Contemporary Mathematics
Volume **180**, 1994

ELLIPTIC PRECONDITIONERS USING FAST SUMMATION TECHNIQUES

L. RIDGWAY SCOTT

ABSTRACT. We introduce the idea of using fast summation methods together with Green's function techniques to reduce the work in approximating the inverse matrix for the discretization of elliptic partial differential equations. The resulting method has a high degree of parallelism.

There are several reasons for studying the iterative methods introduced in this paper. For perspective, we list two. One arises in multigrid solvers, where there is a need to have a fast method on the coarsest grid. The coarse grid equation is solved repeatedly and may be of substantial size in engineering applications. In parallel multigrid solvers, the "coarsest" may need to be coarser than in the sequential case in order to keep good parallel efficiency. Another application of these methods arises in time-stepping methods, whether linear or nonlinear, where a particular system is inverted repeatedly in many algorithms [2]. If time integration is long, typically an implicit method would be employed which requires solving a linear system. In this case, the efficiency of the linear solver is critical. Such a system could be so small and irregular that a multi-level procedure would not be considered. We now elaborate these two examples.

Multigrid techniques for solving the linear systems arising from the discretization of elliptic partial differential equations involve a reduction in problem size to a coarse grid. This coarse grid may be relatively large due to the need to resolve the domain geometry (or the problem solution). Since the coarse-grid problem must be solved repeatedly, the limiting factor in the efficiency of the overall method may be the ability to solve the coarse-grid problem quickly. On

1991 *Mathematics Subject Classification.* Primary 65N22, 65N55, 65Y05; Secondary 65NF10, 65N06, 65N30.

We acknowledge generous support from the National Science Foundation, award number ASC-9217374 (which includes funds from DARPA), and award number DMS-9105437.

This paper is in final form and no version of it will be submitted for publication elsewhere.

the other hand, it is not necessary to solve the coarse-grid problem very accurately in the context of multigrid techniques. If a fast and sufficiently accurate preconditioner is available, then simple iteration can be used to produce a suitable approximation to the solution to the coarse-grid problem. We introduce and analyze a new technique for doing this using fast summation methods [6].

In practical problems, the coarsest mesh that can describe the geometry is quite large. Since the resulting linear system related to the coarse grid is unstructured, a popular technique for for solving it is direct factorization [7]. This has the advantage that the repeated coarse-grid solves require only back-solves, after the initial factorization. This can be quite efficient for two-dimensional problems if the unknowns are suitably ordered [7]. However, for three-dimensional problems even optimal orderings lead to work estimates that are less than ideal (cf. [3]). Moreover, the sequential nature of a back-solution makes it difficult to parallelize efficiently. The techniques presented here have the same level of efficiency independent of the domain dimension, and they are trivially parallel.

The basic idea of the technique is as follows. The solution to a linear system can be written as the product of the inverse matrix and the data vector. Moreover, this has a significant amount of parallelism in that each component of the solution can be computed independently of the others. The difficulty is that, for an $n \times n$ system, this requires $\mathcal{O}(n^2)$ work. However, the inverse matrix for the discretization of elliptic partial differential equations has significant structure that can be exploited to approximate its action with significantly less work. We explore this in detail in the context of finite element methods. We introduce the idea of using fast summation methods [6] together with Green's function techniques [9] to reduce the work to $\mathcal{O}(n \log n)$ in the sequential case. The key point is that this technique does not use any particular structure of the mesh, only structure of the differential equation being solved. Fast summation methods can be viewed as adaptive domain decomposition methods together with a projection on each subdomain.

1. A Model Problem

Let $\Omega \subseteq \mathbb{R}^d$ be a convex polygon and define

$$a(u,v) = \int_\Omega \nabla u \cdot \nabla v \, dx. \tag{1.1}$$

We consider the variational formulation of the Dirichlet problem for Laplace's equation, as follows: find $u \in V := \mathring{H}^1(\Omega)$ such that

$$a(u,v) = (f,v) \quad \forall \ v \in V \tag{1.2}$$

where $f \in L^2(\Omega)$.

