A robust and accurate solver of Laplace's equation with general boundary conditions on general domains in the plane

Ojala, Rikard

Published in:
Journal of Computational Mathematics

DOI:
10.4208/jcm.1201-m3644

2012

A robust and accurate solver of Laplace's equation with general boundary conditions on general domains in the plane

Rikard Ojala
Numerical Analysis, Centre for Mathematical Sciences,
Lund University, Box 118, SE-221 00 Lund, Sweden
Email: rikardo@maths.lth.se

Abstract

A robust and general solver for Laplace's equation on the interior of a simply connected domain in the plane is described and tested. The solver handles general piecewise smooth domains and Dirichlet, Neumann, and Robin boundary conditions. It is based on an integral equation formulation of the problem. Difficulties due to changes in boundary conditions and corners, cusps, or other examples of non-smoothness of the boundary are handled using a recent technique called recursive compressed inverse preconditioning. The result is a rapid and very accurate solver which is general in scope, its performance is demonstrated via some challenging numerical tests.

Mathematics subject classification: 65R20, 65E05.
Key words: Laplace's equation, Integral equations, Mixed boundary conditions, Robin boundary conditions

1. Introduction

This paper presents techniques for computing highly accurate solutions to Laplace's equation on simply connected domains in the plane using an integral equation formulation of the problem. The solver is capable of handling Dirichlet, Neumann, and Robin boundary conditions on fairly general domains with piecewise smooth boundaries. While such problems can be treated using, for example, the finite element method, there is much to gain by their treatment in an integral equation setting, for example robustness, high accuracy, speed and the dimensionality reduction.

This paper is in a sense a continuation of the paper [9] by Helsing in which Laplace's equation was solved on simply connected smooth domains in the plane with mixed Dirichlet and Neumann boundary conditions. With homogeneous Dirichlet or Neumann boundary conditions, and for smooth domains and smooth boundary data, the sought single or double layer density is itself smooth, but this is no longer true for mixed boundary conditions, even for smooth domains, since singularities or other non-polynomial-like behaviour tends to plague the density near the points where the boundary conditions change. In [9], these difficulties were resolved with very good results using recursive compressed inverse preconditioning first introduced in [12].

In applications, however, one is very likely to encounter domains which are only piecewise smooth, for example a domain may have corners. As is the case with the change in boundary conditions, corners also tend to introduce non-polynomial like behaviour of the layer density. The methods of Bremer and Rokhlin [4] and Bremer, Rokhlin and Sammis [5] produce impressive results for such problems with homogeneous boundary conditions, but as a further difficulty, it is not uncommon for boundary conditions to change at these corners. This paper will address

such domain/boundary condition configurations, and in addition, we will also implement Robin, or third kind boundary conditions. The perhaps most well known application of such boundary conditions is Newton's law of cooling in heat transfer, but there are others in a variety of fields, for example robotic motion [7], chemical vapor deposition modeling [16], and modeling of macromolecular transport in medicine [3].

While the high accuracy property of integral equation solvers is not essential for many practical applications, it is always good to know that the capability of high accuracy is available, and if not needed one may trade some of the accuracy for speed. One might argue that an accuracy of twelve digits in the solution, an accuracy which is well within the reach of the methods of this paper, is a bit over the top in many cases, but for some problems, due to ill-posedness, the end result may be meaningless unless the underlying solver is accurate enough. As an example, when reconstructing harmonic functions from Cauchy boundary data [10], one can easily lose ten digits of accuracy even on smooth domains, demanding very accurate underlying solvers. Such problems, but with corners, can potentially be addressed using the methods of this paper.

The paper is organized as follows. Section 2 introduces the integral equation formulation for the problem, and this integral equation is discretized in Section 3. In Section 4 we discuss the post-processor and in Section 5 the performance of the method in terms of accuracy, robustness and speed is illustrated via some challenging numerical examples. Finally, in Section 6 we discuss possible improvements and future work.

2. The integral equation

In the following we will, for convenience and to make notation short, make no distinction between points or vectors in the real plane $\mathbb{R}^2$ and points in the complex plane $\mathbb{C}$. Consider a simply connected domain Ω, with piecewise smooth boundary Γ. Let Γ be comprised of three disjoint boundary segments, Γ_{D}, Γ_{N}, Γ_{R}, describing Dirichlet, Neumann, and Robin boundary conditions respectively. We seek a function $U(z)$, harmonic in Ω, that satisfies :

$$\lim_{\Omega \ni \tau \to z} U(\tau) = f_{\mathrm{D}}(z), \qquad z \in \Gamma_{\mathrm{D}} \tag{2.1}$$

$$\lim_{\Omega \ni \tau \to z} \frac{\partial U}{\partial n}(\tau) = f_{\mathrm{N}}(z), \qquad z \in \Gamma_{\mathrm{N}} \tag{2.2}$$

$$\lim_{\Omega \ni \tau \to z} \frac{\partial U}{\partial n}(\tau) + \alpha(z) U(\tau) = f_{\mathrm{R}}(z), \qquad z \in \Gamma_{\mathrm{R}}, \tag{2.3}$$

where $\partial/\partial n$ denotes the derivative with respect to the outward normal, and the bounded functions $f_{\mathrm{D}}(z)$, $f_{\mathrm{N}}(z)$, and $f_{\mathrm{R}}(z)$ is the Dirichlet, Neumann, and Robin data on the corresponding part of the boundary. The parameter $\alpha(z) \geq 0$ is a bounded, real valued function on Γ_{R}.

Along the lines of [9], we now let the solution $U(z), z \in \Omega \cup \Gamma_{\mathrm{N}} \cup \Gamma_{\mathrm{R}}$ be represented by a real density $\mu(z), z \in \Gamma$,

$$U(z) = \frac{1}{\pi} \int_{\Gamma_{\mathrm{D}}} \mu(\tau) \Im \left\{ \frac{\mathrm{d}\tau}{\tau - z} \right\} - \frac{1}{\pi} \int_{\Gamma_{\mathrm{N}} \cup \Gamma_{\mathrm{R}}} \mu(\tau) \log |\tau - z| \mathrm{d}|\tau|, \qquad z \in \Omega \cup \Gamma_{\mathrm{N}} \cup \Gamma_{\mathrm{R}}. \tag{2.4}$$

