Numerical Methods for (Time-Dependent) HJ PDEs

Ian Mitchell
Department of Computer Science
The University of British Columbia

research supported by
National Science and Engineering Research Council of Canada

Outline

• Basic representation and approximation of functions which solve evolutionary PDEs
  – Shocks / kinks for HJ PDEs
• Level set methods, alternative software and schemes
• Convergence, consistency, stability & monotonicity
• Terminology from level set methods
• Toolbox of Level Set Methods
• Example: approximating the identical vehicle collision avoidance reach tube

Representing a Continuous Function

\[ \psi : \mathbb{R}^d \to \mathbb{R} \]

• Computer representations must be finite
  – Consequently, we are forced to construct a discrete, finite representation of \( \psi \): “discretization”
• Combination of basis functions
  \[ \psi(x) = \sum_{j=0}^{n} c_j \eta_j(x) \]
  – If \( \eta_j \) are trigonometric, we get spectral methods
  – If \( \eta_j \) have local support, we get finite element (FE) methods
• Create grid of state space, store value of \( \psi \) at the nodes
  – Called finite difference (FD) because of derivative approximation
• Create grid of state space, store average nearby value of \( \psi \)
  – Called finite volume (FV), uncommon outside fluid mechanics

Solving an Evolution PDE

\[ D_t \phi(x, t) + H(x, D_x \phi(x, t)) = 0 \]

• Although we can represent a time dependent function \( \phi \) as a function in \( \mathbb{R}^{d+1} \), most often it is represented as a collection of functions in \( \mathbb{R}^{d} \) at a set of time instants
• Much of the literature for (time-dependent) HJ PDEs grew out of conservation law schemes, so there is shared terminology
• In a Lagrangian approach, the function representation moves with the underlying flow
• In an Eulerian approach, the function representation does not move with the underlying flow
  – It is often fixed, but may be adaptive
  – Updates are done without following the underlying flow
• In a semi-Lagrangian approach, the underlying flow is used to update a fixed representation
Pros and Cons

- **Lagrangian**
  - Easy concentration of resources in regions of high complexity, but other regions may become sparse
  - Challenging to collect topological information, detect shocks

- **Eulerian**
  - Easy to collect topological information and detect shocks but challenging to adapt representation in regions of high complexity
  - CFL timestep restrictions may slow computations

- **Semi-Lagrangian**
  - Mapping between Eulerian and Lagrangian representations causes loss of accuracy due to interpolation

Shocks

- In an HJ PDE framework, the Lagrangian approach corresponds to following individual optimal trajectories
  - Objective function along the trajectory starts with the terminal cost at the terminal location, and then accumulates running cost as trajectory is followed backwards
  - But locally optimal trajectories can cross
  - How do we assign objective function value at states from which multiple trajectories can arise?
  - Viscosity solution requires us to take the best one
  - A “shock” occurs where two (or more) optimal trajectories meet with the same value
  - PDE solution may not be differentiable at these locations (perhaps “kink” is better)

Outline

- Basic representation and approximation of functions which solve evolutionary PDEs
  - Shocks / kinks for HJ PDEs
- Level set methods, alternative software and schemes
- Convergence, consistency, stability & monotonicity
- Terminology from level set methods
- Toolbox of Level Set Methods
- Example: approximating the identical vehicle collision avoidance reach tube

Level Set Methods

- Adopts Eulerian approach because of the shock detection problem
- Originally designed for dynamic implicit surface evolution
  - Representing the moving surface of a fluid
  - Merging and pinch-off handled automatically
- Easy to implement
  - Finite difference representation and approximation
  - Dimension by dimension treatment of spatial terms
  - Method of lines treatment of temporal terms
- Borrows extensively from conservation laws
  - Schemes with high orders of accuracy
- Tries to avoid complications of boundary conditions
  - Reinitialization procedure for implicit surfaces
- Implementation available: Toolbox of Level Set Methods
### 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

### Alternatives

- **Semi-Lagrangian schemes**
  - Falcone, Ferretti, Soravia…
- **Viability schemes**
  - Saint-Pierre
- Many reachability algorithms unrelated to PDEs
  - See slides from the final talk

### Convergence and Related Concepts

- Since we cannot solve the PDE exactly, we would like that our approximation approaches the true solution as some refinement parameter goes to zero: "convergence"
  - For our representation, \( x \rightarrow 0 \) and \( \Delta t \rightarrow 0 \)
- Convergence can be challenging to prove directly
- For linear PDEs, consistency + stability implies convergence
  - HJ PDE is not linear, but Barles & Souganidis (1991) showed that consistency + stability + monotonicity implies convergence
- Consistency: As \( x \rightarrow 0 \) and \( \Delta t \rightarrow 0 \), the difference approximation approaches the differential equation
- Stability: Small errors made in a single step will not be compounded over time into big errors
- Monotonicity: An increase in the approximate solution will lead to an increase in the numerical Hamiltonian

