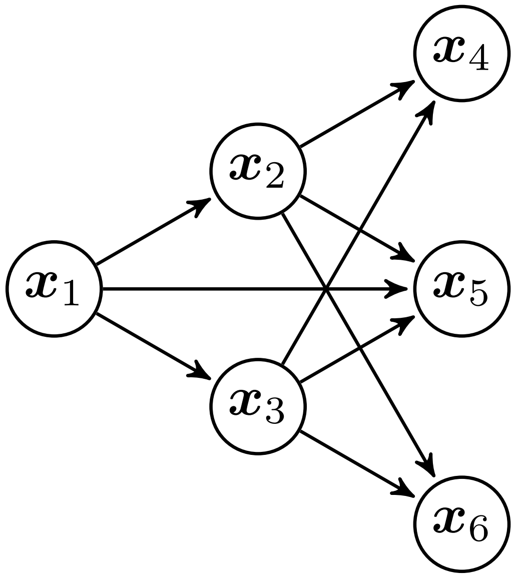
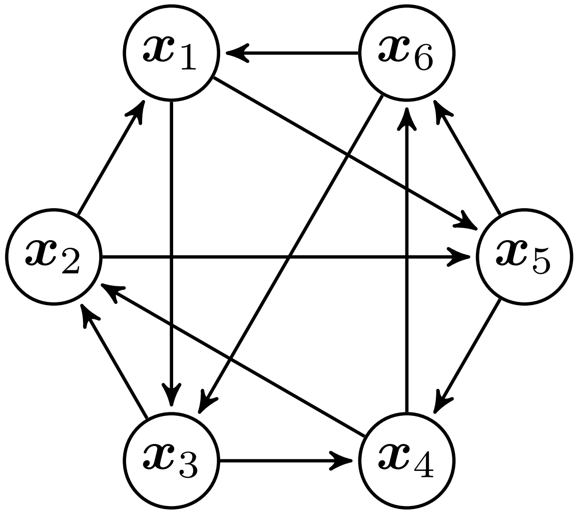
(Fast) Iterative and Non-iterative Methods for Discrete & Continuous Dynamic Programming Equations.

Alexander Vladimirsky
Department of Mathematics,
Cornell University,
Ithaca, NY 14853
vlad@math.cornell.edu

Causality as a source of efficiency.

	“Easy”	**“Hard”**
Shortest Paths on Graphs		
Efficient Numerics for PDEs	$u_t - H(\nabla u, u, t, \boldsymbol{x}) = 0$	$H(\nabla u, u, \boldsymbol{x}) = 0$

Deterministic Shortest Paths in a Graph.

Nodes: $X = \{\boldsymbol{x_1}, \ldots, \boldsymbol{x_M}\}$
Transition time-penalty: $K_{ij} \geq \delta > 0$ (assumed $+\infty$ if $\boldsymbol{x_j} \notin N(\boldsymbol{x_i})$)
Bounded degree of nodes: $|N(\boldsymbol{x_i})| < \kappa \ll M$ for all i
Exit time-penalty: $q(\boldsymbol{x_i})$ for all $\boldsymbol{x_i} \in Q \subset X$

Dynamic Programming: The **value function** $U(\boldsymbol{x_i}) = U_i$ is the minimum required total-time-to-exit starting from $\boldsymbol{x_i}$.

Bellman's Optimality Principle:

The vector $U = (U_1, \ldots, U_M)$ is a fixed point of an operator $\mathcal{F} = \tilde{\boldsymbol{R}}^M \mapsto \tilde{\boldsymbol{R}}^M$.

$$U_i \;=\; \mathcal{F}_i(U) \;=\; \begin{cases} \min_{\boldsymbol{x_j} \in N(\boldsymbol{x_i})} \{K_{ij} + U_j\}, & \text{if } \boldsymbol{x}_i \notin Q; \\ q(\boldsymbol{x_i}), & \text{if } \boldsymbol{x}_i \in Q. \end{cases}$$

A coupled system of M non-linear equations!

General Fixed Point Iterations

Consider a **general** coupled nonlinear system $U = \mathcal{F}(U)$.

Fixed point iterations: $U^{k+1} = \mathcal{F}(U^k)$, where $U^0 \in \tilde{\boldsymbol{R}}^M$ is a suitable initial guess.
$O(M)$ cost per one iteration, assuming $\mathcal{F}_i(V)$ evaluation costs $O(1)$.

- Similar to recovering a stationary solution of a “time-dependent” problem.
- In Dynamic Programming known as the “Value Iterations”.
- Infinitely many iterations are generally needed to converge
(on an infinite-precision computer).
- Estimates for the rate of convergence are not always available.
- Causal properties $\implies$ finitely many iterations for any U^0.

E.g., for shortest path in a graph with $\delta > 0$,
$U_i^k = U_i$ for any $\boldsymbol{x}_i$ whose optimal path has length $\leq k$.

So, (# of iterations) $\leq$ (the maximum # of links in a shortest path) $\leq M$.
Thus, the total cost is $O(M^2)$, assuming a fixed $\kappa \ll M$.

Fast methods: speed up convergence either by increasing the rate or

by decreasing the total number of needed iterations (in finitely-convergent problems).

“Relaxed Iterations” and Types of Causality

Gauss-Seidel Relaxation:

$$U_i^{k+1} = \mathcal{F}_i(U_1^{k+1}, \ldots, U_{i-1}^{k+1}, U_i^k, \ldots, U_M^k).$$

In finitely-convergent problems, the number of iterations now heavily depends on the **ordering** of the variables/nodes/gridpoints !

The ordering is **causal** if using it a *single* G-S iteration leads to convergence.

The operator $\mathcal{F}$ is causal
- **causal** if there exists a causal ordering of gridpoints.
- **explicitly causal** if a causal ordering is a priori known.
- **monotone-causal** if in the causal ordering $U_i < U_j \quad \Longrightarrow \quad i < j$.

Explicitly causal examples: shortest paths in acyclic graphs, explicit time-marching schemes for time-dependent PDEs.

Monotone-causal example: shortest paths in di-graphs with $K_{ij} > \delta \geq 0$, upwind finite-difference discretization of the Eikonal PDE.

For non-causal problems: can a good ordering improve the rate or decrease the number of iterations needed to reach a fixed tolerance?

Generic Label-Correcting Algorithm

- Dynamically defined ordering.
- Temporary labels V_i's.
- Maintains a list L of recently updated nodes.

- Every time a node is removed from L, all those potentially depending on it are re-evaluated and added to L (if not there yet).
- Terminates when L is empty.

- Upon termination $V_i = U_i$ for all nodes $\boldsymbol{x}_i$, **regardless of the particular add/remove choices**.
- Always converges in at most M^2 evaluations for all causal problems.

Good add/remove choices might reduce the number of evaluations.

Generic Label-Correcting Algorithm

```
Initialization:
for each $\boldsymbol{x}_i \in Q$ do
    $V_i \leftarrow q_i$
for each $\boldsymbol{x}_i \notin Q$ do
    if $N(\boldsymbol{x}_i) \bigcap Q \neq \emptyset$ then
        $V_i \leftarrow \min_{\boldsymbol{x}_j \in N(\boldsymbol{x}_i) \bigcap Q} \{K_{ij} + q_j\}$
        add $\boldsymbol{x}_i$ to the list $L$
    else
        $V_i \leftarrow \infty$

Main Loop:
while $L$ is nonempty do
    Remove a node $\boldsymbol{x}_j$ from the list $L$
    for each $\boldsymbol{x}_i \notin Q$ such that $\boldsymbol{x}_j \in N(\boldsymbol{x}_i)$ and $V_j < V_i$ do
        $\widetilde{V} \leftarrow K_{ij} + V_j$
        if $\widetilde{V} < V_i$ then
            $V_i \leftarrow \widetilde{V}$
            if $\boldsymbol{x}_i \notin L$ then
                add $\boldsymbol{x}_i$ to the list $L$
```

How to add & remove nodes from L ?

- **Bellman-Ford [1957] :**

L is a queue; remove from the top, add to the bottom.

To speed up things: process “the most influential” nodes first.

- **D’Esopo-Pape [1974] :** same as B-F, but add at the top if that’s not the first time for this node.

But for these problems “added early” is not the same as “more influential”. Instead, smaller label $\Longleftrightarrow$ more influence.

- **Dijkstra [1959] :** remove the smallest label;

natural implementation: L is a binary heap.

- **Dial [1969] :** remove a bunch of closely clustered “smallest” labels simultaneously; natural implementation: L is a collection of “buckets”.

These latter **label-setting** methods have a great property:
once removed from L, a node never re-enters it.

As a result, the number of re-evaluations per node is bound by κ.

Label-Setting Methods

Causality: Each node depends only on the “smaller” neighbors.

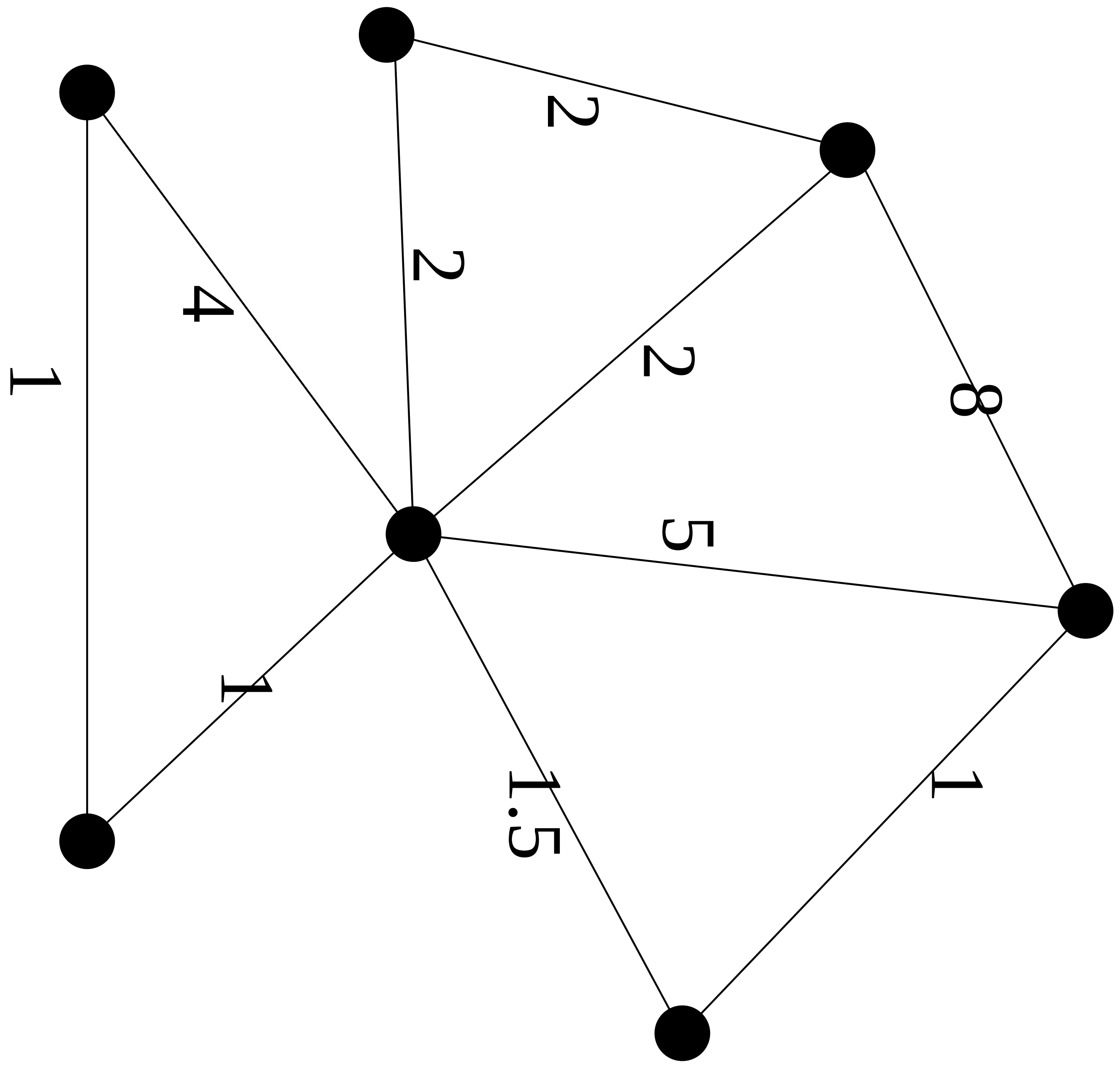
“If you use The Known
to tentatively compute The Still Unknown
then the smallest of The Tentatively Known
is actually Known.”

