MieSolver: a Matlab object-oriented Mie series software

Stuart C. Hawkins,
Department of Mathematics and Statistics,
Macquarie University, Sydney, NSW 2109, Australia
stuart.hawkins@mq.edu.au

Abstract

We provide a user manual for our object-oriented Matlab package MieSolver, which provides efficient and easy to use tools for simulating wave propagation through a heterogeneous configuration of nonidentical circular cylinders. MieSolver allows great flexibility in the physical properties of each cylinder and the cylinders may have opaque or penetrable cores, and an arbitrary number of penetrable layers. The solver is based on the Mie series solution and hence is numerically stable and highly accurate.

1 Installation and verification

Download

Download the CALGO *zip* file and unpack it.

Your system may automatically unpack the archive file for you. If your system does not do this then you can unpack it using either

unzip XXXX.zip

where XXXX is the algorithm number or by opening it with Winzip (Windows).

Setting up Matlab

In the directory *Matlab* there are two subdirectories *Src* and *Drivers*; add both of these to Matlab's search path.

Below we refer to the *Src* directory as the MieSolver root directory.

Run example codes

Several example scripts are included in the *Drivers* directory. These can be executed by running the following scripts:

```
mie_example_soft
```

```
mie_example_hard
mie_example_robin
mie_example_coated
mie_example_dielectric
```

The figures produced by these examples are included in the Appendix.

We recommend users follow the quick start guide in Section 3 to learn more about how to use the package.

Contents

2 Introduction

We describe the use of our object-oriented Matlab software package MieSolver. This package provides software tools for simulation of wave scattering in two dimensions by circular obstacles. Heterogeneous configurations containing many nonidentical obstacles with various material properties, with and without penetrable layers, can be simulated. The two dimensional wave scattering problem arises in models for propagation of acoustic, elastic and electromagnetic waves in domains containing parallel circular cylinders and we refer to the accompanying paper [3] for further details and references.

The object-oriented structure of our package facilitates easy description of the scattering problem, including the incident wave and the obstacles in a configuration, even for large heterogeneous configurations of nonidentical obstacles. As demonstrated in Section 3, only a few lines of simple code are required to setup and solve a wave scattering simulation with a single scatterer. Examples in Section 8 show that simulations with multiple scatterers and layered scatterers are similarly easy to code.

Our code is based on the well known Mie series, which has been widely described in books, including [1, 4, 5]. For further references to the extensive literature on the Mie series for the two dimensional scattering problem we refer to [3]. It is well known that the Mie series yields highly accurate solutions. In [3] we demonstrate eight digit accuracy of our code in extensive numerical experiments for a wide range of configurations with varying numbers of obstacles and various material parameters.

This article includes the minimal mathematical details required to use the package and understand the underlying algorithm. For complete mathematical details we ask the user of this manual to refer to [3]. *We request that any publications that make use of our MieSolver package cite our accompanying paper [3].*

A note on position vectors in MieSolver In our MieSolver code, and in this manual, we represent real valued vectors (such as position vectors) $(x, y)^T \in \mathbb{R}^2$ by complex numbers $x + yi$.

3 Quick start guide

In this section we describe how to setup, simulate and visualise the scattered field and far field induced by a plane wave interacting with a cylinder comprising a sound-soft core and a penetrable layer.

First we set up the sound-soft scatterer core (with radius 0.5 and centre at the origin) using

```
s = scatterer(0,0.5,'SOFT');
```

Then we add the penetrable layer (with outer radius 1 and refractive index 1.5) using

```
s.addCoating(1,1.5);
```

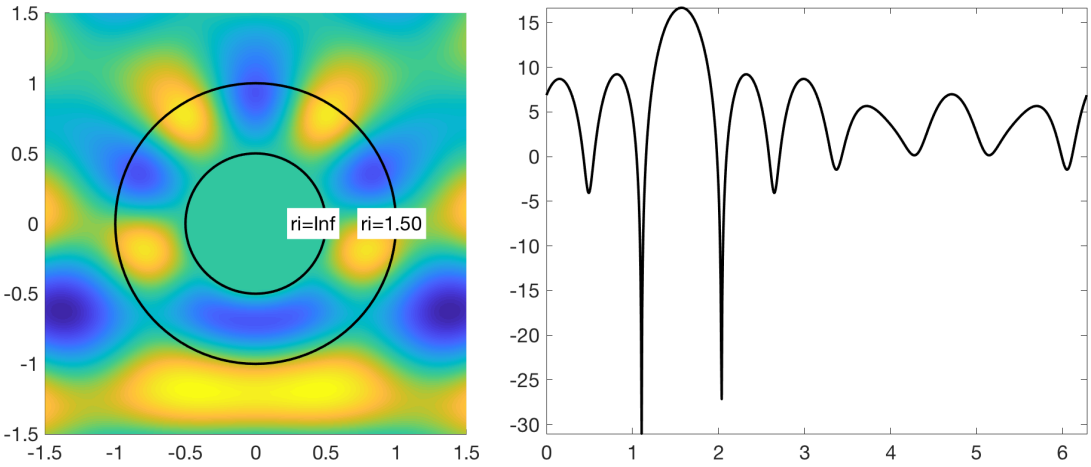


Figure 1: Visualisations of the total field (left) and cross section (right) of a cylinder with a sound-soft core and a penetrable layer produced using the code in the Quick Start Guide (Section 3).

