The Flexible, Extensible and Efficient Toolbox of Level Set Methods

Ian Mitchell
Department of Computer Science
University of British Columbia

http://www.cs.ubc.ca/~mitchell
mitchell@cs.ubc.ca

research supported by
the Natural Science and Engineering Research Council of Canada

Level Set Methods

- Numerical algorithms for dynamic implicit surfaces and time-dependent Hamilton-Jacobi / degenerate parabolic partial differential equations
- Applications in
  - Graphics, Computational Geometry and Mesh Generation
  - Differential Games
  - Financial Mathematics and Stochastic Differential Equations
  - Fluid and Combustion Simulation
  - Image Processing and Computer Vision
  - Robotics, Control and Dynamic Programming
  - Verification and Reachable Sets

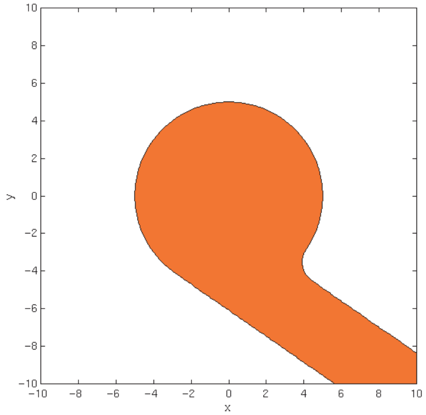
Implicit Surface Functions

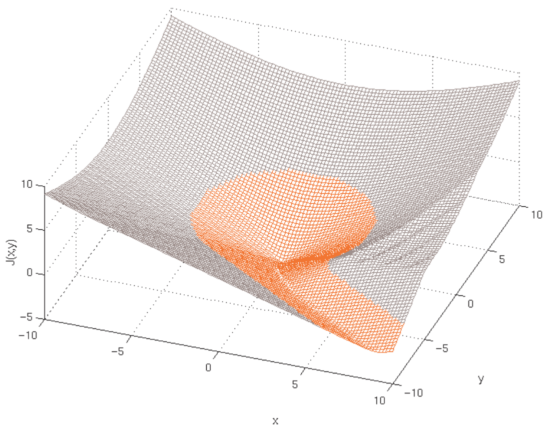
- Surface $\mathcal{S}(t)$ and/or set $\mathcal{G}(t)$ are defined implicitly by an isosurface of a scalar function $\varphi(t,x)$, with several benefits
  - State space dimension does not matter conceptually
  - Surfaces automatically merge and/or separate
  - Geometric quantities are easy to calculate

$$\varphi : \mathbb{R} \times \Omega \rightarrow \mathbb{R} \qquad \Omega \subseteq \mathbb{R}^d$$

$$G(t) = \{x \in \Omega \mid \varphi(t,x) \leq 0\}$$

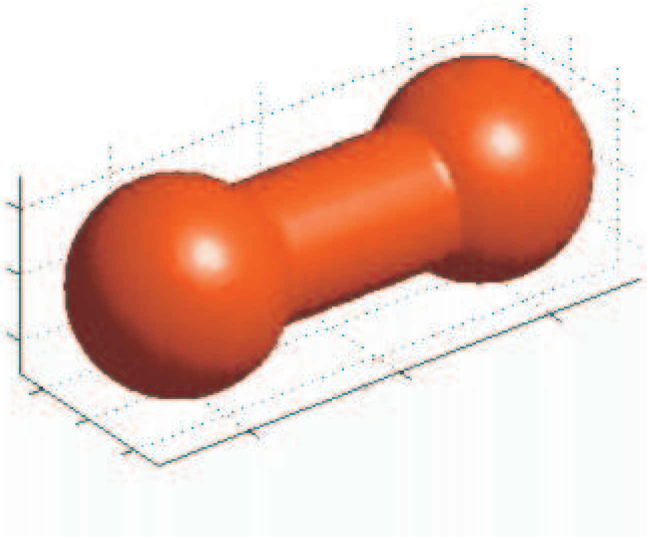
$$S(t) = \partial G(t) = \{x \in \Omega \mid \varphi(t,x) = 0\}$$



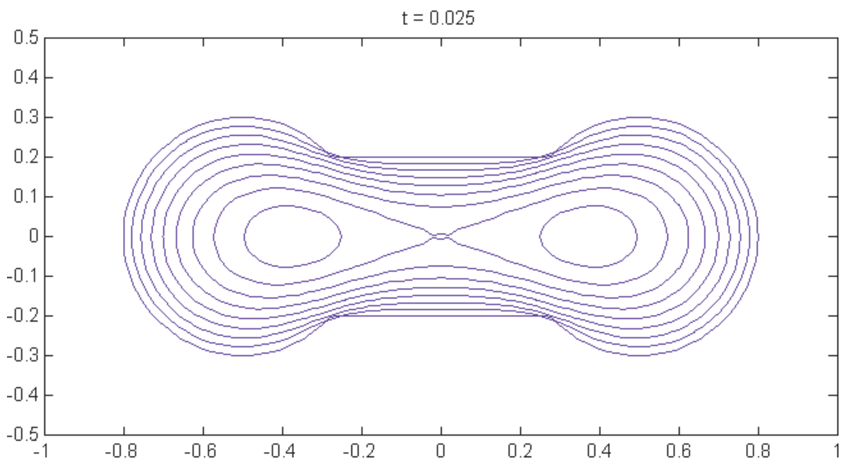


Implicit Surface Benefits

- Easy to represent a variety of shapes
- Unified framework for many types of motion
- Surface parameters easily approximated
- Topological changes are automatic
- Conceptual complexity independent of dimension
- Easy to visualize
- Easy to implement (?)



shrinking dumbbell



contour slice through midplane

“Hamilton-Jacobi” Equations

$$D_t\varphi(t,x) + G(t,x,\varphi,D_x\varphi,D_x^2\varphi) = 0$$

$$\varphi(x,0) = g(x) \text{ bounded and continuous}$$

$$G(t,x,r,p,\mathbf{X}) \leq G(t,x,s,p,\mathbf{Y}), \text{ if } r \leq s \text{ and } \mathbf{Y} \leq \mathbf{X}$$

- Time-dependent partial differential equation (PDE)
  - With second derivative terms: degenerate hyperbolic PDE
- In general, classical solution will not exist
  - Viscosity solution φ will be continuous but not differentiable
- For example, classical Hamilton-Jacobi-Bellman
  - Finite horizon optimal cost leads to terminal value PDE

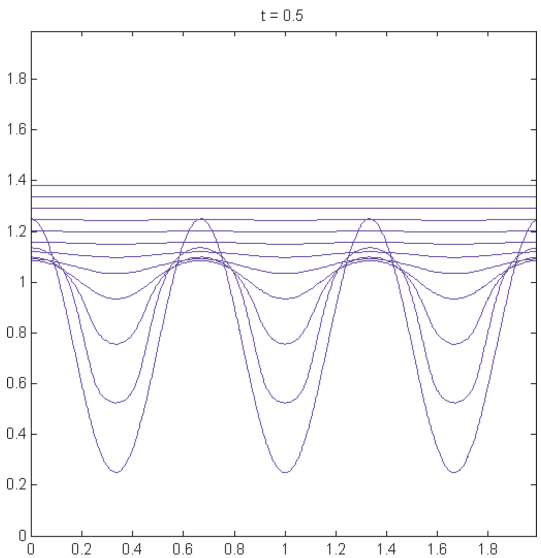
$$\varphi(t,x(t)) = \min_{u(\cdot)}\left[g(x(T)) + \int_t^T \ell(x(s),u(s))ds\right]$$

$$D_t\varphi(t,x) + \min_u\left[D_x\varphi(t,x)\cdot f(x,u) + \ell(x,u)\right] = 0$$

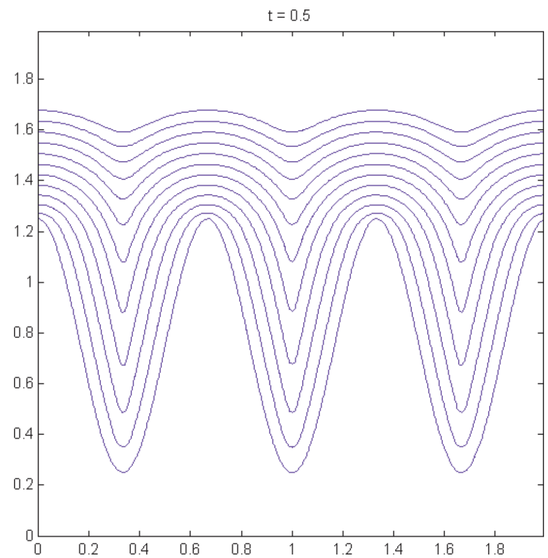
Viscosity Solution

- Well defined weak solution of HJ PDE
  - Limit of vanishing viscosity solution, where it exists
  - Kinks form where characteristics meet
- Example

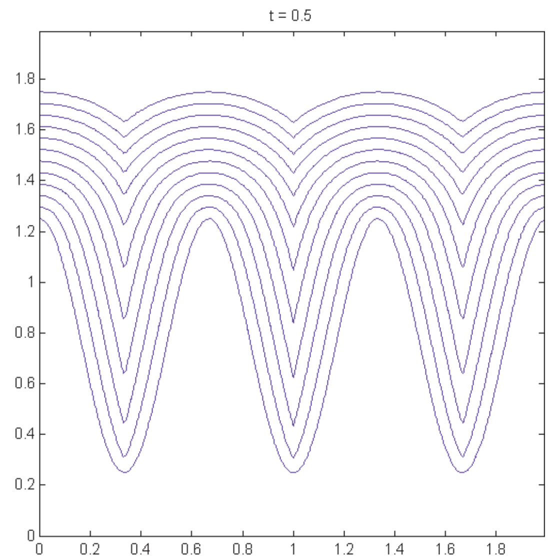
$$D_t\varphi(t,x) + (1 - b\kappa(t,x))\|D_x\varphi(t,x)\| = 0$$



$b = 0.25$



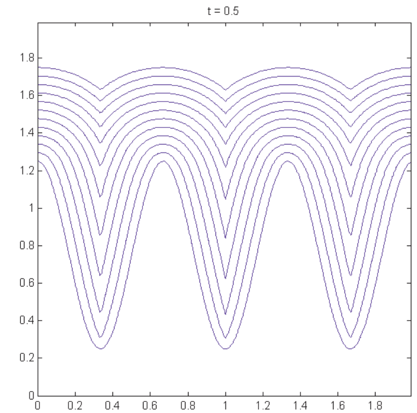
$b = 0.025$

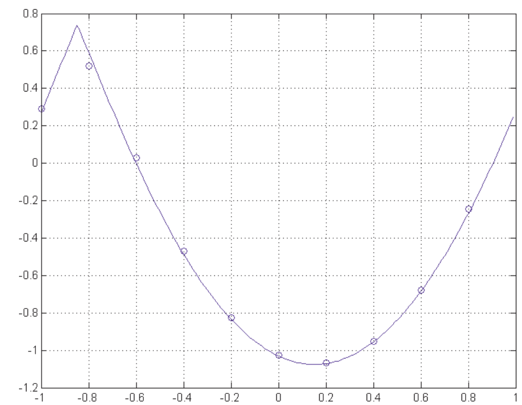


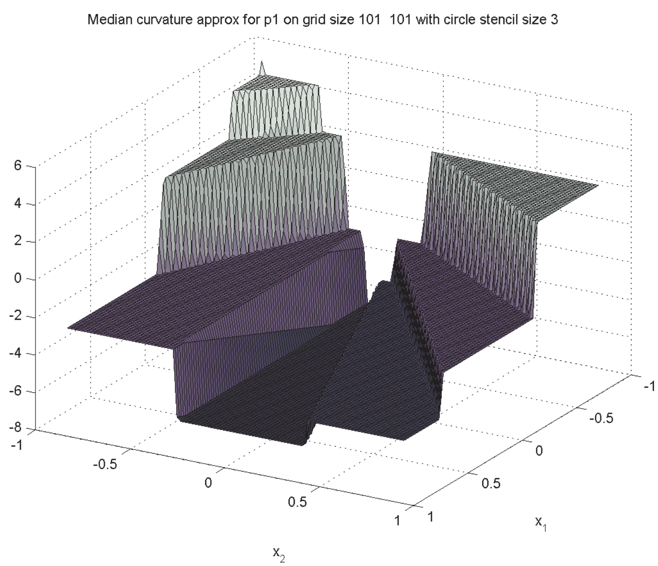
$b = 0$

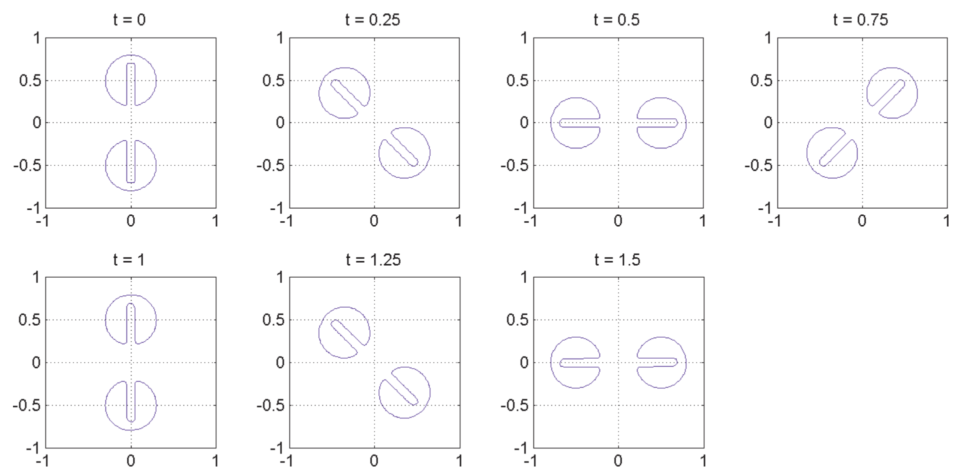
Outline

- Level set methods: dynamic implicit surfaces and the Hamilton-Jacobi equation
- Toolbox of level set methods: features and examples
- Adding schemes
  - How to achieve flexibility and efficiency
  - SSP RK integrators
  - Monotone motion by mean curvature









The Toolbox: What Is It?

- A collection of Matlab routines for level set methods
  - Fixed Cartesian grids
  - Arbitrary dimension (computationally limited)
  - Vectorized code achieves reasonable speed
  - Direct access to Matlab debugging and visualization
  - Source code is provided for all toolbox routines
- Underlying algorithms
  - Solve various forms of time-dependent Hamilton-Jacobi PDE
  - First and second spatial derivatives
  - First temporal derivatives
  - High order accurate finite difference approximation schemes
  - Explicit temporal integration
- Implements schemes from many sources
  - For citations, see the 140 page indexed user manual

The Toolbox: What Can It Do?