In order to simplify notation we define the following integral operators:

$$K_{\mathrm{Cau}}\mu(z) = \frac{1}{\pi}\int_{\Gamma_{\mathrm{D}}} \mu(\tau)\Im\left\{\frac{\mathrm{d}\tau}{\tau - z}\right\} \tag{2.5}$$

$$K_{\mathrm{dCau}}\mu(z) = \frac{1}{\pi}\int_{\Gamma_{\mathrm{D}}} \mu(\tau)\Im\left\{\frac{n_z \mathrm{d}\tau}{(\tau - z)^2}\right\} \tag{2.6}$$

$$K_{\mathrm{Log}}\mu(z) = \frac{1}{\pi}\int_{\Gamma_{\mathrm{N}}\cup\Gamma_{\mathrm{R}}} \mu(\tau)\log|\tau - z|\mathrm{d}|\tau| \tag{2.7}$$

$$K_{\mathrm{dLog}}\mu(z) = \frac{1}{\pi}\int_{\Gamma_{\mathrm{N}}\cup\Gamma_{\mathrm{R}}} \mu(\tau)\Im\left\{\frac{n_z}{n_\tau}\frac{\mathrm{d}\tau}{\tau - z}\right\} \tag{2.8}$$

where n_z and n_τ are the outward unit normals at z and τ. Note that $K_{\mathrm{dCau}}\mu(z)$ and $K_{\mathrm{dLog}}\mu(z)$ are the normal derivatives of $K_{\mathrm{Cau}}\mu(z)$ and $K_{\mathrm{Log}}\mu(z)$, respectively. Combining (2.4) with (2.1-2.3) and using the above notation gives the system

$$(I + K_{\mathrm{Cau}} - K_{\mathrm{Log}})\,\mu(z) = f_{\mathrm{D}}(z), \qquad z \in \Gamma_{\mathrm{D}}, \tag{2.9}$$

$$(I + K_{\mathrm{dCau}} + K_{\mathrm{dLog}})\,\mu(z) = f_{\mathrm{N}}(z), \qquad z \in \Gamma_{\mathrm{N}}, \tag{2.10}$$

$$(I + \alpha(z)K_{\mathrm{Cau}} + K_{\mathrm{dCau}} - \alpha(z)K_{\mathrm{Log}} + K_{\mathrm{dLog}})\,\mu(z) = f_{\mathrm{R}}(z), \qquad z \in \Gamma_{\mathrm{R}}, \tag{2.11}$$

for the density $\mu(z)$, where I is the identity operator. The question of existence and uniqueness of solutions to this system is difficult, and seems to be open. For homogeneous Dirichlet and Neumann boundary conditions on Lipschitz domains good results exist, however. See [15] and [6].

We define a point on the boundary as being a *singular point* if either or both of the following holds:

- the boundary conditions change at the point
- the boundary is non-smooth at the point, that is, the point is located on a corner, cusp, spline-knot, or at another type of boundary irregularity.

We call a part of the boundary between two singular points a *boundary component*. A boundary component is well-behaved in the sense that it is smooth and the boundary conditions on it does not change.

When dealing with a system of integral equations it is of paramount importance that the system is of Fredholm's second kind with compact operators. This often ensures well-conditioning and a clustered spectrum which in turn opens for high accuracy and efficient iterative linear equation solvers. The system (2.9,2.10,2.11) is of Fredholm's second kind, but the integral operators are only compact if we remove a neighborhood around the singular points, that is, set the kernels to zero around the singular points. How to retain the favourable properties associated with the compactness of the operators in the presence of singular points will be described below.

3. Discretization

The system (2.9,2.10,2.11) will be discretized using Nyström discretization with composite 16-point Gauss–Legendre quadrature as the basic quadrature. Let the boundary Γ be given positive orientation and let $\tau(t), t_0 \le t \le t_1$ be a parametrization of Γ. Place a *mesh* consisting of n_{pan} quadrature panels on Γ such that:

- singular points are located at break points between panels,
- the panels have approximately the same length,
- the number of panels on any boundary component should be at least four.

The motivation behind these requirements will become clear below.

When the mesh is constructed we set up a *grid* of $16n_{\text{pan}}$ Gauss–Legendre nodes, t_k, and weights, w_k, associated with integration in t. The parametrized and discretized quantities will be denoted:

- $\tau_k = \tau(t_k), \tau'_k = \tau'(t_k)$, the boundary, and its derivative with respect to parameter.
- $\mu_k = \mu(\tau_k)$, the layer density.
- $f_k = f(\tau_k)$, the boundary data.

Note that since all singular boundary points are located at break points between panels, there will be no difficulties in terms of non-uniqueness defining the derivatives with respect to parameter.

3.1. Basic and specialized quadratures

Several different integral operators will be discretized, and we will try to keep the treatment short. For details and alternatives see Section 2 of [9] and Sections 4 to 6 of [11]. We first note that if, on some connected subset Γ_{p} of Γ, the integral kernel $K(\tau, z)$ and density $\mu(\tau)$ are smooth and a discretization with N_p points is enough for them to be well-resolved, then the basic quadrature

$$\int_{\Gamma_{\text{p}}} \mu(\tau)K(\tau, z)\mathrm{d}\tau = \int_{t_\alpha}^{t_\beta} \mu(\tau(t))K(\tau(t), z)\tau'(t)\mathrm{d}t \approx \sum_{k=1}^{N_p} \mu_k K(\tau_k, z)\tau'_k w_k \tag{3.1}$$

is accurate. For example this is true whenever $z \in \Gamma_{\text{p}}$ and when K is the kernel of K_{Cau} or K_{dLog} since these kernels are smooth, they both have limits for the case $\tau_k = z$. Another example is when z is sufficiently far away from Γ_{p}. There are situations in our context when the accuracy of (3.1) suffers, however. These are

- when panels meet at singular points,
- when panels are close in space but not in parameter,
- when K is the kernel of K_{Log}, in which case K is not smooth.

These cases are illustrated in Fig. 3.1. In order to ensure high accuracy we need to employ special quadrature techniques, these are derived in a similar way for each kernel type and will now be discussed. We will use product integration quadrature of interpolatory type, borrowing terminology from [13], that is we compute quantities of the form

$$p_k = \int_{\Gamma_{\text{p}}} \tau^k K(\tau, z)\mathrm{d}\tau, \qquad k = 0, 1, ...15, \tag{3.2}$$

where K is some kernel, and recover the new quadrature weights $\widetilde{w}_k$ from the solution to a Vandermonde system with the p_k's as the right hand side. See Section 5 in [11]. These weights

