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 short note presents the main features of the code `mshdist` for computing the signed distance function to a discrete contour, associated to the journal article [1].

1. **Files structures.** `Mshdist` reads two types of data files: `.mesh` files (for meshes), and `.sol` files (for scalar fields defined at the vertices of a mesh).

- A `.mesh`file contains all the required information about the associated mesh; it is the standard meshing formate used by INRIA programs. Such a 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
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 contains data supported by an associatedmesh; it is organized as follows:

```
/* Header */
MeshVersionFormatted 1

Dimension
2

/* Number of vertices for supporting solution */
SolAtVertices
3030

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

/* List of solutions associated to the previous mesh */
0.92393
0.000270181
0.886448
0.000515695
...

/* Ending keyword */
End
```

LISTING 2. Organization of a `.sol` file

2. **First mode: distancing algorithm.** The first option of `mshdist` generates the signed distance function d_Ω to a domain Ω supplied by means of a mesh of its boundary $\partial\Omega$ (composed of edges in $2d$, triangles in $3d$), at the vertices of a computational mesh of a bounding box D. The associated line of command is:

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

This operation produces a file `box.sol`, which contains the information about d_Ω at the vertices of the mesh `box.mesh`.

Note that `contour.mesh` could be supplied itself as a volume mesh of the domain Ω (i.e. by means of triangles in $2d$, tetrahedra in $3d$). In this case, `mshdist` will not read the information about the volume part of the mesh, and will only retain information contained in the fields `Edges` (in $2d$) or `Triangles` (in $3d$) in the mesh file `contour.mesh`.

If the supplied contour is not orientable (i.e. it does not define unambiguously an interior and an exterior), the program fails, and an error message is issued.

Unless `mshdist` is explicitly told not to do so, the contour mesh `contour.mesh` is automatically *scaled* so that its bounding box is a given percentage `SIZE` of the bounding box of the mesh `box.mesh` (so as to avoid problems when computational boxes are not expressed in the same units as the models of interest). By default, `SIZE` is set to 95%; this value can be changed in the file `mshdist.h`. 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).

Eventually, recall that, for attributing a sign to the distance function, `mshdist` starts from an exterior triangle (tetrahedron in $3d$) to Ω (typically an element located at a corner of D). If no scale is applied, such an element should be provided by the user (for it may depend on the application !); in this case, the user should specify a point exterior to Ω, by changing the coordinates of `p` on the lines

```
/* identify triangle close to lower corner (boundary) */
p[0] = 0.05;
p[1] = 0.05;
```

LISTING 3. specifying an exterior point (element) to `mshdist`

of the function `sgndist_2d`(resp `_3d`) in files `mshdis1_2d.c`or `mshdis1_3d.c`.

For instance, to generate the signed distance function to the 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 Figure 1.

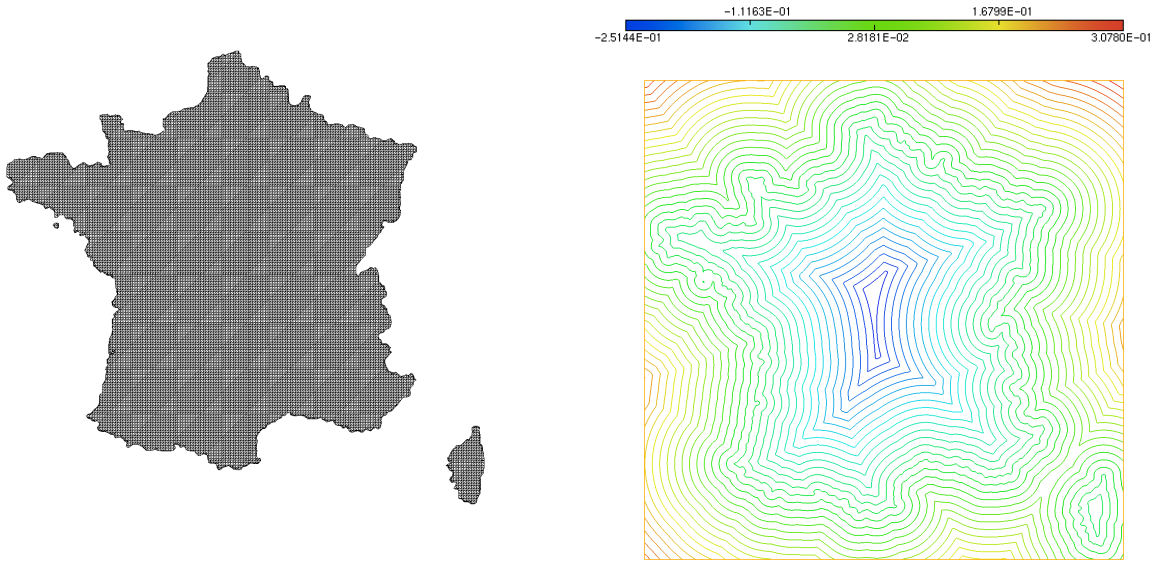

FIGURE 1. *(Right) Isovalues of the signed distance function to the contour supplied in* `frmap.mesh` *(left).*

3. **Second mode: redistancing algorithm.** The second option of `mshdist` concerns redistancing, an operation of great interest in the context of the level set method (see for instance the monograph [2]). By entering the command line

```
mshdist box.mesh
```

`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 a domain $\Omega \subset D$. Then, `mshdist` regenerates the signed distance function to this domain, and prints it in the file `box.sol` (be careful: the original solution file is overwritten).