$0 = D_t\varphi(t,x)$	temporal derivative
$+\, v(t,x)\cdot D_x\varphi(t,x)$	convection
$+\, a(t,x)\|D_x\varphi(t,x)\|$	normal motion
$+\, \mathrm{sign}(\varphi(x,0))(\|D_x\varphi(t,x)\| - 1)$	reinitialization
$+\, H(t,x,\varphi,D_x\varphi)$	general HJ
$-\, b(t,x)\kappa(t,x)\|D_x\varphi(t,x)\|$	mean curvature
$-\, \mathrm{trace}[\sigma(t,x)\sigma^T(t,x)D_x^2\varphi(t,x)]$	stochastic DEs
$+\, \lambda(t,x)\varphi(t,x)$	discounting
$+\, F(t,x,\varphi)$,	forcing

$D_t\varphi(t,x) \geq 0$, $\quad D_t\varphi(t,x) \leq 0$, growth constraints

$\varphi(t,x) \leq \psi(t,x)$, $\quad \varphi(t,x) \geq \psi(t,x)$, masking constraints

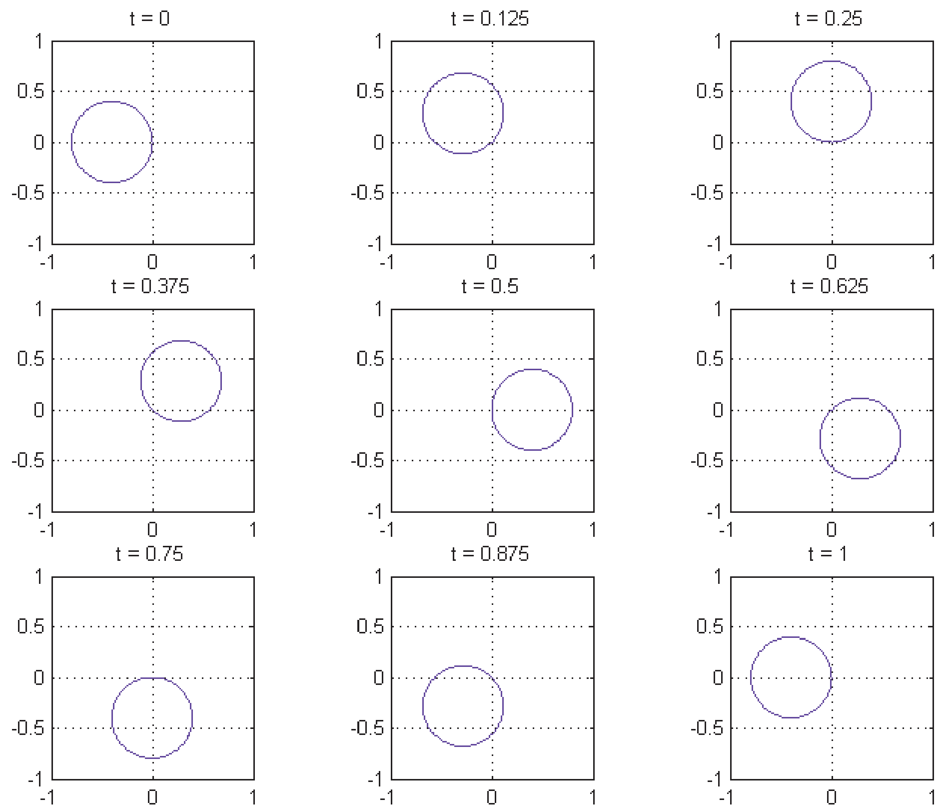
$\varphi : \mathbb{R} \times \Omega \to \mathbb{R}^k$ vector level sets

Convective Flow

- Motion by externally generated velocity field

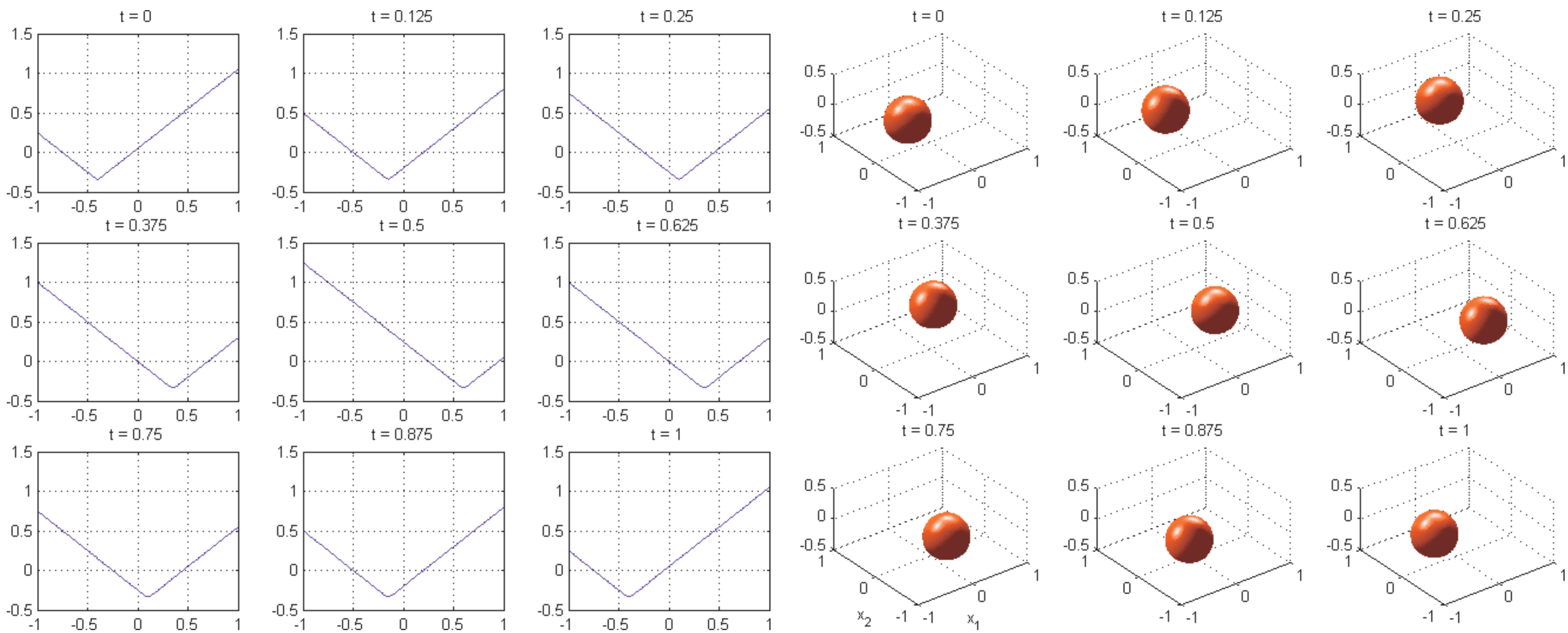
$$D_t\varphi(t,x) + v(t,x)\cdot D_x\varphi(t,x) = 0$$

- Example: rigid body rotation about the origin



Dimensionally Flexible

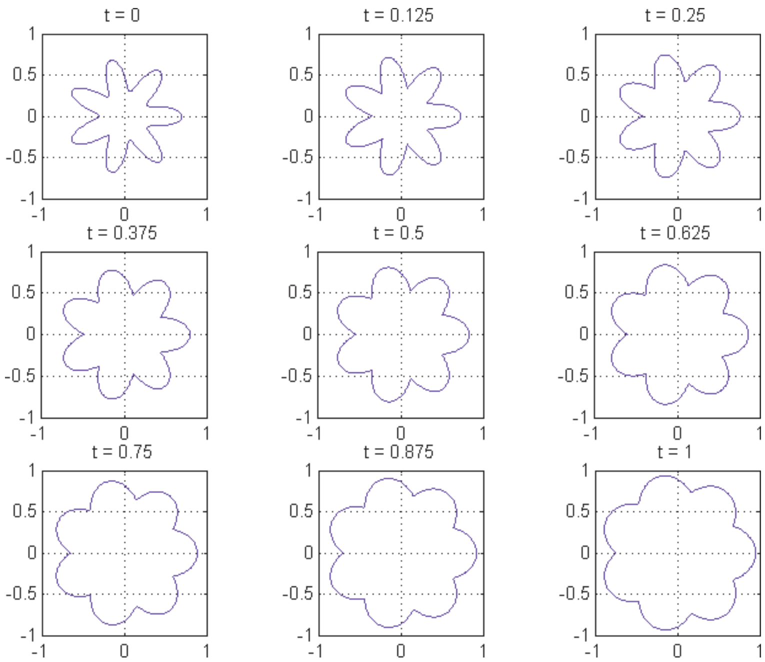
- Core code is dimensionally independent
  - Cost in memory and computation is exponential
  - Visualization in dimensions four and above is challenging
  - Dimensions one to three are quite feasible



Motion in the Normal Direction

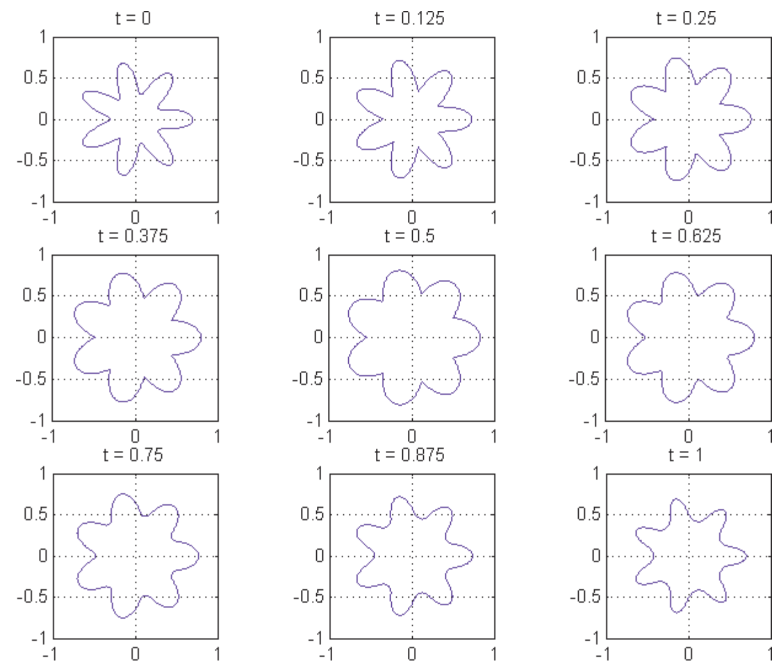
- Motion by externally generated speed function

$$D_t\varphi(t,x) + a(t,x)\|D_x\varphi(t,x)\| = 0$$



constant outward speed

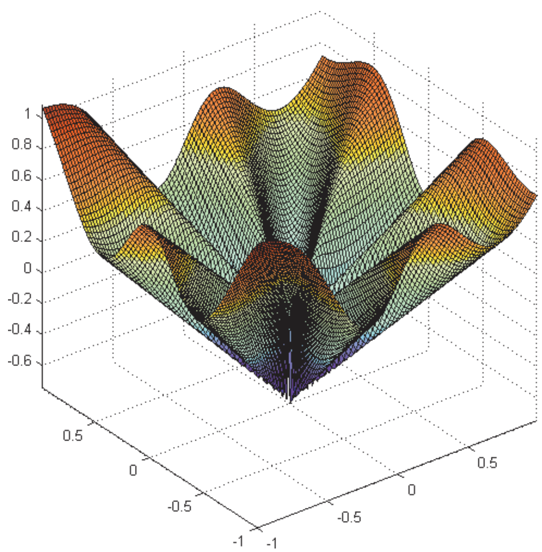
constant speed switches direction outward at first, inward thereafter



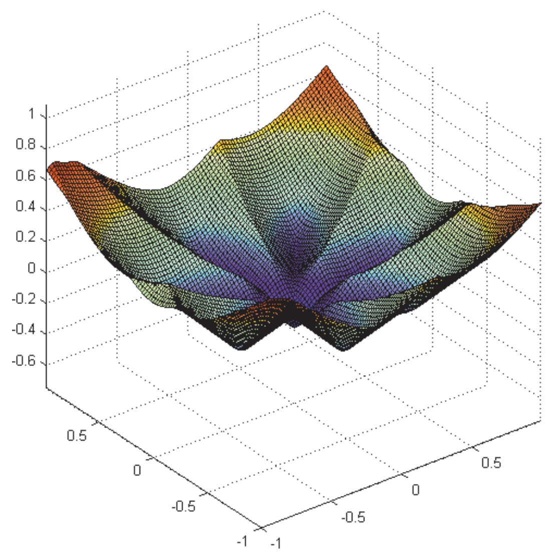
Reinitialization Equation

- Returning the gradient to unit magnitude

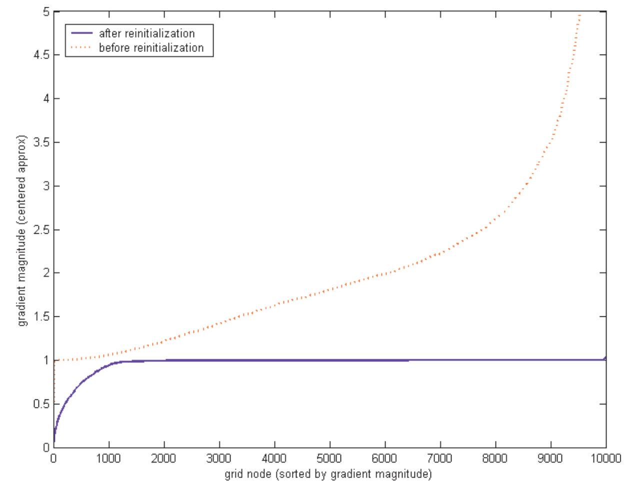
$$D_t\varphi(t,x) + \mathrm{sign}(\varphi(0,x))(\|D_x\varphi(t,x)\| - 1) = 0$$



