EDtoolbox manual, v 0.1

Peter Svensson, NTNU

January 23, 2018

1 Introduction

2 How to compute scattering with the EDtoolbox

There is one main program/function in EDtoolbox v 0.1: `EDmain_convexESIE`. It computes the scattered sound pressure, for a set of frequencies, according to this model:

$$\begin{aligned} p_{\mathrm{total}} &= p_{\mathrm{direct}} + p_{\mathrm{GA}} + p_{\mathrm{diff\ 1}} + p_{\mathrm{diff\ 2}} + p_{\mathrm{diff\ 3}} + \cdots \\ &= p_{\mathrm{direct}} + p_{\mathrm{GA}} + p_{\mathrm{diff\ 1}} + p_{\mathrm{HOD}} \end{aligned} \tag{1}$$

where HOD stands for higher-order diffraction. These four terms are stored in output files under the names `tfdirect`, `tfgeom`, `tfdiff`, `tfinteqdiff`. The program is run by assigning values to six input structs, `geofiledata, Sindata, Rindata, envdata, controlparameters, filehandlingparameters`, see Section 4.

It can be noted that EDtoolbox does not give the resulting sound pressure at a receiver; instead it gives the transfer function, in order to be independent of source amplitude/signal. The transfer functions are defined such that a free-field radiation case has the transfer function

$$\mathrm{TF}_{\mathrm{free-field}} = \frac{\mathrm{e}^{-\mathrm{j}kr}}{r}$$

and all other transfer functions are scaled accordingly.

3 Geometry format

The EDtoolbox handles only polyhedra, including polygonally shaped thin discs/plates. In the EDtoolbox, a polyhedron is defined in terms of 'corners' (vertices) and 'planes' (faces/polygons). These can either be specified directly in the input struct geofiledata (fields `.corners` and `.planecorners`), or in a separate file of the .cad-format, which is a format exported by the CATT-Acoustic software. Figure 1 shows a simple example: a cuboid box.

3.1 Corners

The .corners field is straightforward: it is a matrix of size [ncorners,3] where row n contains the x-,y- and z-coordinates of corner number n. If the .cad-file had a non-contiguous numbering of the corners, a renumbering will be done for the EDtoolbox, starting with number 1. For the example in Fig. 1, this matrix would have the first few lines as

```
geofiledata.corners = [-0.2 -0.44 -0.32;...
0.2 -0.44 -0.32; 0.2 0.2 -0.32; ...
```

3.2 Planes

The .planecorners field is a matrix of size [nplanes,nmaxnumberofcornersperplane] where row n gives the corners that define plane n. The corners must be defined in a counter-clockwise order, as seen from the frontal side of the plane. The example in Fig. 1 would have its planes defined as

```
geofiledata.planecorners = [1 4 3 2;5 6 7 8; ...
```

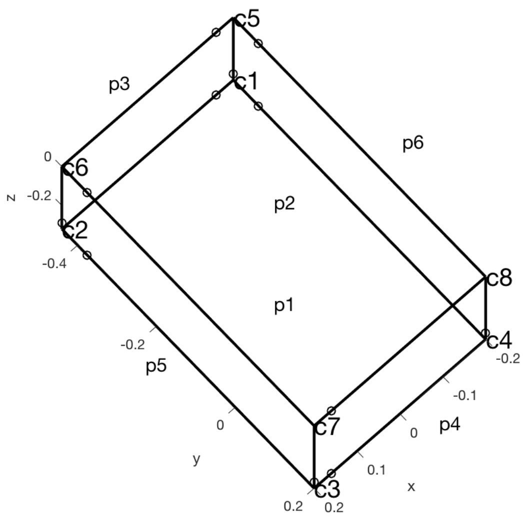

Figure 1: Illustration of a cuboid scattering object. Corner numbers and plane numbers are indicated.

3.3 The `planedata` struct

Based on the definition of corners and planes as described above, the function `EDreadcad` or `EDreadgeomatrices` generates the `planedata` struct, which is passed on internally inside the main function. If the function `EDreadcad` was used to specify the geometry, then it is possible to specify `filehandlingparameters.savecadgeofile = 1`, and this file will contain the `planedata` struct.

