Faster convergence and higher accuracy for the Dirichlet–Neumann map[☆]

Johan Helsing

Numerical Analysis, Centre for Mathematical Sciences,
Lund University, Box 118, SE-221 00 LUND, Sweden

Abstract

New techniques allow for more efficient boundary integral algorithms to compute the Dirichlet–Neumann map for Laplace's equation in two-dimensional exterior domains. Novelties include a new post-processor which reduces the need for discretization points with 50 per cent, a new integral equation which reduces the error for resolved geometries with a factor equal to the system size, systematic use of regularization which reduces the error even further, and adaptive mesh generation based on kernel resolution.

Keywords: Dirichlet–Neumann map, Integral equations, Nyström method, Potential theory, Fast multipole method
2000 MSC: 65D25, 65E05, 65N35, 76D27

1. Introduction

The 1993 GGM paper [3] on solving Laplace's equation and computing the Dirichlet–Neumann map on domains exterior to M closed contours has fueled a rapid development in computational multicomponent fluid flow and multiphase materials science [2, 8, 9, 14]. There has also been some recent basic algorithmic development. The complexity in M of the original scheme has been reduced, the stability for large M has been improved, and contours that lie close to each other can be treated efficiently [5, 6, 7]. This paper

[☆]This work was supported by the Swedish Research Council under contract 621-2007-6234.

Email address: helsing@maths.lth.se (Johan Helsing)
URL: http://www.maths.lth.se/na/staff/helsing/ (Johan Helsing)

presents algorithmic improvements for domains exterior to just a single closed contour. It has consequences for viscous fingering in a Hele–Shaw cell, a problem for which a computational race is going on [2, 9] and where the cost is dominated by the exterior Dirichlet–Neumann solver [2].

2. Mikhlin's integral equation

Let D be a domain exterior to a contour Γ with positive orientation enclosing the origin. To simplify the transition between real and complex notation we shall make no distinction between points in the real plane $\mathbb{R}^2$ and points in the complex plane $\mathbb{C}$. Points in $\mathbb{C}$ will be denoted z or τ. The exterior Dirichlet problem reads: find $U(z)$ such that

$$\Delta U(z) = 0\,, \quad z \in D\,, \tag{1}$$

$$\lim_{D \ni z \to \tau} U(z) = f(\tau)\,, \quad \tau \in \Gamma\,, \tag{2}$$

where $f(\tau)$ is the prescribed Dirichlet data.

In order to write (1,2) as a Fredholm second kind integral equation Mikhlin, see § 31 of [11], suggested the representation

$$U(z) = \frac{1}{2\pi}\int_\Gamma \Im\left\{\frac{\mu(\tau)\,\mathrm{d}\tau}{\tau - z}\right\} + \frac{1}{2\pi}\int_\Gamma \mu(\tau)\,\mathrm{d}|\tau| + a\log|z|\,, \quad z \in D\,, \tag{3}$$

where $\mu(z)$ is a real dipole density and a is a real number.

The problem (1,2) does not have a unique solution. Asymptotic boundary conditions for $U(z)$ at infinity are needed. One can choose a solution with leading behavior $a\log|z|+b$, where either a is prescribed and b is unknown or b is prescribed and a is unknown. We choose to prescribe a, since this seems to have most relevance for the viscous fingering problem. Let the contour have a parameterization $z(t)$, $p < t \le q$, so that $z(p) = z(q)$. Let $z'(t) = \mathrm{d}z(t)/\mathrm{d}t$, $\mu(t) = \mu(z(t))$, and $f(t) = f(z(t))$. Mikhlin's integral equation then reads, see also eq. (18) of [9],

$$\mu(t) - \frac{1}{\pi}\int_p^q \Im\left\{\frac{\mu(s)z'(s)\,\mathrm{d}s}{z(s) - z(t)}\right\} - \frac{1}{\pi}\int_p^q \mu(s)|z'(s)|\,\mathrm{d}s = 2a\log|z(t)| - 2f(t)\,. \tag{4}$$

Upon solving (4) for $\mu(t)$ the Dirichlet–Neumann map can be computed via

$$(n_z \cdot \nabla U)(t) = \frac{\mathrm{d}}{\mathrm{d}\sigma_t}\Im\left\{\frac{1}{2\pi\mathrm{i}}\int_p^q \frac{\mu(s)z'(s)\,\mathrm{d}s}{z(s) - z(t)}\right\} + a\Re\left\{\frac{n_z(t)}{z}\right\}\,, \tag{5}$$

where $\mathrm{d}\sigma_t$ is an infinitesimal element of arc length, $n_z(t) = n_x + \mathrm{i}n_y$ is the outward unit normal of Γ at $z(t)$, and the integral is to be interpreted in the Cauchy principal value sense. See [3] for details.

3. Classic spectrally accurate Nyström schemes

The classic treatment of Mikhlin's integral equation (4) is Nyström discretization with N nodes and weights, t_j and w_j, according to the composite trapezoidal rule [3]. This gives superalgebraic convergence for the approximations to $\mu(t_j)$. The kernel in (4) has a limit for $s \to t$ which can be computed analytically and used for the diagonal entries of the system matrix in the discretization. This is standard and done in [2, 3, 9]. In the discretization of the post-processor (5) one can use the alternate point trapezoidal rule [13] for the Cauchy principal value integral and Fourier approximation and FFT for the differentiation with respect to arc length. This retains superalgebraic convergence and is the choice in [3]. We refer to this combination of methods as the scheme *Classic I*.