initial implicit surface function



reinitialized to signed distance



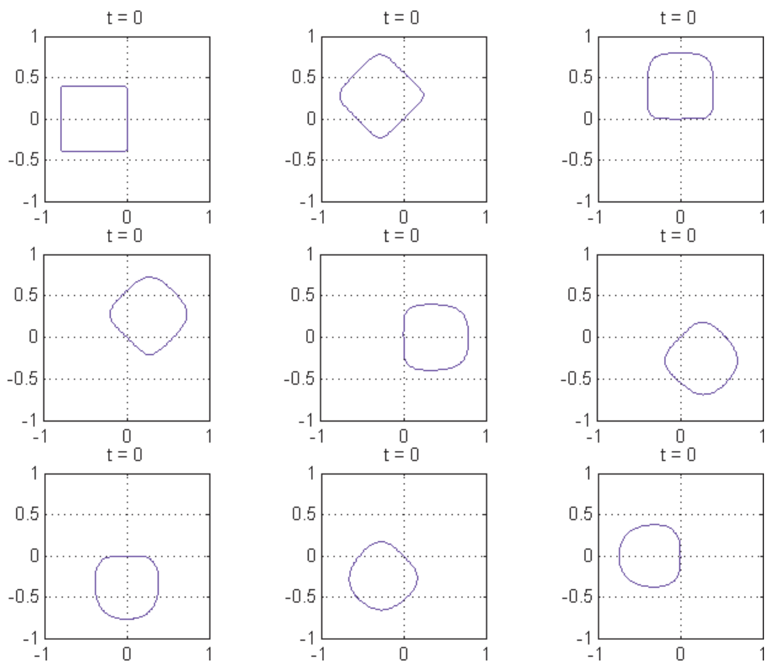
compare gradient magnitudes

General Hamilton-Jacobi

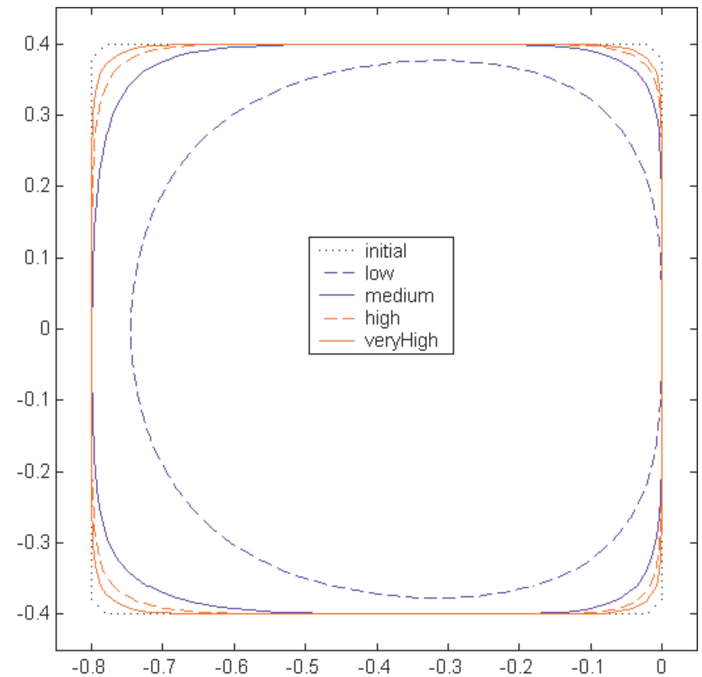
- Motion may depend nonlinearly on gradient

$$D_t\varphi(t,x) + H(t,x,D_x\varphi(t,x)) = 0$$

- Example: rigid body rotation about the origin



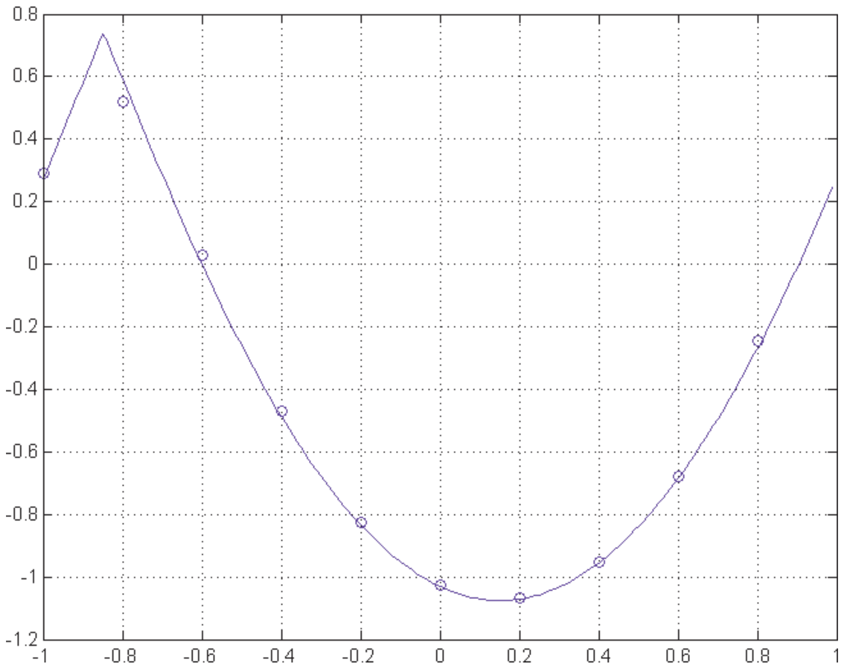
rotate a square once around



compare errors of various schemes

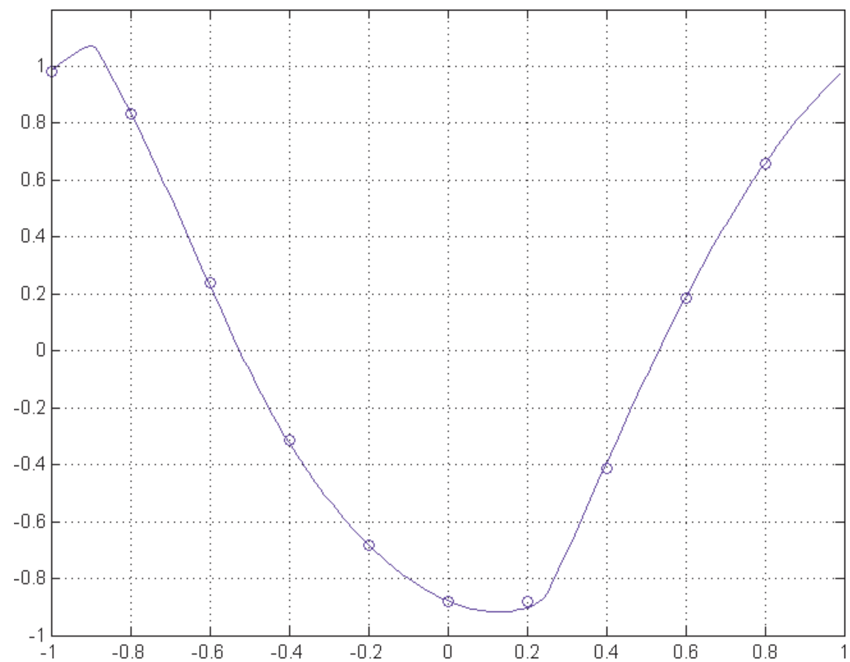
General Hamilton-Jacobi

$$D_t\varphi(t,x) + H(t,x,D_x\varphi(t,x)) = 0$$



$$H(t,x,p) = \frac{1}{2}\left(\alpha + \sum_i p_i\right)^2$$

Burgers' equation



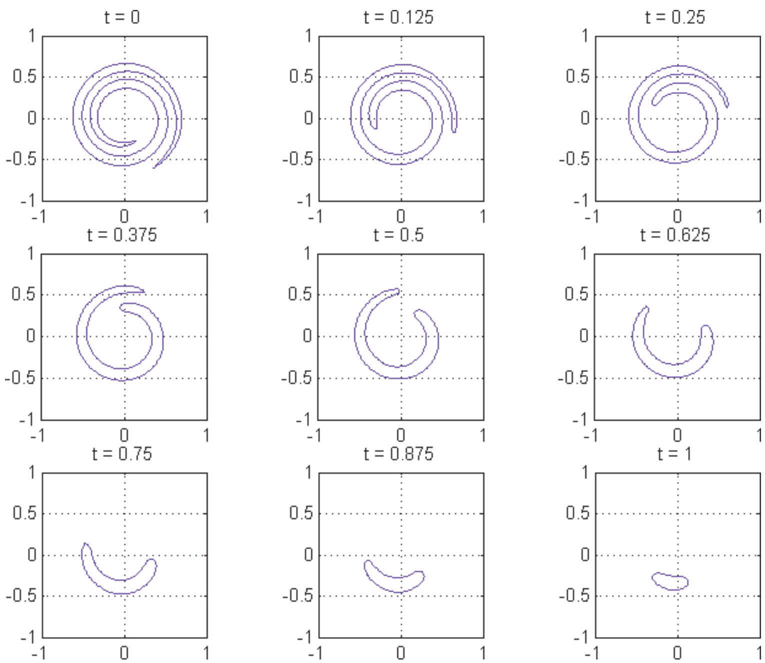
$$H(t,x,p) = -\cos\left(\alpha + \sum_i p_i\right)$$

Nonconvex Hamiltonian

Motion by Mean Curvature

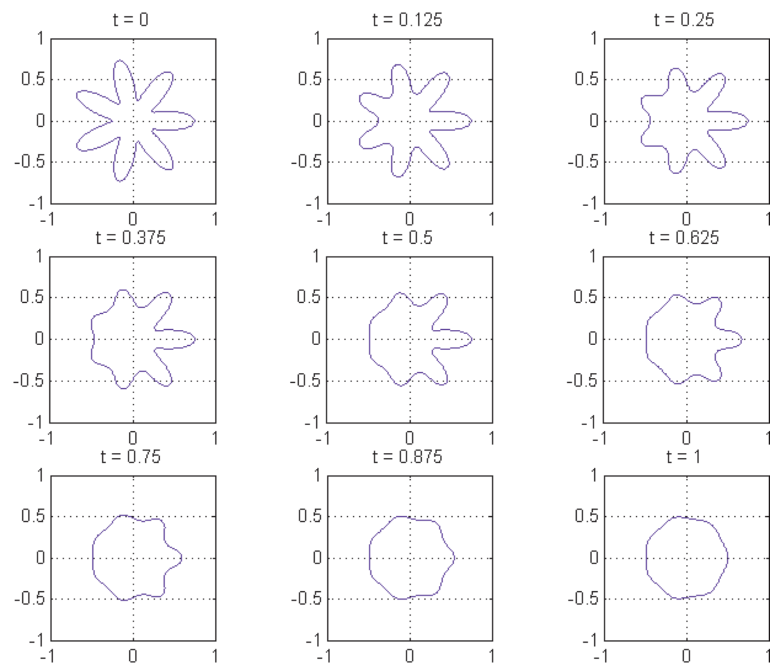
- Interface speed depends on its curvature κ

$$D_t\varphi(t,x) - b(t,x)\kappa(t,x)\|D_x\varphi(t,x)\| = 0$$



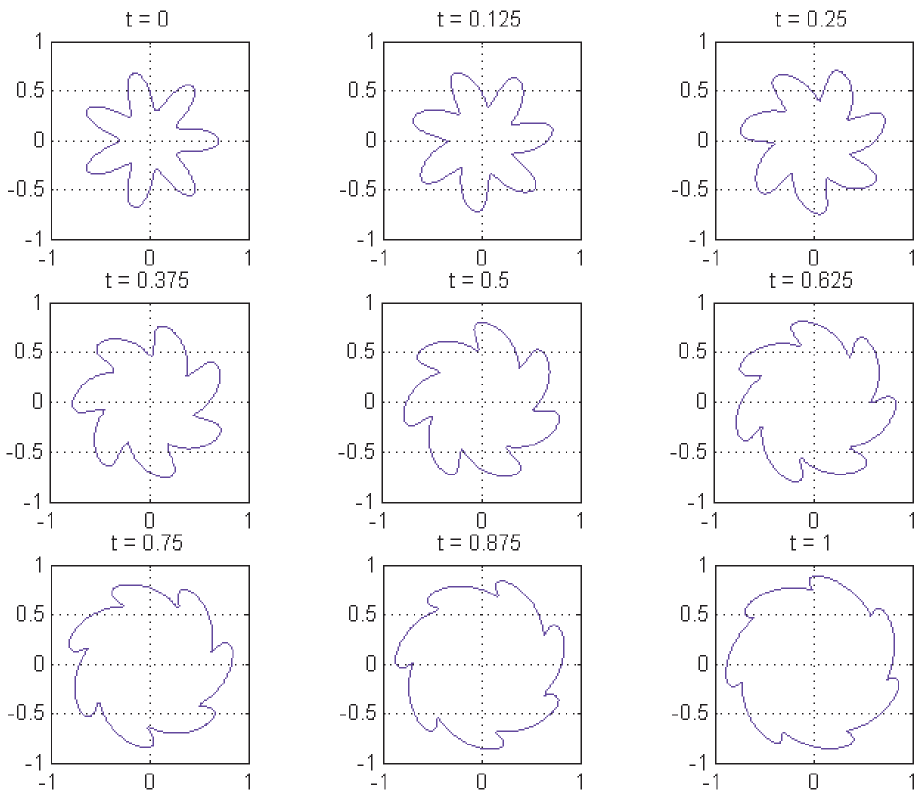
shrinking spiral

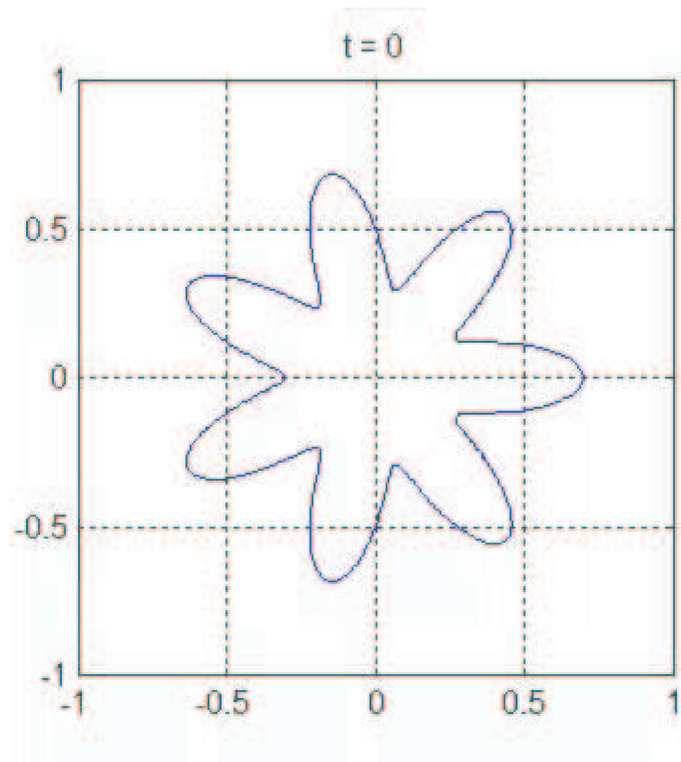
shrinking star
speed depends on time



Combining Terms

- Terms can be combined to generate complex but accurate motion
  – Example: rotation plus outward motion in normal direction

