Upwind Value Iteration on HJB equations with Solving Discontinuous Cost Function

Yuxiang Ma
Department of Mechanical Engineering, MIT
Cambridge, MA 02139
yxma20@mit.edu

***Abstract*—In dynamic programming, an optimal cost function and its corresponding optimal policy can be determined by the Hamilton-Jacobi-Bellman equation and the addictive cost function as long as the optical cost function is differentiable at that state and time. However, optimal cost functions are usually discontinuous, which makes the HJB equation hard to use. The value iteration algorithms introduced in class can not capture discontinuity or kinks very well. This work explores upwind discretization on value iteration problems. Classic upwind schemes and upwind ENO (essentially non-oscillatory) schemes are implemented. The result shows that upwind schemes can capture discontinuity and kinks better, and upwind NEO schemes can further reduce numerical dissipation and thus suppress numerical dissipation and undesired numerical oscillation.**

***Index Terms*—Dynamic programming, upwind scheme, ENO scheme**

I. Introduction & Related Work

Value Iteration is a powerful dynamic programming algorithm, but it has poor performance on systems with nonsmooth, discontinuous cost-to-go functions or policies. Many factors could make a system gain discontinuities and kinks, such as interfaces, collision, bounded input, joint limits, etc. In reality, many systems are affected by these factors, which has greatly limited the utilization of value iteration algorithms. A simple example is the minimum-time problem, in which we approximated the cost function of the double integrator on a mesh. Instead of the optimal bang-bang policy, value iteration gives a solution that oscillates around the bang-bang switching region.

The value iteration is essentially a PDE-solving algorithm. The PDE being solved, i.e. the Hamilton-Jacobi-Bellman (HJB) Equation is a variant of the Hamilton–Jacobi (HJ) equations, which are nonlinear PDEs of common mechanical systems. HJ equations relate the time-derivative with a quantity that depends on the states of the system and the first-order state partial derivatives. HJB equation has the same feature, and it further includes a system input that affects the system dynamics.

To apply value iteration algorithms to more real systems, we have to find ways to stably capture discontinuity and kinks. Similar to HJB equations, discontinuities and kinks are very common in the simulation of JB equations, such as shock waves, and interfaces. For the solving of JB equations, several numerical techniques have been developed for this purpose [1].

This work applies the numerical difference techniques in [1] to the solving of the HJB equations. Upwind schemes and ENO (essentially non-oscillatory) schemes are explored. Using the minimum time control problem as an example, this report compared the performance of upwind schemes and upwind ENO schemes with different orders, and the result shows that high-order upwind difference schemes capture kinks better in the minimum time problem because of less numerical dissipation. Meanwhile, upwind ENO schemes have slightly worse performance on the minimum time, because ENO suppresses non-physical oscillation by adding numerical dissipation to discontinuous and non-smooth regions.

Using the minimum time control problem as an example, this work shows that upwind schemes can capture kinks better because high-order upwind schemes have less numerical dissipation. Compared to pure upwind schemes of the same order, upwind ENO schemes show slightly worse performance on the minimum time problem. For problems with actual discontinuities, upwind ENO schemes might work better than pure upwind schemes.

II. Value Iteration Algorithm

A. The Hamilton-Jacobi-Bellman Equation

With additive cost (or instantaneous cost) defined as $l(\mathbf{x}, \mathbf{u})$, the long-term cost-to-go for a trajectory is $\int_0^T l(\mathbf{x}, \mathbf{u})dt$, where T can be a finite real number (finite horizon) or ∞ (infinite horizon). For a system $\dot{\mathbf{x}} = \boldsymbol{f}(\mathbf{x}, \mathbf{u})$, the optimal cost-to-go J^* can be found by searching the system input $\mathbf{u}$. This leads to the equation

$$0 = \min_{\mathbf{u}} \left[\ell(\mathbf{x}, \mathbf{u}) + \frac{dJ^*}{dt} \right]. \tag{1}$$

$$\begin{aligned} &\text{minimize}_{\pi} t_f \\ &\text{subject to } \mathbf{x}(t_0) = \mathbf{x}_0, \\ &\qquad\qquad \mathbf{x}(t_f) = \mathbf{0} \quad 10mm \\ &\qquad\qquad \ddot{q}(t) = u(t), \\ &\qquad\qquad |u(t)| \leq 1 \end{aligned} \tag{2}$$

