User Guide for IPscatt—a MATLAB Toolbox for the Inverse Medium Problem in Scattering

Florian Bürgel[*], Kamil S. Kazimierski[†], Armin Lechleiter[‡]

User Guide for Release ipscatt2017, updated June 2, 2019

Abstract

IPscatt is a free, open-source MATLAB toolbox facilitating the solution of the two and three-dimensional inverse medium problem in time-independent scattering (also known as time-harmonic scattering). For a survey of the mathematical concepts beyond the provided implementation we refer to the corresponding *algorithm paper*. In particular, the *algorithm paper* describes the direct scattering problem and the implemented variational reconstruction scheme. This *user guide* contains installation instructions, a technical description, practical guides for using the library with code snippets as well as resulting figures and finally some applications.

Keywords Inverse Scattering Problem, Parameter Identification, MATLAB Toolbox, Denoising, Sparsity Regularization, Total Variation Regularization, Primal-Dual Algorithm, User Guide.

Contents

[*]University of Bremen, Center for Industrial Mathematics, Germany, fbuergel@uni-bremen.de.

[†]University of Graz, Institute of Mathematics and Scientific Computing, Austria, kazimier@uni-graz.at. The Institute of Mathematics and Scientific Computing is a member of NAWI Graz (www.nawigraz.at) and BioTechMed Graz (www.biotechmed.at).

[‡]University of Bremen, Center for Industrial Mathematics, Germany.

Introduction

Inverse medium problems in scattering attempt to identify the contrast, that is the squared refractive index minus one, of a penetrable medium from measurements of waves scattered from that medium. This parameter identification problem has many interesting application areas, e.g. in non-destructive testing procedures. As a nonlinear and ill-posed problem it is challenging, although we limit ourselves to time-independent scattering; the huge system sizes after discretization makes the problem even more demanding. Therefore we provide the free, open-source software IPscatt, which is an efficient and effective MATLAB toolbox for the nonlinear inverse medium problem in time-independent scattering in two and three dimensions. The variational regularization scheme, on which the reconstruction part of IPscatt is based, minimizes the defect and the penalty terms that take into account a priori information. This scheme was introduced in [BKL17, Sec. 4] and relies on the so-called primal-dual algorithm due to Pock, Bischof, Cremers and Chambolle, see [PCBC09, CP11]. To the authors' best knowledge, IPscatt is the first toolbox that combines sparsity promoting and total variation-based regularization to jointly improve the reconstruction quality. Further, physical bounds are used as a priori information on the scattering object restricting the real and imaginary part of the contrast.

This *user guide* was written for readers, who are interested in using the software IPscatt, and is part of the supplementary material[*], which contains first, the *source code*, second, this *user guide* and third, the *source code documentation* containing a detailed description of each function.

Also, an *algorithm paper* for IPscatt was written, which contains an overview of the software framework, presents its key features, describes the direct scattering problem as well as the underlying variational reconstruction scheme, compares IPscatt with other software packages and discusses the application areas.

Organization of this User Guide This *user guide* is organized as follows: Sec. 1 contains some general information about the software, installation instructions and the first code snippet to demonstrate a typical work flow. Then, the technical description containing a typical program flow and the most important variables is given in Sec. 2. Next, we show the main functionality of IPscatt on selected examples in Sec. 3. The modular design of IPscatt is shown on selected examples in Sec. 4. We hope this encourages users of IPscatt to take parts of it and use them for their own applications. For the reader's convenience we repeat some formulas from the *algorithm paper* in the Appendix if we refer to them, in particular from the direct scattering problem and the implemented reconstruction scheme. Further, in the Appendix we repeat the continuous and discretized forms of the direct scattering problem from the *algorithm paper* in a tabular and also give them in the source code form to build a connection between mathematical formulas and their implementation.

1 Installation

This section contains the metadata of the software, installation instructions and a first demonstration of the toolbox. Furthermore, we give a first code snippet to present a typical work flow and refer to the most important guides for a quick introduction to the toolbox IPscatt.

Software Metadata The software package IPscatt is a MATLAB toolbox for the inverse medium problem in time-independent scattering. It is provided as free software under the GNU General Public License (GPLv3). The toolbox is written in MATLAB and is therefore supported on any operating system which MATLAB supports. It was extensively tested in MATLAB R2016b under Linux, but short tests under Windows 7 and 10 were also successful. Apart from MATLAB the software library IPscatt depends on the "Signal Processing Toolbox" and the "Image Processing Toolbox". Unfortunately, IPscatt does not work with Octave (at least in version 4.0.0). As already mentioned we provide as supporting material, apart from this *user guide*, a *source code documentation* containing a detailed description of each function.

All computations in this *user guide* were carried out on a workstation with an Intel(R) Core(TM) i7-3770 CPU with 3.40 GHz and 32 GByte RAM. For the given code snippets we recommend more than 5 GByte RAM. This, in particular, is true for the three-dimensional reconstruction example as it is the most RAM-consuming one.

[*]Supplementary material is provided on https://calgo.acm.org/ and on http://www.fbuergel.de/ipscatt/ in the latest version.

Installation For installation just unpack the compressed archive. It contains the folder code with the *source code*, the folder doc with the *source code documentation* and the folder guide with this *user guide*. The directory structure of code is described in the following list.

- 3rdparty (3rd party code)
- conv (convenience functions)
- docCreate (creates the documentation)
- guides (code for the guides)
- incontrasts (input of predefined contrasts)
- inseti (input of structure array seti)
- output (place to store files and figures)
- proc (process)
  - auxi (auxiliary routines)
  - expData (working with real-world data)
  - expSetup (experimental set-up)
  - intOps (integral operators)
  - norms (definitions of norms)
  - operators (operators)
  - plots (functions for plots)
  - plotsAux (auxilary functions for plots)
  - recon (reconstruction process)
  - reconAux (associated auxiliary functions)
  - setData (sets geometry and simulation)
  - setInput (general input, e.g. make folders)
- tests (test functions)
  - auxi (auxiliary functions)

Demonstration For a first impression of IPscatt we use the routine demo in MATLAB with code as current folder. This executes a demonstration script in which all important parts of IPscatt are executed. However, to keep the run time low the number of transmitters/receivers, discretization points and reconstruction steps was set very low. This means that the resulting reconstruction is not representative for IPscatt. The generated figures are saved in a folder with the current date as prefix inside the directory output.

Quick Start and Typical Work Flow Input parameters can be set in the fields of the structure array[†] seti, that is a mutable data structure. IPscatt takes care that all necessary fields are set in a consistent manner, e.g. if these fields are empty as in Lst. 1, default parameters will be set. In the routine to set the data, setData, this concerns e. g. the underlying grid, the positions of the transmitters and receivers, the type of the incident field as well as the predefined contrast of the scattering object. Furthermore, setData solves the direct scattering problem and adds Gaussian noise to the simulated data. In the end, the variational reconstruction starts by recon after reconstruction-dependent parameters were set in setRecon.

```
init;
seti = struct; seti = setData(seti);
seti = setRecon(seti); seti = recon(seti);
```

Listing 1: Quick Start Example.

For a quick introduction to the toolbox IPscatt see the guides in Sec. 3, in particular Sec. 3.1, 3.2, 3.3, 3.4, 3.8 and 3.11. These guides will show how to: generate a grid; use some standard transmitter-receiver geometries for the experimental set-up; load predefined contrasts; set geometry and simulation easily; simulate data with noise; reconstruct in the case of two and three-dimensional scattering.

2 Technical Description

The typical program flow of IPscatt is visualized in Fig. 1, in which the right column contains the code of a typical work flow. Optional parts of IPscatt are shown in Fig. 2. Furthermore, the most important fields of the structure array seti are listed in Tab. 1. We will take a closer look at some of the functions and fields in the guides in Sec. 3. However, a detailed reference of all existing functions is in the *source code documentation*. In particular, this *source code documentation* is interesting for advanced usage and further development of IPscatt for which the toolbox was designed. Therefore IPscatt supports the user by providing tests of internal functions. In particular, it tests the proper working of the forward operator and its derivative. These tests may take a long time and are started by seti = runtests() with code as current folder.

[†]We follow MATLAB's nomenclature denoting an *associative array* as *structure array* and *return values* as *output arguments*.

Function: setGeomSim.m, see Sec. 3.4 (geometry and simulation):

- Grid (CD and ROI) (Sec. 3.1) (setGrid.m ↦ seti.grid, seti.gridROI).
- Experimental set-up (Sec. 3.2):
  - Transmitters' positions (setIncPnts.m ↦ seti.incPnts).
  - Receivers' positions (setMeasPnts.m ↦ seti.measPnts).
  - Incident field (setIncField.m ↦ seti.incField).
  - Measurement kernel (setMeasKer.m ↦ seti.measKer).
- Contrast (Sec. 3.3) (↦ seti.qROIexact).

Auxiliary routines	*Main program*	*Code*
	init.m (Initialization, see Sec. 3) ↓	init;

setGeomSim.m
(see above)
 	
forward.m:
1. init.m.
2. setGeomSim.m.
3. forward.m, Sec. 3.5:
↦ FFqMeas = $\mathcal{F}(q)$.
(Underlying: mimo.m.)
	setData.m
	(Sec. 3.8 and 3.10)
	↙ ↘
	Simulated data with noise
	1. setGeomSim.m, Sec. 3.4:
	↦ seti.incField,
	↦ seti.qROIexact.
	2. forward.m, Sec. 3.5:
	↦ seti.FmeasExact.
	3. addNoise.m, Sec. 3.8:
	↦ seti.FmeasDelta.
	
	Real-world data (Fresnel)
	(loadData.m, Sec. 3.10)
	1. readRAWData.m, Sec. 3.9:
	(via uTotRX, uIncRX)
	↦ seti.FmeasDelta.
	2. setGeomSim.m.
	3. matchIncField.m, Sec. 3.10:
	(via uIncROI)
	↦ seti.incField.
	
	↦ seti.FmeasDelta
	(data—simulated data with noise or real-world data)
	↓
		seti = struct;
		seti = setData(seti);

forward.m, derivative.m:
1. init.m.
2. setGeomSim.m.
3. forward.m, Sec. 3.5:
↦ FFqMeas = $\mathcal{F}(q)$
or
derivative.m, Sec. 3.6:
↦ JA, JB to compute $\mathcal{F}'(q)[h]$ = JA*h*JB.
(Underlying: mimo.m).
	setRecon.m
	(Sets required fields in seti, Sec. 3.11.)
	↦ seti.alpha, seti.beta, seti.physBounds.
	↓
	recon.m
	(Variational reconstruction, Sec. 3.11.)
	$\boxed{\text{Inner iteration}}$ by pda.m ↦ h
	↓↑
	$\boxed{\text{Outer iteration}}$ by minPda.m ↦ $q := q + h$
	(with q = seti.qROIcomp and h = h).
	
	↦ seti.qROIcomp
	(Reconstructed contrast as a vector.)
		seti = setRecon(seti);
		seti = recon(seti);

Figure 1: The typical program flow of IPscatt with the most important output arguments (emphasized by ↦).

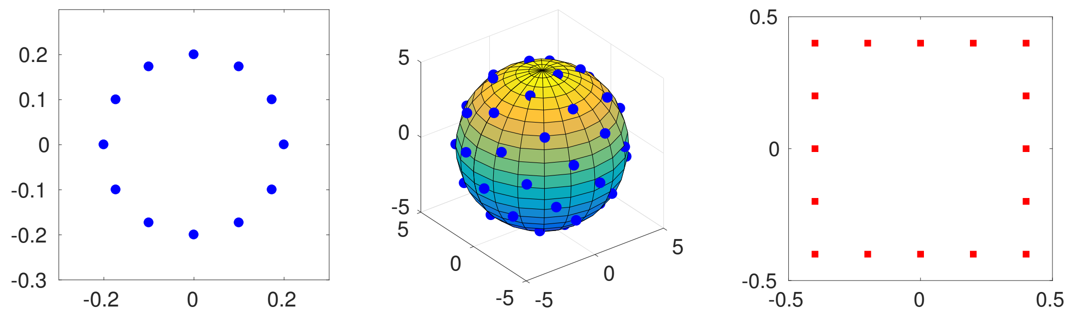
Optional Parts of IPscatt

- ADJOINT OF THE DERIVATIVE AT THE DEFECT: This adjoint is computed by: 1. init.m, 2. setGeomSim.m, 3. adjOfDer.m, see Sec. 3.7: $\mapsto$ ADFFq $= [\mathcal{F}'(q)]^*[\mathcal{F}(q) - F^\delta_{\text{meas}}]$ by the underlying routine mimo.m.
- CONVENIENCE MODE: start.m supports to load presets and save figures/files automatically, see Sec. 3.13.
- CHOOSING THE REGULARIZATION PARAMETERS: As in the convenience mode, figures and files are saved, and moreover computations are repeated for various input arguments of α and β via varalpha.m and varbeta.m, see Sec. 3.15. (This proceeding is also provided for different noise levels δ via vardelta.m.)

Figure 2: Optional parts of IPscatt.

Name	*Type*	*Size*	*Description*
FIELDS IN setData, PART 1:			(Not in Fig. 1, but important.)
dim	2 or 3	1	Dimension of the problem (default: 2).
rCD	floating-point	1	Computational domain (CD) has size $[-r, r)^d$ with $r =$ rCD (default: 0.2) and $d =$ dim.
nCD	integer	1	Discretization points for each dimension of CD (in samples) (default: 256).
incNb	integer	1	Number of transmitters (default: 35).
measNb	integer	1	Number of receivers (default: 35).
nROI	integer	1	Discretization points for each dimension of region of interest (ROI) (in samples).
FIELDS IN setData, PART 2:			
incPnts	matrix in $\mathbb{R}$	dim $\times$ incNb	Coordinates of transmitters.
measPnts	matrix in $\mathbb{R}$	dim $\times$ measNb	Coordinates of receivers.
gridROI	matrix in $\mathbb{R}$	dim $\times$ nROI^dim	Grid in region of interest (ROI).
incField	matrix in $\mathbb{C}$	nROI^dim $\times$ incNb	Incident fields evaluated on the region of interest (ROI) for each transmitter.
measKer	matrix in $\mathbb{C}$	measNb $\times$ nROI^dim	Measurement kernel in region of interest (ROI) for each receiver.
qROIexact	vector in $\mathbb{C}$	nROI^dim $\times$ 1	Predefined contrast evaluated on region of interest.
FmeasExact	matrix in $\mathbb{C}$	measNb $\times$ incNb	Exact data, i.e. scattered field at receivers' positions.
FmeasDelta	matrix in $\mathbb{C}$	measNb $\times$ incNb	Data (simulated with noise or real-world).
FIELDS IN setRecon:			
alpha	floating-point	1	Regularization parameter α for the sparsity penalty, see (5), (default: 500).
beta	floating-point	1	Regularization parameter β for total variation penalty, see (5), (default: 10^{-5}).
physBounds	vector in $\mathbb{R}$	1×4	Bounds for real/imaginary part of contrast: $[a, b, c, d]$, see (5), (default: $[-1, 3, 0, 3]$).
FIELDS IN recon:			
qROIcomp	vector in $\mathbb{C}$	nROI^dim $\times$ 1	Reconstructed contrast (by computation).