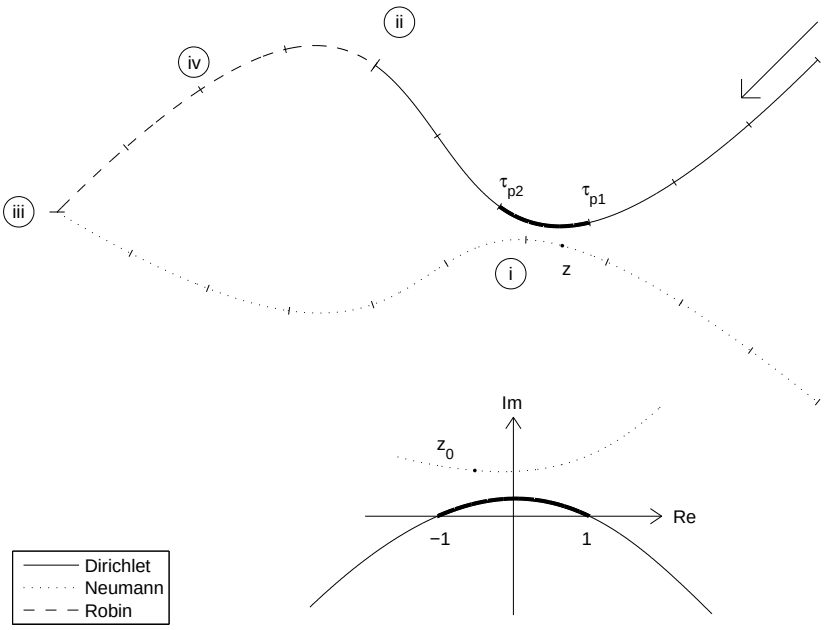

Fig. 3.1. Part of a boundary demonstrating the cases where basic quadrature is likely to fail: (i), where the boundary falls in on itself; (ii), where the boundary conditions change; (iii), at corners; and (iv), on boundaries with Robin boundary conditions, because of the need to evaluate the nonsmooth kernel of K_{Log} for points on the boundary. The figure also demonstrates the translation and scaling of a panel between τ_{p_1} and τ_{p_2}, and a target point z on the boundary to set up for specialized quadrature computation. The arrow describes the orientation of the parametrization.

will be computed on a panel-to-panel basis so in the following we let Γ_{p} consist of one panel on the boundary, and we assume that the density $\mu(z)$ is reasonably smooth on Γ_{p}. In order to facilitate simple analytical expressions for the p_k's we rotate, scale and translate the panel so that its original endpoints τ_{p_1} and τ_{p_2} end up at -1 and 1, respectively. See Fig. 3.1. This does not affect the kernels of K_{Cau} and K_{dLog}, since they are scale and translation invariant, but the kernels of K_{Log} and K_{dCau} are not scale invariant so a scaling factor $\gamma = (\tau_{p_2} - \tau_{p_1})/2$ is required. We denote the scaled and translated panel as $\Gamma_{\mathrm{p}'}$, and the target point z is scaled and translated into z_0.

- **Discretization of K_{Cau} and K_{dLog}**
  In the case of
  $$K_{\mathrm{Cau}}\mu(z) = \frac{1}{\pi}\int_{\Gamma_{\mathrm{p}}} \mu(\tau)\Im\left\{\frac{\mathrm{d}\tau}{\tau - z}\right\}, \tag{3.3}$$
  we will compute
  $$p_k = \int_{\Gamma_{\mathrm{p}'}} \frac{\tau^k}{\tau - z_0}\mathrm{d}\tau, \qquad k = 0, 1, ...15. \tag{3.4}$$
  These quantities can be computed recursively by
  $$p_k = z_0 p_{k-1} + \frac{1 - (-1)^k}{k}, \qquad k = 1, 2, ..., 15 \tag{3.5}$$

with $p_0 = \log(1-z_0) - \log(-1-z_0)$. The weights $\widetilde{w}_k$ are obtained from the solution to the Vandermonde system by dividing by π and taking the imaginary part. For K_{dLog} we use the same procedure, but instead of dividing by π we multiply the solution by $n_z/(\pi n_{\tau_i})$ element-wise, where $n_{\tau_i}, i = 1, 2, ..., 16$ is the outward normal at the 16 points on the panel, before taking the imaginary part.

- **Discretization of K_{dCau}**
  The kernel of the hypersingular operator

$$K_{\text{dCau}}\mu(z) = \frac{1}{\pi}\int_{\Gamma_{\text{p}}} \mu(\tau)\Im\left\{\frac{n_z \mathrm{d}\tau}{(\tau - z)^2}\right\} \tag{3.6}$$

  does not have a limit for $\tau = z$ and thus is not smooth, but this does not pose a problem since (3.6) is never evaluated for $z \in \Gamma_{\text{p}}$. The quantities

$$q_k = \int_{\Gamma_{\text{p}'}} \frac{\tau^k \mathrm{d}\tau}{(\tau - z_0)^2}, \qquad k = 0, 1, ...15, \tag{3.7}$$

  are given by

$$q_k = z_0 q_{k-1} + p_{k-1}, \qquad k = 1, 2, ..., 15 \tag{3.8}$$

  with $q_0 = -1/(1+z_0) - 1/(1-z_0)$, and with p_k as in (3.4). The resulting solution to the Vandermonde system is multiplied by $\frac{n_z}{\gamma\pi}$ and the imaginary part taken.

- **Discretization of K_{Log}**
  The logarithmic integral operator

$$K_{\text{Log}}\mu(z) = \frac{1}{\pi}\int_{\Gamma_{\text{p}}} \mu(\tau)\log|\tau - z|\mathrm{d}|\tau| \tag{3.9}$$

  is in some ways the most troublesome one of the lot. Not in terms of general difficulty, but since the kernel is non-smooth and needs to be evaluated for $z \in \Gamma_{\text{p}}$. The above technique works equally well but the number of target points z that requires specialized quadrature is substantially higher, at least if a large portion of the boundary has Robin boundary conditions, since for each panel on Γ_{R} we need to invoke this procedure for each of the 16 discretization points on the panel, as well as for a couple on the neighboring panels. Noticing that

$$\frac{1}{\pi}\int_{\Gamma_{\text{p}}} \mu(\tau)\log|\tau - z|\mathrm{d}|\tau| = \frac{1}{\pi}\Im\left\{\int_{\Gamma_{\text{p}}} \frac{\mu(\tau)}{n_\tau}\log(\tau - z)\mathrm{d}\tau\right\} \tag{3.10}$$

  we compute the quantities

$$r_k = \int_{\Gamma_{\text{p}'}} \tau^k \log(\tau - z_0)\mathrm{d}\tau, \qquad k = 0, 1, ..., 15, \tag{3.11}$$

  given by

$$r_k = \frac{\log(1-z_0) - (-1)^{k+1}\log(-1-z_0) - p_{k+1}}{k+1} + \log(\gamma)\frac{1-(-1)^{k+1}}{k+1} \tag{3.12}$$

  The resulting solution to the Vandermonde system is multiplied by $\gamma/(\pi n_{\tau_i})$ and the imaginary part taken to produce the final weights.