If the problem is defined as a finite-horizon problem, the optimal cost-to-go is time-varying $J^*(\mathbf{x}, t)$, i.e.

$$\frac{dJ^*}{dt} = \frac{\partial J^*}{\partial \mathbf{x}} f(\mathbf{x}, \mathbf{u}) + \frac{\partial J^*}{\partial t},$$

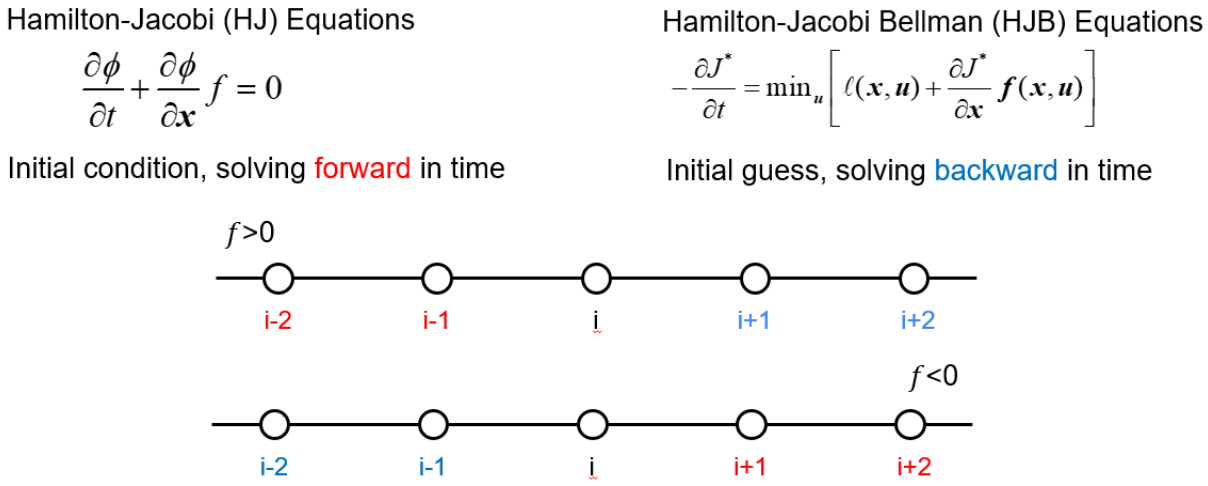

Fig. 1. Different upwind directions for HJ equation and HJB equations .

The equation 1 becomes the Hamilton-Jacobi-Bellman (HJB) equation

$$0 = \min_{\mathbf{u}} \left[\ell(\mathbf{x}, \mathbf{u}) + \frac{\partial J^*}{\partial t} \boldsymbol{f}(\mathbf{x}, \mathbf{u}) + \frac{\partial J^*}{\partial t} \right],$$

in which $\frac{\partial J^*}{\partial t}$ is independent on $\mathbf{u}$. The HJB equation can be writen as

$$-\frac{\partial J^*}{\partial t} = \min_{\mathbf{u}} \left[\ell(\mathbf{x}, \mathbf{u}) + \frac{\partial J^*}{\partial \mathbf{x}} \boldsymbol{f}(\mathbf{x}, \mathbf{u}) \right] \tag{3}$$

The optimal policy $\pi^*(\mathbf{x})$ is the one that minimizes the cost-to-go

$$\pi^*(\mathbf{x}) = \text{argmin}_{\mathbf{u}} \left[\ell(\mathbf{x}, \mathbf{u}) + \frac{\partial J^*}{\partial \mathbf{x}} f(\mathbf{x}, \mathbf{u}) \right]. \tag{4}$$