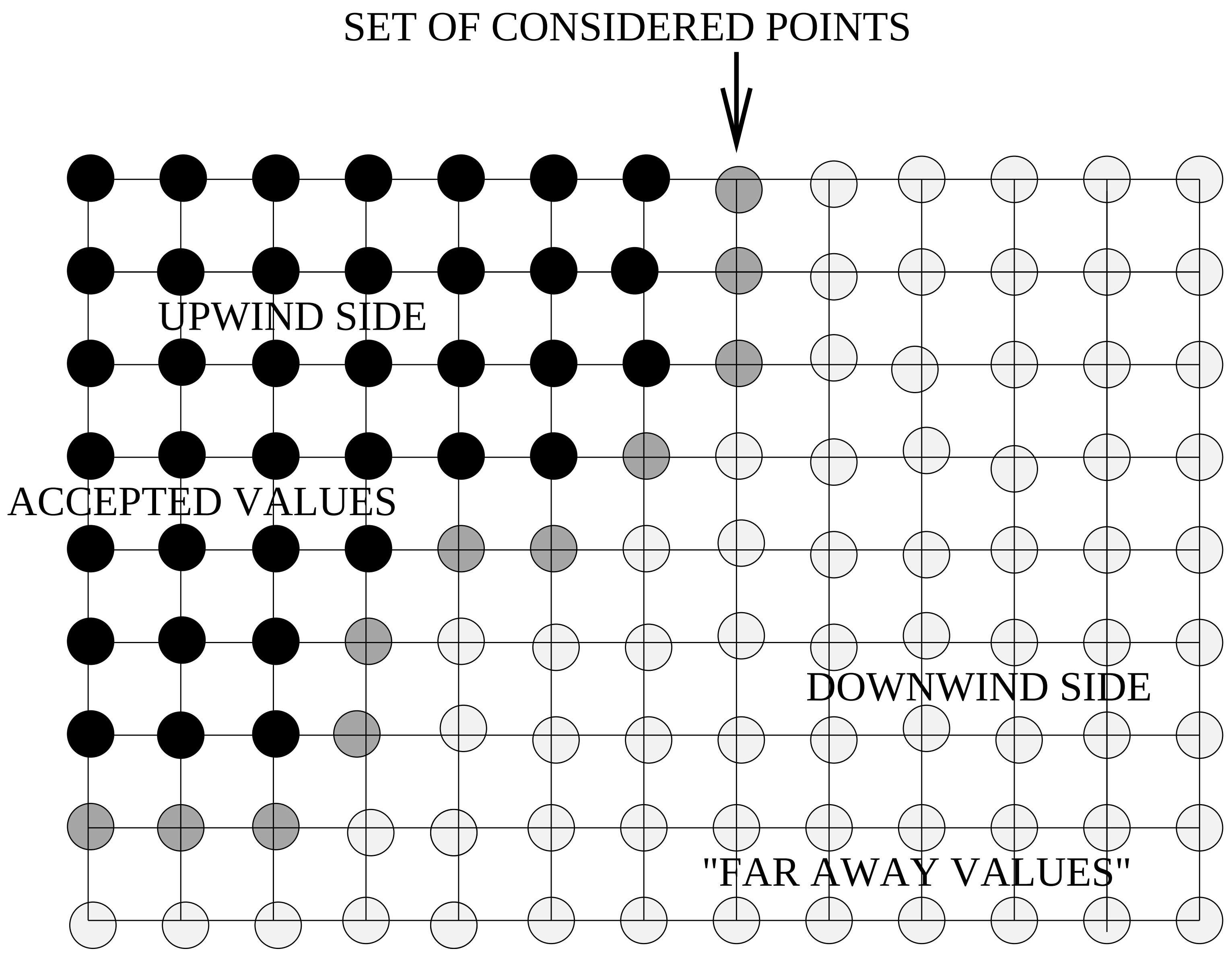
Dijkstra’s has $O(M \log M)$ complexity if implemented using heap-sort.

Dial’s has $O(M)$ complexity relying on a list of “buckets” of width δ.

These are **single-pass** or essentially **non-iterative** – the bound on the number of “iterations” is independent from M and known in advance.

Disadvantages : label-setting methods rely on more sophisticated data structures and are harder to parallelize.


2
2
4
1
2
8
5
1
1.5
1


SET OF CONSIDERED POINTS
UPWIND SIDE
ACCEPTED VALUES
DOWNWIND SIDE
"FAR AWAY VALUES"

Again: how to add & remove nodes from L ?

Many label-correcting algorithms strive to attain the same good properties of label-setting methods, but without sorting the labels.

Three label-correcting algorithms by Bertsekas (& co-authors):

- **SLF (Small Labels First) [1993]:** same as B-F, but add at the top if the label is smaller than the current top.
- **LLL (Large Labels Last) [1996]:** let $\overline{V}$ be the average of labels currently in L; if the top is bigger than $\overline{V}$, don't remove but simply shift it to the bottom instead.
- **SLF/LLL :** an obvious combination of the above.

Parallel/Asynchronous versions also available (& scale fairly well).

- **Thresholding :** split L into two queues L_1 and L_2; maintain a threshold value $\widehat{V}$; if the new label is below $\widehat{V}$, add it to L_1; otherwise – to L_2. Once L_1 is empty, increment $\widehat{V}$ (by how much?), interchange L_1 & L_2, and continue. [**Glover, F., Glover, R., and Klingman; 1986**].

Discretizing Hyperbolic Boundary Value Problems.

$$H(\nabla u(\boldsymbol{x}), u(\boldsymbol{x}), \boldsymbol{x}) = 0, \quad \boldsymbol{x} \in \Omega \subset R^n; \qquad u(\boldsymbol{x}) = 0 \text{ on } \partial\Omega$$

Wanted: the “correct” solution $u(\boldsymbol{x})$.

Usual discretization strategy: $\nabla u(\boldsymbol{x_i})$ is approximated using U_i and $NU(\boldsymbol{x_i}) = \left\{U_j | \boldsymbol{x_j} \text{ is adjacent to } \boldsymbol{x_i}\right\}$.

Need to solve $\overline{H}(U_i, NU(\boldsymbol{x_i}), \boldsymbol{x_i}) = 0$ at each grid point.

- system of M non-linear **coupled** equations;
- solved by iterations (sometimes with Gauss-Seidel relaxation);
- often too slow (even with relaxation)

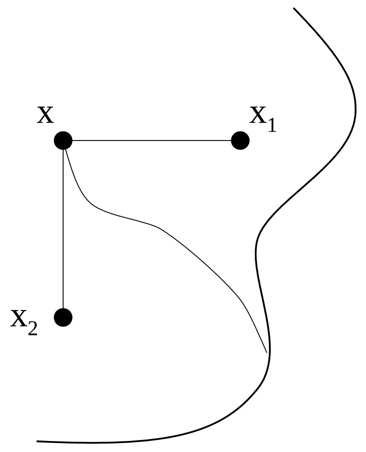
Can we do any better?
Can we de-couple the system and solve equations one at a time?

Causality in PDE discretizations.

$$H(\nabla u(\boldsymbol{x}), u(\boldsymbol{x}), \boldsymbol{x}) = 0, \quad \boldsymbol{x} \in \Omega \subset R^2; \qquad u(\boldsymbol{x}) = 0 \text{ on } \partial\Omega$$

Wanted:
the ordering of grid points $\boldsymbol{x_1}, \ldots, \boldsymbol{x_M}$ such that if U_i depends upon U_j then $i > j$.

- finding such an ordering $\quad\Longleftrightarrow\quad$ de-coupling the system
- only some discretizations possess this property



Why should this be at all possible?
The characteristic contains all the information
about the value of u at the point;
no need to look in all the directions.

Upwinding discretization: If simplex $\boldsymbol{x}\boldsymbol{x_1}\boldsymbol{x_2}$ contains the characteristic for the point $\boldsymbol{x}$, then solve for $U(x)$ the discretized equation $\tilde{H}(U(\boldsymbol{x}), U(\boldsymbol{x_1}), U(\boldsymbol{x_2}), \boldsymbol{x}) = 0.$

A slight problem: characteristic direction not known in advance.

Our hope: the dependency graph for the computational elements will be **acyclic**; traversing from the leaves up will give a causal ordering.
Easy to check/exploit in the linear case (**Lesaint & Raviart 1974**).

But what if the dependency graph is not acyclic ?
One solution: a multi-layer graph structure with the inter-dependency cycles not allowed to involve "higher" layers.
Resolve each level (if necessary - by iterations) before moving up.
Additional step: build your mesh to avoid the cycles (if possible).

Several methods based on the above :

• Discontinuous Galerkin:
Haber et al. 2001 (hyperbolic conservation laws); **Barth 2001** (static Eikonal).
• Heuristic speed-up in iterative solvers for static HJ :
Falcone et al. 1997; **Bornemann & Rasch 2004**;
Polymenakos, Bertsekas, & Tsitsiklis 1998; ...

Four more ideas for the non-linear case:

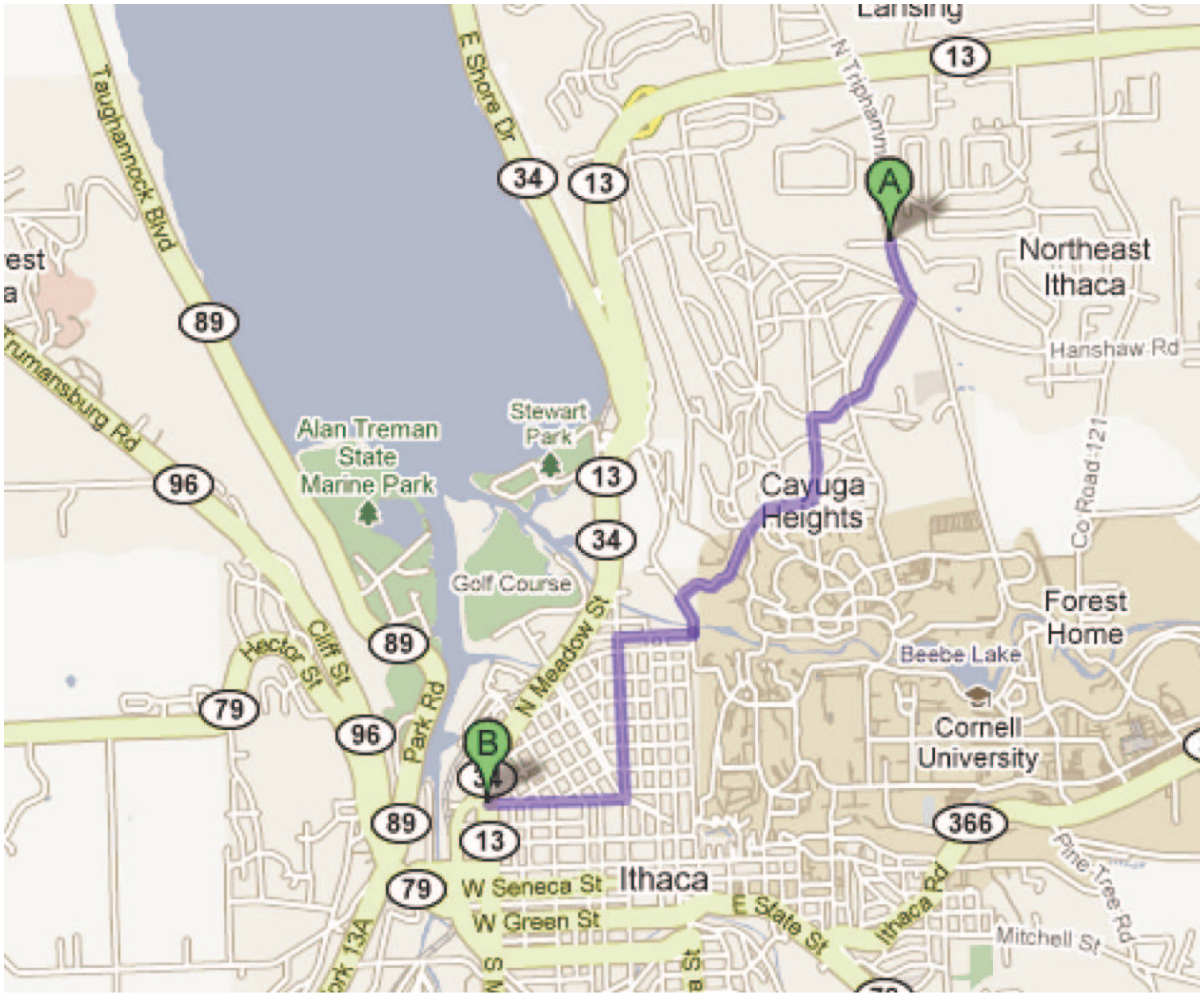
• raise the dimensionality to enable time-marching (e.g., *Level Set Methods*);

• pretend that one of the dimensions is time-like (e.g., *Paraxial Approximation*);

• locally enlarge the stencil to ensure the cycles are absent (e.g., *Ordered Upwind*);

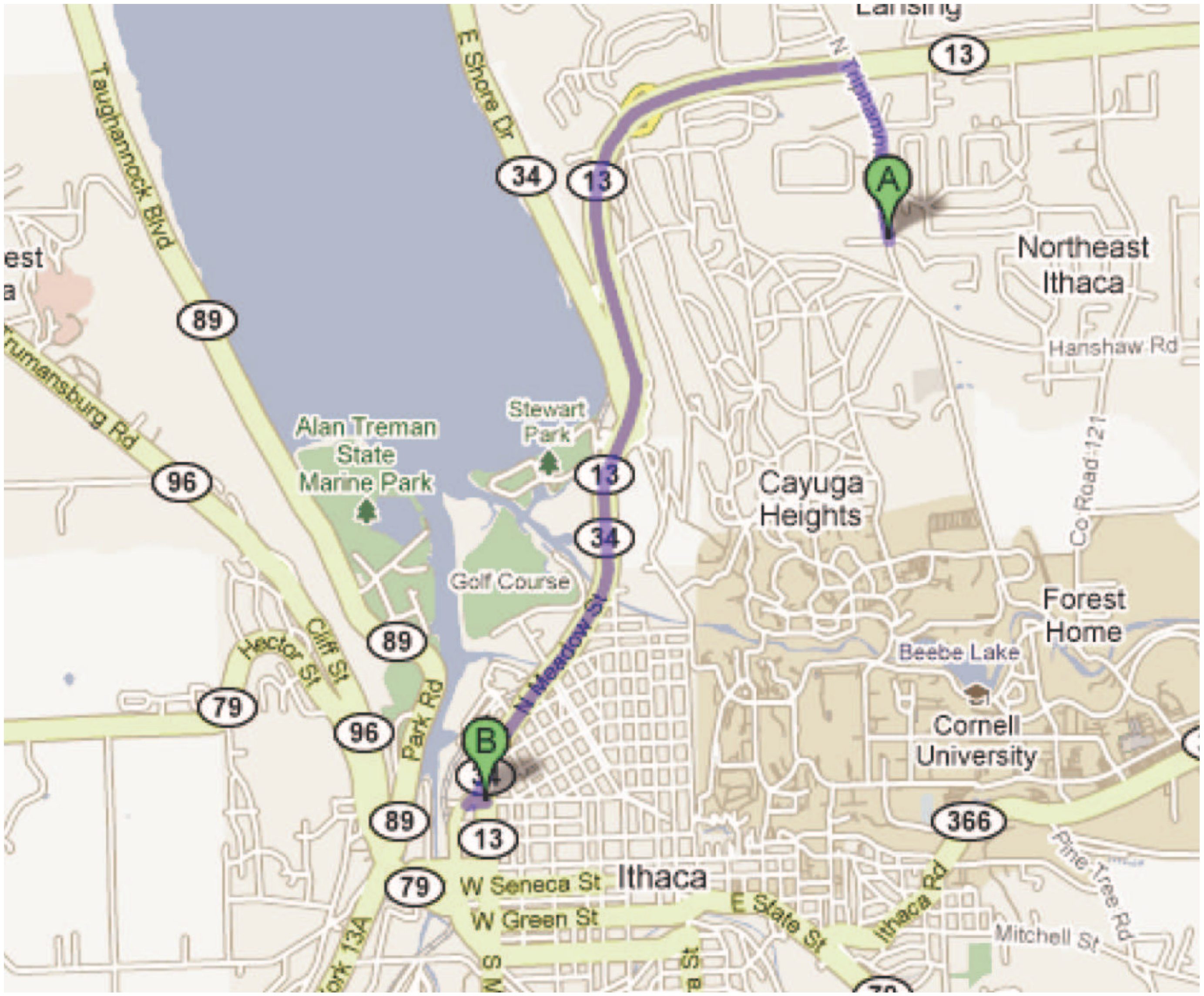
• alternate between different guesses for the acyclic structure (e.g., *Fast Sweeping*).

Driving directions: explicit causality?



11 minutes of driving

Driving directions: non-explicit causality!



8 minutes of driving

Continuous Optimal Trajectory Problem

Controlled system :

$$\begin{cases} \boldsymbol{y}'(t) = f(\boldsymbol{y}(t), \boldsymbol{a}(t))\, \boldsymbol{a}(t), & \|\boldsymbol{a}(t)\| = 1, \qquad f : R^2 \times S_1 \longmapsto R_+ \\ \boldsymbol{y}(0) = \boldsymbol{x}, & \boldsymbol{x} \in \Omega \subset R^2 \end{cases}$$

Assumptions : f and q are Lipschitz, $0 < F_1 \leq f(\boldsymbol{x}, \boldsymbol{a}) \leq F_2$, $\Upsilon = F_2/F1$.

Time-to-boundary $T(\boldsymbol{x}, \boldsymbol{a}(\cdot)) = \inf\{t \in R_{+,0} | \boldsymbol{y}(t) \in \partial\Omega\}$.
Exit time-penalty $q : \partial\Omega \longmapsto R_{+,0}$.
Total Time $\text{Cost}\,(\boldsymbol{x}, \boldsymbol{a}(\cdot)) = T\,(\boldsymbol{x}, \boldsymbol{a}(\cdot)) + q\,(\boldsymbol{y}\,(T(\boldsymbol{x}, \boldsymbol{a}(\cdot))))$.

Value function : $u(\boldsymbol{x}) = \inf_{\boldsymbol{a}(\cdot)} \text{Cost}(\boldsymbol{x}, \boldsymbol{a}(\cdot))$.

Bellman's Optimality Principle : $u(\boldsymbol{x}) = \inf_{\boldsymbol{a}(\cdot) \in \mathcal{A}} \{\tau \; + \; u(\boldsymbol{y}(\tau))\}$.

Hamilton-Jacobi-Bellman PDE:

$$\begin{aligned} \max_{\boldsymbol{a} \in S_1} \{(\nabla u(\boldsymbol{x}) \cdot (-\boldsymbol{a})) f(\boldsymbol{x}, \boldsymbol{a})\} \;=\; 1, \qquad & \boldsymbol{x} \in \Omega \\ u(\boldsymbol{x}) = q(\boldsymbol{x}), \qquad & \boldsymbol{x} \in \partial\Omega. \end{aligned}$$

Hamilton-Jacobi-Bellman PDE

Naive derivation:

$$\begin{aligned}
u(\boldsymbol{x}) &= \tau + \inf_{\boldsymbol{a}(\cdot)} \{u(\boldsymbol{y}(\tau))\} \\
&= \tau + \inf_{\boldsymbol{a}(\cdot)} \left\{u\left(\boldsymbol{x} + \tau f(\boldsymbol{x}, \boldsymbol{a}(0))\boldsymbol{a}(0) + O(\tau^2)\right)\right\} \\
&= \min_{\boldsymbol{a}\in S_1} \left\{\tau + u\left(\boldsymbol{x} + \tau f(\boldsymbol{x}, \boldsymbol{a})\boldsymbol{a} + O(\tau^2)\right)\right\} \\
&= \min_{\boldsymbol{a}\in S_1} \left\{\tau + u(\boldsymbol{x}) + \tau(\nabla u(\boldsymbol{x}) \cdot \boldsymbol{a}) f(\boldsymbol{x}, \boldsymbol{a}) + O(\tau^2)\right\} \\
&= \tau \min_{\boldsymbol{a}\in S_1} \{1 + (\nabla u(\boldsymbol{x}) \cdot \boldsymbol{a}) f(\boldsymbol{x}, \boldsymbol{a}) + O(\tau)\} + u(\boldsymbol{x}).
\end{aligned}$$

Thus,

$$\min_{\boldsymbol{a}\in S_1} \{(\nabla u(\boldsymbol{x}) \cdot \boldsymbol{a}) f(\boldsymbol{x}, \boldsymbol{a})\} + 1 = 0.$$

Isotropic case: $f = f(\boldsymbol{x}) \implies$ Eikonal PDE:

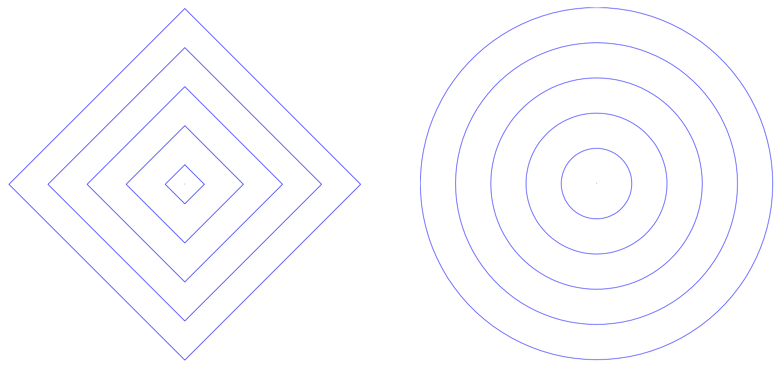
$$\begin{aligned}
\|\nabla u(\boldsymbol{x})\| f(\boldsymbol{x}) &= 1, & \boldsymbol{x} &\in \Omega \\
u(\boldsymbol{x}) &= q(\boldsymbol{x}), & \boldsymbol{x} &\in \partial\Omega.
\end{aligned}$$

The simplest example : $\|\nabla u\| = 1$.

$$\begin{cases} \|\nabla u(\boldsymbol{x})\| = 1, & \boldsymbol{x} \in \Omega \subset R^d \\ u(\boldsymbol{x}) = 0, & \boldsymbol{x} \in \partial\Omega \end{cases}$$

Interpretations of $u(\boldsymbol{x})$:

- Distance from the boundary.
- Value function for the optimal trajectory problem for the unit speed/cost vehicle movement.



