Math 511 Project 1:
Farfield Expansion ABCs for Acoustic Scattering

Shane McQuarrie

Introduction

We consider a wave-scattering problem in the plane with an impenetrable obstacle that reflects an incoming wave. The initial wave, approaching the obstacle, is called the *incident field* (u_{inc}). The wave reflects on the obstacle, creating a *scattering field* (u_{sc}). This problem requires careful custom grid generation if the obstacle has a complicated boundary, but we will only consider the simple case where the object is a cylindrical cavity of circular cross section. Specifically, let the obstacle (also called the *scatterer*) be bounded by the circle Γ of radius $r_0 = 1$.

In order to make numerical approximations for the scattering field u_{sc} we must impose an artificial boundary: another circle, with radius $R > 1$. Let Ω^- be the interior of the annular region bounded by the two circles of radius r_0 and R. The following BVP, based on the *Helmholtz equation*, describes the dynamics of this situation in polar coordinates.

$$\Delta_{r,\theta} u_1 + k^2 u_1 = 0, \qquad \text{in } \Omega^-, \tag{1}$$

$$u_1 = -u_{inc}, \qquad \text{on } \Gamma, \tag{2}$$

$$u_1(R,\theta) = e^{ikR}\frac{f_0(\theta)}{(kR)^{1/2}} \tag{3}$$

$$\partial_r u_1(R,\theta) = e^{ikR}\frac{f_0(\theta)}{(kR)^{1/2}}\left(ik - \frac{1}{2R}\right) \tag{4}$$

The parameter k represents the spatial frequency, which we choose to be 2π. For this project we set the function $u_{inc} = e^{ikx}$, which is the spatial factor of the incident plane wave. We also have an unknown function $f_0(\theta)$ that is part of the leading term of the Farfield expansion.

Existence and Uniqueness

We begin by showing that the system (1)–(4) is equivalent to the following system.

$$\Delta U_1 + k^2 U_1 = 0, \qquad \text{in } \Omega^-, \tag{5}$$

$$U_1 = -u_{inc}, \qquad \text{on } \Gamma, \tag{6}$$

$$\partial_r U_1(R,\theta) + \frac{1}{2R}U_1(R,\theta) - ikU_1(R,\theta) = 0. \tag{7}$$

Proof. ($\Rightarrow$) Suppose that u satisfies (1)–(4). Immediately, u also satisfies (5)–(6). Now using (3) and (4) to substitute for $u(R,\theta)$ and $\partial_r u(R,\theta)$,

$$\begin{aligned}\partial_r u(R,\theta) + \frac{1}{2R}u(R,\theta) - iku(R,\theta) \\ &= e^{ikR}\frac{f_0(\theta)}{(kR)^{1/2}}\left(ik - \frac{1}{2R}\right) + \frac{1}{2R}e^{ikR}\frac{f_0(\theta)}{(kR)^{1/2}} - ike^{ikR}\frac{f_0(\theta)}{(kR)^{1/2}} \\ &= e^{ikR}\frac{f_0(\theta)}{(kR)^{1/2}}\left(ik - \frac{1}{2R} - ik + \frac{1}{2R}\right) = e^{ikR}\frac{f_0(\theta)}{(kR)^{1/2}}\cdot 0 = 0,\end{aligned}$$

so that u satisfies (7) as well. Then any smooth function u satisfying (1)–(4) also satisfies (5)–(7).

($\Leftarrow$) Now suppose that u satisfies (5)–(7). Immediately, u also satisfies (1)–(2). Choosing $f_0(\theta) = u(R,\theta)(kR)^{1/2}e^{-ikR}$, we have

$$e^{ikR}\frac{f_0(\theta)}{(kR)^{1/2}} = e^{ikR}\frac{u(R,\theta)(kR)^{1/2}e^{-ikR}}{(kR)^{1/2}} = u(R,\theta),$$

so that u satisfies (3). Additionally, since u satisfies (7), we have

$$\partial_r u(R,\theta) = -\frac{1}{2R}u(R,\theta) + iku(R,\theta) = u(R,\theta)\left(ik - \frac{1}{2R}\right).$$

Then taking the right side of (4), we have

$$\begin{aligned}e^{ikR}\frac{f_0(\theta)}{(kR)^{1/2}}\left(ik - \frac{1}{2R}\right) &= e^{ikR}\frac{u(R,\theta)(kR)^{1/2}e^{-ikR}}{(kR)^{1/2}}\left(ik - \frac{1}{2R}\right) \\ &= u(R,\theta)\left(ik - \frac{1}{2R}\right) = \partial_r u(R,\theta),\end{aligned}$$

and hence u satisfies (4). Then any smooth function u satisfying (5)–(7) also satisfies (1)–(4).

Thus (1)–(4) and (5)–(7) are equivalent systems. Therefore, since the cited paper [2] shows existence and uniqueness of solutions for (5)–(7), we also have existence and uniqueness of solutions for (1)–(4) as well. □

Initial Observations

Our goal is to approximate u_{sc} by the function u_1 in the system (1)–(4). The exact solution for u_{sc} in the original boundaryless problem (an infinite domain with no artificial boundary) is given by the infinite series

$$u_{sc}(r,\theta) = -\sum_{n=0}^{\infty}\epsilon_n i^n \frac{J_n(kr_0)}{H_n^{(1)}(kr_0)}H_n^{(1)}(kr)\cos(n\theta),$$

where $\epsilon_0 = 1$, $\epsilon_n = 2$ for $n \geq 1$, J_n is the Bessel function of order n, and $H_n^{(1)}$ is the Hankel function of first kind of order n. The following MATLAB function approximates $u_{sc}(r,\theta)$ for a given number of series terms.

```
function uscex = ExactSoln(r, theta, r0, k, N)
    % Calculate the exact solution to the scattering wave using N series terms.

    % Build index and epsilon matrices.
    n = 0:N;
    epsilon = 2*ones(1,N+1); epsilon(1) = 1;

    % Calculate N terms of the infinite series solution.
    kr0 = k * r0;
    uscex = -sum(epsilon .* 1i.^n .* besselj(n,kr0) .* besselh(n,k*r) ...
                                          .* cos(n*theta) ./ besselh(n,kr0));
end
```

To calculate u_{sc} on Ω^-, we need to generate a discrete polar grid. Let PPW be the desired points-per-wavelength in our grid. Then the wavelength of u_{sc} is $\lambda = \frac{2\pi}{k}$ since k is the spatial frequency. The desired number of interior points N for the radial direction is thus