### Outline

- Basic representation and approximation of functions which solve evolutionary PDEs
  - Shocks / kinks for HJ PDEs
- Level set methods, alternative software and schemes
- Convergence, consistency, stability & monotonicity
- Terminology from level set methods
- Toolbox of Level Set Methods
- Example: approximating the identical vehicle collision avoidance reach tube
Method of Lines

\[ D_t \phi(x, t) + H(x, D_x \phi(x, t)) = 0 \]

- One method for dealing with evolution equations that have both spatial and temporal derivatives
- Basic idea: discretize and approximate spatial terms to form a coupled set of ordinary differential equations in time

\[ H(x, D_x \phi(x, t)) \approx \hat{H}(x, D_x^+ \phi(x, t), D_x^- \phi(x, t)) \]

- For example

\[ D_t^+ \phi(x_i) = \frac{\phi(x_{i+1}) - \phi(x_i)}{\Delta x} \quad D_t^- \phi(x_i) = \frac{\phi(x_i) - \phi(x_{i-1})}{\Delta x} \]

- Now solve ODE in time for \( \phi(x_i, t) \) for all nodes \( x_i \)

CFL Condition

- In the simplest approaches to solving the temporal ODE (explicit schemes) require a restriction on the temporal discretization \( \Delta t \) with respect to the spatial discretization \( \Delta x \)
  - Intuitively the restriction corresponds to restricting \( \Delta t \) such that trajectories of the underlying dynamics will not cross more than \( \Delta x \) in time \( \Delta t \)
  - For deterministic systems, \( \Delta t \leq O(\Delta x) \)
  - The constant is related to the velocity of the underlying dynamics: the faster the flow, the smaller \( \Delta t \)
  - Mathematically, the restriction arises from stability

Upwind Finite Differences

- Finite difference approximation of spatial derivative has several options for which neighbouring nodes are used

\[ D_x^+ \phi(x_i) = \frac{\phi(x_{i+1}) - \phi(x_i)}{\Delta x} \quad \text{rightward} \]
\[ D_x^- \phi(x_i) = \frac{\phi(x_i) - \phi(x_{i-1})}{\Delta x} \quad \text{centered} \]
\[ D_x^- \phi(x_i) = \frac{\phi(x_i) - \phi(x_{i-1})}{\Delta x} \quad \text{leftward} \]

- Information travels with the underlying flow, so intuitively we would like to approximate derivatives using neighbours in the upwind (against the flow) direction
- Use \( D_x^+ \) if flow is leftward, \( D_x^- \) if flow is rightward
- Mathematically, other options are unstable

ENO / WENO

- Standard schemes for higher orders of accuracy require underlying function \( \phi \) to have (more) derivatives
  - Attempts to approximate functions without those derivatives lead to incorrect oscillatory approximations
  - Since \( \phi \) may have places without derivatives, Essentially Non-Oscillatory schemes build multiple approximations, and chose the least oscillatory
  - Extension to Weighted ENO combines all approximations with weights that favour least oscillatory approximation near a kink, but in smooth regions achieve even higher order of accuracy
- Not monotonic, so no convergence theory
  - Work very well in practice
Numerical Hamiltonian

\[ H(x, D_x \phi(x, t)) \approx \tilde{H}(x, D_+ \phi, D_- \phi) \]

- Obvious substitution is unstable
- Simplest approximation: Lax-Friedrichs
  - used in Crandall & Lions (1984)
  \[ \tilde{H}(x, D_+ \phi, D_- \phi) = H(x, D_0 \phi) \]
  - Essentially adds dissipation
  \[ \frac{(\alpha/2)(D_+ \phi - D_- \phi)}{(\alpha/2)D_0 \phi} \]
- Upwinding: for \( H(x, p) = p \cdot f(x) \)
  \[ H(x, D_+ \phi, D_- \phi) = \begin{cases} 
  D_+ \phi \cdot f(x), & \text{if } f(x) \leq 0; \\
  D_- \phi \cdot f(x), & \text{if } f(x) \geq 0; 
\end{cases} \]
  - There may not be a single consistent "upwind" direction when the dynamics have inputs

Higher Dimensions

\[ D_x \phi(x) = \begin{bmatrix} D_{x1} \phi(x) \\ D_{x2} \phi(x) \end{bmatrix} \quad f(x, u) = \begin{bmatrix} f_1(x, u) \\ f_2(x, u) \end{bmatrix} \]