Table 1: Most important fields of the structure array seti corresponding to Fig. 1.



(a) Transmitters. (b) Transmitters in 3D. (c) Receivers.

Figure 3: (a) Transmitters on a circle as generated by Lst. 3. (b) Transmitters on a sphere as generated by Lst. 4. (c) Receivers on a square as generated by Lst. 5.

3 Guides

In this section we consider practical guides for using the library with code snippets as well as resulting figures. In addition, we discuss the input and output arguments of the most important functions.

General Use of Guides In order to work properly all examples discussed below must be executed from the main directory of the *source code* as current folder. For example, the file init.m is inside the main directory code. The most code snippets start with this routine to add required subfolders to the paths of MATLAB. As already mentioned a *source code documentation* with a detailed reference of all functions exists. If we refer to the name of a routine, e.g. see init.m, the corresponding *source code documentation* is meant.

3.1 Guide: Generating a Grid

To generate a grid IPscatt provides the routine setGrid. It creates a grid in the computational domain (CD), that has the size $[-r, r)^d$ for a defined $r = \text{seti.rCD}$ and dimension $d = \text{seti.dim}$ (2 or 3). The number of discretization points in each dimension has to be defined by seti.nCD. The resulting grid is stored in seti.grid as a matrix of size $d \times \text{seti.nCD}^d$. The resulting area/volume of the infinitesimal element (pixel/voxel) of CD is stored in seti.dV. Further, the routine generates a grid in the smaller region of interest (ROI) in seti.gridROI, that is a matrix of size $d \times N_\mathrm{D}$. Note that $N_\mathrm{D} = \text{seti.nROI}^d$, where seti.nROI is the computed number of discretization points for each dimension. For more details consult the short example Lst. 2 and see setGrid.m.

```
init; seti.dim = 2; seti.rCD = 0.2; seti.nCD = 256;
seti = setGrid(seti);
```

Listing 2: Generate a grid with setGrid (*source code*: guides/guideSetGrid.m).

3.2 Guide: Creating the Experimental Set-Up

In this section we consider the four parts of the experimental set-up routine expSetup: the transmitters' positions incPnts are generated by setIncPnts, the receivers' positions measPnts are computed in setMeasPnts, the incident fields incField are evaluated on the region of interest in setIncField and the measurement kernel measKer is set up in setMeasKer, see (15).

To create the experimental set-up in two dimensions, IPscatt provides some standard geometric shapes like a circle, a square and a line, along which a certain number of transmitters or receivers are equidistantly arranged. In three dimensions it provides a sphere on which they are roughly equidistantly arranged using the Fibonacci lattice, see [Gon10, Sec. 3.1].

Transmitters' Positions The geometric shape of the transmitters can be generated by the function setIncPnts. It requires the dimension by $d =$ seti.dim (2 or 3) and the type of the incident field by seti.incType ('pointSource' or 'planeWave'). The other input arguments depend on the shape chosen by seti.incPntsType. The user is also free to define his own positions. Further details are given in setIncPnts.m and pntsGeometry.m. Note that plane waves only depend on direction, such that the radius seti.radSrc is set to 1 in this case. The output arguments are the coordinates of the transmitters stored in seti.incPnts as a real matrix of size $d \times N_{\mathrm{i}}$, the number of transmitters $N_{\mathrm{i}} =$ seti.incNb and the approximation of the infinitesimal element of closed contour with control points stored in seti.dSInc. (If a closed contour does not make sense, it is set to 1.)

Lst. 3 arranges 12 transmitters emitting point sources on a circle with radius 0.2 and results in Fig. 3(a).

```
init; seti.dim = 2;
seti.incType = 'pointSource';
seti.incPntsType = 'circle'; seti.radSrc = 0.2; seti.incNb = 12;
seti = setIncPnts(seti);
plot(seti.incPnts(1,:),seti.incPnts(2,:),'b.');
```

Listing 3: Arrange transmitters on a circle (*source code*: guides/guideExpSetupTrans.m).

Lst. 4 arranges 49 transmitters emitting point sources on a sphere with radius 5 and results in Fig. 3(b). Note that in fact 49 transmitters are arranged because 50 is an even number.

```
init; seti.dim = 3;
seti.incType = 'pointSource';
seti.incPntsType = 'sphereFibo'; seti.radSrc = 5; seti.incNb = 50;
seti = setIncPnts(seti);
[sx,sy,sz] = sphere; r = seti.radSrc; surf(sx*r,sy*r,sz*r); hold on;
Pnts = seti.incPnts; s = 100; scatter3(Pnts(1,:),Pnts(2,:),Pnts(3,:),s,'filled','b'); hold off;
```

Listing 4: Arrange transmitters on a sphere (*source code*: guides/guideExpSetupTrans3D.m).

Receivers' Positions The same geometric shapes available for transmitters are also available for receivers. Therefore the function setMeasPnts is the analogy of setIncPnts. The main difference is the choice of the type of the measured field (instead of the incident field) by seti.measType ('nearField' or 'farField'). The shape is chosen by seti.measPntsType, see setMeasPnts.m and pntsGeometry.m for details. The output arguments are analogous to the function setIncPnts, i.e. the coordinates are stored in seti.measPnts, the number in seti.measNb and the infinitesimal element in seti.dSMeas. We stress that a far field only depends on the direction. Therefore IPscatt chooses seti.measPnts as a circle (or a sphere) of radius one in this case and uses these points as argument for the computed far field.

Lst. 5 arranges $4+1$ points on each edge of a square with length 0.8 and resulting receivers are plotted in Fig. 3(c).

```
init; seti.dim = 2;
seti.measType = 'nearField';
seti.measPntsType = 'square'; seti.measNbEdge = 4; seti.measEdgeLength = 0.8;
seti = setMeasPnts(seti);
plot(seti.measPnts(1,:),seti.measPnts(2,:),'rs');
```

Listing 5: Arrange receivers on a square (*source code*: guides/guideExpSetupRece.m).

Incident Field The routine seti = setIncField(seti) evaluates the incident fields on the region of interest (ROI) for each transmitter and stores them in seti.incField.

Details of setIncField Remember that the incident field u^{i} solves the Helmholtz equation, see (1). Typical choices of u^{i} include plane waves and point sources and are provided by IPscatt.

Let $u^{\mathrm{i}}_{\mathrm{plane}}(x) = \exp(\mathrm{i}k\langle x, \theta\rangle)$ be the incident field at position x in the case of a *plane wave* (in 2D and 3D) where θ is the direction. Then the incident fields of N_{i} plane waves with directions θ_j, $j = 1, \dots, N_{\mathrm{i}}$, at points x in ROI are stored as a complex matrix `seti.incField` $= (\exp(\mathrm{i}k\langle x, \theta_j\rangle))$ of size $N_{\mathrm{D}} \times N_{\mathrm{i}}$.

In the case of a *point source* at transmitter position y the incident field is defined by $u^{\mathrm{i}}_{\mathrm{point}}(x) = \Phi(x - y)$ for points x in ROI, where Φ is radiating fundamental solution, see (4), of the Helmholtz equation. The incident fields at points x in ROI of N_{i} point sources at positions y_j, $j = 1, \dots, N_{\mathrm{i}}$, are stored as a complex matrix `seti.incField` $= (\Phi(x - y_j))$ again of size $N_{\mathrm{D}} \times N_{\mathrm{i}}$.

For further information concerning incident fields see [CK13, Ch. 3.5 and 8.4] or [BKL17, Sec. 3.5].

Input Arguments of setIncField The routine setIncField requires the already introduced number of transmitters seti.incNb, their positions seti.incPnts and the grid-related arguments seti.nROI and seti.gridROI. Furthermore, the following arguments are needed:

seti.incType	Type of the incident field ('pointSource' or 'planeWave').
seti.k	Wave number (in reciprocal meters).
seti.model	Chosen model of the problem ('helmholtz2D' or 'helmholtz3D').

Output Argument of setIncField The incident field in the region of interest for each transmitter is stored in seti.incField as a complex matrix of size $N_{\mathrm{D}} \times N_{\mathrm{i}}$. Note that ROI is stored as a vector of size N_{D} (remember that $N_{\mathrm{D}} =$ seti.nROI^d with $d =$ seti.dim) instead of a matrix to use a matrix multiplication when evaluating the forward operator, cf. (19). The number of incident fields is $N_{\mathrm{i}} =$ `seti.incNb`.

Measurement Kernel The measurement kernel seti.measKer is computed in setMeasKer for each receiver. It is needed to evaluate the sensor measurements $u^{\mathrm{s}}|_{\mathrm{RX}} = k^2 \kappa\, q \odot u^{\mathrm{t}} \mathcal{V}$ at the receivers' positions RX for near field data, i.e. to evaluate the forward operator for one incident field, cf. (15) and (20). The used quantities are the wave number $k =$ seti.k, the measurement kernel $\kappa =$ seti.measKer, the contrast $q =$ seti.qROI, the total field $u^{\mathrm{t}} = u^{\mathrm{i}} + u^{\mathrm{s}} =$ uIncROI+uScattROI and the voxel volume $\mathcal{V} =$ seti.dV. The notation $f \odot g$ stresses the pointwise multiplication in the discretized case. The above formula also evaluates sensor measurements at the directions in RX in the case of far field data, cf. (17) for the differing measurement kernel.

Input Arguments of setMeasKer The input arguments of setMeasKer are almost the same as in setIncField. The only difference is that seti.incType, seti.incNb and seti.incPnts are replaced by seti.measType to set the type of the scattered field ('nearField' or 'farField'), $N_{\mathrm{s}} =$ seti.measNb for the number of receivers and seti.measPnts to set the coordinates of the receivers (real matrix of size $d \times N_{\mathrm{s}}$ with $d =$ seti.dim).

Output Argument of setMeasKer The measurement kernel in ROI for each receiver is stored in seti.measKer (complex matrix of size $N_{\mathrm{s}} \times N_{\mathrm{D}}$).

Example for the Incident Field and the Measurement Kernel Lst. 6 demonstrates the use of setIncField and setMeasKer following Lst. 2, 3 and 5. The resulting fields are visualized in Fig. 4.

```
init;
guideExpSetupTrans; guideExpSetupRece; guideSetGrid;
seti.model = 'helmholtz2D';
seti.k = 250;
seti = setIncField(seti);
seti = setMeasKer(seti);
figure(1); imagesc(real(reshape(seti.incField(:,1),[seti.nROI seti.nROI]))); axis xy; colorbar;
figure(2); imagesc(real(reshape(seti.measKer(5,:),[seti.nROI seti.nROI]))); axis xy; colorbar;
```

Listing 6: Incident field (setIncField) and measurement kernel (setMeasKer) (*source code*: guides/guideExpSetup.m).

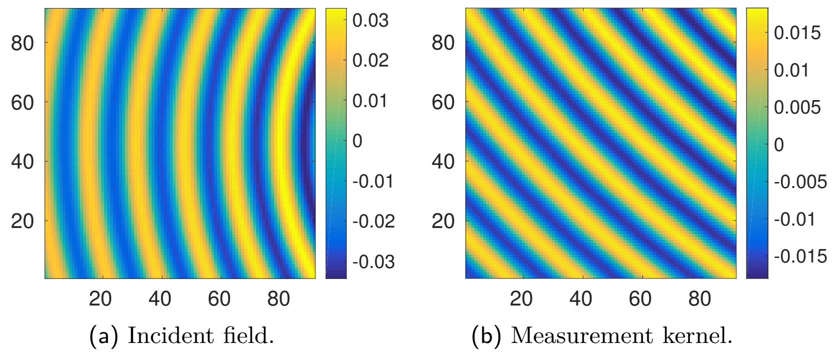


Figure 4: Evaluated fields in the region of interest as generated by Lst. 6. (a) Real part of the incident field of the 1st transmitter. (b) Real part of the measurement kernel of the 5th receiver.

3.3 Guide: Defining a Contrast

The functions generating the predefined contrasts are stored in the subfolders of the directory incontrasts. In 2D e.g. a corner, a cross, a ball and a combination of two corners and one ball are provided. Also, a contrast with one dielectric cylinder and another contrast with two dielectric cylinders as considered in the experiments from Institute Fresnel are available. In 3D IPscatt provides a tripod, a cross, a ball and the edges of a cube. For a full list see incontrastsRef.m. To build own contrasts the user can create new files in the directory incontrasts (or its subfolders)—analogously to existing functions.

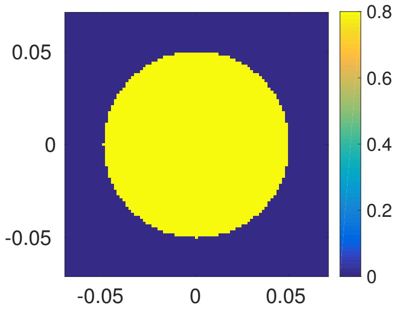
Input Arguments The input arguments are two or three coordinate arrays (depending on the dimension) and the structure array seti. The coordinate arrays are vectors with the size of the region of interest, i.e. $1 \times \text{seti.nROI}^d$ with $d = \text{seti.dim}$, see setGrid in Sec. 3.1. The structure array seti is an input argument to scale the size of the contrast by seti.rCD (as defined in setGrid). Further, some contrasts require additional input, e.g. for balls one needs to set the contrast value seti.qBall and the radius seti.rBall.

Output Argument The output argument is the contrast q stored as a vector, i.e. of size $1 \times \text{seti.nROI}^d$ with $d = \text{seti.dim}$.

