MSHDIST: COMPUTATION OF THE SIGNED DISTANCE FUNCTION TO A DISCRETE CONTOUR.

C. DAPOGNY [1] AND P. FREY [2]

[1] *Laboratoire Jean Kuntzmann, CNRS, Université Grenoble-Alpes, BP 53, 38041 Grenoble Cedex 9, France,*
[2] *UPMC Univ Paris 06, UMR 7598, Laboratoire J.-L. Lions, F-75005 Paris, France.*

This note presents the main features of the open-source code `mshdist` for computing the signed distance function d_Ω to a two- or three-dimensional domain Ω, at the vertices of a computational domain D; the function d_Ω is formally defined by

$$d_\Omega(x) = \begin{cases} -d(x,\partial\Omega) & \text{if } x \in \Omega, \\ 0 & \text{if } x \in \partial\Omega, \\ d(x,\partial\Omega) & \text{otherwise} \end{cases}, \text{ where } d(x,\partial\Omega) := \min_{p\in\partial\Omega} |x-p|.$$

In a nutshell, `mshdist` starts by a precise calculation of d_Ω in the region of D "close" to $\partial\Omega$, which is followed by a "propagation" stage, where this "near field" is extended to the whole domain D. `mshdist` can deal with three slightly different situations in 2d or 3d:

(1) Two simplicial meshes are provided: one for D, and the other for Ω;
(2) One simplicial mesh is supplied for D and Ω is given as the negative subdomain of a level set function $\phi : D \to \mathbb{R}$ (supplied at the vertices of the mesh for D);
(3) One simplicial mesh of D is supplied, in which a submesh is a mesh for Ω.

Some of these features are also available when Ω is a subregion of a three-dimensional surface D.

This code and with the examples illustrating this document can be downloaded from the `github` repository

`https://github.com/ISCDtoolbox/Mshdist`

The underlying theory is described in the journal article [1].

Contents

---

1. File structures

The program `mshdist` relies on two types of data files: `.mesh` and `.sol` files:

- As the name suggests, a `.mesh` file contains information about a simplicial mesh (i.e. composed of triangles in 2d, tetrahedra in 3d). This format is used, in particular, by `inria` programs, or the libraries `ISCDToolbox`[1] and `mmg`[2]. A typical `.mesh` file is organized as follows:

```
/* Header */
MeshVersionFormatted 1

Dimension
2

/* List of the vertices of the mesh: two floats in 2d (three in 3d) for the
coordinates, and an integer for a possible reference */
Vertices

3030     // Number of vertices

1 1 2
1 0.975 0
0.975 1 2
0.983333333333 0.966666666154 0
1 0.95 0
...

/* List of the elements of the mesh: three integers in 2d (four in 3d) for the
indices of the vertices, and one additional integer for a possible reference */
Triangles    // Tetrahedra in 3d

5898        // Number of triangles
900 833 899 0
834 828 770 0
769 834 770 0
900 893 834 0
...

/* Ending keyword */
End
```

Listing 1. Organization of a `.mesh` file

- A `.sol` file is attached to a `.mesh` file: `mshdist` uses this format to read and print information about scalar functions defined at the vertices of a corresponding mesh. A typical `.sol` file associated to the `.mesh` file exemplified in Listing 1 is organized as follows:

```
/* Header */
MeshVersionFormatted 1

Dimension
2

/* Number of vertices in the corresponding mesh */
SolAtVertices
3030
```

[1] `https://github.com/ISCDtoolbox`
[2] `http://www.mmgtools.org/`

```
/* 1 = 1 field, 1 = scalar field */
1 1

/* List of values associated to the vertices of the mesh */
0.92393
0.000270181
0.886448
0.000515695
...

/* Ending keyword */
End
```

LISTING 2. Organization of a `.sol` file

Several additional data or options can be supplied via a `DEFAULT.mshdist` file, to be placed in the directory where `mshdist` is called. These options and the syntax to be adopted are detailed in the following sections.