A few details should be pointed out:

- The conditions we employ for when this specialized quadrature is to be invoked is somewhat ad hoc, but have turned out to work nicely for all geometries and boundary condition configurations tested. Basically, the process runs in two steps. First, we check if z is sufficiently close to the panel Γ_{p} to warrant special treatment. Second, we check whether the basic quadrature (3.1) is likely to be accurate by computing $K\mu(z)$ with $\mu(z) = 1$ and with K chosen similar to the kernel in question, but simpler in the sense that we know the analytical value of the integral. If the numerical result is sufficiently far from the analytical value we use the specialized quadrature, otherwise we stick with basic quadrature. For details in the context of K_{Cau} we refer to Section 6 of [11].

- The computation of p_0 can be somewhat complicated in certain cases, for example when encountering boundaries with narrow passages with high curvature. Here it may happen that z_0 is located between the panel and the real axis, which means that the rotated and scaled curve $\Gamma_{\mathrm{p}'}$ cannot be continuously deformed into a straight curve from -1 to 1. Another example is in the case of K_{Log}, in which case z_0 may be located on $\Gamma_{\mathrm{p}'}$. Under these circumstances, the numerically computed value of p_0 will differ from the analytical value by a multiple of $i\pi$, and must be corrected.

- For future reference we again remark on the importance of the density $\mu(z)$ being reasonably smooth on Γ_{p} for this specialized quadrature to work well. "Reasonably" could be interpreted as the coefficients of the expansion of the density as a Legendre polynomial should decay quickly.

3.2. The linear system and the preconditioner

We are now ready to discretize the system. A *coarse mesh*, fine enough to properly resolve the boundary and the boundary data is constructed according to the rules above. Nyström discretization then gives rise to a $16n_{\mathrm{pan}} \times 16n_{\mathrm{pan}}$ linear system of equations:

$$(\mathbf{I} + \mathbf{K})\boldsymbol{\mu} = \mathbf{f}. \tag{3.13}$$

The integral operator corresponding to $\mathbf{K}$ is not compact due to the singular points, so in this coarse discretization it is in general not sufficiently resolved. Neither is the coarse density $\boldsymbol{\mu}$ resolved since it in general has singularities or other non-polynomial like asymptotic behavior near the singular points. A traditional way to deal with this problem, see for example Chapter 8 in [1], is to refine the mesh near the singular points, but this tends to make the system ill-conditioned and, of course, slower to solve due to the extra discretization points. We would like to increase the resolution close to the singular points without paying the price of more discretization points in the main system. The way to do this is via recursive compressed inverse preconditioning which is thoroughly explained in [9] and [12], and will be briefly outlined here.

We begin by splitting the matrix $\mathbf{K}$ into

$$\mathbf{K} = \mathbf{K}^{\star} + \mathbf{K}^{\circ} \tag{3.14}$$

where $\mathbf{K}^{\star}$ is zero except for 64×64 blocks corresponding to self-interaction on a four panel region around each singular point. $\mathbf{K}^{\circ}$ is then zero in these regions. The point of this is that the integral operator corresponding to $\mathbf{K}^{\circ}$ is compact, while the non-compact parts are isolated

in $\mathbf{K}^\star$. We now introduce a fine grid around each singular point by halving the panels closest to the singular point repeatedly n_{subdiv} times, called a *binary subdivision*. Letting $\mathbf{P}$ be a prolongation operator that takes functions on the coarse grid to the fine grid, this allows us to rewrite (3.13) in terms of a right preconditioned equation for a transformed density $\tilde{\boldsymbol{\mu}}$:

$$(\mathbf{I} + \mathbf{K}^\circ \mathbf{R})\tilde{\boldsymbol{\mu}} = \mathbf{f}, \tag{3.15}$$

where

$$\mathbf{R} = \mathbf{P}_{\text{w}}^{\mathbf{T}}(\mathbf{I} + \mathbf{K}_{\text{fin}}^\star)^{-\mathbf{1}}\mathbf{P}. \tag{3.16}$$

$\mathbf{K}_{\text{fin}}^\star$ is the integral operator discretized on the fine grid around each singular point and $\mathbf{P}_{\text{w}} = \mathbf{W}_{\text{fin}}\mathbf{P}\mathbf{W}_{\text{coa}}^{-\mathbf{1}}$, where $\mathbf{W}_{\text{coa}}$ and $\mathbf{W}_{\text{fin}}$ are diagonal matrices containing the Gaussian quadrature weights on the coarse and fine grids, respectively. The matrix $\mathbf{R}$ is block diagonal, equal to the identity matrix except at 64×64 blocks for each singular point. Since we above introduced the requirement that there be at least four coarse panels between any two singular points, the grids around these are disjoint, and therefore we may compute each block of $\mathbf{R}$ separately. The process is thus inherently parallel and is well suited for distributed computing. In order for this compression to work well, though, one has to take care to ensure that the panels on each side of the singular point are of roughly the same size.

Calculating the blocks of $\mathbf{R}$ from (3.16) is not recommended, however. Not only is the computation of $\mathbf{P}_{\text{w}}^{\mathbf{T}}(\mathbf{I} + \mathbf{K}_{\text{fin}}^\star)^{-\mathbf{1}}\mathbf{P}$ slow for high levels of refinement, it is also ill-conditioned. A better option is to compute each of the blocks of $\mathbf{R}$ recursively. This process is somewhat involved, and its discussion will be omitted here; see Section 6 of [9] and Section 7 of [12] for details. We note, however, that the requirements for recursive compressed inverse preconditioning to be applicable are rather mild: we only need the right hand side and $\mathbf{K}^\circ$ to be well approximated by degree 16 polynomials on every panel on the coarse mesh. Furthermore the density $\mu(z)$ must be integrable. Here, in the context of Laplace's equation, this basically only prohibits us from having non-smooth boundary data on any one boundary component. That is, the boundary data may be non-smooth at the singular points but not elsewhere. In return for this limitation we are able to handle a wide variety of boundary irregularities and the complicated behavior in the density $\mu(z)$ that these cause.

We remark that the matrix $\mathbf{K}^\circ$ is never explicitly formed; its action is computed via four calls to different fast multipole method routines [8], one for each kernel type, and a subsequent subtraction of the of the action of the coarse $\mathbf{K}^\star$. The specialized quadrature corrections to the system, as computed by the specialized quadrature described in Section 3.1 above, is applied on a panel by panel basis.

