[57] F. Walsteijn, *Robust numerical methods for 2D turbulence*, Journal of Computational Physics, v114 (1994), pp.129–145.

[58] P. Woodward and P. Colella, *The numerical simulation of two-dimensional fluid flow with strong shocks*, Journal of Computational Physics, v54 (1984) pp.115–173.

[49] C.-W. Shu and S. Osher, *Efficient implementation of essentially non-oscillatory shock capturing schemes*, Journal of Computational Physics, v77 (1988), pp.439–471.

[50] C.-W. Shu and S. Osher, *Efficient implementation of essentially non-oscillatory shock capturing schemes II*, Journal of Computational Physics, v83 (1989), pp.32–78.

[51] C.-W. Shu, T.A. Zang, G. Erlebacher, D. Whitaker, and S. Osher, *High order ENO schemes applied to two- and three- dimensional compressible flow*, Applied Numerical Mathematics, v9 (1992), pp.45–71.

[52] C.-W. Shu and Y. Zeng, *High order essentially non-oscillatory scheme for viscoelasticity with fading memory*, Quarterly of Applied Mathematics, v55 (1997), pp.459-484.

[53] K. Siddiqi, B. Kimia and C.-W. Shu, *Geometric shock-capturing ENO schemes for subpixel interpolation, computation and curve evolution*, Computer Vision Graphics and Image Processing: Graphical Models and Image Processing (CVGIP:GMIP), v59 (1997), pp.278-301.

[54] G. Sod, *A survey of several finite difference methods for systems of nonlinear hyperbolic conservation laws*, Journal of Computational Physics, v27 (1978), pp.1–32.

[55] T. Sonar, *On the construction of essentially non-oscillatory finite volume approximations to hyperbolic conservation laws on general triangulations: polynomial recovery, accuracy and stencil selection*, Comput. Methods Appl. Mech. Engrg. v140 (1997) pp.157–181.

[56] M. Sussman, P. Smereka, and S. Osher, *A level set approach for computing solution to incompressible two-phase flow*, Journal of Computational Physics, v114 (1994), pp.146–159.

[41] X.-D. Liu, S. Osher and T. Chan, *Weighted essentially non-oscillatory schemes*, Journal of Computational Physics, v115 (1994), pp.200–212.

[42] I. Lomtev and G. Karniadakis, *A discontinuous spectral/hp element Galerkin method for the Navier-Stokes equations*, Int. J. Num. Meth. Fluids, to appear.

[43] S. Osher and J. Sethian, *Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulation*, Journal of Computational Physics, v79 (1988), pp.12–49.

[44] S. Osher and C.-W. Shu, *High-order essentially nonoscillatory schemes for Hamilton-Jacobi equations*, SIAM Journal on Numerical Analysis, v28 (1991), pp.907–922.

[45] P. L. Roe, *Approximate Riemann solvers, parameter vectors, and difference schemes*, Journal of Computational Physics, v43 (1981), pp.357–372.

[46] E. Rouy and A. Tourin, *A viscosity solutions approach to shape-from- shading*, SIAM Journal on Numerical Analysis, v29 (1992), pp.867–884.

[47] J. Sethian, *Level Set Methods: Evolving Interfaces in Geometry, Fluid Dynamics, Computer Vision, and Material Science*, Cambridge Monographs on Applied and Computational Mathematics, Cambridge University Press, New York, New York, 1996.

[48] C.-W. Shu, *Essentially Non-Oscillatory and Weighted Essentially Non-Oscillatory Schemes for Hyperbolic Conservation Laws*, in *Advanced Numerical Approximation of Nonlinear Hyperbolic Equations*, A. Quarteroni, Editor, Lecture Notes in Mathematics, CIME subseries, Springer-Verlag, to appear. ICASE Report 97-65.

Computer-Aided Design of Integrated Circuits and Systems, v14 (1995), pp.917–923.

[32] G. Jiang and D. Peng, *Weighted ENO schemes for Hamilton-Jacobi equations*, SIAM Journal on Scientific Computing, to appear.

[33] G. Jiang and C.-W. Shu, *On cell entropy inequality for discontinuous Galerkin methods*, Mathematics of Computation, v62 (1994), pp.531–538.

[34] G. Jiang and C.-W. Shu, *Efficient implementation of weighted ENO schemes*, Journal of Computational Physics, v126 (1996), pp.202–228.

[35] S. Jin and Z. Xin *Numerical passage from systems of conservation laws to Hamilton-Jacobi equations*, SIAM Journal on Numerical Analysis, v35(1998), pp.2385–2404.

[36] F. Ladeinde, E. O'Brien, X. Cai and W. Liu, *Advection by polytropic compressible turbulence*, Physics of Fluids, v7 (1995), pp.2848–2857.

[37] F. Lafon and S. Osher, *High order two dimensional nonoscillatory methods for solving Hamilton-Jacobi scalar equations*, Journal of Computational Physics, v123 (1996), pp.235–253.

[38] P. D. Lax , *Weak solutions of nonlinear hyperbolic equations and their numerical computation*, Communications in Pure and Applied Mathematics, v7 (1954), pp.159–193.

[39] D. Levy, G. Puppo and G. Russo, *Central WENO schemes for hyperbolic systems of conservation laws*, Mathematical Modelling and Numerical Analysis, to appear.

[40] P.-L. Lions, P.E. Souganidis, *Convergence of MUSCL and filtered schemes for scalar conservation laws and Hamilton-Jacobi equations*, Numer. Math., v69 (1995), pp.441-470.

[23] O. Friedrichs, *Weighted essentially non-oscillatory schemes for the interpolation of mean values on unstructured grids*, Journal of Computational Physics, v144 (1998), pp.194–212.

[24] J.B. Goodman and R.J. LeVeque, *On the accuracy of stable schemes for 2D scalar conservation laws,* Mathematics of Computation, v45 (1985), pp. 15–21.

[25] S. Gottlieb and C.-W. Shu, *Total variation diminishing Runge-Kutta schemes*, Mathematics of Computaion, v67 (1998), pp.73-85.

[26] E. Harabetian, S. Osher and C.-W. Shu, *An Eulerian approach for vortex motion using a level set regularization procedure*, Journal of Computational Physics, v127 (1996), pp.15–26.

[27] A. Harten, *Preliminary results on the extension of ENO schemes to two dimensional problems*, in Proceedings of the International Conference on Hyperbolic Problems, Saint-Etienne, 1986.

[28] A. Harten and S. Chakravarthy, *Multi-dimensional ENO schemes for general geometries*, ICASE Report No. 91-76 (1991)

[29] A. Harten, B. Engquist, S. Osher and S. Chakravarthy, *Uniformly high order essentially non-oscillatory schemes, III*, Journal of Computational Physics, v71 (1987), pp.231–303.

[30] J. Jerome and C.-W. Shu, *Energy models for one-carrier transport in semiconductor devices*, in IMA Volumes in Mathematics and Its Applications, v59, W. Coughran, J. Cole, P. Lloyd and J. White, editors, Springer-Verlag, 1994, pp.185–207.

[31] J. Jerome and C.-W. Shu, *Transport effects and characteristic modes in the modeling and simulation of submicron devices*, IEEE Transactions on

multidimensional case, Mathematics of Computation, v54 (1990), pp.545–581.

[15] B. Cockburn and C.-W. Shu, *TVB Runge-Kutta local projection discontinuous Galerkin finite element method for scalar conservation laws V: multidimensional systems*, Journal of Computational Physics, v141 (1998), pp.199-224.

[16] B. Cockburn and C.-W. Shu, *The local discontinuous Galerkin method for time-dependent convection diffusion systems*, SIAM Journal on Numerical Analysis, to appear.

[17] M. Crandall and P.L. Lions, *Viscosity solutions of Hamilton-Jacobi equations*, Trans. Amer. Math. Soc., v277 (1983), pp.1–42.

[18] M. Crandall and P.L. Lions, *Two approximations of solutions of Hamilton-Jacobi equations*, Mathematics of Computation, v43 (1984), pp.1–19.

[19] A. Dolezal and S. Wong, *Relativistic hydrodynamics and essentially non-oscillatory shock capturing schemes*, Journal of Computational Physics, v120 (1995), pp.266–277.

[20] W. E and C.-W. Shu, *A numerical resolution study of high order essentially non-oscillatory schemes applied to incompressible flow*, Journal of Computational Physics, v110 (1994), pp.39–46.

[21] G. Erlebacher, Y. Hussaini and C.-W. Shu, *Interaction of a shock with a longitudinal vortex*, Journal of Fluid Mechanics, v337 (1997), pp.129–153.

[22] E. Fatemi, J. Jerome and S. Osher, *Solution of the hydrodynamic device model using high order non-oscillatory shock capturing algorithms*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, v10 (1991), pp.232–244.

[7] R. Biswas, K.D. Devine and J. Flaherty, *Parallel, adaptive finite element methods for conservation laws*, Appl. Numer. Math., v14 (1994), pp.255–283.

[8] J. Casper, *Finite-volume implementation of high-order essentially non-oscillatory schemes in two dimensions*, AIAA Journal, v30 (1992), pp.2829–2835.

[9] J. Casper and H. Atkins, *A finite-volume high-order ENO scheme for two dimensional hyperbolic systems*, Journal of Computational Physics, v106 (1993), pp.62–76.

[10] B. Cockburn, *An introduction to the discontinuous Galerkin method for convection-dominated problems*, in *Advanced Numerical Approximation of Nonlinear Hyperbolic Equations*, A. Quarteroni, Editor, Lecture Notes in Mathematics, CIME subseries, Springer-Verlag, to appear.

[11] B. Cockburn and C.-W. Shu, *The Runge-Kutta local projection P^1-discontinuous Galerkin method for scalar conservation laws*, M^2AN, v25 (1991), pp.337–361.

[12] B. Cockburn and C.-W. Shu, *TVB Runge-Kutta local projection discontinuous Galerkin finite element method for scalar conservation laws II: general framework*, Mathematics of Computation, v52 (1989), pp.411–435.

[13] B. Cockburn, S.-Y. Lin and C.-W. Shu, *TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws III: one dimensional systems*, Journal of Computational Physics, v84 (1989), pp.90–113.

[14] B. Cockburn, S. Hou, and C.-W. Shu, *TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws IV: the

Bibliography

[1] R. Abgrall, *On essentially non-oscillatory schemes on unstructured meshes: analysis and implementation*, Journal of Computational Physics, v114 (1994), pp.45–58.

[2] R. Abgrall, *Numerical discretization of the first-order Hamilton-Jacobi equation on triangular meshes.*, Communications in Pure and Applied Mathematics, v49 (1996), pp.1339–1373.

[3] N. Adams and K. Shariff, *A high-resolution hybrid compact-ENO scheme for shock-turbulence interaction problems*, Journal of Computational Physics, v127 (1996), pp.27–51.

[4] T. Barth and P. Frederickson, *High order solution of the Euler equations on unstructured grids using quadratic reconstruction*, AIAA Paper No. 90-0013.

[5] T. Barth and J. Sethian, *Numerical schemes for the Hamilton-Jacobi and level set equations on triangulated domains*, Journal of Computational Physics, v145 (1998), pp.1–40.

[6] F. Bianco, G. Puppo and G. Russo, *High order central schemes for hyperbolic systems of conservation laws*, SIAM Journal on Scientific Computing, to appear.

the convergence to viscosity solutions, and on influences of limiting, integration path, and least-square procedure.

Last but not least, good coding styles are always important in scientific computing, and they become even more essential in the implementation of numerical methods on unstructured meshes. The wise storage management and the good program flow control can have huge influences over the efficiency and the accuracy for the method. There is always room to improve on this matter.

From the observation that HJ equations can be treated as some "integrated version" of conservation laws, we have constructed discontinuous Galerkin methods for HJ equations with techniques to recover the missing constants and a crucial least-square procedure to guarantee the consistency. These methods have the advantage of easy handling of complicated geometry and local mesh refinements, as well as efficient parallel implementations. Numerical examples in one and two space dimensions are shown to illustrate the capability of the methods.

6.2 Conclusions and Future Work

The aim of this dissertation is to construct efficient numerical methods which can numerically resolve shocks and discontinuities for hyperbolic equations while simultaneously producing accurate, non-oscillatory results elsewhere in the computational domain. We have achieved our goals with WENO schemes on triangles for conservation laws, and with discontinuous Galerkin methods for Hamilton-Jacobi equations.

Some improvements are still possible for future work.

For WENO schemes on unstructured meshes, we can implement them to more applications for computing (inviscid and viscous) flow fields. Efforts can be made to further improve the robustness for more complicated triangulations and to reduce the storage requirement; higher order accuracies (above 4th order) could be achieved with larger stencils; the extension to three dimensional case is an interesting topic.

We can apply our discontinuous Galerkin finite element methods for Hamilton-Jacobi equations to more practical cases, such as the level set equations, computer image-processing (curve detection, noise suppression and image enhancement). Efforts can also be made to do further analysis for the methods on

Chapter 6

Concluding Remarks

6.1 Summary

In this work, we have presented two types of high order numerical methods for hyperbolic equations on unstructured meshes. One is finite volume schemes for conservation laws; we have presented the development of third and fourth order WENO schemes based on linear and quadratic polynomials for 2D triangular meshes. Another is discontinuous Galerkin finite element methods for Hamilton-Jacobi equations; we have developed and tested a class of finite element methods for solving HJ equations.

To take the advantage of the WENO (weighted essentially non-oscillatory) philosophy, we have presented the construction of third order schemes using a combination of linear polynomials, and the construction of fourth order schemes using a combination of quadratic polynomials. Accuracy and stability issues are considered in the design of the schemes and verified numerically. Numerical examples for two dimensional scalar equations and Euler equations of gas dynamics are presented; they show the improvement of resolution using high order schemes. A parallel implementation of our WENO schemes on triangular meshes is considered and applied to the computations for Euler equations.

Table 5.5.1: Accuracy, 2D, $H(u,v) = -\cos(u+v+1), t = 0.5/\pi^2$.

	P^1		P^2		P^3	
$N \times N$	L^1 error	order	L^1 error	order	L^1 error	order
10×10	6.47E-02	—	8.31E-03	—	1.35E-02	—
20×20	2.54E-02	1.349	1.93E-03	2.106	1.57E-03	3.104
40×40	1.05E-02	1.274	4.58E-04	2.075	2.39E-04	2.716
80×80	4.74E-03	1.147	1.13E-04	2.019	2.89E-05	3.048
160×160	2.23E-03	1.088	2.83E-05	1.997	4.38E-06	2.722

	P^1		P^2		P^3	
$N \times N$	L^∞ error	order	L^∞ error	order	L^∞ error	order
10×10	1.47E-01	—	1.88E-02	—	2.36E-02	—
20×20	6.75E-02	1.123	7.34E-03	1.357	3.44E-03	2.778
40×40	2.65E-02	1.349	1.83E-03	2.004	4.59E-04	2.906
80×80	1.18E-02	1.167	4.55E-04	2.008	5.78E-05	2.989
160×160	2.23E-03	1.088	1.13E-04	2.010	8.54E-06	2.759

Table 5.6.1: Errors for the level set equation, triangular mesh with P^2.

	Errors for the Solution				Errors by Integration Path			
N	L^1 error	order	L^∞ error	order	L^1 error	order	L^∞ error	order
403	1.02E-03	—	1.32E-03	—	1.61E-04	—	5.71E-04	—
1432	1.23E-04	3.05	2.73E-04	2.27	5.84E-05	1.46	1.68E-04	1.78
5608	1.71E-05	2.85	3.18E-05	3.10	9.32E-06	2.65	4.36E-05	1.95
22238	2.09E-06	3.03	5.01E-06	2.67	1.43E-06	2.70	6.63E-06	2.72

Table 5.4.3: Differences of the solution φ recovered by two different integration paths, non-uniform mesh, Burgers equation.

	P^1		P^2		P^3	
$N \times N$	L^1 error	order	L^1 error	order	L^1 error	order
10×10	8.61E-03	—	2.90E-03	—	1.15E-03	—
20×20	4.64E-03	0.892	1.28E-03	1.180	2.44E-04	2.237
40×40	2.54E-03	0.869	4.12E-04	1.635	3.76E-05	2.698
80×80	1.81E-03	0.489	1.39E-04	1.568	6.71E-06	2.486
160×160	1.09E-03	0.732	3.66E-05	1.925	8.79E-07	2.932

Table 5.4.4: Accuracy for 2D Burgers equation, triangular mesh, $t = 0.5/\pi^2$.

	P^2				P^3			
h	L^1 error	order	L^∞ error	order	L^1 error	order	L^∞ error	order
1	5.48E-02	—	1.52E-01	—	1.17E-02	—	2.25E-02	—
1/2	1.35E-02	2.02	6.26E-02	1.28	1.35E-03	3.12	4.12E-03	2.45
1/4	2.94E-03	2.20	1.55E-02	2.01	1.45E-04	3.22	4.31E-04	3.26
1/8	6.68E-04	2.14	3.44E-03	2.17	1.71E-05	3.08	7.53E-05	2.52

Figure 5.11.1: Computer vision problem, $\varphi(x, y, \infty) = (1 - |x|)(1 - |y|)$.

Figure 5.11.2: Computer vision problem, history of iterations.

Table 5.4.2: Accuracy for 2D Burgers equation, non-uniform rectangular mesh, $t = 0.5/\pi^2$.

	P^1		P^2		P^3	
$N \times N$	L^2 error	order	L^2 error	order	L^2 error	order
10×10	4.47E-01	—	6.28E-02	—	1.61E-02	—
20×20	1.83E-01	1.288	1.50E-02	2.066	2.06E-03	2.966
40×40	8.01E-02	1.192	3.63E-03	2.047	3.48E-04	2.565
80×80	3.82E-02	1.068	9.17E-04	1.985	6.03E-05	2.529
160×160	1.87E-02	1.031	2.34E-04	1.970	8.58E-06	2.813

	P^1		P^2		P^3	
$N \times N$	L^1 error	order	L^1 error	order	L^1 error	order
10×10	8.16E-02	—	9.16E-03	—	3.39E-03	—
20×20	3.41E-02	1.259	2.09E-03	2.132	4.12E-04	3.041
40×40	1.50E-02	1.185	5.21E-04	2.004	7.03E-05	2.551
80×80	7.16E-03	1.067	1.42E-04	1.875	1.24E-05	2.503
160×160	3.50E-03	1.033	3.85E-05	1.883	1.76E-06	2.817

	P^1		P^2		P^3	
$N \times N$	L^∞ error	order	L^∞ error	order	L^∞ error	order
10×10	2.83E-01	—	4.68E-02	—	1.00E-02	—
20×20	1.25E-01	1.179	1.23E-02	1.928	1.39E-03	2.847
40×40	5.74E-02	1.123	3.54E-03	1.797	2.29E-04	2.602
80×80	2.78E-02	1.046	1.15E-03	1.622	5.11E-05	2.164
160×160	1.42E-02	0.969	2.72E-04	2.080	7.16E-06	2.835

Figure 5.11.3: Computer vision problem, $\varphi(x, y, \infty) = (1 - x^2)(1 - y^2)$.

Figure 5.10.1: Control problem, $t = 1$.

Figure 5.10.2: Control problem, $t = 1, w = \mathrm{sign}(\varphi_{\mathrm{y}})$.

The exact steady solution is $\varphi(x, y, \infty) = (1 - x^2)(1 - y^2)$. We again use a uniform rectangular mesh of 40×40 elements, the local Lax-Friedrichs flux (4.4.8)-(4.4.9), impose the exact boundary conditions for $u = \varphi_x, v = \varphi_y$ from the above exact steady solution, and take the exact value at one point (the lower left corner) to recover φ. A continuation method is used, with the steady solution using

$$I_\varepsilon(x, y) = \frac{1}{\sqrt{1 + 4y^2(1 - x^2)^2 + 4x^2(1 - y^2)^2 + \varepsilon}} \tag{5.11.4}$$

for bigger ε as the initial condition for smaller ε. The sequence of ε used are $\varepsilon = 0.2, 0.05, 0$. The results for P^2 and P^3 are presented in Fig. 5.11.3.

Figure 5.9.2: Propagating surfaces on a disk, triangular mesh, $\varepsilon = 0$.

Figure 5.9.3: Propagating surfaces on a disk, triangular mesh, $\varepsilon = 0.1$.

