A numerical method for solving Laplace’s equation in two dimensions on simply-connected domains with smooth boundary

Sam Watson

12 May, 2011

1 Introduction

Let $\Omega \subset \mathbb{R}^2$ be a bounded, simply-connected domain, and suppose that $\partial \Omega$ can be parameterized by a continuous map $s \mapsto (x(s), y(s))$ from the unit circle to $\mathbb{R}^2$. Our goal is to implement an efficient finite-difference method solver for the Laplace equation with Dirichlet boundary conditions, that is

\[
\begin{align*}
\Delta u(x) &= 0, \quad x \in \Omega \\
u(x) &= f, \quad x \in \partial \Omega,
\end{align*}
\]

(1)

where $f$ is a function on $\partial \Omega$ prescribing boundary values of $u$. If $\partial \Omega$ is a rectangle, then the usual square-grid finite-difference scheme with five-point stencil for the Laplacian allows for fast solution of (1) throughout $\Omega$. If $\partial \Omega$ is $C^1$, then boundary integral equations can be used to accurately approximate $u(x)$ at a few points $x$. We will show how to combine these methods to solve (1) throughout $\Omega$ in the case that $\partial \Omega$ is $C^1$. The ideas here are drawn from [1].

Our general strategy consists of three steps, which may be loosely described as follows.

1. Use a boundary integral equation to compute $u$ at all points of $\Omega$ which are close but not extremely close to $\partial \Omega$.

2. Use an interpolation technique to approximate $u$ between $\partial \Omega$ and the layer where $u$ was approximated in Step 1.

3. Extend the solution to all of $\Omega$ by embedding $\Omega$ in a regular region $D$ and implementing a fast solver in $D$.

Step 2 is necessary because the accuracy of the boundary integral equation method in Step 1 breaks down for $x$ too close to the boundary. Step 3 is necessary because boundary integral equation methods are too slow to compute for the large number of lattice points in $\Omega$. (Note that there are $O(h^{-1})$ lattice points within a few layers of $\partial \Omega$, while there are $O(h^{-2})$ lattice points in $\Omega$.) Also, we will see that Steps 1 and 2 are prerequisites for our implementation of the third step.
2 Boundary Integral Equations

Let $G(x, y) := \frac{1}{2\pi} \log|x - y|$ be the two-dimensional Green's function for the Laplacian, and define the kernel

$$K(x, y) := \frac{\partial G}{\partial \nu_y}(x, y), \quad x \in \overline{\Omega}, \; y \in \partial \Omega,$$

where $\nu_y$ is the unit normal. In class we learned that if the boundary dipole density $\phi$ is chosen so that

$$f(t) = \frac{1}{2} \phi(t) + \int_{\partial \Omega} K(s, t) \phi(s) \, ds,$$

then the solution of (1) is given by

$$u(x) = \int_{\partial \Omega} K(x, y(s)) \phi(s) \, ds.$$

Denote by $T_K$ the operator $\phi \mapsto \int_{\partial \Omega} K(s, t) \phi(s) \, ds$. The spectrum of $(I/2 + T_K)$ does not accumulate at 0, and the kernel $K(x, y)$ remains bounded for $x, y \in \partial \Omega$ (the only problem would be $x = y$, but the limit as $s \to t$ of $K(x(s), y(s))$ is finite [2]). Therefore, we can solve (2) accurately by inverting $(I/2 + T_K)$. On the other hand, if $y$ approaches $x$ from the interior of $\Omega$, then $K$ becomes unbounded and (3) cannot be evaluated accurately.