Next we set up the incident plane wave (with wavenumber 5 and direction $\boldsymbol{d} = (\cos\theta, \sin\theta)^T$ for $\theta = \pi/2$) using

```
p = plane_wave(pi/2,5);
```

Next we setup the solver, set the tranmsission type to TE, add the scatterer, and solve to find the scattering coefficients.

```
m = MieSolver(p);
m.transmissionTE();
m.addScatterer(s);
m.solve();
```

We visualise the total field (using the default plotting region) using

```
m.visualiseTotalField()
```

The output is shown in Figure 1. To modify the plotting region please see the detailed instructions in Section 6.

We visualise the far field using

```
m.visualiseRcs()
```

The output is shown in Figure 1.

4 Mathematical Model

Overview

The MieSolver package solves the two dimensional Helmholtz equation, which models propagation of three dimensional waves through a domain containing vertical cylinders. In the two

dimensional Helmholtz model the cylinders are represented by their two dimensional cross sections. Henceforth we use the term "scatterer" to describe the two dimensional cross sections of the cylinders.

Scatterer description

We begin by briefly describing the scattering problem for the case of a single circular scatterer centred at the origin. We assume that the scatterer comprises a homogeneous core with radius r_1 surrounded by $N-1$ homogeneous annular layers with outer radii $r_2, \ldots, r_N$ satisfying

$$r_1 < r_2 < \cdots < r_N.$$

(The case in which the core is not surrounded by any layers satisfies this description with $N = 1$.) The core is then

$$D_1 = B(\mathbf{0}, r_1)$$

where $B(\boldsymbol{x}, r)$ denotes the ball or radius r centred at $\boldsymbol{x}$. The annular layers are

$$D_j = B(\mathbf{0}, r_j) \setminus B(\mathbf{0}, r_{j-1}), \qquad \text{for } 2 \leq j \leq N.$$

It is convenient to denote the outer domain

$$D_{N+1} = \mathbb{R}^2 \setminus B(\mathbf{0}, r_N).$$

PDEs and boundary conditions

The scatterer $D = D_1 \cup \cdots \cup D_N$ is illuminated by an incident wave u^{inc} with wavenumber k, which induces a field u_j in each of the domains D_j for $j = 1, \ldots, N+1$. (If the core is opaque then it is convenient to set the field in the core, $u_1 = 0$.) The induced fields satisfy the two dimensional Helmholtz equation

$$\triangle u(\boldsymbol{x}) + (\nu_j k)^2 u(\boldsymbol{x}) = 0, \qquad \boldsymbol{x} \in D_j, \tag{1}$$

where ν_j denotes the refractive index in D_j for $j = 1, \ldots, N$ and, without loss of generality, we may fix the refractive index in the exterior domain D_{N+1} to be $\nu_{N+1} = 1$.

The induced field u_{N+1} in the exterior domain D_{N+1} is also known as the scattered field, and is required to satisfy the Sommerfeld radiation condition [2, Equation (3.85)]

$$\lim_{|\boldsymbol{x}| \to \infty} \sqrt{|\boldsymbol{x}|} \left(\frac{\partial u_{N+1}}{\partial \boldsymbol{x}}(\boldsymbol{x}) - iku(\boldsymbol{x}) \right) = 0 \tag{2}$$

uniformly as $|\boldsymbol{x}| \to \infty$.

On the interface between the penetrable domains D_j and D_{j+1} for $j = 1, \ldots, N$ we apply transmission boundary conditions

$$\begin{aligned} u_j(\boldsymbol{x}) &= u_{j+1}(\boldsymbol{x}), \qquad \text{for } \boldsymbol{x} \in S_j, \\ \alpha_j \frac{\partial u_j}{\partial n}(\boldsymbol{x}) &= \alpha_{j+1} \frac{\partial u_{j+1}}{\partial n}(\boldsymbol{x}), \qquad \text{for } \boldsymbol{x} \in S_j, \end{aligned} \tag{3}$$

where $S_j = S(\mathbf{0}, r_j)$ is the interface between D_j and D_{j+1}, and $S(\boldsymbol{x}, r)$ denotes the sphere centred at $\boldsymbol{x}$ and with radius r. The parameters α_j and α_{j+1} are transmission parameters that we describe below.