5.11 A Computer Vision Problem

This problem is in [46], the equation is

$$\begin{cases} \varphi_t + I(x,y)\sqrt{1+\varphi_x^2+\varphi_y^2} - 1 = 0, & -1 < x < 1, -1 < y < 1 \\ \varphi(x,y,0) = 0 \end{cases} \tag{5.11.1}$$

with $\varphi = 0$ as the boundary condition. The steady state solution of this problem is the shape lighted by a source located at infinity with vertical direction. The solution is not unique if there are points at which $I(x,y) = 1$. Conditions must be prescribed at those points where $I(x,y) = 1$. Since our method is a finite element method, we need to prescribe suitable conditions at the correspondent elements. We take

$$I(x,y) = \frac{1}{\sqrt{1+(1-|x|)^2+(1-|y|)^2}} \tag{5.11.2}$$

The exact steady solution is $\varphi(x,y,\infty) = (1-|x|)(1-|y|)$. We use a uniform rectangular mesh of 40×40 elements and the local Lax-Friedrichs flux (4.4.8)-(4.4.9). We impose the exact boundary conditions for $u = \varphi_x, v = \varphi_y$ from the above exact steady solution, and take the exact value at one point (the lower left corner) to recover φ. The results for P^2 and P^3 are presented in Fig. 5.11.1, while Fig. 5.11.2 contains the history of iterations to the steady state.

Next we take

$$I(x,y) = \frac{1}{\sqrt{1+4y^2(1-x^2)^2+4x^2(1-y^2)^2}} \tag{5.11.3}$$

Figure 5.8.4: Propagating surfaces, triangular mesh, $\varepsilon = 0$.

Figure 5.8.5: Propagating surfaces, triangular mesh, $\varepsilon = 0.1$.

It is difficult to use rectangular meshes for this problem. Instead we use the triangulation shown in Fig. 5.9.1. Notice that we have again refined the mesh near the center of the domain where the solution develops discontinuous derivatives. There are 1792 triangles and 922 nodes in this triangulation. The solutions with $\varepsilon = 0$ are displayed in Fig. 5.9.2. Notice that the solution at $t = 0$ is shifted downward by 0.2 to show the detail of the solution at later time.

Figure 5.9.1: Triangulation for the propagating surfaces on a disk.

The solution with $\varepsilon = 0.1$ are displayed in Fig. 5.9.3. Notice that the solution at $t = 0$ is again shifted downward by 0.2 to show the detail of the solution at later time.

5.10 An Optimal Control Problem

This problem is from [44], the equation is

$$\begin{cases} \varphi_t + (\sin y)\varphi_x + (\sin x + \operatorname{sign}(\varphi_y))\varphi_y - \frac{1}{2}\sin^2 y - (1 - \cos x) = 0 \\ \varphi(x, y, 0) = 0 \end{cases} \tag{5.10.1}$$

in the domain $-\pi < x < \pi$, $-\pi < y < \pi$ with periodic boundary conditions. We use a uniform rectangular mesh of 40×40 elements and the local Lax-Friedrichs flux (4.4.8)-(4.4.9). The solution at $t = 1$ is shown in Fig. 5.10.1, while the optimal control $w = \operatorname{sign}(\varphi_y)$ is shown in Fig. 5.10.2.

Notice that our method computes $\nabla\varphi$ as an independent variable. It is very desirable for those problems in which the most interesting features are contained in the first derivatives of φ, as in this optimal control problem.

We first use a uniform rectangular mesh of 50×50 elements and the local Lax-Friedrichs flux (4.4.8)-(4.4.9). The results of $\varepsilon = 0$ (pure convection) and $\varepsilon = 0.1$ are presented in Fig. 5.8.1 and Fig. 5.8.2, respectively. Notice that the surface at $t = 0$ is shifted downward by 0.35 in order to show the detail of the solution at $t = 0.3$.

Figure 5.8.1: Propagating surfaces, rectangular mesh, $\varepsilon = 0$.

Figure 5.8.2: Propagating surfaces, rectangular mesh, $\varepsilon = 0.1$.

Next we use a triangulation shown in Fig. 5.8.3. We refine the mesh around the center of domain where the solution develops discontinuous derivatives (for the $\varepsilon = 0$ case). There are 2146 triangles and 1108 nodes in this triangulation. The solutions are displayed in Fig. 5.8.4 and Fig. 5.8.5, respectively, for $\varepsilon = 0$ (pure convection) and $\varepsilon = 0.1$. Notice that we again shift the solution at $t = 0.0$ downward by 0.35 to show the detail of the solutions at later time.

Figure 5.8.3: Triangulation used for the propagating surfaces.

5.9 The Problem of a Propagating Surface on a Unit Disk

The equation is the same as (5.8.1) in the previous example, but it is solved on a unit disk $x^2 + y^2 < 1$ with an initial condition

$$\varphi(x, y, 0) = \sin\left(\frac{\pi(x^2 + y^2)}{2}\right)$$

and a Neumann type boundary condition $\nabla\varphi = 0$.

convergence towards an entropy solution. We show the numerical solution at $t = 1$ in Fig. 5.7.1.

Figure 5.7.1: Two dimensional Riemann problem, $H(u, v) = \sin(u + v)$, $t = 1$.

5.8 The Problem of a Propagating Surface

The controlling equation is

$$\begin{cases} \varphi_t - (1 - \varepsilon K)\sqrt{1 + \varphi_x^2 + \varphi_y^2} = 0, \qquad 0 < x < 1,\ 0 < y < 1 \\ \varphi(x, y, 0) = 1 - \frac{1}{4}(\cos(2\pi x) - 1)(\cos(2\pi y) - 1) \end{cases} \tag{5.8.1}$$

where K is the mean curvature defined by

$$K = -\frac{\varphi_{xx}(1 + \varphi_y^2) - 2\varphi_{xy}\varphi_x\varphi_y + \varphi_{yy}(1 + \varphi_x^2)}{(1 + \varphi_x^2 + \varphi_y^2)^{\frac{3}{2}}}, \tag{5.8.2}$$

and ε is a small constant. Periodic boundary condition is used.

This problem was studied in [43] by using the finite difference ENO schemes. We apply the discontinuous Galerkin method, with the second derivative terms handled by the local discontinuous Galerkin techniques presented and analyzed in [16], which amounts to solving the following system

$$\begin{cases} u_t - \left(\sqrt{1 + u^2 + v^2} + \varepsilon \frac{p(1+v^2) - 2quv + r(1+u^2)}{1+u^2+v^2}\right)_x = 0 \\ v_t - \left(\sqrt{1 + u^2 + v^2} + \varepsilon \frac{p(1+v^2) - 2quv + r(1+u^2)}{1+u^2+v^2}\right)_y = 0 \\ p - u_x = 0 \\ q - u_y = 0 \\ r - v_y = 0 \end{cases} \tag{5.8.3}$$

using the discontinuous Galerkin method. The details of the method, especially the choices of fluxes, which are important for stability, can be found in [16].

Notice that the exact steady state solution is the distance function to the inner boundary of domain when boundary condition is adequately prescribed. We take φ_0 equal to this distance function, and compute the time dependent problem to reach a steady state solution, using the exact steady state solution for the boundary conditions. Four symmetric elements near the outer boundary are updated by the exact steady state solution, all other elements are recovered from (4.4.13) by the shortest path to the nearest one of above four elements. The results are shown in Table 5.6.1. Also shown in Table 5.6.1 is the error (difference) between the numerical solution φ thus recovered, and the value of φ after another integration along a circular path (starting and ending at the same point in (4.4.13)). We can see that the difference is small with the correct order of accuracy, further indicating that the dependency of the recovered solution φ on the integration path is on the order of the truncation errors even for such problems with holes. Finally, the mesh with 1432 triangles and the solution with 5608 triangles are shown in Fig. 5.6.1.

Figure 5.6.1: The level set equation, P^2.

5.7 Two Dimensional Riemann Problem with a Non-convex Flux

Consider

$$\begin{cases} \varphi_t + \sin(\varphi_x + \varphi_y) = 0, & -1 < x < 1,\ -1 < y < 1 \\ \varphi(x, y, 0) = \pi(|y| - |x|) \end{cases} \tag{5.7.1}$$

For this example we use a uniform rectangular mesh with 40×40 elements. The local Lax-Friedrichs flux (4.4.8)-(4.4.9) is used. As was mentioned in section 5.3, we have found out that a nonlinear limiting [14], [15] is needed, for

Figure 5.4.3: Two dimension Burgers' equation, triangular mesh, t=$1.5/\pi^2$.

5.5 Two Dimensional Equation with a Non-convex Flux

Consider

$$\begin{cases} \varphi_t - \cos(\varphi_x + \varphi_y + 1) = 0, & -2 < x < 2,\ -2 < y < 2 \\ \varphi(x, y, 0) = -\cos\left(\frac{\pi(x+y)}{2}\right) \end{cases} \qquad (5.5.1)$$

with periodic boundary conditions.

For this example we use uniform rectangular meshes. The local Lax-Friedrichs flux (4.4.8)-(4.4.9) is used. The solution is smooth at $t = 0.5/\pi^2$. The accuracy of the numerical solution is shown in Table 5.5.1.

The solution has developed a discontinuous derivative at $t = 1.5/\pi^2$. Results with 40×40 elements are shown in Fig. 5.5.1.

Figure 5.5.1: Two dimensional, $H(u, v) = -\cos(u + v + 1), t = 1.5/\pi^2$.

5.6 The Level Set Equation in an Annulus

We now consider the equation

$$\begin{cases} \varphi_t + \mathrm{sign}(\varphi_0)(\sqrt{\varphi_x^2 + \varphi_y^2} - 1) = 0, & \frac{1}{2} < \sqrt{x^2 + y^2} < 1 \\ \varphi(x, y, 0) = \varphi_0(x, y) \end{cases} \qquad (5.6.1)$$

This problem is introduced in [56]. The solution φ to (5.6.1) has the same zero level set as φ_0, and the steady state solution is the distance function to that zero level curve. We use this problem to test the effects using various integration paths (4.4.13) when there is a hole in the region.

Table 5.4.1: Accuracy for 2D Burgers equation, uniform rectangular mesh, $t = 0.5/\pi^2$.

	P^1		P^2		P^3	
$N \times N$	L^1 error	order	L^1 error	order	L^1 error	order
10×10	8.09E-02	—	8.62E-03	—	3.19E-03	—
20×20	3.36E-02	1.268	1.72E-03	2.325	3.49E-04	3.192
40×40	1.48E-02	1.183	3.93E-04	2.130	6.64E-05	2.394
80×80	6.88E-03	1.105	9.74E-05	2.013	1.14E-05	2.542
160×160	3.31E-03	1.056	2.45E-05	1.991	1.68E-06	2.763

	P^1		P^2		P^3	
$N \times N$	L^∞ error	order	L^∞ error	order	L^∞ error	order
10×10	2.62E-01	—	3.56E-02	—	8.65E-03	—
20×20	1.14E-01	1.201	8.40E-03	2.083	1.16E-03	2.899
40×40	5.00E-02	1.189	2.02E-03	2.056	1.98E-04	2.551
80×80	2.39E-02	1.065	4.92E-04	2.038	3.13E-05	2.661
160×160	1.16E-02	1.043	1.21E-04	2.024	4.41E-06	2.827

decay in the same order as the errors. Thus the choice of integration path in recovering φ does not affect accuracy.

At $t = 1.5/\pi^2$, the solution has discontinuous derivatives. Fig. 5.4.1 is the graph of the numerical solution with 40×40 elements (uniform mesh).

Figure 5.4.1: Two dimension Burgers' equation, rectangular mesh, t=$1.5/\pi^2$.

Finally we use triangle based triangulation, the mesh with $h = \frac{1}{4}$ is shown in Fig. 5.4.2. The accuracy at $t = 0.5/\pi^2$ is shown in Table 5.4.4. Similar accuracy pattern is observed as in the rectangular case. The result at $t = 1.5/\pi^2$, when the derivative is discontinuous, is shown in Fig. 5.4.3.

Figure 5.4.2: Triangulation for two dimensional Burgers equation, $h = \frac{1}{4}$.

Figure 5.3.2: One dimension Riemann problem, Godunov flux, $H(u) = \frac{1}{4}(u^2 - 1)(u^2 - 4), t = 1$.

5.4 Two Dimensional Burgers' Equation

The equation is

$$\begin{cases} \varphi_t + \frac{(\varphi_x + \varphi_y + 1)^2}{2} = 0, & -2 < x < 2,\ -2 < y < 2 \\ \varphi(x, y, 0) = -\cos\left(\frac{\pi(x+y)}{2}\right) \end{cases} \tag{5.4.1}$$

with periodic boundary conditions.

We first use uniform rectangular meshes and the local Lax-Friedrichs flux (4.4.8)-(4.4.9). At $t = 0.5/\pi^2$, the solution is still smooth. The errors (computed at the center of the cells) and orders of accuracy are listed in Table 5.4.1. It seems that only k-th order of accuracy is achieved when φ is a piecewise polynomial of degree k.

Next, as in the one dimensional case, we use non-uniform rectangular meshes obtained from the tensor product of one dimensional nonuniform meshes (the meshes in two directions are independent). Again, we give the "real" L^2-errors computed by a 6×6 point Gaussian quadrature as well. The results are shown in Table 5.4.2.

The results in the previous two tables are obtained by updating the element at the left-lower corner with time, and then taking an integration path consisting of line segments starting from the corner and parallel to the x-axis first, then vertically to the point. To further address the issue of the dependency of the computed values of the solution φ on the integration path and starting point, we use another path which starts vertically, then parallelly with the x-axis to reach the point. In Table 5.4.3, we list the *difference* of two recovered solutions φ from these two different integration paths, for the non-uniform mesh cases. We can see that these differences are at the levels of local truncation errors and

Figure 5.2.1: One dimension non-convex, $H(u) = -\cos(u+1)$, $t = 1.5/\pi^2$.

5.3 The One Dimensional Riemann Problem with a Non-convex Flux

This time we consider

$$\begin{cases} \varphi_t + \frac{1}{4}(\varphi_x^2 - 1)(\varphi_x^2 - 4) = 0, & -1 < x < 1 \\ \varphi(x,0) = -2|x| \end{cases} \tag{5.3.1}$$

For this test problem, the discontinuous Galerkin method, as described in chapter 4 and applied in the previous two examples, fails to "open up" the initial single discontinuity in the derivative sufficiently to generate the correct entropy solution. We have found out that a nonlinear total variation bounded limiting, described in detail in [12] for this one dimensional case, is needed for convergence towards the entropy solution. This and the 2D Riemann problem in section 5.7 below are the only two examples in this paper in which we use the nonlinear limiting. We remark that for the finite difference schemes, such nonlinear limiting or the adaptive stencil in ENO is needed in most cases in order to enforce stability and to obtain non-oscillatory results.

Numerical results at $t = 1$ with 81 elements, using the local Lax-Friedrichs flux (4.3.9), is shown in Fig. 5.3.1. The results of using the Godunov flux is shown in Fig. 5.3.2. We can see that while for P^1, the results of using two different monotone fluxes are significantly different in resolution, this difference is greatly reduced for higher order of accuracy. In most of the high order cases, the simple local Lax-Friedrichs flux (4.3.9) should be good enough.

Figure 5.3.1: One dimension Riemann problem, local Lax-Friedrichs flux, $H(u) = \frac{1}{4}(u^2-1)(u^2-4), t = 1$.

Figure 5.1.1: One-dimensional Burgers' equation, t = $3.5/\pi^2$.

5.2 One Dimensional Equation with a Non-convex Flux

Consider

$$\begin{cases} \varphi_t - \cos(\varphi_x + 1) = 0, & -1 < x < 1 \\ \varphi(x,0) = -\cos(\pi x) \end{cases} \tag{5.2.1}$$

with periodic boundary conditions.

The local Lax-Friedrichs flux (4.3.9) and uniform meshes are used. At $t = 0.5/\pi^2$, the solution is still smooth. The accuracy of the numerical solution is listed in Table 5.2.1. We observe similar accuracy as in the previous example.

Table 5.2.1: Accuracy for 1D non-convex, $H(u) = -\cos(u+1)$, $t = 0.5/\pi^2$.

	P^1		P^2		P^3		P^4	
N	L^1 error	order	L^1 error	order	L^1 error	order	L^1 error	order
10	0.84E-01	—	0.10E-02	—	0.34E-03	—	0.24E-04	—
20	0.36E-01	1.23	0.15E-03	2.75	0.30E-04	3.49	0.13E-05	4.28
40	0.15E-01	1.26	0.21E-04	2.84	0.15E-05	4.33	0.59E-07	4.42
80	0.68E-02	1.14	0.27E-05	2.97	0.94E-07	4.00	0.21E-08	4.78

	P^1		P^2		P^3		P^4	
N	L^∞ error	order	L^∞ error	order	L^∞ error	order	L^∞ error	order
10	0.18E+00	—	0.15E-02	—	0.11E-02	—	0.99E-04	—
20	0.73E-01	1.31	0.27E-03	2.43	0.22E-03	2.35	0.13E-04	2.95
40	0.31E-01	1.24	0.47E-04	2.54	0.18E-04	3.63	0.59E-06	4.44
80	0.14E-01	1.16	0.85E-05	2.47	0.14E-05	3.75	0.26E-07	4.49

At $t = 1.5/\pi^2$, the solution has developed corner-like discontinuity in the derivative. The numerical result with 41 elements is shown in Fig. 5.2.1.

range $[-0.1h, 0.1h]$. In order to avoid possible superconvergence at cell centers, we also give the "real" L^2 error (computed by a 6-point Gaussian quadrature in each cell). The results are shown in Table 5.1.2.

Table 5.1.2: Accuracy for 1D Burgers equation (non-uniform mesh), $t = 0.5/\pi^2$.

	P^1		P^2		P^3		P^4	
N	L^2 error	order	L^2 error	order	L^2 error	order	L^2 error	order
10	0.74E+00	—	0.34E-02	—	0.32E-03	—	0.53E-04	—
20	0.34E+00	1.11	0.51E-03	2.76	0.24E-04	3.72	0.20E-05	4.71
40	0.15E+00	1.19	0.65E-04	2.96	0.17E-05	3.82	0.71E-07	4.84
80	0.67E-01	1.17	0.90E-05	2.86	0.13E-06	3.72	0.20E-08	5.15
160	0.31E-01	1.13	0.11E-05	3.02	0.81E-08	4.02	0.74E-10	4.76

	P^1		P^2		P^3		P^4	
N	L^1 error	order	L^1 error	order	L^1 error	order	L^1 error	order
10	0.53E+00	—	0.17E-02	—	0.23E-03	—	0.30E-05	—
20	0.24E+00	1.13	0.21E-03	3.05	0.14E-04	4.01	0.40E-06	2.89
40	0.11E+00	1.19	0.26E-04	2.99	0.78E-06	4.20	0.16E-07	4.65
80	0.47E-01	1.17	0.37E-05	2.82	0.47E-07	4.05	0.61E-09	4.70
160	0.21E-01	1.13	0.41E-06	3.16	0.27E-08	4.15	0.26E-10	4.56

	P^1		P^2		P^3		P^4	
N	L^∞ error	order	L^∞ error	order	L^∞ error	order	L^∞ error	order
10	0.62E+00	—	0.36E-02	—	0.69E-03	—	0.11E-04	—
20	0.29E+00	1.11	0.47E-03	2.94	0.61E-04	3.52	0.16E-05	2.81
40	0.13E+00	1.16	0.67E-04	2.80	0.47E-05	3.70	0.13E-06	3.64
80	0.58E-01	1.14	0.17E-04	2.01	0.62E-06	2.91	0.59E-08	4.45
160	0.27E-01	1.11	0.19E-05	3.11	0.31E-07	4.32	0.33E-09	4.17

At $t = 3.5/\pi^2$, the solution has developed a discontinuous derivative. In Fig. 5.1.1, we show the sharp corner-like numerical solution with 41 elements obtained with P^k for $k = 1, 2, 3, 4$ with a uniform mesh. Here and below, the solid line is the exact solution, the circles are numerical solutions (only one point per element is drawn).

5.1 One Dimensional Burgers' Equation

Consider

$$\begin{cases} \varphi_t + \frac{(\varphi_x+1)^2}{2} = 0, & -1 < x < 1 \\ \varphi(x,0) = -\cos(\pi x) \end{cases} \tag{5.1.1}$$

with periodic boundary conditions.

The local Lax-Friedrichs flux (4.3.9) is used. At $t = 0.5/\pi^2$, the solution is still smooth. We list the errors and the numerical orders of accuracy in Table 5.1.1 We observe that, except for the P^1 case which seems to be only first order, P^k for $k > 1$ seems to provide close to $(k+1)$-th order accuracy. The meshes used are all uniform, and errors are computed at the middle point of each interval.

Table 5.1.1: Accuracy for 1D Burgers equation (uniform mesh), $t = 0.5/\pi^2$.