Table 1: The `planedata` struct

Field name	Size	Values
`.corners`	[ncorners,3]	Taken from input data
`.planecorners`	[nplanes,nmax][1]	Taken from input data
`.planeabstypes`	sparse([nplanes,nn])	[2]
`.planeeqs`	[nplanes,4]	[3]
`.ncornersperplanevec`	[nplanes,1]	No. of corners per plane
`.minvals`	[nplanes,3]	$[\min(x_i), \min(y_i), \min(z_i)]$[4]
`.maxvals`	[nplanes,3]	$[\max(x_i), \max(y_i), \max(z_i)]$[4]
`.planehasindents`	[nplanes,1]	0 or 1
`.indentingcorners`	[nplanes,nmax]	0 or cornernumber [5]
`.cornerinfrontofplane`	[nplanes,ncorners]	-1, 0 or 1 [6]
`.modeltype`	—	'convex_ext' or 'convex_int' or 'singleplate' or 'thinplates' or 'other'

[1] The value nmax is the maximum number of corners per plane.
[2] The values are either taken from the cad-file (`EDreadcad`) or given the value 'RIGID' for each plane (`EDreadgeomatrices`). The size nn depends on the maximum length of absorber names used in the cad file.
[3] The plane equations are on the form that row n contains the four coefficients $[A, B, C, D]$ on the plane equation form $Ax + By + Cy = D$, where $[A, B, C]$ are normalized to give the plane normal vector.
[4] These min- and max-values give each plane's "axis-aligned bounding box (AABB)"
[5] For each plane this matrix gives the number to the first corner in a corner triplet which identifies an indenting corner. The number of the corner is the order given in planecorners for that plane.
[6] These values specify:
1 means that a corner is in front of the plane
0 means that a corner is aligned with the plane, including belonging to the plane
-1 means that a corner is behind the plane

3.4 Edges - the `edgedata` struct

Using the data in the `planedata` struct, all edges are identified by the function `EDedgeo`, and data is stored in the struct `edgedata`. In addition, some more fields are added to the `planedata` struct. These two structs can be inspected if `filehandlingparameters.saveeddatafile` is set to 1 as input. Fig. 2 shows the example that corresponds to Fig. 1.

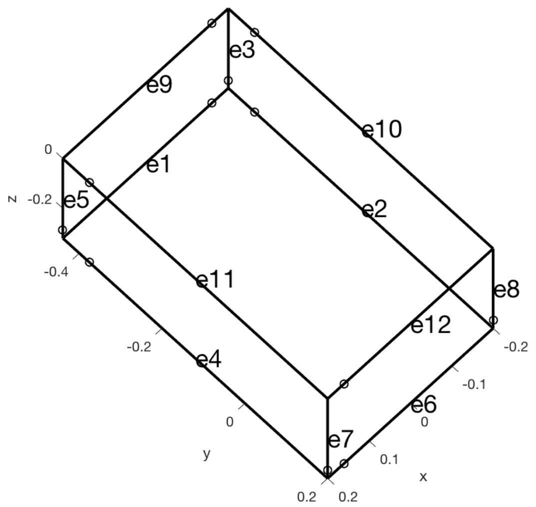

Figure 2: Illustration of a cuboid scattering object, with the derived edge numbers indicated. For each edge, the little circle indicates the starting end.

Table 2: The fields added to the `planedata` struct by `EDedgeo`

Field name	**Size**	**Values**
`.planeisthin`	[nplanes,1]	0 or 1
`.planeseesplane`	[nplanes,nplanes]	-2,-1,0,1 [1]
`.rearsideplane`	[nplanes,1]	0 or planenumber [2]
`.planedata.canplaneobstruct`	[nplanes,1]	0 or 1 [3]
`.reflfactors`	[nplanes,1]	-1 (SOFT), 0 (TOTABS), 1 (RIGID)

[1] These values specify, for each plane-plane pair:
1 means that a plane is in front of the other plane, but obstruction has not been checked
0 means that a plane is completely behind the other plane
-1 means that a plane is aligned with the other plane
-2 means that a (thin) plane is back-to-back with the other (thin) plane
[2] This value is relevant, and non-zero, only for thin planes.
[3] States whether a plane has the potential to obstruct or not (in a plane-to-plane path; which is irrelevant for external convex problems)

Table 3: The `edgedata` struct

Field name	**Size**	**Values**
`.edgecorners`	[nedges,2]	corner numbers [1]
`.closwedangvec`	[nedges,1]	[2]
`.edgestartcoords`	[nedges,3]	[3]
`.edgestartcoords`	[nedges,3]	[3]
`.edgelengthvec`	[nedges,1]	[4]
`.offedges`	[noffedges,1]	[5]
`.edgenvecs`	[nedges,3]	[6]
`.edgenormvecs`	[nedges,3]	[7]
`.edgestartcoordsnudge`	[nedges,3]	[8]
`.edgeendcoordsnudge`	[nedges,3]	[8]
`.edgerelatedcoordsysmatrices`	[nedges,9]	[9]
`.indentingedgepairs`	[nn,2]	[10]
`.planesatedge`	[nedges,2]	The two connected planes for each edge
`.edgesatplane`	[nplanes,nmaxcp][11]	The connected edges for each plane
`.edgeseesplane`	[nplanes,nedges]	-2,-1,0 or 1 [12]