Optimal trajectories - straight lines to the closest points on the boundary.

Shocks - wherever the closest point on the boundary is not unique.

Defining solutions of HJB PDEs

- **Non-existence of smooth solutions.**
- **Non-uniqueness of Lipschitz-continuous weak solutions.**
- **The viscosity solution as defined by Crandall, Evans and Lions.**
  u is the viscosity solution if for every function $\eta \in C^1(\Omega)$ the following holds:
  (i) if $u - \eta$ has a local minimum at $\boldsymbol{x_0} \in \Omega$ then

$$\min_{\boldsymbol{a} \in S_1} \{(\nabla \eta(\boldsymbol{x_0}) \cdot \boldsymbol{a}) f(\boldsymbol{x_0}, \boldsymbol{a})\} + 1 \leq 0; \qquad (1)$$

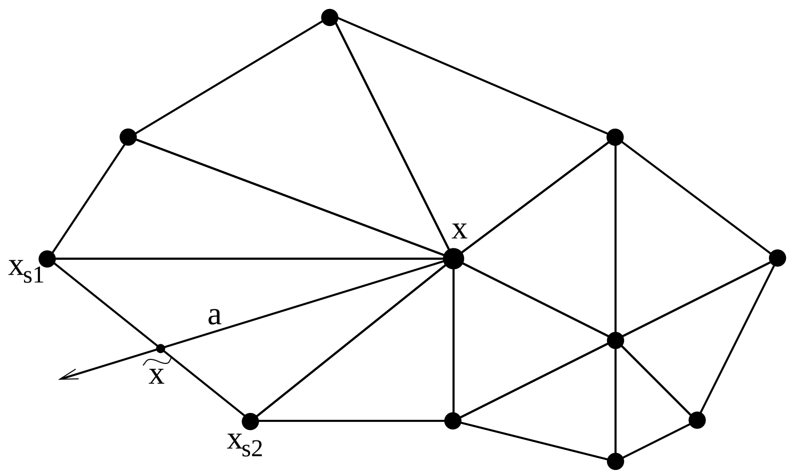
  (ii) if $u - \eta$ has a local maximum at $\boldsymbol{x_0} \in \Omega$ then

$$\min_{\boldsymbol{a} \in S_1} \{(\nabla \eta(\boldsymbol{x_0}) \cdot \boldsymbol{a}) f(\boldsymbol{x_0}, \boldsymbol{a})\} + 1 \geq 0; \qquad (2)$$

Modified definition (Sethian & AV 2001) :
use $S_1^{\eta, \boldsymbol{x}} = \{\boldsymbol{a} \in S_1 \mid \boldsymbol{a} \cdot \nabla \eta(\boldsymbol{x}) \leq -\|\nabla \eta(\boldsymbol{x})\| \Upsilon^{-1}\}$ instead of S_1.

Implication : a bound on the angle between characteristic direction and gradient of the viscosity solution.

A particular semi-Lagrangian discretization:



The idea: just keep going straight **until** the edge of the simplex.
Kushner, 1977; Gonzales & Rofman, 1985.

- $\tilde{\boldsymbol{x}} = \zeta \boldsymbol{x_{s,1}} + (1-\zeta)\boldsymbol{x_{s,2}}$
- $\tau(\zeta) = \|\tilde{\boldsymbol{x}} - \boldsymbol{x}\| = \|(\zeta \boldsymbol{x_{s,1}} + (1-\zeta)\boldsymbol{x_{s,2}}) - \boldsymbol{x}\|.$
- $\boldsymbol{a} = \boldsymbol{a}_\zeta = \frac{\tilde{\boldsymbol{x}} - \boldsymbol{x}}{\tau(\zeta)}.$

$$\begin{aligned} V_s(\boldsymbol{x}) &= \min_{\zeta \in [0,1]} \left\{ \frac{\tau(\zeta)}{f(\boldsymbol{x}, \boldsymbol{a}_\zeta)} + \zeta U(\boldsymbol{x_{s,1}}) + (1-\zeta) U(\boldsymbol{x_{s,2}}) \right\}; \\ U(\boldsymbol{x}) &= \min_{s \in S(\boldsymbol{x})} V_s(\boldsymbol{x}). \end{aligned}$$

A coupled system of M non-linear equations!

Another semi-Lagrangian discretization:

The idea: just keep going straight **until the time τ.**

- $\tilde{\boldsymbol{x}}_{\boldsymbol{a}} = \boldsymbol{x} + \tau f(\boldsymbol{x}, \boldsymbol{a}) \in s$
- $\tilde{\boldsymbol{x}}_{\boldsymbol{a}} = \zeta_1 \boldsymbol{x}_{\boldsymbol{s},\boldsymbol{1}} + \zeta_2 \boldsymbol{x}_{\boldsymbol{s},\boldsymbol{2}} + \zeta_3 \boldsymbol{x}_{\boldsymbol{s},\boldsymbol{3}}$, where $\sum \zeta_i = 1$
- $U(\tilde{\boldsymbol{x}}_{\boldsymbol{a}}) = \zeta_1 U(\boldsymbol{x}_{\boldsymbol{s},\boldsymbol{1}}) + \zeta_2 U(\boldsymbol{x}_{\boldsymbol{s},\boldsymbol{2}}) + \zeta_3 U(\boldsymbol{x}_{\boldsymbol{s},\boldsymbol{3}})$

$$U(\boldsymbol{x}) = \tau + \min_{\boldsymbol{a} \in S_1} \{U(\tilde{\boldsymbol{x}}_{\boldsymbol{a}})\}$$

- same as the previous if τ is so small that $s \in S(\boldsymbol{x})$;
- relies on convergence of a time-discrete control approximation;
- generally solved iteratively (possibly with GS relaxation).

Falcone, 1994 : For **isotropic** f, if $\tau \geq h/F_1$ and the mesh is "consistent", then the scheme converges in at most $D/(\tau F_1)$ iterations **and** each mesh point is updated exactly once. *Resulting complexity:* $O(M)$. *Resulting Error:* $O(h/F_1)$.

Related to: Dial-like method isotropic solvers (e.g., **Tsitsiklis, 1994**) and viability-kernel computation methods
(e.g., **Saint-Pierre, 2001**; **Bayen, Cruck, & Tomlin, 2002**).

Dijkstra-like methods for HJB Equations?

If the motion is **isotropic** (i.e. $\mathrm{f}(\boldsymbol{x}, \boldsymbol{a}) = \mathrm{f}(\boldsymbol{x})$), the same monotone de-coupling works:
"If you use The Known to tentatively compute The Still Unknown, then the smallest of The Tentatively Known is actually Known."

- **Isotropic Optimal Trajectory Problem:** Tsitsiklis' Algorithm (1994).
- **Isotropic Front Propagation Problem:** Sethian's Fast Marching Method (1996).

- Different perspectives (semi-Lagrangian vs. Eulerian).
- Same decoupling: both compute U_i before U_j iff $U_i \leq U_j$.
- Both non-iterative with complexity $O(M \log M)$.

Causality : characteristics = gradient lines for the Eikonal PDE.

Dial-like: Tsitsiklis (1995); Kim et al.(2000); on triangulated meshes: $\delta = \frac{h \cos \beta}{F_2}$, [AV, 2007].

In the general (**anisotropic**) case, this discretization need not be causal.

Ordered Upwind Methods: dynamically extend the stencil

just enough to restore the monotone causality [Sethian & AV, 2001].

Resulting computational complexity: $O(\Upsilon M \log M)$.

Causal Numerical Methods

- Shortest paths on graphs [**Dijkstra, 1959**]; [**Dial, 1969**];
- Isotropic optimal control/front propagation [**Tsitsiklis, 1994**]; [**Sethian, 1996**];
- Anisotropic optimal control/front propagation [**Sethian & AV, 2001**];
- Hybrid optimal control [**Branicky et al., 1999**]; [**Sethian & AV, 2003**];
- Invariant manifolds & quasi-linear PDEs [**Guckenheimer & AV, 2004**];
- Stochastic shortest paths on graphs [**Bertsekas, 2001**]; [**AV, 2007**];
- Games and mean curvature motion [**Cristiani & Falcone, 2006**]; [**Carlini, 2007**];
- Homogenization of HJB PDEs [**Oberman, Takei & AV, 2009**];
- Multi-objective optimal control [**Kumar & AV, 2010**];
- Randomly-terminated optimal control [**Andrews & AV, 2010**];
- Multi-valued solutions of PDEs [**Guckenheimer, Sethian & AV, in progress**];
- ....

Provable causality (not just a heuristic) $\Longrightarrow$ efficient computations.

Relevant in many PDEs & applications (not just HJ & optimal control).

Details: `http://www.math.cornell.edu/~vlad/`

Label-Correcting Methods for HJB PDEs

Bellman-Ford-like:

- for general static HJB [**Bornemann and Rasch; 2006**]
- asynchronous, for GPU-architectures [**Jeong and Whitaker; 2008**]
- for Eikonal [**Bak, McLaughlin, and Renzi; 2009**]

SLF/LLL-like:

- for Eikonal [**Polymenakos, Bertsekas, and Tsitsiklis; 1998**]
- asynchronous [**Bertsekas, Guerriero, and Musmanno; 1996**]

Thresholding-like:

- for Eikonal [**Bak, McLaughlin, and Renzi; 2009**]

Up to this moment: insufficient testing to determine the exact class of problems on which each of these methods outperforms the others.
Comprehensive/representative benchmarks are needed.

Alternating Direction Gauss-Seidel

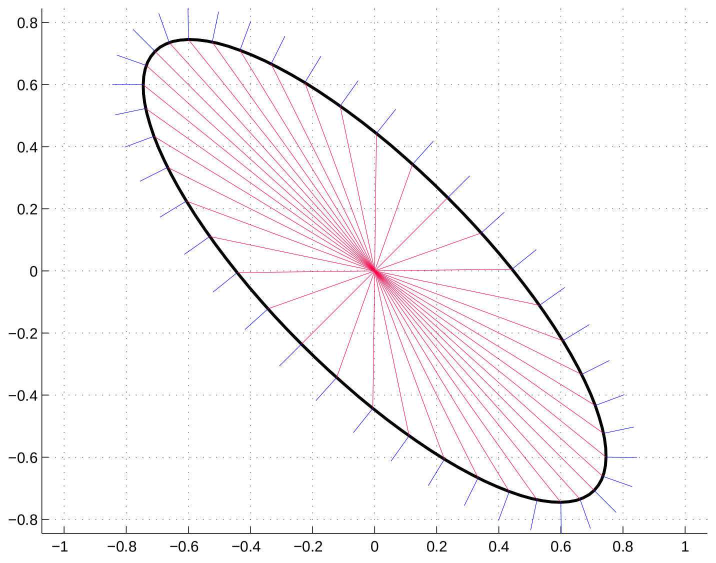
- **Danielsson, 1980;**
- **Boue & Dupuis, 1999;**
- **Dupuis & Szpiro, 2001;**
- **Tsai, Osher, Zhao, 2002-2004;**
- **Kao, Osher, Qian, 2004;**
- **Zhao, 2005;**
- **Zhang, Qian, Zhao, 2006;**
- **Bak, McLaughlin, Renzi, 2009;** ("locking sweeping" version)
- **...**

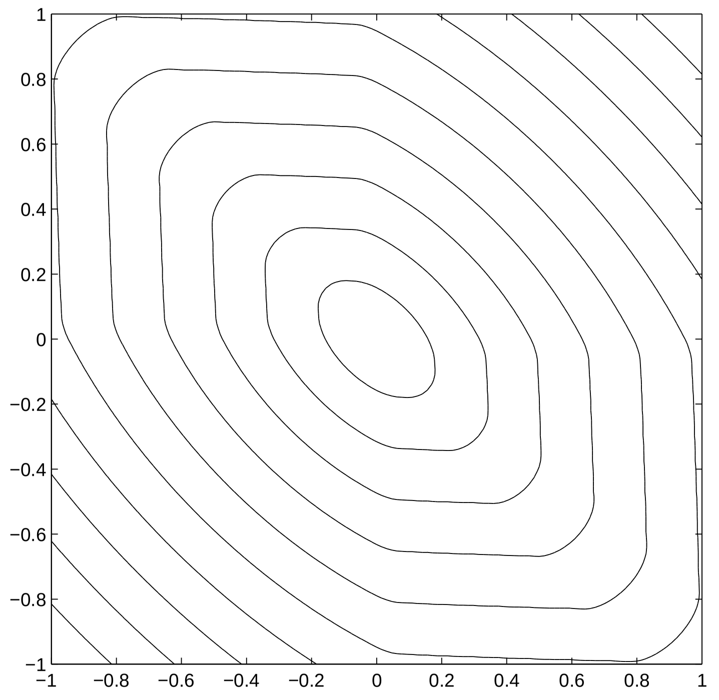
The complexity (in Eikonal case): $\approx O(kM)$, where k is the max number of switches from-quadrant-to-quadrant by a characteristic.