	P^1		P^2		P^3		P^4	
N	L^1 error	order	L^1 error	order	L^1 error	order	L^1 error	order
10	0.17E+00	—	0.14E-02	—	0.21E-03	—	0.57E-05	—
20	0.78E-01	1.12	0.18E-03	2.92	0.13E-04	3.94	0.73E-06	2.97
40	0.35E-01	1.16	0.24E-04	2.97	0.75E-06	4.17	0.32E-07	4.52
80	0.16E-01	1.12	0.28E-05	3.08	0.43E-07	4.12	0.12E-08	4.79
160	0.76E-02	1.02	0.31E-06	3.19	0.25E-08	4.10	0.48E-10	4.59

	P^1		P^2		P^3		P^4	
N	L^∞ error	order	L^∞ error	order	L^∞ error	order	L^∞ error	order
10	0.29E+00	—	0.24E-02	—	0.69E-03	—	0.13E-04	—
20	0.13E+00	1.13	0.33E-03	2.88	0.61E-04	3.51	0.16E-05	2.99
40	0.58E-01	1.15	0.37E-04	3.15	0.58E-05	3.39	0.13E-06	3.64
80	0.27E-01	1.11	0.48E-05	2.97	0.38E-06	3.93	0.59E-08	4.44
160	0.13E-01	1.07	0.59E-06	3.00	0.23E-07	4.07	0.25E-09	4.57

To investigate the accuracy problem further, we use non-uniform meshes obtained by randomly shifting the cell boundaries in a uniform mesh in the

Chapter 5

Numerical Examples for Discontinuous Galerkin Methods for Hamilton-Jacobi Equations

In this chapter, we present numerical examples of one- and two- dimensional Hamilton-Jacobi equations using the methods developed in chapter 4. First we test the accuracies for smooth problems, then apply the methods to the problems with discontinuities in derivatives. In these examples, at least k-th order of accuracy is observed for smooth problems when k-th degree polynomials are used, and derivative singularities are resolved well without oscillations even without limiters.

$$T_{12} = \int_K \eta^3 \, dxdy, \quad T_{13} = \int_K \xi^4 \, dxdy, \quad T_{14} = \int_K \eta^4 \, dxdy;$$

and

$$q_1 = u_K^{(0)} T_0 + u_K^{(1)} T_1 + u_K^{(2)} T_3 + u_K^{(3)} T_2 + u_K^{(4)} T_4 + u_K^{(5)} T_5,$$

$$q_2 = u_K^{(0)} T_1 + u_K^{(1)} T_2 + u_K^{(2)} T_5 + u_K^{(3)} T_{11} + u_K^{(4)} T_7 + u_K^{(5)} T_6,$$

$$q_3 = v_K^{(0)} T_0 + v_K^{(1)} T_1 + v_K^{(2)} T_3 + v_K^{(3)} T_2 + v_K^{(4)} T_4 + v_K^{(5)} T_5,$$

$$q_4 = v_K^{(0)} T_3 + v_K^{(2)} T_4 + v_K^{(1)} T_5 + v_K^{(3)} T_6 + v_K^{(5)} T_7 + v_K^{(4)} T_{12},$$

$$\begin{aligned} q_5 = v_K^{(0)} T_1 \;&+\; v_K^{(1)} T_2 + u_K^{(0)} T_3 + u_K^{(2)} T_4 + (u_K^{(1)} + v_K^{(2)}) T_5 \\ &+\; (u_K^{(3)} + v_K^{(5)}) T_6 + (u_K^{(5)} + v_K^{(4)}) T_7 + v_K^{(3)} T_{11} + u_K^{(4)} T_{12}, \end{aligned}$$

$$q_6 = u_K^{(0)} T_2 + u_K^{(2)} T_6 + u_K^{(5)} T_8 + u_K^{(4)} T_{10} + u_K^{(1)} T_{11} + u_K^{(3)} T_{13},$$

$$q_7 = v_K^{(0)} T_4 + v_K^{(1)} T_7 + v_K^{(5)} T_9 + v_K^{(3)} T_{10} + v_K^{(2)} T_{12} + v_K^{(4)} T_{14},$$

$$\begin{aligned} q_8 = v_K^{(0)} T_2 \;&+\; 2u_K^{(0)} T_5 + (2u_K^{(1)} + v_K^{(2)}) T_6 + 2u_K^{(2)} T_7 + (2u_K^{(3)} + v_K^{(5)}) T_8 \\ &+\; 2u_K^{(4)} T_9 + (2u_K^{(5)} + v_K^{(4)}) T_{10} + v_K^{(1)} T_{11} + v_K^{(3)} T_{13}, \end{aligned}$$

$$\begin{aligned} q_9 = u_K^{(0)} T_4 \;&+\; 2v_K^{(0)} T_5 + 2v_K^{(1)} T_6 + (u_K^{(1)} + 2v_K^{(2)}) T_7 + 2v_K^{(3)} T_8 \\ &+\; (u_K^{(5)} + 2v_K^{(4)}) T_9 + (u_K^{(3)} + 2v_K^{(5)}) T_{10} + u_K^{(2)} T_{12} + u_K^{(4)} T_{14}. \end{aligned}$$

As in the quadratic case, we invert the above 9×9 matrix first, then start the iterations. For each iteration (each Runge-Kutta stage), multiply the inverse by the right side of (4.4.68) to get the expansion coefficients for φ, then project back (by differentiation) to (u, v) and continue for the next iteration (next R-K stage).

It turns out a 9×9 linear system:

$$\begin{pmatrix} T_0 & T_1 & 0 & 0 & T_3 & T_2 & 0 & T_5 & \frac{T_4}{2} \\ T_1 & T_2 & 0 & 0 & T_5 & T_{11} & 0 & T_6 & \frac{T_7}{2} \\ 0 & 0 & T_0 & T_3 & T_1 & 0 & T_4 & \frac{T_2}{2} & T_5 \\ 0 & 0 & T_3 & T_4 & T_5 & 0 & T_{12} & \frac{T_6}{2} & T_7 \\ T_3 & T_5 & T_1 & T_5 & (T_2+T_4) & T_6 & T_7 & (T_7+\frac{T_{11}}{2}) & (T_6+\frac{T_{12}}{2}) \\ T_2 & T_{11} & 0 & 0 & T_6 & T_{13} & 0 & T_8 & \frac{T_{10}}{2} \\ 0 & 0 & T_4 & T_{12} & T_7 & 0 & T_{14} & \frac{T_{10}}{2} & T_9 \\ 2T_5 & 2T_6 & T_2 & T_6 & (2T_7+T_{11}) & 2T_8 & T_{10} & (2T_{10}+\frac{T_{13}}{2}) & (T_8+T_9) \\ T_4 & T_7 & 2T_5 & 2T_7 & (2T_6+T_{12}) & T_{10} & 2T_9 & (T_8+T_9) & (2T_{10}+\frac{T_{14}}{2}) \end{pmatrix} \times$$

$$\begin{pmatrix} b_K \\ d_K \\ c_K \\ e_K \\ g_K \\ f_K \\ p_K \\ m_K \\ n_K \end{pmatrix} = \begin{pmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \\ q_6 \\ q_7 \\ q_8 \\ q_9 \end{pmatrix}; \tag{4.4.68}$$

where

$$T_0 = \text{area}(K), \quad T_1 = \int_K \xi \, dxdy, \quad T_2 = \int_K \xi^2 \, dxdy,$$

$$T_3 = \int_K \eta \, dxdy, \quad T_4 = \int_K \eta^2 \, dxdy, \quad T_5 = \int_K \xi\eta \, dxdy,$$

$$T_6 = \int_K \xi^2\eta \, dxdy, \quad T_7 = \int_K \xi\eta^2 \, dxdy, \quad T_8 = \int_K \xi^3\eta \, dxdy,$$

$$T_9 = \int_K \xi\eta^3 \, dxdy, \quad T_{10} = \int_K \xi^2\eta^2 \, dxdy, \quad T_{11} = \int_K \xi^3 \, dxdy,$$

and

$$b_1 = -h \sum_{e \in \partial K} \int_e \hat{H}_{1,e,K} \, d\Gamma,$$

$$b_2 = -h \sum_{e \in \partial K} \int_e \hat{H}_{1,e,K} \, \xi \, d\Gamma \; + \int_K H(u, \, v) \, dxdy,$$

$$b_3 = -h \sum_{e \in \partial K} \int_e \hat{H}_{1,e,K} \, \eta \, d\Gamma,$$

$$b_4 = -h \sum_{e \in \partial K} \int_e \hat{H}_{1,e,K} \, \xi^2 \, d\Gamma \; + 2 \int_K H(u, \, v) \, \xi \, dxdy,$$

$$b_5 = -h \sum_{e \in \partial K} \int_e \hat{H}_{1,e,K} \, \eta^2 \, d\Gamma,$$

$$b_6 = -h \sum_{e \in \partial K} \int_e \hat{H}_{1,e,K} \, \xi\eta \, d\Gamma \; + \int_K H(u, \, v) \, \eta \, dxdy;$$

$$c_1 = -h \sum_{e \in \partial K} \int_e \hat{H}_{2,e,K} \, d\Gamma,$$

$$c_2 = -h \sum_{e \in \partial K} \int_e \hat{H}_{2,e,K} \, \xi \, d\Gamma,$$

$$c_3 = -h \sum_{e \in \partial K} \int_e \hat{H}_{2,e,K} \, \eta \, d\Gamma \; + \int_K H(u, \, v) \, dxdy,$$

$$c_4 = -h \sum_{e \in \partial K} \int_e \hat{H}_{2,e,K} \, \xi^2 \, d\Gamma,$$

$$c_5 = -h \sum_{e \in \partial K} \int_e \hat{H}_{2,e,K} \, \eta^2 \, d\Gamma \; + 2 \int_K H(u, \, v) \, \eta \, dxdy,$$

$$c_6 = -h \sum_{e \in \partial K} \int_e \hat{H}_{2,e,K} \, \xi\eta \, d\Gamma \; + \int_K H(u, \, v) \, \xi \, dxdy,$$

$\hat{H}_{1,e,K}$ and $\hat{H}_{2,e,K}$ are (4.4.50)-(4.4.51).

We can solve (4.4.66)-(4.4.67) by the third order TVD Runge-Kutta method for

$$u_K^{(0)}, \, u_K^{(1)}, \, u_K^{(2)}, \, u_K^{(3)}, \, u_K^{(4)}, \, u_K^{(5)}, \; v_K^{(0)}, \, v_K^{(1)}, \, v_K^{(2)}, \, v_K^{(3)}, \, v_K^{(4)}, \, v_K^{(5)}.$$

Next we compute the coefficients $b_K, \, c_K, \, d_K, \, e_K, \, g_K, \, f_K, \, p_K, \, m_K, \, n_K$ by (4.4.59).

we have

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} & a_{16} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} & a_{26} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} & a_{36} \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} & a_{46} \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} & a_{56} \\ a_{61} & a_{62} & a_{63} & a_{64} & a_{65} & a_{66} \end{pmatrix} \cdot \frac{d}{dt} \begin{pmatrix} u_K^{(0)}(t) \\ u_K^{(1)}(t) \\ u_K^{(2)}(t) \\ u_K^{(3)}(t) \\ u_K^{(4)}(t) \\ u_K^{(5)}(t) \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \end{pmatrix}, \qquad (4.4.66)$$

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} & a_{16} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} & a_{26} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} & a_{36} \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} & a_{46} \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} & a_{56} \\ a_{61} & a_{62} & a_{63} & a_{64} & a_{65} & a_{66} \end{pmatrix} \cdot \frac{d}{dt} \begin{pmatrix} v_K^{(0)}(t) \\ v_K^{(1)}(t) \\ v_K^{(2)}(t) \\ v_K^{(3)}(t) \\ v_K^{(4)}(t) \\ v_K^{(5)}(t) \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \end{pmatrix}; \qquad (4.4.67)$$

where

$$a_{11} = \text{area}(K), \quad a_{22} = \int_K \xi^2\, dxdy, \quad a_{33} = \int_K \eta^2\, dxdy$$

$$a_{44} = \int_K \xi^4\, dxdy, \quad a_{55} = \int_K \eta^4\, dxdy, \quad a_{66} = \int_K \xi^2\eta^2\, dxdy$$

$$a_{12} = a_{21} = \int_K \xi\, dxdy,\ \ a_{13} = a_{31} = \int_K \eta\, dxdy,\ \ a_{14} = a_{41} = \int_K \xi^2\, dxdy,$$

$$a_{15} = a_{51} = \int_K \eta^2\, dxdy,\ \ a_{16} = a_{61} = \int_K \xi\eta\, dxdy,\ \ a_{23} = a_{32} = \int_K \xi\eta\, dxdy,$$

$$a_{24} = a_{42} = \int_K \xi^3\, dxdy,\ \ a_{25} = a_{52} = \int_K \xi\eta^2\, dxdy,\ \ a_{26} = a_{62} = \int_K \xi^2\eta\, dxdy,$$

$$a_{34} = a_{43} = \int_K \xi^2\eta\, dxdy,\ \ a_{35} = a_{53} = \int_K \eta^3\, dxdy,\ \ a_{36} = a_{63} = \int_K \xi\eta^2\, dxdy,$$

$$a_{45} = a_{54} = \int_K \xi^2\eta^2\, dxdy,\ \ a_{46} = a_{64} = \int_K \xi^3\eta\, dxdy,\ \ a_{56} = a_{65} = \int_K \xi\eta^3\, dxdy,$$

Invert the above matrix at the beginning of iterations, multiply the inverse by the right hand side to get the expansion coefficients for φ after each iteration, then project back (by differentiation) to (u, v) and continue for the next iteration.

(3). The cubic approximation (on triangles).

In element K, approximate φ by

$$\begin{aligned}\varphi(x,y,t) = a_K(t) &+ b_K(t)\xi + c_K(t)\eta + \frac{d_K(t)}{2}\xi^2 + \frac{e_K(t)}{2}\eta^2 + g_K(t)\xi\eta \\ &+ \frac{f_K(t)}{3}\xi^3 + \frac{p_K(t)}{3}\eta^3 + \frac{m_K(t)}{2}\xi^2\eta + \frac{n_K(t)}{2}\xi\eta^2; \end{aligned} \tag{4.4.61}$$

then

$$\begin{aligned} u(x,y,t) &= \frac{1}{h}\Big[u_K^{(0)}(t) + u_K^{(1)}(t)\xi + u_K^{(2)}(t)\eta + u_K^{(3)}(t)\xi^2 \\ &\quad + u_K^{(4)}(t)\eta^2 + u_K^{(5)}(t)\xi\eta\Big], \quad (x,y) \in K, \end{aligned} \tag{4.4.62}$$

$$\begin{aligned} v(x,y,t) &= \frac{1}{h}\Big[v_K^{(0)}(t) + v_K^{(1)}(t)\xi + v_K^{(2)}(t)\eta + v_K^{(3)}(t)\xi^2 \\ &\quad + v_K^{(4)}(t)\eta^2 + v_K^{(5)}(t)\xi\eta\Big], \quad (x,y) \in K. \end{aligned} \tag{4.4.63}$$

From

$$(u_t + H(u,v)_x,\, \mu) = 0, \quad \mu = 1,\, \xi,\, \eta,\, \xi^2,\, \eta^2,\, \xi\eta, \tag{4.4.64}$$

$$(v_t + H(u,v)_y,\, \nu) = 0, \quad \nu = 1,\, \xi,\, \eta,\, \xi^2,\, \eta^2,\, \xi\eta; \tag{4.4.65}$$

$\hat{H}_{1,e,K}$ and $\hat{H}_{2,e,K}$ are (4.4.50)-(4.4.51).

We can solve (4.4.57)-(4.4.58) for $u_K^{(0)}$, $u_K^{(1)}$, $u_K^{(2)}$, $v_K^{(0)}$, $v_K^{(1)}$, $v_K^{(2)}$ by the second order TVD Runge-Kutta method. Now we compute the coefficients b_K, c_K, d_K, e_K, g_K by

$$\| (\varphi_x - u)^2 + (\varphi_y - v)^2 \|_{L^1(K)} = \min_{\psi \in P^k(K)} \| (\psi_x - u)^2 + (\psi_y - v)^2 \|_{L^1(K)} . \quad (4.4.59)$$

Define $R \equiv \| (\varphi_x - u)^2 + (\varphi_y - v)^2 \|_{L^1(K)}$, R is a function of b_K, c_K, d_K, e_K, g_K. To solve (4.4.59), we take the partial derivatives of R with respect to b_K, c_K, d_K, e_K, g_K, and set

$$\frac{\partial R}{\partial b_K} = 0, \quad \frac{\partial R}{\partial c_K} = 0, \quad \frac{\partial R}{\partial d_K} = 0, \quad \frac{\partial R}{\partial e_K} = 0, \quad \frac{\partial R}{\partial g_K} = 0.$$

It turns out a 5×5 linear system:

$$\begin{pmatrix} T_0 & T_1 & 0 & 0 & T_3 \\ 0 & 0 & T_0 & T_3 & T_1 \\ T_1 & T_2 & 0 & 0 & T_5 \\ 0 & 0 & T_3 & T_4 & T_5 \\ T_3 & T_5 & T_1 & T_5 & T_2 + T_4 \end{pmatrix} \begin{pmatrix} b_K \\ d_K \\ c_K \\ e_K \\ g_K \end{pmatrix} = \begin{pmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \end{pmatrix}; \quad (4.4.60)$$

where

$$T_0 = \text{area}(K), \quad T_1 = \int_K \xi \, dxdy, \quad T_2 = \int_K \xi^2 \, dxdy,$$
$$T_3 = \int_K \eta \, dxdy, \quad T_4 = \int_K \eta^2 \, dxdy, \quad T_5 = \int_K \xi\eta \, dxdy;$$

and

$$q_1 = u_K^{(0)} T_0 + u_K^{(1)} T_1 + u_K^{(2)} T_3, \quad q_2 = v_K^{(0)} T_0 + v_K^{(1)} T_1 + v_K^{(2)} T_3,$$

$$q_3 = u_K^{(0)} T_1 + u_K^{(1)} T_2 + u_K^{(2)} T_5, \quad q_4 = v_K^{(0)} T_3 + v_K^{(2)} T_4 + v_K^{(1)} T_5,$$

$$q_5 = v_K^{(0)} T_1 + v_K^{(1)} T_2 + u_K^{(0)} T_3 + u_K^{(2)} T_4 + (u_K^{(1)} + v_K^{(2)}) T_5.$$

From

$$(u_t + H(u,v)_x,\, \mu) = 0, \quad \mu = 1,\, \xi,\, \eta, \tag{4.4.55}$$

$$(v_t + H(u,v)_y,\, \nu) = 0, \quad \nu = 1,\, \xi,\, \eta; \tag{4.4.56}$$

we have

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \cdot \frac{d}{dt} \begin{pmatrix} u_K^{(0)}(t) \\ u_K^{(1)}(t) \\ u_K^{(2)}(t) \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}, \tag{4.4.57}$$

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \cdot \frac{d}{dt} \begin{pmatrix} v_K^{(0)}(t) \\ v_K^{(1)}(t) \\ v_K^{(2)}(t) \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix}; \tag{4.4.58}$$

where

$$a_{11} = \text{area}(K), \quad a_{22} = \int_K \xi^2 \, dxdy, \quad a_{33} = \int_K \eta^2 \, dxdy$$

$$a_{12} = a_{21} = \int_K \xi \, dxdy,\ a_{13} = a_{31} = \int_K \eta \, dxdy,\ a_{23} = a_{32} = \int_K \xi\eta \, dxdy,$$

and

$$b_1 = -h \sum_{e \in \partial K} \int_e \hat{H}_{1,e,K} \, d\Gamma,$$

$$b_2 = -h \sum_{e \in \partial K} \int_e \hat{H}_{1,e,K} \, \xi \, d\Gamma \ + \int_K \ H(u,\, v) \, dxdy,$$

$$b_3 = -h \sum_{e \in \partial K} \int_e \hat{H}_{1,e,K} \, \eta \, d\Gamma;$$

$$c_1 = -h \sum_{e \in \partial K} \int_e \hat{H}_{2,e,K} \, d\Gamma,$$

$$c_2 = -h \sum_{e \in \partial K} \int_e \hat{H}_{2,e,K} \, \xi \, d\Gamma,$$

$$c_3 = -h \sum_{e \in \partial K} \int_e \hat{H}_{2,e,K} \, \eta \, d\Gamma \ + \int_K \ H(u,\, v) \, dxdy;$$

$$v(x,y,t) = \frac{c_{ij}(t)}{h}, \quad (x,y) \in D_{ij}.$$

From

$$(u_t + H(u,v)_x,\ \mu) = 0, \quad \mu = 1, \tag{4.4.46}$$

$$(v_t + H(u,v)_y,\ \nu) = 0, \quad \nu = 1, \tag{4.4.47}$$

we have

$$b'_{ij}(t) = -\frac{1}{h} \sum_{e\in\partial K} \int_e \hat{H}_{1,e,K}\, d\Gamma \tag{4.4.48}$$

$$c'_{ij}(t) = -\frac{1}{h} \sum_{e\in\partial K} \int_e \hat{H}_{2,e,K}\, d\Gamma \tag{4.4.49}$$

where

$$\hat{H}_{1,e,K} = H(\frac{u^{int(K)} + u^{ext(K)}}{2}, \frac{v^{int(K)} + v^{ext(K)}}{2})\, n_1 - \frac{\alpha}{2}(u^{ext(K)} - u^{int(K)}), \tag{4.4.50}$$

$$\hat{H}_{2,e,K} = H(\frac{u^{int(K)} + u^{ext(K)}}{2}, \frac{v^{int(K)} + v^{ext(K)}}{2})\, n_2 - \frac{\beta}{2}(v^{ext(K)} - v^{int(K)}), \tag{4.4.51}$$

and $n = (n_1,\ n_2)$ is the outer normal of side e.