For instance, using the command

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

with the files in the Example directory yields the example in Figure 2.

4. **Generation of the signed distance function to a subdomain.** This option considers an input mesh `box.mesh`, which encloses a domain Ω as a submesh (i.e. the elements of Ω are also elements of the larger mesh). The elements of Ω are identified by their *reference number*. 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...), which should be located in the directory where `mshdist` is used, and is organized as in Listing 4.

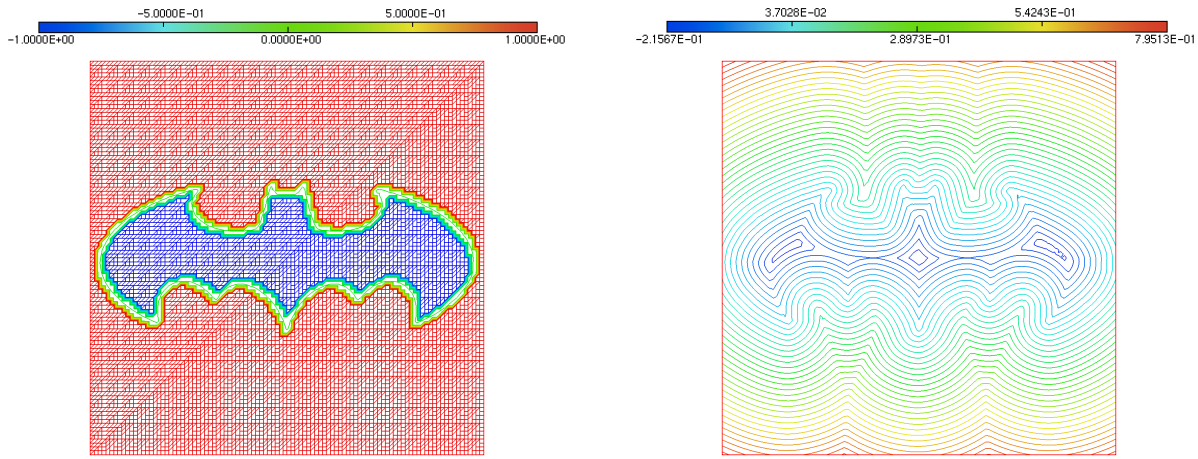

FIGURE 2. *(Left) Isovalues of one (irregular) level set function for some bat-shaped domain* Ω *and (right) isovalues of the signed distance function to* Ω.

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

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

LISTING 4. Organization of a `.sol` file

For example, using the command

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

with the example in the Example directory yields the result in Figure 3.

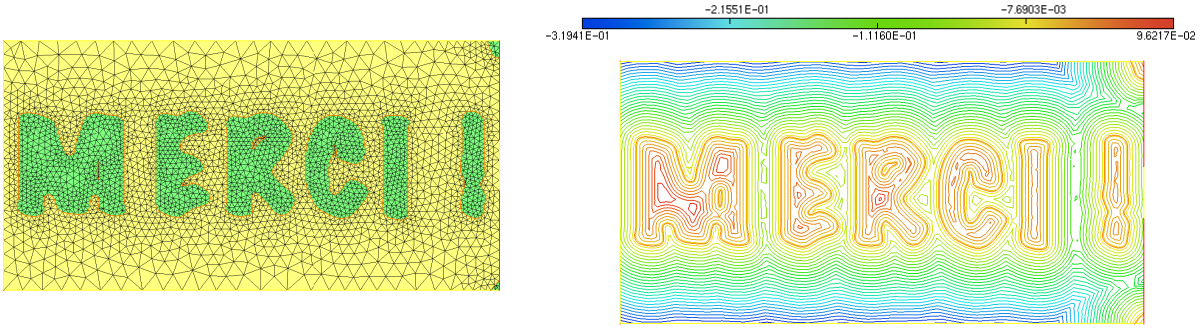

FIGURE 3. *(Right) Isovalues of the signed distance function to the yellows subdomain in the mesh of the left.*

5. **Additional options.**
   - `mshdist` can work in parallel if the command

`-ncpu` *number*

is added to the command line.

- 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 issued 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).