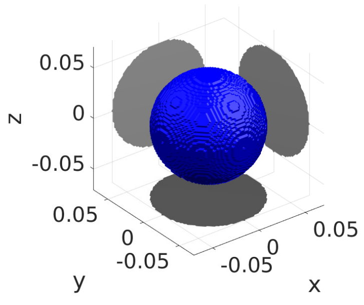
Example In Lst. 7 we consider a ball of radius 0.05 and contrast value 0.8 in two and three dimensions. The results are shown in Fig. 5. Note that the file guideSetGrid3D contains the same code as guideSetGrid apart from seti.dim = 3 instead of seti.dim = 2. Further, note that IPscatt provides the convenience function contourPlotROI to plot the surface of 3D objects.

```
init; seti.rBall = 0.05; seti.qBall = 0.8;
%
guideSetGrid;
q2D = referenceBall2D(seti.gridROI(1,:), seti.gridROI(2,:), seti);
x = seti.gridROI(1,:); y = seti.gridROI(2,:);
figure(1); imagesc(x,y,real(reshape(q2D,[seti.nROI seti.nROI]))); axis xy; colorbar;
%
guideSetGrid3D;
q3D = referenceBall3D(seti.gridROI(1,:), seti.gridROI(2,:), seti.gridROI(3,:), seti);
figure(2); contourPlotROI(q3D, seti, 'real');
```

Listing 7: Predefined contrasts in 2D and 3D (*source code*: guides/guideContrast.m).



(a) Ball in 2D.



(b) Ball in 3D.

Figure 5: Predefined contrasts in 2D and 3D as generated by Lst. 7.

3.4 Guide: Setting up Geometry and Simulation

As already mentioned IPscatt takes care that all necessary fields of the structure array seti are set in a consistent manner. The routine setGeomSim does this for the grid, the experimental set-up (transmitters' and receivers' positions, incident field and measurement kernel) and the contrast as described in Sec. 3.1, 3.2 and 3.3.

Input Arguments The input arguments are the fields of the structure array seti as described in the input and output arguments in the just-mentioned sections. In comparison to Lst. 7 the routine expects the function's name of the chosen contrast as string stored in seti.contrast. Further, in the case of two dimensions seti.rotation contains the number of degrees (default: 0) by which the contrast is rotated in the mathematically positive sense. Empty fields are set to default values.

Output Arguments The essential fields of the output argument seti were already described in the aforementioned sections except for the contrast (evaluated on the region of interest) which is saved in seti.qROIexact as a vector of size $\text{seti.nROI}^d \times 1$ with $d = \text{seti.dim}$, see setGrid in Sec. 3.1.

Example As before we consider a circle as contrast with radius 0.05 and contrast value 0.8. The experimental set-up is not defined explicitly such that default values are used in Lst. 8. Note that seti.G as used in Lst. 8 is a convenience function (defined in setGeomSim) to avoid reshape. For the demonstration of the rotation we refer to Lst. 10.

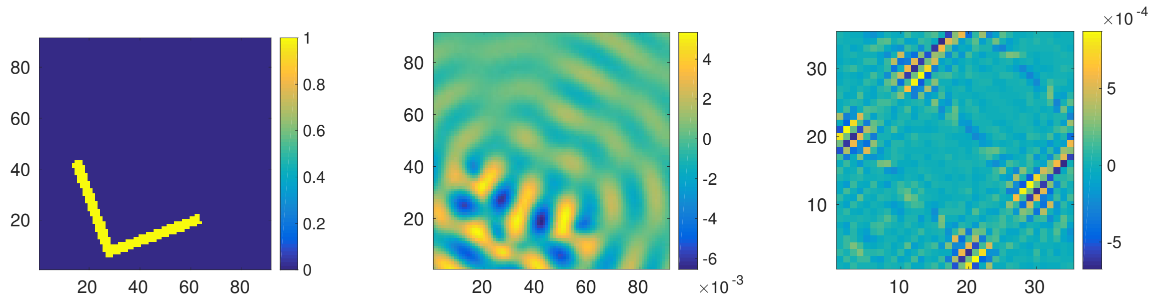
```
init; seti.rBall = 0.05; seti.qBall = 0.8;
seti.contrast = 'referenceBall2D';
seti = setGeomSim(seti);
imagesc(real(seti.G(seti.qROIexact))); axis xy; colorbar;
```

Listing 8: Set the experimental set-up and the contrast by setGeomSim (*source code*: guides/guideSetGeomSim.m).

The routine setGeomSim can be used to set the optional input argument dispDepth. This argument controls the depth of displayed messages, that is demonstrated in Lst. 9. For more information see Sec. 3.13.

```
init; clear seti; seti = struct; seti = setGeomSim(seti,1);
```

Listing 9: Routine setGeomSim with optional input argument dispDepth (*source code*: guides/guideSetGeomSimDisp.m).



(a) Predefined contrast. (b) Data. (c) Scattered field.

Figure 6: The results generated by Lst. 10. (a) The contrast corner2D rotated by 20°. (b) The resulting scattered field in the region of interest for the 7th transmitter. (c) The scattered field at the receivers' positions for each transmitter. (The experimental set-up consists of 35 transmitters and 35 receivers equidistantly arranged on a circle.)

3.5 Guide: Evaluating the Forward Operator

With the defined grid, experimental set-up and contrast we can compute the scattered field at the receivers' positions with the forward operator $\mathcal{F}$ (multi-static contrast-to-measurement operator), see (19). This is done by means of the routine forward. The reader should keep in mind that the receivers measure the total field, in particular when considering real-world data from Institute Fresnel. However, IPscatt directly provides the scattered field, i.e. the difference between the total and incident field.

Input Arguments The routine forward requires the grid, the experimental set-up and the contrast as preparations. Therefore the structure array seti is required as described in Sec. 3.1, 3.2 and 3.3. It is recommended to use setGeomSim to set all necessary fields of the structure array seti in a consistent manner, see Sec. 3.4. Further, the routine requires the contrast qROI (complex matrix stored as a vector of size seti.nROI$^d \times 1$ with $d =$ seti.dim, see setGrid in Sec. 3.1).

Output Arguments

FFqMeas	The field FFqMeas contains the data, i.e. the result of the forward operator $\mathcal{F}$ evaluated on the underlying contrast q, see (21). This result is the scattered field evaluated on receivers' positions for each transmitter. Therefore FFqMeas is a complex matrix of size seti.measNb $\times$ seti.incNb.
FFqROI	The field FFqROI stores the scattered field evaluated on the region of interest (ROI) for each of the N_i transmitters, see (12). Therefore FFqROI is a complex matrix of size seti.nROI$^d \times N_\mathrm{i}$ with $d =$ seti.dim and $N_\mathrm{i} =$ seti.incNb.
seti	The structure array seti contains the settings used for generating the other outputs. We stress that forward() sets all fields to default values. In this case these settings are interesting.

Example Lst. 10 evaluates and visualizes the scattered field in the region of interest and at the receivers' positions for each transmitter. The underlying contrast is a corner plotted with the results in Fig. 6. For more information see forward.m and mimo.m.

```
init; seti.contrast = 'corner2D'; seti.rotation = 20;
seti = setGeomSim(seti);
figure(1); imagesc(real(seti.G(seti.qROIexact))); axis xy; colorbar;
[FFqMeas,FFqROI,seti] = forward(seti,seti.qROIexact);
figure(2); imagesc(real(seti.G(FFqROI(:,7)))); axis xy; colorbar; % 7th transmitter
figure(3); imagesc(real(FFqMeas)); axis xy; colorbar;
```

Listing 10: Compute the scattered field by the routine forward for a corner-shaped contrast rotated by 20° (*source code*: guides/guideForward.m).

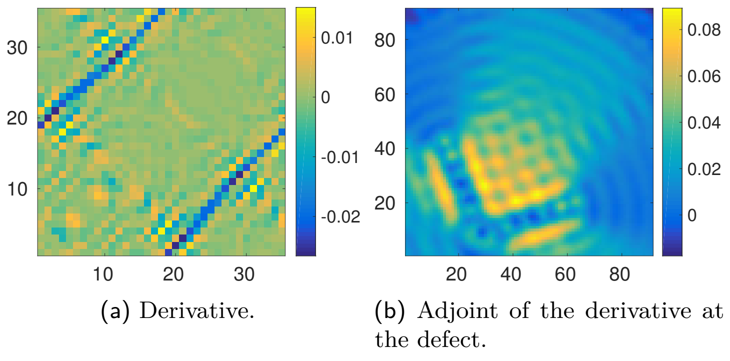


Figure 7: (a) The Fréchet derivative $\mathcal{F}'(q)[h]$ as generated by Lst. 11. (b) The adjoint of the derivative at the defect as generated by Lst. 12.

3.6 Guide: Evaluating the Fréchet Derivative (Jacobian Matrix)

In finite-dimensional spaces the Fréchet derivative $\mathcal{F}'(q)[h]$ of the forward operator $\mathcal{F}$ at the contrast q evaluated on h, see (22), is represented by the Jacobian matrix evaluated on h. This matrix is required in the variational reconstruction, see Sec. 3.11, and evaluated by the routine derivative. Therefore the routine derivative provides auxiliary matrices JA and JB which are necessary for the Jacobian matrix of $\mathcal{F}$ at the contrast q. These matrices can be used to construct $\mathcal{F}'(q)[h]$ via JA*diag(h)*JB. The corresponding function handle is denoted by DFFq in the code snippet. The result is a complex matrix of size seti.measNb $\times$ seti.incNb.

Input Arguments As the routine derivative requires the grid, the experimental set-up and the contrast as preparations, it is recommended to use setGeomSim to set all necessary fields of the structure array seti in a consistent manner, see Sec. 3.4 and Lst. 8. Apart from that, the contrast $q =$ qROI in the region of interest (complex matrix as vector of size seti.nROI$^d \times 1$ with $d =$ seti.dim) is an input argument.

Output Arguments The auxiliary matrices JA and JB are the output arguments, see (23). (These are complex matrices of size seti.measNb $\times N_\mathrm{D}$ and $N_\mathrm{D} \times$ seti.incNb with $N_\mathrm{D} =$ seti.nROI^d and $d =$ seti.dim.)

Example Lst. 11 shows how to evaluate $\mathcal{F}'(q)[h]$. We have chosen a corner-shaped contrast rotated by 20°, see Fig. 6(a), and for the sake of simplicity a constant update $h = 1 + \mathrm{i}$. The result is plotted in Fig. 7(a). For more information see derivative.m and mimo.m.

```
init; seti.contrast = 'corner2D'; seti.rotation = 20;
seti = setGeomSim(seti);
qROI = seti.qROIexact;
[JA,JB] = derivative(seti,qROI);
%
DFFq = @(h) JA*diag(h)*JB;
h = ones(size(qROI))+1i*ones(size(qROI));
DFFqh = DFFq(h);
imagesc(real(DFFqh)); axis xy; colorbar;
```

Listing 11: Compute the Fréchet derivative $\mathcal{F}'(q)[h]$ (*source code*: guides/guideDerivative.m).

3.7 Guide: Evaluating the Adjoint of the Derivative at the Defect

IPscatt can evaluate the adjoint of the derivative of the forward operator applied to the defect $\mathcal{F}(q) - F^\delta_\mathrm{meas}$ by the routine adjOfDer. That is, it computes ADFFq $= [\mathcal{F}'(q)]^*[\mathcal{F}(q) - F^\delta_\mathrm{meas}]$, which is the derivative of the least-squares error of the forward operator, see (24). For the forward operator $\mathcal{F}$, see (19) and Sec. 3.5, for the contrast $q = \mathsf{qROI}$, see Sec. 3.3, and for the data F^δ_{meas} with noise level δ, see Sec. 3.8. Although the routine adjOfDer is not part of our reconstruction scheme, we provide it, because it is an important ingredient of many other reconstruction schemes, e.g. the thresholded, nonlinear Landweber based on the so-called soft-shrinkage operator, see [CS05].

Input Arguments The function adjOfDer requires the grid, the experimental set-up and the contrast. The routine setGeomSim is recommended to set all necessary fields of the structure array seti in a consistent manner, see Sec. 3.4. Further, the contrast qROI in the region of interest (complex matrix stored as vector of size $\mathsf{seti.nROI}^d \times 1$ with dimension $d = \mathsf{seti.dim}$) and the data $\mathsf{FmeasDelta} = F^\delta_{\text{meas}}$ at receivers' positions (matrix of size $\mathsf{seti.measNb} \times \mathsf{seti.incNb}$) are required.

Output Arguments The output arguments are the adjoint of the derivative applied to the defect, ADFFq (complex matrix stored as vector of size $\mathsf{seti.nROI}^d \times 1$ with $d = \mathsf{seti.dim}$), and the structure array seti containing all parameters of the setting. This structure array is interesting if all or some fields were set automatically.

Example Lst. 12 contains the code to compute the adjoint of the derivative applied to the defect $\mathsf{FFqmF} = \mathcal{F}(q) - F^\delta_{\text{meas}}$, where $q = \mathsf{qROI}$ is the contrast from Fig. 6(a). In this simple example the data FmeasDelta is set to zero. The result is shown in Fig. 7(b).

```
init; seti.contrast = 'corner2D'; seti.rotation = 20;
seti = setGeomSim(seti);
qROI = seti.qROIexact;
FmeasDelta = zeros(seti.measNb,seti.incNb);
[ADFFq,seti] = adjOfDer(seti,qROI,FmeasDelta);
imagesc(real(seti.G(ADFFq))); axis xy; colorbar;
```

Listing 12: Compute the adjoint of the derivative at the defect (*source code*: guides/guideAdjOfDer.m).

Note that it is possible to evaluate the defect FFqmF as well as the adjoint of the derivative at the defect, ADFFq, in a single function call by [FFqmF,ADFFq] = mimo(seti,qROI,FmeasDelta,'adjOfDer'), see adjOfDer.m and mimo.m for more information.