Alternatively, one can use partial integration, and rewrite (5) as

$$(n_z \cdot \nabla U)(t) = \Im \left\{ \frac{n_z(t)}{2\pi} \int_p^q \frac{\mu'(s)\,\mathrm{d}s}{z(s) - z(t)} \right\} + a\Re \left\{ \frac{n_z(t)}{z} \right\}, \qquad (6)$$

where $\mu'(t) = \mathrm{d}\mu(t)/\mathrm{d}t$. This is the choice in [9], see their eq. (19). We refer to the combination of the composite trapezoidal rule in Mikhlin's integral equation (4) and FFT differentiation and the alternate point trapezoidal rule in the post-processor (6) as the scheme *Classic II*.

4. A new post-processor

For many reasons we prefer to use composite Gaussian quadrature when solving Fredholm second kind integral equations numerically [6]. Still, for ease of comparison with the classic schemes, the developments of this section will be presented in the composite trapezoidal rule environment.

The first modification has to do with the discretization of the kernel in Mikhlin's equation (4). As an alternative to using limits for the diagonal elements one can use regularization, that is, rewrite the integral operator as

$$\frac{1}{\pi} \int_p^q \Im \left\{ \frac{\mu(s) z'(s)\,\mathrm{d}s}{z(s) - z(t)} \right\} = \mu(t) + \frac{1}{\pi} \int_p^q \Im \left\{ \frac{(\mu(s) - \mu(t)) z'(s)\,\mathrm{d}s}{z(s) - z(t)} \right\} \qquad (7)$$

prior to discretization. A discretization of the right hand side in (7) does not require any limits since the entire integrand vanishes for $s=t$. It is also often more accurate. For example, the action of the discretized regularized integral operator on constant functions $\mu(t)$ will always be numerically exact.

A disadvantage with the post-processor treatment in the classic schemes is that it does not use all available information from the discrete solution $\mu(t_j)$. The alternate point trapezoidal rule disregards every other $\mu(t_j)$ and therefore delays the convergence of (5) and (6). To remedy this, we recommend regularization which, using the definition of the Cauchy principal value, allows us to rewrite (6) as

$$(n_z \cdot \nabla U)(t) = \Im \left\{ \frac{n_z(t)}{2\pi} \int_p^q \frac{\left(\frac{\mu'(s)}{z'(s)} - \frac{\mu'(t)}{z'(t)} \right) z'(s)\, \mathrm{d}s}{z(s) - z(t)} \right\} + a\Re \left\{ \frac{n_z(t)}{z} \right\}. \quad (8)$$

We shall use the standard – not alternate point – composite trapezoidal rule for (8).

A difference between the integrand of the right hand side of (7) and the integrand of (8) is that while the former does not require any limits for $s=t$, the latter does. So, in order to retain spectral accuracy in our new post-processor based on (8) we need the integrand of (8) at $s=t$ with spectral accuracy. That is, we need $\mu'(t_j)$ and also $\mu''(t_j)$ with spectral accuracy. We shall use Nyström differentiation. This technique, analogous in construction to Nyström interpolation, carries the convergence properties of $\mu(t_j)$ over to $\mu'(t_j)$ and $\mu''(t_j)$. See eq. (28) of [6] and Chapter 4.1 of [1] for an error analysis of Nyström interpolation. One gets

$$\mu'(t_j) = 2a\Re \left\{ \frac{z'(t_j)}{z(t_j)} \right\} - 2f'(t_j) + \frac{1}{\pi} \int_p^q \Im \left\{ \frac{z'(t_j)\mu(s)z'(s)\, \mathrm{d}s}{(z(s) - z(t_j))^2} \right\} \quad (9)$$

and

$$\mu''(t_j) = 2a\Re \left\{ \frac{z''(t_j)}{z(t_j)} - \frac{(z'(t_j))^2}{(z(t_j))^2} \right\} - 2f''(t_j) \quad (10)$$

$$+\frac{1}{\pi} \int_p^q \Im \left\{ \left[\frac{z''(t_j)}{(z(s) - z(t_j))^2} + \frac{2(z'(t_j))^2}{(z(s) - z(t_j))^3} \right] \mu(s)z'(s)\, \mathrm{d}s \right\},$$

where we have assumed that $f(t)$ can be differentiated analytically twice with respect to t. In viscous fingering $f(t)$ often involves curvature.

