Chapter 4

A Fast, High-Order Method in Three Dimensions

In this chapter, we present a new, fast, high-order method in three dimensions. This method is motivated in part by the two-dimensional polar coordinates approach discussed in the previous two chapters. One could propose a direct generalization of the two-dimensional approach to a three-dimensional spherical coordinates method. However, to obtain the same $\mathcal{O}(N \log N)$ complexity, one would need to replace the FFTs with fast spherical harmonics transforms, robust versions of which are not readily available. Furthermore, as evidenced by the method for radial integration in the two-dimensional method, evaluation of special functions of high-order presents several mathematical as well as practical difficulties. Therefore, instead of extending the two-dimensional polar coordinates approach, we present a three-dimensional approach based on trapezoidal rule integration and Fourier series approximation in *Cartesian* coordinates. The fast, high-order method we obtain is actually much simpler than the two-dimensional approach while yielding approximately the same accuracy and efficiency.

In particular, while the polar coordinates approach in two dimensions required the identification and resolution of singularities in the Fourier coefficients $(mu)_\ell(r)$, this Cartesian approach in three dimensions requires no special consideration of scatterer geometry. This enables the construction of quite complicated scatterers containing discontinuities, corners, and cusps by simply summing the truncated Fourier series representations of each of the individual components of the scatterer (see Figures 5.13 and 5.14 as well as Table 5.13). There is no need to locate and resolve singularities.

As in the two-dimensional approach, the core of the numerical method is an efficient,

high-order scheme for computing the integral operator $(Ku)(x_j)$ (see (1.6)) at the given discretization points x_j. We thereby obtain the linear system

$$u(x_j) - (Ku)(x_j) = u^i(x_j), \tag{4.1}$$

whose solution $u(x_j)$ approximates the total field. An iterative solver then provides the solution at the discretization points $u(x_j)$. We discuss the advantages and disadvantages of various iterative solvers in Section 4.5.

Let m be contained within a box $\Omega_{[a,b]}$ with corners $a, b \in \mathbb{R}^3$, i.e., $\Omega_{[a,b]} = \{x : a_q \leq x_q \leq b_q, q = 1, 2, 3\}$. Then,

$$(Ku)(x) = -\kappa^2 \int_{\Omega_{[a,b]}} g(x-y)m(y)u(y)dy,$$

where u is the solution of (1.6) in $\mathbb{R}^3$.

Before specifying the details of our approach, we sketch its key aspects. As mentioned in the introduction, we decompose the Green's function into a smooth part with infinite support, $g_{smth}(x)$, and a singular part with compact support, $g_{cmp}(x)$. More precisely, we define $g_{smth}(x)$ and $g_{cmp}(x)$ by

$$g(x) = g(x)(1 - p(x)) + g(x)p(x) = g_{smth}(x) + g_{cmp}(x),$$

where $p(x) \in C^\infty$ is a partition of unity function such that $p(x) = 1$ near $x = 0$ and $p(x) = 0$ outside some neighborhood of $x = 0$. (Of course, there are many such partition of unity functions, and we do not specify a particular choice at this time.) It is then necessary to compute the two convolutions

$$(K_{smth}u)(x) = -\kappa^2 \int_{\Omega_{[a,b]}} g_{smth}(x-y)m(y)u(y)dy \tag{4.2}$$

$$(K_{cmp}u)(x) = -\kappa^2 \int_{\Omega_{[a,b]}} g_{cmp}(x-y)m(y)u(y)dy. \tag{4.3}$$

The following two sections describe the two different high-order methods we use to evaluate $K_{smth}u$ and $K_{cmp}u$, respectively. For both of these convolutions, the substitution of the scatterer by an appropriate Fourier-smoothed scatterer is absolutely key in obtaining high-order accuracy. As will be shown, the methods used to evaluate these convolutions

require the computation of an integral of the form

$$\int_{\Omega_{[a,b]}} m(y)w(y)dy,$$

where w is defined in $\mathbb{R}^3$. In each case, w is the product of the total field u and a known C^∞ function. Hence, the regularity of w is given by the regularity of u, which is related to the regularity of m. In particular, if $m \in L^\infty$, $u \in C^1$; and if $m \in C^{k,\alpha}$, $u \in C^{k+2,\alpha}$ (see Theorem 2.5).