2. First mode: distancing algorithm

The first option of `mshdist` calculates the signed distance function d_Ω to a domain Ω at the vertices of the mesh of a bounding box D. This option thus assumes the datum of:

- A `.mesh` file (`box.mesh` in the present example) for the mesh of a computational domain D where the calculation of d_Ω is intended;
- Another `.mesh` file (here called `contour.mesh`) for the mesh of the domain $\Omega \subset D$; this second file may be either
  - A mesh of the boundary $\partial\Omega$ only (i.e. containing the edges of $\partial\Omega$ in 2d, or its surface triangles in 3d);
  - A mesh of the full domain Ω, in which case `mshdist` only retains the information about the surface part $\partial\Omega$.

The command line for the calculation is:

```
mshdist box.mesh contour.mesh
```

This operation produces a `.sol` file with the same name as the mesh for D but equipped with the `.sol` extension (thus, here `box.sol`). This file contains the desired information about d_Ω at the vertices of the mesh `box.mesh`.

In practice, it often happens that the domain Ω is not a subset of the computational domain D: for instance, one may think that D is a unit square, while the model Ω of interest is expressed in different units. For this reason, unless `mshdist` is explicitly specified not to do so, the contour mesh `contour.mesh` of Ω is automatically *scaled* so that its bounding box is a given percentage `SIZE` of the bounding box of the mesh `box.mesh` . By default, `SIZE` is set to 95%; this value can be modified by using the instruction `-scale` on the command line (see Section 6). This scaling can also be disabled by adding the command `noscale` on the command line; then, the user is responsible for supplying mesh files `box.mesh` and `contour.mesh` with matching sizes.

Last but not least, let us comment shortly about the way `mshdist` sets the sign of the computed distance function d_Ω; again see [1] for the details. In most cases (especially when the scaling feature has not been disabled by the user), the domain Ω is strongly included in D, that is, Ω lies at a distance from ∂D. Then, it is legitimate that `mshdist` endows a positive sign to the values at the vertices inside this connected component of $D \setminus \overline{\Omega}$; this determines the sign of d_Ω at all vertices of D via the coloring algorithm described in [1]. However, it may be desirable for the user to specify different exterior points to the domain (this is typically interesting when $\partial\Omega$ intersects ∂D, when scaling is disabled). The user may specify one, or several points exterior to Ω by using the `DEFAULT.mshdist` file with the syntax exemplified in Listing 3.

```
/* The vertex with coordinates (0.95,0.95) is exterior to $\Omega$ */
ExteriorPoints
1.    // One exterior point is specified
```

```
0.95  0.95
```

LISTING 3. Specification of an exterior point to `mshdist`

Remark 1. If the supplied contour $\partial\Omega$ is not orientable (i.e. it does not delimit an interior and an exterior part), `mshdist` will return an error.

Example 1. We generate the signed distance function to the 2d contour supplied by the mesh `frmap.mesh`, at the vertices of `carre.mesh`: the command

```
mshdist carre.mesh frmap.mesh -ncpu 2
```

yields the result displayed in Fig. 1.