The equations (9,10) need to be discretized, too. We use the composite trapezoidal rule. Here, as in the kernel of (4), computable limits exist for $s = t_j$ but again regularization

$$\frac{1}{\pi}\int_p^q \Im\left\{\frac{z'(t_j)\mu(s)z'(s)\,\mathrm{d}s}{(z(s)-z(t_j))^2}\right\} = \frac{1}{\pi}\int_p^q \Im\left\{\frac{z'(t_j)(\mu(s)-\mu(t_j))z'(s)\,\mathrm{d}s}{(z(s)-z(t_j))^2}\right\} \tag{11}$$

and

$$\frac{1}{\pi}\int_p^q \Im\left\{\left[\frac{z''(t_j)}{(z(s)-z(t_j))^2}+\frac{2(z'(t_j))^2}{(z(s)-z(t_j))^3}\right]\mu(s)z'(s)\,\mathrm{d}s\right\} \tag{12}$$

$$= \frac{1}{\pi}\int_p^q \Im\left\{\left[\frac{z''(t_j)}{(z(s)-z(t_j))^2}+\frac{2(z'(t_j))^2}{(z(s)-z(t_j))^3}\right](\mu(s)-\mu(t_j))z'(s)\,\mathrm{d}s\right\}$$

is an interesting option since the integrands on the right hand sides of (11,12) vanish for $s = t_j$. In Section 7, we shall see how this feedback of accurately computed numerical and analytical derivatives into the algorithm can lead to extreme stability.

5. Differentiated integral equations

When a of (3) is prescribed and when, such as in (6) and (8), only the derivative of $\mu(t)$ is of interest, one can differentiate Mikhlin's equation (4) with respect to t, use partial integration, and arrive at an integral equation for $\mu'(t)$ itself

$$\mu'(t) - \frac{1}{\pi}\int_p^q \Im\left\{\frac{z'(t)\mu'(s)\,\mathrm{d}s}{z(s)-z(t)}\right\} = 2a\Re\left\{\frac{z'(t)}{z(t)}\right\} - 2f'(t)\,. \tag{13}$$

Solving for $\mu'(t)$, rather than for $\mu(t)$, reduces the need for numerical differentiation in the post-processor and the achievable accuracy should be enhanced.

Choosing (8) and given $\mu'(t)$ with spectral accuracy, we only need $\mu''(t)$ with spectral accuracy. Equation (10) can be rewritten

$$\mu''(t_j) = 2a\Re\left\{\frac{z''(t_j)}{z(t_j)} - \frac{(z'(t_j))^2}{(z(t_j))^2}\right\} - 2f''(t_j) \tag{14}$$

$$+\frac{1}{\pi}\int_p^q \Im\left\{\left[\frac{z''(t_j)}{(z(s)-z(t_j))}+\frac{(z'(t_j))^2}{(z(s)-z(t_j))^2}\right]\mu'(s)\,\mathrm{d}s\right\}\,.$$

Here, for $s = t_j$, we only take limits.

Let $v(z)$ be the harmonic conjugate to the part of $U(z)$ which comes from the first term on the right hand side of (3). When a is prescribed, integral equations can be derived both for $v(t)$ and for $v'(t)$. One possibility is

$$v'(t) + \frac{1}{\pi}\int_p^q \Im\left\{\frac{z'(t)v'(s)\,\mathrm{d}s}{z(s)-z(t)}\right\} + \frac{1}{\pi}\int_p^q v'(s)\,\mathrm{d}s = \Re\left\{\frac{z'(t)}{\pi}\int_p^q \frac{(h(s)-h(t))\,z'(s)\,\mathrm{d}s}{z(s)-z(t)}\right\}, \tag{15}$$

where

$$h(t) = \frac{f'(t)}{z'(t)} - \frac{a}{z'(t)}\Re\left\{\frac{z'(t)}{z(t)}\right\}. \tag{16}$$

Having solved (15) for $v'(t)$, the Dirichlet–Neumann map can be computed via the simple post-processor free of numerical differentiation

$$(n_z \cdot \nabla U)(t) = \frac{v'(t)}{|z'(t)|} + a\Re\left\{\frac{n_z(t)}{z(t)}\right\}. \tag{17}$$

6. A regularized implementation of the fast multipole method

We intend to use the GMRES iterative solver [12] for linear systems and the fast multipole method (FMM) [4] for matrix–vector multiplication. We choose an FMM code implemented in MATLAB, which is not so efficient speed wise and memory wise but shares the characteristics of more efficient FMM implementations when it comes to operation count and error propagation.

Our FMM code has two features that are not present in the original FMM scheme [4]. First, our code can compute potential fields due to particles of strengths ρ_j located at points z_j whose contribution to the field at a point z_i is $\rho_j/(z_j - z_i)^n$, where n is a positive integer given as input. This is a minor modification. Second, the code allows for the treatment of regularized kernels in a new and more accurate way. Consider, as an example, the integral

$$\int_p^q \frac{g(t)(\mu(s)-\mu(t))z'(s)\,\mathrm{d}s}{z(s)-z(t)}, \tag{18}$$

where $g(t)$ and $\mu(t)$ are known functions. Discretization gives

$$g(t_i)\mu'(t_i)w_i + g(t_i)\sum_{\substack{j=1\\ j\neq i}}^{N} \frac{(\mu(t_j)-\mu(t_i))z'(t_j)w_j}{z(t_j)-z(t_i)}, \quad i = 1,\ldots,N, \tag{19}$$

or, mathematically equivalent,

$$g(t_i)\mu'(t_i)w_i + g(t_i)\sum_{\substack{j=1\\ j\neq i}}^{N} \frac{\mu(t_j)z'(t_j)w_j}{z(t_j)-z(t_i)} - g(t_i)\mu(t_i)\sum_{\substack{j=1\\ j\neq i}}^{N} \frac{z'(t_j)w_j}{z(t_j)-z(t_i)}\,, \quad i=1,\ldots,N\,. \tag{20}$$