If the problem is defined as an infinite-horizon problem, the optimal cost-to-go becomes time-invariant $J^*(\mathbf{x})$, i.e.

$$\frac{dJ^*}{dt} = \frac{\partial J^*}{\partial \mathbf{x}} f(\mathbf{x}, \mathbf{u}).$$

The time differentiation term in the HJB equation vanishes

$$0 = \min_{\mathbf{u}} \left[\ell(\mathbf{x}, \mathbf{u}) + \frac{\partial J^*}{\partial \mathbf{x}} \boldsymbol{f}(\mathbf{x}, \mathbf{u}) \right] \tag{5}$$

Since the $\min$ operator still applies on the same function, the optimal policy $\pi^*(\mathbf{x})$ remains the same as equation 4.

B. Value Iteration on a Mesh

For a mesh point $\mathbf{x}_i$, let $\dot{\alpha}_i(t) = \hat{J}^*(\mathbf{x}_i, t)$

$$-\dot{\alpha}_i(t) = \min_{\mathbf{u}} \left[\ell(\mathbf{x}_i, \mathbf{u}) + \frac{\partial J^*}{\partial \mathbf{x}_i} \boldsymbol{f}(\mathbf{x}, \mathbf{u}) \right], \tag{6}$$

where temporal and spatial partial derivatives are separated. Now we can use various finite difference approximations to approximate the derivatives. In cases of infinite-horizon problems, the time derivative term vanishes, such as the minimum time problem. The optimal solution (optimal cost-to-go) of the time-invariant HJB equation 5 can be viewed as the steady-state solution of the time-variant HJB equation 3 when the terminating time is a large number. Thus, the time-invariant HJB equation can be solved in the same way as the time-variant HJB equation.

The simplest temporal difference scheme is the first-order forward Euler scheme

$$\dot{\alpha}_i^n = \frac{\hat{J}^*(\mathbf{x}_i, t_{n+1}) - \hat{J}^*(\mathbf{x}_i, t_n)}{\Delta t}. \tag{7}$$

The forward Euler scheme is widely used in solving HJ equations. However, one big difference between HJB equations and HJ equations is that HJB equations are solved backward in time, as shown in Fig. I. Intuitively, when solving HJB equations, we are essentially seeking the optimal system input that minimizes the cost associated with transitioning from a given state to all other states. Therefore, instead of a forward Euler scheme, we use integrate the temporal derivative backward in time (backward Euler scheme)

$$\dot{\alpha}_i^n = \frac{\hat{J}^*(\mathbf{x}_i, t_n) - \hat{J}^*(\mathbf{x}_i, t_{n-1})}{\Delta t}. \tag{8}$$

Higher-order temporal difference schemes, such as the Runge-Kutta method can also be easily implemented [1]. For the minimum-time problem, we are only looking for the steady-state solution. Therefore, the first-order backward Euler scheme is sufficient.

The discretization of spatial derivatives is more subtle, as their accuracy and numerical dissipation directly affect the accuracy and stability of the solution.

III. Spatial Finite Difference Schemes

A. Upwind schemes

The upwind scheme is often favored over the central difference scheme for the HJ equations (and the HJB equation) due to its ability to adapt to the nonlinear terms and capture discontinuities in the solution.

The upwind scheme was originally developed for solving hyperbolic partial differential equations (PDEs) in the field of

computational fluid dynamics (CFD), particularly for problems involving advection-dominated flow phenomena. The term "upwind" refers to the direction from which information enters a volume, aligning with the direction of flow in the problem being solved. Upwind schemes estimate nonlinear terms with higher accuracy and ensure higher numerical stability at the same time. Therefore, upwind schemes are suitable for advection-dominated problems, where nonlinearities and discontinuities present significant challenges for numerical approaches.

Similar to CFD, the HJB equation represents the evolution of a system according to its dynamics and input. The presence of spatial gradient grants the HJB equations a similar "advection-dominated" behavior to hyperbolic PDEs. Upwind schemes ensure that system information propagates in the direction of the system's speed, preserving the solution's characteristics and preventing spurious oscillations.

