The level-set Package for GNU Octave

Daniel Kraft

The Level-Set Method for Shape Optimisation

For a level-set function \( \phi: \mathbb{R}^n \times [0, \infty) \to \mathbb{R} \), 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 \( \Omega_t \subset \mathbb{R}^n \) by the speed method:
\[
\phi_t + F(x) \nabla \phi = 0, \quad \phi(t, 0) = \phi_0
\]
Propagation in time with the level-set equation:
\[
F: \mathbb{R}^n \to \mathbb{R}
\]
\( F \) is a scalar speed field.
Left: \( F(a) < F(b) < F(c) \)

Basic Operations with Level-Set Functions

Set predicates:
- \( \text{ls_inside} \)
- \( \text{ls_isempty}, \text{ls_subset} \)
- \( \text{ls_equal}, \text{ls_disjoint} \)

Basic shapes with \( \text{ls_genbasic} \) 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:
\[
\begin{align*}
\text{nb} &= \text{ls_nb_from_geom}(g, \text{phi}0); \quad \% \text{optional with struct g} \\
\text{d} &= \text{ls_solve_stationary}(	ext{phi}0, F, h, \text{nb}); \\
\text{phi}T &= \text{ls_extract_solution}(t, d, \text{phi}0, F);
\end{align*}
\]

Descent Recording and Replay

The framework around \texttt{so_run_descent} allows also for logging and replay:
- \texttt{so_save_descent} Keep records of all descent iterations.
- \texttt{so_replay_descent} Replay steps without recomputation.
- \texttt{so_explore_descent} Interactively step through the descent.

References