Elliptic regularity (cf. Grisvard [8]) implies that $u \in H^2(\Omega) \cap \mathring{H}^1(\Omega)$. To approximate u, we consider a triangulation $\mathcal{T}$ of Ω, where h denotes the mesh size of $\mathcal{T}$. Let $V_{\mathcal{T}}$ denote C^0 piecewise polynomial functions of degree $\leq k$ with respect to $\mathcal{T}$ that vanish on $\partial\Omega$ [4]. The discretized problem is the following: find $u_h \in V_{\mathcal{T}}$ such that

$$a(u_h, v) = (f, v) \quad \forall\ v \in V_{\mathcal{T}}. \tag{1.3}$$

It is well known [4] that

$$\|u - u_h\|_{H^s(\Omega)} \le C\,h^{k+1-s}\,\|u\|_{H^{k+1}(\Omega)} \text{ for } s = 0, 1 \text{ and } 1 \le m \le k+1. \tag{1.4}$$

Throughout this paper C, with or without subscripts, denotes a generic constant independent of h.

The Green's function, g^x, for (1.2) satisfies

$$v(x) = a(g^x, v) \tag{1.5}$$

for suitably smooth functions v, say $v \in W^1_p(\Omega)$ for $p > d$. By Sobolev's inequality and [12] we see that

$$\|g^x\|_{W^1_q(\Omega)} \le C \sup_{0 \ne v \in \mathring{W}^1_p(\Omega)} \frac{a(g^x, v)}{\|v\|_{W^1_p(\Omega)}} \le C \tag{1.6}$$

where $\frac{1}{p} + \frac{1}{q} = 1$.

Correspondingly, there is a discrete Green's function, $g^x_h \in V_{\mathcal{T}}$, which satisfies

$$v(x) = a(g^x_h, v) \quad \forall\, v \in V_{\mathcal{T}}. \tag{1.7}$$

Using this discrete Green's function, we may write the solution to (1.3) via

$$u_h(x) = a(g^x_h, u_h) = (f, g^x_h) \quad \forall\, x \in \Omega.$$

If we let $\{x_i : 1 \le i \le \dim V_{\mathcal{T}}\}$ denote the interior vertices of $\mathcal{T}$, then the coefficients of u_h with respect to the standard Lagrange basis, $\{\psi_i : 1 \le i \le \dim V_{\mathcal{T}}\}$, for $V_{\mathcal{T}}$ can be determined via

$$u_h(x_i) = (f, g^{x_i}_h) \quad \forall\, i = 1, \dots, \dim V_{\mathcal{T}}. \tag{1.8}$$

Thus the $g^{x_i}_h$ are closely related to the rows of the inverse matrix for the problem (1.3).

2. Approximating the Green's function

The basic singularity of g^x is given by

$$G^x(y) = \begin{cases} \log|x - y| & d = 2 \\ |x - y|^{2-d} & d \ge 3 \end{cases} \tag{2.1}$$

in the sense that there is a constant α such that

$$g^x - \alpha G^x \in \mathring{H}^1(\Omega).$$

See Figure 2.1 which depicts the discrete Green's function with singularity near one of the corners of a square domain. In fact, we can think of defining g^x by writing $g^x = w^x + \alpha G^x$ where $w^x = -\alpha G^x$ on $\partial\Omega$ and

$$a(w^x, v) = 0 \quad \forall\ v \in V.$$

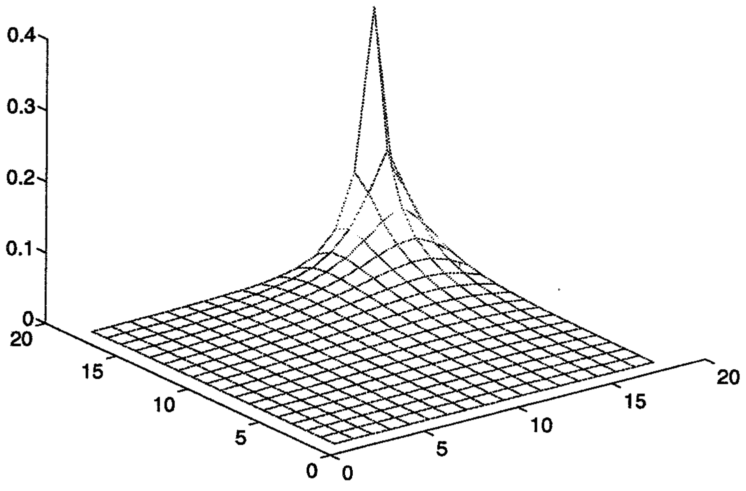

Figure 2.1. Plot of the discrete Green's function with singularity near one of the corners of a square domain with a regular mesh.