$$N = \frac{R - r_0}{\lambda} PPW = \frac{(R - r_0)k}{2\pi} PPW$$

Likewise, a good number of points M for the angular direction is

$$M = \frac{2\pi r_0 k}{2\pi} PPW = r_0 k PPW$$

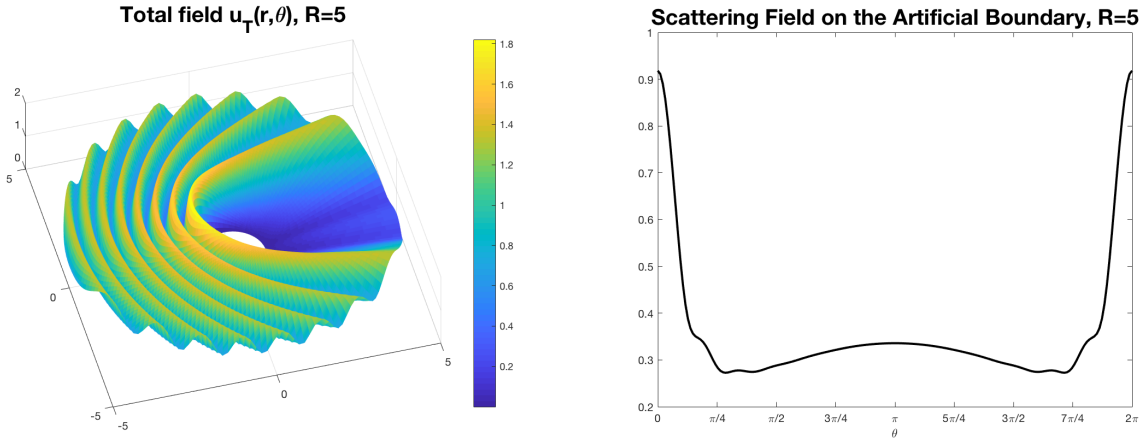
The following code generates the polar grids $r_{0:N+1}$ and $\theta_{0:M}$ and translates them into the Cartesian grids $x_{i,j} = r_i \cos(\theta_j)$ and $y_{i,j} = r_i \sin(\theta_j)$ for plotting convenience.

```
% Assume r0, k, R, and PPW are already defined.

N = ceil(PPW * (R-r0) * k / (2*pi));          % Interior radial divisions
M = ceil(PPW * r0 * k);                       % Angular divisions

r = linspace(r0, R, N+2);                     % N interior, 2 boundary
theta = linspace(0, 2*pi, M+1);               % M interior, 1 overlap

x = r' * cos(theta);
y = r' * sin(theta);
```

Now we can use our `ExactSoln()` function to analytically approximate $u_{sc}(r, \theta)$. We also compute the incident wave $u_{inc} = e^{ikx}$, and the total pressure field $u_T = u_{sc} + u_{inc}$.

```
u_inc = exp(1i*k*x);
u_sc = zeros(N, M);
for i=1:N
    for j=1:M
        u_sc(i,j) = ExactSoln(r(i), theta(j), r0, k, 30);
end
u_T = u_inc + u_sc;
```

We will be particularly interested in how our numerical approximation u_1 matches up with u_{sc} at the artificial boundary. Below, we plot u_T on Ω^- and $u_{sc}(R,\theta)$ for $R = 5$.



The Algorithm

We use centered difference quotients to approximate all partial derivatives in the system, including boundary points. Let U_i^j be the desired numerical approximation to $u_1(r_i, \theta_j)$. Our goal is to set up a linear system $A\mathbf{U} = \mathbf{F}$ to solve for the U_i^j. We proceed from the scatterer $r = r_0$ to the artificial boundary $r = R$.

Boundary Terms: Obstacle Boundary

To begin, we discretize (2), which provides information about U_0^j for each j. As the Cartesian grid for x is given by $x_{ij} = r_i \cos(\theta_j)$, (2) becomes

$$U_0^j = u_1(r_0, \theta_j) = (-u_{inc})_0^j = \left(-e^{ikx}\right)_0^j = -e^{ikr_0\cos(\theta_j)}. \tag{8}$$

So for these terms, we will have nonzero entries in the forcing term $\mathbf{F}$.

Interior Terms

In polar coordinates, the laplacian operator produces three terms:

$$\Delta_{r,\theta} u = u_{rr} + \frac{1}{r} u_r + \frac{1}{r^2} u_{\theta\theta}.$$

Using centered difference quotients for each partial derivative in (1), we obtain the following 5-point stencil to approximate u_1 on the interior domain Ω^- (writing u for u_1 here for clarity).

$$0 = \left(\Delta u + k^2 u\right)_i^j = (u_{rr})_i^j + \left(\frac{1}{r}u_r\right)_i^j + \left(\frac{1}{r^2}u_{\theta\theta}\right)_i^j + \left(k^2 u\right)_i^j$$

$$= \frac{U_{i-1}^j - 2U_i^j + U_{i+1}^j}{(\delta r)^2} + \frac{U_{i+1}^j - U_{i-1}^j}{r_i 2\delta r} + \frac{U_i^{j-1} - 2U_i^j + U_i^{j+1}}{r_i^2(\delta\theta)^2} + k^2 U_i^j$$

$$= U_{i-1}^j \left(\frac{1}{(\delta r)^2} - \frac{1}{r_i 2\delta r}\right) + U_{i+1}^j \left(\frac{1}{(\delta r)^2} + \frac{1}{r_i 2\delta r}\right) + U_i^{j-1}\frac{1}{r_i^2(\delta\theta)^2} + U_i^{j+1}\frac{1}{r_i^2(\delta\theta)^2}$$
$$+ U_i^j \left(-\frac{2}{(\delta r)^2} - \frac{2}{r_i^2(\delta\theta)^2} + k^2\right)$$

$$= U_{i-1}^j \left(\frac{2r_i - \delta r}{2r_i(\delta r)^2}\right) + U_{i+1}^j \left(\frac{2r_i + \delta r}{2r_i(\delta r)^2}\right) + U_i^{j-1}\frac{1}{r_i^2(\delta\theta)^2} + U_i^{j+1}\frac{1}{r_i^2(\delta\theta)^2}$$
$$+ U_i^j \left(k^2 - 2\frac{r_i^2(\delta\theta)^2 + (\delta r)^2}{r_i^2(\delta r)^2(\delta\theta)^2}\right) \tag{9}$$

This equation does not yield any nonzero forcing terms, unlike the inner boundary layer.

Boundary Terms: Artificial Boundary

To discretize (3) and (4), we have an extra challenge because $f_0(\theta)$ is unknown. From (3),

$$U_{N+1}^j = u_1(r_{N+1}, \theta_j) = u_1(R, \theta_j) = e^{ikR}\frac{f_0(\theta_j)}{(kR)^{1/2}}$$
$$\implies \qquad f_0(\theta_j) = U_{N+1}^j \frac{(kR)^{1/2}}{e^{ikR}}$$

To get a centered difference approximation for the r derivative in (4), we introduce a layer of "ghost points" U_{N+2}^j. Then substituting for $f_0(\theta_j)$ from the previous equation, we have

$$\frac{U_{N+2}^j - U_N^j}{2\delta r} = \partial_r u_1(R, \theta_j) = e^{ikR}\frac{f_0(\theta_j)}{(kR)^{1/2}}\left(ik - \frac{1}{2R}\right) = U_{N+1}^j \left(ik - \frac{1}{2R}\right)$$
$$\implies \quad U_{N+2}^j = 2\delta r \left(ik - \frac{1}{2R}\right) U_{N+1}^j + U_N^j \tag{10}$$