[1] Each edge has a starting corner (column 1) and an ending corner (column 2). This diretion is maintained through all calculations.
[2] For each edge, the "closed wedge angle" is given, in radians. For a 90-degree corner inside a room, the value would be $3\pi/2$. For a 90-degree external corner of a scattering box, the value would be $\pi/2$.
[3] This is redundant; could have been found from `planedata.corners(edgecorners(:,1),:)` (edgestartcoords) or from `planedata.corners(edgecorners(:,2),:)` (edgeendcoords)
[4] This is redundant; could have been computed from the knowledge of the starting and ending coordinates for each edge.
[5] The list gives all the edges that are not active; either because they have an open angle of 90 degrees (or 60 or 45 or ...), or because textttfirstskipcorner has been defined.
[6] Each edge has a reference plane/face, which is defined by a right-hand rule: if the right-hand thumb is aligned with the direction of the edge (as defined by the two corners in `.edgecorners`), then the right-hand fingers will come out of the reference plane, and thereby point in the direction of the normal of the reference plane. This matrix gives those normals, for each edge.
[7] `.edgenormvecs` give the normalized vector along the edge
[8] Same as `.edgestartcoords`, but moved a short distance away from the edge endpoint.
[9] For each edge, the 9 values form a matrix which can be used to compute the coordinates of points in the edge-related coordinate system (which is done a real lot). Each row has to be reshaped: `Bmatrix =reshape(edgerelatedcoordsysmatrices(edgenumber,:),3,3);`
[10] indentingedgepairs
[11] nmaxcp = nmaxcornersperplane = the maximum number of corners for any plane [12] These values specify, for each plane-edge pair:
1 means that the edge has at least some point in front of the plane, but obstruction has not been checked
0 means that the edge is completely behind the plane
-1 means that the edge is aligned with the plane, but does not belong to it
-2 means that the edge belongs to the plane

4 Input data

The main program, EDmain_convexESIE, is run with 6 structs containing all input parameters:

```
EDmain_convexESIE(geofiledata,Sindata,Rindata,envdata,...
controlparameters,filehandlingparameters)
```

Table 4: Input data struct `geofiledata`

Field name	Required?	Default value	Size
`.geoinputfile`	Alt. A (see below)	—	—
`.corners`	Alt. B (see below)	—	[ncorners,3]
`.planecorners`	Alt. B (see below)	—	[nplanes,nmax] [1]
`.firstcornertoskip`	—	1e6 [2]	

Three alternatives exist for the struct `geofiledata`
A. An external `.cad`-file is specified in the field `.geoinputfile`
B. If the field `.geoinputfile` is not specified, then the fields `corners` and `planecorners` can give the geometry data.
C. If neither of the two alternatives above apply (e.g., if the entire struct is left empty), then a file opening window will appear, and a `.cad`-file can be selected. Priority will be given to the `.geoinputfile` if both alternatives A and B are given.
See section 3 for more information on the geometry format.
[1] The value nmax is the maximum number of corners per plane.
[2] The field `.firstcornertoskip` implies that all edges with at least one corner number having the value of `.firstcornertoskip`, or higher, will be deactivated. This gives the possibility to study cases with a subset of all the edges of a model.

Table 5: Input data struct `Sindata`

Field name	Required?	Default value	Size/value
`.coordinates`	Yes	—	[nsources,3]
`.doaddsources`	—	0	0 or 1 [1]
`.sourceamplitudes`	—	ones(nsources,1)	[nsources,1]

[1] If this value is set to 1, the contributions from all sources will be added and saved in a single transfer function, after being multiplied by the values in the vector `.sourceamplitudes`. This is a straightforward way to simulate extended sources, or vibration patterns. See section 5 for a description of the scale values.

Table 6: Input data struct `Rindata`

Field name	Required?	Default value	Size
`.coordinates`	Yes	—	[nreceivers,3]

Table 7: Input data struct `envdata`

Field name	Required?	Default value	Size
`.cair`	—	344	—
`.rhoair`	—	1.21	—

Table 8: Input data struct `controlparameters`

Field name	Required?	Default value	Size, or possible values
`.docalctf`	—	1	0 or 1 [1]
`.docalcir`	Irrelevant [2]	1	0 or 1
`.frequencies`	Yes [3]	—	[1,nfrequencies]
`.fs`	Irrelevant [2]	44100	—
`.directsound`	—	1	0 or 1
`.difforder`	—	15	integer >= 0
`.nedgepoints_visibility`	Irrelevant [4]	2 [5]	—
`.Rstart`	—	0 [6]	—
`.discretizationtype`	—	2	0 or 2 [7]
`.ngauss`	—	16 [8]	even integer >= 2