The integrals $(f, g_h^{x_i})$ can be approximated efficiently since the Green's function is quite smooth away from the singularity (see Figure 2.1). Near the singularity we compute the integral exactly, and far away we replace $g_h^{x_i}$ by appropriate averages. To define the process precisely, we fix $x = x_i \in \Omega$ for the moment. Let $\{S_j : j = 1, \ldots, J_h^x\}$ be a subdivision of Ω consisting of groups of triangles in $\mathcal{T}$ with the following property:

$$\operatorname{diam}(S_j) \le \rho \inf\{|x-y| : y \in S_j\} \quad \forall\, j = 2, \ldots, J_h^x \tag{2.2}$$

for some constant $\rho < \infty$. Figure 2.2 depicts a subdivision with $\rho = 2$. Thus S_1 consists of the triangles close to the point x whereas the other members of the subdivision have a size comparable to the distance from the closest point to x.

We begin by describing a piecewise constant approximation. Let

$$\gamma_j^x := \frac{1}{\operatorname{meas}(S_j)} \int_{S_j} g_h^x(y)\, dy \tag{2.3}$$

and define $K^x : L^1(\Omega) \to \mathbb{R}$ by

$$K^x f := \int_{S_1} g_h^x(y) f(y)\, dy + \sum_{j=2}^{J_h^x} \gamma_j^x \int_{S_j} f(y)\, dy. \tag{2.4}$$

Define $\mathcal{K} : L^1(\Omega) \to V_{\mathcal{T}}$ via

$$\mathcal{K}f := \sum_{i=1}^{\dim V_{\mathcal{T}}} (K^{x_i} f)\, \psi_i.$$

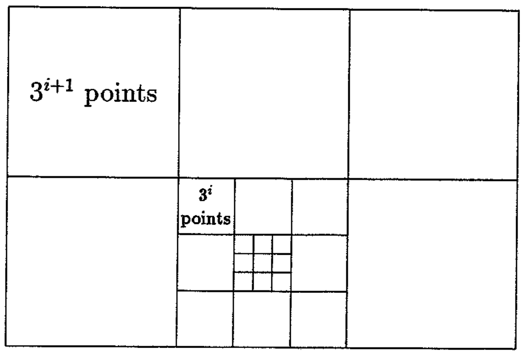

Figure 2.2. Mesh subdivision (aggregation of triangles in $\mathcal{T}$) where $\rho = 2$ in condition (2.2).

Define a function e^x on Ω by

$$e^x(y) = \begin{cases} 0 & \forall\, y \in S_1 \\ g_h^x(y) - \gamma_j^x & \forall\, y \in S_j,\ j \geq 2 \end{cases}$$

for all $x = x_i$. We have the following error representation:

$$u_h(x) - K^x f = \int_\Omega e^x(y) f(y)\, dy. \tag{2.5}$$

We can define a linear operator $\mathcal{I}^x$ by

$$\mathcal{I}^x v(y) = \begin{cases} v(y) & \forall\, y \in S_1 \\ \frac{1}{\operatorname{meas}(S_j)} \int_{S_j} v(z)\, dz & \forall\, y \in S_j,\ j \geq 2 \end{cases}$$

for all $x = x_i$. Using this notation, (2.4) becomes

$$K^x f := \int_\Omega (\mathcal{I}^x g_h^x)(y)\, f(y)\, dy,$$

and (2.5) becomes

$$(u_h - \mathcal{K}f)(x) = \int_\Omega \left(g_h^x - \mathcal{I}^x g_h^x\right)(y) f(y)\, dy. \tag{2.6}$$

More generally, we can imagine defining a family of linear operators $\mathcal{I}^x$ for $x \in \Omega$, each of which is the identity on (the respective) Ω_1 and such that $\mathcal{I}^x v$ is a polynomial of degree r in each S_j, $j \geq 2$, and that (2.6) holds. In the case that $\mathcal{I}^x v \in V_h$ for $v \in V_h$, we may write

$$(u_h - \mathcal{K}f)(x) = a(u_h, g_h^x - \mathcal{I}^x g_h^x).$$

We may formally think of $u_h = A_h^{-1} f$, where A_h denotes the operator associated with the variational problem (1.3). Thus, this becomes

$$\left(A_h^{-1} - \mathcal{K}\right) f(x) = a(u_h, g_h^x - \mathcal{I}^x g_h^x),$$

or, since $f = A_h u_h$, we may write

$$(\mathcal{I} - \mathcal{K}A_h)\, u_h(x) = a(u_h, g_h^x - \mathcal{I}^x g_h^x).$$