Note that the integrands in each case are quite similar to the example in Figure 1.2, with one important exception: Although w is smooth (at least C^1), it is generally not periodic. Hence, direct substitution of the scatterer by its truncated Fourier series will not yield high-order convergence. However, we observe that since m vanishes outside of $\Omega_{[a,b]}$, we can extend the domain of integration without affecting the value of the integral. Similarly, any modification to w outside of the support of m, $supp(m)$, does not affect the integral.

With these observations in mind, we increase the computational domain to the box $\Omega_{[a-\delta,b+\delta]}$ for some $\delta \in \mathbb{R}^3$ such that the components $\delta_q > 0$. This gives us u and hence, w on $\Omega_{[a-\delta,b+\delta]}$. We then multiply $w(y)$ by a smooth cutoff function $p_m(y)$ such that $p_m \in C^\infty$, $p_m(y) = 1$ for $y \in supp(m)$ and $p_m(y) = 0$ for $y \notin \Omega_{[a-\delta,b+\delta]}$. (Of course, as with the partition of unity function, p, there are many such functions, p_m, and we do not specify a particular choice at this time.) This gives us

$$\int_{\Omega_{[a,b]}} m(y)w(y)dy = \int_{\Omega_{[a-\delta,b+\delta]}} m(y)p_m(y)w(y)dy.$$

Since $p_m(y)$ vanishes outside of $\Omega_{[a-\delta,b+\delta]}$, one can extend $p_m(y)w(y)$ as a smooth and periodic function. Following the example in the introduction, we can now substitute m by its truncated Fourier series to obtain high-order accuracy when integrating with the trapezoidal rule (see Figure 1.2 and Table 1.2), i.e., replace m by

$$m^F(x) = \sum_{\ell_1=-F_1}^{F_1} \sum_{\ell_2=-F_2}^{F_2} \sum_{\ell_3=-F_3}^{F_3} m_\ell e^{2\pi i c_\ell \cdot x},$$

where $\ell = (\ell_1, \ell_2, \ell_3)$ and the q^{th} component of c_ℓ, $(c_\ell)_q = \ell_q/[(b_q - a_q) + 2\delta_q]$ for $q = 1, 2, 3$

and $F = (F_1, F_2, F_3)$. This gives

$$\int_{\Omega_{[a,b]}} m(y)w(y)dy \approx \int_{\Omega_{[a-\delta,b+\delta]}} m^F(y)p_m(y)w(y)dy.$$

Thus, we obtain high-order accuracy in such integrals essentially by replacing m by $\widetilde{m} = m^F p_m$ and by replacing $\Omega_{[a,b]}$ by $\Omega_{[\tilde{a},\tilde{b}]}$, where $\tilde{a} = a - \delta$ and $\tilde{b} = b + \delta$.

In Sections 4.1 and 4.2, we describe in more detail the evaluation of $K_{smth}u$ and $K_{cmp}u$, respectively. In Section 4.3, we discuss a few additional details associated with this substitution of m by m^F. Section 4.4 briefly describes the method we use to compute the Fourier coefficients of g_{cmp}. Finally, in Section 4.5, we describe our parallel implementation of the method and discuss the relative advantages of the various linear solvers.

