




# A Fast Iterative Method Python package

Thomas Grandits<sup>1</sup>

<sup>1</sup> Institute of Computer Graphics and Vision, TU Graz

DOI: [10.21105/joss.03641](https://doi.org/10.21105/joss.03641)

## Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

Editor: [Juanjo Bazán](#) 

## Reviewers:

- [@lucaferranti](#)
- [@RobertRosca](#)

Submitted: 13 July 2021

Published: 22 October 2021

## License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](#)).

## Summary

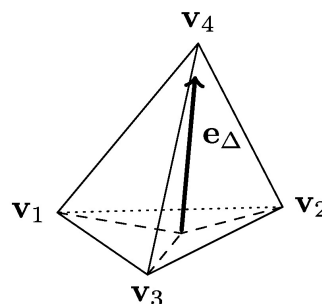
The anisotropic eikonal equation is a non-linear partial differential equation, given by

$$\begin{cases} \langle \nabla \phi, D \nabla \phi \rangle = 1 & \text{on } \Omega \\ \phi(\mathbf{x}_0) = g(\mathbf{x}_0) & \text{on } \Gamma \subset \Omega \end{cases}.$$

In practice, this problem is often associated with computing the earliest arrival times  $\phi$  of a wave from a set of given starting points  $\mathbf{x}_0$  through a heterogeneous medium (i.e. different velocities are assigned throughout the medium). This equation yields infinitely many weak solutions ([Evans, 2010](#)) and can thus not be straight-forwardly solved using standard Finite Element approaches.

`fim-python` implements the Fast Iterative Method (FIM), proposed in ([Fu et al., 2013](#)), purely in Python to solve the anisotropic eikonal equation by finding its unique viscosity solution. In this scenario, we compute  $\phi$  on tetrahedral/triangular meshes or line networks for a given  $D$ ,  $\mathbf{x}_0$  and  $g$ . The method is implemented both on the CPU using [numba](#) and [numpy](#), as well as the GPU with the help of [cupy](#) (depends on [CUDA](#)). The library is meant to be easily and rapidly used for repeated evaluations on a mesh.

The FIM locally computes an update rule to find the path the wavefront will take through a single element. Since the algorithm is restricted to linear elements, the path through an element will also be a straight line. In the case of tetrahedral domains, the FIM thus tries to find the path of the linear update from a face spanned by three vertices  $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$  to the opposite vertex  $\mathbf{v}_4$ . [Figure 1](#) visualizes the update. For triangles and lines, the algorithm behaves similarly but the update origin is limited to a side or vertex respectively. The exact equations used to solve this problem in this repository were previously described (among others) in ([Grandits et al., 2020](#)).



**Figure 1:** Update inside a single tetrahedron

Two different methods are implemented in `fim-python`: In the *Jacobi* method, the above local update rule is computed for all elements in each iteration until the change between two

subsequent iterations is smaller than a chosen  $\varepsilon$ . This version of the algorithm is best suited for the GPU, since it is optimal for a SIMD (single instruction multiple data) architecture. The *active list* method is more closely related to the method presented in (Fu et al., 2013): We keep track of all vertices that require a recomputation in the current iteration on a so-called active list which we keep up-to-date.

## Comparison to other tools

There are other tools available to solve variants of the eikonal equation, but they differ in functionality to `fim-python`.

`scikit-fmm` implements the Fast Marching Method (FMM) (Sethian, 1996), which was designed to solve the isotropic eikonal equation ( $D = cI$  for  $c \in \mathbb{R}$  and  $I$  being the identity matrix). The library works on uniform grids, rather than meshes.

`GPUMUM: Unstructured Eikonal` implements the FIM in CUDA for triangulated surfaces and tetrahedral meshes, but has no Python bindings and is designed as a command line tool for single evaluations.

## Statement of need

The eikonal equation has many practical applications, including cardiac electrophysiology, image processing and geoscience, to approximate wave propagation through a medium. In the example of cardiac electrophysiology (Franzone et al., 2014), the electrical activation times  $\phi$  are computed throughout the anisotropic heart muscle with varying conduction velocities  $D$ .

`fim-python` tries to wrap the FIM for CPU and GPU into an easy-to-use Python package for multiple evaluations with a straight-forward installation over `PyPI`. This should provide engineers and researchers alike with an accessible tool that allows evaluations of the eikonal equation for general scenarios.

## References

- Evans, L. C. (2010). *Partial differential equations* (Second, Vol. 19, p. xxii+749). American Mathematical Society, Providence, RI. <https://doi.org/10.1090/gsm/019>
- Franzone, P. C., Pavarino, L. F., & Scacchi, S. (2014). *Mathematical cardiac electrophysiology* (Vol. 13). Springer. <https://doi.org/10.1007/978-3-319-04801-7>
- Fu, Z., Kirby, R., & Whitaker, R. (2013). A Fast Iterative Method for Solving the Eikonal Equation on Tetrahedral Domains. *SIAM Journal on Scientific Computing*, 35(5), C473–C494. <https://doi.org/10.1137/120881956>
- Grandits, T., Gillette, K., Neic, A., Bayer, J., Vigmond, E., Pock, T., & Plank, G. (2020). An inverse Eikonal method for identifying ventricular activation sequences from epicardial activation maps. *Journal of Computational Physics*, 419, 109700. <https://doi.org/10.1016/j.jcp.2020.109700>
- Sethian, J. A. (1996). A fast marching level set method for monotonically advancing fronts. *Proceedings of the National Academy of Sciences*, 93(4), 1591–1595. <https://doi.org/10.1073/pnas.93.4.1591>