Let $y = x_j$ for some other nodal point, and consider the quantity

$$(\mathcal{I} - \mathcal{K}A_h)\, u_h(x) - (\mathcal{I} - \mathcal{K}A_h)\, u_h(y) = a(u_h, e^x - e^y)$$

where $e^x := g_h^x - \mathcal{I}^x g_h^x$. Dividing by $|x-y|$ and taking the maximum over all x and y, we find

$$\|(\mathcal{I} - \mathcal{K}A_h)\, u_h\|_{W^1_\infty(\Omega)} \leq C\, \|u_h\|_{W^1_\infty(\Omega)} \sup_{x,y \in \Omega} \frac{\|e^x - e^y\|_{W^1_1(\Omega)}}{|x-y|}.$$

Thus we have proved the following.

THEOREM 2.7. *There is a constant C depending only on the variational problem such that*

$$\|\mathcal{I} - \mathcal{K}A_h\|_{W^1_\infty(\Omega) \to W^1_\infty(\Omega)} \leq C \sup_{x,y \in \Omega} \frac{\|e^x - e^y\|_{W^1_1(\Omega)}}{|x-y|}$$

where $e^x := g_h^x - \mathcal{I}^x g_h^x$.

3. Error analysis

We are unable to give a complete error analysis of the term arising in Theorem 2.7, but we can give some partial results as an indicator as to what might be important factors effecting the method. We expand the expression $e^x - e^y$ as

$$\begin{aligned} e^x - e^y &= g_h^x - \mathcal{I}^x g_h^x - g_h^y + \mathcal{I}^y g_h^y \\ &= (\mathcal{I} - \mathcal{I}^x)\,(g_h^x - g_h^y) - (\mathcal{I}^x - \mathcal{I}^y)\, g_h^y. \end{aligned}$$

We will show that we can write

$$g_h^x - g_h^y = |x-y| \int_0^1 \tilde{g}_h^{y+t(x-y)}\, dt \tag{3.1}$$

where $\tilde{g}_h^z$ denotes the derivative Green's function [9] defined by

$$a(\tilde{g}_h^z, v) = \frac{(x-y)}{|x-y|} \cdot \nabla v(z) \quad \forall\, v \in V_h. \tag{3.2}$$

This can be seen by using the fundamental theorem of calculus,

$$\int_0^1 \frac{(x-y)}{|x-y|} \cdot \nabla v(y+t(x-y))\,dt = v(x) - v(y),$$

and using the definition of the discrete Green's function. Note that the integral on the right-hand side of (3.1) defines a function in the finite dimensional space V_h, and

$$\int_0^1 a\left(\tilde{g}_h^{y+t(x-y)}, v\right)\,dt = a\left(\int_0^1 \tilde{g}_h^{y+t(x-y)}\,dt, v\right)$$

because the map $z \to \tilde{g}_h^z$ is a continuous map from $\Omega \to V_h$.

Applying (3.1), we find

$$|x-y|^{-1}\,\|(\mathcal{I}-\mathcal{I}^x)\,(g_h^x - g_h^y)\|_{W_1^1(\Omega)} \leq \max_{t\in[0,1]} \left\|(\mathcal{I}-\mathcal{I}^x)\,\tilde{g}_h^{y+t(x-y)}\right\|_{W_1^1(\Omega)}$$

From [9], we have

$$\|(\mathcal{I}-\mathcal{I}^x)\,\tilde{g}_h^z\|_{W_1^1(\Omega)} \leq C\,\|(\mathcal{I}-\mathcal{I}^x)\,\tilde{g}^z\|_{W_1^1(\Omega)} + \|(\mathcal{I}-\mathcal{I}^x)\,(\tilde{g}^z - \tilde{g}_h^z)\|_{W_1^1(\Omega)}$$

where $\tilde{g}^z$ solves (1.2) with f being a directional derivative of a mollified Dirac delta function [9]. Then from (2.2) we find

$$\begin{aligned}
\|(\mathcal{I}-\mathcal{I}^x)\,\tilde{g}^z\|_{W_1^1(\Omega)} &\leq C\sum_{j=2}^{J_h^x} \operatorname{diam}(S_j)^r \int_{S_j} |z-y|^{-n-r}\,dy \\
&\leq C\rho^r \sum_{j=2}^{J_h^x} \int_{S_j} |z-y|^{-n}\,dy \\
&\leq C\rho^r \int_{\Omega\setminus S_1} |z-y|^{-n}\,dy \\
&\leq C\rho^r |\log(\operatorname{diam}(S_1))|.
\end{aligned} \tag{3.3}$$

By choosing ρ sufficiently small (or r sufficiently large, if $\rho < 1$), we can make this term as small as we like.

