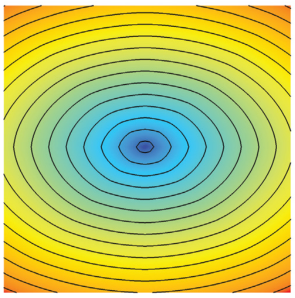
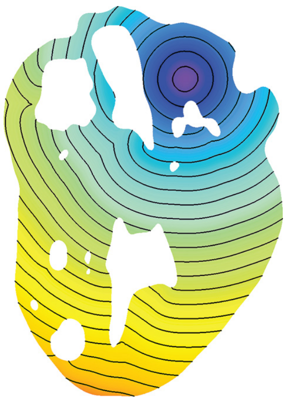
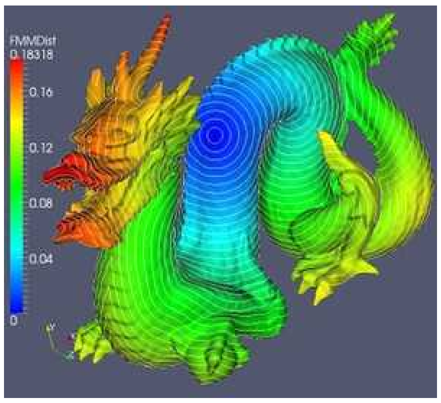
The Eikonal Equation







An eikonal equation is a non-linear first-order partial differential equation that is encountered in problems of wave propagation or in Hamiltonian system. It may be used to computer the continuos shortest path (geodesic) between points, electromagnetic potential, the arrival time of an acoustic wave, etc.

In most generic term the problem is: find the function u that satisfies an equation of the type

$$\begin{cases} H(\mathbf{x}, \nabla u(\mathbf{x})) = 1 & \mathbf{x} \in \Omega \\ u(\mathbf{x}) = g(\mathbf{x}) & \mathbf{x} \in \Gamma \subset \partial\Omega \end{cases} \qquad (1)$$

In most cases we have

$$H(\mathbf{x}, \nabla u(\mathbf{x})) = |\nabla u(\mathbf{x})|_M = \sqrt{\nabla u(\mathbf{x})^T M(\mathbf{x}) \nabla u(\mathbf{x})}, \qquad (1)$$

where M is a symmetric positive definite function. In the simplest cases $M = c^2 I$, so the problem reads simply

$$\begin{cases} |\nabla u| = 1/c & \mathbf{x} \in \Omega \\ u(\mathbf{x}) = g(\mathbf{x}) & \mathbf{x} \in \Gamma \subset \partial\Omega \end{cases} \qquad (1)$$

and c represents the celerity of the wave (often taken equal to 1). If we exclude techniques based on a regularization of the problem, we may consider two main classes of algorithms operating on a mesh covering the domain

1. Fast marching methods. Developed originally by J. Sethian, is a special case of a level-set method. An important feature is that at each step of the algorithm the solution is updated considering among a set of “active nodes” the one which has the smallest value of u. In this way the causality principle is always satisfied (if we interpret u as the arrival time of a wave, the causality principle corresponds to the fact that the past cannot be influenced by the future).
2. Fast sweeping methods. They can be thought as a sort of non linear Gauss-Siedel iteration, where the solution is evolved in an iterative fashion until the changes between two successive iterate is smaller than a given tolerance, or other criterions are met.
3.

Fast marching methods have the advantage of having the solution computed after a number of steps equal to the mesh nodes, but are more difficult to parallelise than Fast sweeping type methods since the each update requires to find the active node with smallest value. They have both been developed originally for regular Cartesian meshes, but later extended to triangular and tetrahedral grids.

Both class of methods rely on the solution of a **local problem,** which is an optimization problem on a single mesh element. In the given literature the local problem is solved exactly, but the solution requires a sufficiently regular grid, and needs to solve a non-linear problem which is (particularly in 3D) rather nasty. For this hands-on, I suggest to solve the local problem with an optimization algorithm using the code contained in LocalProblem/, based on modified version of the *LineSearch* Example of the course. The file `solveEikonalProblem.cpp` contains the function to use, and `main_eikonal.cpp` contains an example.

What you have to do
- Implement the algorithm described in the two references by Z. Fu et al. indicated below. Start with a scalar version. I will suggest to start with triangular meshes on a plane and tetrahedra (the algorithm si basically the same). Let alone the case of triangulated surfaces for the start. This way you can use for the Local Problem directly the solver contained in the LocalProblem/ folder.
- Move to the parallel version using OpenMP
- Use some triangular/tetrahedral mesh to test the problem. You may use the examples indicated in the generateMesh function of matlab or other tools available on the web (for instance the code triangle for triangular mesh and the code gmsh or tetgen for tetrahedral meshes). For the test you can just choose a portion of the domain boundary where to fix u and then see what happens. You may use paraview to see the solution (particularly useful in 3D)

Extra work
You may decide to extend the work to include

- The case of triangulated surfaces. It requires basically a modification (minimal indeed) of the local solver. Of course not triangles are immersed in a 3D space. Done that it would be nice to create a function to compute the geodesic: the shortest line connecting two points on a surface. See for instance here
- You may try to implement the modifications proposed in D. Ganellari et. al, but then to see some real advantage you need to have access to a massively parallel architecture.

Provided Bibliography

in the Bibliography folder:

- A fast iterative method for eikonal equations, Won-ki Jeong,and Ross T. Whitaker. *The original Ross-Won-Witaker algorithm for solving Eikonal equation*
- A fast iterative method for solving the eikonal equation on triangulated surface by Zhisong Fu , Won-ki Jeong, Yongsheng Pan , Robert M. Kirby, and Ross T. Whitaker. *A good description of the algorithm for triangular meshes*
- A fast iterative method for solving the eikonal equation on tetrahedral domains by Zhisong Fu, Robert M. Kirby, and Ross T. Whitaker. *The extension to tetrahedral meshes*

- A massively parallel Eikonal solver on unstructured meshes by Daniel Ganellari, Gundolf Haase, and Gerhard Zumbusch. *A more sophisticated (and complex) data structures designed to implement the code efficiently on massively parallel architecture (GPUs).*

In the LocalProblem/Docs folder

- *LineSearch.pdf* A brief description of the optimization algorithm.
- *NotesOnLocalProblemByOptimization.pdf*. More details of the algorithm implemented in the code provided in the LocalSolver folder