# Python Implementation of Boundary Integral Methods for Optical Cloaking

UNIVERSITY OF CALIFORNIA UCMERCED
E. A. Cortes, C. Carvalho, Applied Math Department

## INTRODUCTION

Our goal is to simulate optical cloaking devices by calculating how light travels through defined layers. Optical cloaking refers to the act of making something invisible in some directions by preventing the scattering of light as it hits the boundary. Cloaking is often seen in science fiction, but also has real world applications for radar and military science.

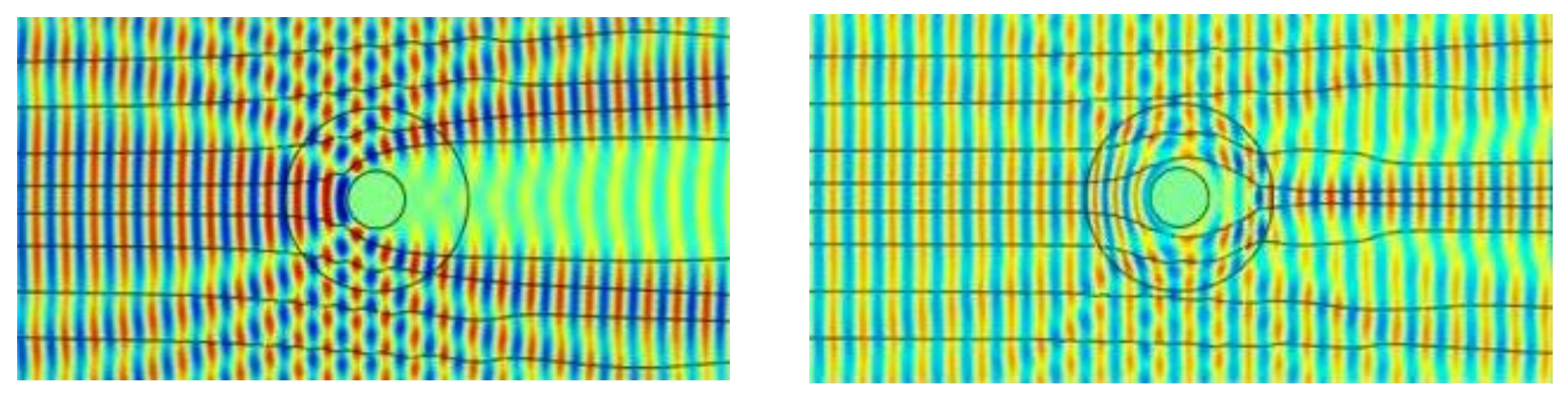
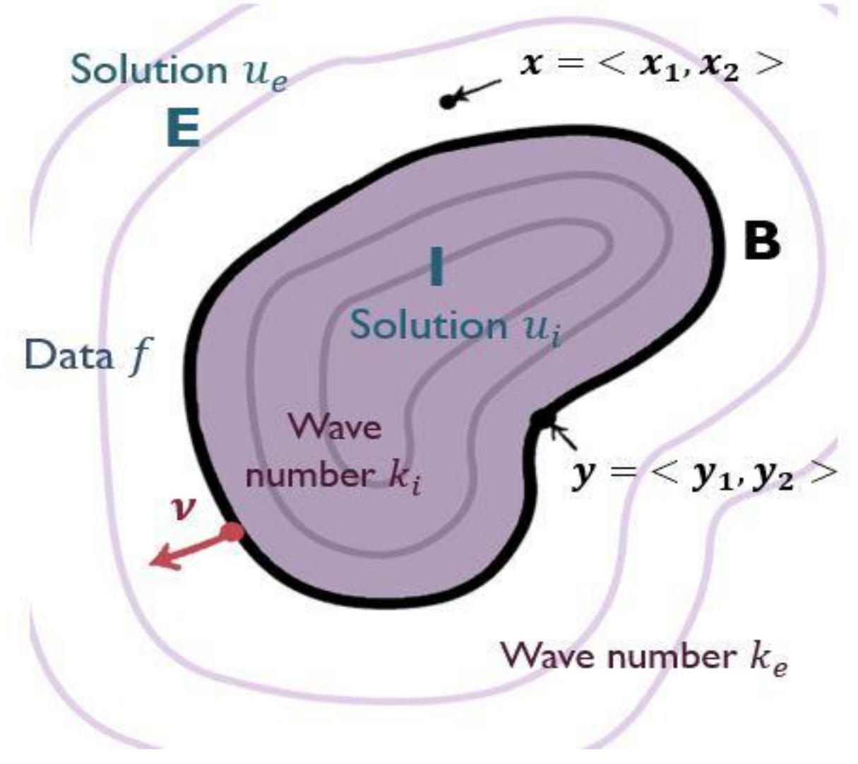


