Acta Numerica (2009), pp. 243–275
doi: 10.1017/S0962492906410011
© Cambridge University Press, 2009
Printed in the United Kingdom

Fast direct solvers for integral equations in complex three-dimensional domains

Leslie Greengard*
Courant Instiute of Mathematical Sciences, New York University, New York, NY 10012, USA
E-mail: greengard@courant.nyu.edu

Denis Gueyffier†
Courant Institute of Mathematical Sciences, New York University, New York, NY 10012, USA
E-mail: dgueyffier@giss.nasa.gov

Per-Gunnar Martinsson‡
Department of Applied Mathematics, University of Colorado at Boulder, 526 UCB, Boulder, CO 80309-0526, USA
E-mail: per-gunnar.martinsson@colorado.edu

Vladimir Rokhlin§
Department of Mathematics and Department of Computer Science, Yale University, 10 Hillhouse Avenue, New Haven CT 06511, USA
E-mail: rokhlin@cs.yale.edu

* This work was supported in part by the Applied Mathematical Sciences Program of the US Department of Energy under Contract DEFG0200ER25053.
† This work was supported by DARPA through the Protein Design Processes Program (DSO contract HR0011-05-1-0044). Present address: NASA GISS and Columbia University, 2880 Broadway, New York, NY 10025.
‡ This work was supported in part by National Science Foundation grants DMS-0748488 and DMS-0610097.
§ This work was supported in part by AFOSR grant FA9550-07-1-0541.

https://doi.org/10.1017/S0962492906410011 Published online by Cambridge University Press

Methods for the solution of boundary integral equations have changed significantly during the last two decades. This is due, in part, to improvements in computer hardware, but more importantly, to the development of fast algorithms which scale linearly or nearly linearly with the number of degrees of freedom required. These methods are typically iterative, based on coupling fast matrix-vector multiplication routines with conjugate-gradient-type schemes. Here, we discuss methods that are currently under development for the fast, direct solution of boundary integral equations in three dimensions. After reviewing the mathematical foundations of such schemes, we illustrate their performance with some numerical examples, and discuss the potential impact of the overall approach in a variety of settings.

CONTENTS

1 Introduction 244
2 The rapid application of operators 246
3 Fast direct solvers for structured matrices 252
4 Potential theory for the Poisson equation 254
5 A single-level fast solver 258
6 Numerical results I 262
7 Local geometric perturbations 263
8 Numerical results II 265
9 Conclusions 267
Appendix: Rapid inversion of operators 268
References 272

1. Introduction

Modern numerical methods for the solution of boundary integral equations with large numbers of degrees of freedom are currently based on the availability of fast algorithms for matrix-vector multiplication (application of the discretized integral operator). These include fast multipole methods, panel clustering methods, the method of local corrections, multigrid, precorrected-FFT and wavelet-based methods. Briefly stated, discretization of a boundary integral equation yields a dense $N \times N$ matrix, where N denotes the number of degrees of freedom used in describing the surface density defined on the given geometry. As a result, classical solution techniques, based on Gaussian elimination, require $O(N^3)$ work to factor the system matrix. The fast algorithms listed above provide the ability to apply the discretized integral operator to a vector in $O(N)$ or $O(N \log N)$ operations rather than $O(N^2)$. The combination of such a scheme with modern iterative methods – such as GMRES (Saad and Schultz 1986) – and well-conditioned integral

https://doi.org/10.1017/S0962492906410011 Published online by Cambridge University Press

equation formulations has reduced the net work required to $O(N \log N)$, bringing large-scale simulations within practical reach. The literature on this subject is now vast, and we refer the reader to only a few relevant publications: Carrier, Greengard and Rokhlin (1988), Chew, Jin, Michielssen and Song (2001), Darve and Have (2004), Greengard and Helsing (1998), Greengard and Rokhlin (1987, 1997), Hackbusch and Nowak (1989), Kapur and Long (1997), Nabors and White (1991), and Nishimura (2002).

Nevertheless, there are a number of areas where significant improvement can still be made. First, iterative methods deal poorly with ill-conditioned problems or with multiple right-hand sides. This is in marked contrast to classical direct methods which, following the $O(N^3)$ factorization step, require only $O(N^2)$ work to solve the linear system for each subsequent right-hand side. A second motivation for direct methods is that they are extremely effective at handling low-rank perturbations of the system matrix. In particular, the solution to the perturbed system can also be obtained in $O(N^2)$ work. An obvious question, then, is whether direct methods can be accelerated in the same way that iterative solvers have been.

In the last few years, a number of groups have been developing fast direct solvers for boundary integral equations in two and three dimensions that first compute a new type of 'compressed' factorization using $O(N^\alpha \log^\beta N)$ operations, with $1 \leq \alpha \leq 2$ and $0 \leq \beta \leq 2$. Application of the factored inverse then typically requires only $O(N)$ or $O(N \log N)$ operations for each right-hand side and/or low-rank perturbation of the system matrix (Canning and Rogovin 1998, Chandrasekaran *et al.* 2006, Gope, Chowdhury and Jandhyala 2005, Hackbusch 1999, Hackbusch and Khoromskij 2000, Martinsson and Rokhlin 2005, Pals 2004, Zhu and White 2005). Earlier work on direct solvers had focused primarily on volume scattering (Chen 2002, Chew 1989) or the one-dimensional or quasi-one-dimensional case (Eidelman and Gohberg 1999, Greengard and Rokhlin 1991, Lee and Greengard 1997, Martinsson and Rokhlin 2007, Michielssen, Boag and Chew 1996, Starr and Rokhlin 1994).

The main purpose of this paper is to state the obvious; fast, direct methods which result in compressed or 'data sparse' representations of the inverse matrix should have a dramatic impact on simulation and design. Stress analysis, electromagnetic analysis (including radar cross-section calculations), biophysical simulation, and a host of other tasks can be formulated in terms of a given integral equation with multiple right-hand sides (Martinsson 2006). Further, when solving time-dependent partial differential equations in a fixed geometry, a computational bottleneck is often the solution of a potential problem such as the Laplace equation with a new right-hand side at each time step. A second motivation for fast direct solvers is that the solution to 'nearby' problems can be computed very efficiently. In particular, if the geometry undergoes a local perturbation, this

https://doi.org/10.1017/S0962492906410011 Published online by Cambridge University Press

introduces a low-rank perturbation of the system matrix and, as indicated above, rapid updating becomes feasible.

In the next two sections, we review the basic idea that allows for the construction of fast solvers. We then show how to apply these ideas to dielectric interface problems and the exterior Neumann problem in three dimensions, both governed by the Laplace equation. For this, we extend the approach introduced by Martinsson and Rokhlin (2005) for two-dimensional problems. Our formalism, however, is a bit different and leads to some simplification in both presentation and implementation. We illustrate the power of the method for both molecular electrostatics and incompressible potential flow and conclude with some remarks about future directions of research.

2. The rapid application of operators

For uniformly discretized convolution operators, the problem of excessive cost of applying (or inverting) dense matrices was solved with the development of the Fast Fourier Transform (FFT) and related algorithms (Cooley and Tukey 1965). These methods were discovered during the 1960s, and attain their computational speed by exploiting algebraic properties of the operator. Such schemes are exact in exact arithmetic, and are fragile in the sense that they depend on regularity of the input and output data for their applicability. While the FFT revolutionized digital signal processing and a number of other fields, the ability to handle irregular data is often essential. Moreover, many of the integral operators encountered in mathematical physics are not translation-invariant.

Twenty years later, it was observed that many operators originating from physics admit a different type of 'fast' method. Specifically, the kernels of many such operators are smooth in the far field. When operators of this type are discretized, the resulting matrices contain large submatrices whose rank is low (to high precision). Combining this fact with simple structures from computer science (adaptive quadtrees, oct-trees, *etc.*), it is possible to construct algorithms for the application of such matrices to arbitrary vectors for a cost proportional to N in some situations, and $N \log(N)$ in others. Curiously, the first several algorithms of this type did not use the rank argument explicitly; they were restricted to cases where the operator had some special analytical structure, and used the corresponding special functions, for instance, multipole expansions for the Laplace equation in Greengard and Rokhlin (1987), Hermite polynomials in Greengard and Strain (1991), Laguerre polynomials in Strain (1992), *etc.* Naturally, each of these schemes was limited to a narrow class of operators, though some of them were quite efficient.

Another early group of 'fast' techniques replaced the kernel-dependent special functions with some appropriately chosen bases: Chebyshev poly-

https://doi.org/10.1017/S0962492906410011 Published online by Cambridge University Press



Figure 2.1. A continuous charge distribution acting on a separated interval.

nomials in Alpert and Rokhlin (1991) and Rokhlin (1988), wavelets in Beylkin, Coifman and Rokhlin (1991), wavelet-like objects in Alpert, Beylkin, Coifman and Rokhlin (1993), *etc.* This approach is more general and easier to use, since a single scheme is applicable to a wider class of operators. Fast multipole and FFT-based variants have also been developed in a kernel-independent manner. They rely on sampling the governing Green's function in a systematic fashion and thereby building efficient representations of distant interactions (Phillips and White 1997, Ying, Biros, Zorin and Langston 2003, Gimbutas 1999).

Finally, it was observed that the preceding approaches, which build representations that are universally applicable and do not make use of the detailed distributions of sources, were suboptimal. That is, the numbers of basis elements used significantly exceed the ranks of the specific block matrices being approximated. Optimized low-rank approximations can be computed explicitly (via the SVD, QR, or the more recently introduced interpolative decompositions), and the resulting schemes can be noticeably faster than the original FMMs. Early work in this direction includes Hackbusch (1999) and Kapur and Long (1997).

It is instructive to examine this situation in some simple one-dimensional settings.

Example 1. Consider the integral operator defined by the formula

$$f(x) = P(\varphi)(x) = \int_{-1}^{1} \log(x - t)\, \varphi(t)\, \mathrm{d}t, \tag{2.1}$$

with $x \in [3, 5]$ (Figure 2.1). Using the language of classical potential theory, one could say that P is the operator mapping the charge distribution on the interval $[-1, 1]$ to the induced potential created on the interval $[3, 5]$. Decomposing the function $\log(x - t)$ into a Taylor series with respect to t around the point $t = 0$, we find that

$$f(x) = \left(\int_{-1}^{1} \varphi(t)\, \mathrm{d}t \right) \log(x) - \sum_{k=1}^{\infty} \frac{1}{k} \left(\int_{-1}^{1} t^k\, \varphi(t)\, \mathrm{d}t \right) \frac{1}{x^k}. \tag{2.2}$$

Truncating the series on the right-hand side of (2.2) after an appropriately chosen number of terms, we obtain an approximation to f with any desired precision. It is easily seen that the error of a k-term approximation is slightly better than $1/3^k$, so that the number of terms required to obtain a specified

https://doi.org/10.1017/S0962492906410011 Published online by Cambridge University Press

Table 2.1. Numbers of terms (Taylor, Legendre, and SVD) required to obtain several accuracies (in the L^2-norm) in Example 1.

ε	n_{Taylor}	n_{Legendre}	n_{SVD}
10^{-30}	61	39	18
10^{-24}	48	32	14
10^{-20}	40	26	12
10^{-16}	31	21	10
10^{-12}	23	16	7
10^{-10}	19	13	6
10^{-6}	10	8	4
10^{-4}	7	6	3

accuracy ε is of the order $-\log_3(\varepsilon)$. In the second column of Table 2.1 we list the numbers of terms actually required to obtain selected accuracies. We also note that (2.2) is a simplified version of the approximation used by most early versions of the FMM.

Obviously, there is nothing magical about Taylor series: many other approximations could be used. For example, decomposing the function φ into a Legendre series on the interval $[-1, 1]$ and performing elementary manipulations, we have

$$\begin{aligned} f(x) &= \int_{-1}^{1} \log(x-t) \left(\sum_{k=0}^{\infty} P_k(t) \int_{-1}^{1} P_k(\tau)\, \varphi(\tau)\, \mathrm{d}\tau \right) \\ &= \sum_{k=0}^{\infty} \left(\int_{-1}^{1} P_k(\tau)\, \varphi(\tau)\, \mathrm{d}\tau \right) \left(\int_{-1}^{1} P_k(t)\, \log(x-t)\, \mathrm{d}t \right) \\ &= 2 \sum_{k=0}^{\infty} \frac{1}{2k+1} \left(\int_{-1}^{1} P_k(\tau)\, \varphi(\tau)\, \mathrm{d}\tau \right) (Q_{k+1}(x) - Q_{k-1}(x)), \end{aligned} \tag{2.3}$$

where Q_k is the 'second' Legendre function (see, for example, Gradshteyn and Ryzhik (2000)). While the exact expression for the convergence rate of the series (2.3) is somewhat cumbersome (it involves standard estimates on the behaviour of Q_k), it is much higher than the convergence rate of (2.2). The numbers of terms required to obtain selected accuracies using this approach are listed in the third column of Table 2.1.