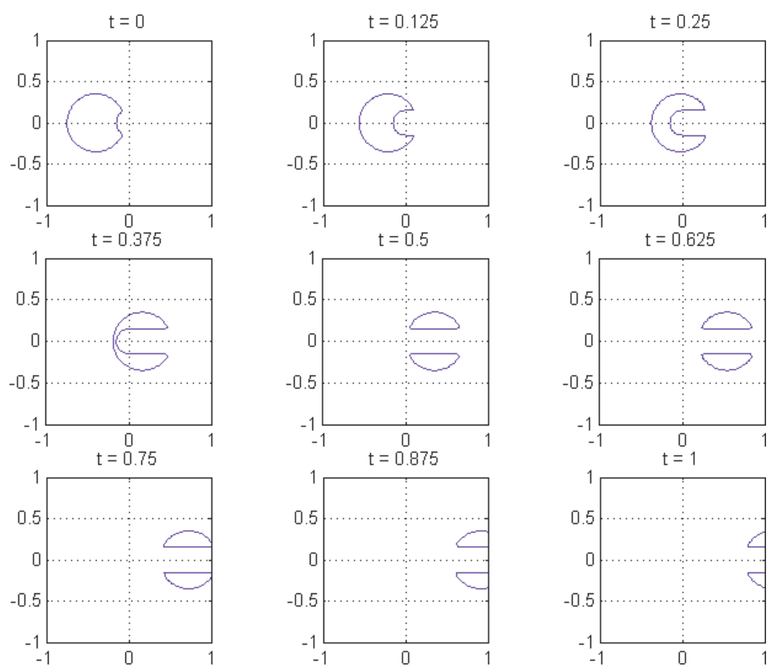




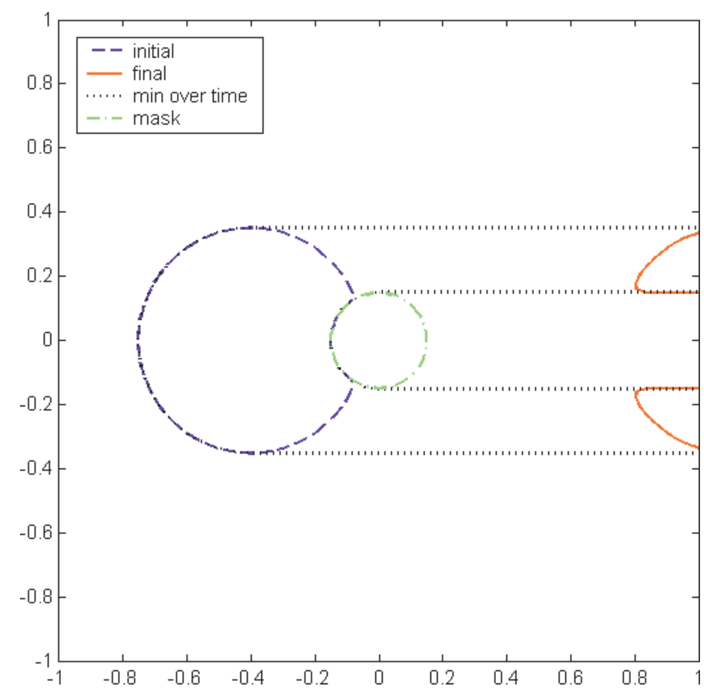
Constraints on Function Value

- Level set function constrained by user supplied implicit surface function
  - Example: masking a region of the state space

$$\varphi(t,x) \le \psi(x) \qquad \varphi(t,x) \ge \psi(x)$$



convective motion to the right



mask with small circle at origin

Constraints on Temporal Derivative

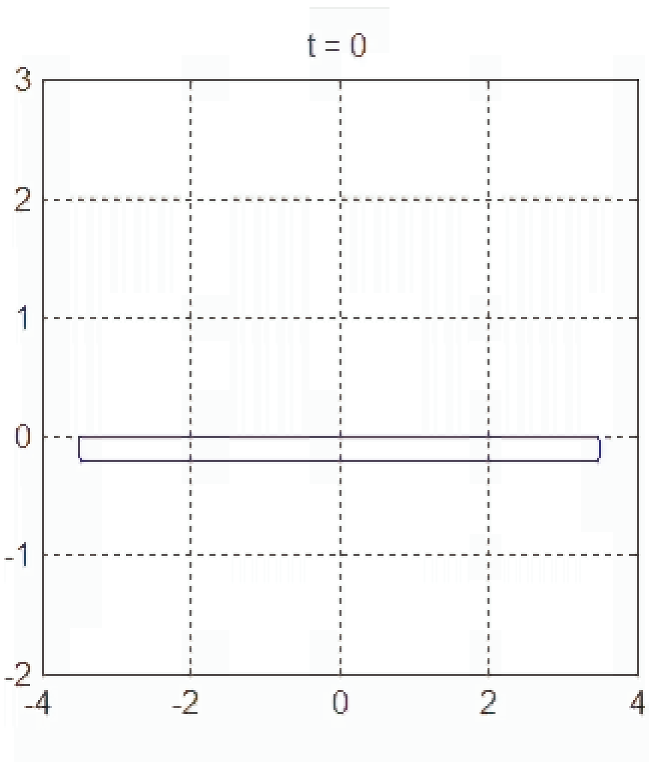
- Sign of temporal derivative controls whether implicit set can grow or shrink

$$D_t\varphi(t,x) \leq 0 \qquad D_t\varphi(t,x) \geq 0$$

- Example: reachable set only grows

$$\frac{d}{dt}\begin{bmatrix} x \\ y \end{bmatrix} = W_p \begin{bmatrix} 0 \\ -1 \end{bmatrix} + \frac{W_p}{R}\begin{bmatrix} y \\ -x \end{bmatrix} b + 2W_e \min\left(\sqrt{x^2+y^2}, S\right) a$$

$$a \in \mathbb{R}^2, \|a\| \leq 1$$
$$b \in [-1, +1]$$
$$W_p, W_e, R, S \text{ constant}$$



Stochastic Differential Equations (v1.1)

- Itô stochastic differential equation

$$dx(t) = f(t, x(t))dt + \sigma(t, x(t))dB(t)$$

- Kolmogorov or Fokker-Planck equation for expected outcome

$$D_t\varphi + f^T D_x\varphi - \tfrac{1}{2}\,\mathrm{trace}\left[\sigma\sigma^T D_x^2\varphi\right] = 0$$

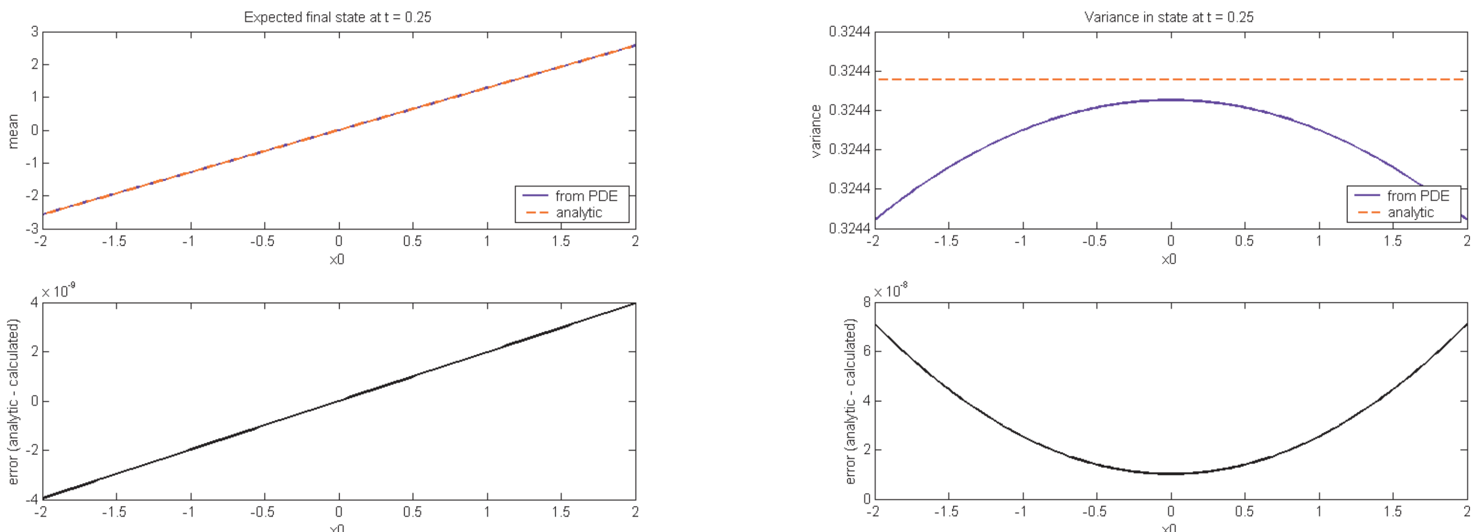
  – Example: linear DE with additive noise

$$f(x,t) = ax,\ \sigma(x,t) = b \text{ where } a = 1,\ b = 0.1$$

$t = 0.25$

value

error



final state mean

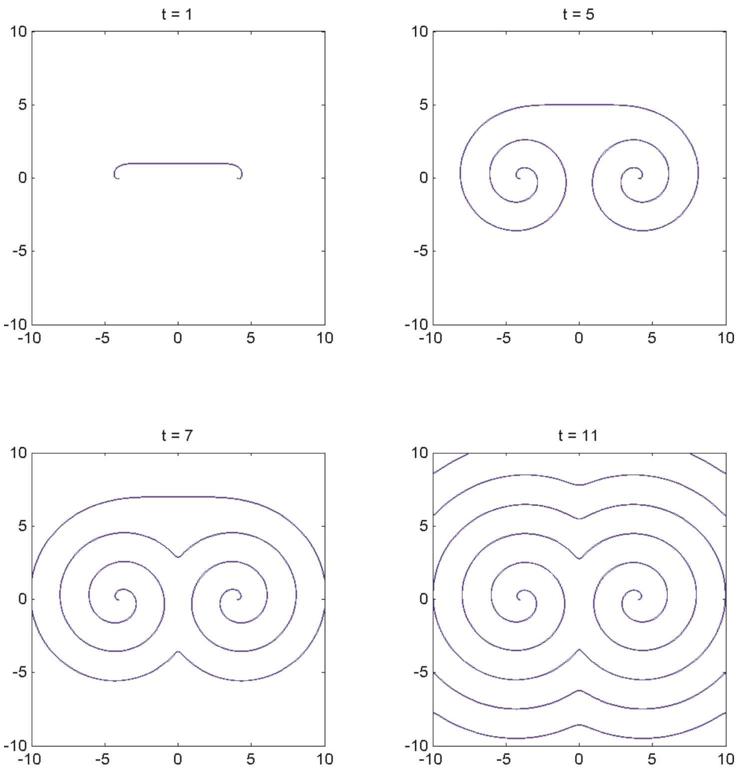
final state variance

Open Curves by Vector Level Sets (v1.1)

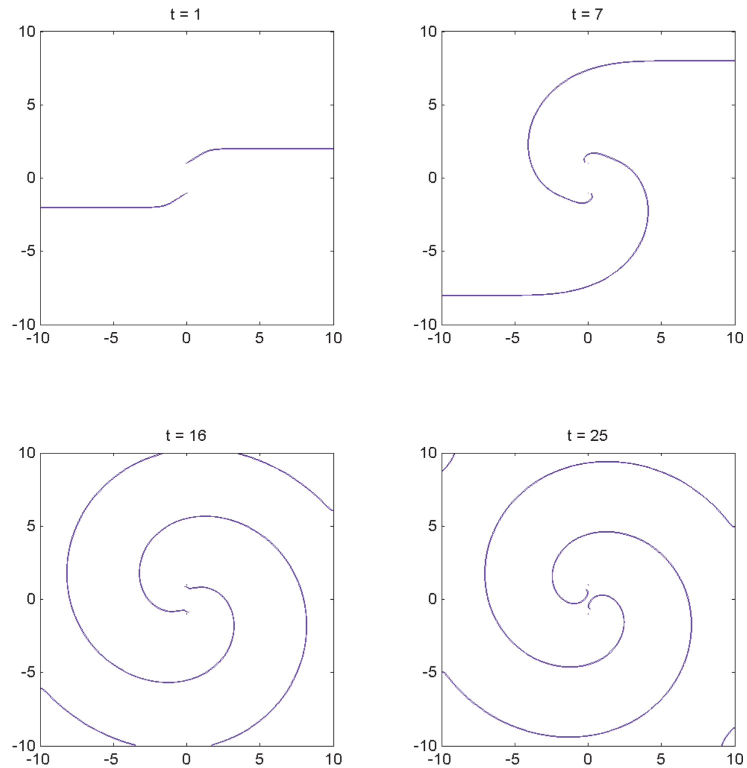
- Normal level set methods can only represent closed curves
- Evolve two level sets in unison to represent an open curve Γ

$$D_t\varphi_1 - \mathsf{sign}(\varphi_2)\,[\lambda\,\mathsf{sign}(\varphi_2)\kappa(\varphi_1) - 1]\,\|D_x\varphi_1\| = 0$$

$$D_t\varphi_2 - \mathsf{sign}(\varphi_1)\,[\lambda\,\mathsf{sign}(\varphi_1)\kappa(\varphi_2) + 1]\,\|D_x\varphi_2\| = 0$$

