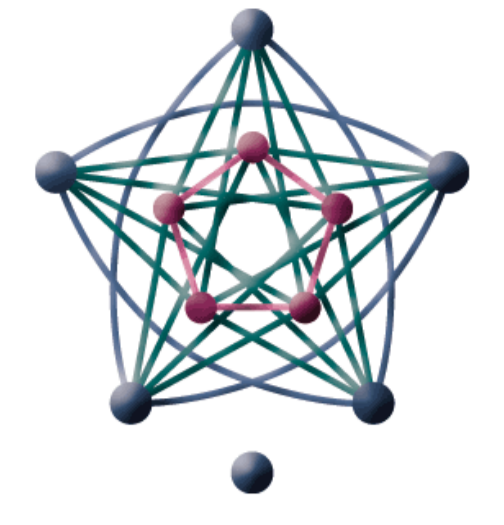




# DISTRIBUTED FAST MARCHING METHODS

Maria Cristina Tugurlan, Blaise Bourdin

Department of Mathematics, Louisiana State University, Baton Rouge, USA  
 ctugur@math.lsu.edu, bourdin@math.lsu.edu



## Abstract

Fast Marching represents a very efficient technique for solving front propagation problems which can be formulated as a boundary value partial differential equations with Dirichlet boundary condition, called Eikonal equation:

$$\begin{cases} F(x)|\nabla T(x)| = 1, & x \in \Omega \\ T(x) = 0, & x \in \Gamma, \end{cases} \quad (1)$$

where  $\Omega$  is a domain in  $\mathbb{R}^p$ ,  $\Gamma$  is the initial position of a curve evolving with normal velocity  $F$ ,  $F > 0$ .

Fast Marching Methods are a necessary step in Level Set Methods, which are widely used today in scientific computing. The classical Fast Marching Methods (FMM), based on finite differences, proposed by Osher and Sethian [1], are theoretically optimal in terms of operation count, and typically sequential. Parallelizing Fast Marching Methods is a step forward for employing the Level Set Methods on supercomputers. All the included simulations are done using the LONI resources.

## Fast Marching Methods

Fast Marching Methods consider only numerical schemes which guarantee a correct *upwind* direction and a correct approximation of the *viscosity solution* [1, 2] by using the numerical scheme:

$$[\max(D_{ij}^- T, -D_{ij}^+ T, 0)^2 + \max(D_{ij}^- T, -D_{ij}^+ T, 0)^2]^{1/2} = \frac{1}{F_{ij}}. \quad (2)$$

The Fast Marching algorithm partitions the grid points in three categories: *accepted*, *narrow band* and *far away*. The algorithm starts by putting the boundary condition into the *accepted* nodes, computing the downwind values for all the adjacent points and adding them to the *narrow band*.

### Fast Marching algorithm

#### loop

```

let P the point with the smallest value in the narrow band
add P to accepted set and delete P from narrow band
tag all neighbors of P as narrow band, if they are not already accepted
if neighbor N is in the far away set then
  delete N from far away set
  add N to narrow band
end if
recompute distances at all narrow band neighbors of P, by solving locally
the Eikonal Equation
end loop

```

**Theorem 1.** As the mesh size is going to zero, i.e.  $\Delta x \rightarrow 0$  and  $\Delta y \rightarrow 0$ , the numerical solution built by the Fast Marching Methods converges to the viscosity solution of (1).

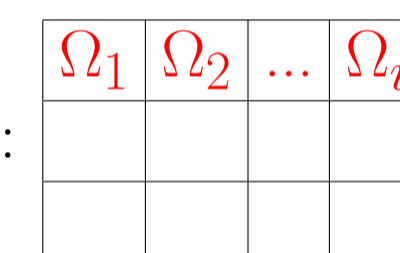
## Implementation

No special data structure, i.e. heap sort tree, is necessary to keep track of exact causal relationships. I maintain a looser relationship and update all *visited* points by using a double chained list and assigning flags to indicate the status of each point:

- flag for *accepted* nodes is set to 0,
- flag for *narrow band* nodes is set to 1,
- flag for *far away* nodes is 2.

For the distribute implementation

- Partition  $\Omega$  in  $\Omega_i$  subdomains:



- Each CPU has access only to its own subdomain
- Make subdomains interaction possible by expanding each subdomain by  $r$  ghost nodes
- Evaluation of (2) requires ghost points
- Ghost points need to be explicitly synchronize:

#### loop

```

Fast Marching on the subdomains (local FM)
{Exchange and update the boundary data at the interfaces}
Ghost points update
Narrow band reconstruction
Keep minimum value and restart the algorithm until all the points are accepted.
end loop

```

## Ghost Points Update

The Ghost points update procedure consists on:

- recompute the boundary values using the ghost nodes
- reassign the flags
- save the minimum value on the boundary
- use it to update the flags in the subdomain

The algorithm is graphically illustrated below.