3.8 Guide: Generating Simulated Data with Noise

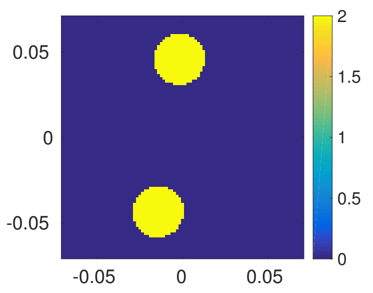
IPscatt provides the option to add a specific relative noise level $\delta > 0$ to the exact data $F_{\text{meas}} = \mathcal{F}(q)$ for the contrast q as in Sec. 3.5. This results in perturbed data

$$F^\delta_{\text{meas}} = F_{\text{meas}} + \delta \, \frac{N_{\text{Re}} + \mathrm{i} N_{\text{Im}}}{\|N_{\text{Re}} + \mathrm{i} N_{\text{Im}}\|_{\text{F}}} \|F_{\text{meas}}\|_{\text{F}},$$

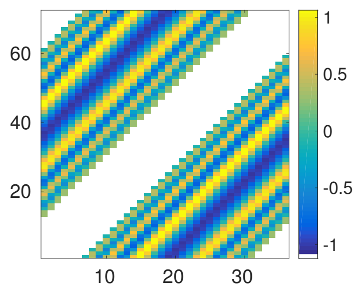
where $N_{\text{Re}}, N_{\text{Im}} \in \mathbb{R}^{N_{\text{s}} \times N_{\text{i}}}$ are two real matrices with normally distributed entries ($N_{\text{s}} = \texttt{seti.measNb}$ is the number of receivers and $N_{\text{i}} = \texttt{seti.incNb}$ the number of transmitters) and Frobenius norm $\|\cdot\|_{\text{F}}$. The corresponding function to add noise is called addNoise.

Input Arguments Input arguments of addNoise are the structure array seti with the infinitesimal element seti.dSMeas, see Sec. 3.2, and the exact data $\mathsf{FmeasExact} = \mathcal{F}(q)$, that was defined as FFqMeas in Sec. 3.5. The following fields of seti are set to default values if they are not defined.

seti.delta	Relative (artificial) noise level δ of the data (default: 0.01).
seti.whichNoise	The following values are used to choose the type of the propability density function to perturb the data: 'laplace', 'uniform', 'normal' (default). By default the mean value of all densities vanishes.
seti.seed	Non-negative integer (default: 0) to control the random number generator.



(a) Contrast of two dielectrics.



(b) Real-world data.

Figure 8: Working with data from Institute Fresnel. (a) Real part of two dielectrics' true contrast from Institute Fresnel data set generated by Lst. 15. (The positions were manually corrected.) (b) Result of Lst. 16. Some measurements of the incident field are missing due to the experimental set-up (transmitter and receiver have a minimal distance).

Output Arguments The structure array seti contains the used settings. Therefore it is only interesting as an output argument if the optional fields delta, whichNoise or seed were set automatically. The output argument FmeasDelta $= F^{\delta}_{\text{meas}}$ contains the perturbed data (a complex matrix of size seti.measNb $\times$ seti.incNb).

Example In Lst. 13 the number of 10 receivers and 5 transmitters was used. The infinitesimal element of the receivers was set to be 1 by seti.dSMeas = 1. Furthermore, a matrix with random entries is used as exact data. This data is perturbed by a normally distributed noise with a relative noise level of 5%. The result is stored in FmeasDelta.

```
init; FmeasExact = rand(10,5) + 1i*rand(10,5); seti.dSMeas = 1;
seti.delta = 0.05; seti.whichNoise = 'normal'; seti.seed = 10;
[seti, FmeasDelta] = addNoise(seti, FmeasExact);
```

Listing 13: Adding noise to exact data (*source code*: guides/guideAddNoise.m).

Example of Convenience Function setData The convenience function setData of IPscatt is helpful as it sets the geometry and simulation by setGeomSim, see Sec. 3.4, and adds the noise by addNoise. The input arguments are the same fields of the structure array seti as in Sec. 3.1, 3.2, 3.3, 3.4 and this section. The output argument is the structure array seti that contains in particular the exact data seti.FmeasExact and the perturbed data seti.FmeasDelta. The usage of setData is shown in Lst. 14. Apart from this generation of synthetic data the routine setData is useful to deal with real-world data from Institute Fresnel. This case will be considered in Lst. 19 in Sec. 3.10.

```
init; seti = struct; seti = setData(seti);
```

Listing 14: Introduction of the function setData (*source code*: guides/guideSetData.m).

3.9 Guide: Working with Real-World Data from Institute Fresnel

As one of its core features IPscatt supports users in loading and working with real-world data from Institute Fresnel. The experimental data from Institute Fresnel comprises three opuses, see ([‡]). The corresponding article to the first opus is [BS01] and the supplementary data is available at ([§]). Likewise, the corresponding article to the second opus is [GSE05] and the supplementary data is available at ([¶]). The user must store the

[‡] http://www.fresnel.fr/3Ddatabase/database.php (Accessed: 20160921).
[§] http://dx.doi.org/10.1088/0266-5611/17/6/301 (Accessed: 20160921).
[¶] http://dx.doi.org/10.1088/0266-5611/21/6/S09 (Accessed: 20160921).

*.exp-files of the first opus in the directory inexpdata/fresnel_opus_1 and the *.exp-files of the second opus in the directory inexpdata/fresnel_opus_2 since IPscatt expects them there.

To read the experimental data the file readRAWData.m is provided in the folder proc/expData. This routine can be used independently of IPscatt. Note that IPscatt supports data with transverse magnetic (TM) polarization (instead of transverse electric, TE), because the Helmholtz equation only models electromagnetic waves in the case of TM, see e.g. [CK13].

Example Lst. 15 generates the true contrast of two dielectrics resulting in Fig. 8(a). The corresponding TM polarized data is loaded by Lst. 16. The output is shown in Fig. 8(b).

```
init;
guideSetGrid;
q2D = fresnel_op1_twodielTM(seti.gridROI(1,:), seti.gridROI(2,:),seti);
x = seti.gridROI(1,:); y = seti.gridROI(2,:);
imagesc(x,y,real(reshape(q2D,[seti.nROI seti.nROI]))); axis xy; colorbar;
```

Listing 15: True contrast of two dielectrics (*source code*: guides/guideWorkingFresnelContrast.m).

```
init;
filename = 'inexpdata/fresnel_opus_1/twodielTM_8f.exp';
[uTotRX, uIncRX, frequencies, rTX, nTX, rRX, nRX] = readRAWData(filename);
imagesc(real(uIncRX(:,:,1))); colorbar; axis xy;
```

Listing 16: Reading data from Institute Fresnel (*source code*: guides/guideWorkingFresnel.m).

Input and Output Arguments of readRAWData The input argument filename of the routine readRAWData is the path to the file with real-world data from Institute Fresnel. The outputs are:

nTX	Number of transmitters.
nRX	Number of receivers.
rTX	Radius of the circle on which the transmitters are arranged (in meters).
rRX	Radius of the circle the receivers are arranged on (in meters).
frequencies	Available frequencies as a vector of size *number of frequencies* $\times$ 1.
uIncRX	Incident field (i.e. without any obstacle) at receivers' positions for each transmitter and each frequency (complex array of size nRX $\times$ nTX $\times$ *number of frequencies*).
uTotRX	Total field (i.e. with an obstacle) at receivers' positions for each transmitter and each frequency (complex array of size nRX $\times$ nTX $\times$ *number of frequencies*).

3.10 Guide: Matching the Incident Fields of Institute Fresnel's Data

Institute Fresnel provides real-world data of the incident fields at the receivers' positions. For further computations we are actually interested in the corresponding incident field in a region instead of the incident field at the receivers' positions. Assuming point sources at the transmitters' positions this matching is achieved by the function matchIncField, that only works in 2D. For this matching we follow [Geh13, Sec. 6.2.2] and [BKL17, Sec. 6].

Details of matchIncField The function [uIncROI,errC] = matchIncField(uIncRX,seti,region) matches incident fields uIncROI in the computational domain (CD) or the region of interest (ROI) (depending on the value of region) corresponding to the data uIncRX, that contains all incident fields at the receivers' positions. This matching is done by computing the coefficients of the two-dimensional radiating series solutions to the Helmholtz equation, that fits the data at the best in a least-squares sense. Furthermore, it computes the relative error errC of the matched incident field at the receivers' positions for each incident field.

Input Arguments of matchIncField Remember that IPscatt expects a two-dimensional problem for matching, i.e. seti.dim = 2. Further input arguments are the number of transmitters seti.incNb, the wave number seti.k, the transmitters' positions seti.incPnts (matrix of size 2 $\times$ seti.incNb)—e.g. seti.incPnts = [5 -2; 0 4] describes coordinates $(5, 0)$ and $(-2, 4)$—, the receivers' positions seti.measPnts (matrix of size 2 $\times$ seti.measNb), the grid of the region of interest (ROI) seti.gridROI (if region = 'ROI') (matrix of size seti.dim $\times$ seti.nROI^2) or the grid of the computational domain (CD) seti.grid (if region = 'CD') (matrix of size seti.dim $\times$ seti.nCD^2). The other input arguments are:

seti.nuMax	Approximation order (default: 7). (A good choice is essential for a good matching.)
uIncRX	Incident field at receivers' positions for each transmitter (complex matrix of size seti.measNb $\times$ seti.incNb).
region	'ROI' (region of interest) or 'CD' (computational domain), usually interesting is 'ROI'.

Output Arguments of matchIncField

uIncROI	Incident field in ROI (or CD) for each transmitter (complex matrix of size seti.nROI^2 $\times$ seti.incNb).
errC	Stores the relative error of the matched incident field at the receivers' positions for every transmitter (vector of size 1 $\times$ seti.incNb).

Example of matchIncField Lst. 17 reads the same data as Lst. 16. For the matching of the incident field it picks the data with a frequency of 5 GHz. The result is shown in Fig. 9(a). The data reading, the generation of a grid, the transfer of the experimental set-up into the structure array seti including the computation of the wave number and setting transmitters' as well as receivers' positions was moved into the routine matchIncFieldTrans, that is available in guides/auxi/matchIncFieldTrans.m.

```
init;
[seti,uTotRX,uIncRX,uScaRX] = matchIncFieldTrans('inexpdata/fresnel_opus_1/twodielTM_8f.exp',5*1E9);
seti.nuMax = 7;
[uIncROI,errC] = matchIncField(uIncRX,seti,'ROI');
imagesc(real(reshape(uIncROI(:,6),[seti.nROI seti.nROI]))); colorbar; axis xy;
```

Listing 17: Matching the incident field (*source code*: guides/guideMatchIncField.m).

Convenience Function loadData The routine loadData is a convenience function to load data from Institute Fresnel in three steps: Initially, it reads the data by readRAWData as in Sec. 3.9. Then it sets geometry and simulation by setGeomSim as in Sec. 3.4 (considering required settings for Insitute Fresnel's data, in particular the transmitters' and receivers' positions). Finally, it matches the incident field by matchIncField.

Input Arguments of loadData The input arguments of loadData are fields of the structure array seti. The most important ones are the path to the data given in the field seti.fresnelFile (default: 'inexpdata/fresnel_opus_1/twodielTM_8f.exp') and the chosen frequency seti.fresnelFreq (default: 5*1E9, i.e. 5 GHz). It is recommended to set all required fields in a consistent manner before executing loadData by setting seti.expSetup = 'fresnel' and using checkConsisExpData. This comprises the parameter seti.nuMax, that was introduced just above, and the input arguments seti.rCD as well as seti.nCD, that are associated to the grid, see Sec. 3.1 (as input arguments of setGrid). In particular, some fields are reseted in order to ensure consistency with Institute Fresnel's data, that requires a two-dimensional problem, point sources and near field data. Therefore the routine sets seti.dim = 2, seti.incType = 'pointSource' and seti.measType = 'nearField'.

Output Arguments of loadData The most important output arguments of loadData are the following.

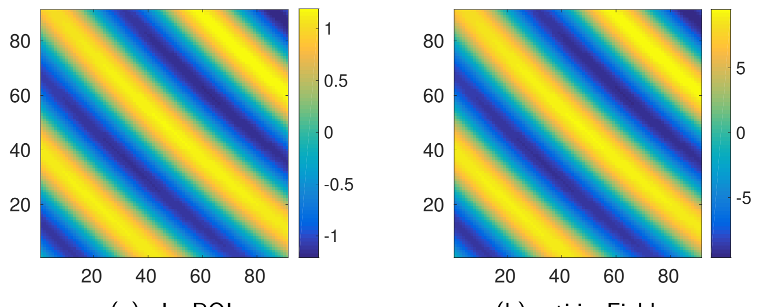


Figure 9: Matched incident field of the 6th transmitter in the region of interest as uIncROI in (a) and scaled as seti.incField in (b) as generated by Lst. 17 and 18.

seti.incNb	Number of transmitters.
seti.measNb	Number of receivers.
seti.radSrc	Radius of the circle on which the transmitters are arranged (in meters).
seti.radMeas	Radius of the circle on which the receivers are arranged (in meters).
seti.k	Wave number ($k = 2\pi f/c$ with frequency f and light velocity c in vacuum).
seti.FmeasDelta	Data, i.e. the scattered field at the receivers' positions for each transmitter (complex matrix of size seti.measNb $\times$ seti.incNb).
seti.incField	Incident field in ROI for each transmitter (complex matrix of size seti.nROI^2 $\times$ seti.incNb). (This field is uIncROI divided by the infinitesimal element seti.dSInc, see Output Arguments of matchIncField for uIncROI and Sec. 3.2 for seti.dSInc.)

Example of loadData Lst. 18 demonstrates an application of loadData.

```
init;
seti.expData = 'fresnel';
seti.rCD = 0.2; seti.nCD = 256;
seti.fresnelFreq = 5*1E9; seti.fresnelFile = 'inexpdata/fresnel_opus_1/twodielTM_8f.exp';
seti.nuMax = 7;
seti = checkConsisExpData(seti,1);
seti = loadData(seti);
imagesc(real(seti.G(seti.incField(:,6)))); colorbar; axis xy;
```

Listing 18: Process Institute Fresnel's data with loadData (*source code*: guides/guideLoadData.m).

Example of setData in the Case of Institute Fresnel's Data We have already seen that the convenience function setData comprises to set geometry as well as simulation and to add noise to the data, see Lst. 14 in Sec. 3.8. Apart from this generation of synthetic data the routine setData is useful to deal with real-world data from Institute Fresnel because it employs checkConsisExpData and loadData in the case of seti.expData = 'fresnel'. Therefore Lst. 19 is the same as Lst. 18 except for the geometry.

```
init;
seti.expData = 'fresnel';
seti.fresnelFreq = 5*1E9; seti.fresnelFile = 'inexpdata/fresnel_opus_1/twodielTM_8f.exp';
seti = setData(seti);
```

Listing 19: Process Institute Fresnel's data with setData (*source code*: guides/guideSetDataFresnel.m).