4.1 High-Order Convolution with Smooth Part

We compute $K_{smth}u$ by means of the trapezoidal rule after substituting m by $\widetilde{m}$ and $\Omega_{[a,b]}$ by $\Omega_{[\tilde{a},\tilde{b}]}$ as described above. Given a number of discretization points $N \in \mathbb{N}^3$. Define the discretization spacing $h \in \mathbb{R}^3$ such that $h_q = (\tilde{b}_q - \tilde{a}_q)/N_q$ for $q = 1, 2, 3$. Define the associated evenly spaced discretization points $x_j, y_k \in \mathbb{R}^3$ such that $(x_j)_q = \tilde{a}_q + j_q h_q$ and $(y_k)_q = \tilde{a}_q + k_q h_q$, where j and k are three-dimensional indices such that their components satisfy $0 \le j_q, k_q \le N_q$ for $q = 1, 2, 3$.

Since $\widetilde{m}(x)$ vanishes on the boundary of $\Omega_{[\tilde{a},\tilde{b}]}$ (because $\widetilde{m}$ is smooth and compactly supported in $\Omega_{[\tilde{a},\tilde{b}]}$), the trapezoidal rule gives

$$(K_{smth}u)(x_j) \approx Prod(h) \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} \sum_{k_3=0}^{N_3-1} g_{smth}(x_j - y_k)\widetilde{m}(y_k)u(y_k),$$

where the notation, $Prod(h)$, which we use throughout this chapter, stands for $Prod(h) = h_1 h_2 h_3$. To further simplify the notation, we denote the triple sum above as $\sum_{k=0}^{N-1}$, which allows us to write

$$\begin{aligned} (K_{smth}u)(x_j) &\approx Prod(h) \sum_{k=0}^{N-1} g_{smth}(x_j - y_k)\widetilde{m}(y_k)u(y_k) \\ &= Prod(h) \sum_{k=0}^{N-1} (g_{smth})_{j-k}\widetilde{m}_k u_k. \end{aligned}$$

Here $(g_{smth})_k = g_{smth}\,(\,(k_1h_1, k_2h_2, k_3h_3)\,)$, $\widetilde{m}_k = \widetilde{m}(y_k)$ and $u_k = u(y_k)$.

Hence, using the trapezoidal rule to evaluate this integral is algorithmically equivalent to computing a discrete convolution. We compute this convolution using FFTs in $\mathcal{O}(Prod(N)\log Prod(N))$ operations [45, pp. 531–536]. Thus, we obtain an efficient and high-order accurate method for computing $K_{smth}u$.

4.2 High-Order Convolution with Singular Part

To obtain high-order accuracy in the computation of (4.3), we approximate $K_{cmp}u$ by a truncated Fourier series. As described in the introduction, a truncated Fourier series provides high-order accuracy if the approximated function is smooth and periodic. Since g_{cmp} and m are both compactly supported, $(K_{cmp}u)(x)$ vanishes for points x sufficiently far from the inhomogeneity. More precisely, assume that the support of g_{cmp} is contained in a box $\Omega_{[-d,d]}$. Then, for $x \notin \Omega_{[a-d,b+d]}$, $(K_{cmp}u)(x) = 0$. Furthermore, since the convolution is a smoothing operation, $(K_{cmp}u)(x)$ is a smooth function, *even in the case of a discontinuous scatterer* (see Theorem 2.5). Therefore, $(K_{cmp}u)$ can be extended as a smooth and periodic function on $\mathbb{R}^3$.

Hence, we may represent $K_{cmp}u$ accurately by a truncated Fourier series if we choose a period for the expansion that contains $\Omega_{[a-d,b+d]}$, i.e.,

$$(K_{cmp}u)(x) \approx \sum_{\ell=-M}^{M} (K_{cmp}u)_\ell e^{2\pi i c_\ell\cdot(x-a)}, \tag{4.4}$$

where $(c_\ell)_q = \frac{\ell_q}{B_q - A_q}$ and $A_q \le a_q - d_q \le b_q + d_q \le B_q$ for $q = 1, 2, 3$. Note that according to our convention, we have denoted the triple sum as $\sum_{\ell=-M}^{M}$. We have also shifted the Fourier basis functions by a. This simplifies the presentation somewhat.