(2). The quadratic approximation (on triangles).

In element K, approximate φ by

$$\varphi(x,y,t) = a_K(t) + b_K(t)\xi + c_K(t)\eta + \frac{d_K(t)}{2}\xi^2 + \frac{e_K(t)}{2}\eta^2 + g_K(t)\xi\eta; \tag{4.4.52}$$

then

$$u(x,y,t) = \frac{1}{h}\left[u_K^{(0)}(t) + u_K^{(1)}(t)\xi + u_K^{(2)}(t)\eta\right], \quad (x,y) \in K, \tag{4.4.53}$$

$$v(x,y,t) = \frac{1}{h}\left[v_K^{(0)}(t) + v_K^{(1)}(t)\xi + v_K^{(2)}(t)\eta\right], \quad (x,y) \in K. \tag{4.4.54}$$

$$p_{ij} = \frac{\Delta x \Delta y}{5(\Delta x)^2 + (\Delta y)^2}(2u_{ij}^{(4)} + 5v_{ij}^{(5)}),$$

while $b_{ij} = u_{ij}^{(0)}$, $c_{ij} = u_{ij}^{(1)}$, $d_{ij} = v_{ij}^{(0)}$, $e_{ij} = v_{ij}^{(2)}$, $m_{ij} = u_{ij}^{(3)}$, $n_{ij} = v_{ij}^{(4)}$.

4.4.6 Details for the triangular element

The application to triangular meshes is the most interesting feature of our discontinuous Galerkin method for HJ equations. Here, we use the monomial polynomials as our basis, this is just for the convenience to present the least-square procedure as well as the DG part. It seems to work fine for current methods (P^1, P^2 and P^3). However, we should point out that the monomial basis is known to lead to ill-conditioning (although we don't see such problem in all current methods), the orthogonal polynomials should be used for higher order methods. On the other hand, we need to take care of some linear transformations between different bases when orthogonal basis is used, since derivatives of orthogonal basis are involved.

By using the monomial basis, we present here the details for the cases $k = 1, 2, 3$.

For an arbitrary triangle element K, we use $\xi = \frac{(x-x_i)}{h}$, $\eta = \frac{(y-y_j)}{h}$ to construct base functions, where (x_i, y_j) is the barycenter of K, and h is some scale of K (we take h to be the square root of the area of K).

(1). The linear approximation (on triangles).

In element K, approximate φ by

$$\varphi(x, y, t) = a_K(t) + b_K(t)\xi + c_K(t)\eta \tag{4.4.45}$$

Then

$$u(x, y, t) = \frac{b_{ij}(t)}{h}, \quad (x, y) \in D_{ij},$$

$$\frac{dv_{ij}^{(4)}(t)}{dt} = -\frac{180}{\Delta x(\Delta y)^2}\cdot\left[\frac{(\Delta y)^2}{12}\left(2\int_{y_{j-\frac{1}{2}}}^{y_{j+\frac{1}{2}}}(H_{i+\frac{1}{2},\cdot}-H_{i-\frac{1}{2},\cdot})\,dy-\right.\right.$$
$$\left.\int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}}(G_{\cdot,j+\frac{1}{2}}-G_{\cdot,j-\frac{1}{2}})\,dx\right)+\int_{y_{j-\frac{1}{2}}}^{y_{j+\frac{1}{2}}}(G_{i+\frac{1}{2},\cdot}-G_{i-\frac{1}{2},\cdot})(y-y_j)^2\,dy$$
$$\left.-2\int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}}\int_{y_{j-\frac{1}{2}}}^{y_{j+\frac{1}{2}}}H(u,v)\,(y-y_j)\,dxdy\right], \qquad (4.4.43)$$

$$\frac{dv_{ij}^{(5)}(t)}{dt} = -\frac{144}{(\Delta x)^2\Delta y}\cdot\left[\frac{\Delta y}{2}\int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}}(H_{\cdot,j+\frac{1}{2}}+H_{\cdot,j-\frac{1}{2}})\,(x-x_i)\,dx+\right.$$
$$\frac{\Delta x}{2}\int_{y_{j-\frac{1}{2}}}^{y_{j+\frac{1}{2}}}(G_{i+\frac{1}{2},\cdot}+G_{i-\frac{1}{2},\cdot})\,(y-y_j)\,dy$$
$$\left.-\int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}}\int_{y_{j-\frac{1}{2}}}^{y_{j+\frac{1}{2}}}H(u,v)\,(x-x_i)\,dxdy\right]; \qquad (4.4.44)$$

where $H_{i\pm1/2,\cdot}, H_{\cdot,j\pm1/2}, G_{i\pm1/2,\cdot}, G_{\cdot,j\pm1/2}$ are from (4.4.19)-(4.4.24); and the equations for $u_{ij}^{(0)}, u_{ij}^{(1)}, u_{ij}^{(2)}, v_{ij}^{(0)}, v_{ij}^{(1)}, v_{ij}^{(2)}$ are same as (4.4.28)-(4.4.33).

We can solve (4.4.28)-(4.4.33) and (4.4.39)-(4.4.44) for

$$u_{ij}^{(0)},\ u_{ij}^{(1)},\ u_{ij}^{(2)},\ u_{ij}^{(3)},\ u_{ij}^{(4)},\ u_{ij}^{(5)},\ v_{ij}^{(0)},\ v_{ij}^{(1)},\ v_{ij}^{(2)},\ v_{ij}^{(3)},\ v_{ij}^{(4)},\ v_{ij}^{(5)}$$

by the third order TVD Runge-Kutta method; then by least-square, update

$$b_{ij},\ c_{ij},\ d_{ij},\ e_{ij},\ g_{ij},\ m_{ij},\ n_{ij},\ s_{ij},\ p_{ij}.$$

Since we use orthogonal polynomials, the least-square procedure (4.4.10) yields

$$g_{ij} = \frac{\Delta x\Delta y}{(\Delta x)^2+(\Delta y)^2}\,(u_{ij}^{(2)}+v_{ij}^{(1)}),$$

$$s_{ij} = \frac{\Delta x\Delta y}{(\Delta x)^2+5(\Delta y)^2}\,(5u_{ij}^{(5)}+2v_{ij}^{(3)}),$$

$$(\frac{y-y_j}{\Delta y})^2 - \frac{1}{12}, \ \frac{(x-x_i)(y-y_j)}{\Delta x\,\Delta y}; \qquad (4.4.38)$$

we have

$$\begin{aligned}\frac{du_{ij}^{(3)}(t)}{dt} &= -\frac{180}{(\Delta x)^2\Delta y}\cdot\Bigg[\frac{(\Delta x)^2}{12}\Bigg(2\int_{y_{j-\frac{1}{2}}}^{y_{j+\frac{1}{2}}}(H_{i+\frac{1}{2},\cdot}-H_{i-\frac{1}{2},\cdot})\,dy- \\ &\int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}}(G_{\cdot,j+\frac{1}{2}}-G_{\cdot,j-\frac{1}{2}})\,dx\Bigg)+\int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}}(G_{\cdot,j+\frac{1}{2}}-G_{\cdot,j-\frac{1}{2}})(x-x_i)^2\,dx \\ &-2\int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}}\int_{y_{j-\frac{1}{2}}}^{y_{j+\frac{1}{2}}}H(u,v)\,(x-x_i)\,dxdy\Bigg], \qquad (4.4.39)\end{aligned}$$

$$\begin{aligned}\frac{du_{ij}^{(4)}(t)}{dt} &= -\frac{180}{(\Delta y)^3}\cdot\Bigg[-\frac{(\Delta y)^2}{12}\Bigg(\int_{y_{j-\frac{1}{2}}}^{y_{j+\frac{1}{2}}}(H_{i+\frac{1}{2},\cdot}-H_{i-\frac{1}{2},\cdot})\,dy- \\ &2\int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}}(G_{\cdot,j+\frac{1}{2}}-G_{\cdot,j-\frac{1}{2}})\,dx\Bigg)+\int_{y_{j-\frac{1}{2}}}^{y_{j+\frac{1}{2}}}(H_{i+\frac{1}{2},\cdot}-H_{i-\frac{1}{2},\cdot})(y-y_j)^2\,dy\Bigg], \\ &\qquad (4.4.40)\end{aligned}$$

$$\begin{aligned}\frac{du_{ij}^{(5)}(t)}{dt} = -\frac{144}{\Delta x(\Delta y)^2} \quad\cdot\quad &\Bigg[\frac{\Delta x}{2}\int_{y_{j-\frac{1}{2}}}^{y_{j+\frac{1}{2}}}(H_{i+\frac{1}{2},\cdot}+H_{i-\frac{1}{2},\cdot})\,(y-y_j)\,dy+ \\ &\frac{\Delta y}{2}\int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}}(G_{\cdot,j+\frac{1}{2}}+G_{\cdot,j-\frac{1}{2}})\,(x-x_i)\,dx \\ &-\int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}}\int_{y_{j-\frac{1}{2}}}^{y_{j+\frac{1}{2}}}H(u,v)\,(y-y_j)\,dxdy\Bigg], \ (4.4.41)\end{aligned}$$

$$\begin{aligned}\frac{dv_{ij}^{(3)}(t)}{dt} &= -\frac{180}{(\Delta x)^3}\cdot\Bigg[-\frac{(\Delta x)^2}{12}\Bigg(\int_{y_{j-\frac{1}{2}}}^{y_{j+\frac{1}{2}}}(H_{i+\frac{1}{2},\cdot}-H_{i-\frac{1}{2},\cdot})\,dy- \\ &2\int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}}(G_{\cdot,j+\frac{1}{2}}-G_{\cdot,j-\frac{1}{2}})\,dx\Bigg)+\int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}}(H_{\cdot,j+\frac{1}{2}}-H_{\cdot,j-\frac{1}{2}})(x-x_i)^2\,dx\Bigg], \\ &\qquad (4.4.42)\end{aligned}$$

$$
\begin{aligned}
& u_{ij}^{(3)}(t)\,\left((\frac{x-x_i}{\Delta x})^2-\frac{1}{12}\right)+ \\
& u_{ij}^{(4)}(t)\,\left((\frac{y-y_j}{\Delta y})^2-\frac{1}{12}\right)+ \\
& \left. u_{ij}^{(5)}(t)\,\frac{(x-x_i)(y-y_j)}{\Delta x\,\Delta y}\right], \qquad (4.4.35)
\end{aligned}
$$

$$
\begin{aligned}
v(x,y,t) \;=\; & \frac{1}{\Delta y}\left[v_{ij}^{(0)}(t)+v_{ij}^{(1)}(t)\,\frac{x-x_i}{\Delta x}+v_{ij}^{(2)}(t)\,\frac{y-y_j}{\Delta y}+\right. \\
& v_{ij}^{(3)}(t)\,\left((\frac{x-x_i}{\Delta x})^2-\frac{1}{12}\right)+ \\
& v_{ij}^{(4)}(t)\,\left((\frac{y-y_j}{\Delta y})^2-\frac{1}{12}\right)+ \\
& \left. v_{ij}^{(5)}(t)\,\frac{(x-x_i)(y-y_j)}{\Delta x\,\Delta y}\right]; \qquad (4.4.36)
\end{aligned}
$$

where

$$u_{ij}^{(0)}(t)=b_{ij}(t),\; u_{ij}^{(1)}(t)=c_{ij}(t),\; u_{ij}^{(2)}(t)=g_{ij}(t),$$

$$u_{ij}^{(3)}(t)=m_{ij}(t),\; u_{ij}^{(4)}(t)=\frac{p_{ij}(t)}{2},\; u_{ij}^{(5)}(t)=s_{ij}(t);$$

$$v_{ij}^{(0)}(t)=d_{ij}(t),\; v_{ij}^{(1)}(t)=g_{ij}(t),\; v_{ij}^{(2)}(t)=e_{ij}(t),$$

$$v_{ij}^{(3)}(t)=\frac{s_{ij}(t)}{2},\; v_{ij}^{(4)}(t)=n_{ij}(t),\; v_{ij}^{(5)}(t)=p_{ij}(t).$$

From

$$
\begin{aligned}
(u_t+H(u,v)_x,\;\mu)=0, \qquad & \mu=1,\;\frac{x-x_i}{\Delta x},\;\frac{y-y_j}{\Delta y},\;(\frac{x-x_i}{\Delta x})^2-\frac{1}{12}, \\
& (\frac{y-y_j}{\Delta y})^2-\frac{1}{12},\;\frac{(x-x_i)(y-y_j)}{\Delta x\,\Delta y}, \qquad (4.4.37)
\end{aligned}
$$

and

$$(v_t+H(u,v)_y,\;\nu)=0, \qquad \nu=1,\;\frac{x-x_i}{\Delta x},\;\frac{y-y_j}{\Delta y},\;(\frac{x-x_i}{\Delta x})^2-\frac{1}{12},$$

$$
\left[\frac{\Delta y}{2}\int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}}(H_{\cdot,j+\frac{1}{2}}+H_{\cdot,j-\frac{1}{2}})\,dx+\int_{y_{j-\frac{1}{2}}}^{y_{j+\frac{1}{2}}}(G_{i+\frac{1}{2},\cdot}-G_{i-\frac{1}{2},\cdot})(y-y_j)\,dy\right.
$$

$$
\left.-\int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}}\int_{y_{j-\frac{1}{2}}}^{y_{j+\frac{1}{2}}}H(u,v)\,dxdy\right], \tag{4.4.33}
$$

where $H_{i\pm 1/2,\cdot}, H_{\cdot,j\pm 1/2}, G_{i\pm 1/2,\cdot}, G_{\cdot,j\pm 1/2}$ are from (4.4.19)-(4.4.24).

We can solve (4.4.28)-(4.4.33) for $u_{ij}^{(0)}, u_{ij}^{(1)}, u_{ij}^{(2)}, v_{ij}^{(0)}, v_{ij}^{(1)}, v_{ij}^{(2)}$ by the second order TVD Runge-Kutta method; then by least-square procedure, update b_{ij}, c_{ij}, d_{ij}, e_{ij}, g_{ij}. In fact, since we use orthogonal polynomials, the least-square procedure (4.4.10) yields

$$
g_{ij}=\frac{\Delta x\Delta y}{(\Delta x)^2+(\Delta y)^2}\,(u_{ij}^{(2)}+v_{ij}^{(1)}),
$$

while $b_{ij}=u_{ij}^{(0)}$, $d_{ij}=u_{ij}^{(1)}$, $c_{ij}=v_{ij}^{(0)}$, $e_{ij}=v_{ij}^{(2)}$.

(3). The cubic approximation for φ $(k=3)$.

$$
\begin{aligned}
\varphi(x,y,t)= \quad & a_{ij}(t)+b_{ij}(t)\,\frac{x-x_i}{\Delta x}+\frac{c_{ij}(t)}{2}\,(\frac{x-x_i}{\Delta x})^2+d_{ij}(t)\,\frac{y-y_j}{\Delta y}+\\
& \frac{e_{ij}(t)}{2}\,(\frac{y-y_j}{\Delta y})^2+g_{ij}(t)\,\frac{(x-x_i)(y-y_j)}{\Delta x\,\Delta y}+\\
& \frac{m_{ij}(t)}{3}\left((\frac{x-x_i}{\Delta x})^3-\frac{1}{4}\frac{x-x_i}{\Delta x}\right)+\\
& \frac{n_{ij}(t)}{3}\left((\frac{y-y_j}{\Delta y})^3-\frac{1}{4}\frac{y-y_j}{\Delta y}\right)+\\
& \frac{s_{ij}(t)}{2}\left((\frac{x-x_i}{\Delta x})^2-\frac{1}{12}\right)\,\frac{y-y_j}{\Delta y}+\\
& \frac{p_{ij}(t)}{2}\left((\frac{y-y_j}{\Delta y})^2-\frac{1}{12}\right)\,\frac{x-x_i}{\Delta x}
\end{aligned} \tag{4.4.34}
$$

Then

$$
u(x,y,t) \;=\; \frac{1}{\Delta x}\left[u_{ij}^{(0)}(t)+u_{ij}^{(1)}(t)\,\frac{x-x_i}{\Delta x}+u_{ij}^{(2)}(t)\,\frac{y-y_j}{\Delta y}+\right.
$$

$$(v_t + H(u,v)_y,\, \nu) = 0, \quad \nu = 1, \frac{x-x_i}{\Delta x}, \frac{y-y_j}{\Delta y}, \tag{4.4.27}$$

we have

$$\frac{du_{ij}^{(0)}(t)}{dt} = -\frac{1}{\Delta y}\left[\int_{y_{j-\frac{1}{2}}}^{y_{j+\frac{1}{2}}} (H_{i+\frac{1}{2},\cdot} - H_{i-\frac{1}{2},\cdot})\, dy + \int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} (G_{\cdot,j+\frac{1}{2}} - G_{\cdot,j-\frac{1}{2}})\, dx\right], \tag{4.4.28}$$

$$\begin{aligned}\frac{du_{ij}^{(1)}(t)}{dt} &= -\frac{12}{\Delta x \Delta y}\cdot \\ &\left[\frac{\Delta x}{2}\int_{y_{j-\frac{1}{2}}}^{y_{j+\frac{1}{2}}} (H_{i+\frac{1}{2},\cdot} + H_{i-\frac{1}{2},\cdot})\, dy + \int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} (G_{\cdot,j+\frac{1}{2}} - G_{\cdot,j-\frac{1}{2}})(x-x_i)\, dx\right. \\ &\left. - \int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \int_{y_{j-\frac{1}{2}}}^{y_{j+\frac{1}{2}}} H(u,v)\, dxdy\right],\end{aligned} \tag{4.4.29}$$

$$\begin{aligned}\frac{du_{ij}^{(2)}(t)}{dt} &= -\frac{12}{(\Delta y)^2}\cdot \\ &\left[\int_{y_{j-\frac{1}{2}}}^{y_{j+\frac{1}{2}}} (H_{i+\frac{1}{2},\cdot} - H_{i-\frac{1}{2},\cdot})(y-y_j)\, dy + \int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} (G_{\cdot,j+\frac{1}{2}} + G_{\cdot,j-\frac{1}{2}})\, dx\right],\end{aligned} \tag{4.4.30}$$

$$\frac{dv_{ij}^{(0)}(t)}{dt} = -\frac{1}{\Delta x}\left[\int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} (H_{\cdot,j+\frac{1}{2}} - H_{\cdot,j-\frac{1}{2}})\, dx + \int_{y_{j-\frac{1}{2}}}^{y_{j+\frac{1}{2}}} (G_{i+\frac{1}{2},\cdot} - G_{i-\frac{1}{2},\cdot})\, dy\right], \tag{4.4.31}$$

$$\begin{aligned}\frac{dv_{ij}^{(1)}(t)}{dt} &= -\frac{12}{(\Delta x)^2}\cdot \\ &\left[\int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} (H_{\cdot,j+\frac{1}{2}} - H_{\cdot,j-\frac{1}{2}})(x-x_i)\, dx + \int_{y_{j-\frac{1}{2}}}^{y_{j+\frac{1}{2}}} (G_{i+\frac{1}{2},\cdot} + G_{i-\frac{1}{2},\cdot})\, dy\right],\end{aligned} \tag{4.4.32}$$

$$\frac{dv_{ij}^{(2)}(t)}{dt} = -\frac{12}{\Delta x \Delta y}\cdot$$

$$G_{\cdot,j+\frac{1}{2}}(x) = -\frac{\beta}{2}(u_{\cdot,j+1} - u_{\cdot,j}), \tag{4.4.21}$$

$$G_{i+\frac{1}{2},\cdot}(y) = -\frac{\alpha}{2}(v_{i+1,\cdot} - v_{i,\cdot}); \tag{4.4.22}$$

and

$$u_{i,\cdot} = u(x_i, y, t), \quad u_{\cdot,j} = u(x, y_j, t), \tag{4.4.23}$$

$$v_{i,\cdot} = v(x_i, y, t), \quad v_{\cdot,j} = v(x, y_j, t); \tag{4.4.24}$$

α is an estimate of the (local) maximum of $\left|\frac{\partial H}{\partial u}\right|$, β is an estimate of the (local) maximum of $\left|\frac{\partial H}{\partial v}\right|$,

(2). The quadratic approximation for ϕ $(k = 2)$.