Correspondingly from [11] and [9] we deduce that

$$\begin{aligned}
\|(\mathcal{I}-\mathcal{I}^x)\,(\tilde{g}^z - \tilde{g}_h^z)\|_{W_1^1(\Omega)} &\leq C\,\|\tilde{g}^z - \tilde{g}_h^z\|_{W_1^1(\Omega\setminus S_1)} \\
&\leq Ch^k \operatorname{diam}(S_1)^{-k}.
\end{aligned} \tag{3.4}$$

Provided $h/\operatorname{diam}(S_1)$ is sufficiently small, this term will be small.

Estimates (3.3) and (3.4) are in competition to the extent that the former increases and the latter decreases as a function of $\operatorname{diam}(S_1)$. The latter requires $\operatorname{diam}(S_1)$ to be comparable to h and then the former requires $\rho^r|\log h|$ to be small. For fixed $\rho < 1$, this means that r must be chosen to grow as $\mathcal{O}(\log|\log h|)$ as h tends to zero.

To complete the analysis, we need to consider an estimate for

$$|x-y|^{-1}\,\|(\mathcal{I}^x - \mathcal{I}^y)\,g_h^y\|_{W_1^1(\Omega)}\,.$$

This appears to require some restrictions on the regularity of the interpolation process. However, we postpone further analysis to a later publication.

4. Complexity of initialization

The application of the operator $\mathcal{K}$ requires a great deal of initial computation. In particular, information related to the complete inverse of A_h is apparently necessary. We now address the question of how much work is involved and to what extent this work can be done in parallel. There are several contexts in which one might be able to assess (and amortize) the cost of initialization. In a time stepping scheme, this cost could be compared with the savings accrued over a large number of time steps. In a multigrid application, the cost of initialization for a coarse grid solver can be compared with the cost of a fine grid sweep. This allows a more precise comparison, so we carry this out in some detail in an example.

We consider an example based on regular meshes in d dimensions. Suppose the fine mesh consists of N^d points, and the coarse mesh consists of n^d points. We consider the construction of the operator $\mathcal{K}$ on the coarse mesh. We need to solve n^d equations of the form (1.3) for $f = \delta^x$. Each of these can be done in parallel, as they are completely independent. The complexity of solution of (1.3) depends on the structure of the problem, but we make the simplifying assumption that we can use multigrid and solve each one in an amount of work $\mathcal{O}(n^d)$.

Let us assume that the S_j's are based on the coarser meshes arising in a multigrid solution. The formation of the required averages (2.3) can be done as follows. We describe the process for a triangular mesh in two dimensions ($d = 2$). First, we form the averages on pairs of neighboring triangles which form a rectangle. This takes $\mathcal{O}(n^d)$ work. Then we average these in groups of four rectangles, forming averages over S_j's consisting of eight triangles. This takes again $\mathcal{O}(n^d)$ work, but produces only one quarter as many new averages. Grouping these averages again in groups of four rectangles and forming averages over such groups takes only one quarter as much work as before and produces only one quarter as many new averages. Since these are progressing geometrically, we see the overall work and storage is $\mathcal{O}(n^d)$. The number of terms arising in (2.4) is $\mathcal{O}(\log n)$, but the averages of f can similarly be computed in $\mathcal{O}(n^d)$ work. These averages must be computed for each of $\mathcal{O}(n^d)$ points, but they can be done completely in parallel.

Thus the total work for the initialization phase takes

$$\mathcal{O}\left(\frac{n^{2d}}{P}\right) \quad \text{work}$$

for P processors provided $1 \leq P \leq n^d$. This is to be compared with the cost of a fine grid sweep, which can be done (for certain smoothers [1]) in

$$\mathcal{O}\left(\frac{N^d}{P}\right) \quad \text{work}$$

for P processors provided $1 \le P \ll N^d$. Here we are assuming N^d is large with respect to P in order to ignore the cost of communication in, say, a typical V cycle.