**Figure 1.** The scattering of light by an uncloaked object (left) versus a cloaked object (right) [3].

We use boundary integral equation methods to compute the solution in layered boundaries, and we implement in Python an approximation using the periodic trapezoid rule (PTR) and the Kress Quadrature.

## TRANSMISSION PROBLEM



**Helmholtz Equations**

$$\Delta u_e + k_e^2 u_e = 0, \quad x \ \epsilon \ E$$
$$\Delta u_i + k_i^2 u_i = 0, \quad x \ \epsilon \ I$$

**Transmission Conditions**

$$u_e = u_i, \quad x \ \epsilon \ B$$
$$\frac{1}{k_i}\frac{\partial u_i}{\partial \nu} = \frac{1}{k_e}\frac{\partial u_e}{\partial \nu}, \quad x \ \epsilon \ B$$

(1) **Boundary Integral Equations**

$$u_e(x) = f(x) + \int_B \frac{\partial \Phi^e}{\partial \nu(y)}(x,y)u_i(y) - \frac{k_e}{k_i}\int_B \Phi^e(x,y)\frac{\partial u_i(y)}{\partial \nu(y)} d\sigma_y, \quad x \ \epsilon \ E$$

$$u_i(x) = -\int_B \frac{\partial \Phi^i}{\partial \nu(y)}(x,y)u_i(y) + \int_B \Phi^i(x,y)\frac{\partial u_i(y)}{\partial \nu(y)}(y) d\sigma_y, \quad x \ \epsilon \ I$$

Note that $\Phi$ represents the Fundamental Solution to the Helmholtz Equation:

$$\Phi(x,y) = \frac{i}{4}H_0^{(1)}(k|x-y|)$$

We compute $u_i$ and $\frac{\partial u_i}{\partial \nu}$ on the boundary using this boundary integral system:

$$(2) \quad \begin{bmatrix} \frac{I}{2} - D^e & \frac{k_e}{k_i}S^e \\ \frac{I}{2} + D^i & -S^i \end{bmatrix} \begin{bmatrix} u_i \\ \frac{\partial u_i}{\partial \nu} \end{bmatrix} = \begin{bmatrix} f \\ 0 \end{bmatrix}$$

## NUMERICAL METHOD

By parameterizing the boundary B, we can apply the PTR to approximate $u_e$ and $u_i$. The PTR allows us to use a summation to get our solution. However, to solve the system on the boundary, we must use the Kress quadrature because $\Phi^e$ and $\Phi^i$ are singular on the boundary [1].