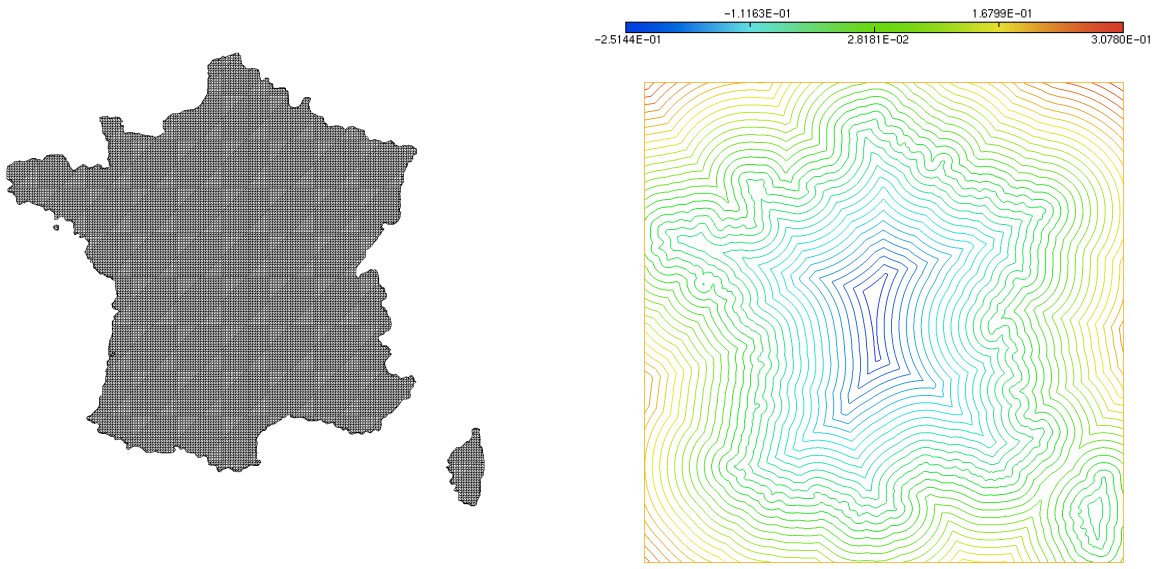

FIGURE 1. *(Right) Isovalues of the signed distance function to the contour supplied in* `frmap.mesh` *(left).*

Example 2. In 3d, we generate the signed distance function to the contour in `2handle.mesh`, at the vertices of `cube.mesh`: the command

```
mshdist cube.mesh 2handle.mesh -ncpu 2
```

yields the result displayed in Fig. 2.

3. SECOND MODE: REDISTANCING ALGORITHM

The second running mode of `mshdist` carries out *redistancing*, an operation of much interest in the context of the level set method (see for instance the monograph [2]). Only one mesh $\mathcal{T}$ is supplied, describing the computational domain D; Ω is known via the datum of a level set function $\phi : D \to \mathbb{R}$, that is:

$$\forall x \in D, \quad \begin{cases} \phi(x) < 0 & \text{if } \ x \in \Omega, \\ \phi(x) = 0 & \text{if } \ x \in \partial\Omega, \\ \phi(x) > 0 & \text{if } \ x \in D \setminus \overline{\Omega}, \end{cases}$$

supplied as a `.sol` file, at the vertices of $\mathcal{T}$.

In practice when the following command line is used

```
mshdist box.mesh
```

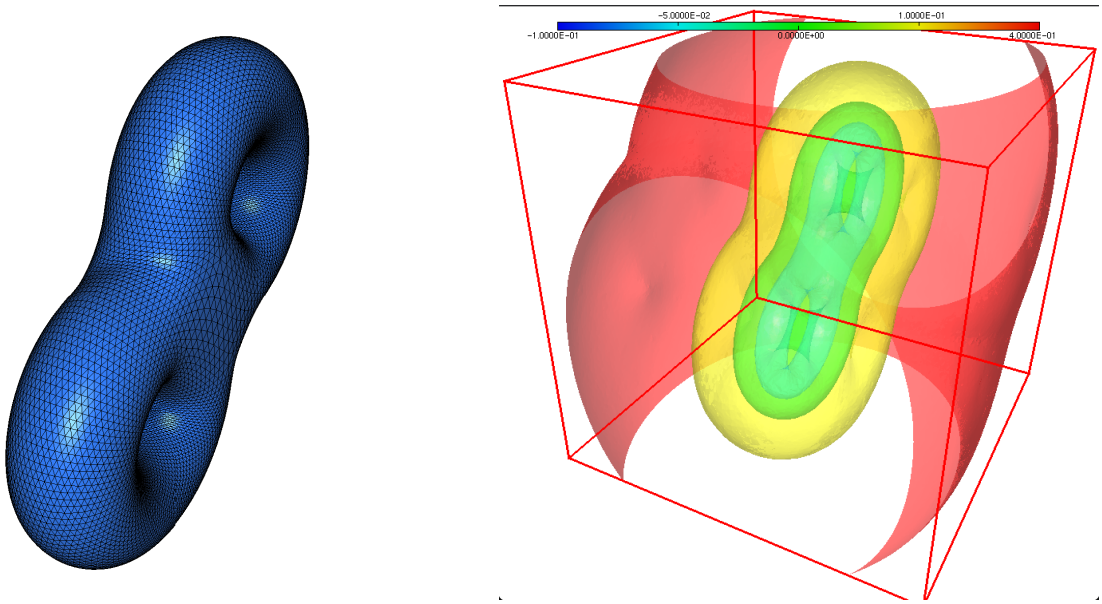

FIGURE 2. *(Right) Isosurfaces of the signed distance function to the contour supplied in* `2handle.mesh` *(left).*

`mshdist` understands that a solution file `box.sol` exists (defined at the vertices of the input mesh of D), which contains the data of a level set function associated to Ω. Then, `mshdist` calculates d_Ω, and prints it in the file `box.sol` (be careful: the original solution file is overwritten).