Thus the cost of initialization will be comparable to a fine grid sweep when

$$n^2 = \mathcal{O}(N), \tag{4.1}$$

and this result is essentially independent of the number of processors, provided $1 \le P \le n^d = \mathcal{O}(N^{d/2})$, and independent of dimension. To quantify this, we note that (4.1) holds when $n = 10$ and $N = 100$; this corresponds to a typical three-dimensional problem [1] by today's standards and would allow for $P = 1000$. It also holds for $n = 30$ and $N = 1000$, and this coarse grid size corresponds to some of the two-dimensional examples presented subsequently. The natural amount of parallelism would be $P = \mathcal{O}(n^d) = \mathcal{O}(N^{d/2})$ in general, and thus it is scalable.

5. Complexity of the fast-summation algorithm

In the computation of (2.4) in parallel, the averages of f have to be accumulated at each processor. After a communication phase to be described subsequently, each processor does

$$\mathcal{O}\left(\frac{n^d \log n}{P}\right) \tag{5.1}$$

floating point operations as depicted in (2.4) without further communication. We now compare the cost of communication with (5.1).

It will not be necessary to have all averages at each processor. For processors far away, only the coarser averages are needed. These can be accumulated as follows. For simplicity, we assume that data is distributed in a parallel computer having a network that has a d-dimensional mesh as a subnetwork. In particular, we assume that each processor can communicate with a neighbor in a d-dimensional mesh simultaneously, without contention. To keep the analysis simple, we ignore the effect of the degree of approximation r which would grow like $\mathcal{O}(\log|\log h|)$ as h tends to zero.

First, we form the averages on groups of rectangles residing completely within one processor; this takes $\mathcal{O}\left(\frac{n^d}{P}\right)$ floating point operations. For simplicity, let us assume that groups of rectangles of a particular size fit precisely in one of the processors, so that only averages over boxes (rectangles) of larger size require an exchange of data. More precisely, we assume the set of triangles assigned to a processor forms one of the groups S_j. In $\mathcal{O}(\log P)$ steps, these P values can be accumulated at each processor, and the remaining combinations can be done redundantly.

Let λ denote the number of floating point operations that can be done in the amount of time corresponding to the latency of communication, and let F denote the number of floating point operations that can be performed in the time that it takes to communicate one word (typical values for λ would range from ten to one hundred and for F would range from three to thirty.) Then the time for these combinations is comparable to

$$\mathcal{O}(\lambda \log(P) + FP)$$

floating point operations. The data to be reduced locally corresponds to a d-dimensional mesh with P cells. Doing the remaining combinations thus requires $\mathcal{O}(P)$ floating point operations. Thus the majority of time is spent in the communication of the initial averages (assuming $F \gg 1$).

In a fixed number of processors around each given processor, local information will need to be communicated. The number of such processors is bounded by $\mathcal{O}(\rho^d)$. This contributes an amount of time [5] comparable to

$$\mathcal{O}\left(2d\left\lceil \rho^d \right\rceil \left(\lambda + F\frac{n^d}{P}\right)\right)$$

floating point operations, where $\lceil x \rceil$ indicates the smallest integer not less than x.

If P is larger than $\frac{n^d}{P}$ ($P \geq n^{d/2}$), then the algorithm above for combining averages over larger boxes may not be optimal. Instead a series of local combinations can be done. At the first stage, there are P averages; neighbors exchange their averages (in $2d$ steps) and then compute averages over 2^d neighboring boxes. This produces $P/2^d$ new averages. One of the processors from each group of 2^d is chosen to exchange the resulting average with other groups of 2^d processors in $2d$ steps. These values are averaged and the value is passed back to all 2^d processors in each subset in $2d$ steps. This process is then repeated recursively, by subdividing the subsets into subsets. This clearly terminates after $\mathcal{O}(\log P)$ steps, since the box size is doubling at each step. The number of communication steps is $\mathcal{O}\left(2^d \log P\right)$, with a constant amount of data exchanged at each step. This corresponds to an amount of time equivalent to

$$\mathcal{O}\left(\lambda 2^d \log P\right)$$

floating point operations. With the assumption $P \leq n^d$, this corresponds to less than

$$\mathcal{O}\left(\lambda 2^d d \log n\right)$$

floating point operations.