Now using (9) with $i = N+1$, and then substituting in for U^j_{N+2} using (10), we have

$$\begin{aligned}
0 &= U^j_N \left(\frac{2R-\delta r}{2R(\delta r)^2}\right) + U^j_{N+2}\left(\frac{2R+\delta r}{2R(\delta r)^2}\right) \\
&\quad + U^{j-1}_{N+1}\frac{1}{R^2(\delta\theta)^2} + U^{j+1}_{N+1}\frac{1}{R^2(\delta\theta)^2} + U^j_{N+1}\left(k^2 - 2\frac{R^2(\delta\theta)^2+(\delta r)^2}{R^2(\delta r)^2(\delta\theta)^2}\right)
\end{aligned}$$

$$\begin{aligned}
0 &= U^j_N \left(\frac{2R-\delta r}{2R(\delta r)^2}\right) + \left(2\delta r\left(ik - \frac{1}{2R}\right)U^j_{N+1} + U^j_N\right)\left(\frac{2R+\delta r}{2R(\delta r)^2}\right) \\
&\quad + U^{j-1}_{N+1}\frac{1}{R^2(\delta\theta)^2} + U^{j+1}_{N+1}\frac{1}{R^2(\delta\theta)^2} + U^j_{N+1}\left(k^2 - 2\frac{R^2(\delta\theta)^2+(\delta r)^2}{R^2(\delta r)^2(\delta\theta)^2}\right)
\end{aligned}$$

$$\begin{aligned}
0 &= U^j_N \frac{2}{(\delta r)^2} + U^j_{N+1}\left(ik - \frac{1}{2R}\right)\left(\frac{2R+\delta r}{R\delta r}\right) \\
&\quad + U^{j-1}_{N+1}\frac{1}{R^2(\delta\theta)^2} + U^{j+1}_{N+1}\frac{1}{R^2(\delta\theta)^2} + U^j_{N+1}\left(k^2 - 2\frac{R^2(\delta\theta)^2+(\delta r)^2}{R^2(\delta r)^2(\delta\theta)^2}\right)
\end{aligned}$$

$$\begin{aligned}
0 &= U^j_N \frac{2}{(\delta r)^2} + U^j_{N+1}\left(k^2 - 2\frac{R^2(\delta\theta)^2+(\delta r)^2}{R^2(\delta r)^2(\delta\theta)^2} + \left(ik - \frac{1}{2R}\right)\left(\frac{2R+\delta r}{R\delta r}\right)\right) \\
&\quad + U^{j-1}_{N+1}\frac{1}{R^2(\delta\theta)^2} + U^{j+1}_{N+1}\frac{1}{R^2(\delta\theta)^2}. \qquad (11)
\end{aligned}$$

Matrix Construction

Recall that $U^j_i = u(r_i, \theta_j)$ for $i = 0, 1, \ldots, N+1$ and $j = 0, 1, \ldots, M$ (N interior radial points, plus one boundary layer on each side, and $M+1$ angular points). In addition, since $\theta_0 \equiv \theta_M \equiv 0 \mod 2\pi$, we have $U^0_i = U^M_i$, and hence we need only consider $j = 0, 1, \ldots, M-1$.

Because the boundary conditions are given for $r = r_0$ and $r = R$, we set up the vector $\mathbf{U}$ of unknowns so that the terms representing u_1 at the same radius are adjacent.

$$\mathbf{U} = [U^0_0, U^1_0, \ldots, U^{M-1}_0, U^0_1, U^1_1, \ldots\ldots, U^{M-1}_{N+1}]^\mathsf{T},$$

which has $M(N+2)$ elements. To simplify notation, let $\mathbf{U}_{[i]}$ be the subset of M elements of $\mathbf{U}$ that correspond to radius $r = r_i$: $\mathbf{U}_{[i]} = [U^0_i, U^1_i, \ldots, U^{M-1}_i]^\mathsf{T}$. Then

$$\mathbf{U} = [\mathbf{U}^\mathsf{T}_{[0]}, \mathbf{U}^\mathsf{T}_{[1]}, \ldots, \mathbf{U}^\mathsf{T}_{[N+1]}]^\mathsf{T}.$$

Note that each of the vector components of $\mathbf{U}$ have M elements and that there are $N+2$ such blocks. We construct A by considering each block component.

The boundary condition given in (8) yields the equation

$$I\mathbf{U}_{[0]} = \begin{bmatrix} -e^{ikr_0\cos(\theta_0)} \\ -e^{ikr_0\cos(\theta_1)} \\ \vdots \\ -e^{ikr_0\cos(\theta_{M-1})} \end{bmatrix} = -e^{ikr_0\cos(\boldsymbol{\theta})}, \tag{12}$$

where I is the $M \times M$ identity matrix and $\boldsymbol{\theta} = [\theta_0, \ldots, \theta_{M-1}]^\mathsf{T}$.

The interior terms ($r_0 < r < R$, $i = 1, 2, \ldots, N$) are governed by (9), which becomes

$$D_i^- \mathbf{U}_{[i-1]} + T_i \mathbf{U}_{[i]} + D_i^+ \mathbf{U}_{[i+1]} = \mathbf{0}, \tag{13}$$

where D_i^- and D_i^+ are diagonal and T_i is nearly tridiagonal, and each matrix is $M \times M$:

$$D_i^- = \frac{2r_i - \delta r}{2r_i(\delta r)^2} I, \qquad D_i^+ = \frac{2r_i + \delta r}{2r_i(\delta r)^2} I,$$

$$T_i = \begin{bmatrix} k^2 - 2\frac{r_i^2(\delta\theta)^2 + (\delta r)^2}{r_i^2(\delta r)^2(\delta\theta)^2} & \frac{1}{r_i^2(\delta\theta)^2} & & & \frac{1}{r_i^2(\delta\theta)^2} \\ \frac{1}{r_i^2(\delta\theta)^2} & k^2 - 2\frac{r_i^2(\delta\theta)^2 + (\delta r)^2}{r_i^2(\delta r)^2(\delta\theta)^2} & \frac{1}{r_i^2(\delta\theta)^2} & & \\ & \ddots & \ddots & \ddots & \\ & & \frac{1}{r_i^2(\delta\theta)^2} & k^2 - 2\frac{r_i^2(\delta\theta)^2 + (\delta r)^2}{r_i^2(\delta r)^2(\delta\theta)^2} & \frac{1}{r_i^2(\delta\theta)^2} \\ \frac{1}{r_i^2(\delta\theta)^2} & & & \frac{1}{r_i^2(\delta\theta)^2} & k^2 - 2\frac{r_i^2(\delta\theta)^2 + (\delta r)^2}{r_i^2(\delta r)^2(\delta\theta)^2} \end{bmatrix}.$$

The corner terms of T_i are for $j = 0$ and $j = M - 1$, and are consistent since $U_i^{-1} = U_i^{M-1}$ and $U_i^M = U_i^0$.