The two sums in (20) can be evaluated for all i with two standard FMM calls. The single sum in (19) is harder to evaluate throughout the FMM as it stands. But the sum in (19) has the advantage over the sums in (20) in that its individual terms do not blow up as $t_j \to t_i$. The effect of accumulated roundoff error in the summation is therefore much smaller. As a compromise between speed and accuracy we implement the following option: single sums such as (19) are treated as two separate sums in parallel in all parts of the FMM except for where the field due to nearest neighbors is computed directly. There, the single sum is used. After all, it is only in the nearest-neighbor interaction that $z(t_j)$ is close to $z(t_i)$. The cost of such a single regularized FMM call is roughly the same as the cost of the two standard FMM calls needed for (20).

We expect regularized FMM to have impact on the discretized equations (8), (11), and (12), where terms in sums could be very large in modulus. We do not expect impact on (4) with (7), where large terms only appear in the real part of the integrand within curly brackets. Here standard FMM will always be used. The imaginary part of the second sum of (20) with $g(t) = 1$, appearing on the diagonal of the system matrix in this context, only has to be computed once and stored prior to starting GMRES.

7. A numerical example

We now present a numerical convergence study performed in MATLAB on a SunBlade 100 workstation. We compare five schemes: *Classic I* and *Classic II* of Section 3 using standard FMM, *New I* being the combination of Mikhlin's integral equation (4) and the post-processor (8) with (9) and (10) using limits and standard FMM, *New II* being the combination of Mikhlin's integral equation (4) with (7) using standard FMM and the post-processor (8) with (9) and (10) using (11), (12) and regularized FMM, and *New III* being the combination of the differentiated integral equation (13) and the post-processor (8) with (14) using limits and standard FMM.

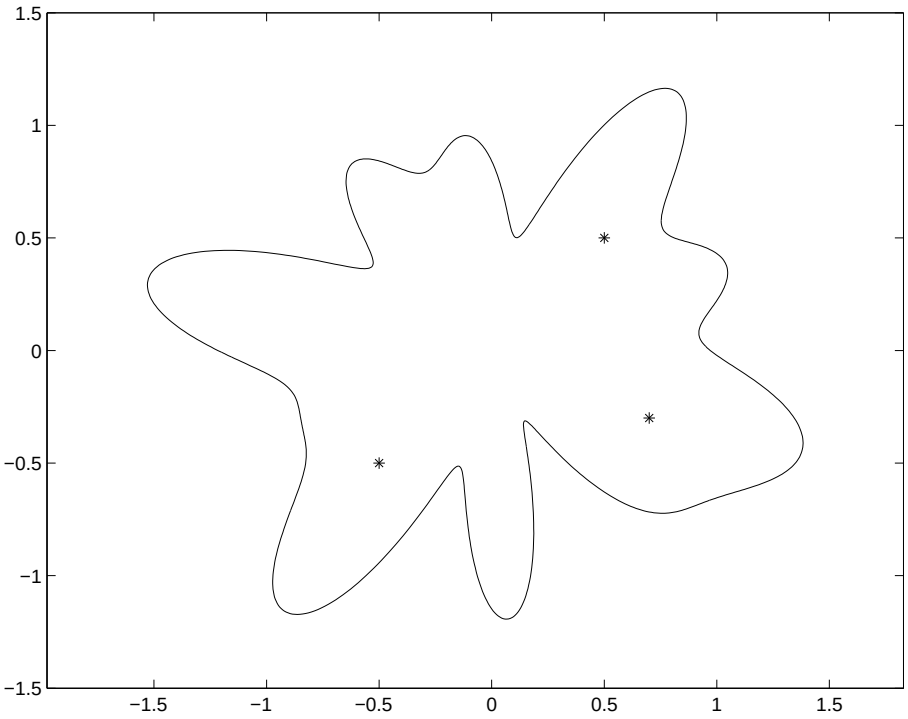

Figure 1: A domain exterior to a butterfly-shaped contour. The stars indicate where the sources and sinks of the reference solution (22) are located.

The domain exterior to the contour of Figure 1 is taken as a test example. The boundary and its parameterization are given by

$$z(t) = \left(1 + \sum_{k=2}^{10} \alpha_k \cos(kt + \beta_k)\right) e^{it}\,, \quad -\pi < t \leq \pi\,, \tag{21}$$

where α_k and β_k are random numbers in $[0, 0.2]$ and $[-\pi, \pi]$. The composite trapezoidal rule then corresponds to equal polar angles between the discretization points. This should be a reasonable choice in this particular example considering the location of boundary parts with high curvature and extra need for resolution. Parameterization in arc length could be preferable for other boundary shapes. The function

$$U_{\text{ref}}(z) = 1 + 1.5 \log|z - z_1| + 1.5 \log|z - z_2| - 2 \log|z - z_3|\,, \quad z \in D\,, \tag{22}$$

where z_1, z_2, and z_3 are marked by stars in Figure 1, is used for Dirichlet data $f(\tau)$ in (2) and also for Neumann data in the error estimate (23) below.