3.11 Guide: Variational Reconstruction

Before executing the variational reconstruction by the routine recon we have to set the fields of the structure array seti for the forward operator in a consistent manner. Therefore it is recommended to use the function setData, see Sec. 3.8 and 3.10. Furthermore, the routine recon requires reconstruction-specific fields of seti. For this preparation IPscatt provides the function setRecon, which is the subject of the first part of this guide.

Details of setRecon The routine setRecon consists of the functions setInvType, checkConsisRec and setFuncsPda with the mentioned structure array seti as input and output argument.

In setInvType some default parameters for the variational reconstruction are set in dependent on the inversion method number seti.invNo. We will confine ourselves to discussing the default choice of seti.invNo = 6. In this case IPscatt uses the variational reconstruction scheme which is described in the *algorithm paper* and in detail in [BKL17, Sec. 4]. As the reconstruction scheme relies on the primal-dual algorithm, see [CP11], the reconstruction's type is chosen as seti.inv = 'pda'. Further, the minimization functional for the inner iteration is essentially (5) and the assignment $F(Kh) = f_{\mathrm{dis}}(h) + f_{\mathrm{spa}}(h)$ as well as $G(h) = f_{\mathrm{spa}}(h) + f_{\mathrm{phy}}(h)$ is used for the required split of the minimization functional into two parts F and G with $\min_{h \in \mathbb{C}^{N_{\mathrm{D}}}} F(Kh) + G(h)$, where K is a continuous linear operator.

The routine setFuncsPda defines the functions for the primal-dual algorithm, i.e. essentially f_{dis}, f_{spa}, f_{tv}, f_{phy} as well as K_{dis} and K_{tv} (the parts of the linear operator K) as fields fd, fs, fg, fp, Kd, Kg of the structure array seti and their adjoints KdAdj and KgAdj, see setFuncsPda.m.

Output and Optional Input Arguments of setRecon If they are not already defined, the function setRecon sets all reconstruction-specific fields of seti. This is done in a consistent and appropriate manner. In particular, the following main fields are set.

seti.alpha	Regularization parameter α for the sparsity penalty (default: 500), see (5).
seti.beta	Regularization parameter β for the total variation penalty (default: 10^{-5}), see (5).
seti.tau	Tolerance parameter τ of the discrepancy principle (default: 2.5), see (6).
seti.physBounds	Bounds for real/imaginary part of contrast: $[a, b, c, d]$ (default: [-1,3,0,3]), see (5).
seti.useDis	0 or 1 (default): Set this to 1 to stop the outer iterations by the discrepancy principle.
seti.nOut	Maximal number of the outer iterations (default: 30).
seti.pdaN	Number of the inner iterations (primal-dual algorithm) (default: 50).
seti.useTolOut	0 (default) or 1: Set this to 1 to stop the inner iterations by the outer tolerance principle, see stopping strategy 1 in Sec. 3.12.
seti.useTolIn	0 (default) or 1: Set this to 1 to stop the inner iterations (primal-dual algorithm) by the inner tolerance principle, see stopping strategy 2 in Sec. 3.12.

For more information see setRecon.m, setInvType.m, checkConsisRec.m and setFuncsPda.m.

Details of recon The routine recon executes the above-mentioned variational reconstruction scheme of the contrast. It uses the functions minPda for the outer and pda for the inner iteration. The relative discrepancy of the reconstructed contrast $q^{(m)}$ after m outer iterations is $\mathrm{dis}(m) := \|\mathcal{F}(q^{(m)}) - F^{\delta}_{\mathrm{meas}}\|_{\mathrm{F}} / \|F^{\delta}_{\mathrm{meas}}\|_{\mathrm{F}}$, where $m = 1, 2, \ldots$ is the index of the outer iteration. Furthermore, the relative error is $\mathrm{err}(m) := \|q^{(m)} - q_{\mathrm{exa}}\|_2 / \|q_{\mathrm{exa}}\|_2$, where q_{exa} is the exact contrast.

Additionally, recon provides a way to load the reconstruction result from a previous computation—to plot and save reconstruction-depending figures—and offers the opportunity to save discrepancies, errors and reconstruction results. This is helpful for prototyping purposes and is presented in Sec. 3.13.

Input Argument of recon As already mentioned the routine recon requires fields of the structure array seti, that were defined in setData, see Sec. 3.8 and 3.10, and setRecon.

Output Argument of recon The most important fields of the output argument seti are the following.

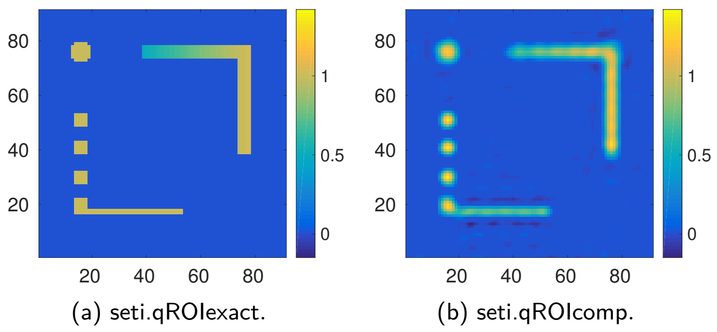

(a) seti.qROIexact. (b) seti.qROIcomp.

Figure 10: Real part of the exact and reconstructed contrast in (a) and (b) as generated by Lst. 20.

seti.qROIcomp	Computationally reconstructed contrast (complex vector of size seti.nROI^d $\times$ 1 with d = seti.dim).
seti.iOutStop	Stop index of outer iterations.
seti.dis	Relative discrepancy for each outer iteration (vector of size $1 \times$ seti.nOut).
seti.err	Relative error for each outer iteration (vector of size $1 \times$ seti.nOut).

Example Lst. 20 reconstructs the contrast from perturbed data and compares it with the exact one (seti.qROIexact, see Sec. 3.3). The run time is approximately 1 min and the results are shown in Fig. 10. The reconstruction stops after seti.iOutStop = 11 outer iterations with a relative discrepancy of seti.dis(seti.iOutStop) = 0.024 and a relative error of seti.err(seti.iOutStop) = 0.359. Remember that the number of inner iterations is fixed to 50 by the parameter seti.pdaN in this example; see Sec. 3.12 for more sophisticated methods.

```
init;
seti = struct;
seti = setData(seti);
seti = setRecon(seti);
seti = recon(seti);
figure(1); imagesc(real(seti.G(seti.qROIexact))); axis xy; colorbar;
figure(2); imagesc(real(seti.G(seti.qROIcomp))); axis xy; colorbar;
```

Listing 20: Variational reconstruction (*source code*: guides/guideRecon.m).

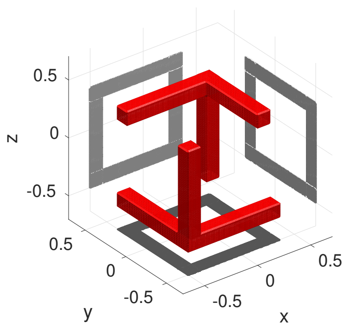
Lst. 21 computes the same as Lst. 20, but displays messages of the functions, because of the choice of the additional input argument dispDepth to 4. This additional input argument was already mentioned in Sec. 3.4. For the details we refer to Sec. 3.13.

```
init; seti = struct; seti = setData(seti,4); seti = setRecon(seti,4); seti = recon(seti,4);
```

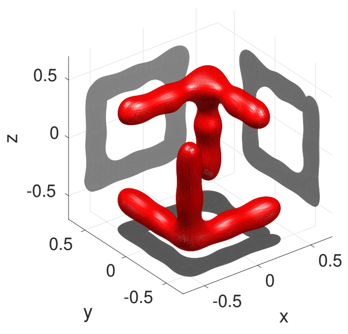
Listing 21: Variational reconstruction with display of messages (*source code*: guides/guideReconDetails.m).

Example in Three Dimensions Lst. 22 shows a three-dimensional reconstruction of two tripods. The results are plotted in Fig. 11. Note that the used convenience function contourPlotROI was already mentioned in Sec. 3.3. Furthermore, note that there is no text information displayed during the run time of several hours; see Lst. 21 to change this behavior.

For more information about the variational reconstruction see recon.m.



(a) seti.qROIexact.



(b) seti.qROIcomp.

Figure 11: Real part of the exact and reconstructed contrast in (a) and (b) as generated by Lst. 22. The reconstruction stops after 9 outer iterations with a relative discrepancy of 0.012 and a relative error of 0.659. The run time was 4.1 h.

```
init;
seti.dim = 3; seti.rCD = 2.0; seti.k = 10;
seti.contrast = 'twoTripods3D';
seti.incNb = 35; seti.measNb = 35;
seti.radSrc = 5; seti.radMeas = 5;
seti.tau = 1.25;
seti = setData(seti); seti = setRecon(seti); seti = recon(seti);
figure(1); contourPlotROI(seti.qROIexact, seti, 'real');
figure(2); contourPlotROI(seti.qROIcomp, seti, 'real');
```

Listing 22: Variational reconstruction in three dimensions (*source code*: guides/guideRecon3D.m).

3.12 Guide: Stopping Strategies of the Inner Iteration

In Sec. 3.11 the reconstruction uses a fixed number of inner iterations as defined by seti.pdaN. To use more sophisticated stopping criteria, mentioned in the *algorithm paper* and explained in minTolOut.m and minTolIn.m, IPscatt provides stopping strategies 1 and 2. They are activated by seti.useTolOut = 1 and seti.useTolIn = 1. If a stopping strategy is activated and strategy-dependent fields of seti are not defined, they are set to default values in setRecon (to be precisely in checkConsisRec). A field, that is set in both cases, is the maximal number of inner iterations by seti.pdaNmax (default: 250).

In the first case (stopping strategy 1) the outer tolerance (default: 0.05) in seti.relDisTol is set too, see minTolOut.m. In the second case (stopping strategy 2), these additional fields of seti are ThetaStart (default: 0.925), ThetaMax (default: 0.95) and TolGamma (default: 0.90) to compute the inner tolerance ThetaiOut, see minTolIn.m. (The second strategy follows an inexact stopping rule for a Newton-like method, see [Rie01].)

Example Lst. 23 reconstructs as in Lst. 20, but with activated stopping strategy 1. To use the second stopping strategy in this example replace seti.useTolOut = 1 by seti.useTolIn = 1.

```
init;
seti.useTolOut = 1;
seti = setData(seti); seti = setRecon(seti); seti = recon(seti);
```

Listing 23: Variational reconstruction with stopping strategy 1 (*source code*: guides/guideStop.m).

3.13 Guide: Setting Up Test Bench for Variational Reconstruction

The proceeding to identify the contrast was described in Lst. 21 in Sec. 3.11. It is a common scenario for the inverse medium problem. To omit a repetition of these steps IPscatt provides the convenience function start. In addition to the steps in Lst. 21, this function plots the results and saves these figures as well as other files in a subfolder of the directory output. Note that start closes all figures and usually clears almost all variables.

Details of setInput The just-mentioned typical closing of figures and clearing of variables are tasks of setInput. Additionally, it creates a directory with path seti.dirname in the folder output for the later storage of figures and other files. Note that setInput evaluates the input argument inseti (in the case of existence) to load input parameters, i.e. fields of the structure array seti, that were saved in the corresponding file.

Input Arguments of setInput Note that all input arguments are optional.

inseti	File's name in the folder inseti containing input parameters as fields of the structure array seti.

For the input arguments usevaralpha, usevarbeta and usevardelta (each 0 or 1) we refer to Sec. 3.15. The following fields of the structure array seti can be set in the file whose name the variable inseti refers to. They are set to default values if they are not defined by the user. We omit to explain seti.dirSuffix, seti.dirSuffixAdd and seti.fileSuffix because it is clear from the context.

seti.dirOutput	Name of folder in which directories for output files/figures are created (default: 'output').
seti.dirDatetime	Date and time separated by the character "T" in the format <yyyyMMdd>T<HHmmss>, e.g. 20161006T105735.
seti.dirname	Dirname for files and figures from current computation, default: <seti.dirOutput>/<seti.datetime>_<seti.inseti><seti.dirSuffix><seti.dirSuffixAdd> If seti.inseti is empty, 'noinseti' is inserted; e.g. output/20161006T102740_noinseti.

Output Arguments of setInput Most fields of seti were already described as input arguments. Note that seti.inseti = inseti or is empty, if inseti does not exist.

Example Before we discuss the example we recommend to start MATLAB with the option nodisplay as indicated in Lst. 24 because this avoids verbose program output. Regardless, the related figures are stored.

```
matlab −nodisplay
```

Listing 24: Using MATLAB with the option nodisplay.

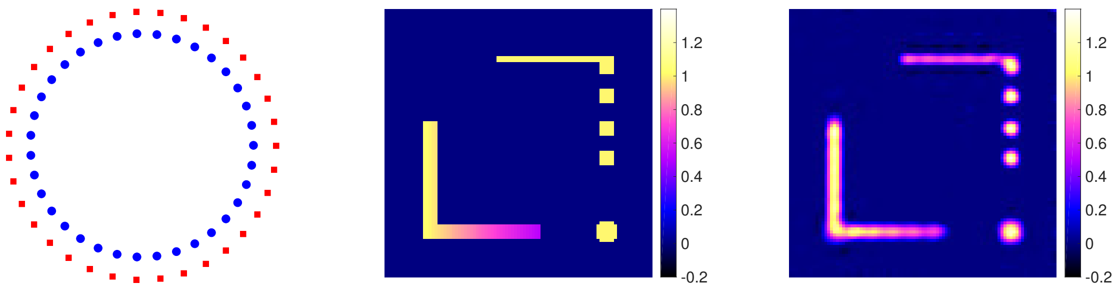
To get an idea of start its essential functions are demonstrated in Lst. 25.

```
inseti = 'exampleMod'; init; setInput;
seti = setData(seti,4,2); seti = setRecon(seti,4,2); seti = recon(seti,4,2);
```

Listing 25: Variational reconstruction with messages and storing of figures/files (*source code*: guides/guideReconOut.m).

As already mentioned all input parameters are stored in the structure array seti. In this example they were loaded from the file inseti/exampleMod.m. Changes of parameters must be done in such a file, since almost all variables are cleared in the routine setInput. Therefore IPscatt provides the file example.m, that contains the most important parameters and a short explanation. The user can change this file or create own files and name them. However, it is important to choose names other than existing functions in IPscatt.