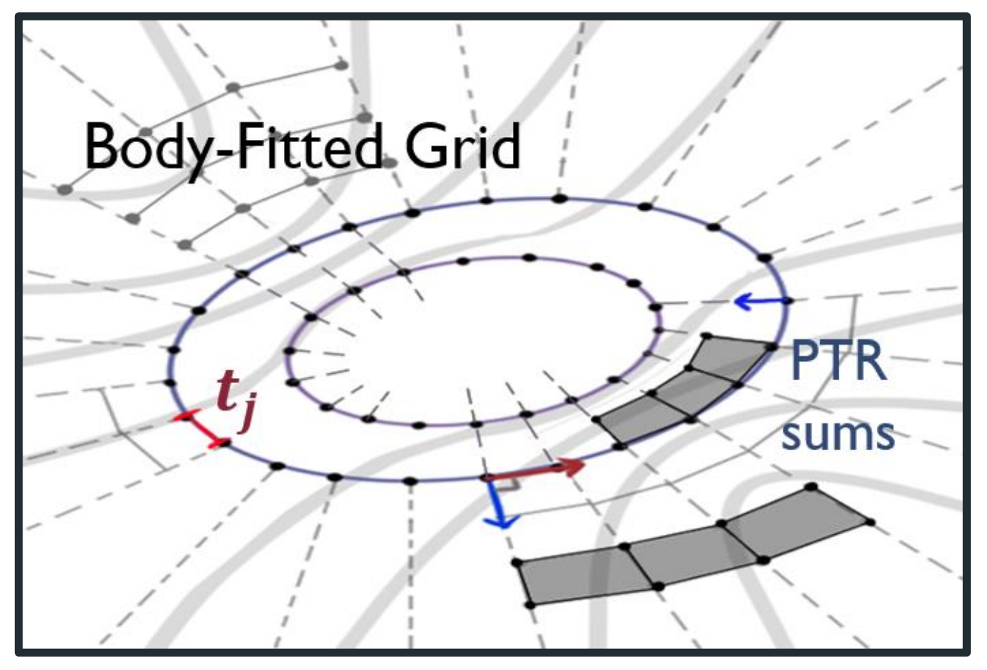


**Figure 2.** We compute the solution on the body-fitted grid using the PTR. The solution is computed in layers.

## VALIDATION

To validate our method, we compute the solution which is known for

$$f(x) = e^{ik_e(\cos(\pi/4)x + \sin(\pi/4)y)}$$
$$k_e = k_i = 3$$

**Step 1 – Calculate $u_i$ and $\frac{\partial u_i}{\partial \nu}$ from (2)**

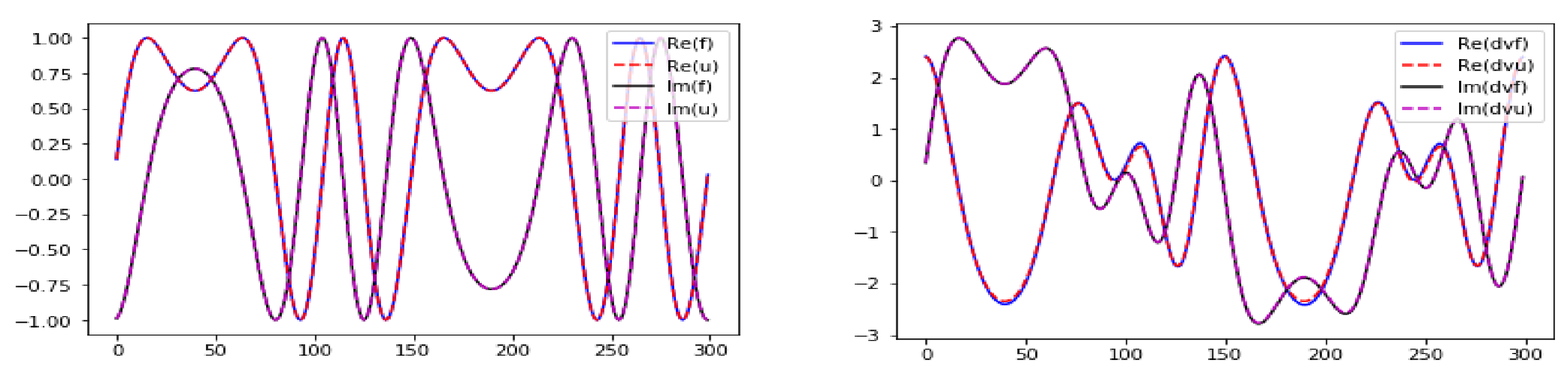


**Figure 3.** A comparison of the exact values of the boundary data (solid) versus the ones found by the Kress Quadrature (dashed).