1) First-order upwind scheme: The following part talks about spatial difference schemes, i.e. upwind schemes. For conciseness, I ignore the subscript in the component expression of vectors. For example, x is a component of $\mathbf{x}$, and f is a component of $\boldsymbol{f}$. When the components of two vectors appear in the same equation, they are corresponding to each other. The first-order upwind scheme is

$$\frac{\partial J^*}{\partial x_i} = \frac{J_i^* - J_{i-1}^*}{\Delta x} \qquad \text{for} \qquad f < 0, \tag{9}$$

$$\frac{\partial J^*}{\partial x_i} = \frac{J_{i+1}^* - J_i^*}{\Delta x} \qquad \text{for} \qquad f > 0. \tag{10}$$

The upwind scheme here is different from the common upwind scheme applied in HJ equations [1]. Since the HJB equations are solved backward in time, upwind actually means information is transported from the future upstream to the current downstream. Thus, the upwind direction here is opposite to the upwind direction in the HJ equations, as shown in Fig. I. A compact form of the scheme can be developed by defining

$$f^+ = \max(f, 0), \qquad f^- = \min(f, 0) \tag{11}$$

and

$$J_x^- = \frac{J_i^* - J_{i-1}^*}{\Delta x}, \qquad J_x^+ = \frac{J_{i+1}^* - J_i^*}{\Delta x}, \tag{12}$$

then the compact form becomes

$$-\dot{\alpha}_i^n = \min_{\mathbf{u}} \left[\ell(\mathbf{x}_i, \mathbf{u}) + J_x^- f^- + J_x^+ f^+ \right]. \tag{13}$$

2) Higher-order upwind scheme: Higher-order upwind schemes are usually preferred because of the lower numerical dissipation and dispersion [1], [2]. In practice, upwind schemes with up to fifth-order accuracy are commonly used [2]–[4].

To construct a compact form for second order upwind scheme, we only have to switch 12 with the following gradient computations [1]

$$\begin{aligned} J_x^- &= \frac{3J_i^* - 4J_{i-1}^* + J_{i-2}^*}{2\Delta x}, \\ J_x^+ &= \frac{-J_{i+2}^* + 4J_{i+1}^* - 3J_i^*}{2\Delta x}, \end{aligned} \tag{14}$$

Similarly, construct the third-order upwind scheme can be constructed [1], [4]

$$\begin{aligned} J_x^- &= \frac{2J_{i+1}^* + 3J_i^* - 6J_{i-1}^* + J_{i-2}^*}{6\Delta x}, \\ J_x^+ &= \frac{-J_{i+2}^* + 4J_{i+1}^* - 3J_i^* - 2J_{i-1}^*}{6\Delta x}, \end{aligned} \tag{15}$$

$$\begin{aligned} J_x^- &= \frac{-3J_{i+2}^* + 30J_{i+1}^* + 20J_i^* - 60J_{i-1}^* + 15J_{i-2}^* - 2J_{i-3}^*}{60\Delta x}, \\ J_x^+ &= \frac{2J_{i+3}^* - 15J_{i+2}^* + 60J_{i+1}^* - 20J_i^* - 30J_{i-1}^* + 3J_{i-2}^*}{60\Delta x}. \end{aligned} \tag{16}$$

:

$$\begin{aligned} J_x^- &= \frac{4J_{i+3}^* - 42J_{i+2}^* + 252J_{i+1}^* + 105J_i^* - 420J_{i-1}^* + 126J_{i-2}^* - 28J_{i-3}^* + 3J_{i-4}^*}{420\Delta x}, \\ J_x^+ &= \frac{-3J_{i+4}^* + 28J_{i+3}^* - 126J_{i+2}^* + 420J_{i+1}^* - 105J_i^* - 252J_{i-1}^* + 42J_{i-2}^* + 4J_{i-3}^*}{420\Delta x}, \end{aligned} \tag{17}$$

The seventh-order upwind scheme suffers from convergence issues, and I had to add a numerical dissipation term $\alpha(f^- + f^+)(J_x^- - J_x^+)/2$ to equation 13 to stabilize the value iteration. α is a global constant and is decided manually to keep the dissipation as small as possible.