4. Post processing

In this paper our aim is to determine the solution $U(z)$ throughout the interior of a domain Ω. It is not particularly difficult within the present framework to compute quantities like $U(z)$ or $\partial U/\partial n$ on the boundary, but we will not do so here. We will evaluate $U(z)$ via (2.4) on a Cartesian grid in Ω by simply drawing the boundary in a matrix, filling this figure using a flood-fill routine, and finally compute the coordinates of the target points from these filled matrix elements. Some of the target z's will then be close to the boundary which in turn will cause the accuracy to suffer, due to the fact that the integral kernels of (2.4) become increasingly non-polynomial-like for z's close to the boundary. To achieve high accuracy we again utilize

the specialized quadrature discussed in Section 3.1 also in the computation of $U(z)$. The implementation is identical but now only K_{cau} and K_{Log} are of interest. The transformed density $\tilde{\boldsymbol{\mu}}$ is not suitable for numerical integration, however. The weight-corrected density $\hat{\boldsymbol{\mu}}$ given by

$$\hat{\boldsymbol{\mu}} = \mathbf{R}\tilde{\boldsymbol{\mu}}$$

is better in this respect. This density is however not smooth for the panels closest to the singular points, so using specialized quadrature to evaluate $U(z)$ close to singular points yields poor results. We can get around this problem by expanding the density to the fine grid, see [9]. One actually does not have to expand all the way to the fine grid, which is typically subdivided more than 50 times: an expanded density with eight subdivisions usually suffices for ordinary field evaluations. For points very close to singular points this figure needs to increase though, more on this in the numerical examples.

5. Numerical examples

In this section we are going to demonstrate the techniques described above on some setups. The solver is implemented in Matlab R2008b with some time-critical components written in C, and the computer is an ordinary workstation equipped with an Intel Core i7 processor and 6 Gb RAM. Roughly, this is what the program does:

1. Discretize the boundary according to the rules laid down at the beginning of Section 3.

2. Compute the blocks of $\mathbf{R}$ for each singular point, using recursion for speed and stability. The specialized quadrature technique described above is used whenever needed. The number of subdivisions in the calculation of the blocks depends largely on the types of boundary conditions meeting at the singular point in question. We use between 55 and 85 subdivisions depending only on this factor.

3. Compute corrections to the system matrix. These are of two types: corrections due to two panels on the boundary being close in space but not in parameter, and corrections due to the non-smoothness of the log-kernel. See Fig. 3.1. The corrections are computed using the specialized quadrature techniques.

4. Solve the system using GMRES, see [14], with the stagnation avoiding technique in Section 8 of [11]. For each iteration, the multiplication with the main system matrix $\mathbf{K}^{\circ}$ is implemented using four fast multipole calls, one for each kernel type. The action of the coarse $\mathbf{K}^{\star}$ is then subtracted and the corrections applied. The iterations continue until the relative residual falls below machine epsilon.

5. Expand the solution $\tilde{\boldsymbol{\mu}}$ to a fine grid with the panels closest to the singular points subdivided eight times, and compute $U(z)$ throughout Ω using two fast multipole calls, one for K_{Cau} and one for K_{Log}. For points close to Γ specialized quadrature is again used.

Although the computation of the blocks of $\mathbf{R}$ is parallel in nature, the lack of pertinent Matlab packages available prevented this fact to be exploited properly. The fast multipole codes used, implemented in C, are parallelized using threads, though, as is the specialized quadrature code for the field calculations.

We will consider two basic types of problem setups.

1. We prescribe the harmonic function

$$U(z) = \Re\left\{\sum_{k=1}^{n_p} \frac{1}{z - z_k}\right\} \tag{5.1}$$

to be the solution, where the n_p points z_k are all located outside Ω. The Dirichlet, Neumann, and Robin boundary conditions are then computed on the respective boundary components from this function. Note that in this case, the solution is smooth.

2. We set the Dirichlet, Neumann, and Robin data to be constant over each boundary component, with these constant values being random in the interval $[-5, 5]$. This is usually a harder problem than the first one since here, the solution $U(z)$ is not necessarily smooth. The reference solution is taken as the solution computed with twice the number of panels.

The parameter function in the Robin boundary condition is set to $\alpha(z) = |z|$ in all tests having Robin boundary conditions.

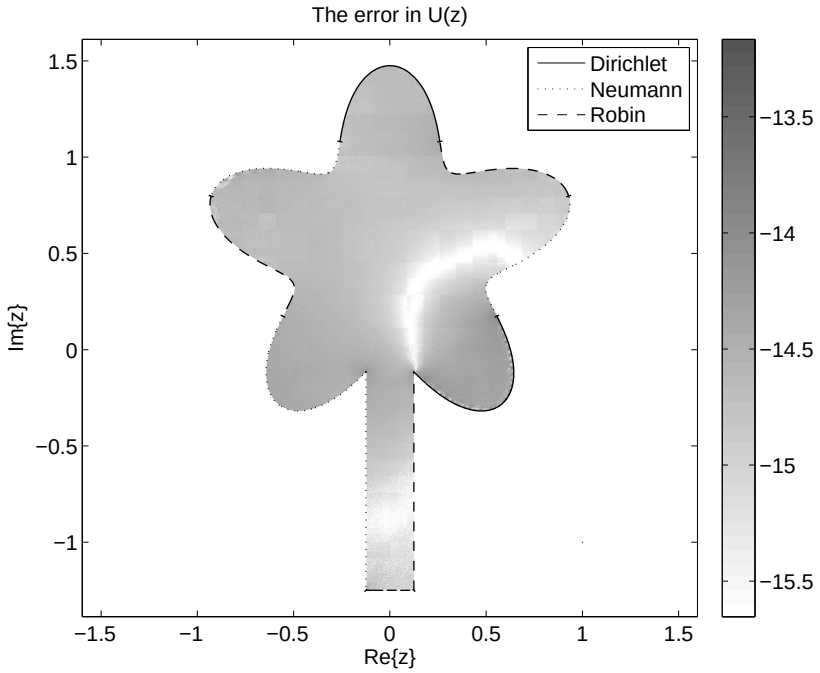

Fig. 5.1. The 10-logarithm of the estimated relative error of $U(z)$ in the tree domain, the maximum error is $6.8 \cdot 10^{-14}$. The square shaped artifacts are due to the fast multipole calculation of the solution field.