If the core is penetrable with refractive index ν_1 then the incident wave induces a field u_1 in D_1, which also satisfies (1). In this case we apply transmission boundary conditions

$$\begin{aligned} u_1(\boldsymbol{x}) &= u_2(\boldsymbol{x}), \qquad \text{for } \boldsymbol{x} \in S_1, \\ \alpha_1 \frac{\partial u_1}{\partial n}(\boldsymbol{x}) &= \alpha_2 \frac{\partial u_2}{\partial n}(\boldsymbol{x}), \qquad \text{for } \boldsymbol{x} \in S_1, \end{aligned} \tag{4}$$

where $S_1 = S(\mathbf{0}, r_1)$. Here α_1 and α_2 are transmission parameters that we describe below.

If the core is opaque then a Neumann, Dirichlet or Robin boundary condition is applied on S_1 depending on the material properties of the core. For example, if the core is sound soft then we apply the boundary condition

$$u_2(\boldsymbol{x}) = 0, \qquad \text{for } \boldsymbol{x} \in S_1. \tag{5}$$

If the core is sound hard then we apply the boundary condition

$$\frac{\partial u_2}{\partial n}(\boldsymbol{x}) = 0, \qquad \text{for } \boldsymbol{x} \in S_1. \tag{6}$$

Alternatively we may apply the Robin boundary condition

$$\frac{\partial u_2}{\partial n} + i\mu u_2(\boldsymbol{x}) = 0, \qquad \text{for } \boldsymbol{x} \in S_1, \tag{7}$$

where $\mu \in \mathbb{C}$ is a fixed parameter.

In many applications the quantity of interest is the far field

$$u^\infty(\widehat{\boldsymbol{x}}) = \lim_{|\boldsymbol{x}|\to\infty} \sqrt{|\boldsymbol{x}|} e^{-ik|\boldsymbol{x}|} u_{N+1}(\boldsymbol{x}), \tag{8}$$

or the associated cross section (in decibels)

$$\sigma^{\mathrm{dB}}(\widehat{\boldsymbol{x}}) = 10 \log_{10} 2\pi |u^\infty(\widehat{\boldsymbol{x}})|^2. \tag{9}$$

Here $\widehat{\boldsymbol{x}}$ is a unit vector that corresponds to the direction of observation.

Transmission parameters

The transmission parameters in (3) and (4) can be specified by the user as functions of the refractive index or individually for each core. Alternatively, default parameters can be chosen for acoustic scattering or electromagnetic scattering (TE and TM polarisation) problems. The procedures for setting the transmission parameters are given in Section 5.

Solution representation

A detailed description of the method of solution is given in [3]. The induced field in the core is represented as

$$u_1(\boldsymbol{x}) = \sum_{n=-\infty}^{\infty} a_n^{(1)} J_{|n|}(\nu_1 kr) e^{in\theta} \tag{10}$$

for expansion coefficients $(a_n^{(1)})_{n\in\mathbb{Z}}$. Here we use polar coordinates $\boldsymbol{x} = (r\cos\theta, r\sin\theta)^T$, and J_n denotes the Bessel function of order n. Similarly, we represent the induced field in each annular layer as

$$u_j(\boldsymbol{x}) = \sum_{n=-\infty}^{\infty} a_n^{(j)} J_{|n|}(\nu_j kr) e^{in\theta} + \sum_{n=-\infty}^{\infty} b_n^{(j)} H_{|n|}^{(1)}(\nu_j kr) e^{in\theta}, \tag{11}$$

for $2 \leq j \leq N$, with expansion coefficients $(a_n^{(j)})_{n\in\mathbb{Z}}$ and $(b_n^{(j)})_{n\in\mathbb{Z}}$. Here $H_n^{(1)}$ denotes the Hankel function of the first kind of order n. We represent the induced field in the exterior as

$$u_{N+1}(\boldsymbol{x}) = \sum_{n=-\infty}^{\infty} b_n^{(N+1)} H_{|n|}^{(1)}(kr) e^{in\theta} \tag{12}$$

for expansion coefficients $(b_n^{(N+1)})_{n\in\mathbb{Z}}$. This expansion automatically satisfies the radiation condition (2).

For several important types of incident field, such as plane waves and point sources, the incident field in the exterior has the expansion

$$u^{\text{inc}}(\boldsymbol{x}) = \sum_{n=-\infty}^{\infty} a_n^{(N+1)} J_{|n|}(kr) e^{in\theta} \tag{13}$$

with known coefficients $(a_n^{(N+1)})_{n\in\mathbb{Z}}$. For plane waves and point sources an analytical expression is available for the coefficients [3, Equation (8)].

Numerical determination of coefficients

In practice the infinite series in (10)–(13) must be truncated. In our algorithm we replace summation for $n = -\infty, \ldots, \infty$ by summation for $n = -n_{\max}, \ldots, n_{\max}$ where

$$n_{\max} = \text{ceil}(n^*), \qquad n^* = \begin{cases} x + 1 + 4x^{1/3}, & \text{for } x \leq 8, \\ x + 2 + 4.05\, x^{1/3} & \text{for } 8 < x < 4200, \\ x + 2 + 4x^{1/3} & \text{otherwise.} \end{cases} \tag{14}$$

Here $x = \pi s = kr_N$ where s is the diameter of the scatterer in wavelengths,

$$s = \frac{2r_N}{\lambda} = \frac{kr_N}{\pi}. \tag{15}$$

Here $\lambda = 2\pi/k$ denotes the wavelength. This choice for $n_{\max}$ is based on a formula for the analogous truncation parameter for Mie scattering by spheres chosen by Wiscombe [6] to give about eight digits accuracy.