Finally, (11) yields the matrix equation

$$\tilde{D}_{N+1}\mathbf{U}_{[N]} + \tilde{T}_{N+1}\mathbf{U}_{[N+1]} = \mathbf{0}, \quad \text{where} \tag{14}$$

$$\tilde{D}_{N+1} = \frac{2}{(\delta r)^2} I, \qquad \tilde{T}_{N+1} = T_{N+1} + \left(ik - \frac{1}{2R}\right)\left(\frac{2R + \delta r}{R\delta r}\right) I.$$

Combining (12)–(14) yields the following linear system, which approximates the solution u_1 to the system (1)–(4).

$$A\mathbf{U} = \begin{bmatrix} I & & & & & \\ D_1^- & T_1 & D_1^+ & & & \\ & D_2^- & T_2 & D_2^+ & & \\ & & \ddots & \ddots & \ddots & \\ & & & D_N^- & T_N & D_N^+ \\ & & & & \tilde{D}_{N+1} & \tilde{T}_{N+1} \end{bmatrix} \begin{bmatrix} \mathbf{U}_{[0]} \\ \mathbf{U}_{[1]} \\ \mathbf{U}_{[2]} \\ \vdots \\ \mathbf{U}_{[N]} \\ \mathbf{U}_{[N+1]} \end{bmatrix} = \begin{bmatrix} -e^{ikr_0\cos(\boldsymbol{\theta})} \\ \mathbf{0} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix} = \mathbf{F}$$

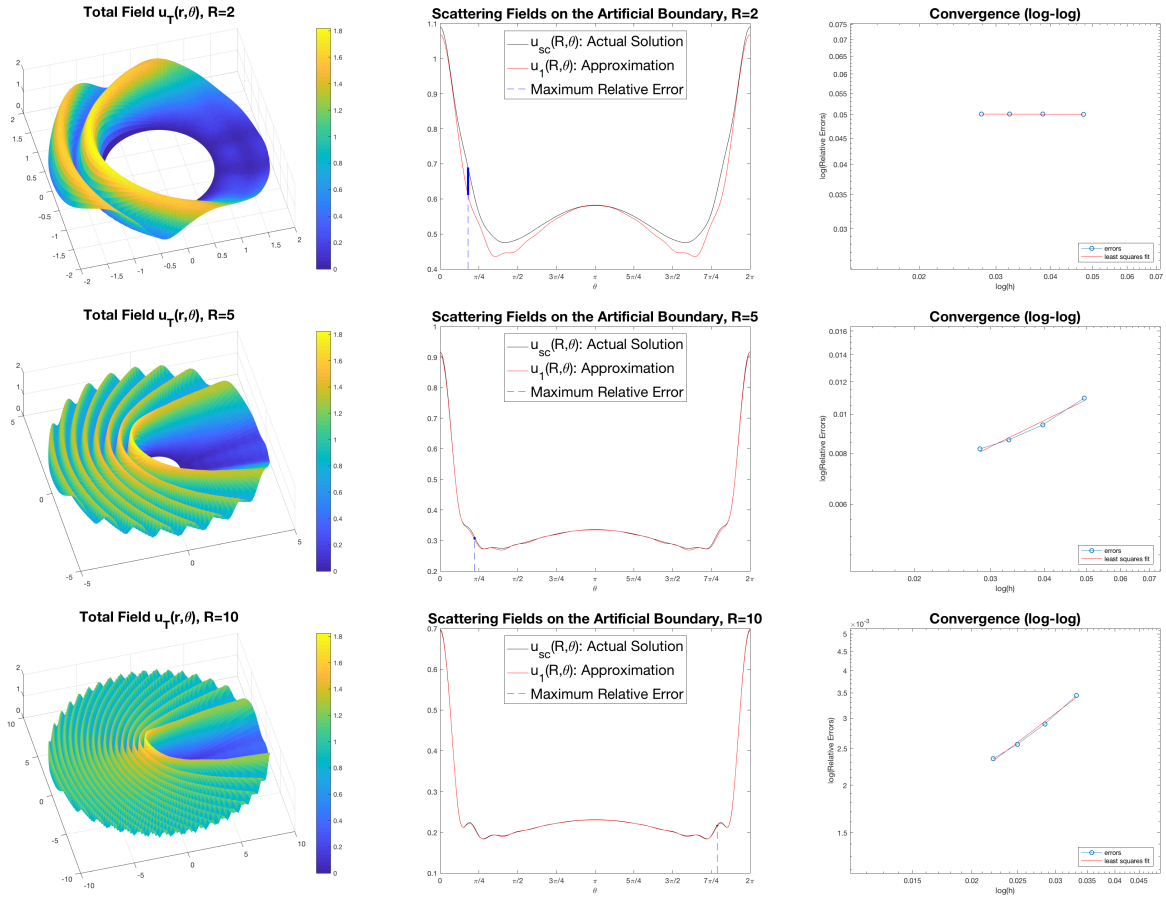
Implementation

The following code builds the matrix A, the vector $\mathbf{F}$, and solves for the vector $\mathbf{U}$.

```
% Assume R, r0, k, M, and N are already defined.
%% Initialize grid spacing and common variables. ----------------------------------
dr = (R-r0)/(N+1);
dr2 = dr^2;

dt = 2*pi/M;
dt2 = dt^2;

I = eye(M)

%% Set up the finite difference system AU = F for this BVP. -------------------------
system_size = M*(N+2);

% Forcing term (mostly zeros).
F = zeros(system_size,1);
F(1:M) = -exp(1i * k * r0 * cos(theta(1:M)));

% Initialize A with sparse memory allocation.
A = spalloc(system_size, system_size, 5*M*(N+1));

% Inner boundary terms.
A(1:M,1:M) = I;

% Interior terms.
for n=1:N
    ri = r(n+1);
    ri2 = ri^2;

    % Construct T.
    T = I * (k2 - 2*(ri2*dt2 + dr2)/(ri2*dr2*dt2));
    off_diag = 1 / (ri2*dt2);
    off_diags = ones(1, M-1) * off_diag;
    T = T + diag(off_diags, -1) + diag(off_diags, 1);
    T(1,end) = off_diag;
    T(end,1) = off_diag;

    % D^- (subdiagonal)
    A(n*M+1:(n+1)*M,(n-1)*M+1:n*M) = I * (2*ri - dr) / (2*ri*dr2);
    % T (main diagonal)
    A(n*M+1:(n+1)*M,n*M+1:(n+1)*M) = T;
    % D^+ (superdiagonal)
    A(n*M+1:(n+1)*M,(n+1)*M+1:(n+2)*M) = I*(2*ri + dr)/(2*ri*dr2);
end

% Continued on next page...
```

```
% Outer boundary terms.
A(end-M+1:end,end-(2*M)+1:end-M) = 2 * I / dr2;              % D tilde

T = I * (k2 - 2*(R2*dt2 + dr2)/(R2*dr2*dt2));
off_diag = 1/(R2*dt2);
off_diags = ones(1, M-1) * off_diag;
T = T + diag(off_diags, -1) + diag(off_diags, 1);
T(1,end) = off_diag;
T(end,1) = off_diag;
T = T + I * (1i*k - 1/(2*R))*((2*R + dr)/(R*dr));
A(end-M+1:end,end-M+1:end) = T;                              % T tilde

%% Solve the system and unpack results. ------------------------------------------

u_1 = reshape(conj(A\F), M, N+2)';
u_1 = [u_1, u_1(:,1)];                        % Connect theta=0 to theta=2pi.
```

Below we plot approximations of the total pressure $u_T = u_{inc} + u_1$, and comparisons of u_{sc} to u_1 at the artificial boundary, for $R = 2, 5$, and 10 with $PPW = 30$. The log-log plots on the right estimate the order of this method by varying PPW.



The following table provides more detail on the log-log convergence plots.