We must now compute the Fourier coefficients $(K_{cmp}u)_\ell$. We have

$$\begin{aligned}
(K_{cmp}u)_\ell &= -\frac{\kappa^2}{Prod(B-A)} \int_{\Omega_{[A,B]}} (K_{cmp}u)(x) e^{-2\pi i c_\ell\cdot(x-a)} dx \\
&= -\frac{\kappa^2}{Prod(B-A)} \int_{\Omega_{[A,B]}} m(y)u(y) e^{-2\pi i c_\ell\cdot(y-a)} dy \int_{\Omega_{[-d,d]}} g_{cmp}(z) e^{-2\pi i c_\ell\cdot z} dz \\
&= -\kappa^2 (g_{cmp})_\ell\ (mu)_\ell,
\end{aligned}$$

where

$$(g_{cmp})_\ell = \int_{\Omega_{[-d,d]}} g_{cmp}(z) e^{-2\pi i c_\ell \cdot z} dz$$

and

$$(mu)_\ell = \frac{1}{Prod(B-A)} \int_{\Omega_{[A,B]}} m(y)u(y) e^{-2\pi i c_\ell \cdot (y-a)} dy.$$

Note that the coefficients $(g_{cmp})_\ell$ are computed with respect to a slightly different set of basis functions (they are not shifted by a) and that we integrate over $\Omega_{[-d,d]}$ instead of $\Omega_{[A,B]}$. Because $g_{cmp}(x)$ is known analytically, we need only compute its Fourier coefficients once at the beginning of each run. Furthermore, because of the singularity in g_{cmp}, we must take special care to compute these coefficients accurately. Our method for computing these coefficients is presented in Section 4.4.

To compute the Fourier coefficients $(mu)_\ell$, we again use the trapezoidal rule. As in the previous section, we obtain high-order accuracy by substituting m by $\widetilde{m}$

$$\begin{aligned}
(mu)_\ell &= \frac{1}{Prod(B-A)} \int_{\Omega_{[A,B]}} m(y)u(y) e^{-2\pi i c_\ell \cdot (y-a)} dy \\
&\approx \frac{1}{Prod(B-A)} \int_{\Omega_{[\tilde{a},\tilde{b}]}} \widetilde{m}(y)u(y) e^{-2\pi i c_\ell \cdot (y-a)} dy \\
&\approx \frac{Prod(h)}{Prod(B-A)} \sum_{j=0}^{N-1} \widetilde{m}_j u_j e^{-2\pi i c_\ell \cdot (j_1 h_1, j_2 h_2, j_3 h_3)}.
\end{aligned}$$

Note that, as indicated in the second equation above, we need only integrate over the domain $\Omega_{[\tilde{a},\tilde{b}]}$ since this domain contains the support of $\widetilde{m}$.

We can evaluate this expression with an FFT by choosing $A = (A_1, A_2, A_3)$ and $B = (B_1, B_2, B_3)$ such that $(B_q - A_q)/h_q$ is an integer for each $q = 1, 2, 3$. In other words, the larger domain $\Omega_{[A,B]}$ must be exactly an integer number of cells larger than the smaller domain $\Omega_{[\tilde{a},\tilde{b}]}$ in each dimension. Then, defining $\widetilde{N} \in \mathbb{N}^3$ such that $\widetilde{N}_q = (B_q - A_q)/h_q$ for $q = 1, 2, 3$, and $(c_\ell)_q = \ell_q/(B_q - A_q)$, we obtain

$$(mu)_\ell \approx \frac{1}{Prod(\widetilde{N})} \sum_{j=0}^{\widetilde{N}-1} m_j u_j e^{-2\pi i \ell \cdot (j_1/\widetilde{N}_1, j_2/\widetilde{N}_2, j_3/\widetilde{N}_3)},$$

where $m_j u_j = 0$ if $j_q > N_q$ for any $q = 1, 2, 3$. Hence, this discrete Fourier transform can be evaluated by means of an FFT in $\mathcal{O}(Prod(\widetilde{N}) \log Prod(\widetilde{N}))$ operations for $|\ell_q| \leq \widetilde{N}_q/2$.