**Step 2 – Compute solution inside/outside boundary from (1)**

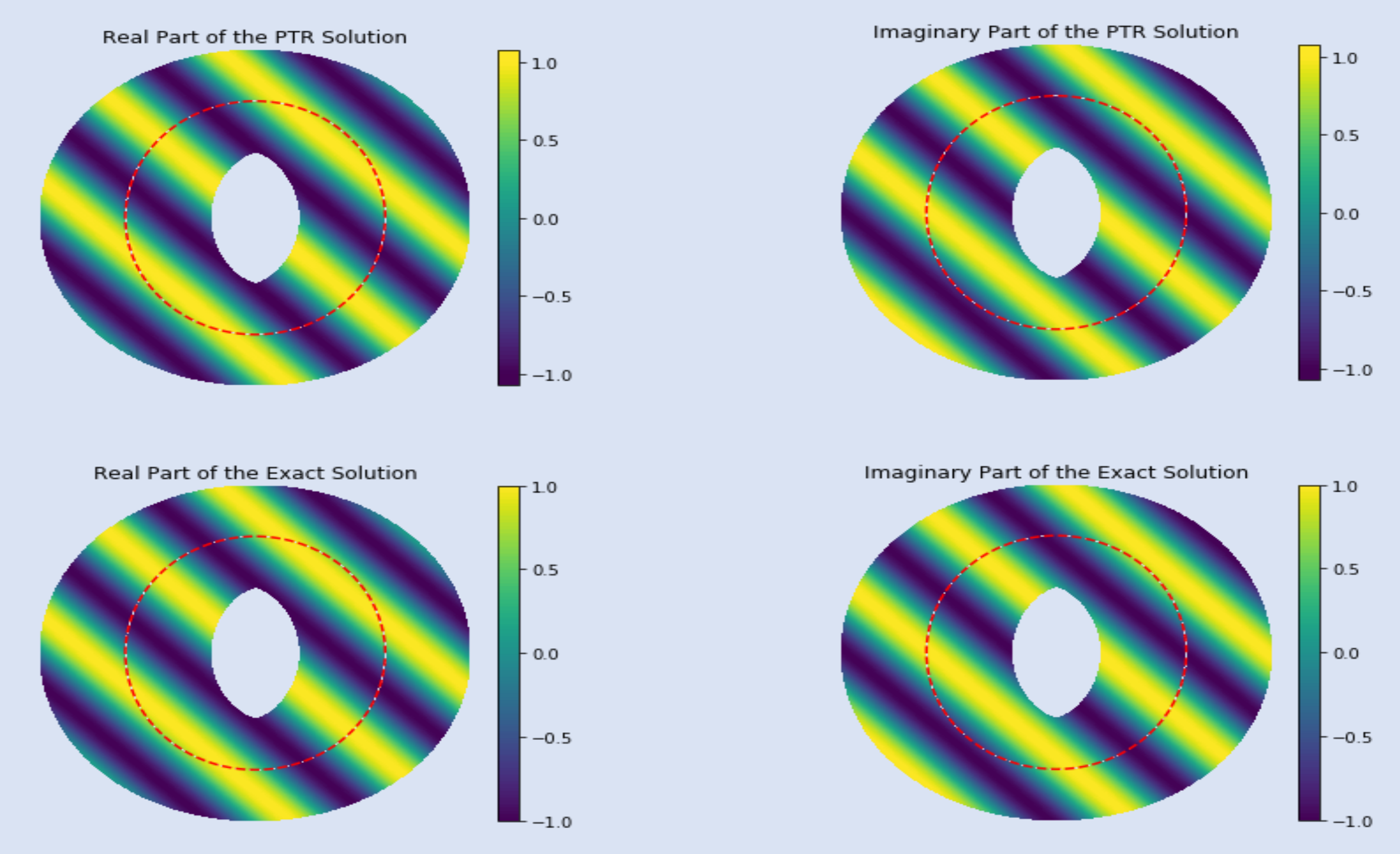
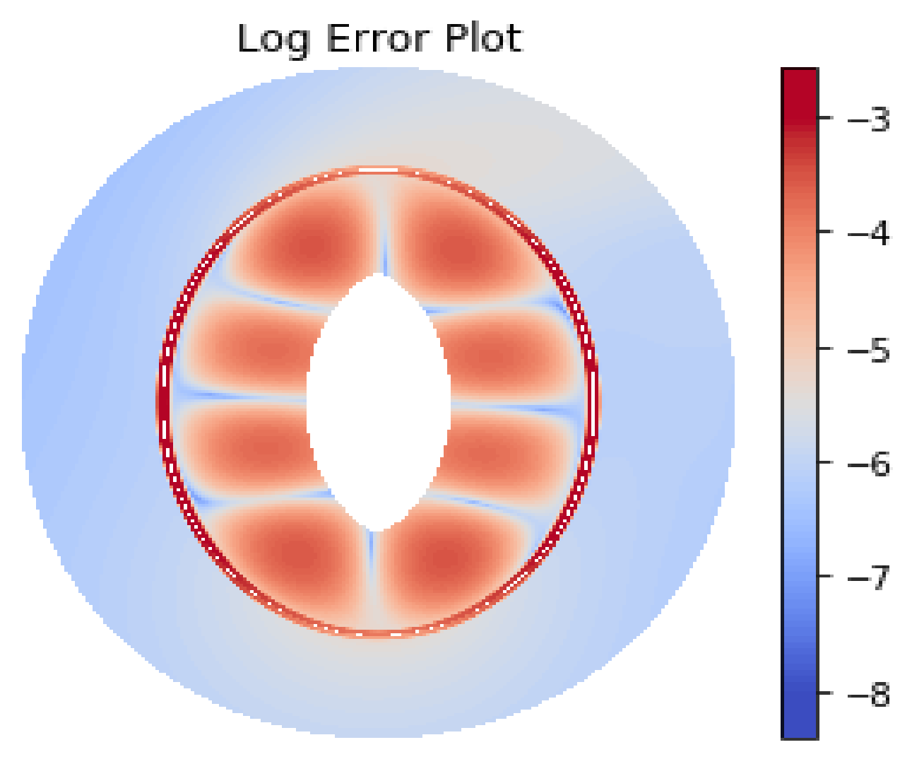