We approximate φ in D_{ij} by

$$\begin{aligned} \varphi(x,y,t) &= a_{ij}(t) + b_{ij}(t)\,\frac{x-x_i}{\Delta x} + \frac{d_{ij}(t)}{2}\,(\frac{x-x_i}{\Delta x})^2 + c_{ij}(t)\,\frac{y-y_j}{\Delta y} \\ &+ \frac{e_{ij}(t)}{2}\,(\frac{y-y_j}{\Delta y})^2 + g_{ij}(t)\,\frac{(x-x_i)(y-y_j)}{\Delta x\,\Delta y}. \end{aligned} \tag{4.4.25}$$

Then

$$u(x,y,t) = \frac{1}{\Delta x}\left[u_{ij}^{(0)}(t) + u_{ij}^{(1)}(t)\,\frac{x-x_i}{\Delta x} + u_{ij}^{(2)}(t)\,\frac{y-y_j}{\Delta y}\right], \quad (x,y) \in D_{ij},$$

$$v(x,y,t) = \frac{1}{\Delta y}\left[v_{ij}^{(0)}(t) + v_{ij}^{(1)}(t)\,\frac{x-x_i}{\Delta x} + v_{ij}^{(2)}(t)\,\frac{y-y_j}{\Delta y}\right], \quad (x,y) \in D_{ij};$$

where

$$u_{ij}^{(0)}(t) = b_{ij}(t),\ u_{ij}^{(1)}(t) = d_{ij}(t),\ u_{ij}^{(2)}(t) = g_{ij}(t),$$

$$v_{ij}^{(0)}(t) = c_{ij}(t),\ v_{ij}^{(1)}(t) = g_{ij}(t),\ v_{ij}^{(2)}(t) = e_{ij}(t).$$

From

$$(u_t + H(u,v)_x,\ \mu) = 0, \quad \mu = 1, \frac{x-x_i}{\Delta x}, \frac{y-y_j}{\Delta y}, \tag{4.4.26}$$

4.4.5 Details for the rectangular element

We now describe in details the above discontinuous Galerkin method for (4.4.1) on a rectangular element for the cases $k = 1, 2, 3$.

(1). The linear approximation for φ ($k = 1$).

Consider the cell $D_{ij} = [x_{i-1/2}, x_{i+1/2}] \times [y_{j-1/2}, y_{j+1/2}]$ with center (x_i, y_j), and let $\Delta x = x_{i+1/2} - x_{i-1/2}$, $\Delta y = y_{j+1/2} - y_{j-1/2}$. We approximate φ by

$$\varphi(x,y,t) = a_{ij}(t) + b_{ij}(t)\,\frac{(x-x_i)}{\Delta x} + c_{ij}(t)\,\frac{(y-y_j)}{\Delta y}. \qquad (4.4.14)$$

Then

$$u(x,y,t) = \frac{b_{ij}(t)}{\Delta x}, \quad (x,y) \in D_{ij},$$

$$v(x,y,t) = \frac{c_{ij}(t)}{\Delta y}, \quad (x,y) \in D_{ij}.$$

From

$$(u_t + H(u,v)_x,\ \mu) = 0, \quad \mu = 1, \qquad (4.4.15)$$

$$(v_t + H(u,v)_y,\ \nu) = 0, \quad \nu = 1, \qquad (4.4.16)$$

where $(\cdot,\cdot)$ is the inner product in $L^2(D_{ij})$, we have

$$b'_{ij}(t) = -\frac{1}{\Delta y}\Big(\int_{y_{j-\frac{1}{2}}}^{y_{j+\frac{1}{2}}} (H_{i+\frac{1}{2},\cdot} - H_{i-\frac{1}{2},\cdot})\, dy + \int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} (G_{\cdot,j+\frac{1}{2}} - G_{\cdot,j-\frac{1}{2}})\, dx\Big), \qquad (4.4.17)$$

$$c'_{ij}(t) = -\frac{1}{\Delta x}\Big(\int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} (H_{\cdot,j+\frac{1}{2}} - H_{\cdot,j-\frac{1}{2}})\, dx + \int_{y_{j-\frac{1}{2}}}^{y_{j+\frac{1}{2}}} (G_{i+\frac{1}{2},\cdot} - G_{i-\frac{1}{2},\cdot})\, dy\Big); \qquad (4.4.18)$$

where

$$H_{i+\frac{1}{2},\cdot}(y) = H\Big(\frac{u_{i+1,\cdot} + u_{i,\cdot}}{2}, \frac{v_{i+1,\cdot} + v_{i,\cdot}}{2}\Big) - \frac{\alpha}{2}(u_{i+1,\cdot} - u_{i,\cdot}), \qquad (4.4.19)$$

$$H_{\cdot,j+\frac{1}{2}}(x) = H\Big(\frac{u_{\cdot,j+1} + u_{\cdot,j}}{2}, \frac{v_{\cdot,j+1} + v_{\cdot,j}}{2}\Big) - \frac{\beta}{2}(v_{\cdot,j+1} - v_{\cdot,j}), \qquad (4.4.20)$$

path. However this difference is on the level of truncation errors and does not affect the order of accuracy. See examples in the next chapter. It is important here that φ is a single function and u and v are just its x and y derivatives. Otherwise the second approach would be questionable in effectively recovering φ.

In the next chapter we will only show numerical results obtained with the second approach, see the remarks in the previous section for the one dimensional case.

We remark that the procedure discussed above is easily implemented in any triangulation, e.g., for both rectangles and triangles.

4.4.4 On the integration path

In the second approach to recover the missing constant, we need, for each element, to pick another element with complete information, and use (4.4.13) with some integration path. In general, points A and B in (4.4.13) are the centers of elements. We update the mean of φ by (4.4.12) in several elements on the boundary of the computational domain, say four corners in a rectangular domain, these elements are called the target elements. For each element with B as its center, pick a target element which is the nearest and can be reached without crossing a derivative discontinuity (if possible); point A is the center of this target element.

To find an integration path between A and B, start from neighbors of the element with point A, pick one neighbor whose center C is the nearest to B; connect A and C with a line segment as the first part of our integration path; substitute A with C and proceed inductively until B is reached. This procedure is particularly useful for triangular meshes.

square:

$$||(\varphi_x - u)^2 + (\varphi_y - v)^2||_{L^1(K)} = \min_{\psi \in P^k(K)} ||(\psi_x - u)^2 + (\psi_y - v)^2||_{L^1(K)}. \quad (4.4.10)$$

We will give more details on the above procedure for the rectangular and triangular elements in the following subsections.

4.4.3 Recovery of the constant

The previous subsections determine φ for each element up to a constant, we have a scheme only for $\nabla\varphi$. The missing constant can again be obtained in one of the following two ways:

1. By requiring that

$$\int_K (\varphi_t + H(\varphi_x, \varphi_y))\, v\, dxdy \;=\; 0, \qquad (4.4.11)$$

for all $v \in V_h^0$ and for all $K \in \mathcal{T}_h$, that is,

$$\int_K (\varphi_t + H(\varphi_x, \varphi_y))\; dxdy \;=\; 0, \qquad \forall K \in \mathcal{T}_h\,; \qquad (4.4.12)$$

2. By using (4.4.12) to update only one (or a few) elements, e.g., the corner element(s), then after reaching the final time T of the computation, use

$$\varphi(B,t) = \varphi(A,t) + \int_A^B (\varphi_x\, dx + \varphi_y\, dy)\,, \quad t = T \qquad (4.4.13)$$

to determine the missing constant. The path should be taken to avoid crossing a derivative discontinuity, if possible.

We remark again that, in the second approach, the recovered values of φ are dependent upon the choice of the starting point A as well as the integration

3. Conservation:

$$\hat{H}_{i,e,K}\left((\nabla\varphi)^{int(K)},(\nabla\varphi)^{ext(K)}\right)=-\hat{H}_{i,e,K'}\left((\nabla\varphi)^{int(K')},(\nabla\varphi)^{ext(K')}\right).$$

where $K \cap K' = e$.

We will again mainly use the simple (local) Lax-Friedrichs flux

$$\hat{H}_{1,e,K}((u,v)^-,(u,v)^+)=H\left(\frac{u^-+u^+}{2},\frac{v^-+v^+}{2}\right)n_1-\frac{1}{2}\alpha(u^+-u^-) \quad (4.4.8)$$

and

$$\hat{H}_{2,e,K}((u,v)^-,(u,v)^+)=H\left(\frac{u^-+u^+}{2},\frac{v^-+v^+}{2}\right)n_2-\frac{1}{2}\beta(v^+-v^-) \quad (4.4.9)$$

where $\alpha = \max_{u,v}|\frac{\partial H(u,v)}{\partial u}|$ and $\beta = \max_{u,v}|\frac{\partial H(u,v)}{\partial v}|$, with the maximum being taken over the relevant (local) range.

Notice that (4.4.5)-(4.4.6) are exactly the discontinuous Galerkin method for the conservation law system (4.4.2) satisfied by the derivatives $(u,v)=\nabla\varphi$. For a rectangular mesh and for $k=1$ this recovers the (local) Lax-Friedrichs monotone scheme [44] for (4.4.1) if we identify $u_{ij}=\frac{\varphi_{i+1,j}-\varphi_{i-1,j}}{2\Delta x}$ and $v_{ij}=\frac{\varphi_{i,j+1}-\varphi_{i,j-1}}{2\Delta y}$; that is, we can deduce the discontinuous Galerkin scheme ($k=1$) for $(u,v)=\nabla\varphi$ from the monotone scheme for (4.4.1).

4.4.2 A least-square procedure

(4.4.5)-(4.4.6) have more equations than the number of degrees of freedoms (an over-determined system) for $k>1$, thus a least square solution is needed. In practice, the least square procedure is performed as follows: we first evolve (4.4.2) for one time step (one inner stage for high order Runge-Kutta methods), using the discontinuous Galerkin method (4.4.5)-(4.4.6) with $u=\varphi_x$ and $v=\varphi_y$; then φ at the next time level (stage) is obtained (up to a constant) by least

where $P^k(K)$ is again the set of all polynomials of degree at most k on the cell K.

Using (4.4.2), we propose a discontinuous Galerkin method for (4.4.1) as follows: find $\varphi \in V_h^k$, such that

$$\int_K \varphi_{xt} v \, dxdy - \int_K H(\varphi_x, \varphi_y) v_x \, dxdy + \sum_{e \in \partial K} \int_e \hat{H}_{1,e,K} \, v \, d\Gamma = 0 \tag{4.4.5}$$

and

$$\int_K \varphi_{yt} v \, dxdy - \int_K H(\varphi_x, \varphi_y) v_y \, dxdy + \sum_{e \in \partial K} \int_e \hat{H}_{2,e,K} \, v \, d\Gamma = 0 \tag{4.4.6}$$

for all $v \in V_h^{k-1}$ and all $K \in \mathcal{T}_h$, *in a least square sense.* Here the numerical flux is

$$\hat{H}_{i,e,K} = \hat{H}_{i,e,K} \left((\nabla\varphi)^{int(K)}, (\nabla\varphi)^{ext(K)} \right), \qquad i = 1, 2 \tag{4.4.7}$$

where the superscript $int(K)$ implies that the value is taken from within the element K, and the superscript $ext(K)$ implies that the value is taken from outside the element K and within the neighboring element K' sharing the edge e with K. The flux (4.4.7) satisfies the following properties:

1. $\hat{H}_{i,e,K}$ is Lipschitz continuous with respect to all its arguments;

2. Consistency:

$$\hat{H}_{1,e,K}(\nabla\varphi, \nabla\varphi) = (H(\nabla\varphi), \, 0) \cdot n = H(\nabla\varphi) \, n_1,$$

$$\hat{H}_{2,e,K}(\nabla\varphi, \nabla\varphi) = (0, \, H(\nabla\varphi)) \cdot n = H(\nabla\varphi) \, n_2,$$

   where $n = (n_1, n_2)$ is the unit outward normal to the edge e of the element K;

4.4 Two Dimensional Case

We consider in this section the case of two spatial dimensions. The algorithm in more spatial dimensions is similar. This time, the scalar Hamilton-Jacobi equation

$$\begin{cases} \varphi_t + H(\varphi_x, \varphi_y) = 0, \\ \varphi(x, y, 0) = \varphi^0(x, y) \end{cases} \tag{4.4.1}$$

is in some sense equivalent to the following conservation law *system*

$$\begin{cases} u_t + H(u, v)_x = 0, \\ v_t + H(u, v)_y = 0, \\ (u, v)(x, y, 0) = (u, v)^0(x, y), \end{cases} \tag{4.4.2}$$

if we identify

$$(u, v) = (\varphi_x, \varphi_y). \tag{4.4.3}$$

For example, a vanishing viscosity solution of (4.4.1) corresponds, via (4.4.3), to a vanishing viscosity solution of (4.4.2), and vice versa [35]. However, (4.4.2) is not a strictly hyperbolic system, which may cause problems in its numerical solution if we treat u and v as independent variables. Instead, we would like to still use φ as our solution variable (a polynomial) and take its derivatives as u and v.

4.4.1 The basic two-dimensional scheme

Assume that (4.4.1) is solved in the domain Ω, which has a triangulation $\mathcal{T}_h$ consisting of triangles or general polygons of maximum size (diameter) h, with the following approximation space

$$V_h^k = \left\{ v : v|_K \in P^k(K), \, \forall K \in \mathcal{T}_h \right\}, \tag{4.4.4}$$

By Lemma 1.1 in [40], $\varphi(x,t) = \int^{x} u(y,t)\,dt + c(t)$ is the viscosity solution of (4.3.1).

Lemma 2.1. [34]. The following L^2 stability result for the derivative φ_x holds for the discontinuous Galerkin method defined above, of any order of accuracy k applied to any nonlinear Hamilton-Jacobi equation (4.3.1):

$$\frac{d}{dt}\int_a^b \varphi_x^2\,dx \leq 0. \tag{4.3.17}$$

□

For a finite interval $[a,b]$, (4.3.17) trivially implies TVB (total variation bounded) property for the numerical solution φ:

$$TV(\varphi) = \int_a^b |\varphi_x|\,dx \leq \sqrt{b-a}\sqrt{\int_a^b \left(\frac{d}{dx}\varphi^0(x)\right)^2\,dx}. \tag{4.3.18}$$

This is a rather strong stability result, considering that it applies even if the derivative of the solution φ_x develops discontinuities, no limiter has been added to the numerical scheme, and the scheme can be of arbitrary high order in accuracy. It also implies convergence of at least a subsequence of the numerical solution φ when $h \to 0$. However, this stability result is not strong enough to imply that the limit solution is the viscosity solution of (4.3.1).

4.3.5 The issue of viscosity solutions

Consider (4.3.2) again with $u = \varphi_x$ in mind. If we have a total-variation boundedness in the discontinuous Galerkin method for (4.3.2) (which is the case if the slope limiting process is used, see [12]), then due to (4.3.17), $u^h \to u$, as $h \downarrow 0$, and u is the entropy solution of (4.3.2).

Now by Lebesgue theorem,

$$\int^x u^h(y,t)\,dy \to \int^x u(y,t)\,dy. \tag{4.3.19}$$

4.3.3 The temporal discretization

Up to now we have only described the spatial discretization and have left the time variable t continuous (the method of lines approach). As in the WENO scheme, time discretization is by the TVD (total variation diminishing) high order Runge-Kutta methods developed in [48], see also [25]. We repeat the description of the second and third order versions used in this chapter as follows: for solving the method of lines ODE

$$\varphi_t = L(\varphi), \tag{4.3.14}$$

the second order TVD Runge-Kutta method is given by

$$\begin{aligned} \varphi^{(1)} &= \varphi^n + \Delta t L(\varphi^n) \\ \varphi^{n+1} &= \frac{1}{2}\varphi^n + \frac{1}{2}\varphi^{(1)} + \frac{1}{2}\Delta t L(\varphi^{(1)}), \end{aligned} \tag{4.3.15}$$

and the third order TVD Runge-Kutta method is given by

$$\begin{aligned} \varphi^{(1)} &= \varphi^n + \Delta t L(\varphi^n) \\ \varphi^{(2)} &= \frac{3}{4}\varphi^n + \frac{1}{4}\varphi^{(1)} + \frac{1}{4}\Delta t L(\varphi^{(1)}) \\ \varphi^{n+1} &= \frac{1}{3}\varphi^n + \frac{2}{3}\varphi^{(2)} + \frac{2}{3}\Delta t L(\varphi^{(2)}). \end{aligned} \tag{4.3.16}$$

These Runge-Kutta methods will also be used for the two dimensional case discussed in the next section.

4.3.4 The stability issue

About the stability of the method proposed above, we can quote the following result of Jiang and Shu [34]. Here we assume compact support or periodic boundary condition for φ.

1. By requiring that

$$\int_{I_j} (\varphi_t + H(\varphi_x))\, v\, dx \;=\; 0, \qquad j = 1, ..., N, \tag{4.3.11}$$

for all $v \in V_h^0$, that is,

$$\int_{I_j} (\varphi_t + H(\varphi_x))\; dx \;=\; 0, \qquad j = 1, ..., N. \tag{4.3.12}$$

2. By using (4.3.12) to update only one (or a few) elements, e.g., the left-most element I_1, then after reaching the final time T of the computation, use

$$\varphi(x_j, t) = \varphi(x_1, t) + \int_{x_1}^{x_j} \varphi_x(x, t)\, dx, \quad t = T \tag{4.3.13}$$

to determine the missing constant for the cell I_j.

We remark that, in the second approach, the recovered values of φ are dependent upon the choice of the starting point x_1. However this difference is on the level of truncation errors and does not affect the order of accuracy. See numerical examples in the next chapter. Both approaches are used in our numerical experiments. They perform similarly for smooth problems, with the first approach giving slightly better results. However, it is our numerical experience that, when there are singularities in the derivatives, the first approach will often produce dents and bumps when the integral path in time passes through the singularities at some earlier time. The philosophy of using the second approach is that one could update only a few elements whose time integral paths do not cross derivative singularities. The numerical results shown in the next chapter are obtained with the second approach.

can then be defined as follows: find $\varphi \in V_h^k$, such that

$$\int_{I_j} \varphi_{xt} v \, dx - \int_{I_j} H(\varphi_x) v_x \, dx + \hat{H}_{j+\frac{1}{2}} v^-_{j+\frac{1}{2}} - \hat{H}_{j-\frac{1}{2}} v^+_{j-\frac{1}{2}} = 0, \quad j = 1, \cdots, N, \tag{4.3.7}$$

for all $v \in V_h^{k-1}$. Here

$$\hat{H}_{j+\frac{1}{2}} = \hat{H}\left((\varphi_x)^-_{j+\frac{1}{2}}, (\varphi_x)^+_{j+\frac{1}{2}} \right) \tag{4.3.8}$$

is a monotone flux, i.e. $\hat{H}$ is non-decreasing in the first argument and non-increasing in the second, symbolically $\hat{H}(\uparrow, \downarrow)$, is Lipschitz continuous in both arguments, and is consistent with H, i.e. $\hat{H}(u, u) = H(u)$. We will mainly use the simple (local) Lax-Friedrichs flux

$$\hat{H}(u^-, u^+) = H\left(\frac{u^- + u^+}{2} \right) - \frac{1}{2}\alpha\,(u^+ - u^-) \tag{4.3.9}$$

where $\alpha = \max_u |H'(u)|$ with the maximum taken over the range covered by u^- and u^+. For other monotone fluxes, e.g. the Godunov flux:

$$\hat{H}(u^-, u^+) = \begin{cases} \min_{u^- \le u \le u^+} H(u) & \text{if } u^- \le u_+, \\ \max_{u^- \ge u \ge u^+} H(u) & \text{if } u^- > u_+; \end{cases} \tag{4.3.10}$$

see [44].

4.3.2 Recovery of the constant

Notice that the method described above is exactly the discontinuous Galerkin method for the conservation law equation (4.3.2) satisfied by the derivative $u = \varphi_x$. This only determines φ for each element up to a constant, since it is only a scheme for φ_x. The missing constant can be obtained in one of the following two ways:

4.3 One Dimensional HJ Equations

In one space dimension (4.1.1) becomes

$$\begin{cases} \varphi_t + H(\varphi_x) = 0, \\ \varphi(x, 0) = \varphi^0(x). \end{cases} \tag{4.3.1}$$

This is a relatively easy case because (4.3.1) is equivalent to the conservation law

$$\begin{cases} u_t + H(u)_x = 0, \\ u(x, 0) = u^0(x), \end{cases} \tag{4.3.2}$$

if we identify $u = \varphi_x$, and differentiate (4.3.1) with respect to x.