Finally, given this high-order approximation of $(mu)_\ell$ and the pre-computed $(g_{cmp})_\ell$, we can sum the Fourier series (4.4) also by means of an FFT to obtain a high-order approximation to $K_{cmp}u$.

4.3 Fourier-Smoothed Scatterers

As we have shown, replacing m by the smooth and periodic function $\widetilde{m}$ and enlarging the integration domain to $\Omega_{[\tilde{a},\tilde{b}]}$ is key in obtaining our high-order method. Table 5.11 in Section 5.2 compares the convergence rates for a discontinuous inhomogeneity with and without this substitution. Although we do not present here a complete theoretical analysis of the method, we expect convergence rates similar to those proved for the two-dimensional method (see Theorem 2.7). For example, a discontinuous, piecewise C^1 scatterer is expected to yield second- and third-order convergence on the interior and exterior of the scatterer, respectively; a C^0, piecewise C^2 scatterer is expected to yield third- and fifth-order convergence on the interior and exterior of the scatterer, respectively. Although gains in the asymptotic rate of convergence may always be expected when substituting m by $\widetilde{m}$, real practical gains are most significant for scatterers with a low degree of regularity. (See Section 3.1.1 for a related discussion on the two-dimensional method.) For this reason, one need not typically perform this substitution for very smooth scatterers such as the C^∞ scatterer considered in Figure 5.11 in Section 5.2. In such cases, the trapezoidal rule alone provides high-order accuracy.

Of course when treating discontinuous scatterers, the user is relatively free to choose the smooth cutoff function p_m and the number of modes F in the truncated Fourier approximation of m. There are couple of competing issues that should figure into this decision. In particular, the smaller δ is, the smaller h becomes for a given N. However, this does not necessarily lead to higher accuracy because a small δ implies a relatively steep rise of p_m from 0 to 1. The more abrupt this rise, the smaller we must make h to achieve a given accuracy, thereby increasing N. The choice of F, on the other hand, depends on the choice of N. The value of N must be large enough to resolve the Fourier oscillations in the integrand. In particular, if F is chosen to be too large, the trapezoidal rule will yield very little accuracy in integrating the highly oscillatory modes in m^F. Hence, to obtain high-order convergence, we choose F to be a fixed fraction of N. In our experience, $F = N/2$ is a good

choice.

4.4 Computation of the Fourier Coefficients of the Green's Function

It remains to compute the coefficients $(g_{cmp})_\ell$, which are essentially the Fourier coefficients of g_{cmp}. As defined previously, $g_{cmp}(x) = g(x)p(x)$ where $p(x)$ has support in $\Omega_{[-d,d]}$ for some $d \in \mathbb{R}^3$ with $d_q > 0$, $q = 1, 2, 3$. Note that our choice of d depends on the same issues that affect our choice of δ for the cutoff function p_m as discussed in the previous section.

Of course, there are many possible choices of such partition of unity functions. Partition of unity functions in three dimensions can be constructed from a partition of unity function ϕ in one dimension as described in in [12],

$$\phi(t) = \begin{cases} 1, & \text{for } |t| \le t_0 \\ \exp\left(\frac{2e^{-1/x}}{x-1}\right), & \text{for } t_0 < |t| < t_1, \quad \text{where } x = (|t| - t_0)/(t_1 - t_0) \\ 0, & \text{for } |t| \ge t_1. \end{cases} \tag{4.5}$$

For example, $p(x) = \phi(|x|)$ and $p(x) = \phi(x_1)\phi(x_2)\phi(x_3)$ are both partition of unity functions in three-dimensions that are centered at the origin. Of course, one may shift the center of the functions to any point in $\mathbb{R}^3$ without difficulty.

