singularity: in the direction of interpolation the number of correction points was $m_1 = 3 + .5 \ln n$, while in the perpendicular direction the number of points was $m_2 = 2$. The error norm (11) is adjusted with respect to these corrections (15, 19) as explained in Sec. 4.3. The coarsest grid (level 1) consisted of $(4+1)^2$ points, level 2 of $(8+1)^2$ points and level 3 of $(16+1)^2$ points, etc.

As can be seen from these tables, for any given n, the order of transfer necessary to obtain a nearly optimal reduction in computing time is much lower than for the one dimensional case (compare Table 4 and 8); otherwise the same conclusions are valid. Mainly because of the additional points needed near the boundary

Table 7. Average error E_k^ℓ (for a grid with $(2^{\ell+1}+1)\times(2^{\ell+1}+1)$ points, employing a coarsest grid of $(2^{k+1}+1)\times(2^{k+1}+1)$ points) in calculating the two-dimensional singular-smooth multi-integration (28), using second order transfers

l	$k=l$	$k=l-1$	$k=l-2$	$k=l-3$	$k=l-4$	$k=l-5$
2	2.312e-1	2.600e-1	–	–	–	–
3	7.685e-2	9.566e-2	1.275e-1	–	–	–
4	1.518e-2	1.513e-2	3.374e-2	6.535e-2	–	–
5	$\sim 4e-3$	3.501e-3	1.023e-2	3.568e-2		–

Table 8. Same as Table 7, but using fourth order transfers

l	$k=l$	$k=l-1$	$k=l-2$	$k=l-3$	$k=l-4$	$k=l-5$
2	2.312e-1	2.327e-1	–	–	–	–
3	7.685e-2	7.805e-2	7.942e-2	–	–	–
4	1.518e-2	1.584e-2	1.663e-2	1.854e-2	–	–
5	$\sim 4e-3$	5.160e-3	5.833e-3	6.659e-3	9.247e-3	–
6	$\sim 1e-3$		1.800e-3	2.376e-3	3.467e-3	
7	$\sim 3e-4$			8.611e-4	1.269e-3	

Table 9. Same as Table 7, but using sixth order transfers

l	$k=l$	$k=l-1$	$k=l-2$	$k=l-3$	$k=l-4$	$k=l-5$
2	2.312e-1	2.315e-1	–	–	–	–
3	7.685e-2	7.636e-2	7.594e-2	–	–	–
4	1.518e-2	1.502e-2	1.491e-2	1.524e-2	–	–
5	$\sim 4e-3$	4.616e-3	4.463e-3	4.360e-3	4.686e-3	–
6	$\sim 1e-3$		1.131e-3	1.016e-3	9.525e-4	1.627e-3
7	$\sim 3e-4$			3.953e-4	3.377e-4	3.769e-4

on every grid, the computing time on the coarse grids increases significantly when high order transfers are used (compare for instance $l=2$, $k=1$ in Tables 10, 11 and 12). However, these high order schemes should be used for large problems only, where the additional computing time due to the extra points becomes small (approximately 15% for level 7 calculations with sixth order transfers). On level 7, when sixth order transfers are used, the computing time is reduced by a fac-

Table 10. CPU time in sec. for Table 7

l	$k=l$	$k=l-1$	$k=l-2$	$k=l-3$	$k=l-4$	$k=l-5$
2	0.113	0.085	–	–	–	–
3	1.35	0.466	0.485	–	–	–
4	19.4	3.04	2.15	2.13	–	–
5	$\sim 310.$	27.6	11.4	10.4		–

Table 11. CPU time in sec. for Table 8

l	$k=l$	$k=l-1$	$k=l-2$	$k=l-3$	$k=l-4$	$k=l-5$
2	0.113	0.155	–	–	–	–
3	1.35	0.703	0.651	–	–	–
4	19.4	4.15	2.79	2.78	–	–
5	$\sim 310.$	32.5	12.7	11.2	11.2	–
6	$\sim 5,000.$		70.0	51.2	50.0	
7	$\sim 80,000.$			235.	218.	

Table 12. CPU time in sec. for Table 9

l	$k=l$	$k=l-1$	$k=l-2$	$k=l-3$	$k=l-4$	$k=l-5$
2	0.113	1.01	–	–	–	–
3	1.35	2.56	2.21	–	–	–
4	19.4	9.86	6.34	5.99	–	–
5	$\sim 310.$	56.7	21.2	17.5	17.3	–
6	$\sim 5,000.$		102.	67.2	63.7	63.4
7	$\sim 80,000.$			278.	252.	248.

tor of 300, while the additional error caused by the coarse grid integration is still negligible.