In the following numerical tests the solver is tuned for high accuracy. If one for any reason feels that these levels of accuracy are unnecessarily high is not hard to modify the solver to trade some of the accuracy for speed. For example, by halving the number of subdivisions in step 2 above and halving the number of digits requested in GMRES, the accuracy of the solution is roughly halved in the tested setups. The solution time is not quite halved, however, since some of the components of the solver remain unaffected, but the runtime is now approximately 40 % faster.

5.1. The tree domain

We begin with a tree-shaped domain with 10 singular boundary points, corresponding to changes in boundary conditions, boundary irregularities, and both. The upper part of the tree consists of a shape parametrized as

$$z(t) = 0.75(1 + 0.3\cos(5t))e^{it+\pi/2} + 0.5i, \qquad -\pi + 0.2 \leq t \leq \pi - 0.2,$$

while the trunk consists of straight lines, see Fig. 5.1. The boundary data is of type two as defined above. There are 106 panels in the discretized mesh which corresponds to 1696 discretization points, and 30 GMRES iterations are required to reach the relative residual stopping criterion of $\varepsilon_{\text{mach}}$. The maximum estimated relative error is $6.8 \cdot 10^{-14}$, and as for timings, the solution at 58834 points is computed in 3.6 seconds. The main part of this is due to the computation of the 10 blocks of $\mathbf{R}$, which takes 2.1 seconds. In this setup one could probably get away with fewer panels and still get the same level of accuracy. The bottleneck is the bottom of the trunk. Since we need at least four panels between any two singular points, and the panels should be of roughly the same length in parameter, we are over-resolving the rest of the tree. This problem could be remedied with some kind of adaptivity scheme, which has been left out in this work, see Section 6.

5.2. The flower domain

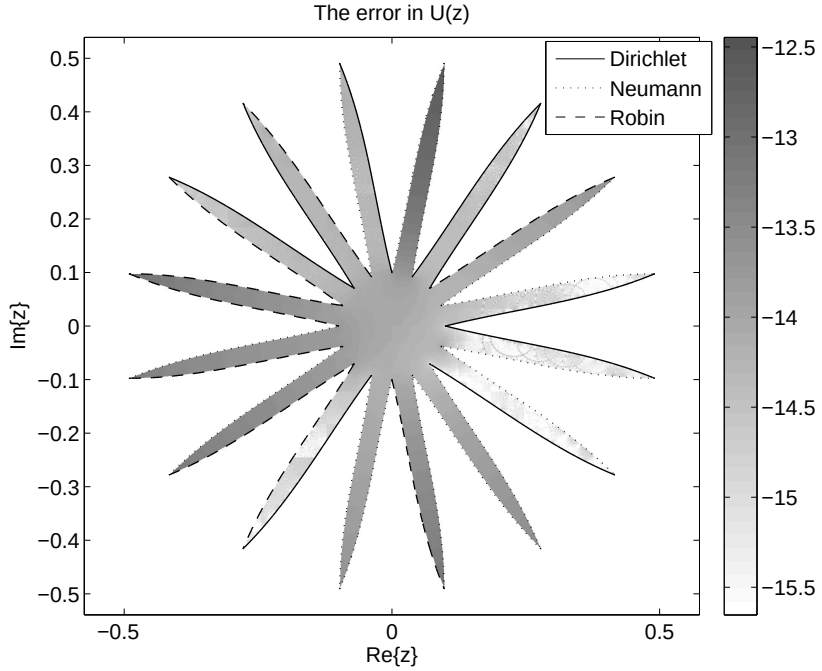

Fig. 5.2. The 10-logarithm of the estimated relative error of $U(z)$ in the flower domain, the maximum error is $3.6 \cdot 10^{-13}$.

The tree domain is perhaps not the ideal setup to really put these methods to the test: the corners are not very sharp, and the boundary does not fall in on itself requiring no matrix corrections due to close lying boundary components. A domain better suited in this respect is

the 16-petal flower depicted in Fig. 5.2. The petals are made up of two cubic splines adjusted so that the opening angle at their tips is 0.4 radians, and so that the derivative of the spline near the center points to the petal tips. The inner radius is 0.1 and the outer radius 0.5. All combinations of boundary conditions meet at the tips and in the center. Again, boundary data of type two is used as defined in the beginning of Section 5.

We begin by assigning four panels to each boundary component. However, since the boundary falls in on itself near the center of the domain, we need to use specialized quadrature to compute corrections to the system matrix in order to achieve high accuracy. But this is a bad idea, as some of the source panels are the ones closest to the singular points. As touched on above, the density is not smooth on these panels, and so the specialized quadrature gives poor results. Instead we subdivide the innermost panels twice to eliminate any need for specialized quadrature to be applied on the panels closest to the singular points. So, in the solver, the number of panels used is 192, six for each boundary component which gives 2048 discretization points in total. This problem is significantly harder to solve than the previous one, which is reflected by the fact that now 80 GMRES iterations are required to reach the stopping criterion. Still the estimated relative error is low; nowhere larger than $3.6 \cdot 10^{-13}$. The time taken to compute the solution at 43845 points in the domain is 11.5 seconds. Again, the bulk of this is due to the computation of the blocks of $\mathbf{R}$. More than half the time is spent doing this.

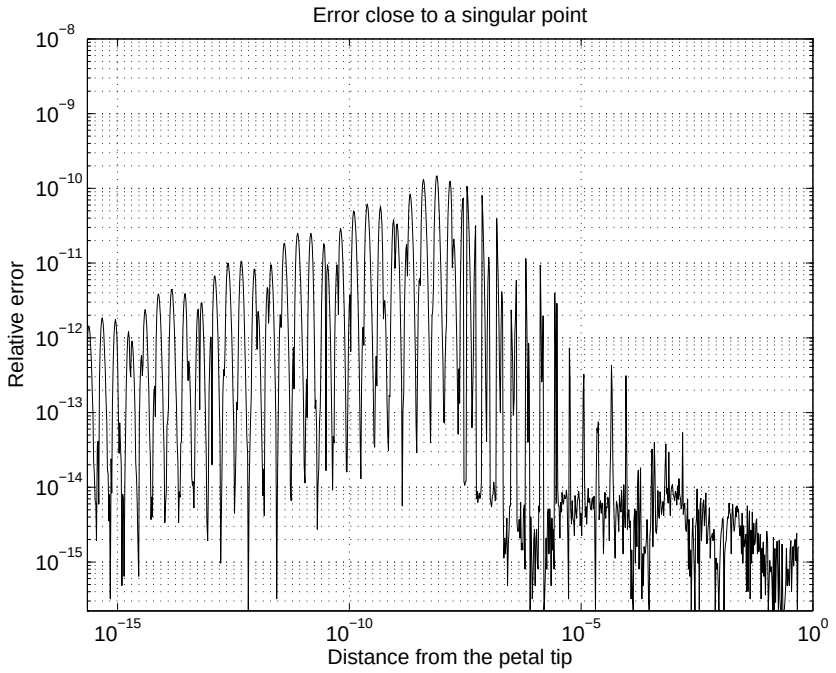

Fig. 5.3. The relative error as we traverse a straight line from the center of the flower to the tip of the petal pointing roughly to four o-clock in Fig. 5.2.