When computing the coefficients $(g_{cmp})_\ell$, we choose the spherically symmetric function $p(x) = \phi(|x|)$ to simplify the computation and choose $t_1 = R$ such that $R \le d$. Changing to spherical coordinates in the integration gives

$$\begin{aligned} (g_{cmp})_\ell &= \int_{\Omega_{[-d,d]}} g_{cmp}(z) e^{-2\pi i c_\ell \cdot z} dz \\ &= \int_0^R \int_{S^1} \frac{e^{i\kappa\rho}}{4\pi\rho} p(\rho) e^{-2\pi i \rho c_\ell \cdot \hat{z}} \rho^2 d\rho \, d\sigma(\hat{z}) \\ &= \int_0^R g_{cmp}(\rho) j_0(2\pi|c_\ell|\rho)\rho d\rho \\ &= \frac{1}{2\pi|c_\ell|} \int_0^R p(\rho) e^{i\kappa\rho} sin(2\pi|c_\ell|\rho) d\rho, \end{aligned}$$

where $\int_{S^1} d\sigma(\hat{z})$ denotes integration over the unit sphere and the second to last equality follows from [17, p. 32].

We wish to emphasize that the simplification hereby achieved is quite significant. The Jacobian associated with the change to spherical coordinates cancels the ρ^{-1} singularity in Green's function. Furthermore, since the integration on the unit sphere can be performed analytically, we are left only with a one-dimensional integral to be evaluated for various values of $|c_\ell|$.

We rewrite the required integral as follows

$$\begin{aligned}(g_{cmp})_\ell &= \frac{1}{\alpha}\int_0^R p(\rho)e^{i\kappa\rho}sin(\alpha\rho)d\rho \\ &= \frac{1}{2i\alpha}\left[\int_0^R p(\rho)e^{i(\kappa+\alpha)\rho}d\rho - \int_0^R p(\rho)e^{i(\kappa-\alpha)\rho}d\rho\right] \\ &= \frac{1}{2i\alpha}\left\{\mathcal{H}[p](\kappa+\alpha) - \mathcal{H}[p](\kappa-\alpha)\right\}, \end{aligned} \tag{4.6}$$

where $\alpha = 2\pi|c_\ell|$ and

$$\mathcal{H}[p](\omega) = \int_0^R p(\rho)e^{i\omega\rho}d\rho. \tag{4.7}$$

Note that since $p(\rho)$ vanishes for $\rho > R$, $\mathcal{H}[p]$ is related to the Laplace transform, $\mathcal{L}[p]$, as follows

$$\mathcal{H}[p](\omega) = \mathcal{L}[p](-i\omega).$$

Observe that $\mathcal{H}[p](-\omega) = \overline{\mathcal{H}[p](\omega)}$ since $p(\rho)$ is real-valued. It is important to note that we can only use (4.6) to compute $(g_{cmp})_\ell$ when $|\ell| \neq 0$. For $|\ell| = 0$, on the other hand, it is not difficult to see that

$$(g_{cmp})_0 = \int_0^R \rho p(\rho)e^{i\kappa\rho}d\rho = \mathcal{H}[\rho\, p(\rho)](\kappa).$$

Therefore, to compute $(g_{cmp})_\ell$, we need an accurate and efficient method for computing $\mathcal{H}[g](\omega)$ for $g(\rho) = p(\rho)$ and $g(\rho) = \rho p(\rho)$. This problem is not trivial since the value of $\omega \leq \kappa + \alpha$ can be quite large, thus producing a highly oscillatory integrand. Furthermore, straightforward integration by means of the trapezoidal rule will give only first-order accuracy since $p(\rho)$ and $\rho p(\rho)$ cannot be extended as smooth and periodic functions. We are able to compute these integrals accurately and efficiently using a modification of the method suggested in [45, pp. 577–584] as described in Appendix C.