5. Solution of Integral Equations

5.1 Straightforward multigridding: a preliminary test

The fast integration, outlined in the previous sections, can of course be used straightforwardly in the solution of integral equations, e.g., by employing it in the relaxation process and residual calculations of the usual FMG (full multigrid) algorithm (described in many articles; see e.g., [4, §1.6]). The relaxation should usually be of the *simultaneous* displacement type (such as Jacobi or Richardson relaxation), so that all the integrations needed in one relaxation sweep can be performed by one multi-integration.

However, in case *successive* displacement (e.g. Gauss-Seidel) schemes have much better smoothing properties, they can be approximately implemented in a *defect-correction* manner. That is, instead of relaxing the system of equations $Lu = f$ (where L is a discretized integral operator, e.g., $Lu_i = h^d \sum_j K_{i,j} u_j$), one can relax $L'u = f'$, where L' is a local approximation to L (e.g., $L'u_i = h^d \sum_{j \in N(i)} K_{i,j} u_j$, where $N(i)$ is some neighborhood of i) and $f' = f + L'u - Lu$. This f' can be calculated simultaneously at all points once per sweep, hence requiring only one multi-integration per sweep for evaluating Lu. In fact, f' may be evaluated only once per *several* sweeps; this is a good idea even for Jacobi-type relaxation, since the major cost is that of the multi-integration (see Sec. 5.2). An alternative to defect corrections is to update the integrals in some neighborhood of each relaxed point. (This alternative *must* be taken when the distributed relaxation described below is used, since some of the changes entering $L'u$ are of lower distribution order than those entering Lu.)

It is important to ensure that relaxation is effectively *local*; i.e., that relaxing at a point x_i introduces only small changes to the discrete integral $h^d \sum_k K_{jk} u_k \approx \int K(x_j, y) u(y) dy$ at points x_j far from x_i; otherwise each such integral would accumulate too many significant changes in a relaxation sweep. This can be achieved using a suitable kind of *distributed relaxation*. For example, instead of updating one unknown at a time ($u_i \leftarrow u_i + \delta_i$, say), three values are simultaneously changed: $u_{i-1} \leftarrow u_{i-1} - \delta_i$, $u_i \leftarrow u_i + 2\delta_i$ and $u_{i+1} \leftarrow u_{i+1} - \delta_i$, where δ_i is chosen such that after these changes the equation at x_i is satisfied. This is called a *second order* distributed relaxation. More generally, a distributed relaxation of order r is a relaxation where each set of simultaneous changes is an r-order difference of a local function (e.g., a multiple of a discrete delta function). Such a relaxation usually ensures that the changes in the integrals are essentially local: the influence of an r-order distributed relaxation at x_i on the integral at x_j behaves like $\partial^r K(x_j, x_i)/\partial x^r$, which decays like $|x_i - x_j|^{-r}$ or $|x_i - x_j|^{-r-1}$ in potential-type kernels. Such distributed relaxation schemes (special cases of the general schemes discussed in [5,§3.5]) can be used either as simultaneous displacement schemes (new values replacing old ones at the end of a sweep), in which case they are called *distributive Jacobi*; or in successive displacements (the changed values being immediately used in relaxing subsequent equations), in which case they are called *distributive Gauss-Seidel*. Near boundaries lower-order distributions can be used.

As a test problem we have solved the equation

$$\lambda U(x) - \int_{-1}^{1} K(x,y)U(y)dy = f(x), \tag{29}$$

where $K = \ln|x-y|$ and f is chosen so that the solution is $U(y) = 1 - y^2$. The discretization is second order accurate ($2s = 2$). For $\lambda = 0$, for example, the smoothing factor of the second order distributive Jacobi and distributive Gauss-Seidel schemes are $\bar{\mu} = 0.415$ and $\bar{\mu} = 0.238$, respectively. Using an underrelaxation factor of 0.6, distributive Jacobi is effective in the entire range of λ: $\bar{\mu}$ lies between $\bar{\mu} = 0.302$ and $\bar{\mu} = 0.400$ (the values for $\lambda = 0$ and $\lambda = \infty$, respectively).

As a preliminary experiment we have treated the case of $\lambda = 3$ and used the second order distributive Jacobi relaxation (by no means the best scheme for this case) in $V(1,1)$ cycles. At the two boundary points a first-order distribution has been employed. Tables 13 and 14 show results obtained with second, fourth and sixth order transfers ($2p = 2,4,6$ resp. (20a-c)) and $m = 3 + 2\ln n$. In Table 13 the L_1 norm of $u^k - U$ is given, where u^k is obtained using k coarser grids in the multi-integration and two $V(1,1)$ cycles per each level of the FMG algorithm. In the first column (labeled $3V(1,1)$), the discretization error is approximated using one additional $V(1,1)$ cycle, to eliminate the algebraic error. From this table it can be concluded that two $V(1,1)$ cycles solve the problem to the level of truncation errors. Whenever the error in the multi-integration was of the same order as the discretization error, the transfer order $2p$ was raised by 2.