R	PPW	h	Relative Error	Ratio	Observed Order	Least Squares Fit
	20	0.04762	5.00931e-02			
	25	0.03846	5.01223e-02	0.99942	-0.00273	
2	30	0.03226	5.01363e-02	0.99972	-0.00159	$0.04976h^{\mathbf{-0.0022034}}$
	35	0.02778	5.01542e-02	0.99964	-0.00238	
	20	0.04938	1.09471e-02			
	25	0.03960	9.40205e-03	1.16433	0.68949	
5	30	0.03306	8.63969e-03	1.08824	0.46804	$0.0516219h^{\mathbf{0.520871}}$
	35	0.02837	8.20923e-03	1.05244	0.33410	
	30	0.03321	3.44569e-03			
	35	0.02848	2.89900e-03	1.18858	1.12457	
10	40	0.02493	2.56308e-03	1.13106	0.92504	$0.0870084h^{\mathbf{0.95186}}$
	45	0.02217	2.34770e-03	1.09174	0.74716	

Analysis

The entire derivation for this finite difference method relies on second-order centered difference quotients, even for the the artificial boundary, so we might hope for the overall algorithm to be second order as well. However, in the best case ($R = 10$) the observed order i still less than 1, and in the worst case ($R = 2$) increasing PPW actually increases the relative error. There are a few things that could be keeping the convergence so low:

- The solution at the boundary only using a single term of the Farfield expansion, $f_0(\theta)$. Though it is the leading term, it is still only a single term, so (3) and (4) could not give completely accurate results, even if our numerical method were perfect.
- The fact that the A is *almost* 5-diagonal, but not quite because of the corner terms in the T_i blocks, which makes the system slightly more difficult to solve numerically than a purely banded matrix.
- The radius R of the artificial boundary being too small to get good numerical approximations.

This last point is especially important, since we can see from the table that increasing R increased the order significantly. In fact, another experiment with $R = 30$ yields a least squares fit of $E(h) = 3.37574h^{2.12023}$. Thus we can say with some confidence that the problem itself is ill-posed for small R, and better suited to numerical methods for larger R.

We continue our analysis after experimenting on a modified version of the original BVP.

BVP with an Additional Farfield Expansion Term

By adding an additional term to the Farfield expansion of the solution, we obtain a new, modified BVP.

$$\Delta_{r,\theta} u_2 + k^2 u_2 = 0, \qquad \text{in } \Omega^-, \tag{15}$$

$$u_2 = -u_{inc}, \qquad \text{on } \Gamma, \tag{16}$$

$$u_2(R,\theta) = \frac{e^{ikR}}{(kR)^{1/2}} \left(f_0(\theta) + \frac{f_1(\theta)}{kR} \right) \tag{17}$$

$$\partial_r u_2(R,\theta) = \frac{e^{ikR}}{(kR)^{1/2}} \left(\left(ik - \frac{1}{2R} \right) f_0(\theta) + \left(ik - \frac{3}{2R} \right) \frac{f_1(\theta)}{kR} \right) \tag{18}$$

$$2i f_1(\theta) = \frac{1}{4} f_0(\theta) + f_0''(\theta) \tag{19}$$

This system has the same discretization for the inner boundary ($r = r_0$) and the interior points as the previous system, but we need to deal with the outer boundary ($r = R$) differently because of the additional unknown function $f_1(\theta)$.

Thus we expand the vector $\mathbf{U}$ of unknowns to have a total of $M(N+4)$ entries.

$$\mathbf{U} = [\mathbf{U}_{[0]}^{\mathsf{T}}, \mathbf{U}_{[1]}^{\mathsf{T}}, \ldots, \mathbf{U}_{[N+1]}^{\mathsf{T}}, \mathbf{f}_0^{\mathsf{T}}, \mathbf{f}_1^{\mathsf{T}}]^{\mathsf{T}}, \quad \text{where}$$

$$\mathbf{f}_i = [f_i(\theta_0), f_i(\theta_1), \ldots, f_i(\theta_{M-1})]^{\mathsf{T}}.$$

The discretization of (17) is given by

$$U_{N+1}^j = u_2(R,\theta_j) = \frac{e^{ikR}}{(kR)^{1/2}} f_0(\theta_j) + \frac{e^{ikR}}{(kR)^{3/2}} f_1(\theta_j),$$

which translates into the linear system

$$I\mathbf{U}_{[N+1]} + E_0 \mathbf{f}_0 + E_1 \mathbf{f}_1 = \mathbf{0}, \tag{20}$$

$$E_0 = -\frac{e^{ikR}}{(kR)^{1/2}} I, \quad E_1 = -\frac{e^{ikR}}{(kR)^{3/2}} I = \frac{1}{kR} E_0. \tag{21}$$

Next, to discretize (19), we use a centered difference quotient to approximate f_0''.

$$\begin{aligned} 2i f_1(\theta_j) &= \frac{1}{4} f_0(\theta_j) + \frac{f_0(\theta_{j-1}) - 2f_0(\theta_j) + f_0(\theta_{j+1})}{(\delta\theta)^2} \\ &= \frac{1}{(\delta\theta)^2} f_0(\theta_{j-1}) + \left(\frac{1}{4} - \frac{2}{(\delta\theta)^2} \right) f_0(\theta_j) + \frac{1}{(\delta\theta)^2} f_0(\theta_{j+1}), \end{aligned}$$

which gives the matrix equation

$$T_f \mathbf{f}_0 - 2iI\mathbf{f}_1 = \mathbf{0}, \tag{22}$$

where

$$T_f = \begin{bmatrix} \left(\frac{1}{4} - \frac{2}{(\delta\theta)^2}\right) & \frac{1}{(\delta\theta)^2} & & & \frac{1}{(\delta\theta)^2} \\ \frac{1}{(\delta\theta)^2} & \left(\frac{1}{4} - \frac{2}{(\delta\theta)^2}\right) & \frac{1}{(\delta\theta)^2} & & \\ & \ddots & \ddots & \ddots & \\ & & \frac{1}{(\delta\theta)^2} & \left(\frac{1}{4} - \frac{2}{(\delta\theta)^2}\right) & \frac{1}{(\delta\theta)^2} \\ \frac{1}{(\delta\theta)^2} & & & \frac{1}{(\delta\theta)^2} & \left(\frac{1}{4} - \frac{2}{(\delta\theta)^2}\right) \end{bmatrix}$$

To handle (17) and (18), we introduce the fictitious layer of points U^j_{N+2} again. Then approximating (18) with a centered difference quotient yields the equation

$$\frac{U^j_{N+2} - U^j_N}{2\delta r} = \frac{e^{ikR}}{(kR)^{1/2}}\left(ik - \frac{1}{2R}\right) f_0(\theta_j) + \frac{e^{ikR}}{(kR)^{3/2}}\left(ik - \frac{3}{2R}\right) f_1(\theta_j)$$

$$U^j_{N+2} = \frac{2\delta r e^{ikR}}{(kR)^{1/2}}\left(ik - \frac{1}{2R}\right) f_0(\theta_j) + \frac{2\delta r e^{ikR}}{(kR)^{3/2}}\left(ik - \frac{3}{2R}\right) f_1(\theta_j) + U^j_N.$$