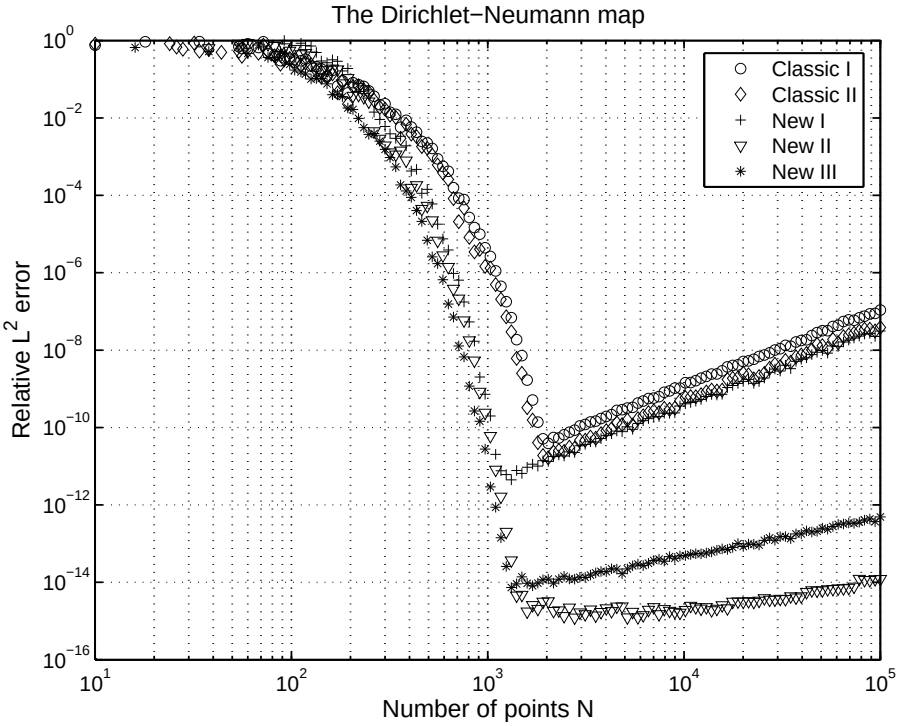

Figure 2: Convergence of the Dirichlet–Neumann map on the boundary of the domain in Figure 1 with Dirichlet data as in (22). The L^2 error (23) is shown as a function of the number of discretization points N for the classic schemes using (5) and (6) and for the new post-processor (8) in combination with both Mikhlin's integral equation (4) and the differentiated integral equation (13).

The coefficient a, which must be prescribed in (4) and (13), is taken as $a=1$ to ensure that the numerical schemes seek $\mu(t)$ compatible with (22).

The linear systems of the five schemes are solved with the GMRES iterative solver including a low-threshold stagnation avoiding technique [6] and a stopping criterion threshold in the relative residual set to machine epsilon ($\epsilon_{\text{mach}} \approx 2 \cdot 10^{-16}$). The precision in the FMM is also set to ϵ_{mach}. The FFT differentiation, needed in *Classic I* and *Classic II*, is carried out with MATLAB's built-in functions `fft` and `ifft`.

Figure 2 shows how the L^2 error

$$E = \left(\frac{\sum_{j=1}^{N} |(n_z \cdot \nabla U_{\text{ref}})(t_j) - (n_z \cdot \nabla U)(t_j)|^2 |z'(t_j)| w_j}{\sum_{j=1}^{N} |(n_z \cdot \nabla U_{\text{ref}})(t_j)|^2 |z'(t_j)| w_j} \right)^{1/2} \tag{23}$$

depends on the number of discretization points N. All five schemes exhibit superalgebraic convergence. But one can see how the classic schemes, which use the alternate point trapezoidal rule in the post-processor, require twice as many discretization points to reach a given relative error as do the new schemes, which use the composite trapezoidal rule throughout.

Figure 2 also shows that overresolution makes the L^2 errors of *Classic I*, *Classic II*, and *New I* grow like $O(N^2)$. This is typical for algorithms involving numerically computed second derivatives (computing $\mu'(t)$ is one derivative and computing the Cauchy principal value has the effect of yet another derivative). The highest achievable accuracy is around 10^{-10} for *Classic I* and *Classic II* and around 10^{-11} for *New I*. Clearly, for geometries that require a finer grid than our "butterfly" to be resolved and when high accuracy is of importance, this asymptotic behavior is not at all good. The schemes *New II* and *New III*, on the other hand, behave better. The L^2 error for *New III* grows like $O(N)$, which is typical for algorithms involving numerically computed first derivatives. The highest achievable accuracy is around 10^{-14}. The L^2 error of *New II* is even smaller. The minor jumps in the error that are visible for $N = 10804$, $N = 17714$, $N = 42088$, and $N = 78094$ correspond to system sizes where FMM decides to introduce yet one level of refinement and where, consequently, the regularized FMM uses less direct evaluation of single sums. See Section 6.

A few words about the computational costs of the five schemes in this example are in order. GMRES typically reaches its stopping criterion threshold for the discretized integral equations (4) and (13) in 35 iterations. This requires 35 standard FMM calls if limits are used and 36 standard FMM calls if (7) is used. To this should be added FMM and FFT differentiation calls needed for the post-processor. The total number of calls for the Dirichlet–Neumann solver is then 36 standard FMM calls and one FFT differentiation call for the *Classic I* and *Classic II* schemes, 39 standard FMM calls for the *New I* scheme, 36 standard FMM calls and three regularized FMM calls for the *New II* scheme, and 38 standard FMM calls for the *New III* scheme.