Example 3. Using the command

```
mshdist bat.mesh -ncpu 2
```

with the files in the Example directory yields the example depicterd in Fig. 3.

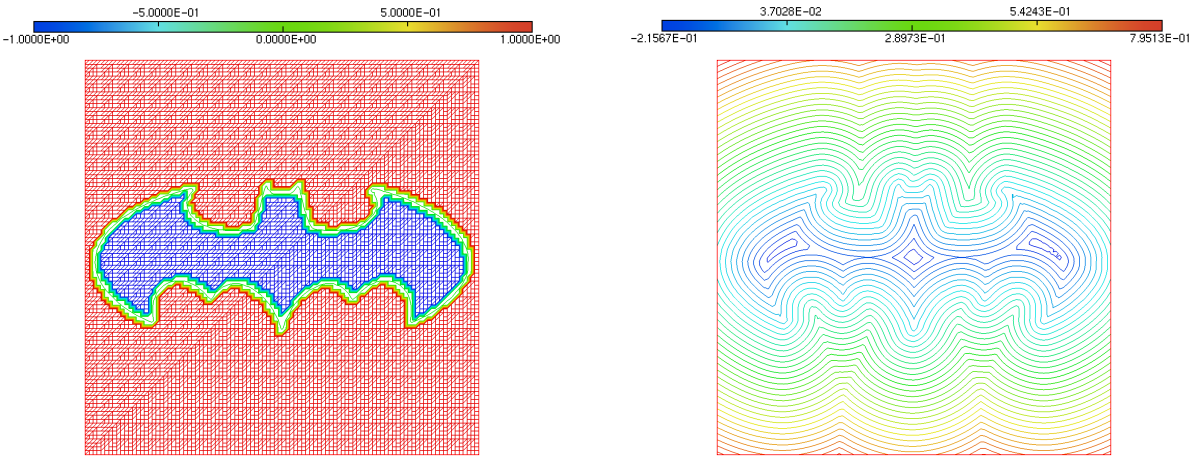

FIGURE 3. *(Left) Isovalues of one (irregular) level set function for a bat-shaped domain* Ω *and (right) isovalues of the signed distance function to* Ω*.*

4. Third mode: generation of the signed distance function to a subdomain enclosed in the supplied mesh

This third option assumes the datum of a single mesh $\mathcal{T}$ of the computational domain D. The latter encloses the domain Ω of interest as a submesh: the elements of Ω are also elements of the larger mesh, and they are identified by their *reference number*, see Fig. 4 (left) for an illustration. By default, they are the elements with label 3.

By using the command line

```
mshdist box.mesh -dom
```

`mshdist` generates a file `box.sol` which contains the signed distance function to Ω.

Note that the arbitrary value 3 for the interior subdomain can be changed in the `DEFAULT.mshdist` file (including the possibility that there may be several interior subdomains, with different references, or that there may be starting edges, vertices...), by using the syntax in Listing 4.

```
/* Keyword and number of interior domains */
InteriorDomains
4

/* References of the interior domains */
3
21
23
25
```

Listing 4. Changing the label of interior domain(s)

Example 4. Using the command

```
mshdist thks.mesh -dom -ncpu 2
```

yields the result depicted in Fig. 4.

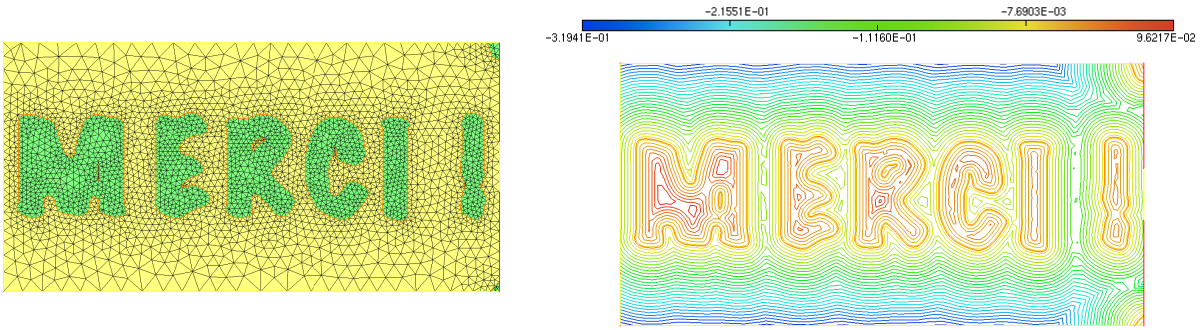

Figure 4. *(Right) Isovalues of the signed distance function to the yellows subdomain in the mesh of the left.*

Example 5. Using the command

```
mshdist chair.mesh -dom -fini
```

yields the result depicted in Fig. 5.