Now using (9) with $i = N+1$, and substituting in for U^j_{N+2} from the previous equation, we have

$$\begin{aligned} 0 &= U^j_N\left(\frac{2R-\delta r}{2R(\delta r)^2}\right) + U^j_{N+2}\left(\frac{2R+\delta r}{2R(\delta r)^2}\right) \\ &\quad + U^{j-1}_{N+1}\frac{1}{R^2(\delta\theta)^2} + U^{j+1}_{N+1}\frac{1}{R^2(\delta\theta)^2} + U^j_{N+1}\left(k^2 - 2\frac{R^2(\delta\theta)^2 + (\delta r)^2}{R^2(\delta r)^2(\delta\theta)^2}\right) \\ &= U^j_N\left(\frac{2R-\delta r}{2R(\delta r)^2}\right) + \left(\frac{2\delta r e^{ikR}}{(kR)^{1/2}}\left(ik - \frac{1}{2R}\right) f_0(\theta_j) + \frac{2\delta r e^{ikR}}{(kR)^{3/2}}\left(ik - \frac{3}{2R}\right) f_1(\theta_j) + U^j_N\right)\frac{2R+\delta r}{2R(\delta r)^2} \\ &\quad + U^{j-1}_{N+1}\frac{1}{R^2(\delta\theta)^2} + U^{j+1}_{N+1}\frac{1}{R^2(\delta\theta)^2} + U^j_{N+1}\left(k^2 - 2\frac{R^2(\delta\theta)^2 + (\delta r)^2}{R^2(\delta r)^2(\delta\theta)^2}\right) \\ &= U^j_N\frac{2}{(\delta r)^2} + f_0(\theta_j)\frac{e^{ikR}(2R+\delta r)}{(kR)^{1/2}R\delta r}\left(ik - \frac{1}{2R}\right) + f_1(\theta_j)\frac{e^{ikR}(2R+\delta r)}{(kR)^{3/2}R\delta r}\left(ik - \frac{3}{2R}\right) \\ &\quad + U^{j-1}_{N+1}\frac{1}{R^2(\delta\theta)^2} + U^{j+1}_{N+1}\frac{1}{R^2(\delta\theta)^2} + U^j_{N+1}\left(k^2 - 2\frac{R^2(\delta\theta)^2 + (\delta r)^2}{R^2(\delta r)^2(\delta\theta)^2}\right). \end{aligned}$$

The matrix equation for this is similar to the corresponding equation for the first BVP, but with an extra term for the $f_1(\theta_j)$.

$$\tilde{D}_{N+1}\mathbf{U}_{[N]} + T_{N+1}\mathbf{U}_{[N+1]} + B_0\mathbf{f}_0 + B_1\mathbf{f}_1 = \mathbf{0}, \tag{23}$$

where $\tilde{D}_{N+1}$, and T_{N+1} are the same that were used for the previous BVP, and

$$B_0 = \frac{e^{ikR}(2R+\delta r)}{(kR)^{1/2}R\delta r}\left(ik - \frac{1}{2R}\right) I, \quad B_1 = \frac{e^{ikR}(2R+\delta r)}{(kR)^{3/2}R\delta r}\left(ik - \frac{3}{2R}\right) I$$

Finally, using (20)–(23), we construct our large linear system to approximate solutions of (15)-(19).

$$A\mathbf{U} = \begin{bmatrix} I & & & & & & & & \\ D_1^- & T_1 & D_1^+ & & & & & & \\ & D_2^- & T_2 & D_2^+ & & & & & \\ & & \ddots & \ddots & \ddots & & & & \\ & & & D_N^- & T_N & D_N^+ & & & \\ & & & & \tilde{D}_{N+1} & T_{N+1} & B_0 & B_1 & \\ & & & & & I & E_0 & E_1 & \\ & & & & & & T_f & -2iI & \end{bmatrix} \begin{bmatrix} \mathbf{U}_{[0]} \\ \mathbf{U}_{[1]} \\ \mathbf{U}_{[2]} \\ \vdots \\ \mathbf{U}_{[N]} \\ \mathbf{U}_{[N+1]} \\ \mathbf{f}_0 \\ \mathbf{f}_1 \end{bmatrix} = \begin{bmatrix} -e^{ikr_0\cos(\boldsymbol{\theta})} \\ \mathbf{0} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix} = \mathbf{F}$$