4.5 Implementation

Because of the large memory and CPU-time requirements of realistic problems in three dimensions, an efficient parallel implementation of the numerical method is quite useful. An advantage of the method is its relative simplicity: roughly, it requires only an efficient (parallel) FFT implementation and an effective (parallel) iterative solver for the linear system. We make use of the parallel FFT implementation *fftw* [25, 26] and the parallel iterative solvers in PETSc [4–6]. In addition, PETSc provides excellent vector scatter and gather routines as well as useful I/O routines. These packages make development of an efficient parallel implementation relatively simple.

The bulk of the method lies in computing the required convolutions (see Sections 4.1 and 4.2). As described previously, we approximate the convolution with the smooth part of the Green's function by means the following discrete convolution

$$(K_{smth}u)(x_j) \approx Prod(h) \sum_{k=0}^{N-1} (g_{smth})_{j-k} \widetilde{m}_k u_k,$$

where the components $j_q = 0, \ldots, N_q$, $h_q = (\tilde{b}_q - \tilde{a}_q)/N_q$ for $q = 1, 2, 3$. This discrete convolution requires the values of $(g_{smth})_{j-k}$ for $0 \le j_q \le N_q$ and $0 \le k_q \le N_q - 1$. Hence, $-N_q + 1 \le j_q - k_q \le N_q$. Therefore, computing this convolution by means of FFTs requires three-dimensional arrays with dimensions $2N$, where the array containing $m_k u_k$ is padded with an appropriate number of zeroes [45, pp. 531–537]. More precisely, we first compute

$$(\hat{g}_{smth})_\ell = \sum_{j=0}^{2N-1} (g_{smth})_j \; e^{-2\pi i \ell \cdot (j_1/2N_1, j_2/2N_2, j_3/2N_3)}$$

and

$$\widehat{mu}_\ell = \sum_{j=0}^{2N-1} \widetilde{m}_j u_j \; e^{-2\pi i \ell \cdot (j_1/2N_1, j_2/2N_2, j_3/2N_3)},$$

where $(g_{smth})_j$ is defined by periodic extension for j outside the range $-N_q + 1 \le j_q \le N_q$ and where $m_j u_j = 0$ outside the range $0 \le j_q \le N_q - 1$. We then use these values to compute the discrete convolution

$$\sum_{k=0}^{N-1} (g_{smth})_{j-k} m_k u_k = \sum_{\ell=0}^{2N-1} (\hat{g}_{smth})_\ell \widehat{mu}_\ell e^{2\pi i \ell \cdot (j_1/2N_1, j_2/2N_2, j_3/2N_3)},$$

for $j_q = 0, \ldots, N_q$.

(Note that this straightforward approach requires a factor of $2^3 = 8$ more memory to store these convolution arrays than to store the unknowns u_j, the smoothed inhomogeneity $\widetilde{m}_j$ and the incident field u^i_j. If memory usage becomes the limiting factor, it is possible to break the $m_j u_j$ array into pieces and compute the convolution with each piece separately. This saves memory, but substantially increases CPU-time. Hence, we use the straightforward approach with $2N$-sized arrays.)

On the other hand, the approximation of the convolution with the singular part of the Green's function requires computation of the following sum

$$(K_{cmp}u)(x_j) \approx \sum_{\ell=-M}^{M} (g_{cmp})_\ell \, (mu)_\ell \, e^{2\pi i \ell \cdot (j_1/\widetilde{N}_1, j_2/\widetilde{N}_2, j_3/\widetilde{N}_3)},$$

where $M_q < \widetilde{N}_q/2$, $j_q = 0, \ldots, N_q - 1$ and

$$(mu)_\ell \approx \frac{1}{Prod(\widetilde{N})} \sum_{j=0}^{\widetilde{N}-1} m_j u_j \, e^{-2\pi i \ell \cdot (j_1/\widetilde{N}_1, j_2/\widetilde{N}_2, j_3/\widetilde{N}_3)}.$$

Hence, these sums may also be computed using FFTs. However, in this case, we use FFTs of three-dimensional arrays of size $\widetilde{N}$.