Naturally, other interesting combinations of techniques than the schemes of this section are possible. For example, the *Classic II* scheme can be improved with the regularization (7) in (4). FFT differentiation can be used instead of Nyström differentiation in the *New I* scheme to gain speed. Equation (15) with (17) can improve on accuracy for underresolved systems. Nevertheless, as a crude conclusion we can say that switching from a classic scheme to a new scheme in this example requires at most ten per cent more

FMM calls for a given accuracy, but only half the system size. The net savings in total computational time could be around 45 per cent.

8. A priori error estimates and adaptive mesh generation

When computing, one often wants a particular accuracy, perhaps the best possible, in the final answer at a low cost. The number of discretization points N needed and their placement should be known in advance. This section addresses such issues of a priori error estimates and adaptive mesh generation. We shall abandon the composite trapezoidal rule in favor of composite Gauss–Legendre quadrature based on n panels with 16 quadrature points each. Composite 16-point Gauss–Legendre quadrature on a uniform mesh may be 25 to 50 per cent more expensive than the trapezoidal rule, depending on the required accuracy. Its advantage, however, is that it lends itself better to mesh adaptivity. While the purpose of mesh adaptivity is to minimize the computational cost, a pleasant side effect is that the particular choice of boundary parameter becomes less important. Since the mesh is refined only where it is needed, the effects of sub-optimal parameterizations are fully counterbalanced. One is free to choose a parameterization which suites a given geometry from a modeling point of view. Given a specified resolution *tol*, we construct an ad hoc algorithm which finds an adaptive panel-mesh and a grid of discretization points which meets this resolution and link it to the overall accuracy. For brevity we only consider (13) in combination with (8) and (14).

Our basic idea is the following: let a well-conditioned integral equation of Fredholm's second kind with an integral operator K, an unknown density μ, and a right hand side f be discretized on a grid on a coarse panel-mesh

$$(\mathbf{I} + \mathbf{K})\boldsymbol{\mu} = \mathbf{f}\,, \tag{24}$$

where the kernel is not fully resolved by the underlying quadrature, and on a grid on a fine panel-mesh

$$(\mathbf{I}_{\text{fin}} + \mathbf{K}_{\text{fin}})\boldsymbol{\mu}_{\text{fin}} = \mathbf{f}_{\text{fin}}\,, \tag{25}$$

where it is resolved. Define the error in the coarse discretization due to insufficient quadrature resolution as

$$E_{\text{resK}} = ||\mathbf{K}\mathbf{W}^{-1}\mathbf{P}^T\mathbf{W}_{\text{fin}} - \mathbf{Q}\mathbf{K}_{\text{fin}}||_\infty\,, \tag{26}$$

where $\mathbf{W}$ and $\mathbf{W}_{\text{fin}}$ are diagonal matrices containing the quadrature weights of the two discretizations, $\mathbf{P}$ is a discretization of a prolongation operator that performs piecewise polynomial interpolation in parameter from discretization points on coarse panels to points on fine panels, and $\mathbf{Q}$ is a discretization of a restriction operator that performs piecewise polynomial interpolation in the other direction. We shall refine the coarse mesh just as much that is needed to make E_{resK} meet *tol*. The relative error in $\boldsymbol{\mu}$ is then approximately bounded by

$$\frac{||\boldsymbol{\mu} - \boldsymbol{\mu}_{\text{fin}}||}{||\boldsymbol{\mu}_{\text{fin}}||} \leq tol\, ||(\mathbf{I} + \mathbf{K})^{-1}||\,, \tag{27}$$

see Section 4.5 of Ref. [10]. Rather than working with the entire matrix $\mathbf{K}$, we shall work with select submatrices of a fixed size. We shall refine the mesh until these submatrices, individually, resolve their parts of the kernel. It is assumed that the right hand side f, by then, also is resolved.

We shall now be more precise. Let $\mathbf{K}$ be the $16n \times 16n$ matrix corresponding to the discretized integral operator of (13) on an n-panel-mesh. Let Γ_i and Γ_j, $i \neq j = 1, \ldots, n$, be two panels on Γ with arc lengths l_i and l_j. Let $\mathbf{K}^{i,j}$ be the 16×16 submatrix of $\mathbf{K}$ with source points on Γ_j and target points on Γ_i. Let $\mathbf{A}^{i,j}$ be the 32×16 submatrix containing the blocks $\mathbf{K}^{i,j}$ and $\mathbf{K}^{j,j}$

$$\mathbf{A}^{i,j} = \begin{bmatrix} \mathbf{K}^{i,j} \\ \mathbf{K}^{j,j} \end{bmatrix}. \tag{28}$$

We shall sweep j from 1 to n and for each j make sure that $\mathbf{A}^{i,j}$ and $\mathbf{A}^{j,i}$ resolve their parts of the kernel for all i belonging to panels Γ_i separated less than a distance $2 \cdot \max(l_i, l_j)$ from Γ_j. If, for an i in this set, $\mathbf{A}^{i,j}$ does not resolve its part of the kernel, then Γ_j is subdivided. If $\mathbf{A}^{j,i}$ does not resolve its part of the kernel, then Γ_i is subdivided. If neither $\mathbf{A}^{i,j}$ nor $\mathbf{A}^{j,i}$ do resolve their parts of the kernel, then the source-panel corresponding to the submatrix that resolve its part of the kernel least well is subdivided. After subdivision, n is increased by one. The work can be organized as to avoid repeated computation of identical quantities.