Finally, one can construct the Singular Value Decomposition (SVD) of the operator P (defined in (2.1) above), representing it in the form

$$f(x) = P(\varphi)(x) = \sum_{k=1}^{\infty} \lambda_k\, (v_k, \varphi)\, u_k(x), \tag{2.4}$$

https://doi.org/10.1017/S0962492906410011 Published online by Cambridge University Press

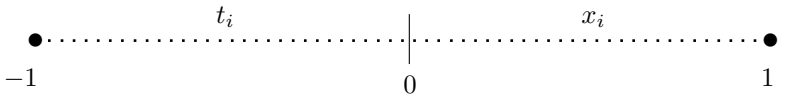


Figure 2.2. Discrete charge distribution acting on an adjacent interval.

where $u_k : [3,5] \to R$, $v_k : [-1,1] \to R$, are the kth left and right singular vectors, respectively, and λ_k is the kth singular value. Truncating the expansion (2.4) after k terms, we obtain a k-term approximation to P that is optimal in the obvious sense. For this case, the numbers of terms required to obtain selected accuracies are listed in the third column of Table 2.1.

Table 2.1 is quite revealing. It turns out that the approximation used by the original FMM is surprisingly inefficient, and the orthogonal expansion is only marginally better. For example, at 16 digits, the optimal number of terms is 10 (which is quite good); the Taylor and Legendre series require 31 and 21 terms respectively.

Example 2. Here, we consider the $n \times n$ matrix $\mathbf{A}_n$ defined by the formula

$$\mathbf{A}_n(i,j) = \log(x_i - t_j), \tag{2.5}$$

with the points $t_1, t_2, \ldots, t_n$, $x_1, x_2, \ldots, x_n \in R$ defined by the formulae

$$\begin{aligned} t_i &= -1 + \frac{(i-1)}{n}, \\ x_i &= \frac{(i-1)}{n}. \end{aligned} \tag{2.6}$$

In other words, the points $\{t_i\}$ are equispaced on the interval $[-1,0]$, and the points $\{x_i\}$ are equispaced on $[0,1]$. None of the classical expansions are applicable in this case (since the 'targets' in this case are not separated from the 'charges'). On the other hand, the following lemma shows that the rank of $\mathbf{A}_n$ defined in (2.5) is small compared to n. The fact that compression does not require the separation of sources and targets will be essential in constructing efficient direct solvers.

Lemma 2.1. To any fixed precision ε, the rank of the matrix $\mathbf{A}_n$ defined in (2.5) is of the order $-\log(n) \cdot \log_3(\varepsilon)$.

Outline of proof. We start by subdividing the interval $[0,1]$ into two subintervals of equal size: $[0,1/2]$ and $[1/2,1]$ (see Figure 2.3). Obviously, the interval $[1/2,1]$ is separated from the interval $[-1,0]$ by its own size, and the rank of their interactions is of the order $\log_3(\varepsilon)$ (see Example 1 above). We proceed by subdividing the interval $[0,1/2]$ into subintervals $[0,1/4]$ and $[1/4,1/2]$; again, the interval $[1/4,1/2]$ is separated from the interval $[-1,0]$ by its own size, and the rank of their interactions is bounded by $\log_3(\varepsilon)$. On the next step, we subdivide the interval $[0,1/4]$ into subintervals $[0,1/8]$,

https://doi.org/10.1017/S0962492906410011 Published online by Cambridge University Press



Figure 2.3. Recursive subdivision of an interval used in Lemma 1.1.

Table 2.2. Number of SVD terms for various accuracies in L^2 in Example 2.

ε	50	100	200	400	800	1600	3200	6400	12800	25600
10^{-30}	27	32	36	41	45	50	54	58	62	67
10^{-24}	23	26	30	34	37	40	44	47	51	54
10^{-20}	20	23	26	29	31	34	37	40	42	45
10^{-16}	17	19	21	23	26	28	30	32	34	36
10^{-12}	13	15	17	18	20	21	23	24	25	27
10^{-10}	11	13	14	15	17	18	19	20	21	22
10^{-6}	8	8	9	10	10	11	11	12	12	12
10^{-4}	5	6	6	6	7	7	7	7	7	7

$[1/8, 1/4]$, and observe that the rank of interactions between the intervals $[-1, 0]$, $[1/8, 1/4]$ is of the order $\log_3(\varepsilon)$. This process will terminate after roughly $\log_2(n)$ steps, and the total rank of interactions between the intervals $[-1, 0]$, $[0, 1]$ is bounded by $-\log(n) \cdot \log_3(\varepsilon)$. □

In Table 2.2, we list the ranks of the matrix $\mathbf{A}_n$ in (2.5) to accuracies varying from 10^{-4} to 10^{-30}, with n varying from 50 to 25600. The ranks were obtained via 'brute force' numerical calculation of the SVD, representing the matrix in the form

$$\mathbf{A}_n = \mathbf{U}_{n,k}\, \mathbf{D}_{k,k}\, (\mathbf{V}_{n,k})^* + O(\epsilon), \tag{2.7}$$

with the columns of each of the matrices $\mathbf{U}_{n,k}$, $\mathbf{V}_{n,k}$ orthonormal, $\mathbf{D}_{k,k}$ a diagonal matrix with positive elements, and k tabulated in Table 2.2 as a function of the L^2-norm of the error. Obviously, once the expansion (2.7) has been constructed, it can be used as a primitive 'fast' algorithm for the application of the matrix $\mathbf{A}_n$ to arbitrary vectors, reducing the cost from n^2 to $(2n + k)\, k$.

Example 3. To illustrate the situation in two dimensions, consider the integral operator $P : L^2(\Omega_1) \to L^2(\Omega_2)$ defined by the formula

$$f(x, y) = P(\sigma)(x, y) = \int_{\Omega_1} \log \|(x, y) - (u, v)\|\, \sigma(u, v)\, \mathrm{d}u\, \mathrm{d}v, \tag{2.8}$$

https://doi.org/10.1017/S0962492906410011 Published online by Cambridge University Press



Figure 2.4. Two squares in the complex plane.

Table 2.3. Numbers of terms (Multipole and SVD) required to obtain various accuracies in Example 3.

ε	$n_{\text{multipole}}$	n_{SVD}
10^{-30}	91	31
10^{-24}	73	25
10^{-20}	61	21
10^{-16}	48	17
10^{-12}	36	13
10^{-10}	30	11
10^{-6}	18	7
10^{-4}	12	5

with Ω_1 the square with vertices $(1,-1)$, $(1,1)$, $(-1,1)$, $(-1,-1)$, and Ω_2 the square with vertices $(5,-1)$, $(5,1)$, $(3,1)$, $(3,-1)$, shown in Figure 2.4.

This situation is standard in the design of FMMs in two dimensions, where P is compressed via the Taylor expansion

$$f(x,y) = \text{Re}\left[\left(\int_{\Omega_1} \sigma(u,v)\right)\log z + \sum_{k=1}^{\infty}\frac{1}{k}\left(\int_{\Omega_1}(u+\mathrm{i}v)^k\,\sigma(u,v)\right)\frac{1}{z^k}\right], \quad (2.9)$$

$z = x + \mathrm{i}y$ and $\text{Re}\,[\cdot]$ denotes the real part of the complex-valued quantity inside the brackets. An obvious alternative is to construct the SVD of P (viewed as an operator mapping $L^2(\Omega_1) \to L^2(\Omega_2)$). The resulting dimensionalities are tabulated in Table 2.3 as a function of the required accuracy ε, together with the numbers of terms in the multipole expansion (2.9) achieving similar accuracy.

Observation 2.1. An examination of Tables 2.1–2.3 indicates that when a 'fast' approximate algorithm is to be constructed, it might be much more efficient to utilize decompositions obtained numerically (via SVD, QR, or

https://doi.org/10.1017/S0962492906410011 Published online by Cambridge University Press

a related procedure) than to use one of the classical expansions. This is quite understandable, since the SVD constructs an *optimal* expansion for the operator in question, including detailed knowledge about the locations of all sources, while the classical approximations are *general* tools. The obvious cost of the purely numerical approach is the need to construct the representations, *i.e.*, the left and right singular vectors, *etc.* In many cases, such as when solving a fixed linear system with multiple right-hand sides, this cost is trivial: the matrices in question are factored once and used repeatedly (see, for example, Martinsson (2006)).

3. Fast direct solvers for structured matrices

The systematic use of techniques of the type described above will allow us to efficiently approximate the (dense) system matrices that arise from integral equation discretizations of surfaces in three dimensions. Before entering into those details, however, let us consider, from a purely linear-algebraic point of view, the solution of a dense linear system with low-rank off-diagonal blocks. For this, we let the linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$ be defined in block matrix form by the equations

$$\begin{aligned} \mathbf{A}_{11}\mathbf{x}_1 + \mathbf{A}_{12}\mathbf{x}_2 + \cdots + \mathbf{A}_{1N}\mathbf{x}_N &= \mathbf{b}_1, \\ \mathbf{A}_{21}\mathbf{x}_1 + \mathbf{A}_{22}\mathbf{x}_2 + \cdots + \mathbf{A}_{2N}\mathbf{x}_N &= \mathbf{b}_2, \\ &\vdots \\ \mathbf{A}_{N1}\mathbf{x}_1 + \mathbf{A}_{N2}\mathbf{x}_2 + \cdots + \mathbf{A}_{NN}\mathbf{x}_N &= \mathbf{b}_N, \end{aligned} \tag{3.1}$$

where $\mathbf{x}_i, \mathbf{b}_i \in \mathbb{R}^{n_i}$ and $\mathbf{A}_{ij} \in \mathbb{R}^{n_i \times n_j}$. Solution of the full linear system of dimension $N_{\text{tot}} = \sum_{i=1}^{N} n_i$ requires $O(N_{\text{tot}}^3)$ work.

Definition 3.1. When needed, the entry in the lth row and mth column of $\mathbf{A}_{ij}$ will be denoted by $\mathbf{A}_{ij}(l, m)$. Likewise, the entry in the lth row and mth column of $\mathbf{A}$ itself will be referred to as $\mathbf{A}(l, m)$.

Suppose now that the diagonal blocks $\mathbf{A}_{ii}$ are full-rank but that each of the $\mathbf{A}_{ij}$ can be decomposed as the product of three low-rank matrices:

$$\mathbf{A}_{ij} = \mathbf{L}_i \mathbf{S}_{ij} \mathbf{R}_j, \tag{3.2}$$

where $\mathbf{L}_i \in \mathbb{R}^{n_i \times k_i}$, $\mathbf{S}_{ij} \in \mathbb{R}^{k_i \times k_j}$, and $\mathbf{R}_j \in \mathbb{R}^{k_j \times n_j}$, with $k_i \ll n_i$ and $k_j \ll n_j$. It is worth noting that the factorization (3.2) is somewhat special in that $\mathbf{L}_i$ depends only on the index i and $\mathbf{R}_j$ depends only on the index j. We will see how such a factorization arises in the next section, but for now we assume it is given.