Implementation

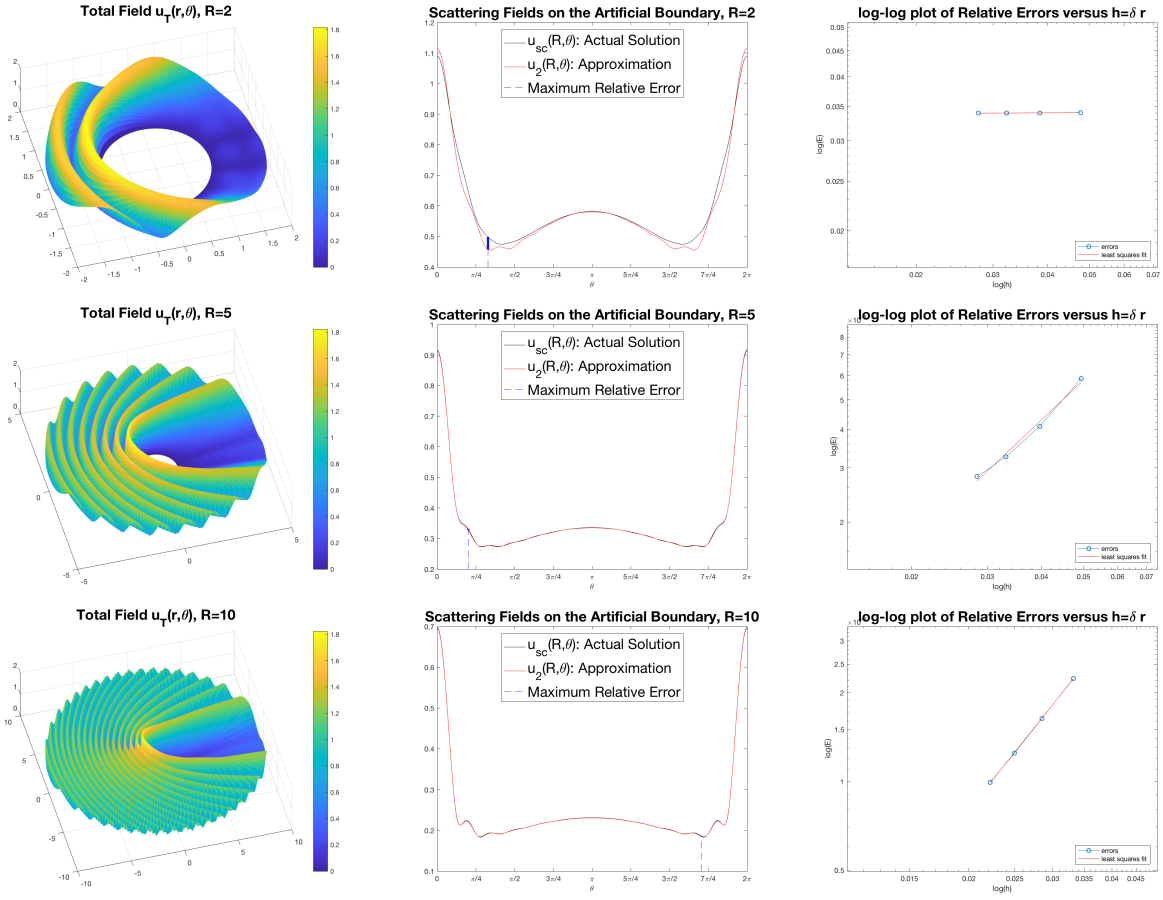
The code for this algorithm is very similar to the code for the original BVP since the first MN rows of A are the same as in the previous case. However, the last three M-block rows are different, and are constructed explicitly.

```
% Assume R, r0, k, M, and N are already defined.
%% Initialize grid spacing and common variables. -------------------------------------
dr = (R-r0)/(N+1);
dr2 = dr^2;

dt = 2*pi/M;
dt2 = dt^2;

I = eye(M)

%% Set up the finite difference system AU = F for this BVP. -------------------------
system_size = M*(N+4);

% Forcing term (mostly zeros).
F = zeros(system_size,1);
F(1:M) = -exp(1i * k * r0 * cos(theta(1:M)));

% Initialize A with sparse memory allocation.
A = spalloc(system_size, system_size, M*(5*N+14));

% Inner boundary terms.
A(1:M,1:M) = I;

% Continued on next page...
```

```
% Interior terms.
for n=1:N
    ri = r(n+1);
    ri2 = ri^2;

    % Construct T.
    T = I * (k2 - 2*(ri2*dt2 + dr2)/(ri2*dr2*dt2));
    off_diag = 1 / (ri2*dt2);
    off_diags = ones(1, M-1) * off_diag;
    T = T + diag(off_diags, -1) + diag(off_diags, 1);
    T(1,end) = off_diag;
    T(end,1) = off_diag;

    % D^- (subdiagonal)
    A(n*M+1:(n+1)*M,(n-1)*M+1:n*M) = I * (2*ri - dr) / (2*ri*dr2);
    % T (main diagonal)
    A(n*M+1:(n+1)*M,n*M+1:(n+1)*M) = T;
    % D^+ (superdiagonal)
    A(n*M+1:(n+1)*M,(n+1)*M+1:(n+2)*M) = I*(2*ri + dr)/(2*ri*dr2);
end

% Outer boundary terms.
A(end-(3*M)+1:end-(2*M),end-(4*M)+1:end-(3*M)) = 2 * I / dr2;% D tilde

T = I * (k2 - 2*(R2*dt2 + dr2)/(R2*dr2*dt2));
off_diag = 1/(R2*dt2);
off_diags = ones(1, M-1) * off_diag;
T = T + diag(off_diags, -1) + diag(off_diags, 1);
T(1,end) = off_diag;
T(end,1) = off_diag;
A(end-(3*M)+1:end-(2*M),end-(3*M)+1:end-(2*M)) = T;             % T

B_0 = I * exp(1i*k*R) * (2*R + dr) * (1i*k - 1/(2*R)) ...       % B_0
                                              / (R*dr*(k*R)^(1/2));
B_1 = I * exp(1i*k*R) * (2*R + dr) * (1i*k - 3/(2*R)) ...       % B_1
                                              / (R*dr*(k*R)^(3/2));
A(end-(3*M)+1:end-(2*M),end-(2*M)+1:end-M) = B_0;
A(end-(3*M)+1:end-(2*M),end-M+1:end) = B_1;

% Farfield expansion terms.
A(end-(2*M)+1:end-M,end-(3*M)+1:end-(2*M)) = I;
E_0 = -exp(1i*k*R) * I / sqrt(k*R);                             % E_0
A(end-(2*M)+1:end-M,end-(2*M)+1:end-M) = E_0;
A(end-(2*M)+1:end-M,end-M+1:end) = E_0 / (k*R);                 % E_1

% Continued on next page...
```

```
T = I * (.25 - (2/dt2));
off_diag = 1 / dt2;
off_diags = ones(1, M-1) * off_diag;
T = T + diag(off_diags, -1) + diag(off_diags, 1);
T(1,end) = off_diag;
T(end,1) = off_diag;
A(end-M+1:end,end-(2*M)+1:end-M) = T;                              % T_f

A(end-M+1:end,end-M+1:end) = -2i * I;

%% Solve the system and unpack results. ------------------------------------------

u_2 = reshape(conj(A\F), M, N+4)';
u_2 = u_2(1:end-2,:);                          % Strip off Farfield terms.
u_2 = [u_2, u_2(:,1)];                         % Connect theta=0 to theta=2pi.
```

The results for this problem are much better than those of the last problem.



R	PPW	h	Relative Error	Ratio	Observed Order	Least Squares Fit
	20	0.04762	3.40458e-02			
	25	0.03846	3.39990e-02	1.00138	0.00644	
2	30	0.03226	3.39685e-02	1.00090	0.00510	$0.0345542h^{\mathbf{0.00492019}}$
	35	0.02778	3.39574e-02	1.00033	0.00218	
	20	0.04938	5.85684e-03			
	25	0.03960	4.09995e-03	1.42852	1.61614	
5	30	0.03306	3.25846e-03	1.25825	1.27149	$0.310691h^{\mathbf{1.32946}}$
	35	0.02837	2.81395e-03	1.15797	0.95879	
	30	0.03321	2.23639e-03			
	35	0.02848	1.63426e-03	1.36844	2.04183	
10	40	0.02493	1.24666e-03	1.31091	2.03342	$2.12357h^{\mathbf{2.01428}}$
	45	0.02217	9.92050e-04	1.25665	1.94467	

Analysis

This algorithm still performs quite poorly for $R = 2$, confirming our hypothesis that for R sufficiently close to r_0, the problem is ill-posed. However, for $R = 5$ the order exceeds 1, and with $R = 10$ the order finally exceeds 2.

To compare this BVP with the previous problem, we list again the observed orders of convergence for each numerical experiment in the following table.

R	Order for BVP 1	Order for BVP 2
2	-0.00220	0.00492
5	0.52087	1.32946
10	0.95186	2.01428

The order more than doubles in every case when moving from BVP 1 to BVP 2. Since the only difference between the two problems is the number of Farfield expansion terms used to determine the boundary condition at the artificial boundary, we conclude that the most important and numerically sensitive part of this problem is what happens at the artificial boundary. This makes a lot of sense since the artificial boundary does not exist physically, and is only part of the problem to simulate the corresponding boundaryless problem.

There are a few ways to potentially alter or improve the methods that we have presented. Since the biggest issues occur at the artificial boundary, we might replace the discretization of the boundary with a higher-order approximation, such as a 5-point backward difference. However, this would come at the cost of a more cluttered matrix structure that may no longer be close to a banded matrix, though it also eliminates the need for a ghost point. On the other hand, we could add more terms to the Farfield expansion (i.e. introduce $f_2(\theta)$, $f_3(\theta)$, etc.), which was the major reason for an increase in order from BVP 1 to BVP 2. However, the boundary condition in BVP 2 may already be sufficient, since we only really expect

at best second-order convergence because that's how we derived our system. Adding more Farfield expansion terms would also alter the matrix structure, though from our experiments it looks like having one or two non-banded blocks doesn't create any significant problems.

A final note: in the second problem, at least M rows and columns of A could be eliminated by solving for one of $u_2(R, \theta)$, $f_0(\theta)$, or $f_1(\theta)$ in terms of the other two. I thought about doing this, but the algebra got pretty bad pretty fast, and it seemed like a better idea to aim for simplicity. However, solving smaller matrices reduces computational costs, so that would improve speed of the algorithm, and perhaps slightly raise the order as well.