It remains to modify the resolution estimator (26) so that it applies to submatrices $\mathbf{A}^{i,j}$ and link it to the overall specified tolerance. For each row in the matrix of (26), the major contributions to the ∞-norm come only from a few submatrices. Considering the approximate nature of the whole approach we simply use

$$E^{i,j}_{\text{resA}} = ||\mathbf{A}^{i,j}(\mathbf{W}^j)^{-1}\mathbf{P}^T\mathbf{W}^j_{\text{fin}} - \mathbf{Q}\mathbf{A}^{i,j}_{\text{fin}}||_\infty\,. \tag{29}$$

Here the 16×16 diagonal matrix $\mathbf{W}^j$ contains the quadrature weights associated with 16-point Gauss–Legendre quadrature on Γ_j. As an approximation to the fully resolved kernel part $\mathbf{A}^{i,j}_{\text{fin}}$ we choose a 48×32 discretization where Γ_j has been subdivided into two sub-panels Γ_{ja} and Γ_{jb} equisized in parameter

$$\mathbf{A}^{i,j}_{\text{fin}} = \begin{bmatrix} \mathbf{K}^{i,ja} & \mathbf{K}^{i,jb} \\ \mathbf{K}^{ja,ja} & \mathbf{K}^{ja,jb} \\ \mathbf{K}^{jb,ja} & \mathbf{K}^{jb,jb} \end{bmatrix} . \tag{30}$$

The 32×32 diagonal matrix $\mathbf{W}^j_{\text{fin}}$ in (29) contains the quadrature weights on Γ_{ja} and Γ_{jb}. The 32×16 matrix $\mathbf{P}$ performs 15th degree polynomial interpolation in parameter from points on Γ_j to points on Γ_{ja} and Γ_{jb}. The upper left 16×16 block of the sparse 32×48 matrix $\mathbf{Q}$ is the identity matrix and its sparse lower right 16×32 block performs piecewise 15th degree polynomial interpolation in parameter from points on Γ_{ja} and Γ_{jb} to points on Γ_j. Note that $\mathbf{P}$ and $\mathbf{Q}$ are independent of j and that the application of $\mathbf{P}^T$ to the left can be sped up via a sparse factorization, see Section 5.1 of Ref. [6].

The resolution estimator (29) measures how well a part of the kernel is resolved on a grid by interpolation in the variable of integration by some of the basis functions that underly the quadrature. Our interpolating polynomials have degree 15. Gaussian quadrature has polynomial degree 31, that is, twice as much. Therefore the resolution estimator is more related to the square root of the resolution sought than to the resolution itself and we take

$$E^{i,j}_{\text{resA}} < \sqrt{tol}\,, \tag{31}$$

as our criterion for when $\mathbf{A}^{i,j}$ does resolve its part of the kernel to the specified resolution.

Figure 3 shows that our simple algorithm has a remarkable ability to predict the relative error in the Dirichlet–Neumann map for values of *tol* down to 10^{-14}, which is the highest accuracy achieved for *New III* in Section 7. The specified *tol* differs from the actual error at most with a factor of ten. The scheme is economical, too. It consistently outperforms the five schemes of Section 7 in terms of high accuracy with few discretization points. The number of points needed to get a given relative accuracy is reduced by over 60 per cent compared to the classic schemes. But the best advantage with mesh adaptivity is that one does not have to worry about what N gives the highest accuracy. One can simply set $tol = \epsilon_{\text{mach}}$ and compute once.

