IGDK Munich – Graz

Optimization and Numerical Analysis for Partial Differential Equations with Nonsmooth Structures

The `level-set` Package for GNU Octave

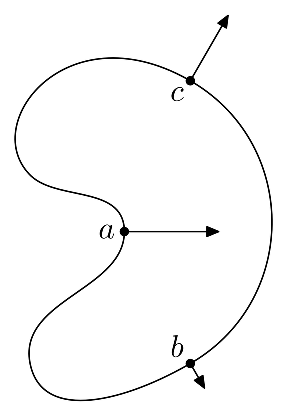
Daniel Kraft

The Level-Set Method for Shape Optimisation

For a level-set function $\phi\colon \mathbb{R}^n \times [0,\infty) \to \mathbb{R}$, we define:

$$\Omega_t = \{x \in \mathbb{R}^n \mid \phi(x,t) < 0\}, \quad \Gamma_t = \{x \in \mathbb{R}^n \mid \phi(x,t) = 0\}$$

Evolution of $\Omega_0 \subset \mathbb{R}^n$ by the speed method:



Propagation in time with the level-set equation:

$$\phi_t + F(x)\,|\nabla\phi| = 0, \quad \phi(\cdot,0) = \phi_0 \qquad (1)$$

$F\colon \mathbb{R}^n \to \mathbb{R}$ is a scalar speed field.
Left: $F(a) < 0 < F(b) < F(c)$

See [3] for a thorough, practical introduction.

Basic Operations with Level-Set Functions

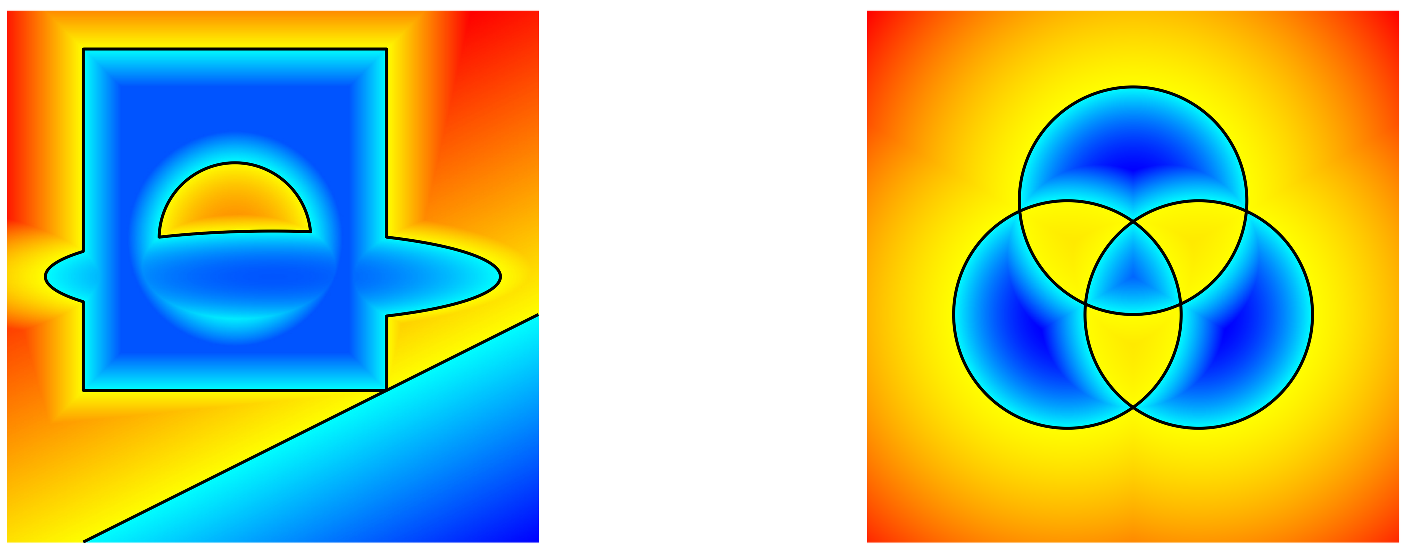
Set predicates:

- `ls_inside`
- `ls_isempty`, `ls_issubset`
- `ls_equal`, `ls_disjoint`

Set operations:

- `ls_complement`
- `ls_union`, `ls_intersect`
- `ls_setdiff`, `ls_setxor`

Basic shapes with `ls_genbasic` and the set operations.