4.3.1 The basic scheme

Assume that (4.3.1) is solved in the interval $a \le x \le b$ which is divided into the following cells:

$$a = x_{\frac{1}{2}} < x_{\frac{3}{2}} < \cdots < x_{N+\frac{1}{2}} = b, \tag{4.3.3}$$

we denote

$$I_j = (x_{j-\frac{1}{2}}, x_{j+\frac{1}{2}}), \quad x_j = \frac{1}{2}\left(x_{j-\frac{1}{2}} + x_{j+\frac{1}{2}}\right), \tag{4.3.4}$$

$$h_j = x_{j+\frac{1}{2}} - x_{j-\frac{1}{2}}, \quad h = \max_j h_j, \tag{4.3.5}$$

and define the following approximation space

$$V_h^k = \left\{ v : v|_{I_j} \in P^k(I_j), j = 1, ..., N \right\}. \tag{4.3.6}$$

Here $P^k(I_j)$ is the set of all polynomials of degree at most k on the cell I_j.

Now we use (4.3.2) to construct a scheme for (4.3.1). A k-th order discontinuous Galerkin scheme for the one dimensional Hamilton-Jacobi equation (4.3.1)

they can be solved by the Runge-Kutta methods. Recall that $h_{j+1/2} = h(u^-_{j+1/2}, u^+_{j+1/2})$, where $u^\pm_{j+1/2} = u^h(x^\pm_{j+1/2}, t)$, we have

$$u^-_{j+1/2} = u_j^{(0)} + 6u_j^{(1)} + 30u_j^{(2)}, \tag{4.2.11}$$

$$u^+_{j-1/2} = u_j^{(0)} - 6u_j^{(1)} + 30u_j^{(2)}. \tag{4.2.12}$$

To achieve total variation stability, some local projection limiter is needed to modify $u^\pm_{j+1/2}$ in (4.2.11) and (4.2.12). Here is a brief description of such procedure, see [12] for more details.

Write

$$u^-_{j+1/2} = u_j^{(0)} + \tilde{u}_j, \quad u^+_{j-1/2} = u_j^{(0)} - \hat{u}_j; \tag{4.2.13}$$

We need to modify $\tilde{u}_j$ and $\hat{u}_j$ by local projection limiter to achieve total variation stability.

$$\tilde{u}_j^{(\mathrm{mod})} = m(\tilde{u}_j, \Delta_+ u_j^{(0)}, \Delta_- u_j^{(0)}), \quad \hat{u}_j^{(\mathrm{mod})} = m(\hat{u}_j, \Delta_+ u_j^{(0)}, \Delta_- u_j^{(0)}); \tag{4.2.14}$$

where m is a minmod function defined by:

$$m(a_1, a_2, \cdots, a_n) = \begin{cases} s \cdot \min_{1\le i\le n} |a_i| & \text{if } \mathrm{sign}(a_1) = \cdots = \mathrm{sign}(a_n) = s, \\ 0 & \text{otherwise.} \end{cases} \tag{4.2.15}$$

We then replace $u^\pm_{j+1/2}$ by $u^{\pm(\mathrm{mod})}_{j+1/2}$ as,

$$u^{-(\mathrm{mod})}_{j+1/2} = u_j^{(0)} + \tilde{u}_j^{(\mathrm{mod})}, \quad u^{+(\mathrm{mod})}_{j-1/2} = u_j^{(0)} - \hat{u}_j^{(\mathrm{mod})}. \tag{4.2.16}$$

Since the function u^h is discontinuous at point $x_{j+1/2}$, we replace the nonlinear flux $f(u_{j+1/2})$ by a *numerical average* flux $h_{j+1/2} = h(u^-_{j+1/2}, u^+_{j+1/2})$, where $u^\pm_{j+1/2} = u^h(x^\pm_{j+1/2}, t)$, and $h(\cdot,\cdot)$ is a Lipschitz continuous *monotone* flux, i.e., $h(\cdot,\cdot)$ is nondecreasing in its first argument and nonincreasing in its second argument. We require $h(u,u) = f(u)$ for the consistency to keep the accuracy. Some of such monotone fluxes can be found in [12].

Now we can choose some basis functions for V_h^k, and get a system of ODEs for the degrees of freedom which can be solved by the high-order TVD Runge-Kutta methods in [49]. As an example, we take $k = 2$, and choose a local orthogonal basis over I_j for V_h^k, $\{\varphi_0^{(j)}, \varphi_1^{(j)}, \varphi_2^{(j)}\}$, each $\varphi_l^{(j)}$ is supported in I_j only,

$$\varphi_0^{(j)}(x) = 1, \quad \varphi_1^{(j)}(x) = x - x_j, \quad \varphi_2^{(j)}(x) = (x - x_j)^2 - \frac{1}{12}\Delta x_j^2, \tag{4.2.5}$$

Define the degrees of freedom as

$$u_j^{(l)} = u_j^{(l)}(t) = \frac{1}{\Delta x_j^{l+1}} \int_{I_j} u^h(x,t)\varphi_l^{(j)}(x)dx, \qquad l = 0, 1, 2, \tag{4.2.6}$$

then

$$u^h(x,t) = u_j^{(0)}\varphi_0^{(j)}(x) + \frac{12}{\Delta x_j} u_j^{(1)}\varphi_1^{(j)}(x) + \frac{180}{\Delta x_j^2} u_j^{(2)}\varphi_2^{(j)}(x), \quad \text{for } x \in I_j \tag{4.2.7}$$

Next, taking $v^h = \varphi_0^{(j)}$, $\varphi_1^{(j)}$, $\varphi_2^{(j)}$, we have

$$\frac{d}{dt}u_j^{(0)} + \frac{1}{\Delta x_j}(h_{j+1/2} - h_{j-1/2}) = 0, \tag{4.2.8}$$

$$\frac{d}{dt}u_j^{(1)} + \frac{1}{2\Delta x_j}(h_{j+1/2} + h_{j-1/2}) - \frac{1}{\Delta x_j^2}\int_{I_j} f(u^h(x,t))\, dx = 0, \tag{4.2.9}$$

$$\frac{d}{dt}u_j^{(2)} + \frac{1}{6\Delta x_j}(h_{j+1/2} - h_{j-1/2}) - \frac{2}{\Delta x_j^3}\int_{I_j} f(u^h(x,t))\,(x - x_j)\, dx = 0; \tag{4.2.10}$$

4.2 The RKDG for Conservation Laws

We only take a one-dimensional scalar equation to describe the method. This section is basically a sketch of [12]. For more general cases and details, see [11, 12, 13, 14, 15].

Consider

$$\begin{cases} u_t + f(u)_x = 0, & x \in \Re, \quad t > 0 \\ u(x,0) = u^0(x), & x \in \Re. \end{cases} \tag{4.2.1}$$

To discretize (4.2.1) in space, we first need to define a partition of the real line. Let $I_j = (x_{j-1/2},\, x_{j+1/2})$, $I = \bigcup I_j$ be such a partition. Denote $\Delta x_j = x_{j+1/2} - x_{j-1/2}$ and $h = \sup_j \Delta x_j$.

As a finite element method, we seek an approximation u_h to u such that for each fixed time $t \geq 0$, $u_h(t)$ belongs to the following finite-dimensional function space

$$V_h = V_h^k \equiv \{v \in L^1(\Re) : v|_{I_j} \in P^k(I_j)\}, \tag{4.2.2}$$

where $P^k(I_j)$ is the space of polynomials of degree at most k on I_j. Note that u_h can be discontinuous across the interfaces $x_{j+1/2}$, this property makes the following weak formulation explicit in a semidiscrete form.

In order to determine the $k+1$ degrees of freedom of u^h, we multiply the equation (4.2.1) by an arbitrary function $v^h \in V_h^k$ and integrate over each I_j and replace the exact solution u by its approximation u^h:

$$\int_{I_j} (\partial_t u^h(x,t))\, v_h(x)\, dx + \int_{I_j} (\partial_x f(u^h(x,t)))\, v_h(x)\, dx = 0, \quad \forall v_h \in V_h^k. \tag{4.2.3}$$

Denoting $u_{j\pm1/2} = u^h(x_{j\pm1/2}, t)$, and get, after a simple formal integration by parts for the second integral in (4.2.3),

$$\begin{aligned} \int_{I_j} (\partial_t u^h(x,t))\, v_h(x)\, dx &- \int_{I_j} f(u^h(x,t))\, \partial_x v_h(x)\, dx \\ &+ f(u_{j+1/2})\, v^h(x_{j+1/2}^-) - f(u_{j-1/2})\, v^h(x_{j-1/2}^+) = 0, \end{aligned} \tag{4.2.4}$$

element depends only on the information of itself and its immediate neighbors, regardless of the order of accuracy. This is in contrast with high order finite volume schemes which must use wide stencils for the high order reconstruction. This compactness is responsible for the efficient parallel implementation of the method, see, e.g. [7].

For details of the Runge-Kutta discontinuous Galerkin method, see the references listed above and also the review paper of Cockburn [10]. This method can also be generalized to solve problems containing higher derivatives, [16], which is important for some of our applications.

It is well known that the Hamilton-Jacobi equation (4.1.1) is closely related to the conservation law (4.1.2), in fact in one space dimension $d = 1$ they are equivalent if one takes $u = \varphi_x$ and differentiates (4.1.1) with respect to x. It is thus not surprising that many successful numerical methods for the Hamilton-Jacobi equation (4.1.1) are adapted from those for the conservation law (4.1.2). The high order finite difference ENO methods in [43] and [44] are such examples.

In the following sections, we will use such relation between HJ equations and conservation laws (CLs) to construct a class of discontinuous Galerkin finite element methods for HJ equations from the RKDG finite element method for solving CLs. By differentiating the HJ, we obtain a system of hyperbolic equations; then we apply RKDG to the system with a least-square idea to ensure the consistency and with some tricks to recover the constant lost during the differentiation (of HJ). We will first describe the simple case of one-dimensional HJ equations, then the two-dimensional case will be discussed in more details. Numerical examples of one and two spatial dimensions will be presented in the next chapter.

tensions were studied in [2]. A second order ENO type finite volume scheme is developed in [37]. However, higher order finite volume schemes face the problem of reconstruction on arbitrary triangulation, which is quite complicated. See, e.g., [1]. More recently, a continuous finite element method for solving (4.1.1) is proposed in [5].

The Runge-Kutta discontinuous Galerkin (RKDG) method [11, 12, 13, 14, 15] is a method devised to numerically solve the conservation law (CL):

$$\begin{cases} u_t + f_1(u)_{x_1} + \cdots + f_d(u)_{x_d} = 0, \\ u(x,0) = u^0(x) \end{cases} \tag{4.1.2}$$

The method has the following attractive properties:

- It can be designed for any order of accuracy in space and time. In fact, p-version or spectral element type version can be designed [42];

- It is an explicit method, thus efficient for solving the hyperbolic problem (4.1.2). No global linear or nonlinear systems must be solved;

- It combines the flexibility of finite element methods in the easy handling of complicated geometry, with the high resolution property for discontinuous solutions of finite difference and finite volume methods through monotone fluxes or approximate Riemann solvers applied at the element interfaces and limiters;

- It has nice stability properties: a local cell entropy inequality for the square entropy can be proven [33] for general triangulation for any scalar nonlinear conservation laws (4.1.2) in any spatial dimensions and for any order of accuracy, without the need of nonlinear limiters. This implies nonlinear L^2 stability and entropy consistency even for discontinuous solutions;

- The method is highly compact: the evolution of the information in any

4.1 Some Technical Backgrounds

In this chapter, we consider the numerical solutions of nonlinear Hamilton-Jacobi (HJ) equations

$$\begin{cases} \varphi_t + H(\varphi_{x_1}, ..., \varphi_{x_d}) = 0, \\ \varphi(x, 0) = \varphi^0(x). \end{cases} \tag{4.1.1}$$

As is well known, the solutions to (4.1.1) are Lipschitz continuous but may have discontinuous derivatives, regardless of the smoothness of the initial condition $\varphi^0(x)$. The non-uniqueness of such solutions also necessitates the definition of viscosity solutions, to single out a unique, practically relevant solution. See Crandall and Lions [17].

A widely applied class of numerical schemes for (4.1.1) is the class of finite difference schemes. In [18], Crandall and Lions proved the convergence of monotone finite difference schemes to the viscosity solutions of (4.1.1). Unfortunately, monotone schemes are at most first order accurate, measured by local truncation errors in smooth regions of the solution. In Osher and Sethian [43] and Osher and Shu [44], a class of high order essentially non-oscillatory (ENO) schemes were introduced, which were adapted from the methodologies for hyperbolic conservation laws [29], [49], [50]. The numerical results in [43] and [44] indicate convergence to the viscosity solutions of (4.1.1) with high order accuracy in smooth regions, and with sharp resolution of the discontinuous derivatives. Recently, weighted ENO (WENO) schemes for conservation laws [41], [34] have also been adapted to the Hamilton-Jacobi equations (4.1.1), [32].

Finite difference methods require a structured mesh, hence are difficult to apply to complicated geometry or for adaptive mesh refinements. Finite volume schemes, which are based on arbitrary triangulation, are thus attractive in such cases. First order monotone type finite volume schemes and their second ex-

Chapter 4

Discontinuous Galerkin Finite Element Methods for Hamilton-Jacobi Equations

In this chapter, we construct discontinuous Galerkin finite element methods for solving the nonlinear Hamilton-Jacobi equations. These methods are based on the Runge-Kutta discontinuous Galerkin finite element method for solving conservation laws. The methods have the flexibility of treating complicated geometry by using arbitrary triangulation, can achieve high order accuracy with a local, compact stencil, and are suited for efficient parallel implementation.

Figure 3.3.6: Forward step problem: 30 contours from 0.32 to 6.15.

Figure 3.3.7: Forward step problem: 30 contours from 0.5 to 12.

Figure 3.3.8: Triangulation for the double Mach reflection.

Figure 3.3.9: Double Mach reflection: $h = \frac{1}{50}$, density.

Figure 3.3.10: Double Mach reflection: $h = \frac{1}{100}$, density.

Figure 3.3.11: Double Mach reflection: $h = \frac{1}{200}$, density.

Figure 3.3.12: Double Mach reflection: $h = \frac{1}{200}$, density (blowed-up region).

Figure 3.3.13: Double Mach reflection: $h = \frac{1}{400}$, density.

Figure 3.3.14: Double Mach reflection: $h = \frac{1}{400}$, density (blowed-up region).

Figure 3.3.4: Riemann problems of Euler equations: velocity.

Figure 3.3.5: Triangulations for the forward step problem: part near the corner.

cases of $\Delta x = \Delta y = \frac{1}{200}$ and $\Delta x = \Delta y = \frac{1}{400}$, we present both the picture of whole region ($[0,3] \times [0,1]$) and a blowed-up region around the double Mach stems. All pictures are the density contours with 30 equally spaced contour lines from 1.5 to 21.5. We can clearly see that the fourth order scheme captures the complicated flow structure under the triple Mach stem much better than the third order scheme, and the characteristic version is much less oscillatory than the component version for the fourth order scheme. We refer to [15] for similar results obtained with discontinuous Galerkin methods.

Figure 3.3.2: Riemann problems of Euler equations: density.

Figure 3.3.3: Riemann problems of Euler equations: pressure.

near the corner. The third mesh has a triangle size of $\Delta x = \Delta y = \frac{1}{80}$ away from the corner, and it is one-quarter of that near the corner. The last mesh has a triangle size of $\Delta x = \Delta y = \frac{1}{160}$ away from the corner, and it is one-half of that near the corner. Fig. 3.3.6 is the contour picture for the density at time $t = 4.0$. Fig. 3.3.7 is the contour picture for the pressure. It is clear that with more triangles near the corner the artifacts from the singularity decrease significantly.

3.3.3 Double Mach reflection

This problem is also from [58]. We solve the Euler equations (3.1.3) in a computational domain of $[0,4] \times [0,1]$. A reflecting wall lies at the bottom of the domain starting from $x = \frac{1}{6}$. Initially a right-moving Mach 10 shock is located at $x = \frac{1}{6}, y = 0$, making a 60° angle with the x-axis. The reflective boundary condition is used at the wall, while for the rest of the bottom boundary (the part from $x = 0$ to $x = \frac{1}{6}$), the exact post-shock condition is imposed. At the top boundary, the flow values are set to describe the exact motion of the Mach 10 shock. The results shown are at $t = 0.2$.

We test both the third and the fourth order schemes. Four triangle sizes are used, they are roughly equal to rectangular element cases of $\Delta x = \Delta y = \frac{1}{50}$, $\Delta x = \Delta y = \frac{1}{100}$, $\Delta x = \Delta y = \frac{1}{200}$, and $\Delta x = \Delta y = \frac{1}{400}$ respectively. For the third order scheme, we use both uniform triangular mesh (equilateral triangles) and locally refined triangular mesh (the refined region has the above triangle sizes, Fig. 3.3.8 shows the region $[0,2] \times [0,1]$ of such a mesh of $\Delta x = \Delta y = \frac{1}{50}$ locally). For the fourth order, we use uniform triangular mesh only. For the

Table 3.2.6: Accuracy for 2D Euler equation of smooth vortex evolution, non-uniform meshes, WENO schemes.

	P^1 (3rd order)				P^2 (4th order)			
h	L^1 error	order	L^∞ error	order	L^1 error	order	L^∞ error	order
$h_0/2$	2.12E-02	—	3.33E-01	—	1.84E-02	—	2.14E-01	—
$h_0/4$	1.28E-02	0.73	2.27E-01	0.55	2.80E-03	2.69	3.43E-02	2.64
$h_0/8$	3.84E-03	1.74	6.85E-02	1.73	2.12E-04	3.72	6.57E-03	2.38
$h_0/16$	8.32E-04	2.21	3.02E-02	1.18	1.09E-05	4.28	5.91E-04	3.48
$h_0/32$	1.26E-04	2.72	5.64E-03	2.42	3.76E-07	4.86	1.97E-05	4.91
$h_0/64$	1.16E-05	3.44	6.19E-04	3.19	1.66E-08	4.50	6.78E-07	4.86

Figure 3.3.1: Mesh for the Riemann problems.

To illustrate the reconstruction near the boundary, take the wall as an example. We flip the triangulation inside the computational domain to the outside as a reflection to the wall. Set all physical cell averages in each triangle outside according to those in its "image" inside, which, for this case, is to set same ρ, p, u and take negative v from inside. Then do all reconstructions outside same as inside computational domain.

The corner of the step is a singularity point. [58] uses an assumption of nearly steady flow in the region near the corner to fix this singularity. In this paper, we do not modify our method near the corner, instead we adopt the same technique as the one used in [15], namely refining the mesh near the corner and using the same scheme in the whole domain.

We use the third order scheme for this problem. Four meshes have been used, see Fig. 3.3.5. For the first mesh, the triangle size away from the corner is roughly equal to a rectangular element case of $\Delta x = \Delta y = \frac{1}{40}$, while it is one-quarter of that near the corner. For the second mesh, the triangle size away from the corner is the same as in the first mesh, but it is one-eighth of that

Table 3.2.5: Accuracy for 2D Euler equation of smooth vortex evolution, uniform meshes, WENO schemes.

	P^1 (3rd order)				P^2 (4th order)			
h	L^1 error	order	L^∞ error	order	L^1 error	order	L^∞ error	order
1	1.87E-02	—	2.95E-01	—	1.30E-02	—	2.05E-01	—
1/2	1.01E-02	0.89	2.09E-01	0.50	2.50E-03	2.38	4.45E-02	2.49
1/4	2.78E-03	1.86	6.37E-02	1.71	1.79E-04	3.80	3.29E-03	3.76
1/8	6.47E-04	2.10	3.05E-02	1.06	6.92E-06	4.69	1.96E-04	4.07
1/16	8.74E-05	2.89	8.14E-03	1.91	2.03E-07	5.09	4.95E-06	5.31
1/32	7.10E-06	3.62	5.66E-04	3.85	7.83E-09	4.70	1.96E-07	4.66

Figure 3.2.5: 2D vortex evolution: fourth order WENO scheme.

$$(\rho_R, u_R, p_R) = (0.5, 0, 0.571).$$

Density at $t = 0.26$ is shown in the last 3 plots in Fig. 3.3.2; pressure is shown in the last 3 plots in Fig. 3.3.3; velocity in the x-direction is shown in the last 3 plots in Fig. 3.3.4.

We can observe a better resolution of the fourth order scheme over the third order one, and also a less oscillatory result from the characteristic version of the fourth order scheme over the component version.

3.3.2 A Mach 3 wind tunnel with a step

This problem is from [58]. We solve the Euler equations (3.1.3) in a wind tunnel of 1 length unit wide and 3 length units long. The step is 0.2 length units high and is located 0.6 length units from the left end of the tunnel. Initially, a right-going Mach 3 flow is used. Reflective boundary conditions are applied along the walls of the tunnel and inflow and outflow boundary conditions are used at the entrance and the exit.

Table 3.2.4: Accuracy for 2D Burgers' equation, non-uniform meshes, WENO schemes.