Thus, the amount of time required to do the computation in (2.4) is of the order of the communication required (or greater). For a machine with large latency λ, it could be useful to attempt to refine further the exchanges and computation of the averages, if it is desired to have $P \approx n^d$.

6. Numerical experiments

We now describe some numerical experiments done to explore the viability of the approximate inverse described in section 2. We took the simplest possible situation, namely, a regular mesh in two dimensions. Thus A_h corresponds to the five-point difference stencil. Moreover, we took a quite simple approach to the averaging. In the first case, which we refer to as the "constant" case, we approximate g_h^x by piecewise constants, averaging over boxes of size 3^i mesh points as indicated in Figure 6.1 in the "constant-1" case and Figure 6.2 in the "constant-2" case. More precisely, in the " -1" aggregation scheme, there are annuli made up of eight boxes of size 3^i mesh points in a square annulus surrounding a similar arrangement for $i-1$. This is depicted in Figure 2.2.

The " -2" aggregation scheme is more difficult to describe, but the boxes utilized are apparent from Figure 6.2.

Table 6.1 indicates the spectral radius for the various approximation schemes. The different aggregation schemes are indicated by the appendage " -1" for the coarser aggregation (see Figure 2.2) and " -2" for the finer aggregation (see Figure 6.2). In the "linear" cases, we instead interpolate g_h^x bilinearly, but not continuously. For the sake of reference, we give the spectral radius of the red/black SOR iterative method (with the optimal ω), as it is a well known iterative method with a comparable level of parallelism.

Mesh size	constant-1	constant-2	linear-1	linear-2	R/B SOR
8×8	0.9248	0.7031	0.3785	0.1604	0.4903
10×10	1.4372	1.0288	0.5075	0.2035	0.5604
12×12	1.9436	1.5778	0.6284	0.2345	0.6138
14×14	2.4531	2.0219	0.5765	0.2593	0.6558
16×16	3.0231	2.5137	0.8618	0.2796	0.6895
18×18	3.5189	2.9493	0.9377	0.2962	0.7173
20×20	3.9470	3.4080	0.9976	0.3096	0.7406
22×22	4.4126	3.8490	1.0642	0.3206	0.7603
24×24	4.8771	4.3090	1.2120	0.3296	0.7773
26×26	5.3262	4.8157	1.3855	0.3371	0.7920

Table 6.1. Spectral radius for various iterative schemes. The columns marked "constant" and "linear" contain the spectral radius of $\mathcal{I}-\mathcal{K}A_h$ for different mesh schemes indicated in Figures 6.2 (" -1") and 6.3 (" -2"), respectively. The column marked "R/B SOR" gives the spectral radius for the red/black SOR iterative method (with the optimal ω). The "Mesh size" indicates the number of mesh points in each direction of a regular two-dimensional mesh.

What we conclude from Table 6.1 is that the method certainly works, but not arbitrarily well. In particular, it may be necessary to take the estimates (3.3) and (3.4) seriously, increasing the order r of approximation and reducing the constant ρ in condition (2.2), which controls the grading of the mesh, appropriately.

We note that the matrix $\mathcal{K}$ as defined here is not symmetric, but it is very nearly symmetric. We computed the norm of $\mathcal{I} - \widetilde{\mathcal{K}}A_h$, where $\widetilde{\mathcal{K}} = \frac{1}{2}(\mathcal{K} + \mathcal{K}^t)$ is the symmetric part of $\mathcal{K}$, and the results agreed with those in the table to three significant digits.

References

[1] B. Bagheri, A. Ilin & L. R. Scott, Parallelizing UHBD, Research Report UH/MD 167, Dept. Math., Univ. Houston, 1993, and Parallel 3-d mosfet simulation, In *Proceedings of the* 27th *Annual Hawaii International Conference on System Sciences* vol. 1, T.N.Mudge and B.D. Shriver, ed's, IEEE Computer Soc. Press, 1994, pp. 46–54.

[2] B. Bagheri, S. Zhang & L. R. Scott, Implementing and using high–order finite element methods, *Comp. Meth. Appl. Mech. & Eng.*, to appear.

[3] R. E. Bank & L. R. Scott, On the conditioning of finite element equations with highly refined meshes, *SIAM J. Num. Anal.* **26**, (1989), 1383–1394.