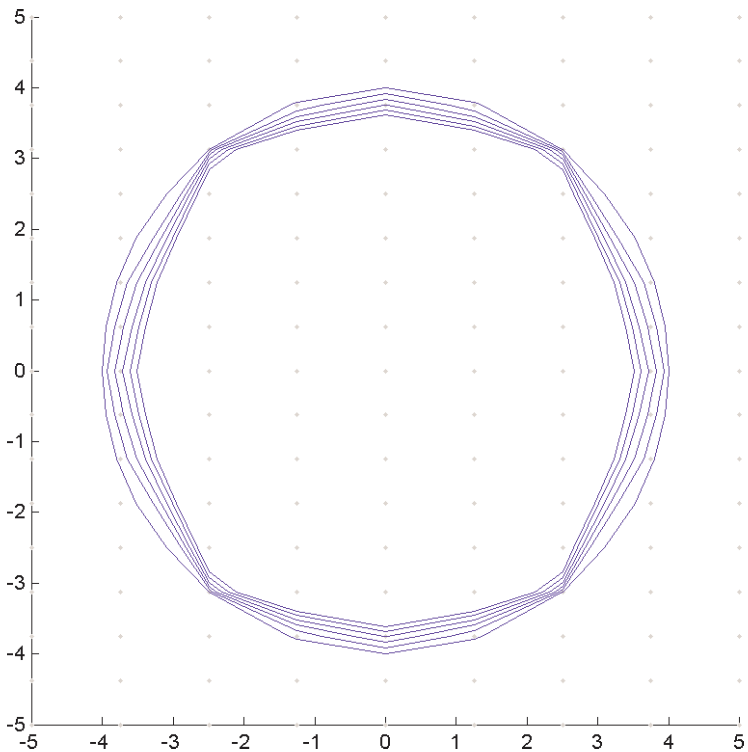


$$\Gamma(t) = \{x \mid \varphi_1(t,x) = 0 \wedge \varphi_2(t,x) > 0\}$$

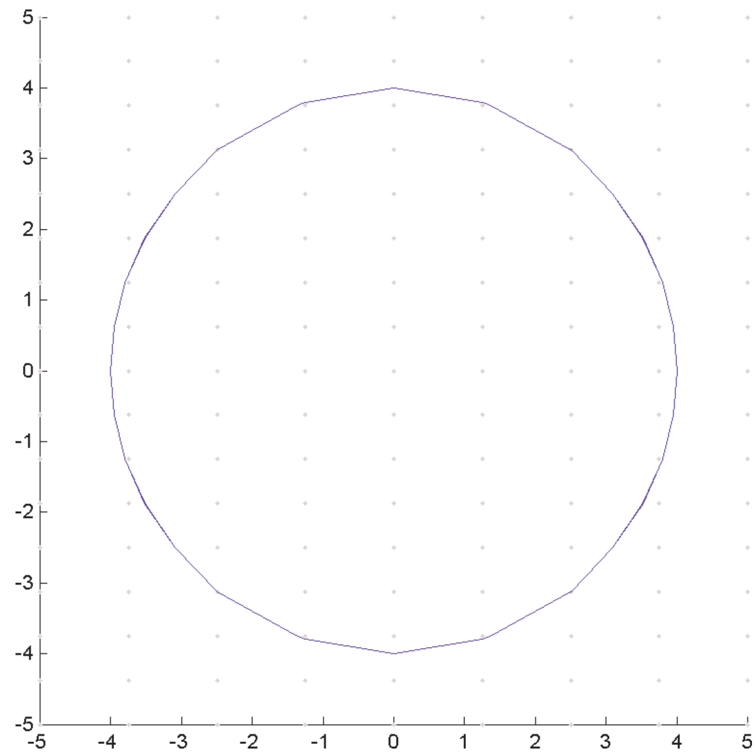


Reinitialization with Subcell Fix (v1.1)

- Different treatment for nodes adjacent to the interface
  - Distance to interface is estimated and alternative update results in less movement
- Compare interface locations after $160i$ iterations, i = 0, 1, …, 5



without subcell fix



with subcell fix

Continuous Reachable Sets

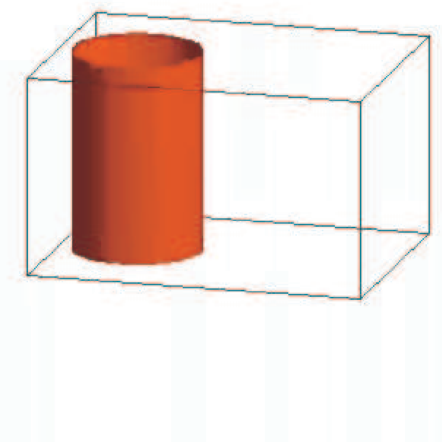
- Nonlinear dynamics with adversarial inputs

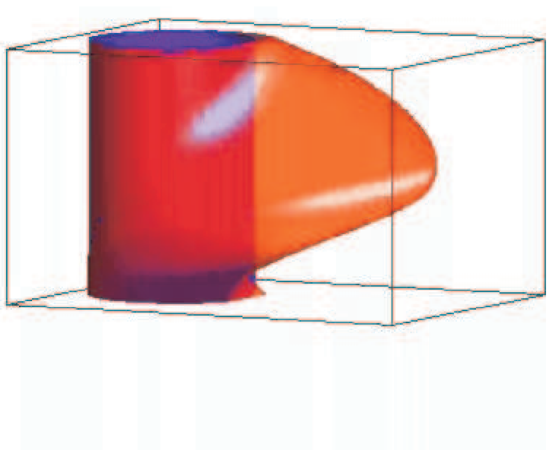
$$D_t\varphi(t,x) + \min\left[0, H(x, D_x\varphi(t,x))\right] = 0$$

$$H(x,p) = \max_{a\in\mathcal{A}} \min_{b\in\mathcal{B}} \left[p \cdot f(x,a,b)\right]$$

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -v_a + v_b \cos x_3 + a x_2 \\ v_b \sin x_3 - a x_1 \\ b - a \end{bmatrix}$$
$$= f(x,a,b)$$

$$a \in \mathcal{A} = [-1, +1]$$
$$b \in \mathcal{B} = [-1, +1]$$
$$v_a, v_b \text{ constant}$$





Hybrid System Reachable Sets

- Mixture of continuous and discrete dynamics

switch condition:
discrete control input
initiating maneuver

switch condition:
$t = \pi$

straight1
$\dot{x} = f_s(x)$
q_1

arc1
$\dot{x} = f_a(x)$
q_2

straight2
$\dot{x} = f_s(x)$
q_3

state reset:
Rotate x clockwise 90°

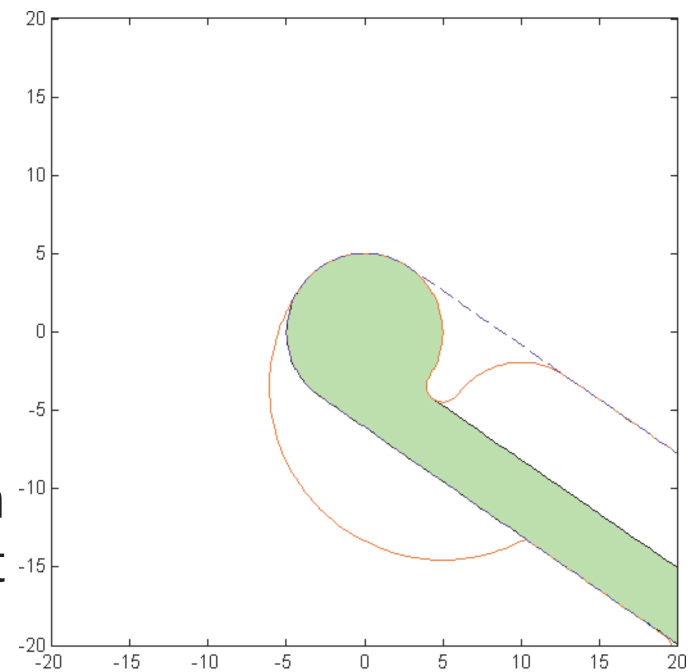
state reset:
Rotate x clockwise 90°

$$f_s(x) = \begin{bmatrix} -v_a + v_b \cos\psi_r \\ v_a \sin\psi_r \end{bmatrix}$$

$$f_a(x) = \begin{bmatrix} -v_a + v_b \cos\psi_r + \omega x_2 \\ v_a \sin\psi_r - \omega x_1 \end{bmatrix}$$

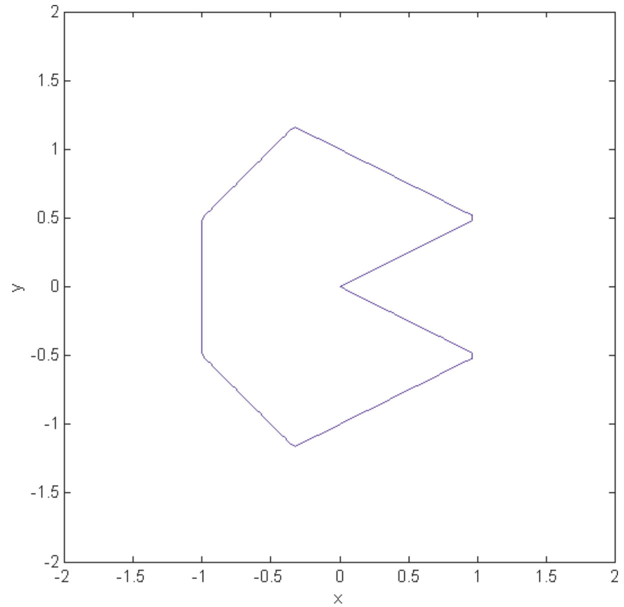
v_a, v_b, ψ_r, ω constant

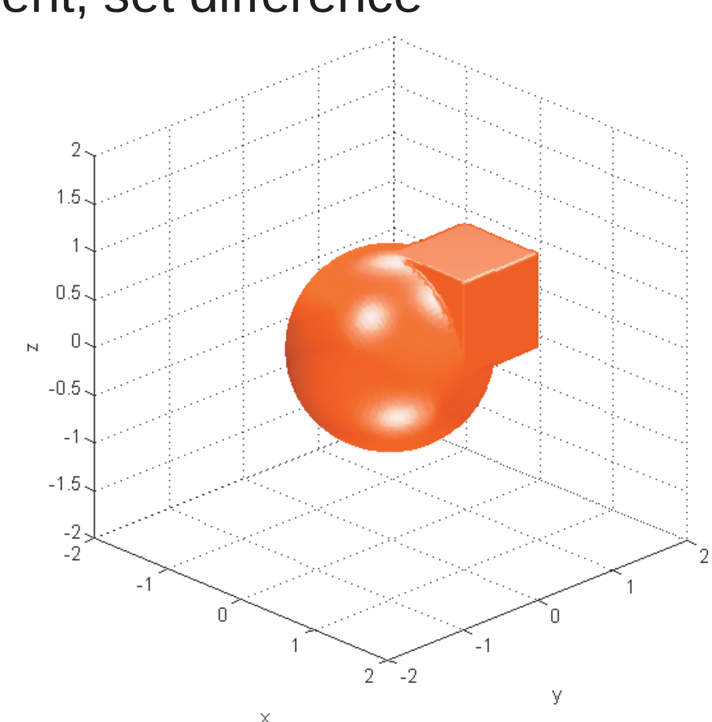
set of states leading to collision
whether maneuver is initiated or not



Constructive Solid Geometry

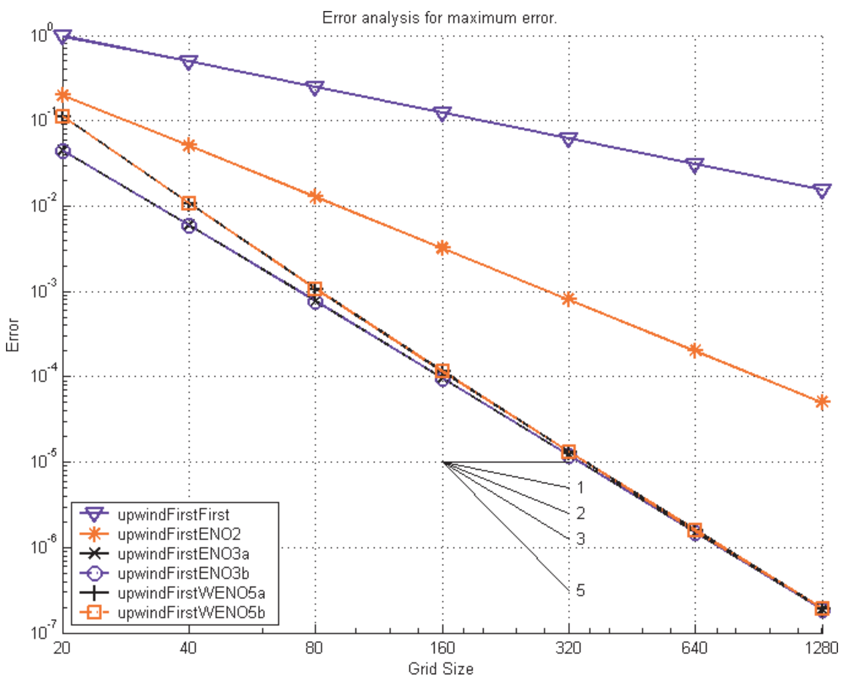
- Simple geometric shapes have simple algebraic implicit surface functions
  - Circles, spheres, cylinders, hyperplanes, rectangles
- Simple set operations correspond to simple mathematical operations on implicit surface functions
  - Intersection, union, complement, set difference



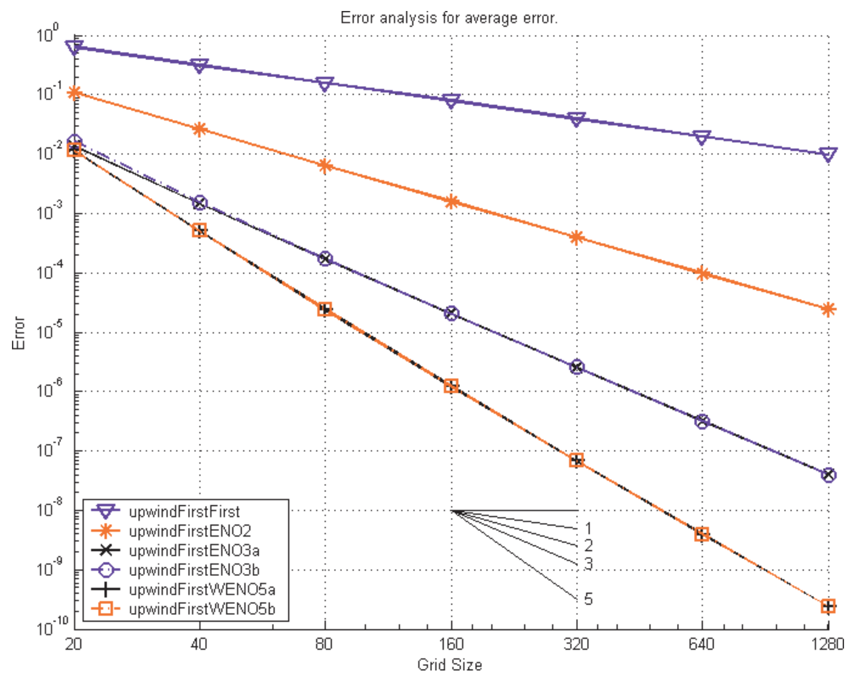


High Order Accuracy

- Temporally: explicit, Total Variation Diminishing Runge-Kutta integrators of order one to three
- Spatially: (Weighted) Essentially Non-Oscillatory upwind finite difference schemes of order one to five
  - Example: approximate derivative of function with kinks