Substituting the trucated series (10)–(12) into the boundary conditions (3) and (4)–(7) yields a linear system for the unknown coefficients. The matrix in the linear system has a block structure with roughly two blocks for each layer (the exact number depends on the material properties of the core). Each block is a $(2n_{\max}+1) \times (2n_{\max}+1)$ diagonal matrix. Therefore it is efficient to assemble the system matrix using Matlab's `sparse` function and solve the linear system using Matlab's sparse direct solver `backslash`. We refer to [3] for full details.

Multiple scatterers

In the case of scattering by a configuration containing M scatterers with centres $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M$ we describe each scatterer as above, with appropriate modification of the centres from $\mathbf{0}$ to $\boldsymbol{x}_j$ for $j = 1, \ldots, M$.

Each scatterer contributes its own component of the form (12) to the scattered field in the exterior, so that the scattered field in the exterior is given by

$$u(\boldsymbol{x}) = \sum_{j=1}^{M} \sum_{n=-\infty}^{\infty} b_n^{(j,N_j+1)} H_{|n|}^{(1)}(kr_j) e^{in\theta_j} \tag{16}$$

where we use polar coordinates $\boldsymbol{x} - \boldsymbol{x}_j = (r_j \cos\theta_j, r_j \sin\theta_j)^T$ *i.e.* polar coordinates for $\boldsymbol{x}$ with local origin $\boldsymbol{x}_j$. Here $b_n^{(j,N_j+1)}$ are the expansion coefficients from (12) on the jth scatterer (and we assume the jth scatterer has N_j domains). The interior fields inside each scatterer are precisely as described above for the single scatterer case, after appropriate modification of the origin for the local polar coordinates.

Part I

The MieSolver classes

5 The `scatterer` class

A MieSolver scatterer comprises a homogeneous core and may contain further homogeneous layers. A scatterer is built up from the core by adding layers.

Instantiation

The `scatterer` class represents a scatterer with a core. An instance of the `scatterer` class with a sound-soft core (*i.e.* a homogeneous Dirichlet boundary condition) is created using

```
S = scatterer(x,r,'SOFT');
```

where `x` is the the centre of the scatterer and `r` the radius of the core.

Instances of the `scatterer` class with sound-hard (*i.e.* a homogeneous Neumann boundary condition), Robin or penetrable cores are created similarly, using

```
S = scatterer(x,r,'HARD');
```

```
S = scatterer(x,r,'ROBIN');
```

```
S = scatterer(x,r,'DIELECTRIC');
```

where `x` is the the centre of the scatterer and `r` the radius of the core.

In the Robin case, the parameter μ in (7) will subsequently need to be set using the `setRobinParameter` method before the scattered field can be computed. Alternatively the Robin parameter can be set at the time of instantiation using

```
S = scatterer(x,r,'ROBIN',mu);
```

where `mu` is the value of the Robin parameter.

In the case of a penetrable core, the refractive index ν of the core will subsequently need to be set using the `setRefractiveIndex` method before the scattered field can be computed. Alternatively the refractive index can be set at the time of instantiation using

```
S = scatterer(x,r,'DIELECTRIC',nu);
```

where `nu` is the value of the refractive index in the core.

For acoustic scattering problems, the density ρ of the core will subsequently need to be set using the `setDensity` method before the scattered field can be computed. Alternatively the density can be set at the time of instantiation using

```
S = scatterer(x,r,'DIELECTRIC',nu,rho);
```

where `nu` is the value of the refractive index in the core and `rho` is the density of the core.

Adding a layer

A penetrable layer is added to the outside of the scatterer using

```
S.addCoating(r,nu);
```

where `r` is the outer radius of the layer and `nu` is the refractive index of the layer.

For acoustic scattering problems a penetrable layer is added to the outside of the scatterer using

```
S.addCoating(r,nu,rho);
```

where `r` is the outer radius of the layer, `nu` is the refractive index of the layer and `rho` is the density of the layer.

Setting the refractive index

The refractive index in each domain D_j can be modified. For example,

```
S.setRefractiveIndex(j,nu)
```

sets the refractive index of domain `j` to `nu`.

If the scatterer has no coatings then

```
S.setRefractiveIndex(nu)
```

sets the refractive index of the core to `nu`.

Setting the density

The density in each domain D_j can be modified. For example,

```
S.setDensity(j,rho)
```

sets the density of domain `j` to `rho`.

If the scatterer has no coatings then

```
S.setDensity(rho)
```

sets the density of the core to `rho`.