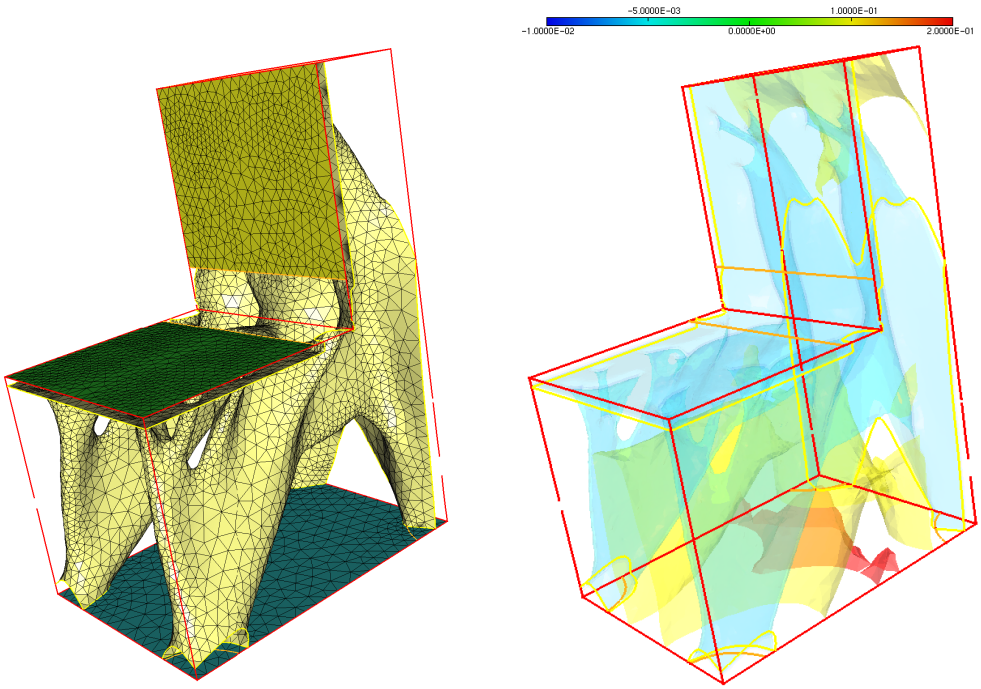

FIGURE 5. *(Right) Isovalues of the signed distance function to the subdomain depicted in the mesh of the left.*

5. Generation of the signed distance function on a surface

The second and third options of `mshdist`, described in Sections 3 and 4 can be carried out when the ambient medium D is a 3d surface, provided the `-surf` option is appended to the command line. For instance, the redistancing instruction

```
mshdist sphere.mesh -surf
```

assumes that the file `sphere.mesh` contains the information about a surface D, and that a `sphere.sol` file exists, containing the values at its vertices of a level set function $\phi : D \to R$ for a region $\Omega \subset D$.

More precisely, this command line can accommodate the following two situations:

(1) `sphere.mesh` is a surface triangulation, and a `sphere.sol` file is present in the same repository, containing the values of ϕ, defined at the vertices of this mesh;
(2) `sphere.mesh` is a full 3d mesh (i.e. containing tetrahedra as well as surface triangles for the boundary) of a domain D, and a `sphere.sol` file is present in the same repository, containing the values of a level set function for a subdomain of the boundary ∂D. Hence, it is understood that the values at the internal vertices have no meaning.

Example 6. Using the command line

```
mshdist sphere.mesh -surf -fmm
```

(where `sphere.mesh` is a full tetrahedral mesh) yields the result depicted on Fig. 6, where the initial level set function has been redistanced on the surface part only.