Messages in Lst. 25 are displayed because dispDepth was set to 4. This already-mentioned additional input argument controls the depth of displayed messages by an integer between 0 and 5 (messages are suppressed by 0). Furthermore, the optional input argument out was set to 2 to plot and save figures as well as files; any storing would be suppressed by 1, plotting would be suppressed by 0. The plots are stored in the created subfolder in the directory output. The most important figures are explained in Fig. 12. A full list of stored



(a) Transmitters and receivers. (b) Predefined contrast. (c) Reconstructed contrast.

Figure 12: The most important figures generated by start of IPscatt, see Lst. 26. (a) The positions of the transmitters (blue circles) and receivers (red squares) (filename: fig_01_iOut_00.png). (b) The real part of the predefined contrast (filename: fig_02_iOut_00.png). (c) The real part of the reconstruction (filename: fig_14_iOut_11.png). (Note that the number 11 changes because it is the number of the outer iterations.) The relative noise level was $\delta = 0.01$. The run time was 4 min, the relative discrepancy 0.028 and the relative error 0.358.

figures is in start.m. A folder with the current date and time is created for each computation and figures are saved as png by default (other formats are possible, see example.m).

To get the same result as in Lst. 25 it is easier to use the convenience function start as in Lst. 26.

```
inseti = 'exampleMod'; start;
```

Listing 26: Convenience function start.

It is also possible just to use start without defining inseti. Then default input parameters are used. Of course, it is possible to use setInput and avoid inseti by defining fields of seti after setInput as demonstrated in Lst. 27.

```
init; setInput;
seti.contrast = 'cornerBallSparseMod2D';
seti = setData(seti,4,2); seti = setRecon(seti,4,2); seti = recon(seti,4,2);
```

Listing 27: Control the running of all processes (*source code*: guides/guideSeti.m).

Store and Load Simulated Data The routine setData provides a way to save and load the exact and perturbed data seti.FmeasExact as well as seti.FmeasDelta for further computations. This is, for example, helpful to avoid the "inverse crime", explained in Sec. 3.14. They are saved as FmeasExact and FmeasDelta in the file save_Fmeas.mat inside the folder of the path seti.dirname if no field expData in seti exists and out equals 2.

To load them, the path in seti.loadFmeas has to bet set. (If the field is empty, i.e. in the code '', nothing is loaded.) The stored perturbed data is usually used because seti.useFmeasDelta is set to 1 by default. Perturbed data is generated again from the stored exact data if it is set to 0.

Store and Load the Reconstruction It is also possible to store and load the result of the reconstruction process, e.g. to continue it. The following input arguments belong to recon and setRecon (to be exactly they are in checkConsisRec). However, we mention them here. If the user does not define them, they are set to default values. In addition, the setting is only useful in the case of an existing folder to store the files.

seti.loadqROIcomp	Path to load a mat-file containing the reconstructed contrast qROIcomp and the number of outer iterations iOutStop (default: empty, i.e. in the code '' or [], then no data is loaded).
seti.saveqROIcomp	Path to store the reconstructed contrast. The default filename is essentially save_qROIcomp_iOutStop.mat in the same subfolder of output containing the figures.
seti.savedata	0 (default) or 1: If this is set to 1, relative discrepancies, errors, and differences of the iterated contrasts are saved (as save_dis.mat, save_err.mat and save_dif.mat).

Example of Storing and Loading the Reconstruction Lst. 28 does the default reconstruction of perturbed simulated data and creates a new subfolder in the folder output, e.g. 20170308T143238_noinseti. The reconstruction was stopped after 11 outer iterations (by the discrepancy principle) and took 3.5 min.

```
inseti = ''; start;
```

Listing 28: A reconstruction process to be continued.

Two steps are necessary to continue the reconstruction: First, the computed contrast is loaded by setting seti.loadqROIcomp to output/20170308T143238_noinseti/save_qROIcomp_iOutStop.mat under the use of seti.dirname, see Lst. 29. Second, the tolerance parameter τ = seti.tau of the discrepancy principle (6) is decreased from 2.5 to 2.0; it was set to 2.5 by default in Lst. 28. Note that some files in this folder are overwritten by new ones. The continued reconstruction is stopped after 21 outer iterations (by the discrepancy principle), i.e. the continuation did outer iterations number 12 to 21.

```
seti.tau = 2.0;
seti.loadqROIcomp = sprintf('%s/save_qROIcomp_iOutStop.mat',seti.dirname);
seti = setRecon(seti,4,2); seti = recon(seti,4,2);
```

Listing 29: The continuation of Lst. 28 (*source code*: guides/guideConvReconStoreLoadCont.m).

3.14 Guide: Avoiding the Inverse Crime

The forward operator is just a physical model of the reality. Therefore the usage of the same forward operator for the generation and inversion of synthetic data leads to reconstructions that are "too good to be true", see [KS05] or [MS12]. This problem is known in literature as "inverse crime", see [CK13]. To avoid it we follow [MS12, Ch. 2.3.6] and generate synthetic data on a fine grid but reconstruct on a coarse grid not sharing common factors, e.g. the computational domain is discretized by $N = 1127$ points in each dimension in 2D ($N = 563$ in 3D) for computing synthetic data and by $N = 256$ for reconstructions (in 2D and 3D), see [BKL17, Sec. 5].

Therefore we will consider an example of storing and loading simulated data to avoid the "inverse crime": Lst. 30–32 show how to avoid it by generating perturbed data on a fine grid (with 1127 discretization points in each dimension) before preparing the reconstruction on a coarse grid (with 256 points in each dimension). First, to store the data in a folder that does not depend on time, we create a file in the folder inseti to redefine seti.dirname, see Lst. 30.

```
seti.dirOutput = 'output';
seti.dirname = sprintf('%s/storeLoadSim',seti.dirOutput);
seti.nCD = 1127;
```

Listing 30: File inseti/guideConvSimSaveIn.m with input parameters to save data.

Second, the source code in Lst. 31 involves this file in inseti, such that setInput creates the directory storeLoadSim in the folder output and the routine setData stores the exact and perturbed data in output/storeLoadSim/save_Fmeas.mat. (This process takes 1 min.)

```
inseti = 'guideConvSimSaveIn';
init; setInput; seti = setData(seti,4,2);
```

Listing 31: Simulate data and save them (*source code*: inseti/guideConvSimSave.m).

It is also possible to use the code snippet inseti = 'guideConvSimSaveIn'; start; to run the whole process including reconstruction and terminate it before the reconstruction starts, i.e. when the message "## setRecon – variational reconstruction" appears.

Third, seti.loadFmeas is set to the just stored file to load the data as in Lst. 32. Afterwards the reconstruction can be started by seti = setRecon(seti); seti = recon(seti);. Note that seti.nCD is set to the default value 256, because the field does not exist.

```
clear all; init;
seti.loadFmeas = 'output/storeLoadSim/save_Fmeas.mat';
seti = setData(seti);
```

Listing 32: Load exact and perturbed data (*source code*: guides/guideConvSimLoad.m).

The most convenient method to avoid the "inverse crime" is: first, create a file in the folder inseti with all desired fields of seti, increase the discretization seti.nCD, e.g. to 1127, and leave seti.loadFmeas empty; second, start the whole process by start and terminate it before the reconstruction starts; third, set in the created file (from the first step) both seti.loadFmeas to the just now stored save_Fmeas.mat and the discretization seti.nCD to a smaller value, e.g. 256; and fourth, start the whole process by start.

3.15 Guide: Choosing the Regularization Parameters

A careful selection of the regularization parameters α and β is necessary for a pleasing reconstruction. This requires extensive numerical experiments. Therefore IPscatt provides the functions varalpha and varbeta. They employ the routine start several times trying to identify the contrast with chosen various inputs of α and β.

For example, to try reconstructions with regularization parameter α set to 100, 500 and 1000, it has to be set alpha = [100; 500; 1000]; in the file varalpha.m. Afterwards it is started as in Lst. 33. Of course, a file in the folder inseti can be chosen to set deviating values from default.

```
inseti = ''; varalpha;
```

Listing 33: Various regularization parameters.

For a systematic search of reasonable regularization parameters we recommend first to set α = seti.alpha = 0 and find a suitable β = seti.beta and second to look for a proper α, see [BKL17, Sec. 5].

In addition, IPscatt provides the routine vardelta to try out different noise levels δ.

4 Applications

We demonstrate the modular design of IPscatt on three examples: first, the computation of the Born approximation and the scattered field, second, the Born approximation of the inverse medium problem and third, the linearization at a specific contrast. We hope this encourages users of IPscatt to take parts of it and use them for their own applications.

Comparison of Born Approximation and Scattered Field The Born approximation $u_{\mathrm{B}}(q) = V(q \cdot u^{\mathrm{i}})$, see [KG08, Ch. 4.2] or [Bor26], is linear and approximates the scattered field u^{s} for relatively small wave numbers k = seti.k, see Fig. 13(a), by ignoring the repeated scattering inside the region of interest. (The source code generating Fig. 13(a) can be found in guides/guideBornk.m.) In Lst. 34 the volume potential operator V, see (9), is defined and the Born approximation uBorn is computed for a given incident field $u^i =$ uIncROI and contrast $q =$ qROI. Further, the scattered field $u^s =$ uScattROI is computed by solving the Lippmann-Schwinger integral equation (3). Of course, if the Born approximation was computed, the result can also used as second input argument of solveLippmannSchwinger.

```
% Generate uIncROI (e.g. of 1st transmitter) and qROI:
init; seti.k = 100; seti = setData(seti); uIncROI = seti.dSInc.*seti.incField(:,1); qROI = seti.qROIexact;
% Define the volume potential operator V and compute uBorn and uScattROI:
V  = @(x) seti.k^2.*helmholtz2Dr2r(x, seti);
QU = @(x) qROI .* x;
uBorn = V(QU(uIncROI));
uScattROI = solveLippmannSchwinger(@(x) V(QU(x)), V(QU(uIncROI)), seti);
```

Listing 34: Comparison of Born approximation and scattered field (*source code*: guides/guideBorn.m).

Born Approximation of the Inverse Medium Problem The Born approximation of the inverse medium problem in scattering is the linearization at 0. In the variational reconstruction scheme of IPscatt the evaluation of the forward operator $\mathcal{F}(q+h)$ is approximated by its linearization $\mathcal{F}'(q)[h] + \mathcal{F}(q)$, see (5), such that $\mathcal{F}(h) \approx \mathcal{F}'(0)[h]$ for the linearization at $q = 0$. As the reconstruction starts with the initial contrast $q = 0$, it is sufficient to restrict the number of outer iterations seti.nOut to 1 and set a high number of inner iterations seti.pdaN to employ the Born approximation of the inverse medium problem, see Lst. 35. For Fig. 13(b) it was chosen seti.pdaN $= 1000$ and the wave number seti.k $= 70$ (in reciprocal meters). Note that the wave number k was chosen as compromise because the Helmholtz equation (2) is highly nonlinear for a great k (compared to the obstacle), such that the Born approximation has a high error, but the scattering effect of the obstacle is small in the case of a low k—both yield to bad reconstructions.

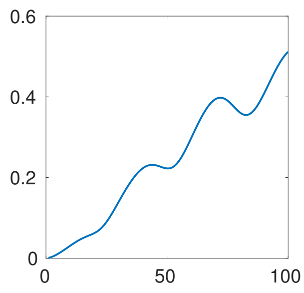
```
init; seti.nOut = 1; seti.pdaN = 1000; seti.k = 70;
seti = setData(seti); seti = setRecon(seti); seti = recon(seti);
imagesc(real(seti.G(seti.qROIcomp))); colorbar; axis xy;
```

Listing 35: Born approximation of the inverse medium problem (*source code*: guides/guideBornInv.m).

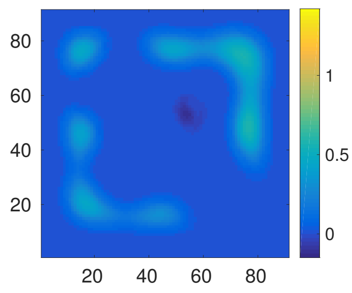
Linearization at a Specific Contrast Another application of IPscatt is the linearization of the forward operator $\mathcal{F}$, see (19), at a specific contrast q. An elementary way to compute FFqhMeas $= \mathcal{F}(q+h)$ and its linearization FFqhMeasLin $= \mathcal{F}'(q)[h] + \mathcal{F}(q)$ is given in Lst. 36, see also Sec. 3.5 and 3.6 for the forward operator and its derivative. It is necessary and sufficient to employ setKernel, setIncField and setMeasKer after changing the wave number $k =$ seti.k. The relative error of the linearization in comparison to the forward operator in dependence of the wave number k is given in Fig. 13(c), the corresponding source code in guides/guideLin.m.

```
init; seti.incNb = 1; seti.measNb = 2; seti = setData(seti);
q = seti.qROIexact; h = 0.1.*(rand(size(q)) + 1i*rand(size(q)));
seti.k = 50; seti = setKernel(seti); seti = setIncField(seti); seti = setMeasKer(seti);
% F(q+h):
  [FFqhMeas,~,~] = forward(seti,q+h);
% Linearization F'(q)[h] + F(q):
  [FFqMeas,~,~] = forward(seti,q); [JA,JB] = derivative(seti,q);
  FFqhMeasLin = JA*diag(h)*JB + FFqMeas;
```

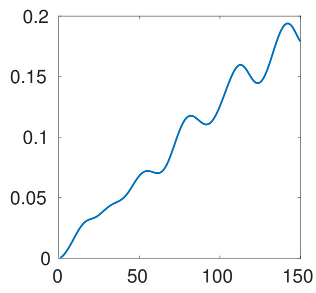
Listing 36: Linearization at a specific contrast (*source code*: guides/guideLin.m).



(a) Error of the Born approximation.



(b) Born approximation of the inverse medium problem.



(c) Error of the linearization.

Figure 13: (a) Relative error of the Born approximation (in comparison to the scattered field) in dependence of the wave number k (in reciprocal meters). (b) Reconstruction of Fig. 10(a) with the Born approximation of the inverse medium problem using wave number $k = 70\,\mathrm{m}^{-1}$ and seti.pdaN $= 1000$ inner iteration steps. Run time 1.4 min, relative discrepancy 0.34, relative error 0.78. (c) Relative error of linearization $\mathcal{F}'(q)[h] + \mathcal{F}(q)$ in comparison to $\mathcal{F}(q+h)$ in dependence of the wave number k.