	P^1 (3rd order)				P^2 (4th order)			
h	L^1 error	order	L^∞ error	order	L^1 error	order	L^∞ error	order
$h_0/2$	2.01E-02	—	9.16E-02	—	4.18E-03	—	2.376-02	—
$h_0/4$	3.85E-03	2.38	1.80E-02	2.35	2.90E-04	3.85	2.61E-03	3.18
$h_0/8$	5.79E-04	2.73	3.39E-03	2.41	1.85E-05	3.97	1.92E-04	3.77
$h_0/16$	5.34E-05	3.44	3.55E-04	3.26	1.18E-06	3.97	1.35E-05	3.83
$h_0/32$	5.12E-06	3.38	2.95E-05	3.59	7.45E-08	3.99	9.99E-07	3.76
$h_0/64$	5.82E-07	3.14	4.23E-06	2.80	4.67E-09	4.00	7.56E-08	3.72

Figure 3.2.4: 2D vortex evolution: third order WENO scheme.

We consider the following Riemann type initial conditions:

$$(\rho, u, p) = \begin{cases} (\rho_L, u_L, p_L) & \text{if } x \le 0 \\ (\rho_R, u_R, p_R) & \text{if } x > 0. \end{cases}$$

The first test case is Sod's problem [54]. The initial data is

$$(\rho_L, u_L, p_L) = (1, 0, 1),$$

$$(\rho_R, u_R, p_R) = (0.125, 0, 0.1).$$

Density at $t = 0.40$ is shown in the first 3 plots in Fig. 3.3.2; pressure is shown in the first 3 plots in Fig. 3.3.3; velocity in the x-direction is shown in the first 3 plots in Fig. 3.3.4.

The second test case is the Riemann problem proposed by Lax [38]:

$$(\rho_L, u_L, p_L) = (0.445, 0.698, 3.528),$$

Table 3.2.3: Accuracy for 2D Burgers' equation, uniform meshes, WENO schemes.

	P^1 (3rd order)				P^2 (4th order)			
h	L^1 error	order	L^∞ error	order	L^1 error	order	L^∞ error	order
2/5	2.76E-02	—	8.18E-02	—	8.64E-03	—	2.106-02	—
1/5	4.63E-03	2.58	1.20E-02	2.77	6.05E-04	3.84	1.73E-03	3.60
1/10	6.97E-04	2.73	2.16E-03	2.47	3.94E-05	3.94	1.18E-04	3.87
1/20	7.12E-05	3.29	1.90E-04	3.51	2.50E-06	3.98	7.42E-06	3.99
1/40	7.63E-06	3.22	2.36E-05	3.01	1.57E-07	3.99	4.63E-07	4.00
1/80	9.08E-07	3.07	2.96E-06	3.00	9.83E-09	4.00	2.89E-08	4.00

Figure 3.2.3: 2D Burgers' equation: $t = 5/\pi^2$, 4th order with splitting.

3.3 Non-smooth Problems for 2D Euler Equations

In this section, we apply the third and the fourth order triangular WENO schemes to several model problems for two-dimensional Euler equations.

3.3.1 Riemann problems

The two-dimensional triangular WENO methods are applied to one dimensional shock tube problems. We consider the solution of the Euler equations (3.1.3) in a domain of $[-1, 1] \times [0, 0.2]$ with a triangulation of 101 vertices in the x-direction and 11 vertices in the y-direction. The velocity in the y-direction is zero, and periodic boundary condition is used in the y-direction. Part of the mesh is shown in Fig. 3.3.1. The pictures shown below are obtained by extracting the data along the central cut line for 101 equally spaced points.

Table 3.2.2: Accuracy for 2D linear equation, non-uniform meshes, WENO schemes.

	P^1 (3rd order)				P^2 (4th order)			
h	L^1 error	order	L^∞ error	order	L^1 error	order	L^∞ error	order
$h_0/2$	2.79E-01	—	5.28E-01	—	1.77E-02	—	6.41E-02	—
$h_0/4$	8.43E-02	1.73	2.32E-01	1.19	8.85E-04	4.32	3.07E-03	4.38
$h_0/8$	2.53E-02	1.74	7.47E-02	1.64	4.08E-05	4.44	1.43E-04	4.42
$h_0/16$	2.24E-03	3.50	1.14E-02	2.71	1.82E-06	4.49	6.37E-06	4.49
$h_0/32$	1.18E-04	4.25	6.83E-04	4.06	8.95E-08	4.35	3.36E-07	4.25
$h_0/64$	6.21E-06	4.25	3.15E-05	4.44	4.92E-09	4.19	2.00E-08	4.07

Figure 3.2.2: 2D Burgers' equation: $t = 5/\pi^2$, non-uniform mesh.

Fig. 3.2.4 and Fig. 3.2.5 are the results for the long time evolution of the vortex. These results are similar to those obtained with the linear schemes in Fig. 3.1.3 and Fig. 3.1.4.

Figure 3.1.4: 2D vortex evolution: fourth order linear scheme.

Table 3.2.1: Accuracy for 2D linear equation, uniform meshes, WENO schemes.

	P^1 (3rd order)				P^2 (4th order)			
h	L^1 error	order	L^∞ error	order	L^1 error	order	L^∞ error	order
2/5	2.66E-01	—	4.30E-01	—	1.38E-02	—	2.94E-02	—
1/5	8.11E-02	1.71	1.93E-01	1.16	1.80E-03	2.94	2.74E-03	3.42
1/10	2.65E-02	1.62	6.16E-02	1.65	8.87E-05	4.34	1.46E-04	4.23
1/20	2.68E-03	3.31	8.77E-03	2.81	4.34E-06	4.35	7.11E-06	4.36
1/40	1.44E-04	4.22	4.88E-04	4.17	2.30E-07	4.24	3.71E-07	4.26
1/80	8.05E-06	4.16	2.40E-05	4.35	1.34E-08	4.10	2.12E-08	4.13

To demonstrate the application for shock computations, we continue the the above calculation to $t = 5/\pi^2$ when discontinuities develop. Fig. 3.2.1 is the result for $h = 1/20$ of a uniform mesh. Fig. 3.2.2 is the result for $h = h_0/16$ of a non-uniform mesh. We can see that the shock transitions are sharp and non-oscillatory.

We also apply the splitting technique for negative weights (see section 2.9.4) to the fourth order scheme. Fig. 3.2.3 is the result for a rather non-uniform mesh. The left is the mesh, the right is the result at $t = 5/\pi^2$.

Figure 3.2.1: 2D Burgers' equation: $t = 5/\pi^2$, uniform mesh.

3.2.3 Two-dimensional vortex evolution problem

It is same as defined in section 3.1.3, (3.1.3). The accuracy results are shown in Table 3.2.5 for the uniform meshes and in Table 3.2.6 for the non-uniform meshes. We can see that the correct orders of accuracy are again obtained by the third and fourth order WENO methods.

Table 3.1.6: Accuracy for 2D Euler equation of smooth vortex evolution, non-uniform meshes, linear schemes.

	P^1 (3rd order)				P^2 (4th order)			
h	L^1 error	order	L^∞ error	order	L^1 error	order	L^∞ error	order
$h_0/2$	1.81E-02	—	2.98E-01	—	7.00E-03	—	8.16E-02	—
$h_0/4$	7.74E-03	1.28	1.44E-01	1.05	1.18E-03	2.57	1.61E-02	2.34
$h_0/8$	1.67E-03	2.21	2.47E-02	2.54	8.17E-05	3.85	1.31E-03	3.62
$h_0/16$	2.86E-04	2.55	4.79E-03	2.37	4.70E-06	4.12	1.10E-04	3.57
$h_0/32$	3.94E-05	2.86	7.95E-04	2.59	2.68E-07	4.13	7.73E-06	3.83
$h_0/64$	5.07E-06	2.96	1.25E-04	2.67	1.56E-08	4.10	5.99E-07	3.69

Figure 3.1.3: 2D vortex evolution: third order linear scheme.

3.2 Accuracy Test for Triangular WENO Schemes

We use the same examples as in section 3.1 to test the accuracy of third and fourth order triangular WENO schemes constructed in the previous section.

3.2.1 Two-dimensional linear equation

It is same as defined in section 3.1.1, (3.1.1). The accuracy results are shown in Table 3.2.1 for the uniform meshes and in Table 3.2.2 for the non-uniform meshes. We can see that the correct orders of accuracy are obtained by the third and fourth order WENO methods.

3.2.2 Two-dimensional Burgers' equation

It is same as defined in section 3.1.2, (3.1.2). The accuracy results are shown in Table 3.2.3 for the uniform meshes and in Table 3.2.4 for the non-uniform meshes. We can see that the correct orders of accuracy are again obtained by the third and fourth order WENO methods.

and boundary conditions is just the passive convection of the vortex with the mean velocity.

The reconstruction procedure is applied to each component of the solution ξ. We first compute the solution to $t = 2.0$ for the accuracy test. The meshes are the same as those in section 3.1.1 suitably scaled for the new spatial domain. The accuracy results are shown in Table 3.1.5 for the uniform meshes and Table 3.1.6 for the non-uniform meshes. The errors presented are those of the cell averages of ρ.

Table 3.1.5: Accuracy for 2D Euler equation of smooth vortex evolution, uniform meshes, linear schemes.

	P^1 (3rd order)				P^2 (4th order)			
h	L^1 error	order	L^∞ error	order	L^1 error	order	L^∞ error	order
1	1.65E-02	—	2.60E-01	—	5.26E-03	—	7.89E-02	—
1/2	6.31E-03	1.39	1.21E-01	1.10	7.36E-04	2.84	1.62E-02	2.28
1/4	1.31E-03	2.27	2.53E-02	2.26	5.40E-05	3.77	1.03E-03	3.98
1/8	2.21E-04	2.57	4.66E-03	2.44	2.32E-06	4.54	5.36E-05	4.26
1/16	2.98E-05	2.89	6.44E-04	2.86	1.10E-07	4.40	2.48E-06	4.43
1/32	3.77E-06	2.98	8.23E-05	2.97	6.37E-09	4.11	1.25E-07	4.31

We then fix the mesh at $h = \frac{1}{8}$ (uniform) and compute the long time evolution of the vortex. Fig. 3.1.3 is the result by the third order scheme at $t = 0$ and after 1, 5 and 10 time periods, and Fig. 3.1.4 is the result by the fourth order scheme. We show the line cut through the center of the vortex for the density ρ. It is easy to see the difference between the third and fourth order schemes. The fourth order scheme gives almost no dissipation even after 10 periods, while the dissipation is quite noticeable for the long time results of the third order scheme. The mesh chosen is on the borderline of resolution for the fourth order scheme and does not give enough resolution for long time simulation of the third order scheme.

3.1.3 Two-dimensional vortex evolution problem for the Euler equations

See [48] for a description of this problem. We consider the compressible Euler equations of gas dynamics:

$$\xi_t + f(\xi)_x + g(\xi)_y = 0 \tag{3.1.3}$$

where

$$\xi = (\rho, \rho u, \rho v, E),$$

$$f(\xi) = (\rho u, \rho u^2 + p, \rho uv, u(E+p)),$$

$$g(\xi) = (\rho v, \rho uv, \rho v^2 + p, v(E+p)).$$

Here ρ is the density, (u, v) is the velocity, E is the total energy, p is the pressure, and

$$E = \frac{p}{\gamma - 1} + \frac{1}{2}\rho(u^2 + v^2)$$

with $\gamma = 1.4$.

The mean flow is $\rho = 1$, $p = 1$, and $(u, v) = (1, 1)$. We add, to the mean flow, an isentropic vortex (perturbations in (u, v) and the temperature $T = \frac{p}{\rho}$, no perturbation in the entropy $S = \frac{p}{\rho^\gamma}$):

$$(\delta u, \delta v) = \frac{\varepsilon}{2\pi} e^{0.5(1-r^2)}(-\overline{y}, \overline{x})$$

$$\delta T = -\frac{(\gamma - 1)\varepsilon^2}{8\gamma\pi^2} e^{1-r^2}, \qquad \delta S = 0,$$

where $(\overline{x}, \overline{y}) = (x - 5, y - 5)$, $r^2 = \overline{x}^2 + \overline{y}^2$, and the vortex strength $\varepsilon = 5$.

The computational domain is taken as [0,10]×[0,10], and periodic boundary conditions are used.

It is clear that the exact solution of the Euler equation with the above initial

in cell averages. The exact solution of (3.1.2) can be obtained from the exact solution for the one-dimensional Burgers' equation via the transformation $\xi = (x+y)/2$, $\eta = (x-y)/2$.

Table 3.1.3: Accuracy for 2D Burgers' equation, uniform meshes, linear schemes.

	P^1 (3rd order)				P^2 (4th order)			
h	L^1 error	order	L^∞ error	order	L^1 error	order	L^∞ error	order
2/5	2.67E-02	—	7.75E-02	—	8.63E-03	—	2.18E-02	—
1/5	3.65E-03	2.87	1.16E-02	2.74	6.08E-04	3.83	1.70E-03	3.68
1/10	4.60E-04	2.99	1.52E-03	2.93	3.97E-05	3.94	1.16E-04	3.87
1/20	5.75E-05	3.00	1.91E-04	2.99	2.51E-06	3.98	7.37E-06	3.98
1/40	7.18E-06	3.01	2.38E-05	3.01	1.57E-07	4.00	4.62E-07	4.00
1/80	8.96E-07	3.00	2.97E-06	3.00	9.83E-09	4.00	2.89E-08	4.00

We then use the same non-uniform meshes as in section 3.1.1, shown in Fig. 3.1.2 for the coarsest case. The accuracy result is shown in Table 3.1.4.

Table 3.1.4: Accuracy for 2D Burgers' equation, non-uniform meshes, linear schemes.

	P^1 (3rd order)				P^2 (4th order)			
h	L^1 error	order	L^∞ error	order	L^1 error	order	L^∞ error	order
$h_0/2$	1.69E-02	—	7.95E-01	—	3.96E-03	—	1.88E-02	—
$h_0/4$	2.23E-03	2.92	1.23E-02	2.69	2.87E-04	3.79	2.17E-03	3.12
$h_0/8$	2.84E-04	2.97	1.69E-03	2.86	1.90E-05	3.92	1.81E-04	3.58
$h_0/16$	3.57E-05	2.99	2.22E-04	2.93	1.20E-06	3.99	1.34E-05	3.77
$h_0/32$	4.48E-06	2.99	3.00E-05	2.89	7.57E-08	3.99	1.00E-06	3.74
$h_0/64$	5.63E-07	2.99	4.26E-06	2.82	4.75E-09	4.00	7.57E-08	3.72

ones. The accuracy result is shown in Table 3.1.2. We remark that h here is only a rough indicator of the mesh size. There are more triangles here than in the uniform mesh cases with the same h.

Figure 3.1.2: Non-uniform mesh with $h = 1$ for accuracy test.

Table 3.1.2: Accuracy for 2D linear equation, non-uniform meshes, linear schemes.

	P^1 (3rd order)				P^2 (4th order)			
h	L^1 error	order	L^∞ error	order	L^1 error	order	L^∞ error	order
$h_0/2$	1.21E-01	—	2.25E-01	—	4.95E-03	—	1.73E-02	—
$h_0/4$	1.81E-02	2.74	3.74E-02	2.59	2.90E-04	4.09	1.42E-03	3.61
$h_0/8$	2.36E-03	2.94	5.39E-03	2.80	2.21E-05	3.71	8.32E-05	4.09
$h_0/16$	3.00E-04	2.98	7.19E-04	2.91	1.29E-06	4.10	5.09E-06	4.03
$h_0/32$	3.78E-05	2.99	9.40E-05	2.94	7.76E-08	4.06	3.16E-07	4.01
$h_0/64$	4.75E-06	2.99	1.22E-05	2.95	4.75E-09	4.03	1.95E-08	4.02

3.1.2 Two-dimensional Burgers' equation

Consider

$$u_t + \left(\frac{u^2}{2}\right)_x + \left(\frac{u^2}{2}\right)_y = 0 \qquad (3.1.2)$$

with the initial condition $u_0(x, y) = 0.3 + 0.7 \sin(\frac{\pi}{2}(x + y))$, $-2 \le x \le 2$, $-2 \le y \le 2$, and periodic boundary conditions.

We first use the same uniform triangular meshes as in section 3.1.1, shown in Fig. 3.1.1 for the coarsest case $h = \frac{2}{5}$. In Table 3.1.3, the accuracy results are shown for both the third order scheme and the fourth order scheme, at $t = 0.5/\pi^2$ when the solution is still smooth. The errors presented are those of the point values at the 6 quadrature points of each triangle, as in this nonlinear case it is easier to obtain the exact solution of the PDE in point values than

3.1 Accuracy Test for Linear Schemes

3.1.1 Two-dimensional linear equation

Consider

$$u_t + u_x + u_y = 0 \tag{3.1.1}$$

with the initial condition $u_0(x, y) = \sin(\frac{\pi}{2}(x + y))$, $-2 \le x \le 2$, $-2 \le y \le 2$, and periodic boundary conditions.

We first use uniform triangular meshes which are obtained by adding one diagonal line in each rectangle, shown in Fig. 3.1.1 for the coarsest case $h = \frac{2}{5}$. In Table 3.1.1, the accuracy results are shown for both the third order scheme (from the combination of linear polynomials) and the fourth order scheme (from the combination of quadratic polynomials), where h is the length of the rectangles, at $t = 2.0$. The errors presented are those of the cell averages of u.

Figure 3.1.1: Uniform mesh with $h = \frac{2}{5}$ for accuracy test.

Table 3.1.1: Accuracy for 2D linear equation, uniform meshes, linear schemes.

	P^1 (3rd order)				P^2 (4th order)			
h	L^1 error	order	L^∞ error	order	L^1 error	order	L^∞ error	order
2/5	1.80E-01	—	2.79E-01	—	1.40E-02	—	2.17E-02	—
1/5	2.81E-02	2.68	4.37E-02	2.68	9.11E-04	3.94	1.41E-03	3.94
1/10	3.65E-03	2.95	5.72E-03	2.93	5.57E-05	4.03	8.72E-05	4.02
1/20	4.60E-04	2.99	7.22E-04	2.99	3.43E-06	4.02	5.39E-06	4.02
1/40	5.76E-05	3.00	9.05E-05	3.00	2.12E-07	4.02	3.34E-07	4.01
1/80	7.21E-06	3.00	1.13E-05	3.00	1.32E-08	4.01	2.07E-08	4.01

We then use non-uniform meshes, shown in Fig. 3.1.2 for the coarsest case $h = h_0 = 1$, where h is just an average length. The refinement of the meshes is done in a uniform way, namely by cutting each triangle into 4 smaller similar

Chapter 3

Numerical Examples for WENO Schemes

In this chapter, We will implement the third and the fourth order WENO (and linear) schemes developed in the previous chapter to some two-dimensional test problems. First, for the third and the fourth order linear schemes (section 2.8), accuracy tests are given for linear, Burgers' equation and the smooth vortex problem. Next, for the third and the fourth order WENO schemes, the same accuracy tests are given. Finally, some non-smooth problems for two-dimensional Euler equations are tested using our WENO schemes.

As an explicit method, the WENO schemes constructed above are easily implemented on an IBM SP-2 parallel computer. The parallel efficiency is over 90% when 16 processors are used. Most of the numerical examples with large meshes in the next section are obtained with the SP-2 parallel computer using 16 processors at the Center for Fluid Mechanics of Brown University.

stants.

* For the third order case, per triangle, the number of multiplications is 306 and the number of additions is 234. This is because for each quadrature point there are 3 reconstructions, for the 3 $\tilde{p}_s$ in (2.9.1), each with 5 multiplications and 4 additions in (2.9.2). It also involves 3 evaluations of the smooth indicator (2.9.3), for the 3 $\tilde{p}_s$ in (2.9.1), each with 12 multiplications and 9 additions in (2.9.5).
* For the fourth order case, per triangle, the number of multiplications is 468 and the number of additions is 300. This is because for each quadrature point there are 6 reconstructions, for the 6 q_s in (2.7.7), each with 6 multiplications and 5 additions. There are also 6 evaluations of the smooth indicator (2.9.3) for the whole triangle, for the 6 q_s in (2.7.7), each with 42 multiplications and 20 additions.

- Compute the numerical flux (2.2.4) at each quadrature point, then form the residue and forward in time.
- The total costs per triangle per residue evaluation, including the WENO reconstruction and flux evaluation (2.2.4), are 390 multiplications/divisions, 285 additions/subtractions, and 6 flux evaluations for the third order scheme, and 594 multiplications/divisions, 375 additions/subtractions, and 6 flux evaluations for the fourth order scheme. In our implementation on a SUN Ultra1 workstation, the third order WENO scheme takes 33.6 microseconds per triangle per residue evaluation, while the fourth order WENO scheme takes 35.9 microseconds. This is about 6 times of the CPU time needed for a linear scheme of the same order of accuracy implemented in a most economical way.

the right of each side.