Appendix: Linear System Summary

Linear system the first BVP, (1)–(4):

$$A_1\mathbf{U}_1 = \begin{bmatrix} I & & & & & & \\ D_1^- & T_1 & D_1^+ & & & & \\ & D_2^- & T_2 & D_2^+ & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & D_N^- & T_N & D_N^+ & \\ & & & & \tilde{D}_{N+1} & \tilde{T}_{N+1} & \end{bmatrix} \begin{bmatrix} \mathbf{U}_{[0]} \\ \mathbf{U}_{[1]} \\ \mathbf{U}_{[2]} \\ \vdots \\ \mathbf{U}_{[N]} \\ \mathbf{U}_{[N+1]} \end{bmatrix} = \begin{bmatrix} -e^{ikr_0\cos(\boldsymbol{\theta})} \\ \mathbf{0} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix} = \mathbf{F}_1,$$

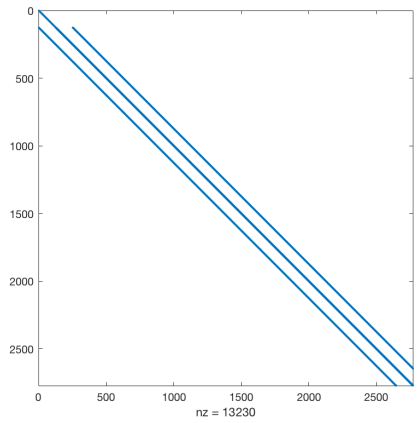
A_1 is $M(N+2) \times M(N+2)$, $\mathbf{U}_1$ and $\mathbf{F}_1$ are $M(N+2) \times 1$.

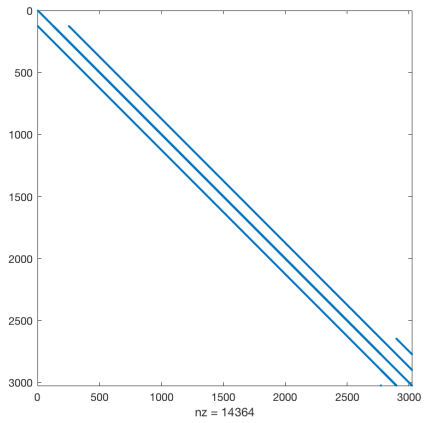
Linear system for the second BVP, (15)–(19):

$$A_2\mathbf{U}_2 = \begin{bmatrix} I & & & & & & & \\ D_1^- & T_1 & D_1^+ & & & & & \\ & D_2^- & T_2 & D_2^+ & & & & \\ & & \ddots & \ddots & \ddots & & & \\ & & & D_N^- & T_N & D_N^+ & & \\ & & & & \tilde{D}_{N+1} & T_{N+1} & B_0 & B_1 \\ & & & & & I & E_0 & E_1 \\ & & & & & & T_f & -2iI \end{bmatrix} \begin{bmatrix} \mathbf{U}_{[0]} \\ \mathbf{U}_{[1]} \\ \mathbf{U}_{[2]} \\ \vdots \\ \mathbf{U}_{[N]} \\ \mathbf{U}_{[N+1]} \\ \mathbf{f}_0 \\ \mathbf{f}_1 \end{bmatrix} = \begin{bmatrix} -e^{ikr_0\cos(\boldsymbol{\theta})} \\ \mathbf{0} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix} = \mathbf{F}_2$$

A_2 is $M(N+4) \times M(N+4)$, $\mathbf{U}_2$ and $\mathbf{F}_2$ are $M(N+4) \times 1$.

Each block matrix is $M \times M$ and each vector is $M \times 1$.
I is the $M \times M$ identity matrix.
$\mathbf{U}_{[i]} = [U_i^0, U_i^1, \ldots, U_i^{M-1}]^\mathsf{T}$
$\mathbf{f}_i = [f_i(\theta_0), f_i(\theta_1), \ldots, f_i(\theta_{M-1})]^\mathsf{T}$ for $i = 0, 1$.
$\boldsymbol{\theta} = [\theta_0, \ldots, \theta_{M-1}]^\mathsf{T}$.
Sparsity patterns of A_1 (left) and A_2 (right) for $R = 2$, $PPW = 20$:





Appendix: Matrix Key

$$D_i^- = \frac{2r_i - \delta r}{2r_i(\delta r)^2} I \qquad D_i^+ = \frac{2r_i + \delta r}{2r_i(\delta r)^2} I$$

$$T_i = \begin{bmatrix} k^2 - 2\frac{r_i^2(\delta\theta)^2+(\delta r)^2}{r_i^2(\delta r)^2(\delta\theta)^2} & \frac{1}{r_i^2(\delta\theta)^2} & & & \frac{1}{r_i^2(\delta\theta)^2} \\ \frac{1}{r_i^2(\delta\theta)^2} & k^2 - 2\frac{r_i^2(\delta\theta)^2+(\delta r)^2}{r_i^2(\delta r)^2(\delta\theta)^2} & \frac{1}{r_i^2(\delta\theta)^2} & & \\ & \ddots & \ddots & \ddots & \\ & & \frac{1}{r_i^2(\delta\theta)^2} & k^2 - 2\frac{r_i^2(\delta\theta)^2+(\delta r)^2}{r_i^2(\delta r)^2(\delta\theta)^2} & \frac{1}{r_i^2(\delta\theta)^2} \\ \frac{1}{r_i^2(\delta\theta)^2} & & & \frac{1}{r_i^2(\delta\theta)^2} & k^2 - 2\frac{r_i^2(\delta\theta)^2+(\delta r)^2}{r_i^2(\delta r)^2(\delta\theta)^2} \end{bmatrix}$$

$$\tilde{D}_{N+1} = \frac{2}{(\delta r)^2} I \qquad \tilde{T}_{N+1} = T_{N+1} + \left(ik - \frac{1}{2R}\right)\left(\frac{2R+\delta r}{R\delta r}\right) I$$

$$E_0 = -\frac{e^{ikR}}{(kR)^{1/2}} I \qquad E_1 = -\frac{e^{ikR}}{(kR)^{3/2}} I = \frac{1}{kR} E_0$$

$$B_0 = \frac{e^{ikR}(2R+\delta r)}{(kR)^{1/2} R\delta r}\left(ik - \frac{1}{2R}\right) I \qquad B_1 = \frac{e^{ikR}(2R+\delta r)}{(kR)^{3/2} R\delta r}\left(ik - \frac{3}{2R}\right) I$$

$$T_f = \begin{bmatrix} \left(\frac{1}{4} - \frac{2}{(\delta\theta)^2}\right) & \frac{1}{(\delta\theta)^2} & & & \frac{1}{(\delta\theta)^2} \\ \frac{1}{(\delta\theta)^2} & \left(\frac{1}{4} - \frac{2}{(\delta\theta)^2}\right) & \frac{1}{(\delta\theta)^2} & & \\ & \ddots & \ddots & \ddots & \\ & & \frac{1}{(\delta\theta)^2} & \left(\frac{1}{4} - \frac{2}{(\delta\theta)^2}\right) & \frac{1}{(\delta\theta)^2} \\ \frac{1}{(\delta\theta)^2} & & & \frac{1}{(\delta\theta)^2} & \left(\frac{1}{4} - \frac{2}{(\delta\theta)^2}\right) \end{bmatrix}$$