B. Upwind ENO scheme

High-order upwind schemes use several points to estimate one derivative, and the choice of the point patch or stencil is only slightly biased, i.e. one point offset. However, sometimes stencils with more offset might be a better choice, when the finite difference stencil could include a nearby discontinuity or kink. Inspired by this idea, essentially nonoscillatory (ENO)

schemes compute the numerical flux using the smoothest possible stencils. ENO schemes suppress non-physical oscillation in regions near discontinuities by adding numerical diffusion and therefore lead to higher-accuracy solutions. To explore the effect of upwind ENO schemes on the solution of HJB equations, I implement the following second-order upwind ENO scheme [1] in drake. The second-order upwind ENO scheme decides between two three-point stencils to approximate spatial derivative. For J_x^-, we decide between $\{i-2, i-1, i\}$ and $\{i-1, i, i+1\}$; for J_x^+, we decide between $\{i-1, i, i+1\}$ and $\{i, i+1, i+2\}$. Taking J_x^+ as an example, we first approximate the second-order derivative with the two patches

$$\begin{aligned} J_{xx}^0 &= \frac{J_{i-1}^* - 2J_i^* + J_{i+1}^*}{2\Delta x^2}, \\ J_{xx}^1 &= \frac{J_i^* - 2J_{i+1}^* + J_{i+2}^*}{2\Delta x^2}. \end{aligned} \tag{18}$$

Then we choose the stencil with a lower absolute second-order variation to approximate the spatial derivative

$$\begin{aligned} J_x^+ &= \frac{3J_{i+1}^* - 4J_i^* + J_{i-1}^*}{2\Delta x}, \quad \text{if} \quad |J_{xx}^0| < |J_{xx}^1|, \\ J_x^+ &= \frac{-J_{i+2}^* + 4J_{i+1}^* - 3J_i^*}{2\Delta x}, \quad \text{if} \quad |J_{xx}^0| > |J_{xx}^1|. \end{aligned} \tag{19}$$

Third-order ENO scheme and weighted essentially nonoscillatory scheme (WENO) are the most popular upwind schemes, especially the WENO scheme [1], [3]. [1] provides a comprehensive description of how to formulate it. However, I did not have time to implement it because of the limited time, and this could be a feature to be added in the future.

IV. Results

I explored the performance of different-order upwind schemes and upwind ENO schemes.

A. Upwind scheme

Fig. IV shows the optimal policy computed by upwind schemes with different orders. The value iteration results become more and more close to the closed-form solution. The reason is that higher-order schemes have less numerical dissipation, and thus higher-order schemes are better at capturing kinks and discontinuities. As shown in Fig IV, although not perfect, the seventh-order upwind scheme is able to capture the steep transition better.

From the results, one might conclude that we should use upwind schemes with as high orders as possible. However, as discussed in the last section, the upwind scheme becomes unstable when it is higher than 5th order. I spent some time tuning the amount of numerical dissipation such that the dissipation is not too big to smoothen the discontinuous features. Upwind schemes with even higher orders would become even harder to stabilize, making schemes with higher than 7th order unusable.