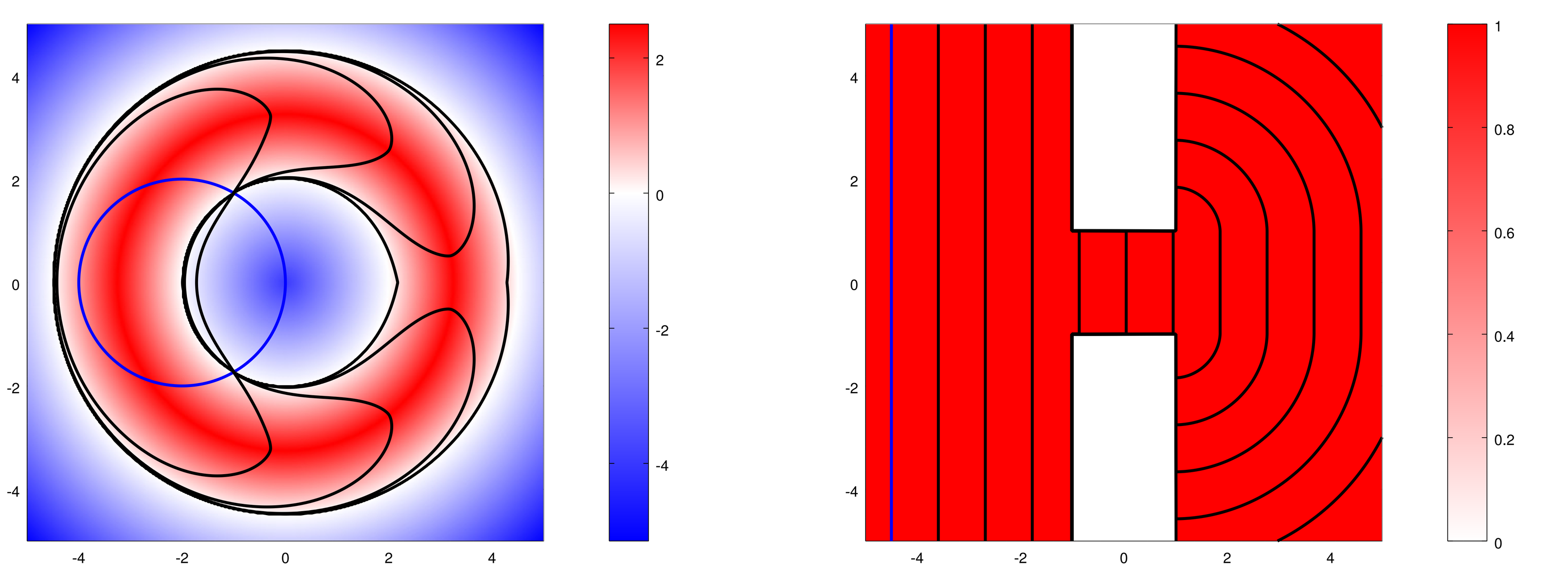
Composite Fast Marching

Applying (3) once for $F \geq 0$ and once for $F \leq 0$, we can evolve shapes for arbitrary speed fields: Composite Fast Marching [2]

This is also beneficial if we need shapes for the same F and different times.

Basic usage outline:

```
nb = ls_nb_from_geom (g, phi0); % optional with struct g
d = ls_solve_stationary (phi0, F, h, nb);
phiT = ls_extract_solution (t, d, phi0, F);
```



Descent Recording and Replay

The framework around `so_run_descent` allows also for logging and replay:

- `so_save_descent` Keep records of all descent iterations.
- `so_replay_descent` Replay steps without recomputation.
- `so_explore_descent` Interactively step through the descent.

Computing the Time-Evolution of Shapes

Solution of (1) with time stepping: `ls_time_step`

Alternative idea based on the Eikonal equation:

$$F(x)\,|\nabla d_0(x)| = 1 \text{ outside of } \Omega_0, \quad d_0 = 0 \text{ on } \Gamma_0 \cup \Omega_0 \qquad (2)$$

This yields an F-induced distance to the initial geometry.

We can then use the Hopf-Lax formulas [1], [2]:

$$\begin{aligned} \phi(x,t) &= \inf\{\phi_0(y) \mid d(x,y) \leq t\} \\ \Omega_t &= \{x \in \mathbb{R}^n \mid d_0(x) < t\} \\ \Gamma_t &= \{x \in \mathbb{R}^n \mid d_0(x) = t\} \end{aligned} \qquad (3)$$

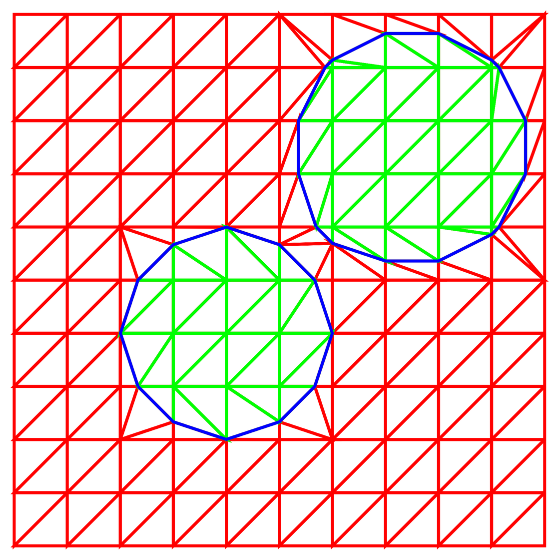
If $F \geq 0$ is not the case, one can split the domain and combine results.