Note that the error on certain petals of the flower domain is significantly larger that for others. In general, the petals having part Dirichlet conditions fares better, which is likely to be because the solution on a petal with part Dirichlet conditions are more local in nature. That is, the solution depends largely on the Dirichlet conditions close by. In contrast, the solution on a petal with, for example, Neumann conditions, depends largely on the solution field in the

center, making it less local in nature. This behavior is not believed to be a short-coming of the solver, but rather a property of the problem solved.

To demonstrate the robustness of the solver, we will now show how it handles evaluation of $U(z)$ extremely close to a singular point. $U(z)$ is computed on a line from the center out to the tip of the petal pointing roughly to four o-clock in Fig. 5.2. The boundary conditions are of the same kind but now boundary data of type one as described above is used. Furthermore, an expanded density with eight subdivisions does not suffice for field evaluation this close to the singular point; we use fifty subdivisions for the singular point we approach, but leave the other ones at eight. At the petal tip, we have Dirichlet and Neumann boundary conditions meeting, and the density exhibits complicated behavior as the tip is approached. On the side with Neumann conditions, the density behaves approximately like $t^{-0.7327}$ as t tends to zero, which is a very strong singularity; the density is not square integrable. On the side with Dirichlet conditions it behaves approximately like $t^{0.2669}$. Both exponents are calculated from the expanded density. As can be seen in Fig. 5.3, the relative error is bounded by $2 \cdot 10^{-10}$ as we approach the petal tip, and is never out of control. The calculated solution is useful even approaching the numerical limit of closeness. If one is interested in the behavior of the density μ very close to singular points, the same procedure can be used to expand the density even further towards the singular point than is done here.

5.3. The star domain

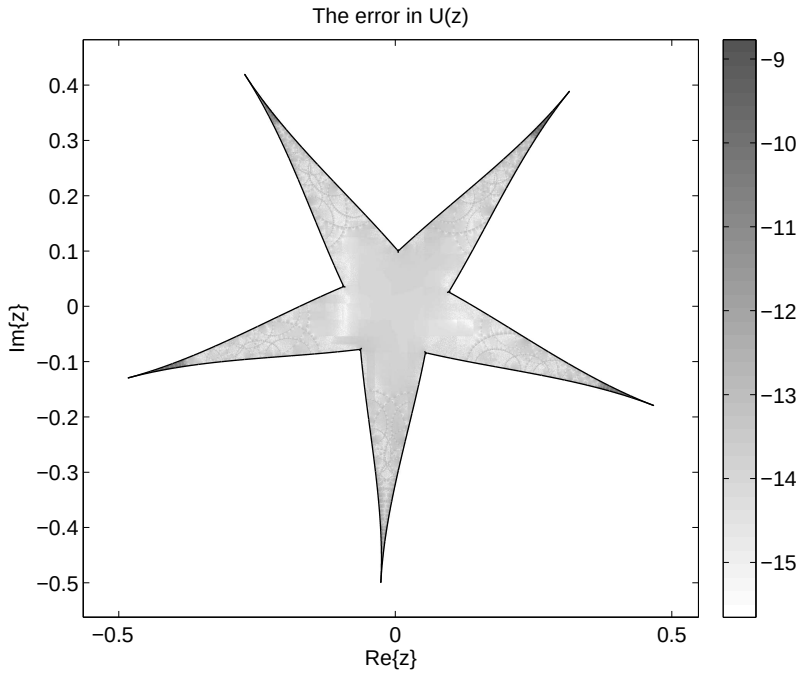

Fig. 5.4. The 10-logarithm of the estimated relative error in $U(z)$ in the 5-armed star domain, the maximum error is $1.7 \cdot 10^{-9}$, which is attained near the tips of the star. The error in the majority of the domain is well below 10^{-13}.

The solver is also capable of handling certain pathological boundaries. The tips of the 5-

armed star of Fig. 5.4 have opening angles equal to zero and are therefore cusps. Dirichlet boundary conditions of type two are applied, and the panels closest to the center of the star is subdivided twice as with the flower domain. The discretized boundary then consists of 60 panels, six on each boundary component, for a total of 960 discretization points. The main system requires only 26 GMRES iterations to solve, and the solution at 27760 points is computed in 3.3 seconds total. As can be seen in Fig. 5.4 the error is very low throughout most of the domain, but increases near the tips. The maximum relative error is $1.7 \cdot 10^{-9}$. Evaluating $U(z)$ close to the singular points works up until a distance of about 10^{-5}, but closer than that the solver fails due to severe cancellation. The point here is that even though there are difficulties near the singular points, the solution on a global scale does not have to suffer. Furthermore, in this case, with Dirichlet boundary conditions, one could compute a quite good approximation to the solution close to the tips by simply taking the mean of the closest few boundary data points.

5.4. The tribal domain

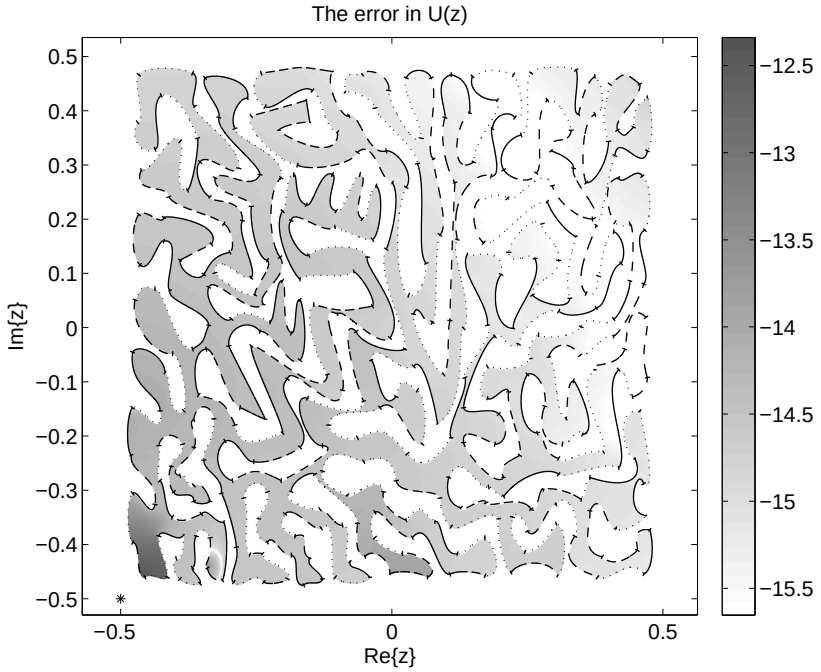

Fig. 5.5. The 10-logarithm of the relative error in $U(z)$ in the large scale tribal domain. The maximum relative error is $4.58 \cdot 10^{-13}$, and is attained near the single source point at $-0.5 - 0.5i$. As before solid lines mean Dirichlet boundary conditions, dotted Neumann and dashed Robin.