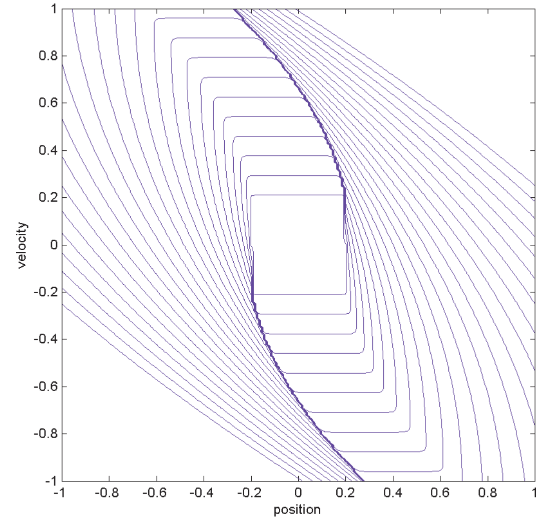
maximum error

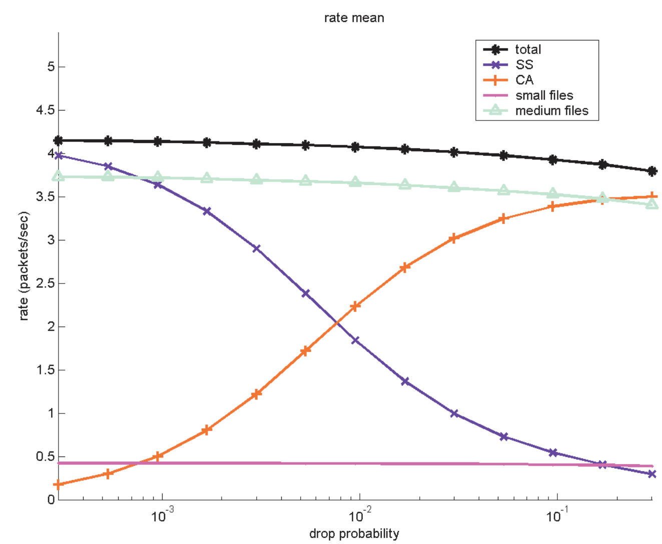


average error

Other Available Examples

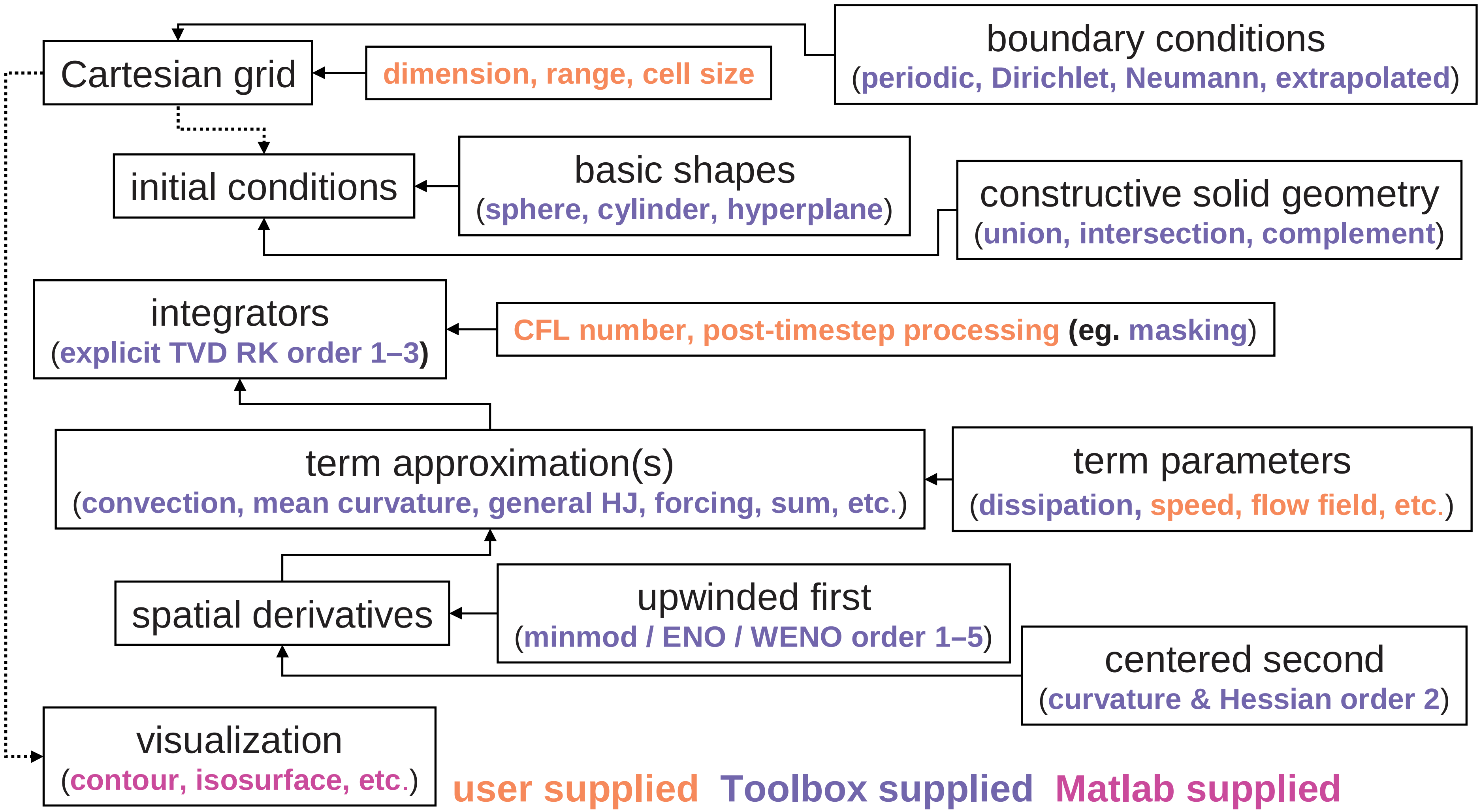
- Hybrid Systems Computation & Control
  - Mitchell & Templeton (2005)
  - Stationary HJ PDE for minimum time to reach or cost to go
  - Stochastic hybrid system model of Internet TCP transmission rate
- Journal of Optimization Theory & Applications
  - Kurzhanski, Mitchell & Varaiya (2006)
  - State constrained optimal control
- Following David Donoho's "Reproducible Research" initiative





The Toolbox: How to Use It

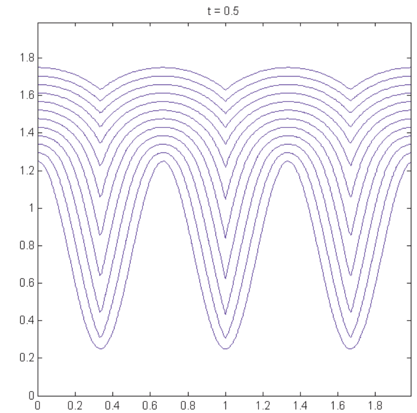
- Cut and paste from existing examples
- Most code is for initialization and visualization



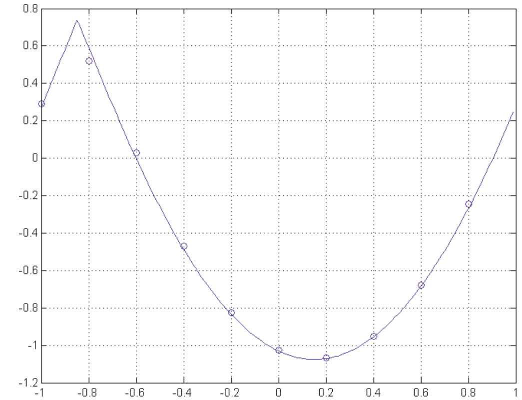
Other Level Set Software Packages

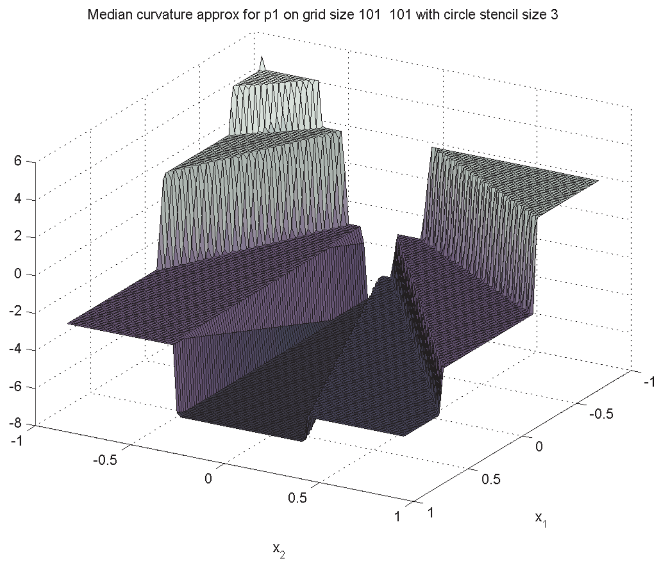
- Level Set Method Library (LSMLIB) [Chu & Prodanovic]
  - C/C++/Fortran with Matlab interface, dimensions 1–3
  - two types of motion, fast marching & velocity extension
  - localized algorithms, serial and parallel execution
- Multivac C++ [Mallet]
  - C++, dimension 2
  - six types of motion, fast marching
  - localized algorithms
  - application: forest fire propagation and image segmentation
- “A Matlab toolbox implementing level set methods” [Sumengen]
  - Matlab, dimension 2
  - three types of motion
  - application: vision and image processing
- Toolbox Fast Marching [Peyré]
  - Matlab interface to C++, dimensions 2–3
  - Static HJ PDE only

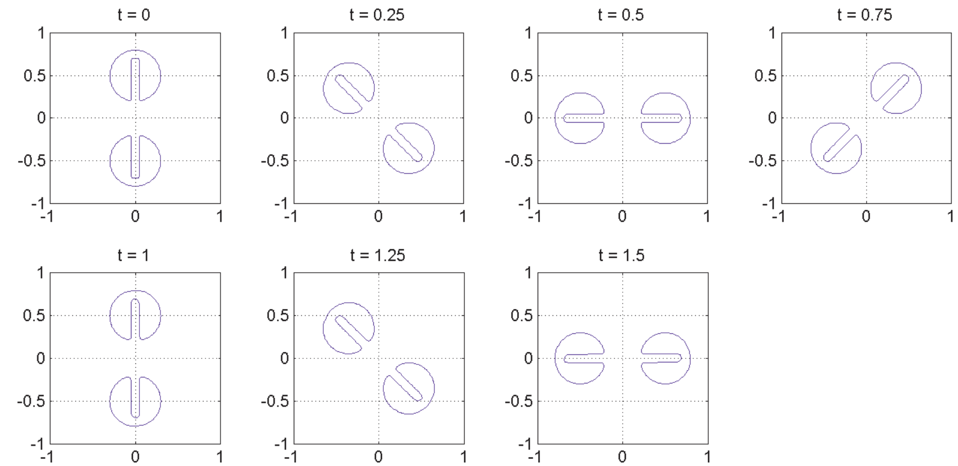
Outline



- Level set methods: dynamic implicit surfaces and the Hamilton-Jacobi equation
- Toolbox of level set methods: features and examples
- Adding schemes
  - How to achieve flexibility and efficiency
  - SSP RK integrators
  - Monotone motion by mean curvature

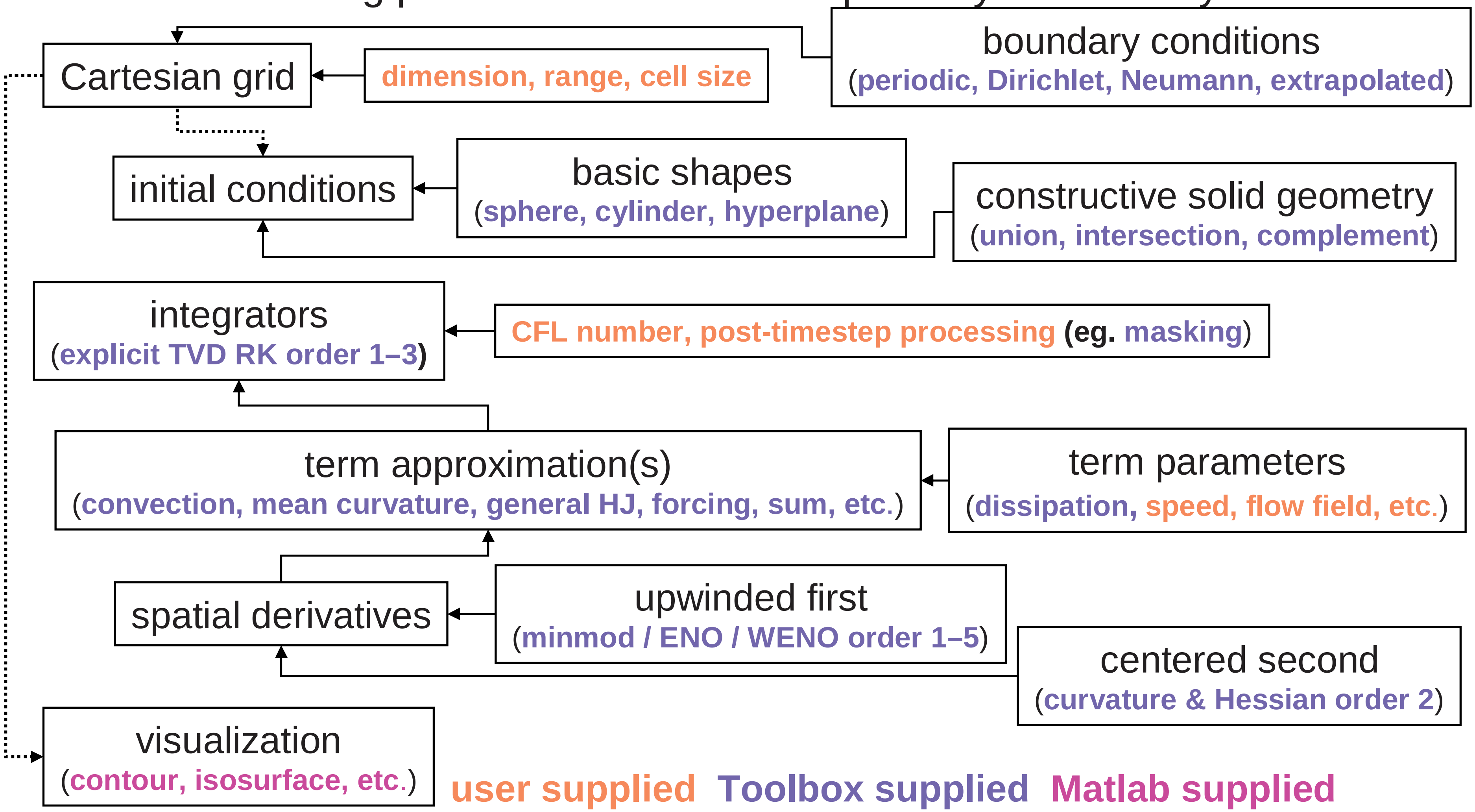






The Toolbox: How to Extend It

- Choose appropriate class of routines to modify
- Use coding patterns to achieve compatibility & efficiency