The computing time is given in Table 14. An $O(n \log n)$ complexity is quite clearly shown. The actual running time could be substantially reduced by using a better relaxation scheme (hence less relaxation sweeps per cycle and/or less cycles), or, even further (by a factor close to 12), by the techniques we discuss next (Sec. 5.2).

5.2 Advanced multigridding

The work of the integral solver can be drastically reduced by having it resort to full-order multi-integration as seldom as possible. In fact, only one such multi-integration on the finest grid, plus some much less expensive ones (on coarser grids and/or using lower accuracy) is all that is needed.

The proposed procedure for obtaining a solution to $O(h^{2s})$ accuracy on the finest grid is an FMG algorithm as follows. First the equations are solved to a similar accuracy on a coarser grid, say with meshsize γh. (This is done by a similar procedure — so the algorithm is defined recursively.) The grid-γh solution is then interpolated to grid h to serve there as the first approximation. The order of this interpolation should be $2s$ or a little higher (if we want to have $O(h^{2s})$ accuracy also in *derivatives* of the solution; see [5, §7.1]).

Table 13. L_1 norm of $u^k - U$, where U is the solution of the integral equation (29), and u^k is the approximate solution obtained using k coarser grids in each of the multi-integrations and 2 $V(1,1)$ cycles at each level of the FMG algorithm

l	$2p$	$3V(1,1)$	$k=0$	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$
2	2	1.76e-3	2.11e-3	1.98e-3	–	–	–	–	–
3	2	4.37e-4	5.61e-4	3.67e-4	4.48e-4	–	–	–	–
4	2	1.08e-4	1.37e-4	2.21e-4	4.77e-4	5.08e-4	–	–	–
4	4	1.08e-4	1.37e-4	1.44e-4	1.43e-4	1.41e-4	–	–	–
5	4	2.65e-5	3.45e-5	3.62e-5	3.88e-5	4.10e-5	4.08e-5	–	–
6	4	6.45e-6	8.99e-6	9.60e-6	1.07e-5	1.23e-5	1.35e-5	1.35e-5	–
7	4	1.56e-6	2.38e-6	3.12e-6	3.69e-6	4.69e-6	6.22e-6	7.67e-6	7.67e-6
7	6	1.56e-6	2.38e-6	2.37e-6	2.35e-6	2.32e-6	2.28e-6	2.26e-6	2.26e-6
8	6	4.67e-7	5.95e-7	5.92e-7	5.87e-7	5.76e-7	5.59e-7	5.38e-7	5.36e-7
9	6	~1.2e-7	~1.5e-7	1.47e-7	1.46e-7	1.43e-7	1.36e-7	1.27e-7	1.18e-7
10	6	~3.0e-8	~4.0e-8			3.38e-8	3.07e-8	2.54e-8	2.20e-8

Table 14. CPU time in sec. for Table 13

l	$2p$	$k=0$	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$
2	2	0.64	0.63	–	–	–	–	–
3	2	1.06	1.04	1.08	–	–	–	–
4	2	1.85	1.77	1.73	1.77	–	–	–
4	4	2.44	2.63	2.45	2.34	–	–	–
5	4	4.62	4.06	4.10	4.05	4.11	–	–
6	4	11.8	7.85	7.30	7.25	7.38	7.35	–
7	4	38.0	18.0	13.9	13.4	13.4	13.5	13.5
7	6	39.9	20.3	16.6	16.0	16.0	15.9	16.1
8	6	140.	54.1	34.7	31.1	30.8	30.8	31.0
9	6	~500.	169.	83.8	65.9	62.8	63.1	63.5
10	6	~2000.			140.	123.	122.	120.

The first approximation thus obtained, denoted u^0, is now used for an $O(h^{2s})$ calculation of the integrals by the multi-integration. In our example (29), this multi-integration should cost about $Wn = 11sn \ln n$ operations (see (26c)). From this point on no such full multi-integration is needed. Instead, whenever needed (e.g., after each relaxation sweep or coarse-grid correction), a multi-integration is done on the solution *increment*, i.e., the difference between the current solution and u^0, and the integrals thus calculated are added to those of u^0. The incremental multi-integration can be of lower order: since the increment is only about γ^s times the truncation errors, the multi-integration can employ p and m which are $O(s \log \gamma)$, instead of $O(s \log n)$, hence requiring $O(sn \log \gamma)$ operations only. Since the required number of multigrid cycles (each employing a couple of relaxation sweeps followed by a coarse-grid correction) is $O(s \log \gamma)$, the total number of operations in all the fine-grid incremental multi-integrations is $O\big(s^2 n (\log \gamma)^2\big)$.