Setting the Robin parameter

If the core has a Robin boundary condition then the Robin parameter μ in (7) is set using

```
S.setRobinParameter(mu);
```

where `mu` is the value of the Robin parameter.

Getting the Robin boundary condition parameter

The Robin boundary condition parameter μ in (7) of the core is obtained using

```
mu = S.getRobinParameter()
```

If the core does not have a Robin boundary condition then this method returns `mu = Inf`.

Getting the refractive index

The refractive indices of the domains comprising `S` are obtained using

```
nu = S.getRefractiveIndex()
```

Here `nu` is a vector and `nu(j)` contains the refractive index of the `j`th domain. The entry `nu(end)` contains the refractive index of the exterior domain. If the core is opaque then `nu(1)` is `Inf`.

The refractive index of the `j`th domain is obtained using

```
nu = S.getRefractiveIndex(j)
```

If the scatterer has no coatings then

```
nu = S.getRefractiveIndex()
```

returns the refractive index of the core. If the core is opaque then the result is `nu = Inf`.

Getting the density

The densities of the domains comprising `S` are obtained using

```
rho = S.getDensity()
```

Here `rho` is a vector and `rho(j)` contains the density of the `j`th domain. The entry `rho(end)` contains the density of the exterior domain. If the core is opaque then `rho(1)` is `Inf`.

The density of the `j`th domain is obtained using

```
rho = S.getDensity(j)
```

If the scatterer has no coatings then

```
rho = S.getDensity()
```

returns the density of the core. If the core is opaque then the result is `rho = Inf`.

Getting the radius of the scatterer

The radius `r` of the scatterer `S` is obtained using

```
r = S.getRadius()
```

Getting the centre of the scatterer

The centre `x` of the scatterer `S` is obtained using

```
x = S.getCentre()
```

Getting the acoustic size of the scatterer

The acoustic size s in (15) is obtained using

```
s = S.getSize(wavenumber)
```

where `wavenumber` is the wavenumber.

Visualising the scatterer

The scatterer is visualised in a 2D figure using

```
S.show()
```

The linetype for the visualisation can be specified using *e.g.*

```
S.show('k-')
```

The default visualisation includes labels showing the refractive index of each domain. These labels can be omitted using *e.g.*

```
S.show('k-',0)
```

or

```
S.show([],0)
```

6 The `MieSolver` class

The scattering problem described in Section 4 is setup and solved using the `MieSolver` class.

Instantiation

A `MieSolver` is constructed using an incident field of class `incident` and and one or more scatterers described using objects of class `scatterer`. An instance of the `MieSolver` class is created using

```
P = MieSolver(inc);
```

where `inc` is an object of class `incident`. The object `P` has no scatterers associated with it and scatterers must be subsequently added using the `addScatterer` method before the induced field can be computed.

Alternatively, one or more scatterers can be added at the time of instantiation using

```
P = MieSolver(inc,S);
```

or, for many scatterers,

```
P = MieSolver(inc,S1,S2,...);
```

where `S`, `S1`, `S2` are objects of class `scatterer`. The scatterers can also be specified in a cell array, for example,

```
P = MieSolver(inc,{S1,S2});
```

Setting explicit transmission boundary condition parameters

To use acoustic scattering type transmission parameters, specified using the density in each domain, use the command

```
P.transmissionAcoustic()
```

In particular the parameter α_j in the transmission conditions (3) and (4) is then given by

$$\alpha_j = \frac{1}{\rho_j}$$

where ρ_j is the density in the jth domain.

Note: this option allows each transmission parameter to be specified explicitly (in terms of its reciprocal).

Setting transverse electric (TE) transmission boundary condition parameters

To use TE type transmission parameters use the command

```
P.transmissionTE()
```

In particular the parameter α_j in the transmission conditions (3) and (4) is then given by

$$\alpha_j = 1.$$

Setting transverse magnetic (TM) transmission boundary condition parameters

To use TM type transmission parameters use the command

```
P.transmissionTM()
```

In particular the parameter α_j in the transmission conditions (3) and (4) is then given by

$$\alpha_j = \frac{1}{\nu_j^2}$$

where ν_j is the refractive index in the jth domain.

Setting custom transmission boundary condition parameters

To use custom transmission parameters specified as a function of refractive index, use the command

```
P.transmissionCustom(f)
```

where `f` is a function of one variable. In particular the parameter α_j in the transmission conditions (3) and (4) is then given by

$$\alpha_j = f(\nu_j).$$

For example, transverse electric (TE) type transmission conditions are specified using

```
f = @(nu) 1/nu^2;
```

Adding a scatterer

A scatterer is added using

```
P.addScatterer(S)
```

where `S` is an object of class `scatterer`.

Getting a scatterer

The `j`th scatterer is obtained using

```
S = P.getScatterer(j)
```

Adjusting the truncation parameter

The truncation parameter $n_{\max}$ in Section 4 is chosen automatically for each scatterer to give about eight digits accuracy in the computed solution using (14). This truncation parameter may be adjusted to increase the accuracy. The truncation parameter is incremented by `m` using

```
P.incrementNmax(m)
```

This method must be used before the `solve` method. If the `incrementNmax` method is used more than once then its effects will be compounded.