To consider this failure more carefully, let $B_1$ be the set of nodes in $\Omega$ which have neighbors in $\mathbb{R}^2 \setminus \Omega$, and for $n \geq 1$ let $B_{n+1}$ be the set of nodes in $\Omega \setminus \bigcup_{k=1}^{n-1} B_k$ which have neighbors in $B_n$. The nodes in $B_n$ for $n = 1, 2, 3$ are shown below in green, blue, and red, respectively, for an elliptical region $\Omega$. A bit of numerical experimentation suggests that (when the boundary node spacing matches the lattice spacing) the uniform error of the integral equation approximation on $B_n$ decays by roughly an order of magnitude each time $n$ is incremented (see Section 4 for details). Accordingly, we will solve for $U$ on $B_3$ and interpolate to find $U$ on $B_1 \cup B_2$. The interpolation method is not emphasized in [1]; I used Mathematica's built-in interpolation by choosing for each grid point a set of 5 nearest neighbors in $B_3 \cup \partial \Omega$ and interpolating.

The three layers of lattice points $B_1$, $B_2$, and $B_3$, in decreasing proximity to $\partial \Omega$. 

![The three layers of lattice points](image)
3 Extending to the Interior

Now we will show how to use the methods in the previous sections, which identify $u$ at a relatively small number of points, to solve (1) throughout $\Omega$. Suppose that $U = u(x_i, x_j)$ is the (analytic) solution of (1) in $\Omega$ evaluated at the grid points. From the previous step, we have a numerical approximation $U$ at all the grid points of $\Omega$ within three layers of $\partial \Omega$. Let $R$ be a square containing $\Omega$, and set $U = 0$ in $R \setminus \Omega$. Note that $U$ is now defined everywhere except the interior points of $\Omega$ which are more than three layers away from the boundary. Since the stencil of the (second-order) Laplacian has radius 1 (in graph distance), this allows us to compute $\Delta U$ at the lattice points in $(R \setminus \Omega) \cup B_1$ (these are all the points in the square except those inside $\Omega$ which have no neighbors outside $\Omega$). Hence we can pose the following PDE in $R$:

$$
\Delta V = \begin{cases} 
\Delta U & \text{in } (R \setminus \Omega) \cup B_1 \\
0 & \text{in } \Omega \setminus B_1,
\end{cases}
$$

with zero boundary conditions. Since the discrete Laplacian is a second-order accurate approximation of the Laplacian, $\Delta U = O(h^2)$ in $\Omega \setminus B_1$ while $\Delta V = 0$ in $\Omega \setminus B_1$. Everywhere else in $R$, we see that $\Delta V = \Delta U$ by construction. Moreover, $V$ agrees with $U$ on $\partial R$ (since both are zero). Altogether, we have $\Delta(V - U) = O(h^2)$. Since we have a second-order solver for the problem (4), this shows that $V - U = O(h^2)$.

This method is very easy to implement, and is very fast and accurate. The boundary integral equation and interpolation steps are the bottlenecks of this algorithm, in both accuracy and time.

4 Numerical Results

For numerical experiments, we let $\Omega$ be an ellipse centered at $(1/2, 1/2)$ in the unit square $R = [0,1]^2$ with semi-major axis 0.35 and semi-minor axis 0.3. We used the harmonic functions $u(x, y) = x^2 - y^2$ and $u(x, y) = x^4 - 6x^2y^2 + y^4$ for boundary conditions and accuracy testing. We find that the accuracy of the boundary integral equation method increases in the number of lattice points separating $x$ and $\partial \Omega$, as shown in the following table. For this experiment, we have used the same number of points $N$ in our subdivision of the lattice and in positioning the nodes around $\partial \Omega$.

| $N$ | $\sup_{x \in B_1} |U(x) - u(x)|$ | $\sup_{x \in B_2} |U(x) - u(x)|$ | $\sup_{x \in B_3} |U(x) - u(x)|$ |
|-----|------------------------|------------------------|------------------------|
| 20  | 0.445798               | 0.0604274              | 0.000461818            |
| 40  | 0.406644               | 0.0326514              | 0.00320584             |
| 60  | 0.934614               | 0.0277384              | 0.00376409             |
| 80  | 0.591568               | 0.049624               | 0.00332378             |
| 100 | 0.730315               | 0.0346357              | 0.00259199             |

The uniform error for boundary integral equation methods for varying mesh widths $1/N$, at the three layers $B_1, B_2, B_3$ of decreasing proximity to $\partial \Omega$. We used $N$ boundary nodes for these calculations. We can see that the performance near $\partial \Omega$ is poor.