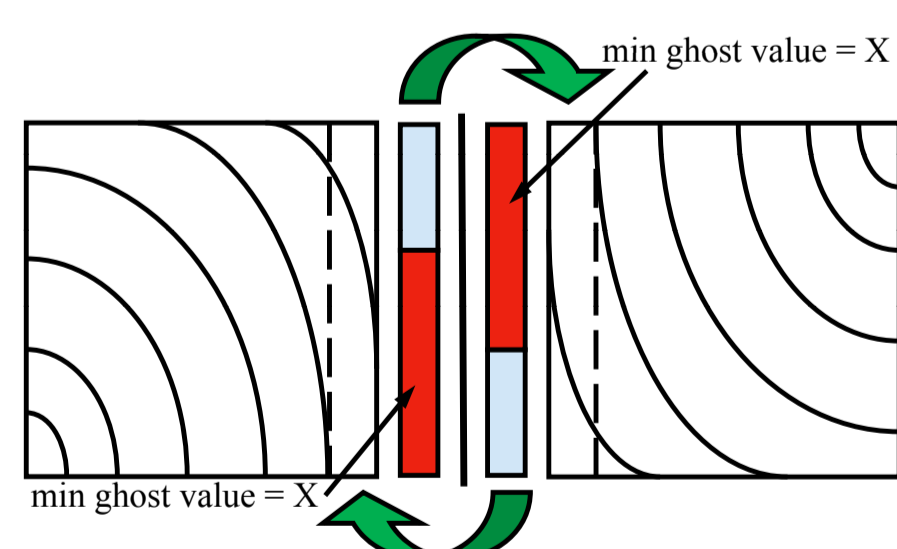


FIGURE 1: After Local FM

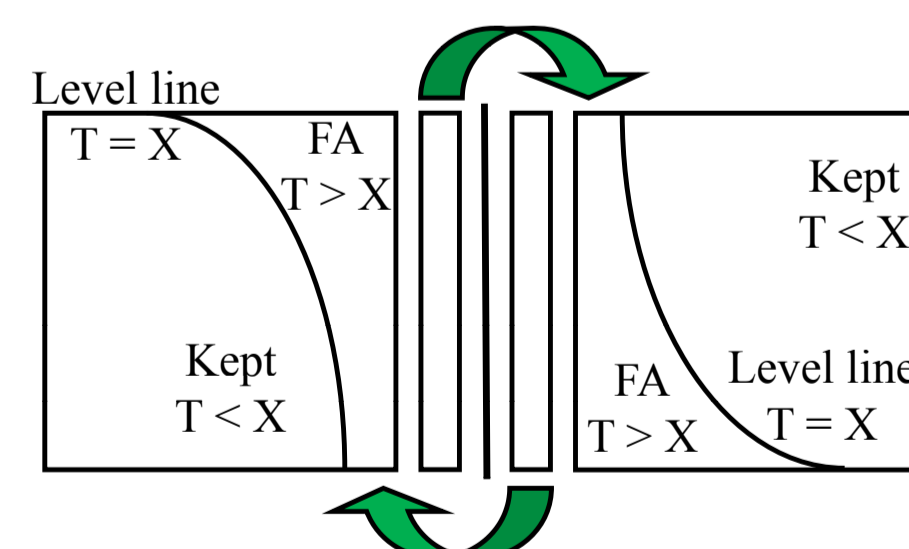


FIGURE 2: After Ghost Update

Approaches to update the boundary values at the interfaces:

- group the overlapping nodes of each subdomain in a sorted list
  - use this list to recompute the boundary values
- and
- perform a Fast Sweeping [3] at the interfaces and
  - update the overlapping regions of the subdomains

## Numerical Results

The efficiency of the parallel algorithms depends on:

- the required amount of communication between subdomains, i.e. how often boundary nodes change to *accepted* status.
- preserving the upwind structure of the solution during the algorithm execution.