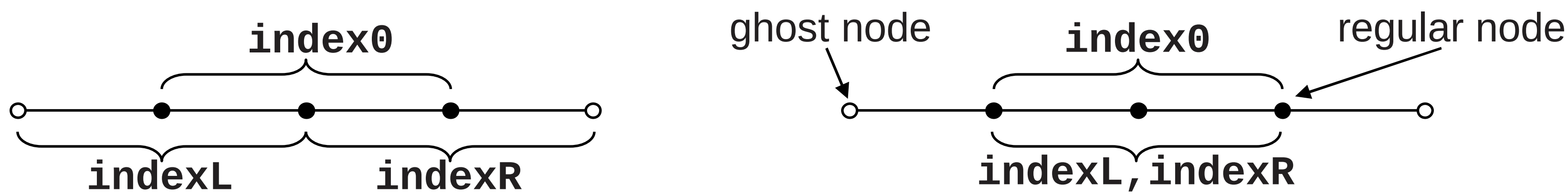


Coding Patterns: Functions of x

- All scalar functions of $x \in \Omega \subset \mathbb{R}^d$ are stored in d-dimensional Matlab arrays
  - Each element corresponds to a node in the grid
- Permits flexible and efficient operations—no explicit loops!
  - Implement $a(t,x)||D_x\varphi(t,x)||$ as **`speed .* magnitude`**
  - Dimensionally independent
  - Command interpretation overhead trivial
  - Matlab implementation optimized for cache & processor efficiency
- Vector functions of x are stored in cell arrays
  - For example, distance to origin for d = 2 generated by **`sqrt(grid.xs{1}.^2 + grid.xs{2}.^2)`**
  - Also convenient for calls to Matlab functions: **`interpn(grid.xs{:},data,samples{:})`**
- Requires structured, dense data layout
  - Unstructured and adaptive meshes are infeasible
  - Localized algorithms are of dubious benefit

Coding Patterns: Indexing

- Dimensionally independent indexing required in boundary conditions and finite difference derivative approximation
  - Use cell vectors to generate index lists



in dimension **`diffDim`** | in other dimensions

```
index0 = cell(grid.dim, 1);
for d = 1 : grid.dim
  index0{d} = (1 : grid.N(d)) + ghostNodes;
end
indexL = index0;
indexL{diffDim} = indexL{diffDim} - 1;
indexR = index0;
indexR{diffDim} = indexR{diffDim} + 1;
Dx2 = (data(indexR{:}) - 2 * data(index0{:}) ...
       + data(indexL{:})) / grid.dx(diffDim)^2;
```

Additional SSP RK Integrators

- Basic Toolbox includes standard explicit, Strong Stability Preserving Runge-Kutta temporal integrators
  - Choices are order p and number of substeps s
  - For standard schemes: $s = p = 1, 2, 3$
- Alternative: set $s > p$ [Spiteri & Ruuth, SINUM 2002]
  - Additional work on substeps offset by larger CFL constraint
  - Schemes specified by parameters α_{ik} and β_{ik}
- Implemented
  - Integrator for general α–β schemes
  - Integrator with α–β tables for (s, p) schemes: (1,1), (2,2), (3,3), (3,2), (4,2), (4,3), (5,3), (5,4)
  - ODE test problems to validate order of accuracy

SSP RK Schemes

- To solve ODE system ${}^{d}/_{dt}\Phi(t) = \mathcal{L}(t,\Phi(t))$
  - Introduce substep sample times $t^{(i)}$

$$t^{(i)} = t_n + \Delta t \sum_{k=0}^{i-1} c_{ik}, \text{ where } c_{ik} = \beta_{ik} + \sum_{j=k+1}^{i-1} \alpha_{ij} c_{jk},$$

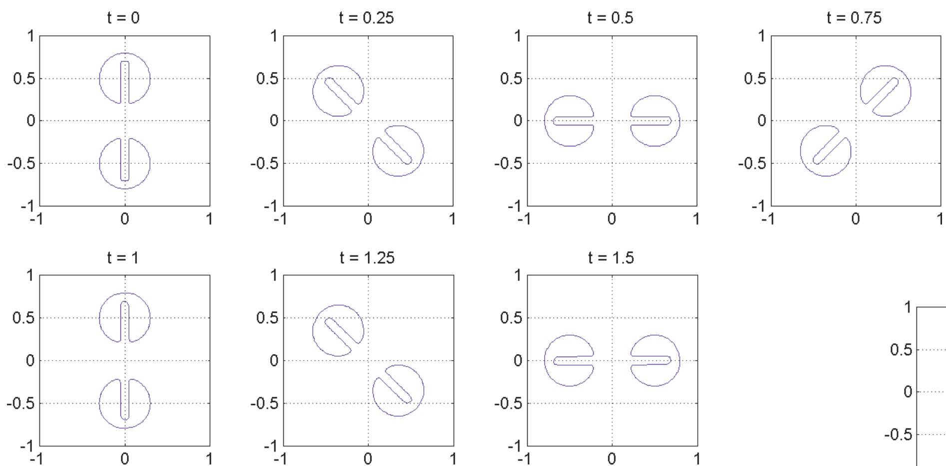
  - Scheme given by (for i = 1, 2, …, s)

$$\begin{aligned}
\Phi^{(0)} &= \Phi(t_n), \\
\Phi^{(i)} &= \sum_{k=0}^{i-1} \left[\alpha_{ik}\Phi^{(k)} + \Delta t \beta_{ik} \mathcal{L}(t^{(k)}, \Phi^{(k)})\right], \\
\Phi(t_{n+1}) &= \Phi^{(s)}.
\end{aligned}$$

  - Only the forward operator $\mathcal{L}$ is required if $\beta_{ik} \geq 0$

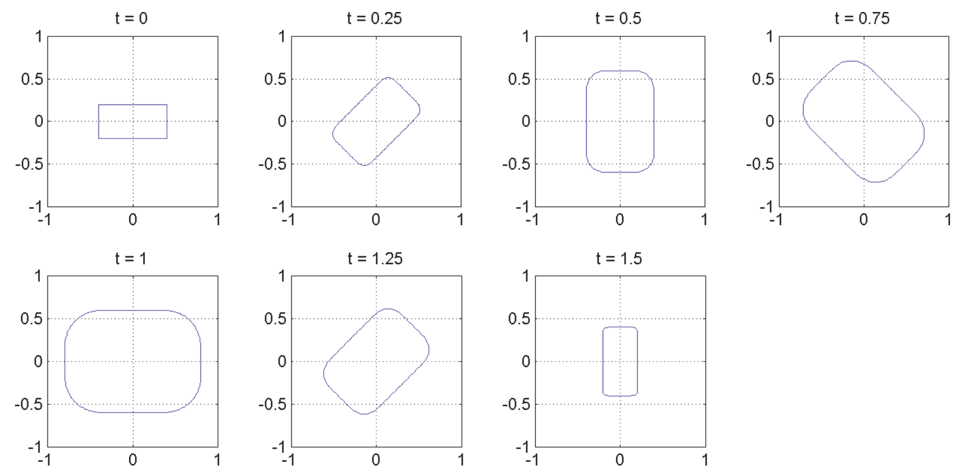
SSP RK Examples

- Five examples
  - Three initial conditions: circle, rectangle, Zalesak's disks
  - Two flow fields: rotation or rotation + normal direction
- Simple shapes and flow fields chosen to minimize effect of spatial derivative approximation errors
  - Spatial approximation fifth order accurate WENO



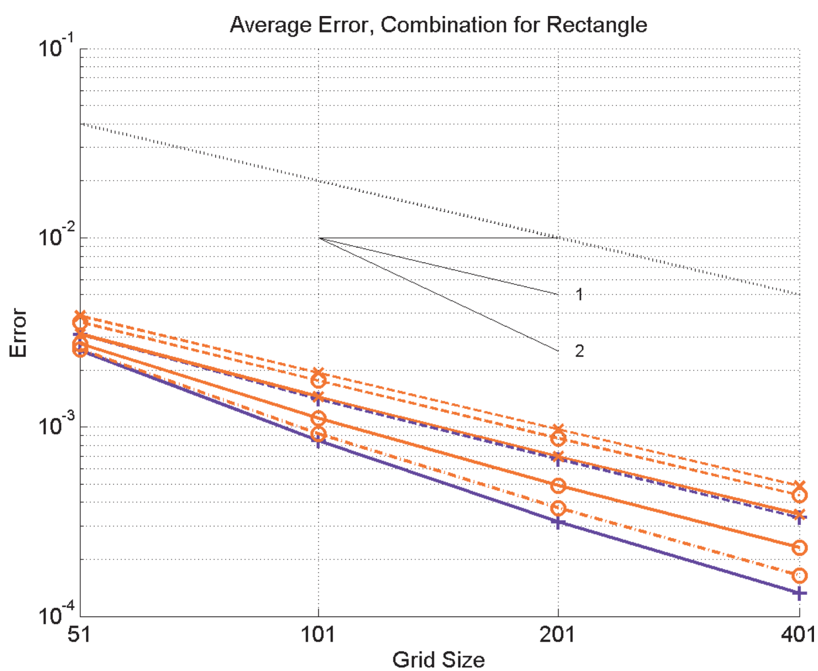
rotation for Zalesak's disks

combination for rectangle

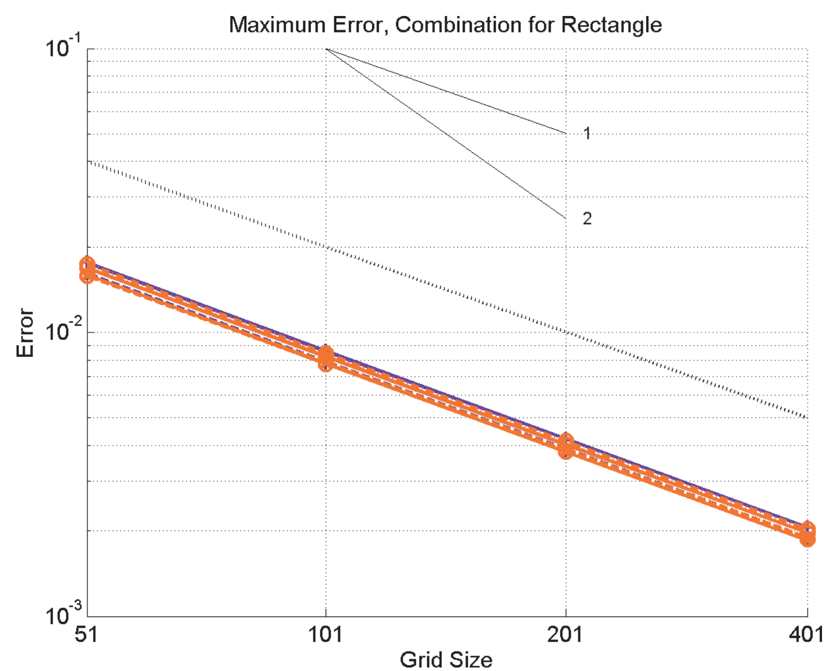


SSP RK Results: Accuracy

- Error measured against analytic solution
  - Only measured at nodes adjacent to interface
- Combination motion for rectangle is representative
  - Order of accuracy of temporal scheme makes little difference
  - When differences exist, traditional schemes (blue) are slightly more accurate than new schemes (red)



average error



maximum error

SSP RK Results: Efficiency

Scheme Source	Steps s	Order p	CFL Bound	Convection Time (sec)	Combination Time (sec)
Shu & Osher	2	2	1.0	530	957
	3	3	1.0	853	1662
Spiteri & Ruuth	3	2	2.0	390	1097
	4	2	3.0	385	840
	4	3	2.0	602	1110
	5	3	$\sim$2.65	542	1257
	5	4	$\sim$1.51	847	2221

- Timing platform
  - 201^2 grid with rectangular initial conditions and CFL factor 0.75
  - Matlab 7.2 (R2006a) in Windows XP version 2002 SP2
  - Intel Pentium M laptop, 1.7 GHz with 1 GB memory
- Significant time savings achieved with large CFL numbers

Motion by Mean Curvature

- Traditional approach

$$\Delta_1 \varphi = \|D_x \varphi\| \kappa(\varphi) = \|D_x \varphi\| \, \mathrm{div} \left(\frac{D_x \varphi}{\|D_x \varphi\|} \right)$$

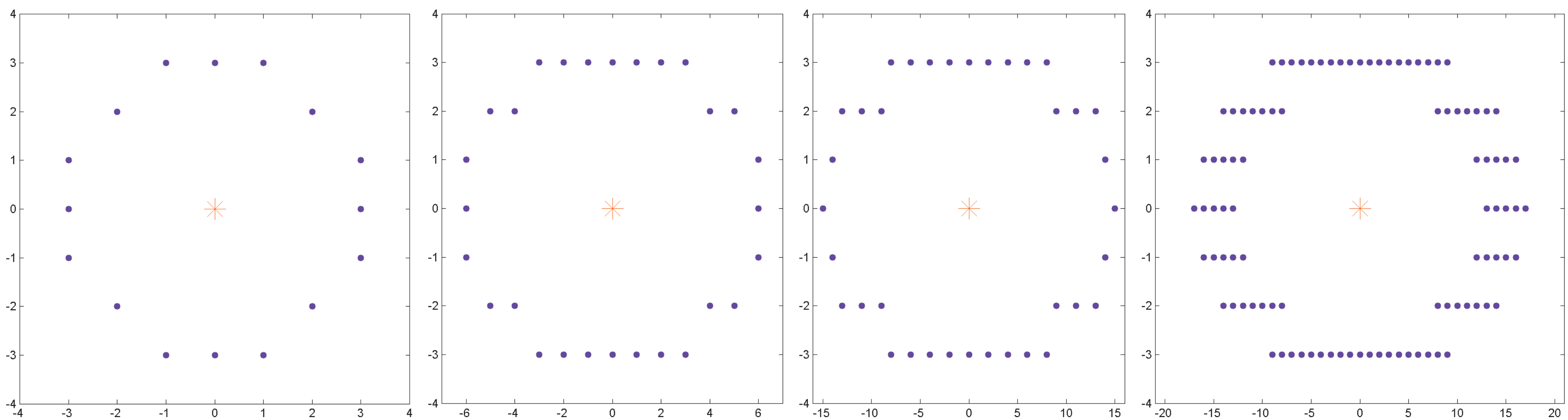
$$= \sum_{i=1}^{d} \frac{\partial^2 \varphi}{\partial x_i^2} - \frac{1}{\|D_x \varphi\|^2} \sum_{i,j=1}^{d} \frac{\partial^2 \varphi}{\partial x_i \partial x_j} \frac{\partial \varphi}{\partial x_i} \frac{\partial \varphi}{\partial x_j}$$

  - Use centered differences to approximate partial derivatives
  - Not monotone, so convergence theory does not apply
- Alternative [Oberman, *Numerische Mathematik* 2004]
  - Gather circular stencil $\mathcal{S}_x$ of nodes around x
  - Let $\varphi_*(x)$ = median{ $\varphi(x_k)$ | $x_k \in \mathcal{S}_x$ }
  - Then monotone approximation is

$$\Delta_1 \varphi(x) = \frac{2(\varphi_*(x) - \varphi(x))}{d_x^2} + \mathcal{O}(d_x^2 + d_\theta),$$