- Treat each dimension independently
  - For example, upwinding numerical Hamiltonian
    \[ \tilde{H}(x, D_+ \phi, D_- \phi) = \sum_{i=1}^{d} D_{xi} \phi \cdot f_i(x) \]
    \begin{align*}
    D_{xi} \phi &= \\
    &= \begin{cases} 
    D_{x1} \phi & \text{if } f_1(x) \leq 0; \\
    D_{x2} \phi & \text{if } f_1(x) \geq 0; 
    \end{cases} 
\end{align*}

TVD / SSP

- Basic scheme is forward Euler (FE)
  \[ D_t \psi(t) = f(t, \psi(t)) \text{ becomes} \]
  \[ \psi(t + \Delta t) = \psi(t) + \Delta f(t, \psi(t)) \]
- Combination of FE in time and the ENO / WENO spatial schemes described previously are shown to be stable (or not)
- Higher order of accuracy in time: Total Variation Diminishing (original name) or Strong Stability Preserving (SSP) temporal integration schemes
  - If a spatial scheme is stable using FE in time, then it will be stable using any SSP scheme
- In practice, the order of accuracy in space seems much more important to final results than the order of accuracy in time
  - Typically there is a big difference between first and second order accurate schemes, and then diminishing benefits for the extra expense of going to higher order

Implicit Surface Reinitialization

- Restriction of implicit surface function to signed distance often produces a more accurate representation
  - Gradient magnitude is not too small or too large, so location of and normal to the surface is easy to estimate
- Evolution of surface may perturb signed distance
  - Converging or diverging flow
  - Non-physical boundary conditions
- However, value of implicit surface function away from zero level set does not matter
- Reinitialization rebuilds a signed distance function from an implicit surface function without changing the zero level set
- Several available schemes
  - Fast marching (uses auxiliary static HJ PDE)
  - Reinitialization equation (uses auxiliary time-dependent HJ PDE)
  - Toolbox supports the latter but not the former
Reducing the Cost of Level Set Methods

- Solve Hamilton-Jacobi equation only in a band near interface
- Computational challenge: handling stencils near edge of band
  - “Narrowbanding” uses low order accurate reconstruction whenever errors are detected
  - “Local level set” modifies Hamiltonian near edge of band
- Data structure challenge: handling merging and breaking of interface
- Not supported in the Toolbox

Outline

- Basic representation and approximation of functions which solve evolutionary PDEs
  - Shocks / kinks for HJ PDEs
- Level set methods, alternative software and schemes
- Convergence, consistency, stability & monotonicity
- Terminology from level set methods
- Toolbox of Level Set Methods
- Example: approximating the identical vehicle collision avoidance reach tube

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?

\[
0 = D_t \varphi(t, x) = \frac{\partial \varphi}{\partial t} + \nabla \varphi \cdot \nabla (v(t, x) \cdot \nabla \varphi(t, x)) + a(t, x) \left| \nabla \varphi(t, x) \right| \quad \text{temporal derivative}
\]

\[
+ \nabla \left( \frac{\partial \varphi}{\partial t} + a(t, x) \left| \nabla \varphi(t, x) \right| \right) \quad \text{convection}
\]

\[
+ \sigma(t, x) \left| \nabla \varphi(t, x) \right| \quad \text{normal motion}
\]

\[
+ \text{sign}(\varphi(x, 0)) (\left| \nabla \varphi(t, x) \right| - 1) \quad \text{reinitialization}
\]

\[
+ h(t, x, \varphi, D_\varphi \varphi) \quad \text{general HJ}
\]

\[
- k(t, x, \varphi) \left| \nabla \varphi(t, x) \right| \quad \text{mean curvature}
\]

\[
- \text{trace}[\sigma(t, x) \sigma^T(t, x) D^2_\varphi(t, x)] \quad \text{stochastic DEs}
\]

\[
+ \lambda(t, x) \varphi(t, x) \quad \text{discounting}
\]

\[
+ F(t, x, \varphi) \quad \text{forcing}
\]

\[
D_\varphi \varphi(t, x) \geq 0, \quad \text{growth constraints}
\]

\[
D_\varphi \varphi(t, x) \leq 0, \quad \text{growth constraints}
\]

\[
\varphi(t, x) \leq \psi(t, x), \quad \text{masking constraints}
\]

\[
\varphi(t, x) \geq \psi(t, x), \quad \text{masking constraints}
\]

\[
\varphi : \mathbb{R} \times \Omega \rightarrow \mathbb{R}^k \quad \text{vector level sets}
\]
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
  \[ D_t \varphi(t, x) + a(t, x) \| D_x \varphi(t, x) \| = 0 \]