**Theorem 2.** The parallel algorithm complexity is in between  $\mathcal{O}(N^2)$  and  $\mathcal{O}(N^3)$ .

The simulations are done using the LONI resources and the test grid sizes are:

30×30 60×60 100×100 120×120 150×150 164×164  
 180×180 200×200 240×240 256×256 300×300 600×600

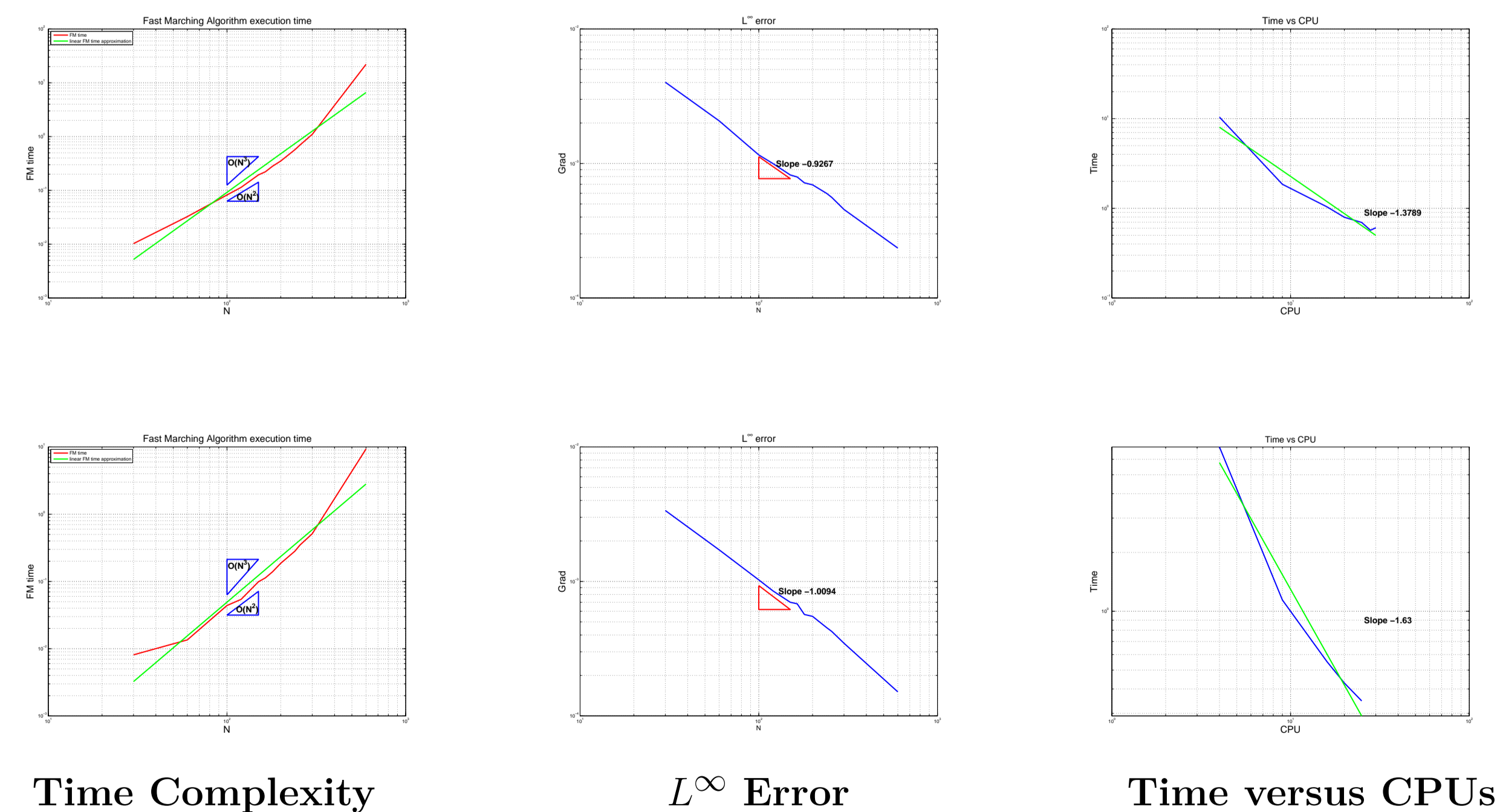


FIGURE 3: Numerical experiments for images with one and two source nodes in the corners

## References

- [1] S. Osher, J.A. Sethian, *Fronts Propagating with Curvature Dependent Speed: Algorithms Based on Hamilton-Jacobi Formulations*, Journal of Computational Physics, 79 (1988), pp.12-49.
- [2] J.A. Sethian, *A Fast Marching Level Set Method for Monotonically Advancing Fronts*, Proc. Natl. Acad. Sci, Vol. 93 (1996), pp. 1591-1595.
- [3] H.K.Zhao, *Fast Sweeping Method for Eikonal Equations*, Math. Comp., 74 (2005), pp. 603-627.