Very fast for problems where characteristics are largely straight lines and the domain geometry is simple, but in general - pay the price for inhomogeneity even in the absence of anisotropy.
See **(Gremaud & Kuster, 2006);** **(Hysing & Turek, 2005)**.

Anisotropic speed: the failure of Dijkstra-like methods.

The “monotone ordering” decoupling does not work here:
For $H(\nabla u(\boldsymbol{x}), \boldsymbol{x}) = 1$ the characteristics and the gradient lines do not have to be the same. **Nor do they have to lie in the same simplex!**





characteristic for $\boldsymbol{x}$ lies in the simplex $\boldsymbol{x}\boldsymbol{x_1}\boldsymbol{x_2}$ $\not\Rightarrow$ $u(\boldsymbol{x}) > \max\{u(\boldsymbol{x_1}), u(\boldsymbol{x_2})\}$

Building OUMs for Anisotropic Problems.

A more general **causality principle** is needed.

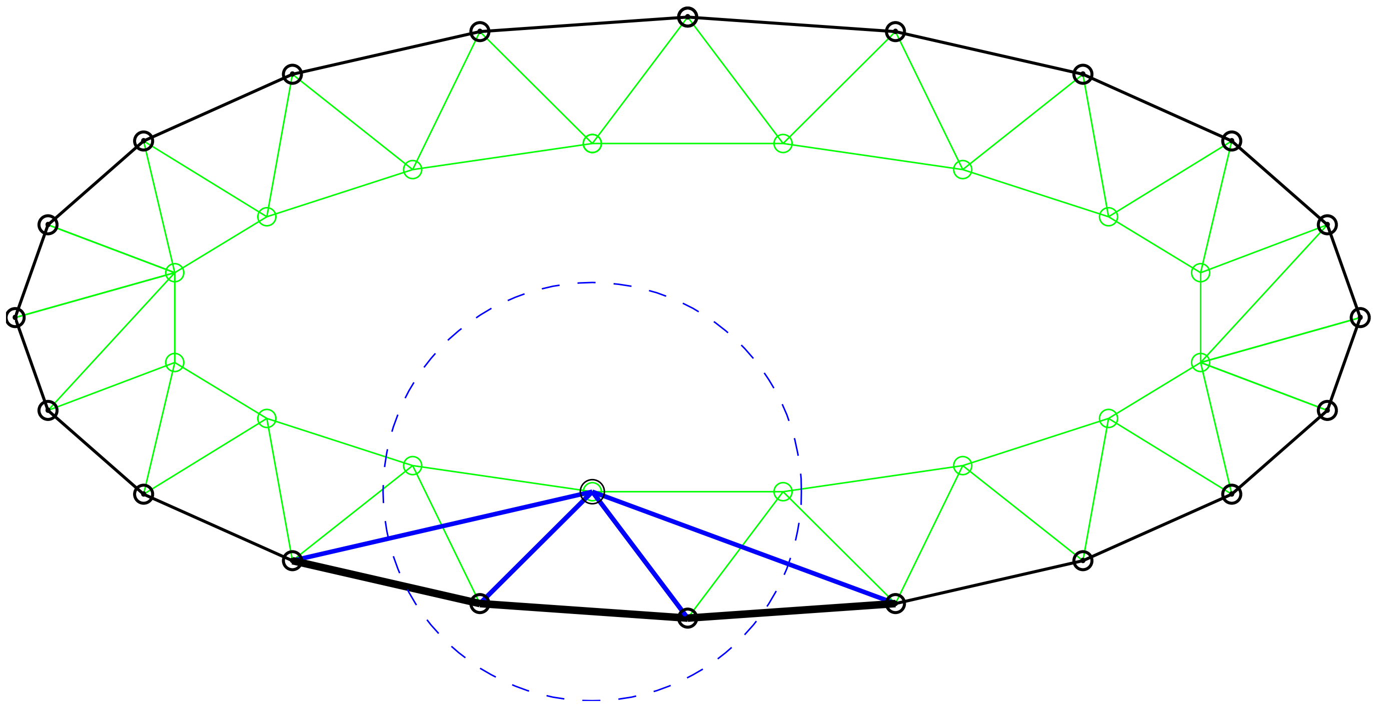
- **Lemma 1.** Consider the characteristic passing through a point $\bar{\boldsymbol{x}} \in \Omega$ and a level curve $u(\boldsymbol{x}) = C$, where $q_{max} < C < u(\bar{\boldsymbol{x}})$. The characteristic intersects that level set at some point $\tilde{\boldsymbol{x}}$. If $\bar{\boldsymbol{x}}$ is distance d away from the level set then

$$\|\tilde{\boldsymbol{x}} - \bar{\boldsymbol{x}}\| \leq d\frac{F_2}{F_1} = d\Upsilon. \tag{3}$$

- **Lemma 2.** Consider an unstructured mesh X of diameter h on Ω. Consider a simple closed curve Γ lying inside Ω with the property that for any point $\boldsymbol{x}$ on Γ, there exists a mesh point y inside Γ such that $\|\boldsymbol{x} - \boldsymbol{y}\| < h$. Suppose the mesh point $\bar{\boldsymbol{x}}_i$ has the smallest value $u(\bar{\boldsymbol{x}}_i)$ of all of the mesh points inside the curve. If the characteristic passing through $\bar{\boldsymbol{x}}_i$ intersects that curve at some point $\tilde{\boldsymbol{x}}_i$ then

$$\|\tilde{\boldsymbol{x}}_i - \bar{\boldsymbol{x}}_i\| \leq h\frac{F_2}{F_1} = h\Upsilon. \tag{4}$$

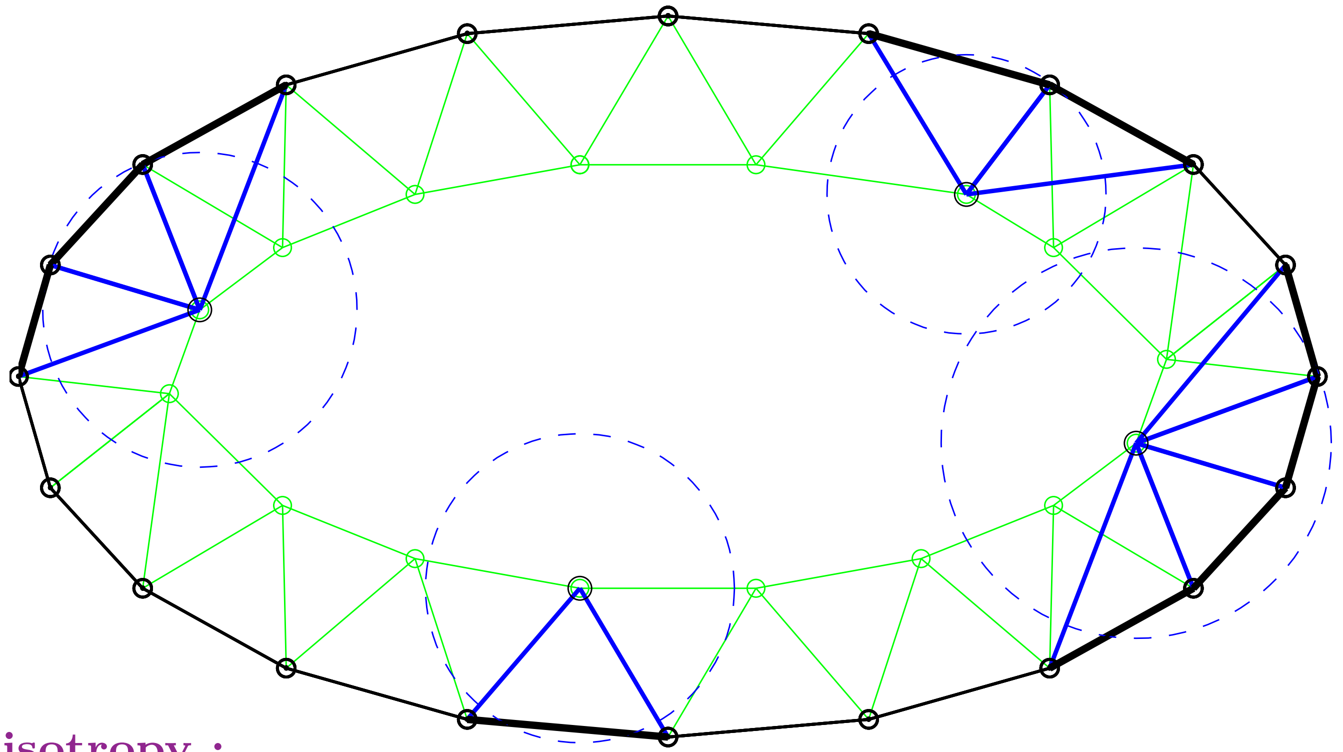
AcceptedFront, Considered Nodes, and NearFront.



$$NF(\boldsymbol{x_i}) = \left\{ (\boldsymbol{x_j}, \boldsymbol{x_k}) \in AF \mid \exists \tilde{\boldsymbol{x}} \text{ on } (\boldsymbol{x_j}, \boldsymbol{x_k}) \text{ s.t. } \|\tilde{\boldsymbol{x}} - \boldsymbol{x_i}\| \le h \frac{F_2}{F_1} \right\}.$$

$$U(\boldsymbol{x}) = \min_{(\boldsymbol{x_j}, \boldsymbol{x_k}) \in NF(\boldsymbol{x_i})} V_{j,k}(\boldsymbol{x}).$$

Localizing Anisotropy Coefficient for OUMs.