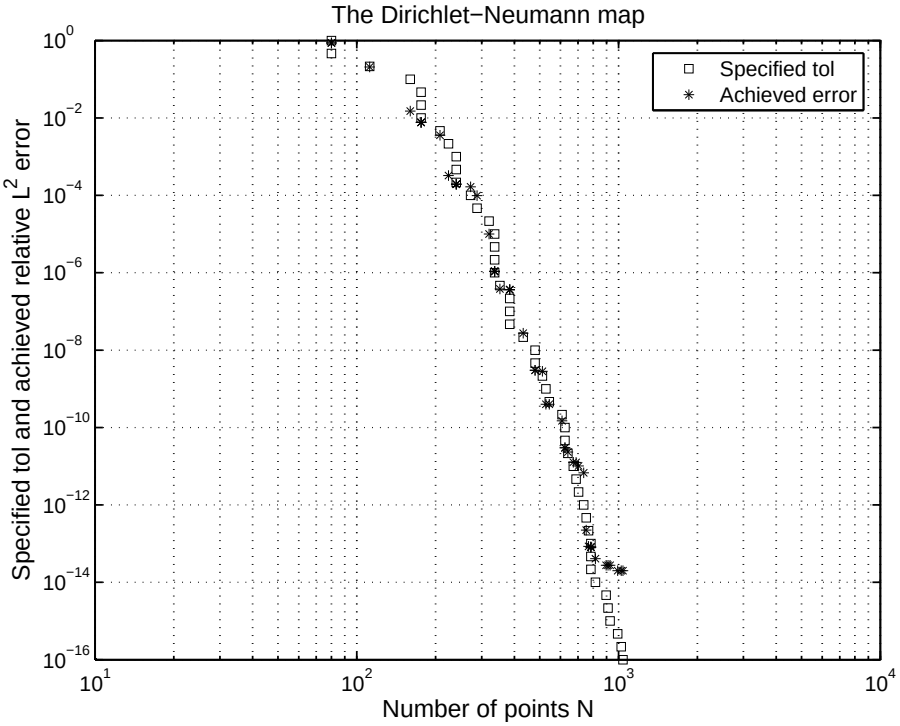

Figure 3: Given $tol \in [10^{-16}, 1]$, a grid is adaptively determined and the Dirichlet–Neumann map is computed via (13) and (8) along with the error (23). Several values of tol could give the same grid. For example, both $tol = 10^{-8}$ and $tol = 4.6 \cdot 10^{-9}$ give a grid with $N = 480$ points and an achieved error of $3 \cdot 10^{-9}$. The problem is the same as in Figure 2 and the presentation of data is done as to facilitate comparison. But note that N is variable in Figure 2, while the specified tol is variable in this figure.

9. Conclusions

Boundary integral algorithms for viscous fingering simulation are seen as relatively mature and their computational complexity may not have improved since 1994, see Ref. [2] and references therein. Still, this paper shows that there is plenty of room for improvement when it comes to computational economy and achievable accuracy. That is, in the part of the simulations that involve the Dirichlet–Neumann solver, which currently is the dominating cost. We have modified the integral equation itself and the post-processor and introduced adaptive mesh generation. This, everything else held constant, should open up for algorithms that are at least twice as fast and an order of magnitude more accurate.

References

[1] K. E. Atkinson, *The numerical Solution of Integral Equations of the Second Kind*, Cambridge University Press, Cambridge, 1997.

[2] P. Fast and M.J. Shelley, *Moore's law and the Saffman-Taylor instability*, J. Comput. Phys., **212**(1), pp. 1–5 (2006), doi:10.1016/j.jcp.2005.06.022

[3] A. Greenbaum, L. Greengard, and G.B. McFadden, Laplace's equation and the Dirichlet–Neumann map in multiply connected domains, J. Comput. Phys. **105**(2), pp. 267–278 (1993), doi:10.1006/jcph.1993.1073

[4] L. Greengard and V. Rokhlin, A fast algorithm for particle simulations, J. Comput. Phys. **73**(2), pp. 325–348 (1987), doi:10.1016/0021-9991(87)90140-9

[5] J. Helsing, *Approximate inverse preconditioners for some large dense random electrostatic interaction matrices*, BIT Numer. Math., **46**(2), pp. 307–323 (2006), doi:10.1007/s10543-006-0057-0

[6] J. Helsing and R. Ojala, On the evaluation of layer potentials close to their sources, J. Comput. Phys. **227**(5), pp. 2899–2921 (2008), doi:10.1016/j.jcp.2007.11.024

[7] J. Helsing and E. Wadbro, Laplace's equation and the Dirichlet-Neumann map: a new mode for Mikhlin's method, J. Comput. Phys. **202**(2), pp. 391–410 (2005), doi:10.1016/j.jcp.2004.06.024

[8] T.Y. Hou, J.S. Lowengrub, and M.J. Shelley, *Boundary Integral Methods for Multicomponent Fluids and Multiphase Materials*, J. Comput. Phys., **169**(2), pp. 302–362 (2001), doi:10.1006/jcph.2000.6626

[9] S. Li, J.S. Lowengrub, and P.H. Leo, *A rescaling scheme with application to the long-time simulation of viscous fingering in a Hele–Shaw cell*, J. Comput. Phys., **225**(1), pp. 554–567 (2007), doi:10.1016/j.jcp.2006.12.023

[10] P.G. Martinsson and V. Rokhlin, A fast direct solver for boundary integral equations in two dimensions, J. Comput. Phys. **205**(1), pp. 1–23 (2005), doi:10.1016/j.jcp.2004.10.033

[11] S.G. Mikhlin, *Integral Equations and their applications to certain problems in mechanics, mathematical physics and technology, 2nd ed.*, Pergamon Press, London, 1964.

[12] Y. Saad and M.H. Schultz, GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM J. Sci. Stat. Comp. **7**(3), pp. 856–869 (1986), doi: 10.1137/0907058

[13] A. Sidi and M. Israeli, *Quadrature methods for periodic singular and weakly singular Fredholm integral equations*, J. Sci. Comput., **3**(2), pp. 201–231 (1988), doi:10.1007/BF01061258

[14] K. Thornton, N. Akaiwa, and P.W. Voorhees, Large-scale simulations of Ostwald ripening in elastically stressed solids: I. Development of microstructure, Acta Mater., **52**(5), pp. 1353–1364 (2004), doi:10.1016/j.actamat.2003.11.037