The final example is a large scale setup consisting of hand-drawn cubic splines in a complicated pattern, see Fig. 5.5. Boundary conditions of type one are used. The geometry consists of 348 boundary components, and 5211 panels, or 83376 points, are used in its discretization. On average, about 15 panels are used per boundary component, which is a good deal more than for the two previous geometries, but because of the generally higher curvature the extra panels are required to fully resolve the density and the integral operators. 170 GMRES iterations are

needed to reach full convergence in the density, and the resulting maximum relative error of the solution is $4.58 \cdot 10^{-13}$. The maximum error is attained close to the source point at $-0.5 - 0.5\mathrm{i}$, which is not surprising since the prescribed solution varies the most in that area. Some adaptivity scheme could again be used to alleviate the problem, see Section 6. The solution at 95357 points is computed in just over four minutes. Of this, 81 seconds is due to the computation of the blocks of $\mathbf{R}$, and 130 seconds is spent in GMRES. The post processor requires 23 seconds to complete.

6. Conclusions and outlook

We have demonstrated a solver for Laplace's equation on general simply connected domains in the plane with general boundary conditions. The solutions are very accurate, are computed fairly quickly, and the solver can be subjected to difficult domains and boundary conditions without failing, demonstrating robustness. These results show that the recursive compressed inverse preconditioning technique is applicable with good results also for problems of this type. This paper could perhaps be the first step towards a general solver for Laplace's equation in the plane, a solver capable of handling most of the problems arising in applications. There are, however, a number of points that need to be addressed in order to make this solver a truly general one. The most important is likely to be that of adaptivity, a general solver should be able to meet a prescribed accuracy while at least roughly minimizing the use of computational resources doing so. The need for adaptivity arises in two contexts:

- *The configuration of panels making up the coarse mesh*, that is, the number of panels and the position of these. Automatic determination of an even near optimal mesh for a given level of accuracy seems to be very difficult. There are a number of integral operators involved, and the singularities of the computed density at corners and changes in boundary conditions must be taken into account. Schemes ensuring a certain accuracy in the solution for smooth domains with homogeneous boundary conditions have been developed, see for example [2], but these techniques do not extend to the present circumstances.

- *The number of subdivisions needed in the computation of the blocks of* $\mathbf{R}$ *of equation (3.16).* The number of subdivisions required to properly resolve the singular behavior of the sought density is not known beforehand, but it is crucial that the number used is chosen high enough to ensure the target accuracy. While the proposed simple scheme with a fixed number of subdivisions depending only on the type of boundary conditions entering the singular point works well for computing high accuracy solutions, it can likely be improved upon. Perhaps also taking into account the opening angle of a corner, or the curvature of the boundary near the singular point. As the wall clock times showed, the computation of the blocks of $\mathbf{R}$ is time consuming, so any gains here would quickly be noticeable in the total running time.

These, and further improvements such as capability to handle multiply connected interior and exterior domains is left for future work.

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

References

[1] K. Atkinson, "The Numerical Solution of Integral Equations of the Second Kind", Cambridge University Press, 1997.

[2] K. Atkinson and Y. Jeon, "Algorithm 788: Automatic boundary integral equation program for the planar Laplace equation", ACM Trans. Math. Softw. 24(4), 395-417 (1998), doi:10.1145/293686.293692

[3] I. Balsim, M. A. Neimark and D. S. Rumschitzki, "Harmonic solutions of a mixed boundary problem arising in the modeling of macromolecular transport into vessel walls", Comput. Math. Appl.,59 1897-1908 (2010) doi:10.1016/j.camwa.2008.11.020

[4] J.Bremer and V. Rokhlin, "Efficient discretization of Laplace boundary integral equations on polygonal domains", J. Comput. Phys, 229(7) 2507-2525 (2010), doi:10.1016/j.jcp.2009.12.001

[5] J.Bremer, V. Rokhlin and I. Sammis, "Universal quadratures for boundary integral equations on two-dimensional domains with corners", J. Comput. Phys, 229(22) 8259-8280 (2010), doi:10.1016/j.jcp.2010.06.040

[6] B. E. J. Dahlberg and C. E. Kenig, "Hardy spaces and the Neumann problem in L^p for Laplace's equation in Lipschitz domains", Annals of Mathematics, 125, 437-465(1987)

[7] S. Garrido, L. Moreno, D. Blanco, "Robotic Motion Using Harmonic Functions and Finite Elements", J. Intell. Robot. Syst., 59, 5773 (2010) , doi:10.1007/s10846-009-9381-3

[8] L. Greengard and V. Rokhlin "A fast algorithm for particle simulations", J. Comput. Phys. 73(2),325-348 (1987), doi:10.1016/0021-991(87)90140-9

[9] J. Helsing "Integral equation methods for elliptic problems with boundary conditions of mixed type", J. Comput. Phys. (2009), doi:10.1016/j.jcp.2009.09.004

[10] J. Helsing, B. T. Johansson, "Fast reconstruction of harmonic functions from Cauchy data using integral equation techniques", Inverse Probl. Sci. Engn., 18(3), 381-399, (2010), doi:10.1080/17415971003624322

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

[12] J. Helsing and R. Ojala, "Corner singularities for elliptic problems: integral equations, graded meshes, quadrature", J. Comput. Phys. 227(20) 8820-8840 (2008)

[13] G. Monegato, "Quadrature Formulas for Functions with Poles Near the Interval of Integration", Math. Comput. 47(175) 301-312 (1986)

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

[15] G. Verchota, "Layer potentials and regularity for the Dirichlet problem for Laplace's equation in Lipschitz domains", J. Funct. Anal., 59 (1984) 572-611.

[16] L. J. Willett, S. K. Loyalka, and R. V. Tompson, "Vapor deposition modeling for an entrenched wafer geometry", J. Vac. Sci. Technol., 17(1), 212-228 (1999), doi: 10.1116/1.581575