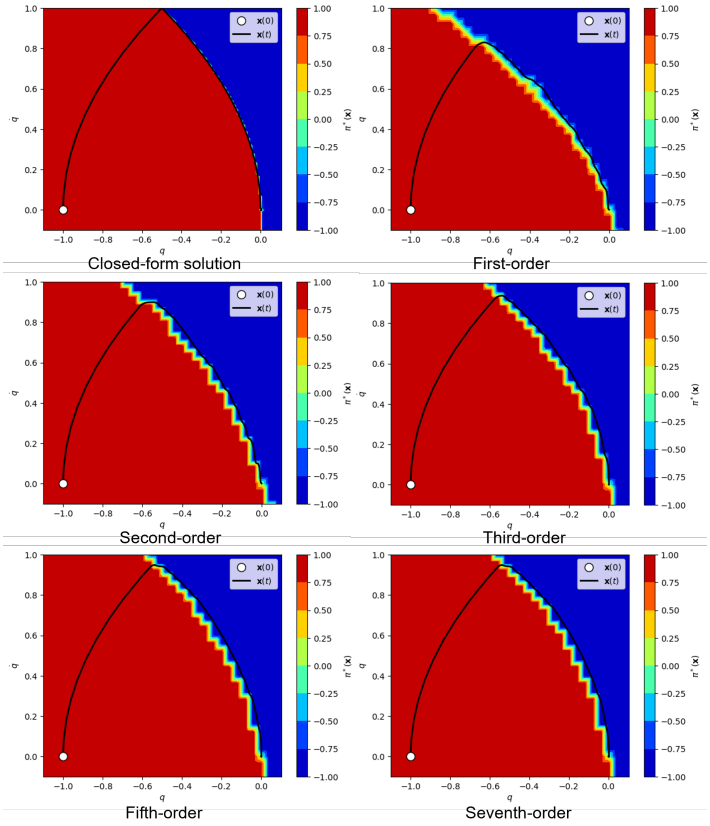

Fig. 2. Comparison of upwind schemes with different orders.

B. Upwind ENO scheme

The upwind ENO scheme is a combination of the upwind scheme and the ENO (essentially non-oscillatory) scheme. The upwind ENO scheme is expected to have better performance than pure upwind schemes on systems with discontinuous cost-to-go functions. However, this can not be demonstrated on the minimum time problem, because the cost-to-go function only has kinks but no discontinuity. A future direction of this world could be evaluating upwind and upwind ENO methods on more systems, especially systems with actual discontinuities. As shown in IV-B, the second-order upwind ENO scheme actually has a slightly worse result, compared with the pure second-order upwind scheme. The reason is that the ENO scheme adds dissipation to discontinuous and non-smooth regions, which helps to reduce non-physical oscillation, but slightly affects the accuracy of the solution. Similar results are observed in [3].

V. Conclusion

This study explored the numerical difference techniques to solve Hamilton-Jacobi-Bellman (HJB) equations, exploring the efficacy of both upwind schemes and essentially non-oscillatory (ENO) schemes. Utilizing the minimum time control problem as a case study, I compared the performance of upwind schemes and upwind ENO schemes of varying orders.

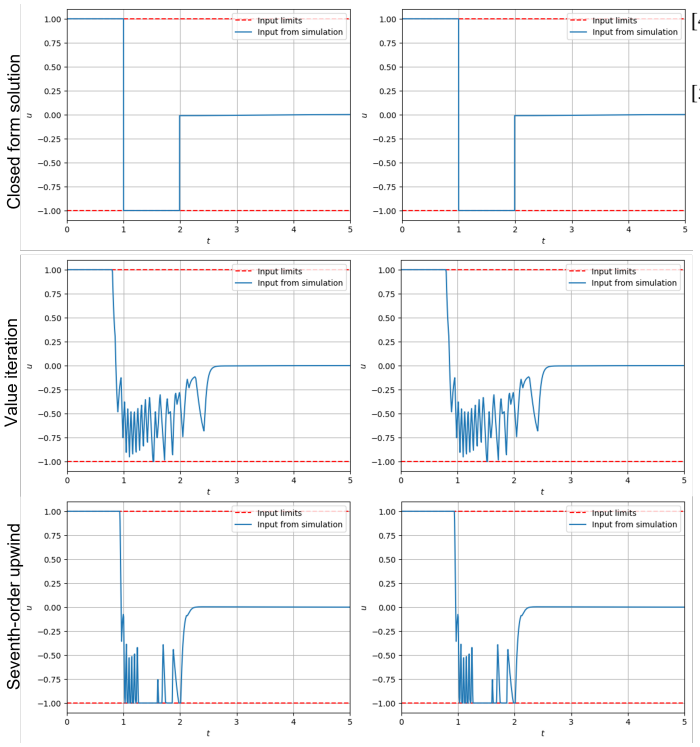

Fig. 3. Comparison between classic value iteration and seventh order solution (as well as the closed form solution) on minimum time problem.