Under these conditions, we can introduce the auxiliary variables

$$\mathbf{y}_j = \mathbf{R}_j \mathbf{x}_j \tag{3.3}$$

https://doi.org/10.1017/S0962492906410011 Published online by Cambridge University Press

and rewrite the original linear system (3.1) in the form

$$\begin{aligned}
\mathbf{A}_{11}\mathbf{x}_1 + \mathbf{L}_1\mathbf{S}_{12}\mathbf{y}_2 + \cdots + \mathbf{L}_1\mathbf{S}_{1N}\mathbf{y}_N &= \mathbf{b}_1,\\
\mathbf{L}_2\mathbf{S}_{21}\mathbf{y}_1 + \mathbf{A}_{22}\mathbf{x}_2 + \cdots + \mathbf{L}_2\mathbf{S}_{2N}\mathbf{y}_N &= \mathbf{b}_2,\\
&\vdots\\
\mathbf{L}_N\mathbf{S}_{N1}\mathbf{y}_1 + \mathbf{L}_N\mathbf{S}_{N2}\mathbf{y}_2 + \cdots + \mathbf{A}_{NN}\mathbf{x}_N &= \mathbf{b}_N.
\end{aligned} \tag{3.4}$$

The coupled linear system (3.3), (3.4), can be written in block matrix form:

$$\left(\begin{array}{cccc|cccc}
\mathbf{A}_{11} & 0 & \cdots & 0 & 0 & \mathbf{L}_1\mathbf{S}_{12} & \cdots & \mathbf{L}_1\mathbf{S}_{1N}\\
0 & \mathbf{A}_{22} & \cdots & 0 & \mathbf{L}_2\mathbf{S}_{21} & 0 & \cdots & \mathbf{L}_2\mathbf{S}_{2N}\\
0 & 0 & \cdots & 0 & \cdots & \cdots & \cdots & \cdots\\
0 & 0 & \cdots & \mathbf{A}_{NN} & \mathbf{L}_N\mathbf{S}_{N1} & \mathbf{L}_N\mathbf{S}_{N2} & \cdots & 0\\
\hline
\mathbf{R}_1 & 0 & \cdots & 0 & -\mathbf{I}_1 & 0 & \cdots & 0\\
0 & \mathbf{R}_2 & \cdots & 0 & 0 & -\mathbf{I}_2 & \cdots & 0\\
0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0\\
0 & 0 & \cdots & \mathbf{R}_N & 0 & 0 & \cdots & -\mathbf{I}_N
\end{array}\right)
\left(\begin{array}{c}
\mathbf{x}_1\\ \mathbf{x}_2\\ \cdots\\ \mathbf{x}_N\\ \hline \mathbf{y}_1\\ \mathbf{y}_2\\ \cdots\\ \mathbf{y}_N
\end{array}\right)
=
\left(\begin{array}{c}
\mathbf{b}_1\\ \mathbf{b}_2\\ \cdots\\ \mathbf{b}_N\\ \hline 0\\ 0\\ \cdots\\ 0
\end{array}\right), \tag{3.5}$$

where $\mathbf{I}_j$ is the identity matrix of dimension n_j.

In (3.5), we have replaced the dense linear system of (3.1) with N_{tot} unknowns by a sparse system of dimension $N_{\text{tot}} + K_{\text{tot}}$, where $K_{\text{tot}} = \sum_{i=1}^{N} k_i$. The sparsity, of course, depends strongly on how few auxiliary variables (K_{tot}) are needed. Note that the zero structure of the new system allows us to form the Schur complement in the $\{\mathbf{y}_j\}$ variables quite easily. Elementary row elimination leads from (3.5) to

$$\begin{pmatrix}
\mathbf{E}_{11} & \mathbf{S}_{12} & \cdots & \mathbf{S}_{1N}\\
\mathbf{S}_{21} & \mathbf{E}_{22} & \cdots & \mathbf{S}_{2N}\\
\cdots & \cdots & \cdots & \cdots\\
\mathbf{S}_{N1} & \mathbf{S}_{N2} & \cdots & \mathbf{E}_{NN}
\end{pmatrix}
\begin{pmatrix}
\mathbf{y}_1\\ \mathbf{y}_2\\ \cdots\\ \mathbf{y}_N
\end{pmatrix}
=
\begin{pmatrix}
\mathbf{C}_1\mathbf{b}_1\\ \mathbf{C}_2\mathbf{b}_2\\ \cdots\\ \mathbf{C}_N\mathbf{b}_N
\end{pmatrix}, \tag{3.6}$$

where

$$\mathbf{E}_{ii} = (\mathbf{R}_i\mathbf{A}_{ii}^{-1}\mathbf{L}_i)^{-1}, \mathbf{C}_i = \mathbf{E}_i\mathbf{R}_i\mathbf{A}_{ii}^{-1}. \tag{3.7}$$

Combining (3.4), (3.6) and a little algebra, we can write

$$\mathbf{A}^{-1} = \mathbf{D} + \mathbf{B}(\mathbf{E} + \mathbf{S})^{-1}\mathbf{C}, \tag{3.8}$$

where $\mathbf{S}$ is the dense matrix with off-diagonal blocks $\mathbf{S}_{ij}$ given by (3.2), $\mathbf{S}_{ii} \equiv 0$, $\mathbf{E}, \mathbf{C}$ are the block-diagonal matrices with diagonal entries $\mathbf{E}_{ii}, \mathbf{C}_i$, respectively, and $\mathbf{D}, \mathbf{B}$ are block-diagonal matrices with

$$\begin{aligned}
\mathbf{D}_{ii} &= \mathbf{A}_{ii}^{-1}(\mathbf{I}_i - \mathbf{L}_i\mathbf{C}_i),\\
\mathbf{B}_{ii} &= \mathbf{A}_{ii}^{-1}\mathbf{L}_i\mathbf{E}_i.
\end{aligned} \tag{3.9}$$

https://doi.org/10.1017/S0962492906410011 Published online by Cambridge University Press

Note that $\mathbf{E} + \mathbf{S}$ is the Schur complement system in (3.6).

Finally, it is worth computing the cost of this procedure. For concreteness, suppose that $n_1 = n_2 = \cdots = n_N = m$, so that $N_{\text{tot}} = Nm$ and naive inversion would require $O(N^3m^3)$ work. Suppose also that $k_1 = k_2 = \cdots = k_N = k$. In the new procedure, the initial computation of all the A_{ii}^{-1} then requires $O(Nm^3)$ work. Since the Schur complement system is of dimension Nk, inversion requires $O(N^3k^3)$ work. The remaining work is dominated by the computation of the $\{\mathbf{x}_j\}$ once the $\{\mathbf{y}_j\}$ are known using (3.4) or (3.8). We leave it to the reader to verify that this requires $O(N^2k^2 + Nmk^2 + Nm^2)$ work.

In the original paper (Martinsson and Rokhlin 2005), the solver is based directly on (3.8) and (3.9). The explicit introduction of the Schur complement makes the derivation a bit simpler. More importantly, however, the embedding of the system into a larger, sparse system allows for the immediate application of standard sparse matrix technology.

Remark 3.1. The algorithm described above is a 'single-level' scheme. That is, unknowns are assumed to have been grouped in some fashion so that off-diagonal blocks are of minimal rank. This idea can be used recursively. In multilevel versions of the method, the Schur complement ($\mathbf{E}$+$\mathbf{S}$) is treated like the original matrix $\mathbf{A}$. More precisely, the $\mathbf{y}_j$ unknowns are themselves grouped in such a way that off-diagonal blocks of the $\mathbf{E} + \mathbf{S}$ system are of minimal rank, *etc.*

It is the multilevel variants of the preceding analysis that form the basis for many of the fast, direct solvers currently under development. In the end, the method of choice will depend to a large extent on the constants implicit in the $O(N^\alpha \log^\beta N)$ notation, on robustness, and on ease of use. A multilevel scheme in the one-dimensional setting is presented in the appendix to illustrate the recursive nature of the computation.

4. Potential theory for the Poisson equation

We shift our attention now to the classical integral equation approach for solving the Poisson equation in the context of molecular electrostatics. We begin with the formulation of the problem as a partial differential equation:

$$-\nabla \cdot (\epsilon(\mathbf{x})\nabla\Phi(\mathbf{x})) = \sum_{j=1}^{K} q_j \delta(\mathbf{x} - \mathbf{x}_j) \quad \text{in } \mathbb{R}^3, \tag{4.1}$$

where $\epsilon(\mathbf{x})$ equals ϵ_{in} within the molecular surface, and $\epsilon(\mathbf{x})$ equals ϵ_{out} outside. We will denote the interior region by Ω_{in}, the exterior region by Ω_{out}, and the surface by S (Figure 4.1). We assume the sources all lie in the interior of the molecule Ω_{in}. In order to recast the problem as an integral

https://doi.org/10.1017/S0962492906410011 Published online by Cambridge University Press

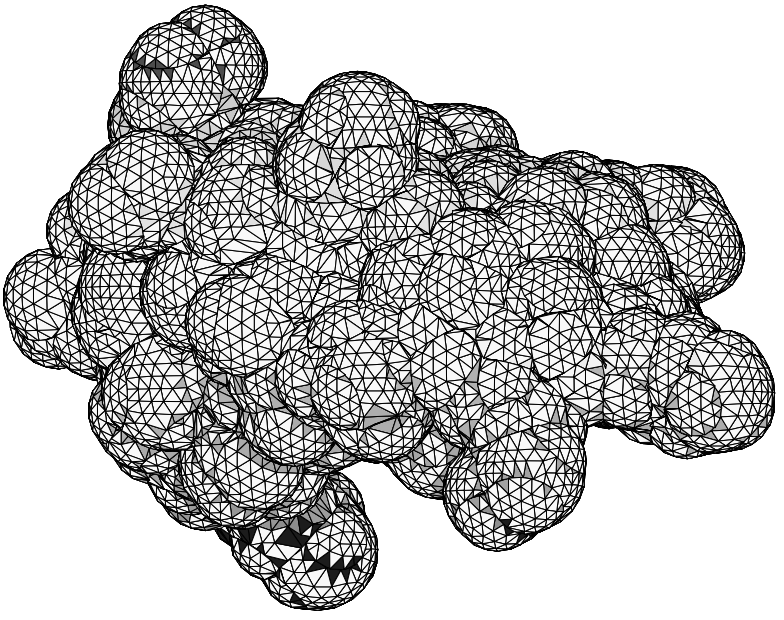


Figure 4.1. A typical molecular surface.

equation, we define the solution in terms of a source function Φ^{source} that accounts for the singular fields due to the point sources and a piecewise harmonic function Φ^{pol}. That is, we write

$$\Phi = \Phi^{\text{source}} + \Phi^{\text{pol}},$$

where

$$\Phi^{\text{source}} = \sum_{j=1}^{K} \frac{q_j}{4\pi\epsilon_{\text{in}}} \frac{1}{\|\mathbf{x} - \mathbf{x}_j\|}.$$

Remark 4.1. A more important model for biophysical applications is actually the linearized Poisson–Boltzmann equation (Davis and McCammon 1990, Rocchia *et al.* 2002, Sharp and Honig 1990), but it is easier to follow the main ideas in the present, simpler context.

Note that Φ^{source} satisfies the Poisson equation and is continuous across the surface S. However, the required interface conditions are that both the total potential Φ and its flux $\epsilon \frac{\partial \Phi}{\partial \nu}$ must be continuous across the dielectric

https://doi.org/10.1017/S0962492906410011 Published online by Cambridge University Press

boundary. For this, it is straightforward to see that Φ^{pol} must satisfy

$$\nabla^2 \Phi^{\text{pol}} = 0 \quad \text{in } \Omega_{\text{in}},\ \Omega_{\text{out}}, \tag{4.2}$$

$$\left[\Phi^{\text{pol}}\right] = 0 \quad \text{on } S, \tag{4.3}$$

$$\left[\epsilon \frac{\partial \Phi^{\text{pol}}}{\partial \nu}\right] = -\left[\epsilon \frac{\partial \Phi^{\text{source}}}{\partial \nu}\right] \quad \text{on } S. \tag{4.4}$$

Here, the expression $[f]$ denotes the jump in the quantity f across S.

Following standard practice, we write Φ^{pol} as a single-layer potential due to a charge distribution σ defined on the surface:

$$\Phi^{\text{pol}}(\mathbf{x}) = \frac{1}{4\pi} \int_S \frac{\sigma(\mathbf{y})}{\|\mathbf{x}-\mathbf{y}\|}\, d\mathbf{y}. \tag{4.5}$$

The single-layer potential (4.5) is continuous across S and satisfies the following jump relations (Guenther and Lee 1988):

$$\frac{\partial \Phi^{\text{pol}}}{\partial \nu_-}(\mathbf{y}_0) \equiv \lim_{\substack{\mathbf{x}\to\mathbf{y}_0 \\ \mathbf{x}\in\Omega_{\text{in}}}} \frac{\partial \Phi^{\text{pol}}}{\partial \nu_0}(\mathbf{x}) = \tfrac{1}{2}\sigma(\mathbf{y}_0) + \int_S \frac{\partial G}{\partial \nu_0}(\mathbf{y}_0,\mathbf{y})\sigma(\mathbf{y})\, d\mathbf{y}, \tag{4.6}$$

$$\frac{\partial \Phi^{\text{pol}}}{\partial \nu_+}(\mathbf{y}_0) \equiv \lim_{\substack{\mathbf{x}\to\mathbf{y}_0 \\ \mathbf{x}\in\Omega_{\text{out}}}} \frac{\partial \Phi^{\text{pol}}}{\partial \nu_0}(\mathbf{x}) = -\tfrac{1}{2}\sigma(\mathbf{y}) + \int_S \frac{\partial G}{\partial \nu_0}(\mathbf{y}_0,\mathbf{y})\sigma(\mathbf{y})\, d\mathbf{y}, \tag{4.7}$$

where $\mathbf{y}_0$ denotes a point on S, $\frac{\partial \Phi^{\text{pol}}}{\partial \nu_0}(\mathbf{x})$ denotes the derivative in the direction of the outward normal at $\mathbf{y}_0$, and $G(\mathbf{x},\mathbf{y})$ denotes the free-space Green's function $\frac{1}{4\pi|\mathbf{x}-\mathbf{y}|}$.