**Figure 4.** Top down plots of the PTR solutions versus the exact solutions for an ellipse-shaped boundary with M = 300 points on the boundary. Boundary indicated by red dashes.

### Log Error of Method on Elliptical Boundary

The log plot of the error between the PTR solution and the exact solution is shown. There is a significant increase in error around the defined boundary. This is because we compute nearly singular integrals.

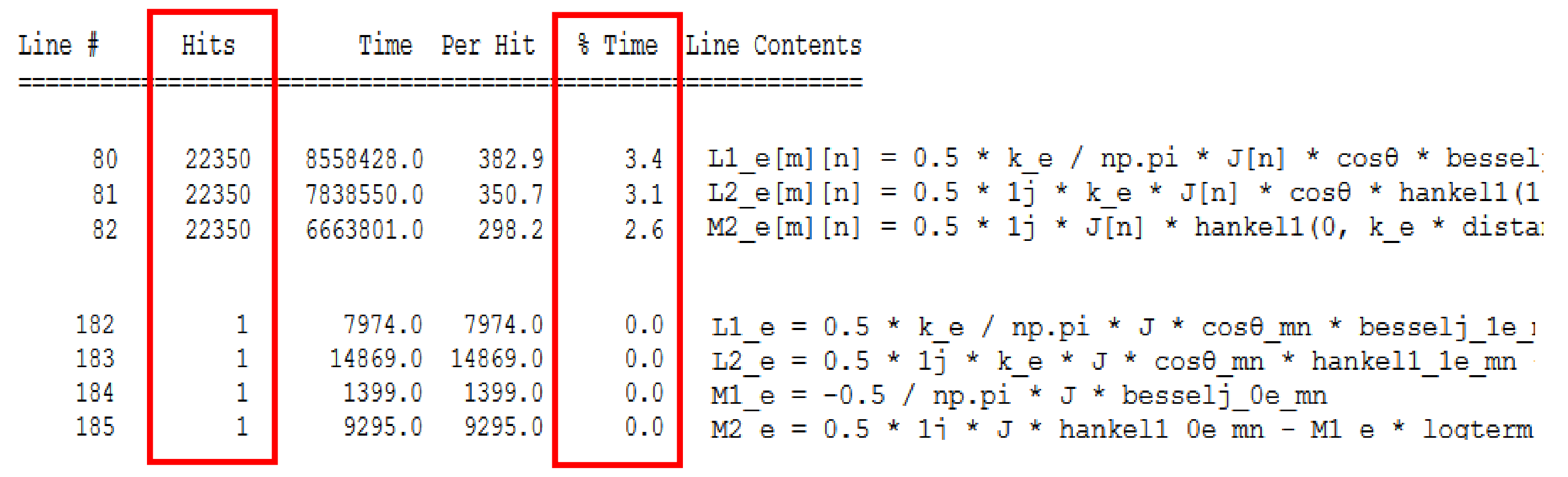


## CODE OPTIMIZATION

For a boundary with **M** points, each with **N** points along the outward and inward normal vectors:

1) Calculation of boundary data takes O(**M** x **M**) = O(**M²**) time
2) Calculation of grid takes O(**M** x **N**) time

Runtime of each function is reduced by doing operations outside of the *for loops* with **M** x **M** and **M** x **N** matrices.