Getting the truncation parameter

The truncation parameter $n_{\max}$ in Section 4 is obtained using

```
n = P.getNmax()
```

Here `n` is a vector and `n(j)` containes the truncation parameter of the `j`th scatterer.

The truncation parameter for the `j`th scatterer is obtained using

```
n = P.getNmax(j)
```

Computing the induced field coefficients

The coefficients of the induced fields described in Section 4 are computed using

```
P.solve()
```

This method must be called before the induced field can be computed or visualised.

Getting the induced field

The induced field described in Section 4 is obtained at points `x` using

```
u = P.getInducedField(x);
```

where `x` is an array of complex numbers representing position vectors. If `x(j)` lies inside the core of any scatterer then `u(j) = 0`.

Getting the total field

The total field described in Section 4 is obtained at points `x` using

```
u = P.getTotalField(x);
```

where `x` is an array of complex numbers representing position vectors. If `x(j)` lies inside the core of any scatterer then `u(j) = 0`.

Getting the far field

The far field (8) is obtained at points `xh` using

```
u = P.getFarField(x);
```

where `x` is an array of complex numbers representing position vectors and `xh = x./abs(x)`. The points in `xh` lie on the unit circle and represent observation directions.

Getting the Cross Section in decibels

The cross section (9) is obtained at points `xh` using

```
u = P.getRcs(x);
```

where `x` is an array of complex numbers representing position vectors and `xh = x./abs(x)`. The points in `xh` lie on the unit circle and represent observation directions.

Visualising the total field

The total field is visualised using

```
P.visualiseTotalField()
```

The figure shows the real part of the total field over a default rectangular plotting region.

To plot the total field over a custom plotting region $[\mathtt{a}, \mathtt{c}] \times [\mathtt{b}, \mathtt{d}]$ use

```
P.visualiseTotalField([a+bi,c+di])
```

The visualisation uses about 10 points per wavelength. If the grid size exceeds 400×400 then the visualisation will be cancelled unless the `-f` override described below is used.

Additional options can be specified using a string `opts`

```
P.visualiseTotalField([a+bi,c+di],opts)
```

Supported options are:

`-f` overrides the 400×400 grid size restriction.

`abs` plots the absolute value of the total field.

Visualising the total field

The cross section (9) is visualised using

```
P.visualiseRcs()
```

The linestyle may additionally be specified *e.g.*

```
P.visualiseRcs('r-')
```

to plot with a solid red line.

7 The `incident` class

Incident fields are represented by the `incident` class. The `incident` class is an abstract class and two child classes `plane_wave` and `point_source` are provided.

Evaluating an incident field

An incident field `p` is evaluated at points `z` using

```
u = p.evaluate(z);
```

Here `z` is an array of complex numbers representing position vectors.

Evaluating the gradient of an incident field

The gradient of an incident field `p` is evaluated at points `z` using

```
[dx,dy] = p.evaluateGradient(z);
```

Here `z` is an array of complex numbers representing position vectors. The outputs `dx` and `dy` are the first and second components of the gradient of the incident field.

Obtaining wavefunction expansion coefficients

The wavefunction expansion coefficients of the incident wave in (13) are given by

```
cof = p.get_coefficients(x,nmax);
```

where `x` is the expansion centre and `nmax` is the truncation parameter.

Addition and scalar multiplication

Two objects `p` and `q` of class `incident` can be added using

```
s = p + q;
```

and substracted using

```
s = p - q;
```

An object `p` of class `incident` can be multiplied by a scalar using *e.g.*

```
s = 3 * p;
```

Incident plane waves

An incident plane wave

$$u^{\text{inc}}(\boldsymbol{x}) = e^{ik\boldsymbol{x}\cdot\widehat{\boldsymbol{d}}} \tag{17}$$

with direction specified by the unit vector $\widehat{\boldsymbol{d}} = (\cos\theta, \sin\theta)^T$ and wavenumber k is represented by the `plane_wave` class. An instance of the `plane_wave` class is created using

```
p = plane_wave(theta,wavenumber);
```

where `theta` is the angle specifying the incident direction and `wavenumber` is the wavenumber.

Incident point sources

An incident point source

$$u^{\text{inc}}(\boldsymbol{x}) = \frac{i}{4} H_0^{(1)}(k|\boldsymbol{x} - \boldsymbol{x}_0|) \tag{18}$$

with source location $\boldsymbol{x}_0$ and wavenumber k is represented by the `point_source` class. An instance of the `point_source` class is created using

```
p = point_source(x0,wavenumber);
```

where `x0` is a complex number representing the source location and and `wavenumber` is the wavenumber.

8 Examples

The solution procedure in the Quick Start Guide (Section 3) is independent of the particular configuration of cylinders. Here we give some examples of how to set up particular configurations.