We will make use of the operator notation

$$\mathbf{K}\sigma(\mathbf{y}_0) = \int_S \frac{\partial G}{\partial \nu_0}(\mathbf{y}_0,\mathbf{y})\sigma(\mathbf{y})\, d\mathbf{y} \tag{4.8}$$

to denote the normal derivative of the single-layer potential restricted to the interface.

It is clear from the representation (4.5) that equation (4.2) is automatically satisfied. Since the single-layer potential is continuous across the interface, (4.3) is also satisfied. Imposing the remaining condition (4.4) and using the jump relations, we obtain a Fredholm integral equation of the second kind for σ:

$$\frac{1}{2}\sigma(\mathbf{y}_0) + \lambda \int_S \frac{\partial G}{\partial \nu_0}(\mathbf{y}_0,\mathbf{y})\sigma(\mathbf{y})\, d\mathbf{y} = -\lambda \frac{\partial \Phi^{\text{source}}}{\partial \nu}(\mathbf{y}_0), \tag{4.9}$$

where $\lambda = (\epsilon_{\text{in}} - \epsilon_{\text{out}})/(\epsilon_{\text{in}} + \epsilon_{\text{out}})$. In operator notation,

$$\frac{1}{2}\sigma + \lambda \mathbf{K}\sigma = -\lambda \frac{\partial \Phi^{\text{source}}}{\partial \nu}.$$

It remains only to discretize and solve (4.9). For this, we assume that S has been specified as a collection of N_T triangles. We approximate the

https://doi.org/10.1017/S0962492906410011 Published online by Cambridge University Press

potential due to the polarization charge as

$$\Phi^{\text{pol}}(\mathbf{x}) = \frac{1}{4\pi}\sum_{j=1}^{N_T}\int_{T_j}\frac{\sigma_j}{\|\mathbf{x}-\mathbf{y}\|}\,\mathrm{d}\mathbf{y}, \tag{4.10}$$

where T_j denotes the jth triangle and σ_j is constant over T_j. Imposing the continuity of flux at each triangle centroid C_j results in the finite-dimensional linear system

$$\frac{1}{2}\sigma_j + \frac{\lambda}{4\pi}\sum_{k=1}^{N_T}\int_{T_k}\frac{\partial G}{\partial \nu_0}(C_j,\mathbf{y})\sigma_k\,\mathrm{d}\mathbf{y} = -\lambda\frac{\partial \Phi^{\text{source}}}{\partial \nu}(C_j). \tag{4.11}$$

This approach is generally referred to as a first-order accurate collocation scheme. We do not recommend such a quadrature rule in general, but higher-order accurate discretizations are rather involved and will distract us from our goals. For recent work on higher-order schemes in the context of molecular modelling, see Bardhan *et al.* (2005).

When $j = k$, the integral contribution in (4.11) is well known to vanish. That is, on a flat triangle,

$$\int_{T_j}\frac{\partial G}{\partial \nu_0}(C_j,\mathbf{y})\sigma_j\,\mathrm{d}\mathbf{y} = 0,$$

and the diagonal matrix entries $\mathbf{A}(i,i)$ of the linear system are simply equal to $\frac{1}{2}$. When j and k are distinct, the entry $\mathbf{A}(j,k)$ is given by

$$\mathbf{A}(j,k) = \frac{\lambda}{4\pi}\int_{T_k}\frac{\partial G}{\partial \nu_0}(C_j,\mathbf{y})\,\mathrm{d}\mathbf{y}, \tag{4.12}$$

namely, λ times the flux through the centroid of T_j due to a unit charge distribution on T_k. It is well defined, although nearly singular for nearby interactions. $\mathbf{A}(j,k)$ can easily be computed by a hybrid numerical/analytic scheme: the triangles T_k and T_j are rotated in space so that T_k lies in the xy-plane with one vertex at the origin and one segment lying along the x-axis. Integration over T_k in the x-direction is performed analytically and integration in the y-direction is performed numerically using Gaussian quadrature. Twenty Gaussian nodes yield more than six digits of accuracy.

Of particular interest is the evaluation of the total electrostatic energy

$$\frac{1}{2}\sum_{i=1}^{K} q_i\tilde{\Phi}(\mathbf{x}_i), \tag{4.13}$$

where

$$\tilde{\Phi}(\mathbf{x}_i) = \Phi^{\text{pol}}(\mathbf{x}_i) + \sum_{\substack{j=1\\ j\neq i}}^{K}\frac{q_j}{4\pi\epsilon_{\text{in}}}\frac{1}{\|\mathbf{x}_i-\mathbf{x}_j\|}.$$

https://doi.org/10.1017/S0962492906410011 Published online by Cambridge University Press

Remark 4.2. We do not intend to survey the literature on numerical methods for the Poisson or Poisson–Boltzmann equation here, except to note that the standard approach in molecular electrostatics has been based on the finite difference or finite element solution of the governing equations (Davis and McCammon 1990, Holst, Baker and Wang 2000, Rocchia *et al.* 2002, Sharp and Honig 1990). There has also been a great deal of work on integral equation methods, particularly since the advent of fast algorithms for computing the matrix vector product in the solution of linear systems like (4.11). In the Poisson–Boltzmann case, these include Altman, Bardhan, Tidor and White (2006), Boschitsch, Fenley and Olson (1999), Huang and Greengard (2002), Kuo *et al.* (2002), Lu, Cheng, Huang and McCammon (2006) and Liang and Subramaniam (1997).

5. A single-level fast solver

We are now in a position to bring the linear algebra of Section 3 together with the potential theory of Section 4. The essential step is the grouping of unknowns in the integral equation so that off-diagonal blocks can be well-approximated by low-rank matrices. In the one-dimensional case, the ordering of unknowns on the interval is sufficient, but in higher dimensions it is not. We begin by sorting all the triangles $\{T_j\}$ that define the surface S in (4.11) into cubic boxes. For this, we first enclose the surface S in a cube B, and subdivide B into $M \times M \times M$ boxes. Each of the N_T triangles is identified with the box containing its centroid. After this sorting step, we let N denote the number of boxes that are non-empty (denoted by $B_1, B_2, \ldots, B_N$). We then reorder the original triangles so that the first n_1 belong to B_1, the next n_2 belong to B_2, *etc.* The triangles in box B_i will be referred to as *surface patch* (or *patch*) P_i.

Definition 5.1. Two disjoint boxes B_j and B_k are called *neighbours* if they share a boundary point. Otherwise, they are called *well-separated*. Two surface patches P_j and P_k are called neighbours or well-separated in accordance with the relation between the boxes B_j and B_k.

Suppose now that box B_i and box B_j are well-separated and that $\mathbf{A}_{ij}$ is the block matrix describing the flux through 'target' triangles in B_i due to charge distributions on 'source' triangles in B_j. From standard multipole estimates (Greengard and Rokhlin 1997), it is clear that, for any fixed precision, the rank of $\mathbf{A}_{ij}$ is bounded independent of the number of triangles n_i and n_j. To be precise, for $\mathbf{x} \in B_i$,

$$\Phi(\mathbf{x}) = \sum_{l=0}^{p} \sum_{m=-l}^{l} M_l^m \frac{Y_l^m(\theta, \phi)}{r^{l+1}} + O\left(\left(\frac{1}{\sqrt{3}}\right)^p\right),$$

https://doi.org/10.1017/S0962492906410011 Published online by Cambridge University Press

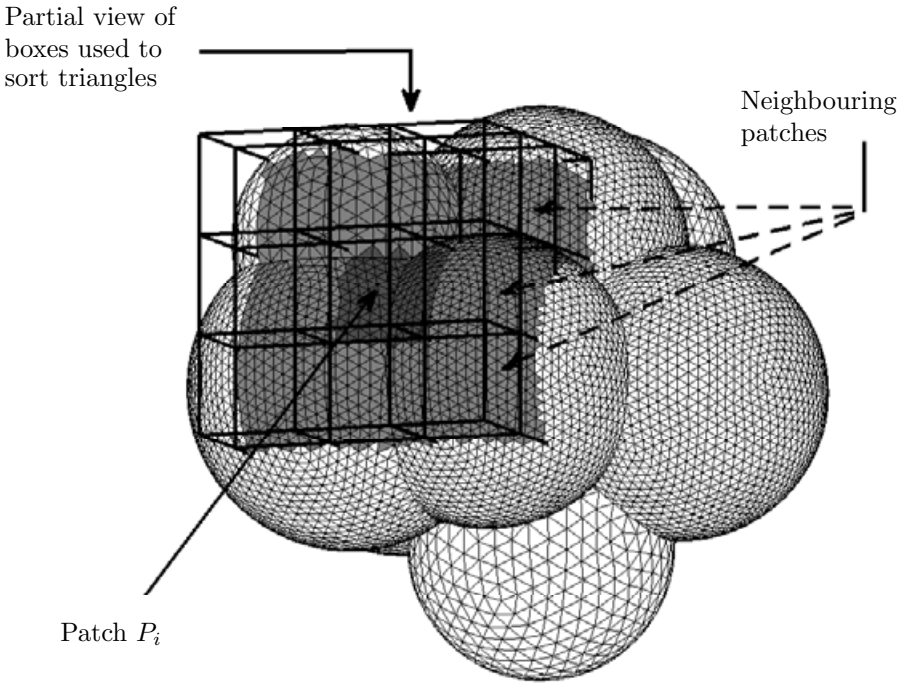


Figure 5.1. A grid of $M \times M \times M$ boxes encloses the surface S. The triangles whose centroids are located in subcube B_i are identified as surface patch P_i (dark grey). Its near-neighbour patches are indicated in light grey. The remaining patches are well-separated.

where (r, θ, ϕ) are the spherical coordinates of $\mathbf{x}$ with respect to the centre of box B_j, $Y_l^m(\theta, \phi)$ denotes the spherical harmonic of order l and degree m, and the $\{M_l^m\}$ are the net multipole moments of the charge distributions on the 'source' triangles in B_j. For a precision ϵ, then, one requires no more than $k = O((\log_{\sqrt{3}} \epsilon)^2)$ degrees of freedom to represent $\Phi(\mathbf{x})$. In short, the field induced in B_i has guaranteed smoothness properties due entirely to the separation between sources and targets and, to precision ϵ, the rank of $\mathbf{A}_{ij}$ is bounded by $(\log_{\sqrt{3}} \epsilon)^2$.

Rather than using multipole expansions, however, one could ask whether there is a purely linear-algebraic construct that expresses this fact. In particular, one could ask whether a subset of k_j source triangles could be used to represent the field in B_i with charge strengths computed from the n_j values σ_j. Similarly, one could ask whether a subset of k_i triangles in B_i could be used to sufficiently sample the field to precision ϵ, in the sense that the field at all n_i triangles in B_i can be computed by some kind of interpolation procedure. This is not a classical factorization like the SVD

https://doi.org/10.1017/S0962492906410011 Published online by Cambridge University Press

used in Section 2, but will be more suitable for our purposes. There are some remarkable results in this direction, for instance, Cheng, Gimbutas, Martinsson and Rokhlin (2005), Goreinov, Yrtyshnikov and Zamarashkin (1997) and Gu and Eisenstat (1996).