Reinitialization Equation

- Returning the gradient to unit magnitude
  \[ D_t \varphi(t, x) + \text{sign}(\varphi(0, x))(\| D_x \varphi(t, x) \| - 1) = 0 \]
General Hamilton-Jacobi
- Motion may depend non-linearly on gradient
  \[ D_t \varphi(t, x) + H(t, x, D_x \varphi(t, x)) = 0 \]
- Example: rigid body rotation about the origin

![Graphs illustrating motion and errors of various schemes](image1)

Motion by Mean Curvature
- Interface speed depends on its curvature \( \kappa \)
  \[ D_t \varphi(t, x) - b(t, x) \kappa(t, x) \| D_x \varphi(t, x) \| = 0 \]

![Graphs illustrating shrinking spiral and shrinking star](image2)

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

![Graphs illustrating combining terms](image3)
### Constraints on Function Value
- Level set function constrained by user supplied implicit surface function
  - Example: masking a region of the state space
    \[ \varphi(t, x) \leq \psi(x) \quad \varphi(t, x) \geq \psi(x) \]

### Constraints on Temporal Derivative
- Sign of temporal derivative controls whether implicit set can grow or shrink
  \[ D_t \varphi(t, x) \leq 0 \quad D_t \varphi(t, x) \geq 0 \]
  - Example: reachable set only grows

### 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
  \[ D_t \varphi + f^T D_x \varphi - \frac{1}{2} \text{trace} \left[ \sigma \sigma^T D^2_x \varphi \right] = 0 \]
  - Example: linear DE with additive noise
    \[ f(x(t)) = ax, \quad \sigma(x(t)) = b \text{ where } a = 1, \ b = 0.1 \]

### 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
  \[ D_{t_1} \varphi_1 - \text{sign}(\varphi_2) |\lambda \text{sign}(\varphi_2) \kappa(\varphi_1) - 1| D_{t_2} \varphi_1 = 0 \]
  \[ D_{t_2} \varphi_2 - \text{sign}(\varphi_1) |\lambda \text{sign}(\varphi_1) \kappa(\varphi_2) + 1| D_{t_1} \varphi_2 = 0 \]
  \[ \Gamma(t) = \{ x \mid \varphi_1(t, x) = 0 \land \varphi_2(t, x) > 0 \} \]
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 160 iterations, \( i = 0, 1, \ldots, 5 \)

Continuous Reachable Sets

- Nonlinear dynamics with adversarial inputs

\[
D_t \varphi(t, x) + \min_{a \in A} [0, H(x, D_t \varphi(t, x))] = 0
\]

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

\[
\frac{d}{dt} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -v_0 + v_1 \cos x_3 + ax_2 \\ v_1 \sin x_3 - ax_1 \\ b - a \end{bmatrix} = f(x, a, b)
\]

\( a \in A = [-1, +1] \)

\( b \in B = [-1, +1] \)

\( v_0, v_1, \) constant

Hybrid System Reachable Sets

- Mixture of continuous and discrete dynamics
- switch condition: \( t = \pi \)

\[
\begin{align*}
\text{straight1:} & \quad \dot{x} = f_s(x) \\
& \quad q_1 \\
\text{arc1:} & \quad \dot{x} = f_a(x) \\
& \quad q_2 \\
\text{straight2:} & \quad \dot{x} = f_s(x) \\
& \quad q_3
\end{align*}
\]

- state reset: Rotate \( x \) clockwise 90°

\[
f_s(x) = \begin{bmatrix} -v_0 + v_1 \cos \psi_r \\ v_1 \sin \psi_r \end{bmatrix}
\]

\[
f_a(x) = \begin{bmatrix} -v_0 + v_1 \cos \psi_r + \omega x_2 \\ v_1 \sin \psi_r - \omega x_1 \end{bmatrix}
\]

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

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
  - 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

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

Implementing Reach Tubes

- Collision avoidance example from the Toolbox
  - See Toolbox documentation section 2.6
- Pitfalls to avoid
  - Failure to include the kernel directories in the Matlab path
  - Grid is too coarse
  - State space dimensions are poorly scaled (be careful to scale both grid and dynamics)
  - Boundary conditions are incorrect
  - Incorrect initialization and/or incorrect grid bounds
  - Numerical instability caused by buggy boundary conditions, too little dissipation in Lax-Friedrichs, poor dimensional scaling, too large CFL, etc.
  - Mixing up ndgrid and meshgrid based grids (see documentation for discussion)

Numerical Methods for (Time-Dependent) HJ PDEs

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