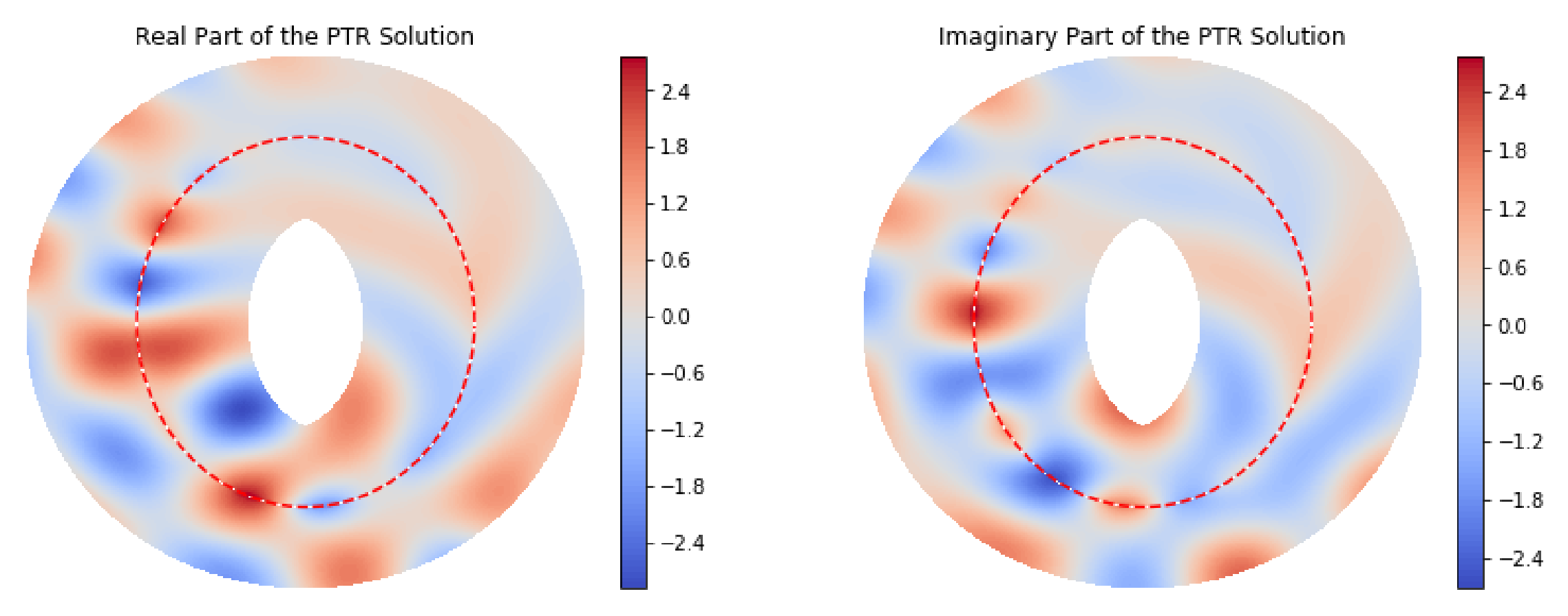
```
Line #      Hits         Time  Per Hit   % Time  Line Contents
==============================================================

    80     22350    8558428.0    382.9      3.4      L1_e[m][n] = 0.5 * k_e / np.pi * J[n] * cosθ * bessel
    81     22350    7838550.0    350.7      3.1      L2_e[m][n] = 0.5 * 1j * k_e * J[n] * cosθ * hankel1(1
    82     22350    6663801.0    298.2      2.6      M2_e[m][n] = 0.5 * 1j * J[n] * hankel1(0, k_e * dista

   182         1       7974.0   7974.0      0.0      L1_e = 0.5 * k_e / np.pi * J * cosθ_mn * besselj_1e_
   183         1      14869.0  14869.0      0.0      L2_e = 0.5 * 1j * k_e * J * cosθ_mn * hankel1_1e_mn
   184         1       1399.0   1399.0      0.0      M1_e = -0.5 / np.pi * J * besselj_0e_mn
   185         1       9295.0   9295.0      0.0      M2_e = 0.5 * 1j * J * hankel1_0e_mn - M1_e * logterm
```

**Figure 5.** Lines from the STEP 1's Python function compared to the optimized form. We reduce runtime percentage line by line by converting them to single matrix operations.