Theorem 5.1. Suppose that the matrix $\mathbf{A}_{ij}$ of dimension $n_i \times n_j$ has rank k to precision ϵ. Then there exists a factorization of $\mathbf{A}_{ij}$ of the form

$$\mathbf{A}_{ij} = \mathbf{L}_i \mathbf{S}_{ij} \mathbf{R}_j + O(\epsilon),$$

where $\mathbf{S}_{ij}$ is a $k \times k$ *submatrix* of $\mathbf{A}_{ij}$, $\mathbf{R}_j$ is a $k \times n_j$ matrix and $\mathbf{L}_i$ is an $n_i \times k$ matrix. Further, this decomposition is well-conditioned; the norms of $\mathbf{L}_i$ and $\mathbf{R}_j$ are bounded by $\sqrt{k(n_1 + n_2)}$.

Proof. This is simply a restatement of Theorem 3 of Cheng *et al.* (2005). □

The preceding theorem is not constructive. It does not say which k row indices and which k column indices to select in the extraction of the submatrix $\mathbf{S}_{ij}$ from $\mathbf{A}_{ij}$. Fortunately, Cheng *et al.* (2005) describe a *QR*-like algorithm which computes such a factorization using $O(n_1\, n_2\, k)$ work. (It should be noted that this factorization is not unique, but that is of no concern to us here.) We will refer to the k triangles in B_i so obtained as the *incoming skeleton* and to the k triangles in B_j as the *outgoing skeleton* (Cheng *et al.* 2005, Goreinov *et al.* 1997). Note that this is precisely what we need. $\mathbf{R}_j$ serves as the mapping from the original vector of n_j charges to the k charges on the outgoing skeleton and $\mathbf{L}_i$ serves as the mapping from the field sampled on the k incoming triangles to the full set of n_i triangles in subcube B_i. We view the skeletonization procedures as *black-box* routines and refer the reader to the original papers for further explanation.

Unfortunately, we are not yet done, since the well-separatedness criterion does not apply to block matrices $\mathbf{A}_{ij}$ when B_i and B_j are neighbours. In this context, simple *a priori* estimates do not apply. Nevertheless, Theorem 5.1 and the construction of the skeletons still apply – it is just that the rank k needs to be determined on the fly, given the desired precision ϵ. (This was a principal motivation of the earlier paper, Gu and Eisenstat (1996).) Recall that we considered precisely such a case in one dimension in Example 2 of Section 2.

To see why the rank of interactions between neighbouring boxes might be low, despite the absence of strict separation, readers familiar with potential theory will recall that on a flat surface, the operator $\mathbf{K}$ in (4.8) is identically zero. It is perhaps reasonable, then, to expect that the block matrix of interactions of two surface patches is in general not full-rank, even at high precision.

https://doi.org/10.1017/S0962492906410011 Published online by Cambridge University Press

5.1. Global skeletonization

The reader may have noticed that a strong form of 'skeletonization' is required in the set-up of the system (3.4). In particular, for each block of unknowns corresponding to patch P_i, $i = 1, \ldots, N$, the same number of triangles k_i is used for both the incoming and outgoing skeletons. One way to accomplish this for patch P_i is to insist that the incoming and outgoing skeletons actually be the same. For the computation, we concatenate all of the $\mathbf{A}_{ij}$ submatrices followed by all of the $\mathbf{A}_{ji}^T$ submatrices, excluding the diagonal blocks:

$$\mathbf{A}_i \equiv \left[\mathbf{A}_{i1}\mathbf{A}_{i2}\cdots\mathbf{A}_{i,i-1}\mathbf{A}_{i,i+1}\cdots\mathbf{A}_{iN}\,\mathbf{A}_{1i}^T\mathbf{A}_{2i}^T\cdots\mathbf{A}_{i-1,i}^T\mathbf{A}_{i+1,i}^T\cdots\mathbf{A}_{Ni}^T\right]. \quad (5.1)$$

The $n_i \times 2(N_{\text{tot}} - n_i)$ matrix $\mathbf{A}_i$ so-formed can then be passed to the skeletonizing machinery of Cheng *et al.* (2005). The number of row indices k_i obtained through skeletonization is not known *a priori*, but returned as part of the calculation.

Lemma 5.2. (Martinsson and Rokhlin 2005) Let $\mathbf{A}_i$ denote the $n_i \times 2(N_{\text{tot}} - n_i)$ matrix defined in (5.1) and let the interpolatory decomposition for $\mathbf{A}_i$ be given by

$$\mathbf{A}_i = \mathbf{L}_i\mathbf{S}_i\mathbf{V}_i + O(\epsilon), \quad (5.2)$$

for a specified precision ϵ, with $\mathbf{L}_i \in \mathbb{R}^{n_i \times k_i}$, $\mathbf{S}_i \in \mathbb{R}^{k_i \times k_i}$ a *submatrix* of $\mathbf{A}_{ij}$, and $\mathbf{V}_i \in \mathbb{R}^{k_i \times 2(N_{\text{tot}} - n_i)}$. Then, the triangles corresponding to the k_i row indices can serve as both the incoming and outgoing skeletons. Moreover, in the notation of Theorem 5.1,

$$\mathbf{A}_{ij} = \mathbf{L}_i\mathbf{S}_{ij}\mathbf{L}_j^T + O(\epsilon).$$

Sketch of proof. Note that skeletonization of the first $N - 1$ blocks of A_i guarantees that the k_i row indices which have been returned serve as a satisfactory incoming skeleton. In other words, the incoming field is sufficiently sampled to the specified precision ϵ at those k_i triangle centroids, and L_i is the mapping from those values to the values on all n_i triangles in patch P_i. The remainder of the theorem follows from considering the transpose of (5.2). That is, simultaneous skeletonization of the second $N - 1$ block matrices guarantees that the k_i indices which have been returned also serve as a satisfactory outgoing skeleton. Further, $\mathbf{L}_i^T$ must serve as the mapping from the given charges on all n_i triangles in patch P_i to effective charges on the skeleton triangles that correctly represent the field on all other patches to precision ϵ. □

As noted in Martinsson and Rokhlin (2005), for the problems of potential theory, one can reduce the cost of this step dramatically. One need not explore all the pairwise interactions between the triangles in patch P_i and those which are well-separated, since any potential field induced inside B_i

https://doi.org/10.1017/S0962492906410011 Published online by Cambridge University Press

by well-separated triangles could equally well be induced by a charge distribution located on the interface separating B_i from its well-separated boxes. That interface is simply the outer boundary of the $3 \times 3 \times 3$ 'supercell' of boxes centred on B_i (the set of boxes shown in Figure 5.1). In other words, an artificial set of triangles on the surface of the supercell can replace all of the A_{ij} blocks in the construction (5.1) corresponding to well-separated patches P_j. The number of triangles needed is determined only by the desired precision ϵ. Likewise, if a subset of triangles inside B_i matches the field on the surface of the supercell to precision ϵ, it will also do so in any well-separated box. Thus, the same artificial set of triangles on the surface of the supercell can replace all of blocks A_{ij}^T corresponding to well-separated patches P_j in the construction of (5.1) as well. Computationally, this means that the dimension of the matrix which must be skeletonized for patch P_i is of the order $n_i \times 2(N_i + M)$, where N_i denotes the number of triangles in the neighbours of B_i only and M denotes the number of triangles on the supercell surface.

6. Numerical results I

We have implemented the one-level direct solver described in the preceding sections, which takes as input an arbitrary triangulated surface and uses the simple piecewise constant approximation of the surface charge density to solve the dielectric interface problem

As an example, we selected atomic coordinates of the protein BPTI from the Protein Data Bank (PDB) and used the STLib package to create a surface triangulation (depicted in Figure 4.1). 20600 triangles were created in the original discretization, and compression with a tolerance of 10^{-2} yielded 7049 'skeleton' triangles (the dimension of the Schur complement in (3.6)). Factorization (in this naive one-level approach) required about 20 minutes, but the subsequent application of A^{-1} required about 0.3 seconds on a single-processor 1.9 GHz workstation. A useful point of comparison is that an FMM-based iterative solution would have required several seconds: much shorter than the factorization time but more than an order of magnitude slower than the application of the compressed inverse. Compression with a tolerance of 10^{-1} yielded 1596 'skeleton' triangles, factorization required about 10 minutes, and the application of A^{-1} required about 0.1 seconds.

The power of the direct solver, of course, is made much more apparent when solving the integral equation repeatedly. In molecular electrostatics, for example, one is often interested in the electrostatic energy as the interior charge distribution is modified. Each new set of charge locations induces a different right-hand side in (4.9). The total cost (after the initial factorization) for M interior charge distributions (solving M PDEs) is then $M \times 0.1$ seconds.

https://doi.org/10.1017/S0962492906410011 Published online by Cambridge University Press

7. Local geometric perturbations

As implied in the Introduction, one of the striking areas of application of fast direct solvers lies in their ability to accelerate the solution of 'nearby' problems. By nearby, we mean that the new system matrix is a low-rank modification of the one for which we already have an efficient factorization. In linear algebra, this idea is generally attributed to Sherman, Morrison and Woodbury (1949, 1950). There is a host of applications of this idea. In the context of integral equations, it has been used successfully, for example, in both electromagnetics (Kastner 1989) and elasticity (James and Pai 1999).

One of our goals is to allow for electrostatic modelling in protein design using the one- or two-body decomposable method of Marshall, Vizcarra and Mayo (2005). This method requires the solution of many electrostatic problems with a fixed protein backbone, with local modifications of only one or two amino acid side chains. Each such modification requires the deletion and insertion of a modest number of triangles in the definition of the molecular surface. Since we will rely heavily on the Sherman–Morrison–Woodbury approach, we briefly summarize it here in the context of boundary integral equations.

7.1. Addition of triangles

Suppose that the matrix $\mathbf{A}$ of system (4.11) has been constructed for a surface with N triangles, but that we now wish to solve the same interface problem with an extra collection of p triangles *added*. We will denote the added triangles by $T_{N+1}, \ldots, T_{N+p}$. (For the moment, we can assume that we are adding a second surface component.) Then, the new linear system that describes the perturbed problem takes the form

$$\begin{pmatrix} \mathbf{A} & \mathbf{B}_+ \\ \mathbf{C}_+ & \mathbf{D}_+ \end{pmatrix} \begin{pmatrix} \boldsymbol{\sigma} \\ \boldsymbol{\sigma}_p \end{pmatrix} = \begin{pmatrix} \mathbf{f} \\ \mathbf{f}_p \end{pmatrix}. \tag{7.1}$$

Here, $\mathbf{B}_+ \in \mathbb{R}^{N\times p}$, $\mathbf{C}_+ \in \mathbb{R}^{p\times N}$, and $\mathbf{D}_+ \in \mathbb{R}^{p\times p}$ with

$$\begin{aligned} \mathbf{B}_+(i,j) &= A(T_i, T_{N+j}), \\ \mathbf{C}_+(i,j) &= A(T_{N+j}, T_i), \\ \mathbf{D}_+(i,j) &= \begin{cases} \frac{1}{2} & \text{if } i = j, \\ A(T_{N+i}, T_{N+j}) & \text{if } i \neq j, \end{cases} \\ \mathbf{f}_p(j) &= -\lambda \frac{\partial \Phi^{\text{source}}}{\partial \nu}(C_{N+j}), \end{aligned}$$

where C_{N+j} is the centroid of the jth added triangle, and $A(T_i, T_j)$ is defined in (4.12).

https://doi.org/10.1017/S0962492906410011 Published online by Cambridge University Press

Block Gaussian elimination yields the Schur complement formula:

$$(\mathbf{I} - \mathbf{D}_+^{-1}\mathbf{C}_+\mathbf{A}^{-1}\mathbf{B}_+)\boldsymbol{\sigma}_p = \mathbf{D}_+^{-1}\mathbf{f}_p - \mathbf{D}_+^{-1}\mathbf{C}_+\mathbf{A}^{-1}\mathbf{f}. \tag{7.2}$$

It is easy to check that setting up this $p \times p$ linear system requires p applications of $\mathbf{A}^{-1}$ to compute $(\mathbf{A}^{-1}\mathbf{B}_+)$, p^2 inner products of N-vectors to compute $\mathbf{C}_+(\mathbf{A}^{-1}\mathbf{B}_+)$, and p^3 operations to compute $\mathbf{D}_+^{-1}(\mathbf{C}_+(\mathbf{A}^{-1}\mathbf{B}_+))$. The right-hand side requires one application of $\mathbf{A}^{-1}$, p inner products of N vectors and one application of $\mathbf{D}_+^{-1}$. Given $\boldsymbol{\sigma}_p$, the first N entries of the solution, namely $\boldsymbol{\sigma}$, can be obtained from

$$\boldsymbol{\sigma} = \mathbf{A}^{-1}\mathbf{f} - \mathbf{A}^{-1}\mathbf{B}_+\boldsymbol{\sigma}_p.$$

7.2. Deletion of triangles

Suppose now that one wants to subtract q triangles $T_{j_1}, T_{j_2}, \ldots, T_{j_q}$ from the original surface instead of adding them. This can be done with a simple trick; one adds new triangles $T_{N+1}, T_{N+2}, \ldots, T_{N+q}$ to the discretization in exactly the same spatial location as $T_{j_1}, T_{j_2}, \ldots, T_{j_q}$, imposing the condition that the charge density on the new triangles $\{\boldsymbol{\sigma}_{N+1}, \boldsymbol{\sigma}_{N+2}, \ldots, \boldsymbol{\sigma}_{N+q}\}$ negates the charge density on the original ones $\{\boldsymbol{\sigma}_{j_1}, \boldsymbol{\sigma}_{j_2}, \ldots, \boldsymbol{\sigma}_{j_q}\}$.