Each calculation of a coarse-grid correction should itself employ multi-integrations on coarser grids, but the needed accuracy of those is even lower: 10% accuracy should be enough, since it is a crude correction function which is being calculated. Hence p and m which are $O(1)$ can be used in these multi-integrations, and their cost is $O(n)$ operations per cycle, hence $O(sn \log \gamma)$ in all the cycles.

In summary, the entire FMG solution process requires only one full multi-integration on the finest grid, plus a similar one on each of the coarser grids (for the recursion needed to obtain the first approximation); the rest of the work is $O\big(s^2 n (\log \gamma)^2\big)$. Hence *the total number of operations to solve a d-dimensional integral equation to accuracy $O(h^{2s})$ is*

$$\left(nW + \frac{n}{\gamma^d} W + \frac{n}{\gamma^{2d}} W + \cdots\right) + O\big(s^2 n (\log \gamma)^2\big) \le (1-\gamma^{-d})^{-1} W n + O\big(s^2 n (\log \gamma)^2\big), \tag{30}$$

where Wn is the work of one fine-grid $O(h^{2s})$ multi-integration. Thus, in principle, for large enough n the total work is as close as one wishes to Wn, the work of just *one* multi-integration (taking, e.g., $\gamma = \log n$). In case of (29), for example, $W \approx 11 s \log n$ (see (26.c)), hence the total number of operations should be about $12 sn \log n + O(s^2 n)$ (taking, e.g., $\gamma = 12$; for moderate n, $\gamma = 2$ is of course preferable, and the operation count is then $22 sn \log n + O(s^2 n)$).

Preliminary tests of the approaches described in this section have been done with the two dimensional problem of Sec. 4.4, confirming the expected gains.

6. Conclusion

6.1 Summary of results

The computing time for multi-integrals with sufficiently smooth kernels has been reduced from $O(n^2)$ to $O(n)$ using multilevel multi-integration. More important, the calculation time for the basic potential-type kernels was reduced from

$O(n^2)$ to $O(n \log n)$. The gain, however, is significant only when working with sufficiently many points (several dozens). This fast multi-integration can be applied straightforwardly to the fast multigrid solution of integral equations.

For a one-dimensional multi-integration with a simple potential-type kernel, the theoretical number of required operations per gridpoint is roughly $W = 11s \ln n$, where $2s$ is the accuracy order of the discretization. If the order of the inter-grid transfers is restricted to 6, for $n \simeq 10,000$ this computing time is just doubled. The solution time for corresponding integral equations should be about $W + O(s^2 n)$.

Using grids coarsened with respect to one variable at a time, such results can readily be extended to higher dimensions. This convenient approach uses approximately four times as much storage as the one-level integration, partly because of the storage of the correction terms (13, 18), and partly because of the storage of the 'half-coarsened' grids. The required order of transfers, relative to the total number of points n, is much lower in problems of higher dimension.

Because the method employs simple equidistant grids, with a coarse to fine mesh ratio of 2, the algorithm is easily programmed, especially by anyone familiar with multigrid programming.

6.2 Non uniform grids and many-body interactions

In several problems of interest, it is inconvenient or impossible to choose the finest grid to be equidistant; e.g. in problems with interacting particles, the fine grid points coincide with the position of the particles. In this case multilevel multi-integration can still be used to speed up the calculation, but the next coarser grid will be constructed as a semi-uniform grid (a uniform grid subdivided wherever the particle density is higher), and as a consequence the coarse grid will no longer be a subset of the fine grid. The transfer of the function K from the finest grid to the next coarser grid should then be performed by a high order interpolation, instead of by simple injection. In fact however, in particle-type problems the kernel K is usually given by a closed form formula (e.g. $K(x, y) = |x - y|^{-1}$) through which it can directly be calculated on every grid, and hence need not be transferred from the finest grid. An effective code of this kind for general many-body interactions has been developed in collaboration with Y. Accad, and will be separately reported.

For integral equations, non uniform grids may result from the need to employ local refinements in those parts of the domain where the solution is less smooth. Multigrid techniques similar to those employed for differential equations (see [4, §9] or [5, §9]) can then be used.