Adapting Median-Based Approach

- How to construct stencils when Δx not constant?
  - Define stencil width w, $\Delta x_{\text{max}} = \max_i \Delta x^{(i)}$ and $d_x = w \, \Delta x_{\text{max}}$
  - Initial stencil $\mathcal{S}'$ contains all nodes at distance $d_x \pm ½\Delta x_{\text{max}}$
  - Nodes are discarded if they are in a similar direction as another node whose distance is closer to d_x
- Stencils $\mathcal{S}$ for various ratios of horizontal Δx to vertical Δx
  - Also, initial stencil $\mathcal{S}'$ on 1:5 grid before discarding nodes



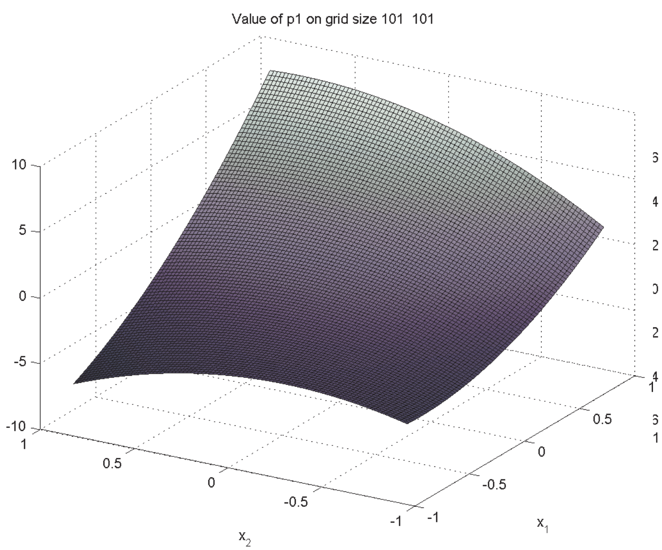
$\mathcal{S}$ (1:1) $\mathcal{S}$ (1:2) $\mathcal{S}$ (1:5) $\mathcal{S}'$ (1:5)

Quantitative Results are Disappointing

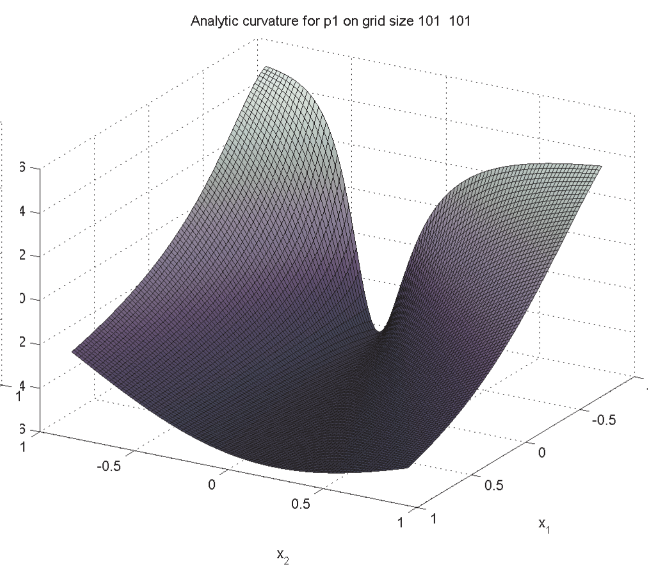
- Test on polynomial with known analytic curvature

$$\varphi_1(x) = 6x_1 + \frac{5}{4}x_2 + \frac{9}{2}\frac{x_1^2}{2} + \frac{14}{5}x_1 x_2 + \frac{26}{5}\frac{x_2^2}{2},$$

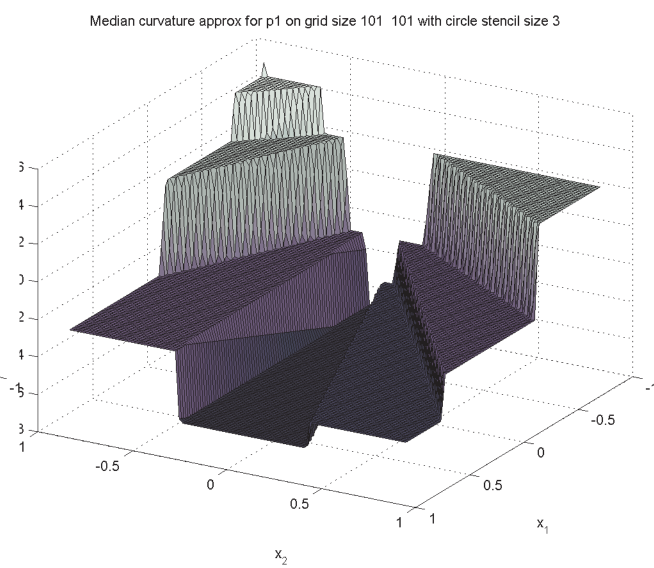
- Error for standard centered difference approximation ~10^{-10}
- Median-based approximation has only eight distinct directions



φ_1 (rotated)



analytic $\Delta_1\varphi_1$



median-based $\Delta_1\varphi_1$ with w = 3

Is Consistency Possible?

- Can we achieve $\mathcal{O}(d_x{}^2 + d_\theta) \to 0$ through some combination of $\Delta x \to 0$ and $w \to \infty$?

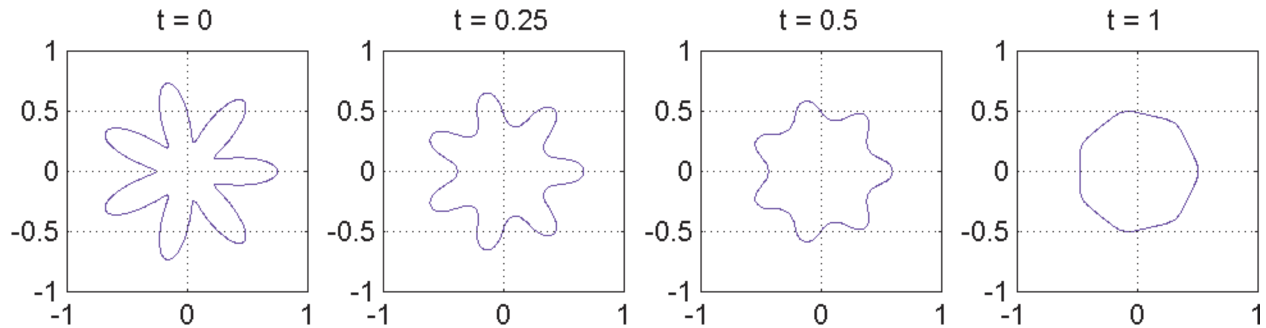
$$|\mathcal{S}| = \mathcal{O}\left(\frac{\text{circumference of stencil}}{\text{distance between } x_k}\right) = \mathcal{O}\left(\frac{2\pi d_x}{\Delta x}\right) = \mathcal{O}(\Delta x^{\gamma-1}),$$

  - Let $d_x = \Delta x^\gamma$ and note $d_\theta = (|\mathcal{S}|^{-1})$, so error is $\mathcal{O}(\Delta x^{2\gamma} + \Delta x^{1-\gamma})$
  - Balancing exponents leads to $\gamma = 1/3$, consistent asymptotic error $\mathcal{O}(\Delta x^{2/3})$, and choice $\Delta x = w^{-1.5}$

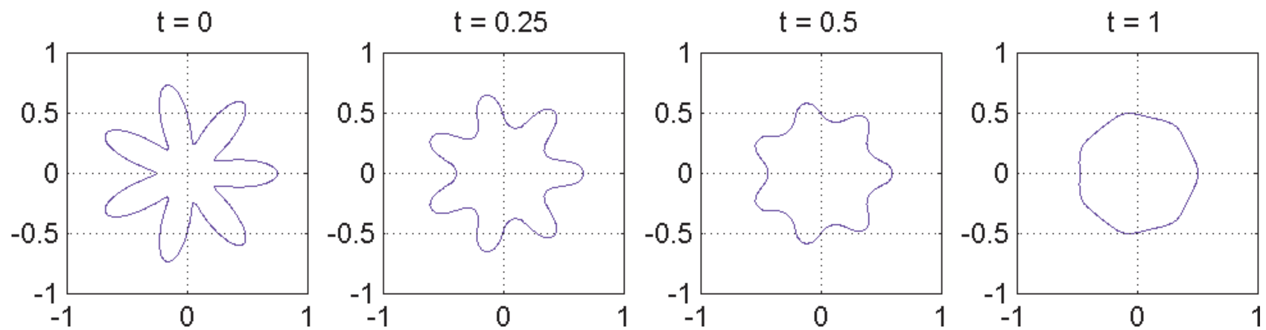
Ratio Δx	Stencil Radius	Grid Size	Circular Stencil w/o interp Mean	Circular Stencil w/o interp Max	Circular Stencil w/ interp Mean	Circular Stencil w/ interp Max	Square Stencil w/ interp Mean	Square Stencil w/ interp Max
1 : 1	1	101×101	1.995	6.523	1.439	5.459	1.439	5.459
	2	284×284	1.425	3.704	1.191	3.554	0.896	2.960
	3	521×521	0.878	2.652	0.796	2.654	0.708	2.302
	4	801×801	0.560	1.647	0.433	1.197	0.604	2.013
	5	1119×1119	0.565	1.696	0.493	1.591	0.547	1.854
1 : 2	1	101×51	1.720	6.436	1.094	5.353	1.094	5.353
	2	284×142	0.929	3.669	0.670	2.580	0.737	2.782
	3	521×261	0.614	2.179	0.445	2.047	0.579	2.293
	4	801×401	0.476	1.642	0.347	1.488	0.512	2.007
	5	1119×560	0.401	1.216	0.295	1.162	0.477	1.849
1 : 5	1	126×26	1.517	6.271	0.938	5.125	0.888	5.125
	2	355×72	0.825	2.739	0.607	2.536	0.601	2.896
	3	651×131	0.593	2.137	0.429	1.830	0.503	2.271
	4	1001×201	0.482	1.751	0.351	1.464	0.458	1.996
	5	1399×281	0.378	1.173	0.271	1.040	0.437	1.843

Why Use Median-Based Approach?

- Qualitative results are quite reasonable
- CFL bound $\mathcal{O}({d_x}^2) = \mathcal{O}(w^2 \Delta x^2)$, so for large w new scheme can be much faster
  - Simulations on 201^2 grid with (2,2) time integrator



standard centered difference (132 seconds)



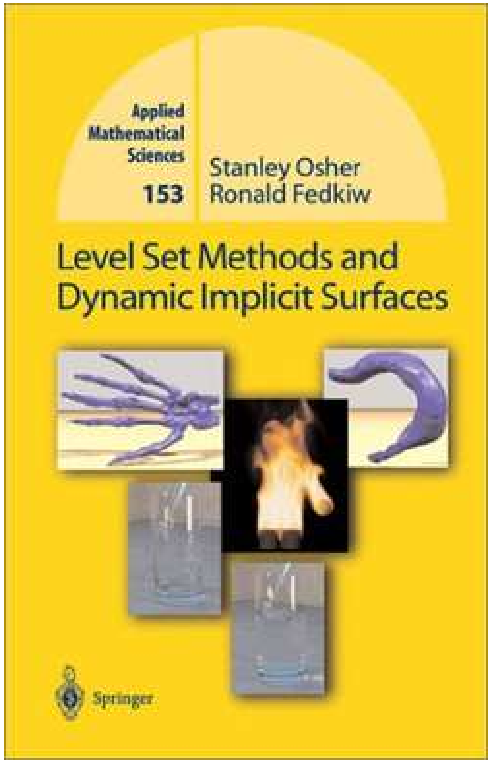
circular stencil w = 5 with interpolation (35 seconds)

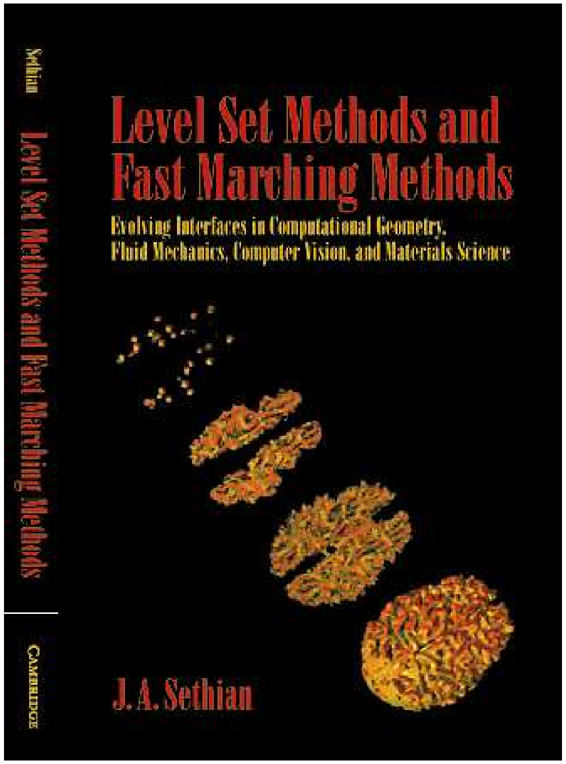
Future Work

- Algorithms
  - Implicit temporal integrators
  - Fast methods for stationary Hamilton-Jacobi
  - General boundary conditions
  - Other numerical Hamiltonians
  - ENO / WENO function value interpolation
  - Particle level set methods
- More application examples
  - Surfaces of codimension two
  - Hybrid system reachable sets and verification
  - Path planning for robotics
  - Image processing, financial math, fluid dynamics, etc.

The Toolbox is not a Tutorial

- Users will need to read the literature
- Two textbooks are available
  - Osher & Fedkiw (2002)
  - Sethian (1999)





The Toolbox: Why Use It?

- Dynamic implicit surfaces and Hamilton-Jacobi equations have many practical applications
- The toolbox provides an environment for exploring and experimenting with level set methods
  - More than twenty examples
  - Approximations of most common types of motion
  - Extensive, indexed user manual
  - High order accuracy
  - Arbitrary dimension
  - Reasonable speed with vectorized code
  - Direct access to Matlab debugging and visualization
  - Source code for all toolbox routines
- The toolbox is free for research use

`http://www.cs.ubc.ca/~mitchell/ToolboxLS`

The Flexible, Extensible and Efficient Toolbox of Level Set Methods

For more information contact

Ian Mitchell

Department of Computer Science

The University of British Columbia

mitchell@cs.ubc.ca

http://www.cs.ubc.ca/~mitchell