We leave it to the reader to verify the following lemma.

Lemma 7.1. Let the original linear system of (4.11) be denote by $\mathbf{A}\boldsymbol{\sigma} = \mathbf{f}$ and let $\{j_1, j_2, \ldots, j_q\}$ denote the indices of q triangles which are to be deleted. Consider the augmented linear system

$$\begin{pmatrix} \mathbf{A} & \mathbf{B}_- \\ \mathbf{C}_- & \mathbf{D}_- \end{pmatrix} \begin{pmatrix} \boldsymbol{\sigma} \\ \boldsymbol{\sigma}_- \end{pmatrix} = \begin{pmatrix} \mathbf{f} \\ 0 \end{pmatrix}, \tag{7.3}$$

where $\mathbf{D}_- \in \mathbb{R}^{q\times q}$, $\mathbf{B}_- \in \mathbb{R}^{N\times q}$, and $\mathbf{C}_- \in \mathbb{R}^{q\times N}$ with

$$\mathbf{D}_-(i,j) = \delta_{ij},$$

$$\mathbf{B}_-(i,l) = \begin{cases} 0 & \text{if } i = j_l, \\ \mathbf{A}(i,j_l) & \text{if } i \neq j_l, \end{cases}$$

$$\mathbf{C}_-(l,i) = \begin{cases} 1 & \text{if } i = j_l, \\ 0 & \text{if } i \neq j_l. \end{cases}$$

Then the solution components in $\boldsymbol{\sigma}$ are equal to those that would be obtained from the linear system (4.11) with the triangles $\{T_{j_1}, T_{j_2}, \ldots, T_{j_q}\}$ deleted.

7.3. Simultaneous addition and deletion of triangles

The low-rank modification of the linear system (4.11) corresponding to adding p triangles $\{T_{N+1}, \ldots, T_{N+p}\}$ and subtracting q triangles with indices $\{j_1, \ldots, j_q\}$ can be carried out simultaneously. It is straightforward to

https://doi.org/10.1017/S0962492906410011 Published online by Cambridge University Press

see that the final system has the following structure:

$$\begin{pmatrix} \mathbf{A} & \mathbf{B}_+ & \mathbf{B}_- \\ \mathbf{C}_+ & \mathbf{D}_+ & \mathbf{B}_* \\ \mathbf{C}_- & 0 & \mathbf{D}_- \end{pmatrix} \begin{pmatrix} \boldsymbol{\sigma} \\ \boldsymbol{\sigma}_+ \\ \boldsymbol{\sigma}_- \end{pmatrix} = \begin{pmatrix} \mathbf{f} \\ \mathbf{f}_p \\ 0 \end{pmatrix}, \tag{7.4}$$

where $\mathbf{B}_+, \mathbf{B}_-, \mathbf{C}_+, \mathbf{C}_-, \mathbf{D}_+, \mathbf{D}_-, \mathbf{f}_p$ are defined above and

$$\mathbf{B}_*(l, m) = A(T_{N+i}, T_{j_m}).$$

8. Numerical results II

Let us now consider a triangulated aircraft with 28252 elements (Figure 8.1). To compute potential flow around the aircraft, we assume the velocity field is given by

$$\nabla\Phi = \nabla\Phi^{\text{in}} + \nabla\Phi^{\text{scat}},$$

where

$$\nabla\Phi^{\text{in}} = (1, 0, 0).$$

That is, we assume the incoming velocity is uniform and oriented along the

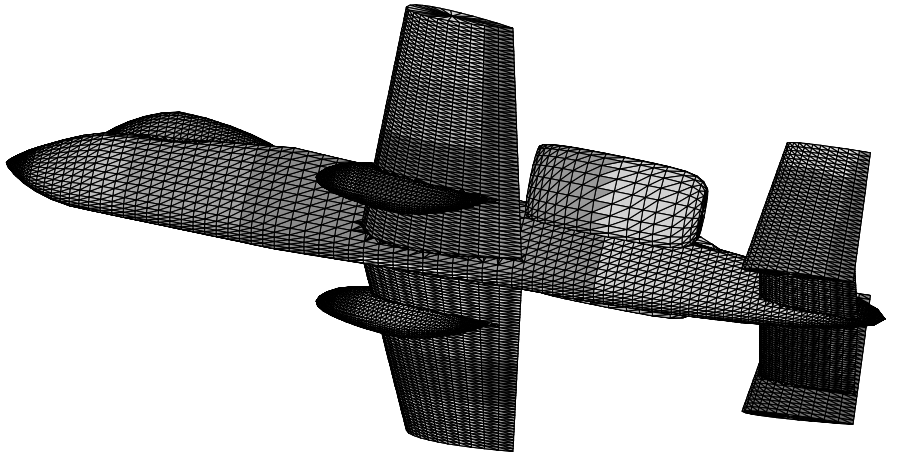


Figure 8.1. A triangulated aircraft.

https://doi.org/10.1017/S0962492906410011 Published online by Cambridge University Press

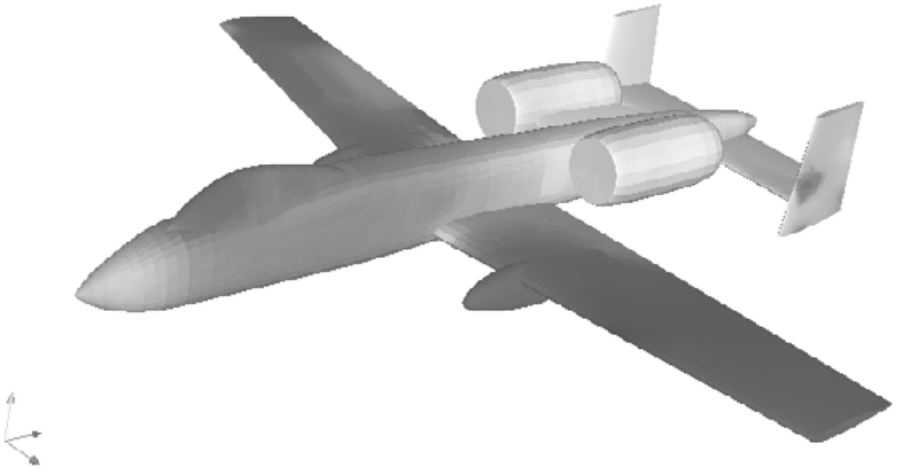


Figure 8.2. The induced potential on the aircraft surface.

x-axis. Φ^{scat} is the 'scattered' field induced by the aircraft in response to the zero normal flow (Neumann) boundary condition:

$$\frac{\partial \Phi}{\partial \nu} = 0$$

or

$$\frac{\partial \Phi^{\text{scat}}}{\partial \nu} = -\frac{\partial \Phi^{\text{in}}}{\partial \nu}.$$

For this the single-layer potential representation yields the integral equation (4.9) with $\lambda = -1$.

After compression to two digits of accuracy, the skeletonized Schur complement structure contains 4759 triangles. Our single-level scheme requires about 15 minutes for this step, while the cost for each subsequent application of the inverse is 0.2 seconds.

Suppose now that we introduce an auxiliary 'flap' (Figure 8.3) with fourteen new triangles. This is not a well-resolved structure: it is simply being used to illustrate the capability of the algorithm discussed in the preceding section. Using the updating method of the preceding section, the inverse matrix had to be applied to fourteen right-hand sides at a total cost of 14×0.2 seconds.

https://doi.org/10.1017/S0962492906410011 Published online by Cambridge University Press

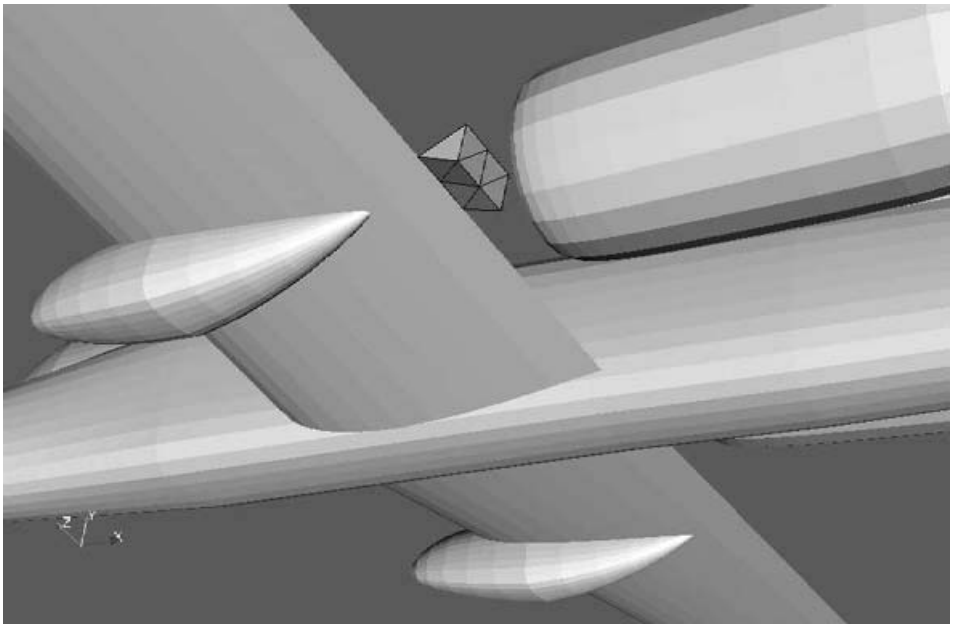


Figure 8.3. A small flap is added to the geometry.

When the number of triangles being added or subtracted is large, the Sherman–Morrison–Woodbury approach becomes inefficient. Rather than doing the update exactly, one can solve the system (7.4) iteratively, with preconditioner

$$\begin{pmatrix} \mathbf{A}^{-1} & 0 & 0 \\ 0 & \mathbf{I} & 0 \\ 0 & 0 & \mathbf{I} \end{pmatrix}. \tag{8.1}$$

9. Conclusions

During the past twenty years, many previously intractable problems in computational physics and engineering were brought within reach by the combination of hardware improvements and fast algorithms. For computational tools to be well-integrated into design processes, however, we will need more. Fast, direct methods are natural candidates for this, since they lead to very fast schemes for problems involving multiple right-hand sides, and rapid updating for modest changes to the system matrix. It is also worth noting that this technology will have an impact in many time-dependent simulations where implicit methods play a role. Such methods typically require the solution of some partial differential equation at each time step, often in a fixed geometry.

https://doi.org/10.1017/S0962492906410011 Published online by Cambridge University Press

In this paper, we have tried to present some of the ideas that form the basis for these direct solvers. This is an active area of research and there are many related schemes currently under development in a variety of research groups. We expect that these methods will mature rapidly over the next few years.

Acknowledgements

We would like to thank David Bindel for several useful discussions and for suggesting the sparse linear-algebraic viewpoint outlined in Section 3.

Appendix: Rapid inversion of operators.

As indicated in the Introduction, there is a body of work addressing the efficient, direct solution of integral equations: see, for instance, Canning and Rogovin (1998), Chandrasekaran *et al.* (2006), Chen (2002), Cheng *et al.* (2005), Chew (1989), Eidelman and Gohberg (1999), Gope *et al.* (2005), Greengard and Rokhlin (1991), Hackbusch (1999), Hackbusch and Khoromskij (2000), Lee and Greengard (1997), Martinsson and Rokhlin (2005, 2007), Michielssen *et al.* (1996), Pals (2004), Starr and Rokhlin (1994), and Zhu and White (2005).