6. Additional options

- As described in [1], all three options of `mshdist` start with the exact calculation of d_Ω at the vertices of the mesh $\mathcal{T}$ of D that are "near" $\partial\Omega$. Depending on the size of the data, this procedure may prove a little long. A faster, albeit less precise procedure may be used by adding the option

```
-fini
```

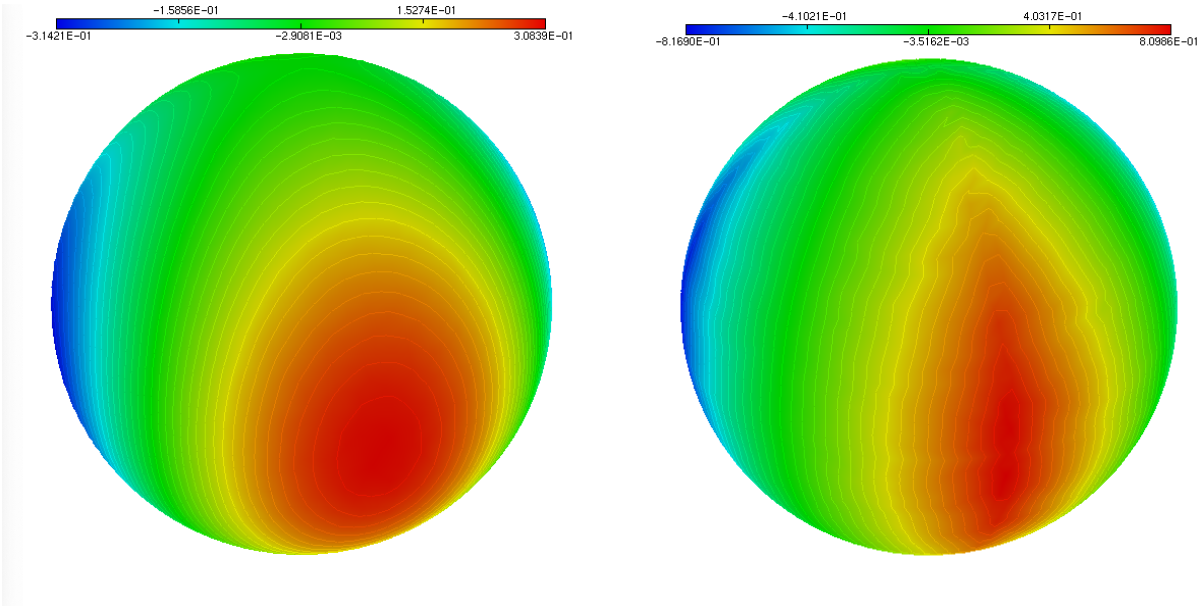

FIGURE 6. *(Left) Isovalues of one level set function for a subdomain of the sphere and (right) isovalues of the signed (geodesic) distance function to this subdomain.*

to the command line.

- Another, less precise, procedure for extending the signed distance function from the datum of the "near field", relying on the celebrated Fast Marching Method [3], has been implemented. The latter can be used by adding the option

  `-fmm`

  to the command line.
- `mshdist` can work in parallel if the command

  `-ncpu` *number*

  is added to the command line (this parallel implementation does not work when the `-fmm` option is used).
- In the distancing mode of Section 2, using

  `-scale` *ratio*

  on the command line imposes the ratio (comprised between 0 and 1) of the scaling of `contour.mesh` inside `box.mesh`. The default value for this ratio is 0.95.
- The `-noscale` command, which is only useful in the distancing mode, has been described above.
- The number of iterations of the process can be controlled by adding

  `-it` *number*

  to the command line.

Acknowledgement. This work was initiated while the first author was partially supported by the Chair "Mathematical modeling and numerical simulation, F-EADS - École Polytechnique - INRIA".

REFERENCES

[1] C. DAPOGNY, P. FREY, *Computation of the signed distance function to a discrete contour on adapted triangulation*, Calcolo, Volume 49, Issue 3, pp. 193-219 (2012).

[2] J.A. SETHIAN, *Level Set Methods and Fast Marching Methods : Evolving Interfaces in Computational Geometry,Fluid Mechanics, Computer Vision, and Materials Science*, Cambridge University Press, (1999).

[3] J.A. SETHIAN, *Fast marching methods.* `SIAM` review, 41(2), (1999), pp. 199–235.