Summary

In this *user guide* we have given installation instructions of the toolbox IPscatt as well as a technical description. Further, we presented the key features of IPscatt in a set of hands-on guides with step-by-step instructions. Finally, we demonstrated the suitability of IPscatt for applications on some examples.

Appendix

For the reader's convenience this appendix contains introductions to the direct scattering problem and the implemented variational reconstruction scheme to repeat formulas from the *algorithm paper* if we refer to them. For technical details we refer to [BKL17]. Furthermore, scattering simulation is one of the main components of IPscatt. Therefore we summarize the essential formulas of the scattering simulation in Tab. 2 in continuous and discretized form as well as source code to build a connection between mathematical formulas and their implementation.

Introduction to the Direct Scattering Problem For a wave number $k > 0$ the incident field u^{i} solves the Helmholtz equation

$$\Delta u^{\mathrm{i}} + k^2 u^{\mathrm{i}} = 0. \tag{1}$$

The contrast is denoted by q. Then the total field u^{t} solves the Helmholtz equation

$$\Delta u^{\mathrm{t}} + k^2(1+q)u^{\mathrm{t}} = 0 \quad \text{in } \mathbb{R}^d \tag{2}$$

with $d = 2$ or 3. The scattered field u^{s} is measured by means of the total field u^{t} via $u^{\mathrm{s}} = u^{\mathrm{t}} - u^{\mathrm{i}}$. The direct scattering problem (forward problem) is to find such a scattered field u^{s} to a given incident field and contrast.

For the consideration of the involved operators two domains are interesting: the computational domain (CD), which is the square/cube around the circle/ball with radius $2R$, i.e. $D_{2R} = [-2R, 2R)^d$, and the region of interest (ROI), which is the square/cube inside the small circle/ball with radius R, i.e. $D = (-R/\sqrt{2}, R/\sqrt{2})^d$.

The solution of the direct scattering problem is interesting in the latter domain, i.e. u^{s} in D. Usually, this problem is solved by a reformulation: Find u^{s} solving the so-called Lippmann-Schwinger integral equation

$$u^{\mathrm{s}} - V(q \cdot u^{\mathrm{s}}) = V(q \cdot u^{\mathrm{i}}) \quad \text{for } x \in D, \tag{3}$$

where the radiating volume potential is defined by

$$(Vf)(x) := k^2 \int_D \Phi(x-y) f(y) \; \mathrm{d}y, \quad x \in \mathbb{R}^d,$$

where Φ is the radiating fundamental solution of the Helmholtz equation,

$$\Phi(x) = \frac{\mathrm{i}}{4} H_0^{(1)}(k|x|) \quad \text{if } x \in \mathbb{R}^2 \setminus \{0\}, \quad \Phi(x) = \frac{1}{4\pi}\frac{\mathrm{e}^{\mathrm{i}k|x|}}{|x|} \quad \text{if } x \in \mathbb{R}^3 \setminus \{0\}, \tag{4}$$

where $H_0^{(1)}$ is the Hankel function of the first kind and order zero, see [AS65, Ch. 9]. (Fast numerical solvers for the Lippmann-Schwinger integral equation are described for example in [Vai00].)

The direct scattering problem is solved by the forward operator $\mathcal{F}$. To be more precise, the forward operator $\mathcal{F}$ is the multi-static contrast-to-measurement operator, i.e. results in the scattered field at the receivers' positions for each of the incident fields.

Introduction to the Variational Reconstruction Scheme The inverse scattering problem is to find a contrast q such that $\mathcal{F}(q)$ matches the data F^{δ}_{meas} with noise level δ. The data can be synthetic data generated by adding noise to $\mathcal{F}(q)$ or real-world data.

The underlying variational reconstruction scheme of IPscatt to solve this problem relies on the so-called primal-dual algorithm due to Pock, Bischof, Cremers and Chambolle, see [PCBC09, CP11]. To apply the primal-dual algorithm we linearize the forward operator $\mathcal{F}$. Instead of the discrepancy $\|\mathcal{F}(q) - F^{\delta}_{\text{meas}}\|_{\text{F}}$ with the Frobenius norm $\|\cdot\|_{\text{F}}$ we consider $\|\mathcal{F}'(q)[h] + \mathcal{F}(q) - F^{\delta}_{\text{meas}}\|_{\text{F}}$ with the Fréchet derivative $\mathcal{F}'(q)$. The functional to be minimized is in a simplified formulation

$$\min_{h\in\mathbb{C}^{N_{\text{D}}}} \underbrace{\frac{1}{2}\|\mathcal{F}'(q)[h] + \mathcal{F}(q) - F^{\delta}_{\text{meas}}\|^2_{\text{F}}}_{=:f_{\text{dis}}(h)} + \underbrace{\alpha\|q+h\|_1}_{=:f_{\text{spa}}(h)} + \underbrace{\beta\|\nabla(q+h)\|_1}_{=:f_{\text{tv}}(h)} + \underbrace{\delta_{[a,b]}(\text{Re}(q+h)) + \delta_{[c,d]}(\text{Im}(q+h))}_{=:f_{\text{phy}}(h)}. \tag{5}$$

The summands are the discrepancy of the linearized problem $f_{\text{dis}}(h)$, the sparsity penalty $f_{\text{spa}}(h)$, the total variation penalty $f_{\text{tv}}(h)$ and the physical bounds $f_{\text{phy}}(h)$.

Finally, the minimization scheme is to minimize (5) for a fixed contrast q with the primal-dual algorithm, that we call *inner iteration.* Afterwards, the *outer iteration* is to update the contrast $q := q + h$ before we minimize (5) again for the just redefined contrast q. Inner and outer iterations are repeated until the outer iteration is stopped by Morozov's discrepancy principle, i.e. if

$$\|\mathcal{F}(q) - F^{\delta}_{\text{meas}}\|_{\text{F}}/\|F^{\delta}_{\text{meas}}\|_{\text{F}} \leq \tau\delta \quad \text{with tolerance parameter } \tau > 1. \tag{6}$$

Scattering Simulation Scattering simulation is one of the main components of IPscatt. Hence, we summarize the basic formulas of the direct scattering problem in Tab. 2 essentially in the chronological order of scattering. Without the claim of completeness we give them in continuous and discretized form as well as source code to build a connection between mathematical formulas and their implementation.

Single-layer potential for source points			
$\boxed{\text{C}}$	$\text{SL}_{\Gamma_{\text{i}}\to D}\colon L^2(\Gamma_{\text{i}}) \to L^2(D)$,	$(\text{SL}_{\Gamma_{\text{i}}\to D}\, g)(x) := \int_{\Gamma_{\text{i}}} \Phi(x-y)g(y)\,\mathrm{d}s(y), \quad x \in D \setminus \Gamma_{\text{i}}.$	(7)
$\boxed{\text{D}}$	$\text{SL}_{N_{\text{i}},N_D}\colon \mathbb{C}^{N_{\text{i}}} \to \mathbb{C}^{N_{\text{D}}}$,	$(\text{SL}_{N_{\text{i}},N_D}\,\underline{g}_{N_{\text{i}}})(\ell) := \sum_{j=1}^{N_{\text{i}}} \omega^{\text{i}}_j u^{\text{i}}_j(x_\ell)\underline{g}_{N_{\text{i}}}(j)$ with $u^{\text{i}}_j(x) = \Phi(x-p_j)$, $j = 1,\dots,N_{\text{i}}$, in the case of source points at p_j and approximations ω^{i}_j = seti.dSInc of the infinitesimal element on Γ_{i}.	
$\boxed{\text{S}}$	SL: incPnts $\to$ ROI,	SL = dSInc(j)*incField(:,j) with dSInc(j) $= \omega^{\text{i}}_j$ and incField $= u^{\text{i}}_j(x)\underline{g}_{N_{\text{i}}}$ in mimo.m.	
$\boxed{\text{C}}$	$\text{SL}_{\mathbb{S}\to D}$	The single-layer potential in the case of plane waves.	

Volume potential operator[‖]			
$\boxed{\mathsf{C}}$	$V_{2R}\colon L^2(D_{2R}) \to L^2(D_{2R})$,	$(V_{2R}f)(x) := \int_{D_{2R}} \Phi_{2R}(x-y)f(y)\,\mathrm{d}y, \quad x \in D_{2R}.$	(8)
$\boxed{\mathsf{D}}$	$V_{N_\mathrm{D}}\colon \mathbb{C}^{N_\mathrm{D}} \to \mathbb{C}^{N_\mathrm{D}}$,	$V_{N_\mathrm{D}}\underline{f}_{N_\mathrm{D}} := \mathcal{R}_N \mathrm{FFT}_N^{-1}(\widehat{\Phi}_N \odot)\mathrm{FFT}_N \mathcal{E}_N \underline{f}_{N_\mathrm{D}}$. It is restricted to ROI and shifted because of the FFT.	
$\boxed{\mathsf{S}}$	V: ROI $\to$ ROI,	V = seti.k^2.*helmholtz2Dr2r with helmholtz2Dr2r computing restrictCDtoROI(ifft2(reshape(seti.kHat,seti.nCD,seti.nCD).*... fft2(extendROItoCD(fROI,seti.ROImask))),seti.ROImask).	(9)
Additional formulas for the volume potential operator			
	$\mathcal{E}\colon L^2(D) \to L^2(D_{2R})$, $\mathcal{R}\colon L^2(D_{2R}) \to L^2(D)$	$\boxed{\mathsf{C}}$: $\mathcal{E}$ extends, $\mathcal{R}$ restricts and $\mathcal{E}^* = \mathcal{R}$.	
		$\boxed{\mathsf{D}}$: $\mathcal{E}_N\colon \mathbb{C}^{N_\mathrm{D}} \to \mathbb{C}^d_N$ and $\mathcal{R}_N\colon \mathbb{C}^d_N \to \mathbb{C}^{N_\mathrm{D}}$.	
		$\boxed{\mathsf{S}}$: extendROItoCD: ROI $\to$ CD, restrictCDtoROI: CD $\to$ ROI.	
	FFT_N, FFT_N^{-1}	$\boxed{\mathsf{D}}$ and $\boxed{\mathsf{S}}$: $\mathrm{FFT}_N =$ fft2, $\mathrm{FFT}_N^{-1} =$ ifft2.	
	Φ_{2R}, $\widehat{\Phi}_N$, seti.kHat	$\boxed{\mathsf{D}}$ and $\boxed{\mathsf{S}}$: The kernel $\Phi_{2R}(x)$ in the computational domain is $k^2\Phi(x)$ if $x \in \overline{B_{2R}}$ and 0 if $x \in \overline{D_{2R}} \setminus \overline{B_{2R}}$. The shifted Fourier coefficients $\widehat{\Phi}_N = S_N^{-1}([(4R)^{d/2}\,\hat{\Phi}_{2R}(j)]_{j\in\mathbb{Z}^d_N})$ of Φ_N, where S_N^{-1} represents the shifting, are defined as seti.kHat in setKernel.m.	(10)
Application of the volume potential operator V_{2R} computing the scattered field u^s			
$\boxed{\mathsf{S}}$	One incident field: uScattROI: ROI $\to$ ROI	The routine solveLippmannSchwinger(Vq, f, seti) solves (3) computing v such that v $-$ Vq(v) = f. Therefore a GMRES is used. For contrast qROI and incident field uIncROI, the scattered field is uScattROI = solveLippmannSchwinger(@(x) V(qROI.*x), V(qROI.*uIncROI), seti).	(11)
$\boxed{\mathsf{S}}$	Multi-static: FFqROI	FFqROI(:,j) = uScattROI for each transmitter $j = 1, \dots, N_\mathrm{i}$ in mimo.m.	(12)
Adjoint of the volume potential operator			
$\boxed{\mathsf{C}}$	$V_{2R}^*\colon L^2(D_{2R}) \to L^2(D_{2R})$,	$(V_{2R}f)^*(x) := \int_{D_{2R}} \overline{\Phi_{2R}(x-y)}f(y)\,\mathrm{d}y, \quad x \in D_{2R}.$	
$\boxed{\mathsf{D}}$	$V_{N_\mathrm{D}}^*\colon \mathbb{C}^{N_\mathrm{D}} \to \mathbb{C}^{N_\mathrm{D}}$,	$V_{N_\mathrm{D}}^*\underline{f}_{N_\mathrm{D}} := \mathcal{R}_N \mathrm{FFT}_N^{-1}(\overline{\widehat{\Phi}_N} \odot)\mathrm{FFT}_N \mathcal{E}_N \underline{f}_{N_\mathrm{D}}$.	
$\boxed{\mathsf{S}}$	VStar: ROI $\to$ ROI,	VStar = seti.k^2.*helmholtz2Dr2rAdjoint, cf. (9) using seti.kHat'.	
Solution-to-data operator (for near field data)			
$\boxed{\mathsf{C}}$	$V_{D\to\Gamma_\mathrm{s}}\colon L^2(D) \to L^2(\Gamma_\mathrm{s})$	$(V_{D\to\Gamma_\mathrm{s}}f)(x) := k^2 \int_D \Phi(x-y)f(y)\,\mathrm{d}y, \quad x \in \Gamma_\mathrm{s}.$	(13)
$\boxed{\mathsf{D}}$	$V_{N_\mathrm{D},N_\mathrm{s}}\colon \mathbb{C}^{N_\mathrm{D}} \to \mathbb{C}^{N_\mathrm{s}}$	$(V_{N_\mathrm{D},N_\mathrm{s}}\underline{f}_{N_\mathrm{D}})(\ell) := h_N^d k^2 \sum_{j\in\mathbb{Z}^d_N} \Phi(x_\ell - y_j^{(N)})\underline{f}_{N_\mathrm{D}}(j)$ with points $y_j^{(N)}$ in ROI, the position x_ℓ of the ℓth receiver and the area/volume h_N^d of the ROI's infinitesimal element (pixel/voxel).	(14)
$\boxed{\mathsf{S}}$	uScattRX: ROI $\to$ measPnts	uScattRX = seti.k^2.*helmholtz2Dr2data(fROI, seti) in simo.m, where helmholtz2Dr2data yields (seti.measKer*fROI)*seti.dV with seti.dV = h_N^d, fROI = $\underline{f}_{N_\mathrm{D}}$ and seti.measKer = $\Phi(x_\ell - y_j^{(N)})$ for near field data defined in setMeasKer.m.	(15)

[‖]More precisely, we consider the *periodized* volume potential operator in this table.