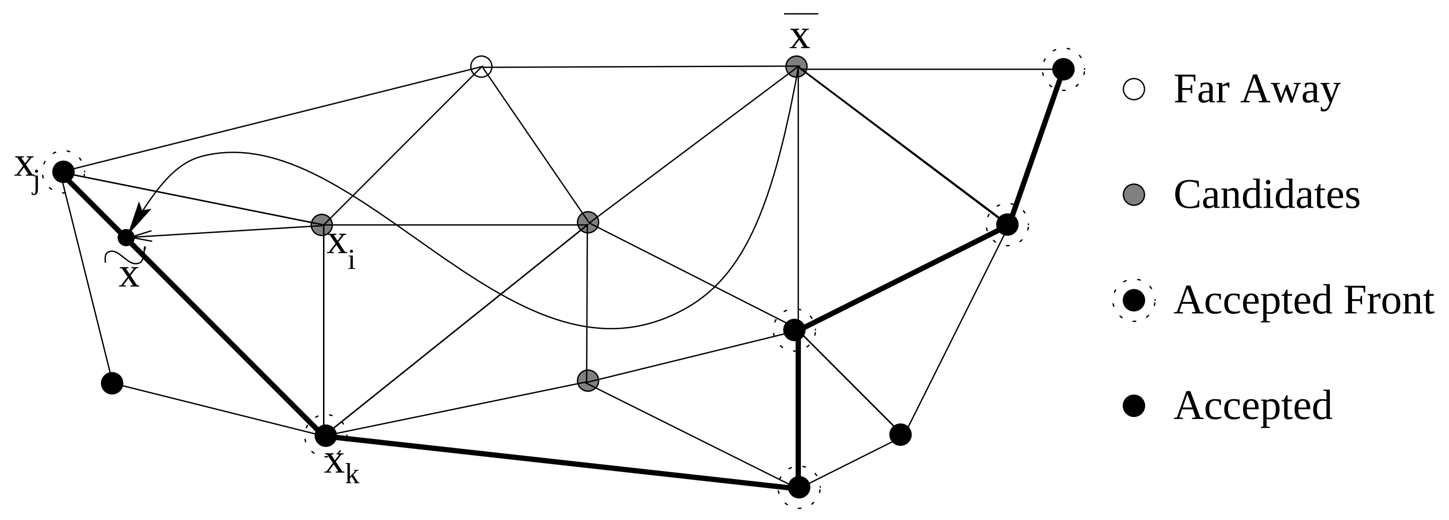
Local Anisotropy :

$$F_1(\boldsymbol{x}) = \min_{\boldsymbol{p}\in S_1} F(\boldsymbol{x},\boldsymbol{p}), \qquad F_2(\boldsymbol{x}) = \max_{\boldsymbol{p}\in S_1} F(\boldsymbol{x},\boldsymbol{p}), \qquad \Upsilon(\boldsymbol{x}) = \frac{F_2(\boldsymbol{x})}{F_1(\boldsymbol{x})}.$$

Generally, $\sup_{\boldsymbol{x}\in\Omega} \Upsilon(\boldsymbol{x}) << \Upsilon$, **for inhomogeneous problems.**

Further improvement: Use $\tilde{\Upsilon}(\boldsymbol{x}) = \frac{F_2(\boldsymbol{x})}{F(\boldsymbol{x},\boldsymbol{n})}$, where $\boldsymbol{n}$ is computed at run time.

OUMs for Anisotropic Optimal Trajectories.

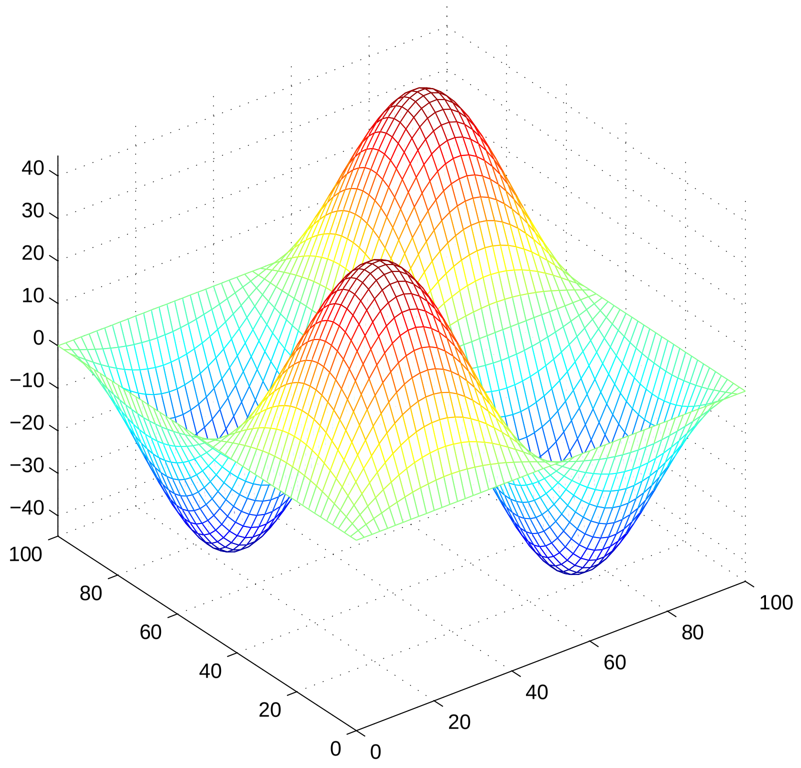


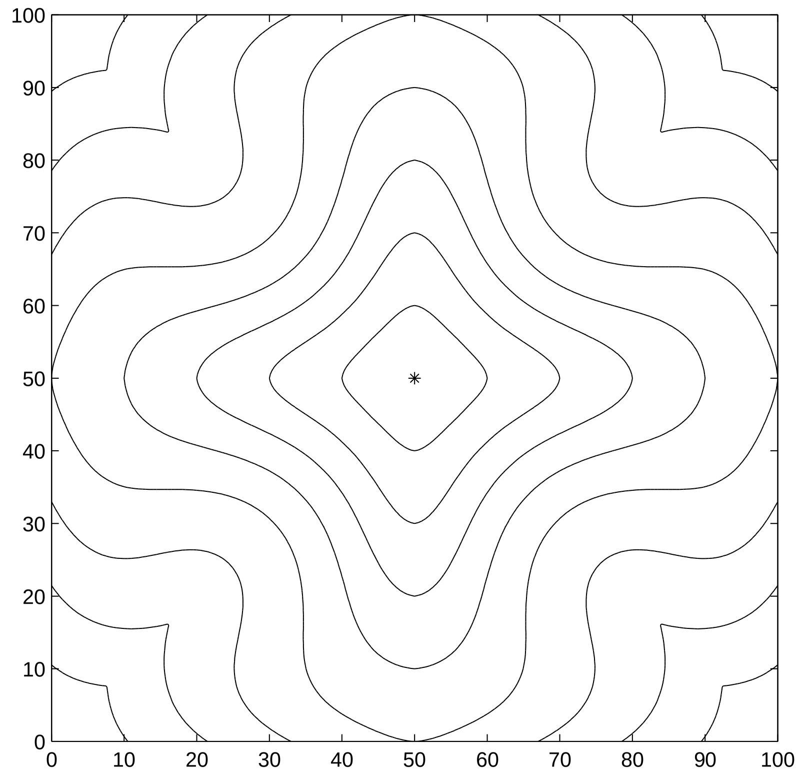
General OUM (Sethian and AV, 2001)

- non-iterative (complexity $O(\Upsilon M \log M)$ on a fixed grid);
- order of convergence depends on a particular simplex-update formula;
- speed up through localizing Υ;
- no extra cost for high inhomogeneity of f;
- extended to hybrid control & a class of quasi-linear PDEs;
- relaxation of small-time-controllability condition.

Traveling on a surface:

$$g(x,y) = 45\, sin(\frac{\pi x}{50}) sin(\frac{\pi y}{50}) \qquad (x,y) \in [0,100] \times [0,100]$$

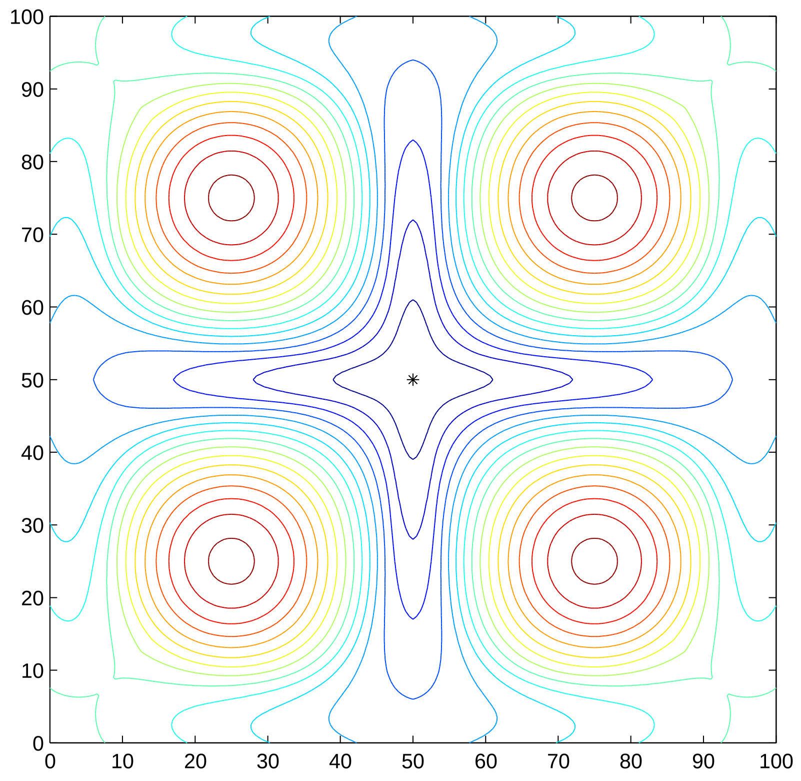


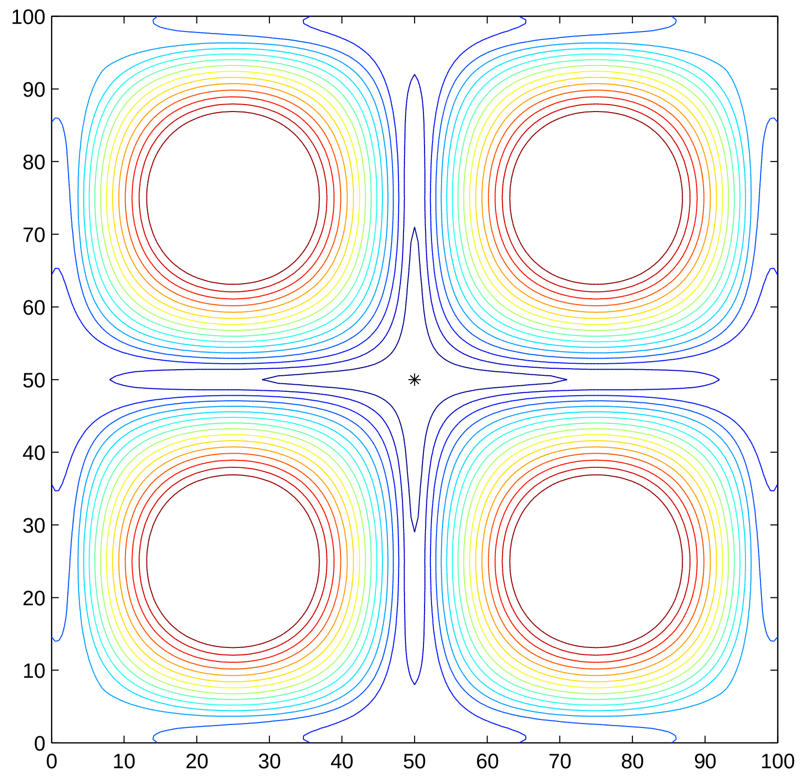


$$f(x,y,\boldsymbol{a}) = \phi(\theta_{\boldsymbol{a}}) \left(1 + (\nabla g(x,y) \cdot \boldsymbol{a})^2\right)^{-\frac{1}{2}} .$$

$\theta_{\boldsymbol{a}}$ - inclination, given the choice of direction $\boldsymbol{a}$ in the plane.