The findings indicate that high-order upwind difference schemes excel in capturing kinks within the minimum time problem due to their reduced numerical dissipation. Conversely, upwind ENO schemes exhibited slightly inferior performance in this specific scenario, attributed to their inherent suppression of non-physical oscillations through the introduction of numerical dissipation in discontinuous and non-smooth regions.

While upwind schemes outperformed upwind ENO schemes for the minimum time problem, it's important to note that the latter may prove more effective in scenarios featuring actual discontinuities. These observations underscore the nuanced trade-offs between numerical dissipation and the suppression of non-physical oscillations when selecting between different numerical schemes for solving HJB equations.

ACKNOWLEDGMENT

The authors would also like to thank Professor Russ Tedrake and the TAs for their effort in teaching and always being helpful during the project.

REFERENCES

[1] S. Osher, R. Fedkiw, and K. Piechor, "Level set methods and dynamic implicit surfaces," *Appl. Mech. Rev.*, vol. 57, no. 3, pp. B15–B15, 2004.

[2] B. Fornberg, "Generation of finite difference formulas on arbitrarily spaced grids," *Mathematics of computation*, vol. 51, no. 184, pp. 699–706, 1988.

[3] F. Henri, M. Coquerelle, and P. Lubin, "An efficient hybrid advection scheme in a level set framework coupling weno5 and houc5 schemes based on kink detection," 2021.

[4] T. K. Sengupta, G. Ganeriwal, and S. De, "Analysis of central and upwind compact schemes," *Journal of Computational Physics*, vol. 192, no. 2, pp. 677–694, 2003.

[5] M. Baer, "findiff software package," 2018, https://github.com/maroba/findiff. [Online]. Available: https://github.com/maroba/findiff

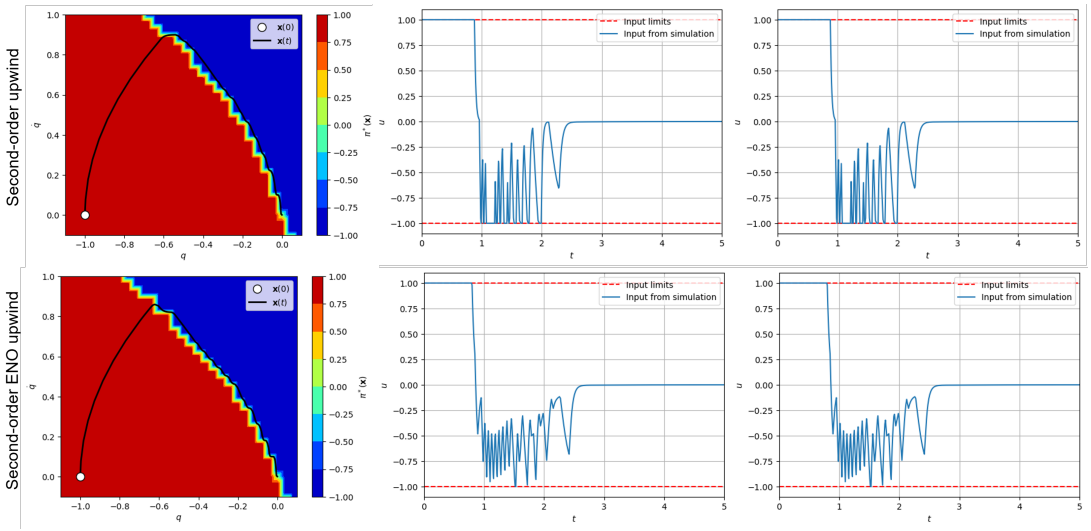

Fig. 4. Comparison between second-order upwind scheme and second-order upwind ENO scheme.