Adjoint of the solution-to-data operator (for near field data)			
$\boxed{\text{C}}$	$V^*_{D\to\Gamma_\mathrm{s}}\colon L^2(\Gamma_\mathrm{s})\to L^2(D)$	$(V^*_{D\to\Gamma_\mathrm{s}} f)(x) := k^2 \int_{\Gamma_\mathrm{s}} \overline{\Phi(x-y)} f(y)\,\mathrm{d}y, \quad x\in D.$	
$\boxed{\text{D}}$	$V^*_{N_\mathrm{D},N_\mathrm{s}}\colon \mathbb{C}^{N_\mathrm{s}}\to\mathbb{C}^{N_\mathrm{D}}$,	$(V^*_{N_\mathrm{D},N_\mathrm{s}} \underline{f}_{N_\mathrm{s}})(j) := \omega^\mathrm{s}_j\, k^2 \sum_{\ell=1,\dots,N_\mathrm{s}} \overline{\Phi(x_\ell - y_j^{(N)})}\, \underline{f}_{N_\mathrm{s}}(\ell)$. ω^s_j = seti.dSMeas: approximations of infinitesimal element of Γ_s.	(16)
$\boxed{\text{S}}$	VGStar: measPnts $\to$ ROI,	VGStar = seti.k^2.*helmholtz2Dr2dataAdjoint under the use of (seti.measKer')*(f.*seti.dSMeas) in helmholtz2Dr2dataAdjoint.m.	
Solution-to-data operator (for far field data)			
$\boxed{\text{C}}$ $\boxed{\text{S}}$	$V_{D\to\mathbb{S}}$	Operator as in (13), (15), but `seti.measKer` $= \gamma \exp(-\mathrm{i}k\langle y,\theta_\ell\rangle)$, where y are points in ROI, θ_ℓ is the direction of the ℓth receiver and $\gamma = \exp(\mathrm{i}\pi/4)/\sqrt{8\pi k}$ if $x\in\mathbb{R}^2$ and $\gamma = 1/(4\pi)$ if $x\in\mathbb{R}^3$.	(17)
Lippmann-Schwinger solution operator			
$\boxed{\text{C}}$	$T_q\colon L^2(D)\to L^2(D)$	$T_q := (I - V_{2R}(q\,\cdot))^{-1}$	(18)
$\boxed{\text{D}}$	$T_{\underline{q}}\colon \mathbb{C}^{N_\mathrm{D}}\to\mathbb{C}^{N_\mathrm{D}}$	$T_{\underline{q}} := (I - V_{N_\mathrm{D}}(\underline{q}\odot\,))^{-1}$ with $\underline{q}\in\mathbb{C}^{N_\mathrm{D}}$.	
$\boxed{\text{C}}$	$T^*_q\colon L^2(D)\to L^2(D)$	$T^*_q := (I - (V_{2R}(q\,\cdot))^*)^{-1}$	
$\boxed{\text{D}}$	$T^*_{\underline{q}}\colon \mathbb{C}^{N_\mathrm{D}}\to\mathbb{C}^{N_\mathrm{D}}$	$T^*_{\underline{q}} := (I - (V_{N_\mathrm{D}}(\underline{q}\odot\,))^*)^{-1}$ with $\underline{q}\in\mathbb{C}^{N_\mathrm{D}}$	
$\boxed{\text{S}}$	TStar: ROI $\to$ ROI	TStar = @(f) solveLippmannSchwinger(VqStar,f,seti) under the use of VqStar = @(x) conj(qROI).*VStar(x). Remember that solveLippmannSchwinger computes v such that v − VqStar(v) = f.	
Forward operator (multi-static contrast-to-measurement operator)			
$\boxed{\text{C}}$	$\mathcal{F}\colon L^p_{\mathrm{Im}\geq 0}(D)\to \mathrm{HS}$	$\mathcal{F}(q) := V_{D\to\Gamma_\mathrm{s}}(q\cdot) T_q\, \mathrm{SL}_{\Gamma_\mathrm{i}\to D}$. Note that HS is the space of all Hilbert-Schmidt operators $\mathrm{HS}(L^2(\Gamma_\mathrm{i}), L^2(\Gamma_\mathrm{s}))$.	(19)
$\boxed{\text{D}}$	$\underline{\mathcal{F}}\colon \mathbb{C}^{N_\mathrm{D}}\to\mathbb{C}^{N_\mathrm{s}\times N_\mathrm{i}}$	$\underline{\mathcal{F}}(\underline{q}) := V_{N_\mathrm{D},N_\mathrm{s}}(\underline{q}\odot\,) T_{\underline{q}}\, \mathrm{SL}_{N_\mathrm{i},N_\mathrm{D}}$.	
	One incident field:		
$\boxed{\text{S}}$	uScattRX: ROI $\to \mathbb{C}^{N_\mathrm{s}}$	Compute [uScattRX, uScattROI] = S(SL) in mimo.m, where S = @(s) simo(qROI, s, seti) in intOpsFuncs.m. The task of simo.m is to compute u^s = uScattROI as in (11), fROI = QU(uIncROI+uScattROI) $= q\odot(u^\mathrm{i} + u^\mathrm{s})$ and uScattRX as in (15).	(20)
	Multi-static:		
$\boxed{\text{S}}$	FFq: ROI $\to \mathbb{C}^{N_\mathrm{s}\times N_\mathrm{i}}$	FFq(:,j) = uScattRX for each transmitter $j = 1,\dots,N_\mathrm{i}$ in mimo.m.	(21)
Fréchet derivative of the forward operator			
$\boxed{\text{C}}$	$\mathcal{F}'(q)\colon L^2(D)\to\mathrm{HS}$, (cf. (19) for HS)	$\mathcal{F}'(q)[h]g = V_{D\to\Gamma_\mathrm{s}}(I + (q\cdot)T_q V_{2R})(h\cdot) T_q\, \mathrm{SL}_{\Gamma_\mathrm{i}\to D}\, g, \quad g\in L^2(\Gamma_\mathrm{i})$. In finite-dimensional spaces the derivative $\mathcal{F}'(q)$ is represented by the Jacobian matrix denoted by $\underline{\mathcal{F}}'(\underline{q})$, see [BKL17, Sec. 3.6]:	(22)
$\boxed{\text{D}}$	$\underline{\mathcal{F}}'(\underline{q})\colon \mathbb{C}^{N_\mathrm{D}}\to\mathbb{C}^{N_\mathrm{s}\times N_\mathrm{i}}$	$\underline{\mathcal{F}}'(\underline{q})[\underline{h}] = A_{N_\mathrm{D},N_\mathrm{s}}(\underline{h}\odot\,)B_{N_\mathrm{D},N_\mathrm{i}}$ for $\underline{h}\in\mathbb{C}^{N_\mathrm{D}}$ with $A_{N_\mathrm{D},N_\mathrm{s}} = V_{N_\mathrm{D},N_\mathrm{s}}\big(I + (\underline{q}\odot\,)T_{\underline{q}}V_{N_\mathrm{D}}\big) \in \mathbb{C}^{N_\mathrm{s}\times N_\mathrm{D}}$, $B_{N_\mathrm{D},N_\mathrm{i}} = T_{\underline{q}}\,\mathrm{SL}_{N_\mathrm{i},N_\mathrm{D}} \in \mathbb{C}^{N_\mathrm{D}\times N_\mathrm{i}}$.	(23)
$\boxed{\text{S}}$	DFFq	[JA,JB] = derivative(seti,qROI) with JA = $A_{N_\mathrm{D},N_\mathrm{s}}$ and JB = $B_{N_\mathrm{D},N_\mathrm{i}}$ as well as qROI = $\underline{q}$, such that DFFq = @(h) JA*diag(h)*JB computes $\underline{\mathcal{F}}'(\underline{q})[\underline{h}]$ by DFFqh = DFFq(h).	

Fréchet derivative's adjoint of the forward operator

$\boxed{\text{D}}$	$[\underline{\mathcal{F}}'(\underline{q})]^*: \mathbb{C}^{N_\text{s}\times N_\text{i}} \to \mathbb{C}^{N_\text{D}}$	$$[\underline{\mathcal{F}}'(\underline{q})]^*\underline{H} = \sum_{j=1}^{N_\text{s}}\sum_{\ell=1}^{N_\text{i}} \underline{H}_{j,\ell}\,\overline{A_{N_\text{D},N_\text{s}}(j,\cdot)}\,\overline{B_{N_\text{D},N_\text{i}}(\cdot,\ell)} \quad \text{for } \underline{H} \in \mathbb{C}^{N_\text{s}\times N_\text{i}}. \qquad (24)$$ Since discretizations of the domain space $L^p_{\text{Im}\geq 0}$ and the inner product in HS, see (19), take into account weights for physical reasons, there is an additional factor ω^s_j/h^d_N, see (14) and (16).
$\boxed{\text{S}}$	ADFFq	[ADFFq,seti] = adjOfDer(seti,qROI,FmeasDelta) where ADFFq = $[\underline{\mathcal{F}}'(\underline{q})]^*[\underline{\mathcal{F}}(\underline{q}) - F^\delta_\text{meas}]$ with qROI = $\underline{q}$ and FmeasDelta = F^δ_meas. Note that the adjoint applied to the defect results in the derivative of the least-squares error of the forward operator.

Table 2: The table contains the basic formulas of the direct scattering problem: the single-layer potential $\text{SL}_{\Gamma_\text{i}\to D}$, the volume potential operator V, the solution-to-data operator $V_{D\to\Gamma_\text{s}}$ and the Lippmann-Schwinger solution operator T_q. In addition, for practical usage their adjoints are given as well as the forward operator $\mathcal{F}$. Important ingredients of many reconstruction and optimization schemes are in the table too: the Fréchet derivative $\mathcal{F}'(q)$ and its adjoint $[\mathcal{F}'(q)]^*$.
Without the claim of completeness they are presented in the order of continuous and discretized formulas and the source code highlighted by the symbols $\boxed{\text{C}}$, $\boxed{\text{D}}$ and $\boxed{\text{S}}$. The notation $(f\cdot)$ is used to denote the operator of pointwise multiplication (with a function f). If we want to stress the pointwise multiplication we use the notation $f\cdot g$ in the continuous case and $f\odot g$ in the discretized one. We underline a symbol to emphasize the discretization.

References

[AS65] Milton Abramowitz and Irene A. Stegun. *Handbook of Mathematical Functions*, volume 44 of *Dover books on intermediate and advanced mathematics*. Dover Publications, New York, unabridged and unaltered republication of the 1964 edition, 1965.

[BKL17] Florian Bürgel, Kamil S. Kazimierski, and Armin Lechleiter. A sparsity regularization and total variation based computational framework for the inverse medium problem in scattering. *Journal of Computational Physics*, 339:1–30, 2017. URL: https://doi.org/10.1016/j.jcp.2017.03.011.

[Bor26] Max Born. Quantenmechanik der Stoßvorgänge. *Zeitschrift für Physik*, 38(11):803–827, 1926. URL: https://doi.org/10.1007/BF01397184.

[BS01] Kamal Belkebir and Marc Saillard. Special section: Testing inversion algorithms against experimental data. *Inverse Problems*, 17(6):1565–1571, 2001. URL: https://doi.org/10.1088/0266-5611/17/6/301.

[CK13] David Colton and Rainer Kress. *Inverse Acoustic and Electromagnetic Scattering Theory*. Springer, New York, 2013. URL: https://doi.org/10.1007/978-1-4614-4942-3.

[CP11] Antonin Chambolle and Thomas Pock. A First-Order Primal-Dual Algorithm for Convex Problems with Applications to Imaging. *Journal of Mathematical Imaging and Vision*, 40(1):120–145, 2011. URL: https://doi.org/10.1007/s10851-010-0251-1.

[CS05] Tony F. Chan and Jianhong (Jackie) Shen. *Image Processing and Analysis*. Society for Industrial and Applied Mathematics, Philadelphia, 2005. URL: https://doi.org/10.1137/1.9780898717877.

[Geh13] Matthias Gehre. *Rapid Uncertainty Quantification for Nonlinear Inverse Problems*. PhD thesis, Universität Bremen, 2013. Retrieved from https://nbn-resolving.de/urn:nbn:de:gbv:46-00103519-10.

[Gon10] Álvaro González. Measurement of Areas on a Sphere Using Fibonacci and Latitude–Longitude Lattices. *Mathematical Geosciences*, 42(1):49, 2010. URL: https://doi.org/10.1007/s11004-009-9257-x.

[GSE05] Jean-Michel Geffrin, Pierre Sabouroux, and Christelle Eyraud. Free space experimental scattering database continuation: experimental set-up and measurement precision. *Inverse Problems*, 21(6):S117, 2005. URL: https://doi.org/10.1088/0266-5611/21/6/S09.

[KG08] Andreas Kirsch and Natalia Grinberg. *The Factorization Method for Inverse Problems*, volume 36 of *Oxford Lecture Series in Mathematics and its Applications*. Oxford University Press, 2008. URL: https://doi.org/10.1093/acprof:oso/9780199213535.001.0001.

[KS05] Jari Kaipio and Erkki Somersalo. *Statistical and Computational Inverse Problems*, volume 160 of *Applied Mathematical Sciences*. Springer, New York, 2005. URL: https://doi.org/10.1007/b138659.

[MS12] Jennifer L. Mueller and Samuli Siltanen. *Linear and Nonlinear Inverse Problems with Practical Applications*. Computational Science & Engineering. SIAM, Philadelphia, 2012. URL: https://doi.org/10.1137/1.9781611972344.

[PCBC09] Thomas Pock, Daniel Cremers, Horst Bischof, and Antonin Chambolle. An algorithm for minimizing the Mumford-Shah functional. In *2009 IEEE 12th International Conference on Computer Vision*, pages 1133–1140, 2009. URL: https://doi.org/10.1109/ICCV.2009.5459348.

[Rie01] Andreas Rieder. On convergence rates of inexact newton regularizations. *Numerische Mathematik*, 88(2):347–365, 2001. URL: https://doi.org/10.1007/PL00005448.

[Vai00] Gennadi Vainikko. Fast Solvers of the Lippmann-Schwinger Equation. In Robert P. Gilbert, Joji Kajiwara, and Yongzhi S. Xu, editors, *Direct and Inverse Problems of Mathematical Physics*, pages 423–440. Springer, Boston, 2000. URL: https://doi.org/10.1007/978-1-4757-3214-6_25.