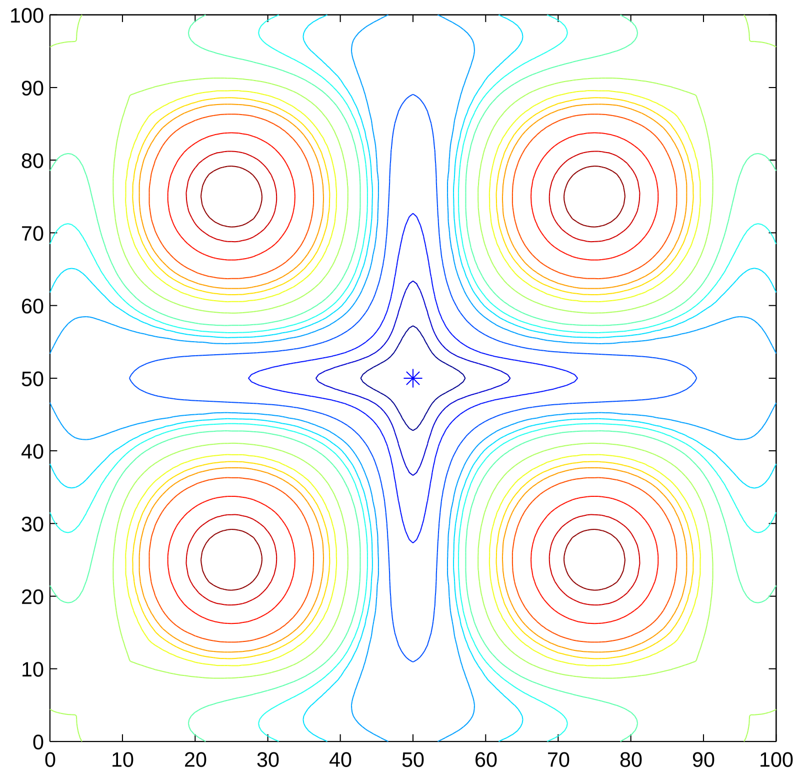
Anisotropic Dynamics: walking VS. skating.

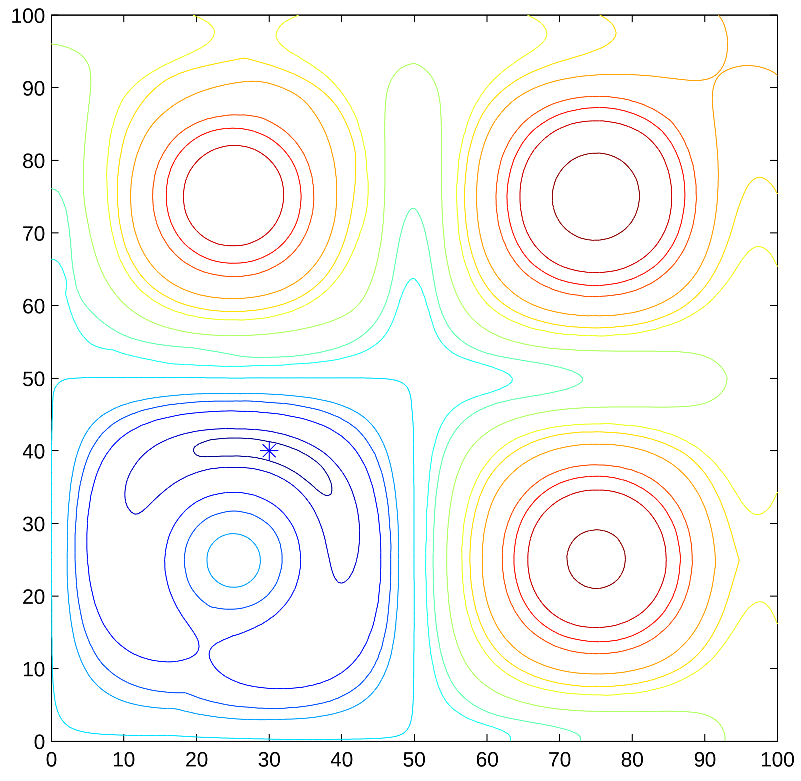




$\phi_w(\theta) = \sin^6(\theta) + 0.1,$ $\qquad$ $\phi_s(\theta) = 2\sin^{40}(\theta) + 0.1$

Walking & mood swings (weather changes in SF?).





$$f_{wm}(\boldsymbol{y}, \boldsymbol{a}, t) = f_w(\boldsymbol{y}, \boldsymbol{a})\psi(t) \qquad \left(\text{e.g., } \psi(t) = 1 + \frac{1}{2}\sin\left(\frac{t\,\pi}{20}\right)\right)$$

$$\max_{\boldsymbol{a}\in S_1}\{(\nabla u(\boldsymbol{x})\cdot\boldsymbol{a})f(\boldsymbol{x},\boldsymbol{a},u(\boldsymbol{x}))\} = 1$$

OUM for Hybrid Optimal Control.

Discrete links (change in continuous state, change in dynamics) :

$$L_{\text{to}}(\boldsymbol{x_i}) = \{ \text{ nodes, to which there is a discrete transition from } \boldsymbol{x_i} \}$$

Assumption : cardinality of $L_{\text{to}}(\boldsymbol{x_i})$ is uniformly bounded by some constant d.

Discretized equation for the value function :

$$U(\boldsymbol{x_i}) \;=\; \min \left\{ \min_{s \in S(\boldsymbol{x_i})} V_s(\boldsymbol{x_i}), \; \min_{\boldsymbol{x_r} \in L_{\text{to}}(\boldsymbol{x_i})} \{U_r + C_{ir}\} \right\}$$

Merging discrete and continuous *causality principles* :

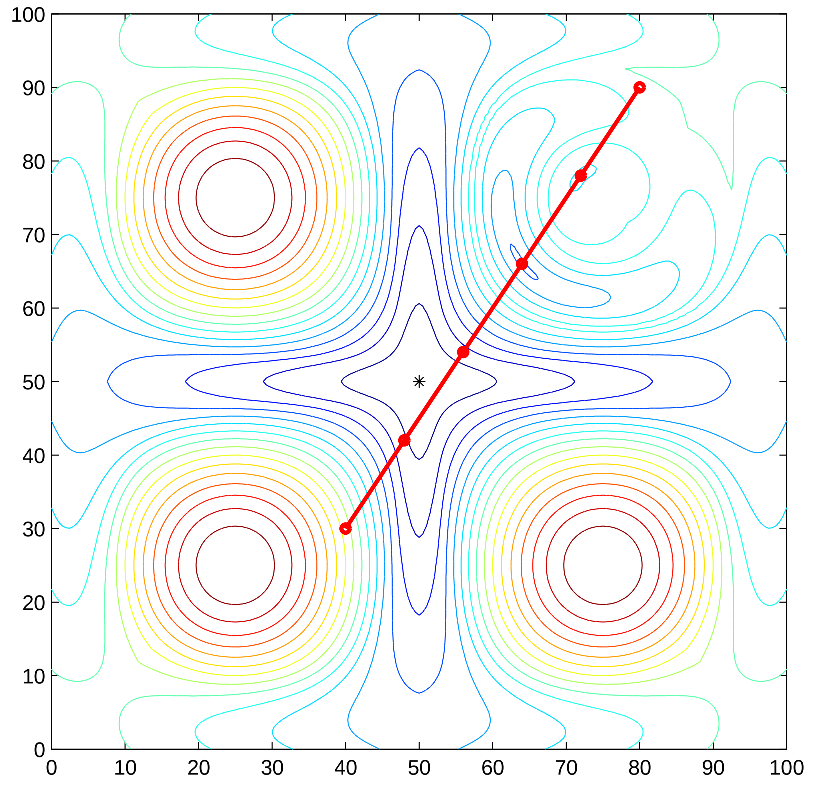
$$LA_{\text{to}}(\boldsymbol{x_i}) = L_{\text{to}}(\boldsymbol{x_i}) \bigcap Accepted$$

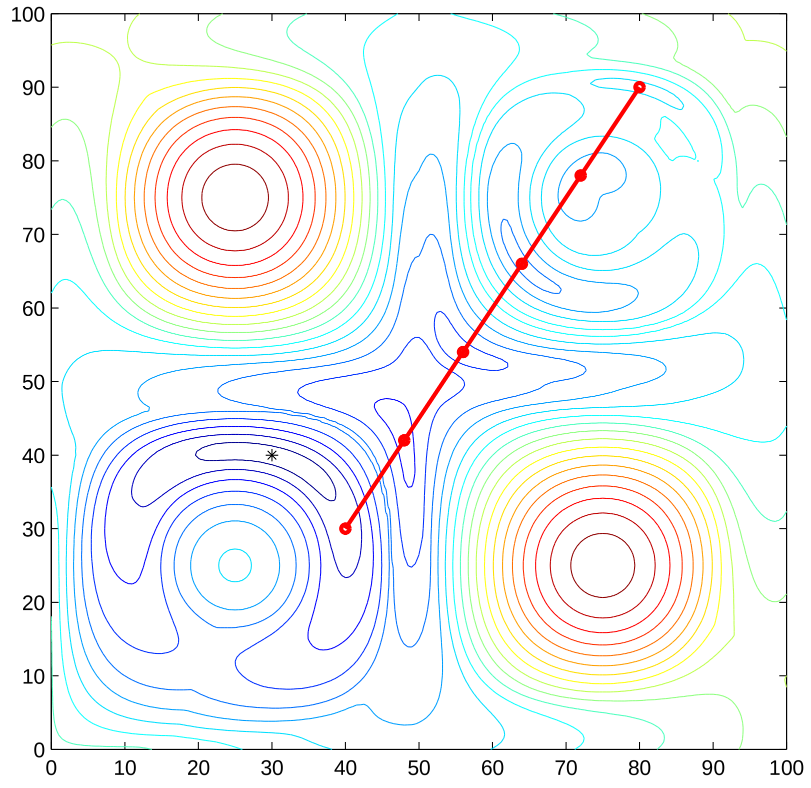
The resulting update formula :

$$U(\boldsymbol{x_i}) \;=\; \min \left\{ \min_{(\boldsymbol{x_j}, \boldsymbol{x_k}) \in NF(\boldsymbol{x_i})} V_{j,k}(\boldsymbol{x_i}), \; \min_{\boldsymbol{x_r} \in LA_{\text{to}}(\boldsymbol{x_i})} \{U_r + C_{ir}\} \right\}$$

The resulting computational complexity : $O((\Upsilon + d) M \log M)$ on a fixed grid.

Hybrid Dynamics: walking AND/OR catching a bus.