Here, we illustrate the basic idea of these multilevel schemes. For this, suppose that we would like to solve the system of linear-algebraic equations

$$A\,X = Y, \tag{A.1}$$

with the $n \times n$ matrix A defined by the formula

$$A_{ij} = \log(|\, y_i - x_j \,|), \tag{A.2}$$

whenever $i \neq j$, and

$$A_{ii} = 1, \tag{A.3}$$

with the points $x_1, x_2, \ldots, x_n \in R$ defined by the formulae

$$x_i = -1 + \frac{2\,(i-1)}{(n-1)}. \tag{A.4}$$

In other words, the points $\{x_i\}$ are equispaced on the interval $[-1, 1]$ (note the similarity to the Example 2 above). Clearly, the matrix A is dense, and solving the system (2.9) directly will require order n^3 operations. Below, we outline a simple multilevel direct procedure that will solve (2.9) in order $n\,(\log(n))^2$ operations.

For simplicity, we will assume that $n = m\,2^k$, where m is a small integer number ($m \sim 20$ is a reasonable choice), and k is a positive integer. We start by subdividing the interval $[-1, 1]$ into two subintervals $[a, j]$, $[j, q]$, with $a = -1$, $j = 0$, and $q = 1$. Each of the intervals $[a, j]$, $[j, q]$ is subdivided into

https://doi.org/10.1017/S0962492906410011 Published online by Cambridge University Press

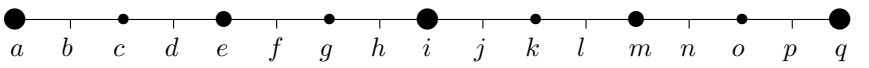


Figure A.1. The interval $[-1, 1]$ is recursively subdivided four times.

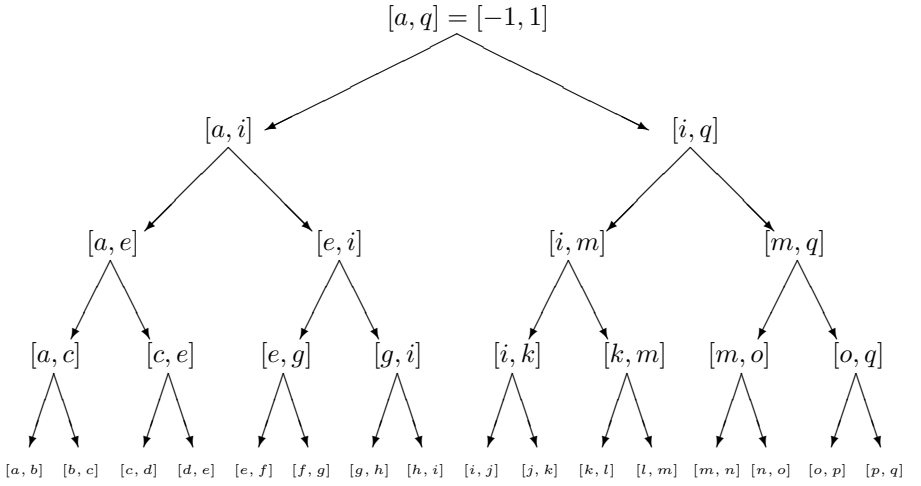


Figure A.2. A depiction of the tree structure created by recursive subdivision.

two subintervals of equal length: $[a, j] = [a, f] \cup [f, j]$, $[j, q] = [j, n] \cup [n, q]$. After continuing this process of recursive splitting for two more steps, we obtain the situation depicted in Figure A.1.

We impose a tree structure on the obtained subdivision, as depicted in Figure A.2; now the intervals $[a, j]$, $[j, q]$, are children of the interval $[a, q]$; the intervals $[a, f]$, $[f, j]$ are children of the interval $[a, j]$, *etc.*

Figure A.4(a) depicts the matrix A, subdivided into a collection of submatrices; with the exception of the diagonal blocks (marked with X), each of the submatrices is responsible for the interaction of a pair of subintervals depicted in Figure A.1 having the same parent (as depicted in Figure A.2).

Observation A.1. Due to Lemma 2.1 above, the numerical rank of each of the submatrices marked with L in Figure A.4 is roughly $\log(n) \cdot \log(1/\varepsilon)$, and we will assume that it has been represented in the form

$$L = \sum_{i=1}^{p} \alpha_i \, \beta_i^*, \tag{A.5}$$

where $p \sim \log(n) \cdot \log(1/\varepsilon)$, and $\alpha_1, \alpha_2, \ldots \alpha_p$, $\beta_1, \beta_2, \ldots \beta_p$, are vectors of appropriate dimensionality.

https://doi.org/10.1017/S0962492906410011 Published online by Cambridge University Press

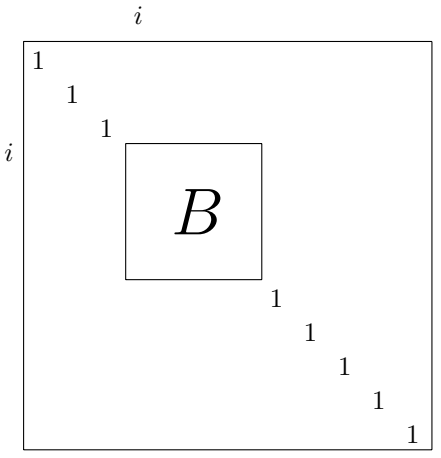


Figure A.3. The matrix $B^{n,i}$ is derived from the $n \times n$ identity matrix by replacing the diagonal block of dimension $m \times m$ beginning at the (i, i) entry with B.

The above observation leads to a 'fast' procedure for the solution of (2.9) in a fairly straightforward manner; below is a description of a rudimentary scheme of this type. We start with a definition.

Definition A.1. Suppose that m, n, i are three positive integers, such that $m \leq n$, $i + m \leq n + 1$, and that B is an $m \times m$ matrix. We will denote by $B^{n,i}$ the $n \times n$ matrix obtained from the $n \times n$ identity matrix by replacing with B the $m \times m$ diagonal block starting at the ith row, and ending at the $(i+m-1)$st row (see Figure A.3). In other words, when the operator $B^{n,i} : R^n \rightarrow R^n$ acts on the vector $x \in R^n$, it applies the operator $B : R^m \; to R^m$ to a subset of the vector x consisting of the elements $x_i, x_{i+1}, x_{i+2}, \ldots, x_{i+m-1}$, and leaves the rest of the elements of x unchanged.

Step 1. We start by inverting each of the diagonal blocks $X_1, X_2, \ldots X_k$ in A (see Figure A.4(a)), obtaining their inverses $Y_1, Y_2, \ldots Y_k$, and multiplying A by the product

$$Y = Y_1^{n,1} \circ Y_2^{n,m+1} \circ Y_3^{n,2m+1} \circ \cdots \circ Y_{2^k}^{n,(2^k-1)m+1}. \tag{A.6}$$

We observe that the matrices $Y_1^{n,1}, Y_2^{n,m+1}, \ldots, Y_{2^k}^{n,(2^k-1)m+1}$ commute with each other, and that the product $B_1 = YA$ has the form depicted in Figure A.4(a) (the 'I' in the diagonal blocks denote identity matrices). We also observe that the cost of this step is of the order $2^k \cdot m^3 + \log_2(n) \cdot m^2$.

https://doi.org/10.1017/S0962492906410011 Published online by Cambridge University Press

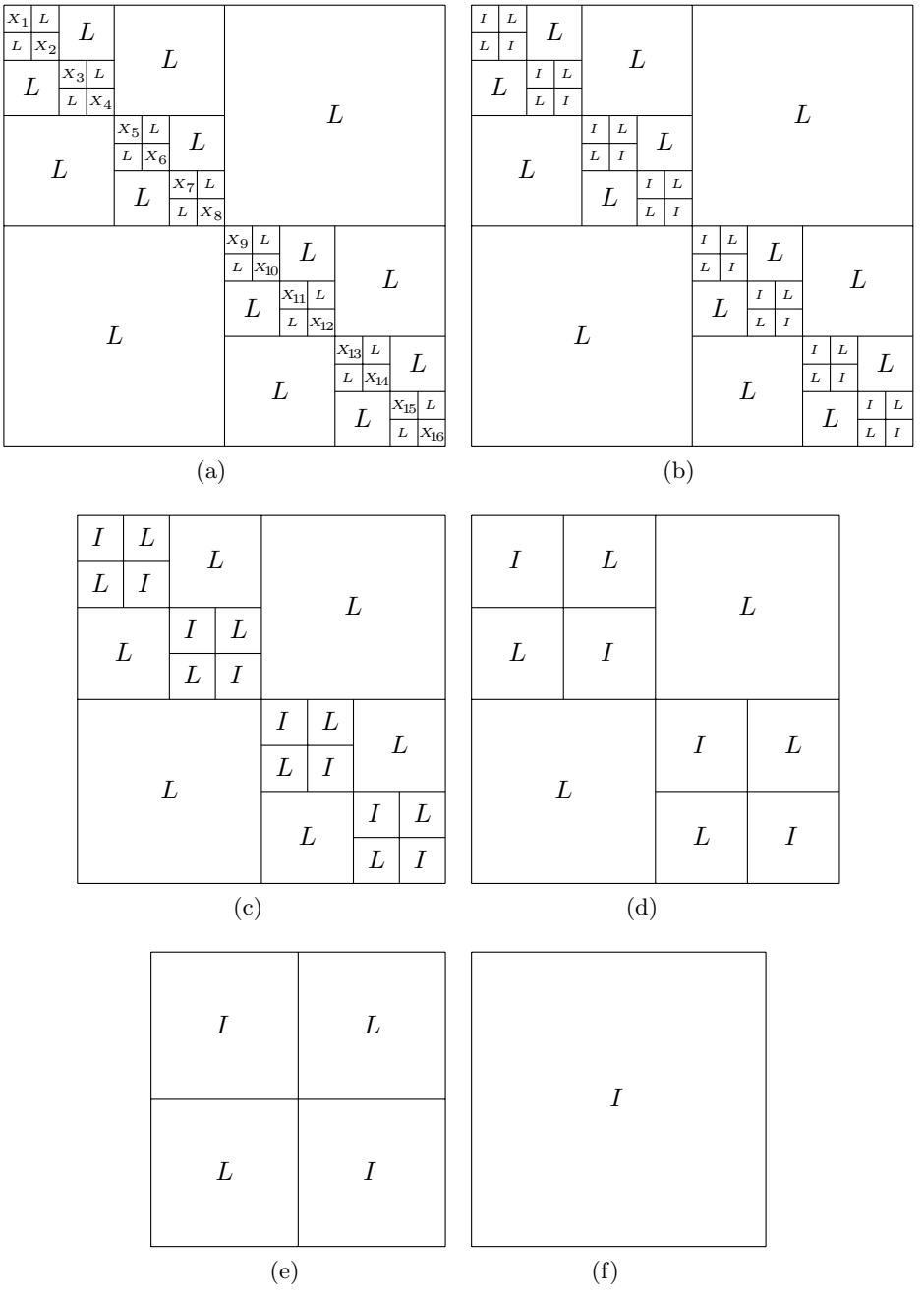


Figure A.4. The system matrix for Example 4 is shown in (a). Each off-diagonal block matrix corresponds to the interactions of a pair of subintervals having the same parent. A graphical depiction of the inversion process is shown in (b)–(f).

https://doi.org/10.1017/S0962492906410011 Published online by Cambridge University Press

Step 2. We observe that each of the diagonal blocks of dimensionality $2m \times 2m$ in B_1 (depicted in Figure A.1) is a sum of the identity and a matrix of rank $2p$ (see (A.5) above) and that the total number of such blocks in Figure A.4(a) is 2^{k-1}. Denoting the inverse of the jth of these blocks by Z_j, we multiply the matrix B_1 from the left by the product

$$Z = Z_1^{n,1} \circ Z_2^{n,m+1} \circ Z_3^{n,2m+1} \circ \cdots \circ Z_{2^k}^{n,(2^k-1)m+1}, \tag{A.7}$$

obtaining the matrix B_2 depicted in Figure A.4(b). We also apply Z to the right-hand side, so that the resulting system of linear equations is equivalent to the initial one. The cost of this step is of the order $2^{(k-1)} \cdot m^2 + \log_2(n) \cdot m^2$.