[1] If the field `.docalctf` is set to 0, edges will be derived and source/receiver visibility will be computed. Please note that to have any use for these calculations, you must specify in the struct `filehandlingparameters` that the proper geometry information is saved.
[2] The sampling frequency, `fs`, and the parameter `.docalcir`, is used in upcoming time-domain calculation functions, but is not read/used by EDmain_convexESIE.
[3] A list of frequencies must be specified for the main function EDmain_convexESIE, and other upcoming frequency-domain versions (unless `.docalctf` is 0). It is not needed for time-domain versions.
[4] This parameter specifies how many points along each edge will be tested for visibility. This is irrelevant for convex scattering bodies since either the whole edge or no part of an edge is visible. It is relevant for upcoming calculation alternatives for non-convex geometries.
[5] The default value of 2 implies that the two end points of each edge will be tested for visbility.
[6] The parameter `.Rstart` determines the phase of the final transfer function (or the definition of time zero in upcoming time-domain calculation alternatives). To simulate an incoming plane wave with amplitude 1, and phase zero, at the origo, then `.Rstart` should be set to the distance to the far-away point source.
[7] The value 0 implies a uniform discretization of the edges. The value 2 gives a Gauss-Legendre discretization. The value 1 is obsolete/not used.
[8] The value `.ngauss` specifies the number of quadrature points along the longest edge. It will be scaled down linearly based on the length of each edge, and an even number of quadrature points will always be chosen.

Table 9: Input data struct `filehandlingparameters`

Field name	Required?	Default value	Possible values
`.outputdirectory`	Yes [1]	Same as geoinputfile [2]	—
`.filestem`	Yes[1]	Name of cad-file	—
`.savesetupfile`	—	1	—
`.savecadgeofile`	—	0	—
`.saveSRdatafiles`	—	0	—
`.saveeddatafile`	—	0	—
`.savesubmatrixdata`	—	0	—
`.saveinteqsousigs`	—	0	—
`.loadinteqsousigs`	—	0	—
`.savepathsfile`	—	0	—
`.saveISEStree`	—	0	—
`.savelogfile`	—	1	—
`.savediff2result`	—	0	—

[1] If the geometry is given in the form of the input fields `.corners` and `.planecorners`, then the fields `.outputdirectory` and `.filestem` must be specified.
[2] Note that a folder called "results" will be generated in the output directory (if it doesn't already exist). All result files will be saved in that results folder.

5 Output data

The main program, EDmain_convexESIE, can generate a lot of output data, as indicated by table 9. There are, however, two files that will always be generated, with the most important output: the TF-terms in Eq. (1). These two files are

`Filestem_tf.mat` (with `tfdirect, tfgeom, tifdiff`)

`Filestem_tfinteq.mat` (with `tfinteqdiff`)

You will have to add these terms yourself:

`tftot = tfdirect + tfgeom + tfdiff + tfinteqdiff .`

6 Program structure for EDmain_convexESIE

The main function EDmain_convexESIE runs through the following blocks, in this given order.

6.1 EDcheckinputstructs

This function checks the input data and assigns default values to input parameters that have not been specified.

6.2 EDreadcad or EDreadgeomatrices

The geometry can either be specified in a separate .cad file, or given as input data matrices. See more on this topic in Section XX. Basically, the geometry is specified as a set of corners (vertices) and planes (faces/polygons). They are stored in a struct called `planedata`.

6.3 EDedgeo

This function identifies all the edges of the polyhedron, and stores data about them in a separate struct called `edgedata`.

6.4 EDSorRgeo

This function is run twice, once to find the visibility data for the source, and the second time to find the visibility data for the receiver. The visibility data tells what edges and planes each source and receiver can see.

6.5 EDfindconvexGApaths

This functions finds all the valid direct sound paths, specular reflection paths, and first-order diffraction paths. This function is specialized on the case of external, convex scattering objects, and for such objects, each source-receiver combination can have maximum one specular reflection. The results are stored in a struct called `firstorderpathdata`.

6.6 EDmakefirstordertfs

Based on the paths specified in the struct `firstorderpathdata`, the function `EDmakefirstordertfs` generates the transfer functions `tfdirect, tfgeom,` and `tfdiff`.

6.7 EDed2geo

This function is run only if difforder > 1. It identifies which edges see which other edges, and stores this information in a struct `edgetoedgedata`. Clearly, this is needed only if the requested diffraction order ¿ 1.

6.8 EDinteg_submatrix

This function is run only if difforder > 1. It identifies the submatrix structure for the subsequent integral equation solving, by the function `EDintegral_convex_tf`.

6.9 EDintegral_convex_tf

This function is run only if difforder > 1. It computes the ackumulation of higher-order diffraction, from order 2 up to a specified diffraction order. The result is stored in the transfer function `tfinteqdiff`.

7 Some examples for EDmain_convexESIE

In the folder `EDexamples`, you will find some examples of different types. Feel free to copy and modify these scripts as you prefer.