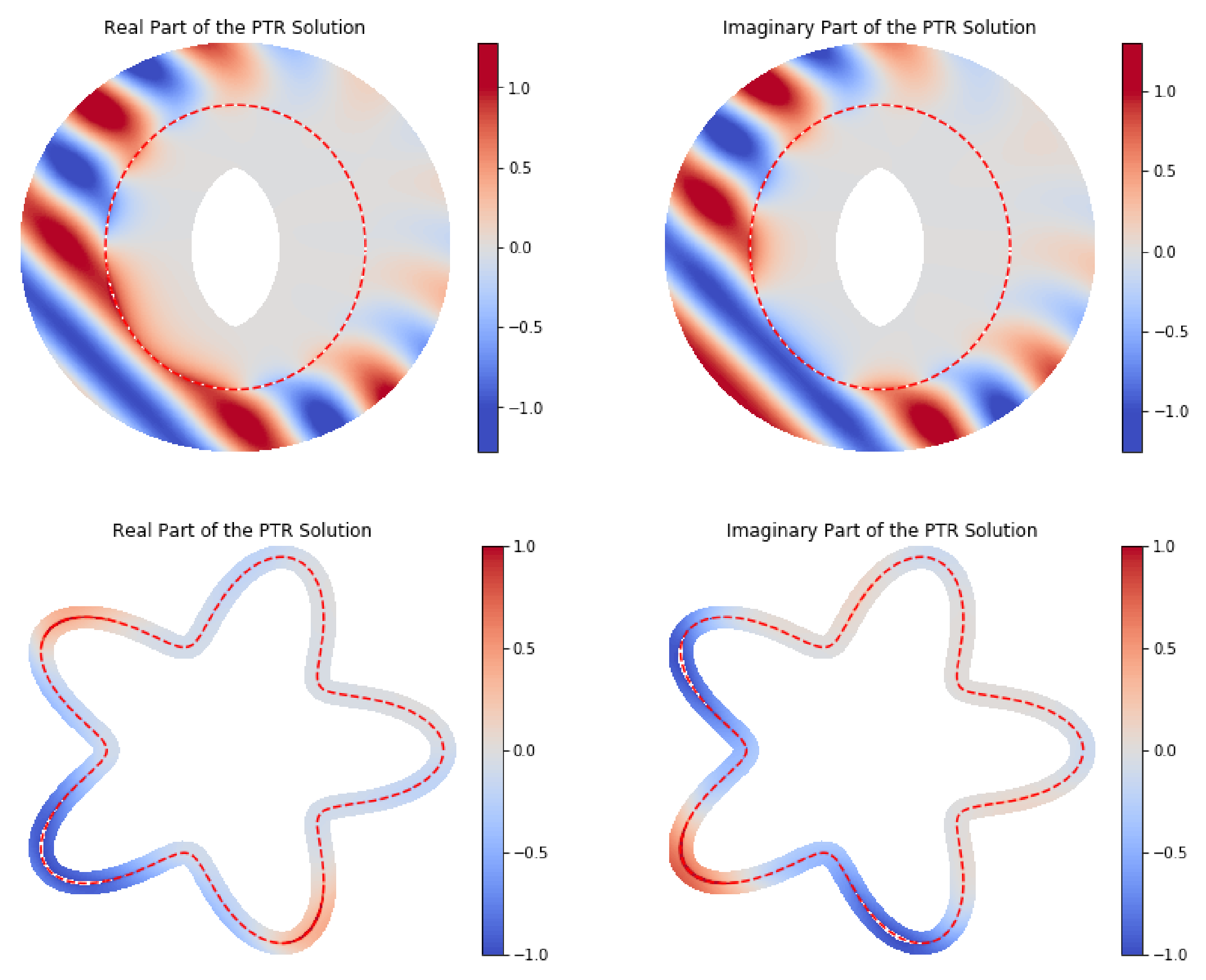
## TRANSMISSION FOR OTHER MATERIALS

Our method can be extended to the transmission of a source in domains with different properties.

$$k_e = 3, k_i = -3$$



$$k_e = 3, k_i = 3i$$



## CONCLUSIONS

- By applying PTR and the Kress Quadrature to the transmission problem, we are able to compute the effect of a source in layered media.
- Because the integrals are nearly singular close to the boundary, we will seek an alternative method to reduce this error [2].
- We can apply our method to different boundary shapes and to domains with different properties (i.e. metamaterials).
- We can simulate cloaking by extending this method to multi-layered domains with different values of *k*.

## REFERENCES

[1] R. Kress, Boundary Integral Equations in Time-Harmonic Acoustic Scattering. Math Comput. Model (1991).
[2] C. Carvalho, S. Kharti, and A.D. Kim, Asymptotic analysis for close evaluation of layer potentials-, submitted. J. Comp. Phys. (2018).
[3] V. Shalev, Illustration of a theoretical cloaking device. (2007).
[4] Boundary Cloaking GitHub Repository (2019).

GitHub QR Code