Single sound-soft cylinder (homogeneous Dirichlet BC)

To setup an instance of the `MieSolver` class for a single sound-soft cylinder with centre `x` and radius `r`

```
s = scatterer(x,r,'SOFT');
m = MieSolver(inc,s);
```

where `inc` is of class `incident`.

Single sound-hard cylinder (homogeneous Neumann BC)

To setup an instance of the `MieSolver` class for a single sound-hard cylinder with centre `x` and radius `r`

```
s = scatterer(x,r,'HARD');
m = MieSolver(inc,s);
```

where `inc` is of class `incident`.

Single absorbing cylinder (homogeneous Robin BC)

To setup an instance of the `MieSolver` class for a single absorbing cylinder with centre `x`, radius `r` and impedance parameter `mu`

```
s = scatterer(x,r,'ROBIN',mu);
m = MieSolver(inc,s);
```

where `inc` is of class `incident`.

Single penetrable cylinder

To setup an instance of the `MieSolver` class for a single penetrable cylinder with centre `x`, radius `r` and refractive index `nu`

```
s = scatterer(x,r,DIELECTRIC',nu);
m = MieSolver(inc,s);
```

where `inc` is of class `incident`.

Single layered cylinder

To setup an instance of the `MieSolver` class for a single layered cylinder with centre `x`, sound-soft core with radius `r0` and a layer with radius `r1` and refractive index `nu`

```
s = scatterer(x,r0,'SOFT');
s.addCoating(r1,nu)
m = MieSolver(inc,s);
```

where `inc` is of class `incident`.

Two sound-soft cylinders (homogeneous Dirichlet BC)

To setup an instance of the `MieSolver` class for two sound-soft cylinders with centres `x1` and `x2` and radii `r1` and `r2` respectively

```
s1 = scatterer(x1,r1,'SOFT');
s2 = scatterer(x2,r2,'SOFT');
m = MieSolver(inc,s1,s2);
```

where `inc` is of class `incident`.

References

[1] C. F. Bohren and D. R. Huffman. *Absorption and Scattering of Light by Small Particles*. Wiley, 1983.

[2] D. Colton and R. Kress. *Inverse Acoustic and Electromagnetic Scattering Theory*. Springer, 2012.

[3] Stuart C. Hawkins. Algorithm XXX: MieSolver—an object-oriented Mie series software for wave scattering by cylinders. *ACM Trans. Math. Softw.*, submitted.

[4] T. Rother and M. Kahnert. *Electromagnetic wave scattering on nonspherical particles: basic methodology and simulations*. Springer, 2nd edition, 2013.

[5] H. C. van de Hulst. *Light Scattering by Small Particles*. Dover Publications Inc., 1981.

[6] W. J. Wiscombe. Improved Mie scattering algorithms. *Applied Optics*, 19:1505–1509, 1980.

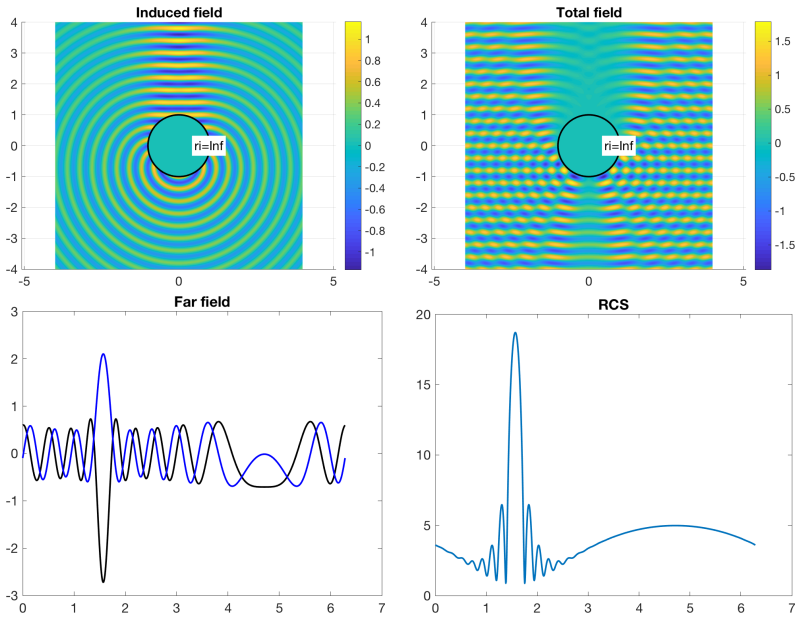
A Output from examples

Several example scripts are included in the *Drivers* directory. These can be executed by running the following scripts:

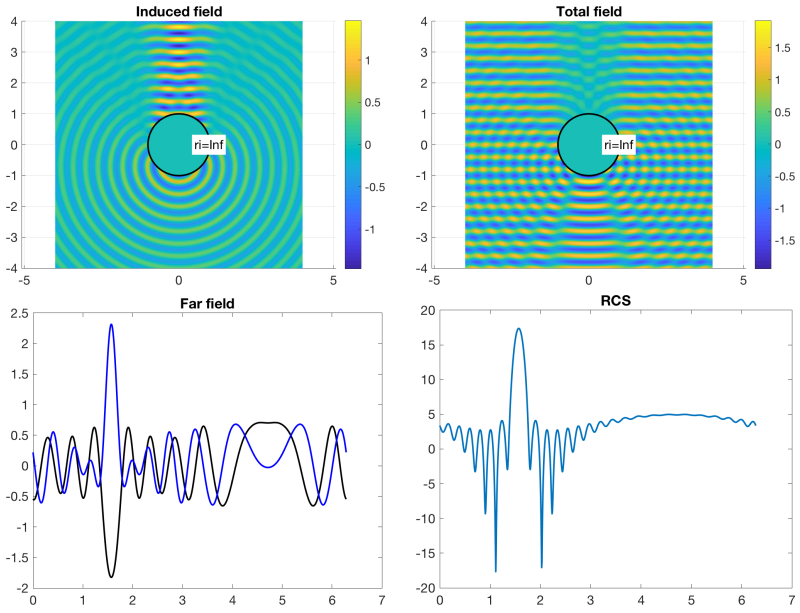
```
mie_example_soft
mie_example_hard
mie_example_robin
mie_example_coated
mie_example_dielectric
```

In this appendix we present the figures produced by these examples, so that they can be used for validating the installation.

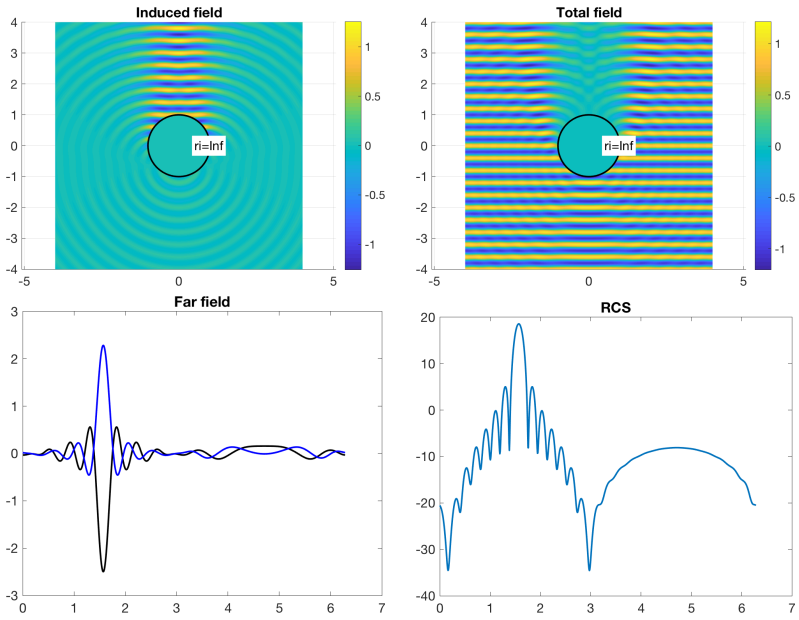
Figures from `mie_example_soft`



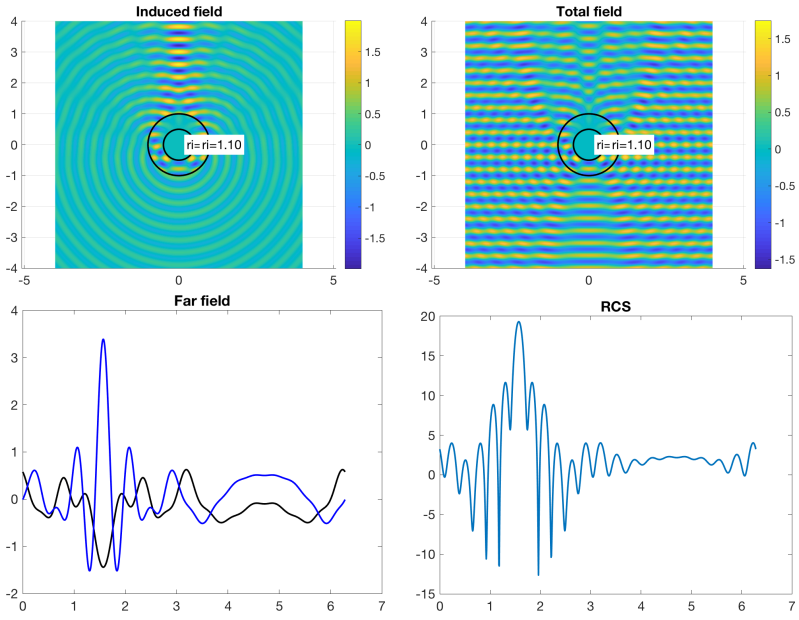
Figures from `mie_example_hard`



Figures from `mie_example_robin`



Figures from `mie_example_coated`



Figures from `mie_example_dielectric`