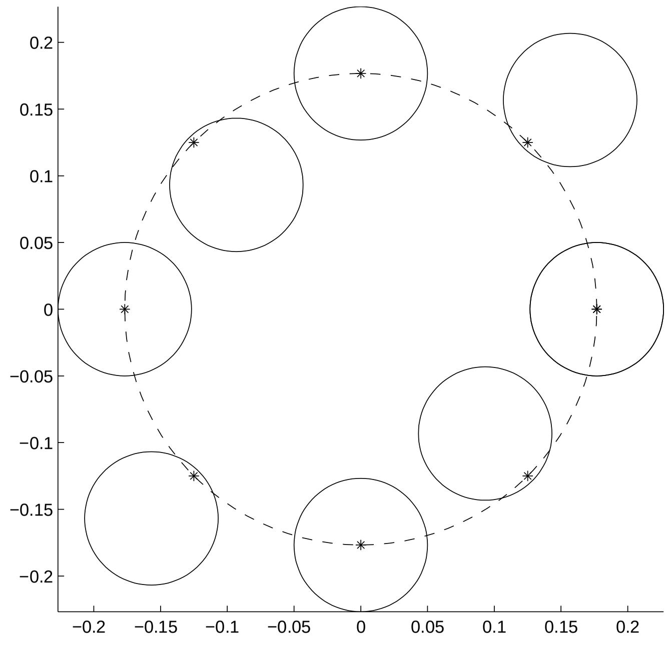


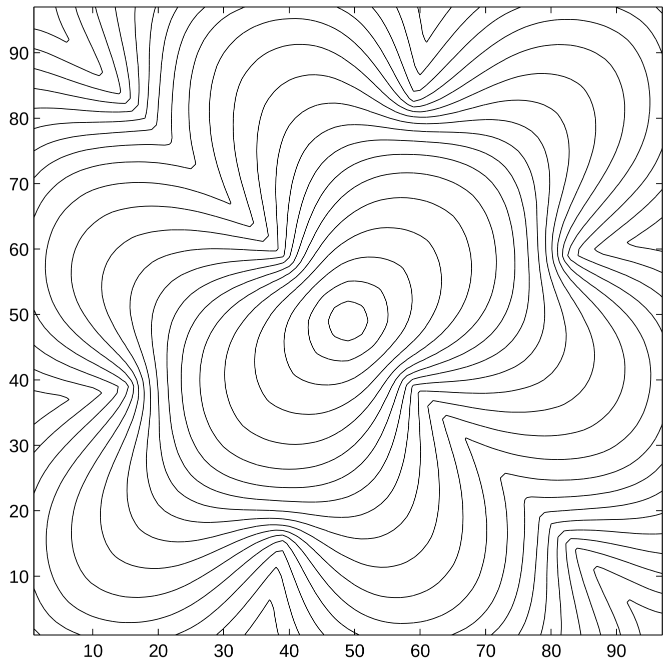


Struggling with a “flow”.

Robot’s dynamics: $$\frac{d\boldsymbol{z}}{dt}(t) \;=\; \tilde{\boldsymbol{a}}(t) + \boldsymbol{b}(\boldsymbol{z}(t)), \qquad \|\tilde{\boldsymbol{a}}(t)\| = 1.$$

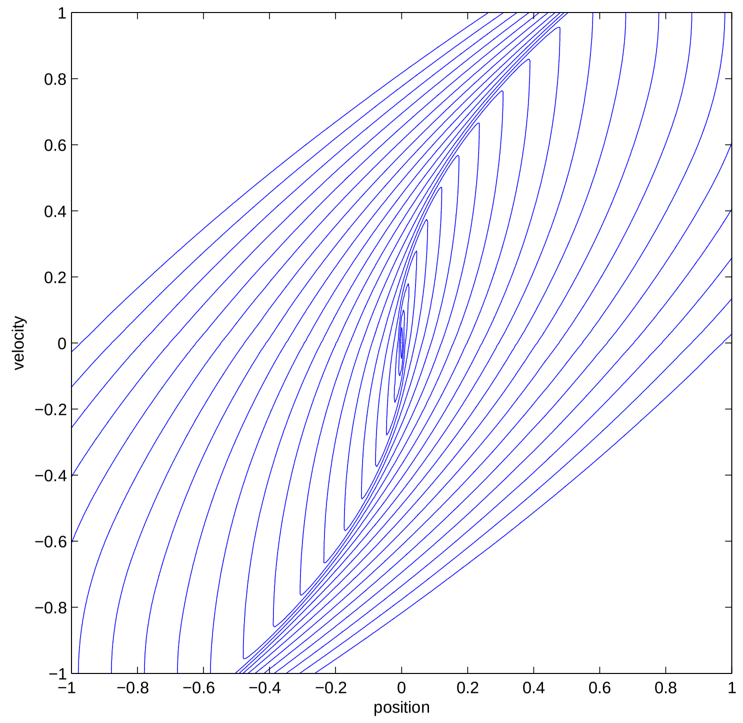
“Flow”: $$\boldsymbol{b}(x,y) = \frac{-.9\sin(4\pi x)\sin(4\pi y)}{\sqrt{x^2+y^2}} \begin{bmatrix} x \\ y \end{bmatrix}. \qquad \Upsilon = 19.$$

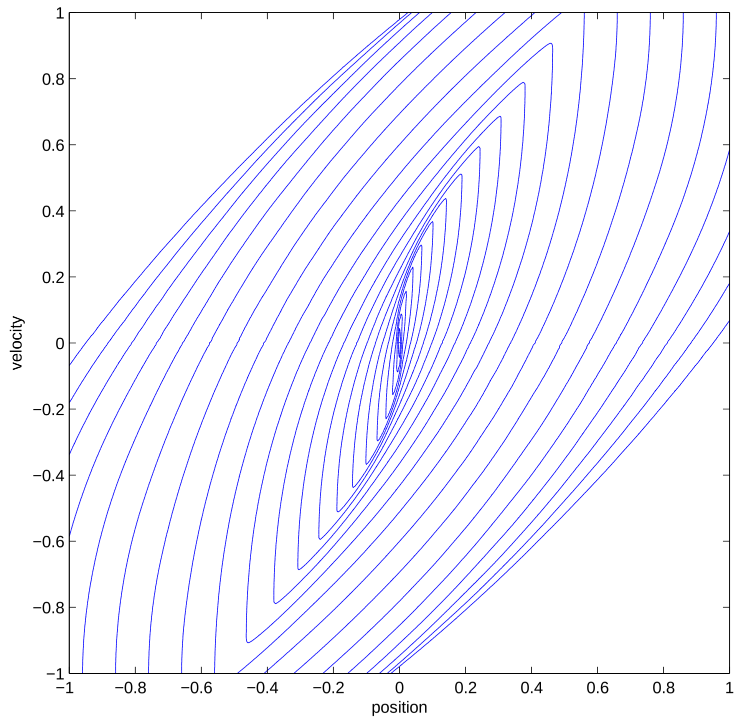




Time-reversed double-integrator:

$$y_1'(t) = y_2(t); \qquad y_2'(t) = a(t); \qquad a(t) \in [-\phi(t), \phi(t)].$$





$\phi(t) = 1$

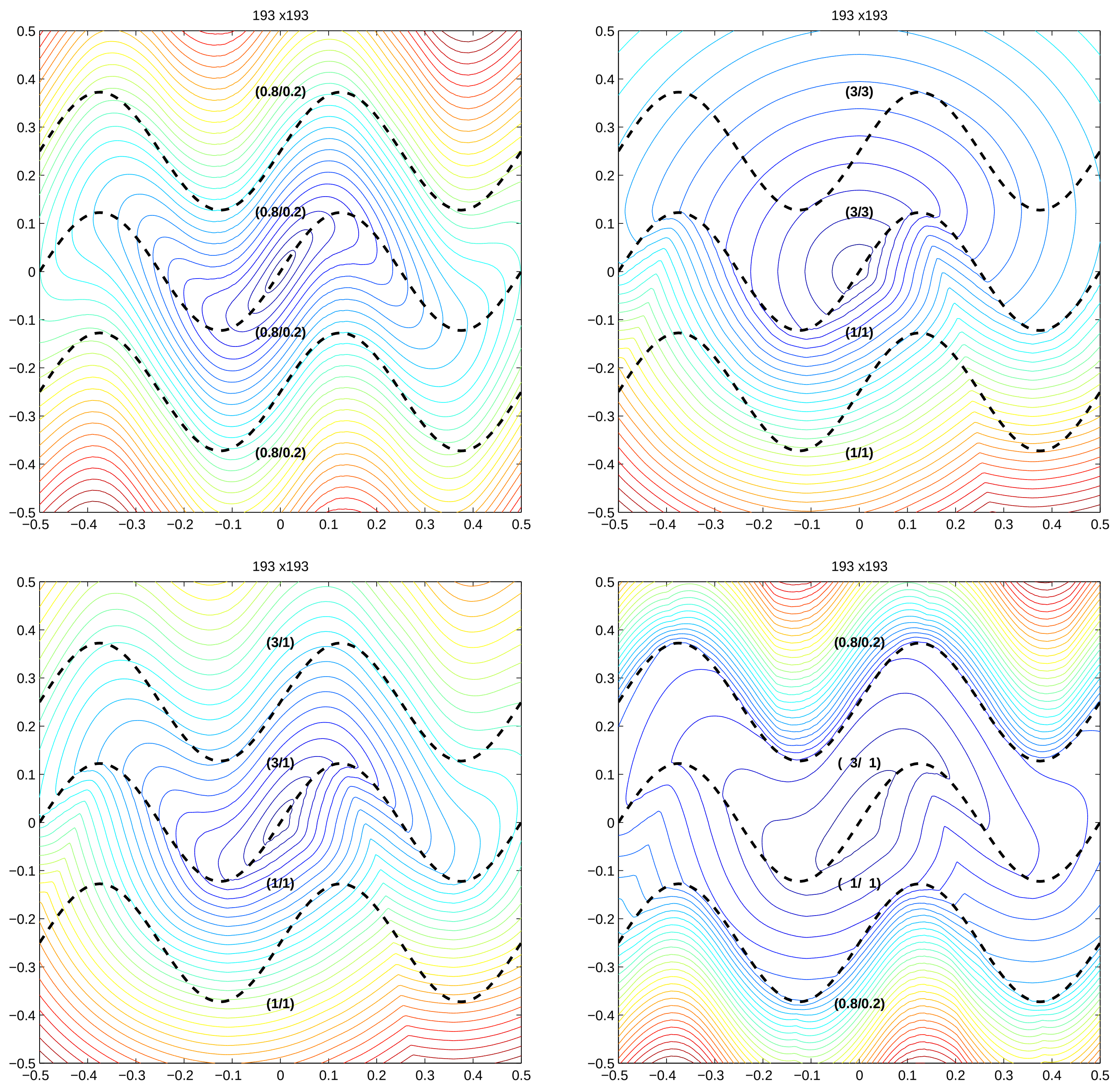
$\phi(t) = 1 + \frac{1}{2}\sin(\pi t)$

Seismic Imaging: multi-layer “model”.

- **Computational domain:** a square $[-a, a] \times [-a, a]$, split into n layers by the curves $y_i(x) = C(x) + b_i$.

$$C(x) = A \sin\left(\frac{r\pi x}{a} + \beta\right)$$

- A is the amplitude of the sinusoidal profile;
- r is the number of periods;
- β is the phase offset.
- In each layer, the anisotropic speed profile S_f is given at every point (x, y) by an ellipse with the bigger axis (of length $2F_2$) tangential to the curve $C(x)$ and the smaller axis (of length $2F_1$) normal to the curve.
- F_1 and F_2 are constants in each layer.


193 x193
(0.8/0.2)
(0.8/0.2)
(0.8/0.2)
(0.8/0.2)
193 x193
(3/3)
(3/3)
(1/1)
(1/1)
193 x193
(3/1)
(3/1)
(1/1)
(1/1)
193 x193
(0.8/0.2)
(3/ 1)
(1/ 1)
(0.8/0.2)
−0.5
−0.4
−0.3
−0.2
−0.1
0
0.1
0.2
0.3
0.4
0.5

… & the last slide.

Causality in the original problem should always be used to build efficient computational methods.

- Discretizations sometimes have worse causal properties than the original PDEs.
- Explicit causality is the easiest to exploit.
- But raising dimensionality just to ensure (explicit) causality is rarely worthwhile. It is typically much better to use non-explicit causality in a lower-dimensional domain.
- Label-Setting type methods can be built to exploit monotone causality.
- Label-Correcting type methods aim to do the same, but can speed up convergence even in some not-fully-causal problems.
- Carefully selected benchmarks and tests are needed for a comprehensive comparison of various “Fast” Methods.