Geometry in 2D

`ls_find_geometry`: Geometric information about Ω and Γ as struct.

`msh`-compatible triangle mesh:

```
phi = ls_normalise (phi, h);
g = ls_find_geometry (phi, h);
g = ls_absolute_geom (g, X, Y);
mesh = ls_build_mesh (g, phi);
```



Distance Functions

Solution of (2) via Sethian's Fast Marching Method [3]: `fastmarching`

For constant speed $F = 1$, this yields distance functions:

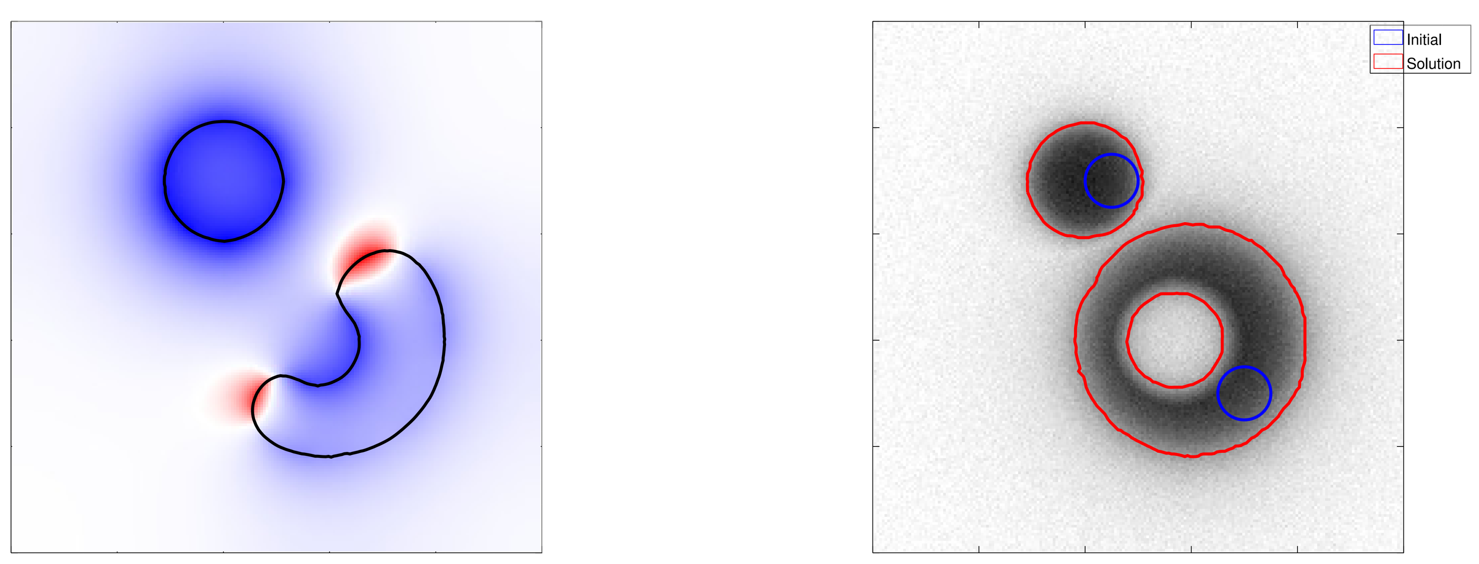
- Distance to Ω: `ls_distance_fcn` and `ls_signed_distance`
- Hausdorff distance of two domains: `ls_hausdorff_dist`

Gradient Descent for Shape Optimisation

General descent algorithm based on level sets:

1. Start with an initial Ω_0 and ϕ_0.
2. Compute shape derivative (usually on the boundary Γ).
3. Extend it to a descent speed field F on Ω.
4. Evolve Ω_0 along F for various times, can be done in parallel.
5. Apply line search rule (e. g., Armijo) and perform step.
6. Repeat until sufficient reduction of the cost or convergence.

Generic, callback-based implementation: `so_run_descent`



Geometric constraints:

- Set $F = 0$ on frozen regions: `ls_enforce_speed`
- Projection of the shape after a step: `ls_enforce`

References

[1] D. Kraft. A Hopf-Lax Formula for the Time Evolution of the Level-Set Equation and a New Approach to Shape Sensitivity Analysis. Preprint IGDK-2015-18, `https://igdk1754.ma.tum.de/foswiki/pub/IGDK1754/Preprints/Kraft_2015A.pdf`. Submitted to: Interfaces and Free Boundaries.

[2] D. Kraft. *A Level-Set Framework for Shape Optimisation*. PhD thesis, University of Graz, 2015.

[3] 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, Cambridge, second edn., 1999.

TUM · Universität der Bundeswehr München · UNI GRAZ · TU Graz