In theory, the FFTs used to compute each of these convolutions need not be related. In practice, however, we save both time and memory by choosing $\widetilde{N} = 2N$. If $\widetilde{N} \neq 2N$, we need to compute the FFT of $(g_{cmp})_\ell \, (mu)_\ell$ and the FFT of $(\hat{g}_{smth})_\ell \, \widehat{mu}_\ell$ separately. Furthermore, we need to store $(g_{cmp})_\ell$, $\ell_q = -M_q, \ldots, M_q$ and $(\hat{g}_{smth})_\ell$, $\ell_q = 0, \ldots, 2N_q - 1$ in two separate arrays with a total of $Prod(\widetilde{N}) + 8Prod(N) > 9Prod(N)$ elements. On the other hand, if $\widetilde{N} = 2N$, our approximation for $(mu)_\ell$ exactly equals $\widehat{mu}_\ell$. Therefore, we need only compute a single FFT of $\hat{g}_\ell \widehat{mu}_\ell$, where $\hat{g}_\ell = (g_{cmp})_\ell + (\hat{g}_{smth})_\ell$. This single array $\hat{g}_\ell$ has size $2N$ and hence a total of $8Prod(N)$ elements.

A further savings that becomes quite significant in a parallel implementation is the communication costs. To compute the convolutions, we must compute FFTs of sizes $\widetilde{N}$ and $2N$ and then we copy a portion of the results into an array of size N. In our parallel code, each of these arrays will, in general, be distributed differently. Hence, copying the results to and from the array of size N involves communication between processors. Therefore, if $\widetilde{N} \neq 2N$, one must first communicate the values of mu into the arrays of size $\widetilde{N}$ and $2N$

for computation of the FFTs, and then communicate the results back from each of these arrays into the original array of size N. On the other hand, if $\widetilde{N} = 2N$, roughly half the communication is required. Hence, because of these memory and communication savings, we choose $\widetilde{N} = 2N$ in our computations.

Once we have implemented the method for computing the matrix-vector product, PETSc implements a wide variety of linear solvers. However, only a few of these linear solvers are appropriate for our linear system, which is non-symmetric and indefinite, namely GMRES, CGS, BiCGSTAB and two different transpose-free QMR methods. Of these, only GMRES and BiCGSTAB perform consistently well. In fact, for each of the other methods, we found an example in which it either rapidly diverged or stagnated. At the same time, the performance of these iterative solvers seems to be somewhat problem dependent: it is certainly possible that there exists a scattering configuration for which BiCGSTAB may perform less well than one of these other approaches. GMRES, on the other hand, performs consistently well, at the price of increased memory requirements and increased computational complexity.

As mentioned in Section 3.2.1, GMRES always requires fewer matrix-vector products to converge to a given residual error than BiCGSTAB. However, at each iteration GMRES stores a new Krylov subspace basis vector whereas BiCGSTAB does not. Hence, in problems which require many iterations, GMRES may rapidly exhaust the system memory. Of course, in such cases, one may restart GMRES after a given number of iterations, thereby limiting the memory used. However, we found that restarted GMRES loses much of its advantage over BiCGSTAB. Therefore, in problems requiring many iterations, BiCGSTAB has become our method of choice.

As in the two-dimensional method, the number of iterations required to achieve a given residual tolerance does not depend on mesh size. However, as the size of the scatterer is increased (as measured in interior wavelengths), the number of required iterations increases. Furthermore, in our numerical experiments, it appears that the number of required iterations increases more for BiCGSTAB than for GMRES. Thus, BiCGSTAB loses its edge over GMRES as the problem size increases.

The numerical examples of Section 5.2 illustrate the versatility, efficiency, and high-order accuracy of the three-dimensional method described in this chapter. Since we have not optimized for parallel performance, we do not present parallel speed-up results. However,

we do include results from parallel computations, from which one can obtain some idea of the parallel performance.