The paper [1] suggests taking $O(N^{3/2})$ points to achieve second-order accuracy (the exponent is obtained by looking an error bound for the trapezoid rule). The table above shows why it is necessary to have a
finer spacing for the boundary nodes than for the lattice: one needs for the three lattice units of separation from \( \partial \Omega \) to correspond to an increasing number of boundary-mesh units if accuracy is to improve. In fact, the following table shows that the approximation can get worse if the boundary mesh doesn’t increase faster than linearly in \( N \). On the other hand, the second table shows that the choice of \( N^{3/2} \) might be faster than necessary.

<table>
<thead>
<tr>
<th>( N )</th>
<th>20</th>
<th>40</th>
<th>60</th>
<th>80</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \sup_{x \in B_3}</td>
<td>U(x) - u(x)</td>
<td>)</td>
<td>( 4.7 \times 10^{-11} )</td>
<td>( 8.6 \times 10^{-8} )</td>
</tr>
</tbody>
</table>

The uniform error on \( B_3 \) with \( 3N \) boundary nodes. Here we used the harmonic function \( u(x, y) = x^4 - 6x^2y^2 + y^4 \) for boundary conditions and accuracy checking.

<table>
<thead>
<tr>
<th>( N )</th>
<th>20</th>
<th>40</th>
<th>60</th>
<th>80</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \sup_{x \in B_3}</td>
<td>U(x) - u(x)</td>
<td>)</td>
<td>( 7.2 \times 10^{-8} )</td>
<td>( 8.6 \times 10^{-9} )</td>
<td>( 2.3 \times 10^{-10} )</td>
</tr>
</tbody>
</table>

The uniform error on \( B_3 \) with \( \lfloor N^{3/2} \rfloor \) boundary nodes. Again, \( u(x, y) = x^4 - 6x^2y^2 + y^4 \). These data suggest that the boundary integral equation error is sixth-order accurate for the choice of \( O(N^{3/2}) \) boundary nodes, rather than second.

Implementing this programme with \( N = 20, 30, \ldots, 100 \) and \( \lfloor N^{3/2} \rfloor \) boundary nodes, we obtain the following graph for the uniform error in \( \Omega \).

![Graph of uniform error in \( \Omega \)](image)

The uniform error in \( \Omega \) for the overall method with \( \lfloor N^{3/2} \rfloor \) boundary nodes and \( u(x, y) = x^4 - 6x^2y^2 + y^4 \). The slope of this line is \(-1.78\), and without the outlier the slope is nearly \(-2\).

We see that the correct second-order accuracy is achieved. Since the BIE methods are sixth-order accurate and the Poisson solver is second-order accurate, this suggests that the interpolation is at least
second-order accurate. For interest, we also show the graph resulting from applying the same algorithm to $u(x, y) = x^2 - y^2$. We see that the algorithm is extremely accurate for this function, but does not show monotonic decay in error. We attribute this phenomenon to the fact that this function is discrete harmonic, so the discrete Laplacian is infinitely accurate rather than second-order accurate. Note that the slope of the initial portion of the graph below is about $-6$, which matches the order of accuracy in the boundary integral equation method.

The uniform error in $\Omega$ for the overall method with $\lfloor N^{3/2} \rfloor$ boundary nodes and $u(x, y) = x^2 - y^2$. Up to about $N = 50$ we find that the error decreases very rapidly (up to $\log(N) = 4$ the slope is about $-6$). This strange behavior can be attributed to the fact that the discrete Laplacian is exact for this function.

References