[4] S. Brenner, L. R. Scott, *The Mathematical Theory of Finite Element Methods*, Springer-Verlag, 1994.

[5] T. W. Clark, R. v. Hanxleden, J. A. McCammon, and L. R. Scott, Parallelizing molecular dynamics using spatial decomposition, Proc. Scalable High-Performance Computing Conference, IEEE Computer Soc. Press, 1994, 95-102.

[6] C. I. Draghicescu, An efficient implementation of particle methods for the incompressible Euler Equations, *SIAM J. Numer. Anal.* **31** (1994), to appear.

[7] A. George, J. W. Liu, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice-Hall, 1981.

[8] P. Grisvard, *Elliptic Problems in Nonsmooth Domains*, Pitman Advanced Publishing Program, Boston, 1985.

[9] R. Rannacher and L. R. Scott, Some optimal error estimates for piecewise linear finite element approximations, *Math. Comp.* **38** (1982), 437–445.

[10] L. R. Scott, Optimal L^∞ estimates for the finite element method on irregular meshes, *Math. Comp.* **30** (1976), 681–697.

[11] L. R. Scott and S. Zhang, Finite element interpolation of non–smooth functions satisfying boundary conditions, *Math. Comp.*, 54:483–493, 1990.

[12] Simader, C. G. (1972) *On Dirichlet's Boundary Value Problem*, Lecture Notes in Math. v. 268, Springer-Verlag, Berlin, 1972.

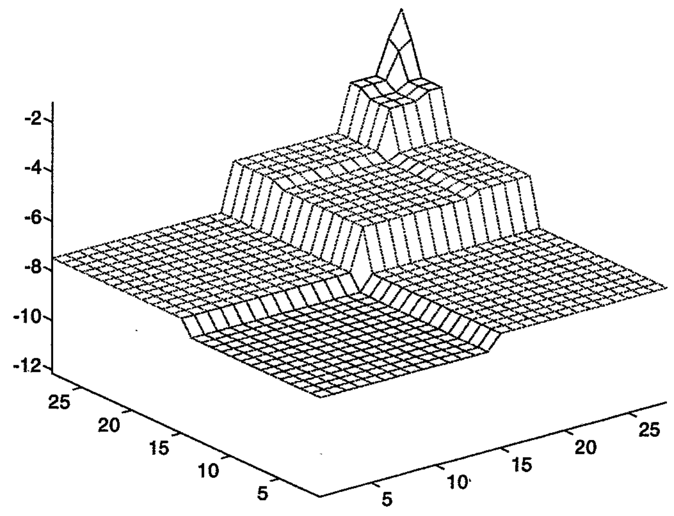

Figure 6.1. Plot of the logarithm of the piecewise constant approximation of the discrete Green's function. Coarse mesh aggregation where $\rho = 2$ in condition (2.2) as depicted in Figure 2.2.

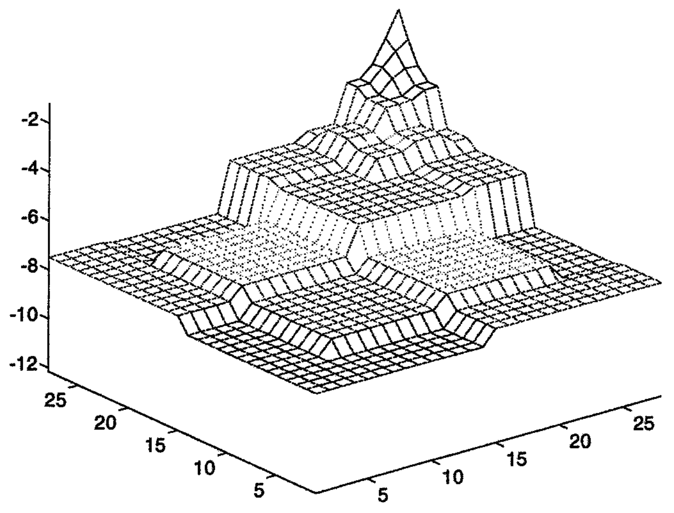

Figure 6.2. Plot of the logarithm of the piecewise constant approximation of the discrete Green's function. Finer mesh aggregation where $\rho = 2/3$ in condition (2.2).

Department of Mathematics, University of Houston, Houston, TX 77204-3476
E-mail address: scott@uh.edu