Step 3. This step is identical to Step 2, except the dimensionality of the diagonal blocks is doubled, and the number of these blocks is halved; the result is depicted in Figure A.4(c), and the cost remains of the order $2^{(k-1)} \cdot m^2 + \log_2(n) \cdot m^2$.

Step *k*. This final step is illustrated in Figures A.4(e) and A.4(f), and its result is the identity matrix in dimension $m \cdot 2^k = n$; its cost is the same as that of Steps 1 and 2.

Summing up the costs of all Steps 1 to k, and observing that $k \sim \log_2(n)$, we obtain the total cost $O(n \log^4(n))$.

Remark A.1. While the cost of the simple scheme described above is of the order $n \log^4(n)$, fairly simple modifications produce algorithms whose cost is proportional to n (see, for example, Martinsson and Rokhlin (2005)). In higher dimensions, logarithmic factors can not be eliminated entirely (at least, not with the currently available techniques), and one ends up with algorithms costing $O(n \log^\alpha(n))$ operations, with $\alpha \in [1, 3]$. This is an active area of research, and both the algorithms and the resulting estimates are undergoing rapid evolution.

REFERENCES

B. Alpert and V. Rokhlin (1991), 'A fast algorithm for the evaluation of Legendre expansions', *SIAM J. Sci. Statist. Comput.* **12**, 158–179.

B. Alpert, G. Beylkin, R. Coifman and V. Rokhlin (1993), 'Wavelet-like bases for the fast solution of second-kind integral equations', *SIAM J. Sci. Comput.* **14**, 159–184.

M. D. Altman, J. P. Bardhan, B. Tidor and J. K. White (2006), 'FFTSVD: A fast multiscale boundary-element method solver suitable for bio-MEMS and biomolecule simulation', *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems* **25**, 274–284.

J. P. Bardhan, M. D. Altman, S. M. Lippow, B. Tidor and J. K. White (2005), 'A curved panel integration technique for molecular surfaces', in *Proc. NSTI–Nanotech 2005 Conf.*, Vol. 1, pp. 512–515.

https://doi.org/10.1017/S0962492906410011 Published online by Cambridge University Press

G. Beylkin, R. Coifman and V. Rokhlin (1991), 'Fast wavelet transforms and numerical algorithms I', *Comm. Pure Appl. Math.* **14**, 141–183.
A. H. Boschitsch, M. O. Fenley and W. K. Olson (1999), 'A fast adaptive multipole algorithm for calculating screened Coulomb (Yukawa) interactions', *J. Comput. Phys.* **151**, 212.
F. X. Canning and K. Rogovin (1998), 'Fast direct solution of standard moment-method matrices', *IEEE Antennas Propag.* **40**, 15–26.
J. Carrier, L. Greengard and V. Rokhlin (1988), 'A fast adaptive multipole algorithm for particle simulations', *SIAM J. Sci. Statist. Comput.* **9**, 669–686.
S. Chandrasekaran, P. Dewilde, M. Gu, W. Lyons and T. Pals (2006), 'A fast solver for HSS representations via sparse matrices', *SIAM J. Matrix Anal. Appl.* **29**, 67–81.
Y. Chen (2002). 'Fast direct solver for the Lippman–Schwinger equation', *Adv. Comput. Math.* **16**, 175–190.
H. Cheng, Z. Gimbutas, P. G. Martinsson and V. Rokhlin (2005), 'On the compression of low rank matrices', *SIAM J. Sci. Comput.* **26**, 1389–1404.
W. C. Chew (1989). 'An N^2 algorithm for the multiple scattering solution of N scatterers', *Micro. Opt. Tech. Lett.* **2**, 380–383.
W. C. Chew, J.-M. Jin, E. Michielssen and J. Song, editors (2001). *Fast and Efficient Algorithms in Computational Electromagnetics*, Artech House, Boston.
J. W. Cooley and J. W. Tukey (1965), 'An algorithm for the machine calculation of complex Fourier series', *Math. Comput.* **19**, 297–301.
E. Darve and P. Have (2004), 'Fast multipole method for Maxwell equations stable at all frequencies', *Royal Soc. London, Trans. Ser. A* **362**, 603–628.
M. E. Davis and J. A. McCammon (1990), 'Electrostatics in biomolecular structure and dynamics' *Chem. Rev.* **90**, 509–521.
Y. Eidelman and I. Gohberg (1999). 'Linear complexity inversion algorithms for a class of structured matrices', *Integral Equations Operator Theory* **35**, 28–52.
Z. Gimbutas (1999), A generalized fast multipole method for non-oscillatory kernels. PhD Dissertation, Yale University.
D. Gope, I. Chowdhury and V. Jandhyala (2005), 'DiMES: Multilevel fast direct solver based on multipole expansions for parasitic extraction of massively coupled 3D microelectronic structures', in *Proc. 42nd Annual Conference on Design Automation*, pp. 159–162.
S. A. Goreinov, E. E. Yrtyshnikov and N. L. Zamarashkin (1997), 'A theory of pseudoskeleton approximations', *Linear Algebra Appl.* **261**, 1–21.
I. S. Gradshteyn and I. M. Ryzhik (2000), *Table of Integrals, Series, and Products*, Academic Press, New York.
L. Greengard and J. Helsing (1998), 'On the numerical evaluation of elastostatic fields in locally isotropic two-dimensional composites', *J. Mech. Phys. Solids* **46**, 1441–1462.
L. Greengard and V. Rokhlin (1987), 'A fast algorithm for particle simulations', *J. Comput. Phys.* **73**, 325–348.
L. Greengard and V. Rokhlin (1991), 'On the numerical solution of two-point boundary value problems', *Comm. Pure Appl. Math.* **44**, 419–452.
L. Greengard and V. Rokhlin (1997), A new version of the fast multipole method for the Laplace equation in three dimensions. In *Acta Numerica*, Vol. 6, Cambridge University Press, pp. 229–269.

https://doi.org/10.1017/S0962492906410011 Published online by Cambridge University Press

L. Greengard and J. Strain (1991). 'The Fast Gauss Transform', *SIAM J. Sci. Statist. Comput.* **12**, 79–94.
M. Gu and S. C. Eisenstat (1996), 'Efficient algorithms for computing a strong rank-revealing *QR* factorization', *SIAM J. Sci. Comput.* **17**, 848–869.
R. B. Guenther and J. W. Lee (1988), *Partial Differential Equations of Mathematical Physics and Integral Equations*, Prentice-Hall, Englewood Cliffs, NJ.
W. Hackbusch (1999), 'A sparse matrix arithmetic based on H-matrices I: Introduction to H-matrices', *Computing* **62**, 89–108.
W. Hackbusch and B. N. Khoromskij (2000), 'A sparse H-matrix arithmetic II: Application to multi-dimensional problems', *Computing* **64**, 21–47.
W. Hackbusch and Z. P. Nowak (1989), 'On the fast matrix multiplication in the boundary element method by panel clustering', *Numer. Math.* **54**, 463–491.
M. J. Holst, N. A. Baker and F. Wang (2000), 'Adaptive multilevel finite element solution of the Poisson–Boltzmann equation I: Algorithms and examples', *J. Comput. Chem.* **21**, 1319–1342.
J. Huang and L. Greengard (2002), 'A new version of the fast multipole method for screened Coulomb interactions interactions in three dimensions', *J. Comput. Phys.* **180**, 642–658.
D. L. James and D. K. Pai (1999), 'ArtDefo: Accurate real time deformable objects', in *Proc. SIGGRAPH 99*, pp. 65–72.
S. Kapur and D. Long (1997), 'IES^3: A fast integral equation solver for efficient 3-D extraction', in *Proc. IEEE International Conference on Computer Aided Design*, pp. 448–455.
R. Kastner (1989), 'An 'add-on method' for the analysis of scattering from large planar structures', *IEEE Trans. Antennas Propag.* **37**, 353–361.
S. S. Kuo, M. D. Altman, J. P. Bardhan, B. Tidor and J. K. White (2002), 'Fast methods for simulation of biomolecular electrostatics', in *Proc. IEEE–ACM Int. Conf. Comput. Aided Design*, pp. 466–473.
J.-Y. Lee and L. Greengard (1997), 'A fast adaptive numerical method for stiff two-point boundary value problems', *SIAM J. Sci. Comput.* **18**, 403–429.
J. Liang and S. Subramaniam (1997), 'Computation of molecular electrostatics with boundary element methods', *Biophys. J.* **73**, 1830.
B. Lu, X. Cheng, J. Huang and J. A. McCammon (2006), 'Order N algorithm for computation of electrostatic interactions in biomolecular systems', *Proc. Nat. Acad. Sci.* **103**, 19314–19319.
S. A. Marshall, C. L. Vizcarra and S. L. Mayo (2005), 'One- and two-body decomposable Poisson–Boltzmann methods for protein design calculations', *Protein Science* **14**, 1293–1304.
P.-G. Martinsson (2006), 'Fast evaluation of electrostatic interactions in multiphase dielectric media', *J. Comput. Phys.* **211**, 289–299.
P.-G. Martinsson and V. Rokhlin (2005), 'A fast direct solver for boundary integral equations in two dimensions', *J. Comput. Phys.* **205**, 1–23.
P.-G. Martinsson and V. Rokhlin (2007), 'A fast direct solver for scattering problems involving elongated structures', *J. Comput. Phys.* **221**, 288–302.
E. Michielssen, A. Boag and W. C. Chew (1996), 'Scattering from elongated objects: Direct solution in $O(N \log^2 N)$ operations', *IEEE Proc. H* **143**, 277–283.

https://doi.org/10.1017/S0962492906410011 Published online by Cambridge University Press

K. Nabors and J. White (1991), 'FASTCAP: A multipole accelerated 3-D capacitance extraction program', *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems* **10**, 1447–1459.
N. Nishimura (2002), 'Fast multipole accelerated boundary integral equation methods', *Appl. Mech. Rev.* **55**, 299–324.
T. Pals (2004), Multipole for scattering computations: Spectral discretization, stabilization, fast solvers. PhD Dissertation, Department of Electrical and Computer Engineering, University of California, Santa Barbara.
J. R. Phillips and J. White (1997), 'A precorrected-FFT method for electrostatic analysis of complicated 3-D structures', *IEEE Trans. Computer-Aided Design* **16**, 1059–1072.
W. Rocchia, S. Sridharan, A. Nicholls, E. Alexov, A. Chiabrera and B. Honig (2002), 'Rapid grid-based construction of the molecular surface for both molecules and geometric objects: Applications to the finite difference Poisson–Boltzmann method', *J. Comput. Chem.* **23**, 128–137.
V. Rokhlin (1985), 'Rapid solution of integral equations of classical potential theory', *J. Comput. Phys.* **60**, 187–207.
V. Rokhlin (1988), 'A fast algorithm for the discrete Laplace transformation', *J. Complexity* **4**, 12–32.
Y. Saad and M. H. Schultz (1986), 'GMRES: A generalized minimum residual algorithm for solving nonsymmetric linear systems', *SIAM J. Sci. Statist. Comput.* **7**, 856–869.
K. A. Sharp and B. Honig (1990), 'Electrostatic interactions in macromolecules: Theory and applications', *Ann. Rev. Biophys. Biophys. Chem.* **19**, 301–332.
J. Sherman and W. J. Morrison (1949), 'Adjustment of an inverse matrix corresponding to changes in the elements of a given column or a given row of the original matrix', *Ann. Math. Statist.* **20**, 621.
P. Starr and V. Rokhlin (1994). 'On the numerical solution of two-point boundary value problems II', *Comm. Pure Appl. Math.* **47**, 1117–1159.
J. Strain (1992), 'The fast Laplace transform based on Laguerre functions', *Math. Comp.* **58**, 275–284.
M. A. Woodbury (1950), Inverting modified matrices. Memorandum Report 42, Statistical Research Group, Princeton University.
L. Ying, G. Biros, D. Zorin and M. H. Langston (2003), 'A new parallel kernel-independent fast multipole method', in *Proc. ACM/IEEE Conf. on Supercomp.*, p. 14.
Z. Zhu and J. K. White (2005), 'FastSies: A fast stochastic integral equation solver for modeling the rough surface effect', in *Proc. 2005 IEEE/ACM Int. Conference on Computer-Aided Design*, pp. 675–682.

Online references

Protein Data Bank: www.rcsb.org
STLib package: www.cacr.caltech.edu/~sean/projects/stlib/html/mst

https://doi.org/10.1017/S0962492906410011 Published online by Cambridge University Press