6.3 Other extensions and limitations

The method described above in terms of multi-integrations is actually applicable to a wide class of tasks of multiplying a vector by an $n \times m$ matrix $K = [K_{ij}]$, when the result is desired to some accuracy $\varepsilon > 0$. Thus, most of the work of multiplying by K can be replaced by multiplying with the smaller $n \times m'$ matrix K', if the following holds. For each $1 \le j \le m$ there exists a subset N_j (the "coarse neighborhood" of the unknown j) and coefficients $\alpha_{jj'}$ (the "coarse-to-fine interpolation" coefficients) such that, for any i,

$$\sum_{j \notin M_i} \left| K_{ij} - \sum_{j' \in N_j} \alpha_{jj'} K'_{ij'} \right| \le \varepsilon.$$

Here M_i is a small subset of columns (the "fine neighborhood" of equation i), for which the multiplication of the vector by the i-th row should still be done in terms of the original matrix K. This replacement of K by K' pays of course whenever the size of each N_j and M_i is small compared with m, and provided m' is substantially smaller than m.

Not in all such cases the full efficiency, exhibited by our examples above, can be attained. To obtain that type of efficiency we have basically assumed that $K_{ij} = K(x_i, x_j)$, where $x_i \in I\!R^d$ and where, for any meshsize h we use (including that of the coarser grids) and any partial derivative ∂, the size of $h^p \partial^p K(x,y)$ decreases exponentially in p, except perhaps in some singular neighborhoods whose relative total volume is $O(hp)$.

In some important cases this assumption will be satisfied for the finest grid used in practice, but not for considerably coarser grids needed in the algorithms of Secs. 4 and 5. This typically happens for oscillatory kernels, such as Bessel or Hankel functions of $|x-y|$, occuring for example in boundary element discretization of highly indefinite problems, where the finest grid is just fine enough to resolve the oscillations.

Often in this situation the full efficiency may still be obtained by writing the unknown function u in the form $u = \sum_{\ell=1}^{L} u_\ell \varphi_\ell$, where φ_ℓ are some fixed oscillatory functions, and the u_ℓ are unknown functions which are transformed in a smoother way by the integral operator. The multi-integrations and the multigrid solvers should then be described in terms of those functions u_ℓ. Often, the number L of such functions should increase on coarser grids, proportionally to their meshsize. (For a similar device in the multigrid treatment of highly indefinite PDEs, see briefly in Sec. 4.2.2 of either [4] or [5], and in much more detail in [14].)

References

[1] J. Ambrosiano, L. Greengard, and V. Rokhlin, Yale University Research Report YALEU/DCS/RR-565, September 1987 (unpublished).

[2] A.W. Appel, *SIAM J. Sci. Stat. Comput.* **6**, 85 (1985).

[3] J. Barnes, and P. Hut, *Nature* **324**, 446 (1986).

[4] A. Brandt, in *Multigrid Methods* (Proc. Köln-Porz, 1981), edited by W. Hackbusch and U. Trottenberg, (*Lecture Notes in Math.* **960**, Springer Verlag, 1982), p. 220. Incorporated later into [5].

[5] A. Brandt, *Multigrid Techniques*: *1984 Guide, with Applications to Fluid Dynamics*. Monograph. (Available as GMD Studien Nr. 85, from GMD-AIW, postfach 1240, D-5205, St. Augustin 1, W. Germany).

[6] A. Brandt, in *Proceedings of the Third Copper Mountain Conference on Multigrid Methods, Copper Mountain, Colorado, April, 1987*, edited by S. McCormick (Marcel-Dekker, 1988), p. 35.

[7] J. Carrier, L. Greengard, and V. Rokhlin, *SIAM J. Sci. Stat. Comput.* **9**, 669 (1988).

[8] G. Dahlquist, and A. Björck, *Numerical Methods* (Prentice-Hall, 1974).

[9] L. Greengard, and V. Rokhlin, *J. Comput. Phys.* **73**, 325 (1987).

[10] Z.P. Nowak and W. Hackbusch, "On the Complexity of the Panel Method", International Conference on Modern Problems in Numerical Analysis, Moscow, September, 1986 (unpublished).

[11] L. Reichel, *J. Comp. Math.* **14**, 125 (1986).

[12] L. Reichel, Report ICM-9, Institute for Constructive Mathematics, University of South Florida, Tampa, FL, 1986 (unpublished).

[13] V. Rokhlin, *J. Comp. Phys.* **60**, 187 (1985).

[14] S. Ta'asan, Ph.D. Thesis, The Weizmann Institute of Science, Rehovot, Israel, 1984 (unpublished).

[15] A. Young, *Proc. Roy. Soc. Lond.* **A224**, 552 (1954).