- Compute and store the following mesh dependent constants:
  - The constant coefficients in the linear combinations of cell averages to recover values of u at Gaussian points, for each stencil. Examples of these constants are given in (2.9.2). There are $5 \times 3 \times 6 = 90$ such constants per triangle for the third order case (5 for each of the three grouped linear polynomials $\tilde{p}_s$ in (2.9.1), for each of the 6 quadrature points), and $6 \times 6 \times 6 = 216$ such constants per triangle for the fourth order case (6 for each of the six quadratic polynomials $\tilde{q}_s$ in (2.7.7), for each of the 6 quadrature points).
  - For the third order case, the constant coefficients in the linear combinations inside each of the squares of the smoothness indicator S in (2.9.3), for each stencil. Examples of these constants are given in (2.9.5). There are $10 \times 3 \times 6 = 180$ such constants per triangle for the third order case (10 for each of the three grouped linear polynomials $\tilde{p}_s$ in (2.9.1), for each of the 6 quadrature points). For the fourth order case, the constant coefficients in the combination of all the quadratic terms of the smoothness indicator S in (2.9.3), for each polynomial q_s in (2.7.7). There are $21 \times 6 = 126$ such constants per triangle (21 quadratic terms, such as $\bar{u}_0^2, \bar{u}_0\bar{u}_i, \ldots$, out of the 6 $\bar{u}$'s in the stencil, for each of the 6 quadratic polynomials q_s in (2.7.7)).
  - The total storage requirement per triangle: 270 numbers for the third order case and 342 numbers for the fourth order case.
- Start the time iteration. For each time stage:
  - For each triangle, compute the lower order reconstructions for each quadrature point and the nonlinear weights, using the prestored con-

the right eigenvector are columns of

$$R = \begin{pmatrix} 1 & 0 & 1 & 1 \\ u - \tilde{n}_x c & -\tilde{n}_y & u & u + \tilde{n}_x c \\ v - \tilde{n}_y c & \tilde{n}_x & v & v + \tilde{n}_y c \\ H - \tilde{\theta}_n c & -\tilde{n}_y u + \tilde{n}_x v & \frac{1}{2}(u^2 + v^2) & H + \tilde{\theta}_n c \end{pmatrix}; \tag{2.9.14}$$

the left eigenvectors are the rows of

$$L = \frac{1}{2} \begin{pmatrix} (b_2 + \tilde{\theta}_n / c) & -(b_1 u + \tilde{n}_x / c) & -(b_1 v + \tilde{n}_y / c) & b_1 \\ 2(\tilde{n}_y u - \tilde{n}_x v) & -2\tilde{n}_y & 2\tilde{n}_x & 0 \\ 2(1 - b_2) & 2 b_1 u & 2 b_1 v & -2 b_1 \\ (b_2 - \tilde{\theta}_n / c) & -(b_1 u - \tilde{n}_x / c) & -(b_1 v - \tilde{n}_y / c) & b_1 \end{pmatrix}, \tag{2.9.15}$$

where

$$\tilde{n}_x = \frac{n_x}{\sqrt{n_x^2 + n_y^2}}, \quad \tilde{n}_y = \frac{n_y}{\sqrt{n_x^2 + n_y^2}}, \quad \tilde{\theta}_n = \tilde{n}_x u + \tilde{n}_y v, \tag{2.9.16}$$

$$c = \sqrt{\frac{\gamma p}{\rho}}, \quad H = \frac{E + p}{\rho}, \quad U = n_x u + n_y v, \tag{2.9.17}$$

$$b_1 = \frac{\gamma - 1}{c^2}, \quad b_2 = \frac{1}{2}(u^2 + v^2) b_1. \tag{2.9.18}$$

2.9.6 Algorithm flow chart and parallel implementation

We summarize the practical aspects and operation and storage costs of implementing the method:

- Generate a triangular mesh. In addition to the coordinates of each node and indices of three vertices for each triangle, the information of connectivity for the mesh is also needed, which includes indices of three neighboring elements for each triangle and indices of two triangles on the left and one

description in the following.

Let us take one side of the triangle which has the outward unit normal (n_x, n_y). Let A be some average Jacobian at one quadrature point,

$$A = n_x \frac{\partial f}{\partial u} + n_y \frac{\partial g}{\partial u}. \tag{2.9.12}$$

For Euler systems, the Roe's mean matrix [45] is used. Denote by R the matrix of right eigenvectors and L the matrix of left eigenvectors of A. Then the scalar triangular WENO scheme can be applied to each of the characteristic fields, i.e. to each component of the vector $v = L\,u$. With the reconstructed point values v, we define our reconstructed point values u by $u = R\,v$.

Here are some details about the matrix of right eigenvectors R, and the matrix of left eigenvectors L for two-dimensional Euler systems of gas dynamics, see [51].

$$\xi_t + f(\xi)_x + g(\xi)_y = 0 \tag{2.9.13}$$

where

$$\xi = (\rho, \rho u, \rho v, E),$$

$$f(\xi) = (\rho u, \rho u^2 + p, \rho uv, u(E + p)),$$

$$g(\xi) = (\rho v, \rho uv, \rho v^2 + p, v(E + p)).$$

Here ρ is the density, (u, v) is the velocity, E is the total energy, p is the pressure, and

$$E = \frac{p}{\gamma - 1} + \frac{1}{2}\rho(u^2 + v^2)$$

with $\gamma = 1.4$.

From [51], eigenvalues for A in (2.9.12) are $U - c\sqrt{n_x^2 + n_y^2}$, U, U, $U + c\sqrt{n_x^2 + n_y^2}$;

where

$$\sigma^+ = \tilde{\gamma}_1^+ + \tilde{\gamma}_2^+ + \tilde{\gamma}_3^+, \quad \sigma^- = \tilde{\gamma}_1^- + \tilde{\gamma}_2^- + \tilde{\gamma}_3^-, \tag{2.9.8}$$

$$\tilde{\gamma}_i^+ = \frac{1}{2}(\tilde{\gamma}_i + 3|\tilde{\gamma}_i|), \quad \tilde{\gamma}_i^- = \tilde{\gamma}_i - \tilde{\gamma}_i^+, \qquad i = 1, 2, 3, \tag{2.9.9}$$

and

$$R^+(x, y) = (\tilde{\gamma}_1^+ \tilde{p}_1(x, y) + \tilde{\gamma}_2^+ \tilde{p}_2(x, y) + \tilde{\gamma}_3^+ \tilde{p}_3(x, y))/\sigma^+ \tag{2.9.10}$$

$$R^-(x, y) = (\tilde{\gamma}_1^- \tilde{p}_1(x, y) + \tilde{\gamma}_2^- \tilde{p}_2(x, y) + \tilde{\gamma}_3^- \tilde{p}_3(x, y))/\sigma^- \tag{2.9.11}$$

Apply the WENO reconstruction procedure in the previous section to $R^+(x, y)$ and $R^-(x, y)$ respectively, combine two results according to (2.9.7) as a WENO reconstruction for $R(x, y)$. It is clear that the above procedure keeps the third order accuracy in smooth regions, while both $R^+(x, y)$ and $R^-(x, y)$ have positive weights.

This splitting technique works for the third order scheme on all triangulations we have encountered so far, while a smaller CFL number (by about one half) should be used. For the fourth order case, it works well for scalar equations (linear and Burgers'), we have not made it work for Euler systems yet, the instability problem still exists for negative weights; it would be due to some problems in the implementation of characteristic decomposition or even some bugs in coding the algorithm; further investigations are underway in these directions.

2.9.5 Extension to the Euler systems

There are two ways to extend the previous results to systems. One is to do so component by component. This is easy to implement and cost effective, and it seems to work well for the third order scheme. We will use componentwise methods for all numerical examples with the third order scheme. Another extension method is by the characteristic decomposition. We will give a brief

computed and stored once the mesh is generated. We do not need to store any information about the polynomial $\tilde{p}_1$ itself.

2.9.4 A splitting technique for negative weights

As we mentioned before, there are some cases when both third order and fourth order reconstructions give negative weights with rather large magnitudes (due to abrupt changing in triangles sizes or orientations), this will cause WENO schemes unstable for shock calculations. One possible solution is to try some good grouping techniques to make them back to positive, it seems very difficult to do so systematically for arbitrary triangulations. Here we introduce a splitting method to handle negative weights directly. This idea of splitting is similar to that for Lax-Friedrichs flux.

The key step here is to separate the negative weights from those positive. For each of "negative" and "positive" groups, the non-linear WENO reconstruction will give a convex combination of polynomials which is stable in shock regions. However, the simple separation would not do the job; it would destroy the whole philosophy of WENO since the stencils from two groups do not talk to each other. Instead, we exchange parts of two groups, namely, we add some amount to a positive weight, then minus same amount as a new negative weight, do the same for all original negative weights. By this way, the original stencils are "copied" to each of negative and positive groups.

Consider the third order case, from (2.9.1) we have

$$R(x,y) = \tilde{\gamma}_1\tilde{p}_1(x,y) + \tilde{\gamma}_2\tilde{p}_2(x,y) + \tilde{\gamma}_3\tilde{p}_3(x,y) \tag{2.9.6}$$

If $\min(\tilde{\gamma}_1, \tilde{\gamma}_2, \tilde{\gamma}_3) < 0$, rewrite $R(x,y)$ as

$$R(x,y) = \sigma^+ R^+(x,y) + \sigma^- R^-(x,y) \tag{2.9.7}$$

2.9.3 Smoothness indicators and non-linear weights

We finally come to the point of smooth indicators and nonlinear weights. For this we follow exactly as in Jiang and Shu [34]. For a polynomial $p(x,y)$ with degree up to k, we define the following measurement for smoothness

$$S = \sum_{1\le|\alpha|\le k} \int_{\triangle} |\triangle|^{|\alpha|-1} (D^\alpha p(x,y))^2 dxdy \tag{2.9.3}$$

where α is a multi-index and D is the derivative operator, for example, when $\alpha = (1,2)$ then $|\alpha| = 3$ and $D^\alpha p(x,y) = \frac{\partial p^3(x,y)}{\partial x \partial y^2}$. The non-linear weights are then defined as:

$$\omega_j = \frac{\tilde{\omega_j}}{\sum_i \tilde{\omega_i}}, \qquad \tilde{\omega_i} = \frac{\gamma_i}{(\varepsilon + S_i)^2} \tag{2.9.4}$$

where γ_i is the i-th coefficient in the linear combination of polynomials (i.e. the $\tilde{\gamma_s}$ in (2.9.1) for the third order case and the γ_s in (2.7.7) for the fourth order case), S_i is the measurement of smoothness of the i-th polynomial $p_i(x,y)$ (i.e. the $\tilde{p}_s$ in (2.9.1) for the third order case and the q_s in (2.7.7) for the fourth order case), and ε is a small positive number which we take as $\varepsilon = 10^{-3}$ for all the numerical experiments in this paper. The numerical results are not very sensitive to the choice of ε in a range from 10^{-2} to 10^{-6}. In general, larger ε gives better accuracy for smooth problems but may generate small oscillations for shocks. Smaller ε is more friendly to shocks. The nonlinear weights ω_j in (2.9.4) would then replace the linear weights γ_j to form a WENO reconstruction.

We emphasize that the smoothness measurements (2.9.3) are quadratic functions of the cell averages in the stencil. For example, it is the 10 constants b_i and c_i, which depend on the local geometry only, such that

$$S = (b_1\bar{u}_0 + b_2\bar{u}_i + b_3\bar{u}_k + b_4\bar{u}_{ia} + b_5\bar{u}_{ib})^2 + (c_1\bar{u}_0 + c_2\bar{u}_i + c_3\bar{u}_k + c_4\bar{u}_{ia} + c_5\bar{u}_{ib})^2 \tag{2.9.5}$$

for the smoothness measurements (2.9.3) of $\tilde{p}_1$ in (2.9.1), that have to be pre-

depend on the local geometry only, such that

$$\tilde{p}_1(x^{G_1}, y^{G_1}) = a_1\bar{u}_0 + a_2\bar{u}_i + a_3\bar{u}_k + a_4\bar{u}_{ia} + a_5\bar{u}_{ib}, \tag{2.9.2}$$

that have to be precomputed and stored once the mesh is generated. We do not need to store any information about the polynomial $\tilde{p}_1$ itself.

For the second quadrature point on side i, (G_2 in Fig. 2.4.1), Group 1 contains $p_3\,(0,i,j)$, $p_4\,(0,i,ia)$, and $p_5\,(0,i,ib)$, with the combination coefficient $\tilde{\gamma}_1 = \gamma_3 + \gamma_4 + \gamma_5$; Group 2 contains $p_2\,(0,k,i)$, $p_8\,(0,k,ka)$, and $p_9\,(0,k,kb)$; with the combination coefficient $\tilde{\gamma}_2 = \gamma_2 + \gamma_8 + \gamma_9$; Group 3 contains $p_1\,(0,j,k)$, $p_6\,(0,j,ja)$, and $p_7\,(0,j,jb)$; with combination coefficient $\tilde{\gamma}_3 = \gamma_1 + \gamma_6 + \gamma_7$. We can do the same thing for the other two sides (j,k).

2.9.2 Positivity of linear weights for the fourth order scheme

From section 2.7.2, there is a degree of freedom (which was chosen to be γ_1) in the determination of the linear weights γ_s in (2.7.7). Our objective is to use this degree of freedom to obtain non-negative linear weights. This again amounts to solving a group of 6 linear inequalities for γ_1. We have found out through our numerical experiments that positivity of linear weights can only be achieved for nearly uniform meshes. In this paper only such meshes are used for the fourth order WENO scheme. The difficulty in doing a grouping here similar to the third order case, is that the stencil to each group would then be quite large, hence shocks may be included in all stencils. Good grouping strategies are still under investigation.

The grouping we will introduce in the following works for most triangulations. There are however cases when it does give some negative coefficients with very small magnitudes, but it does not seem to affect the stability of WENO schemes built upon them[1].

For the first quadrature point on side i (G_1 in Fig. 2.4.1), Group 1 contains $p_2\,(0,k,i)$, $p_4\,(0,i,ia)$, and $p_5\,(0,i,ib)$,

$$\tilde{p}_1 = (\gamma_2 p_2 + \gamma_4 p_4 + \gamma_5 p_5)/(\gamma_2 + \gamma_4 + \gamma_5), \qquad \tilde{\gamma}_1 = \gamma_2 + \gamma_4 + \gamma_5,$$

Group 2 contains $p_3\,(0,i,j)$, $p_6\,(0,j,ja)$, and $p_7\,(0,j,jb)$,

$$\tilde{p}_2 = (\gamma_3 p_3 + \gamma_6 p_6 + \gamma_7 p_7)/(\gamma_3 + \gamma_6 + \gamma_7), \qquad \tilde{\gamma}_2 = \gamma_3 + \gamma_6 + \gamma_7,$$

Group 3 contains $p_1\,(0,j,k)$, $p_8\,(0,k,ka)$, and $p_9\,(0,k,kb)$,

$$\tilde{p}_3 = (\gamma_1 p_1 + \gamma_8 p_8 + \gamma_9 p_9)/(\gamma_1 + \gamma_8 + \gamma_9), \qquad \tilde{\gamma}_3 = \gamma_1 + \gamma_8 + \gamma_9.$$

The resulting linear polynomial

$$\tilde{R}(x,y) = \sum_{s=1}^{3} \tilde{\gamma}_s \tilde{p}_s(x,y) \tag{2.9.1}$$

is identical to $R(x,y)$ in (2.7.1) and in most cases the coefficients $\tilde{\gamma}_s$ can be made non-negative by suitably choosing the value of the degree of freedom γ_1, through the solution of a group of 3 linear inequalities for γ_1.

We remark that for practical implementation, it is the 5 constants a_i, which

[1] Recently we have learned, in a joint project with R. Biswas and J. Djomehri on using adaptive methods with WENO schemes, that this grouping may fail near adaptively refined regions where triangle sizes are changing very abruptly. New grouping techniques to overcome this is under investigation. On the other hand, we will present a technique trying to solving the instability problem caused by negative weights.

For the temporal discretization, the third order TVD Runge-Kutta scheme of Shu and Osher [49] is used. Again, as we stated in section 2.3, for the fourth order scheme, we use $\Delta t = (\Delta x)^{4/3}$ to achieve fourth order accuracy in time, but only for the examples of the accuracy test.

2.9 WENO Reconstruction and WENO Schemes

In this section, we will introduce non-linear weights to make the resulting schemes suitable for shock computations. To ensure stability near shocks, we need non-negative weights, thus we need non-negative linear weights to start with. As there is one degree of freedom in the choice of linear weights for both the third order and fourth order cases described in the previous section, this will be explored to produce positive linear weights. In the case of third order scheme, grouping of polynomials is also used to achieve positivity. Also for the cases of negative weights, we introduce a Lax-Friedrichs type splitting technique to handle the instability problem.

2.9.1 Positivity of linear weights for the third order scheme

We will use the same notations as in the previous section. In (2.7.1), by consistency, $\sum_{s=1}^{9} \gamma_s = 1$. We want to group these 9 linear polynomials into 3 groups:

$$\sum_{s=1}^{9} \gamma_s p_s(x,y) = \sum_{s=1}^{3} \tilde{\gamma}_s \tilde{p}_s(x,y),$$

each $\tilde{p}_s(x,y)$ being still a linear polynomial and a second order approximation to u, with positive coefficients $\tilde{\gamma}_s \geq 0$. We also require the stencils corresponding to the three new linear polynomials $\tilde{p}_s(x,y)$ to be reasonably separated, so that when shocks are present, not all stencils will contain the shock under normal situations.

such that the linear combination of these q_s

$$Q(x,y) = \sum_{s=1}^{6} \gamma_s q_s(x,y) \tag{2.7.7}$$

satisfies

$$Q(x^G, y^G) = p^3(x^G, y^G) \tag{2.7.8}$$

for all $\bar{u}$'s.

As before, (2.7.8) results in 10 linear equations for the 6 unknowns γ_s, which are the coefficients of the 10 cell averages $\bar{u}$'s in (2.7.3). This looks like a grossly over-determined system, but it is in fact under-determined with rank 5, thus allowing a solution for γ_s with one degree of freedom. A crucial observation is again that (2.7.8) is valid for all the 6 cases $u = 1, x, y, x^2, xy, y^2$ under just one constraint on the γ_s, namely $\sum_{s=1}^{9} \gamma_s = 1$, because $q_s(x)$ and $p^3(x)$ all reproduce quadratic functions exactly. We can thus eliminate 5 equations from the 10, resulting in a rank 5 system with one degree of freedom in the solution for γ_s. In practice, we obtain the solution γ_s for $s \geq 2$ with γ_1 as the degree of freedom.

Again, the first effort we would like to make is to use this degree of freedom to obtain a set of non-negative γ_s, which is important for the WENO procedure to be developed later for shock calculations. This has been performed for the mostly near-uniform meshes used in the numerical examples.

2.8 Linear Schemes

From the third and the fourth order reconstructions, we can now obtain the third and the fourth order linear schemes for (2.2.1) by replacing $u^-(G_j,t)$ in (2.2.4) for each side of the cell $\triangle_i$ with the reconstructed values $R(x^G, y^G)$ in (2.7.1) or $Q(x^G, y^G)$ in (2.7.7), respectively. Similarly, $u^+(G_j,t)$ is replaced with the reconstructed values in the neighboring cell for each side of $\triangle_i$.

equality (2.7.6) is valid for all the three cases $u = 1$, $u = x$ and $u = x^2$ with only two conditions: $\gamma_1 + \gamma_2 = 1$ and another one obtained when $u = x^2$, resulting in a solvable 2×2 system for γ_s.

The same argument can be applied in the current two dimensional case. Although there are 10 linear equations for the nine weights γ_s resulting from the equality (2.7.2), we should notice that the equality (2.7.2) is valid for all three cases $u = 1$, $u = x$ and $u = y$ under only one constraint on γ_s, namely $\sum_{s=1}^{9} \gamma_s = 1$, again because $p_s(x)$ and $p^2(x)$ all reproduce linear functions exactly. Thus we can eliminate two equations from the ten, resulting in a rank 8 system with one degree of freedom in the solution for γ_s. In practice, we obtain the solution γ_s for $s \geq 2$ with γ_1 as the degree of freedom.

The first effort we would like to make is to use this degree of freedom to obtain a set of non-negative γ_s, which is important for the WENO procedure to be developed later for shock calculations. Unfortunately, it turns out that, for many triangulations, this is impossible. Some grouping is needed and will be discussed later.

2.7.2 Combination for fourth order scheme

We would like to construct several quadratic polynomials whose weighted average will give the same result as the cubic reconstruction p^3 at each quadrature point (the weights are different for different quadrature points). The following 6 quadratic polynomials are constructed by having the same cell averages as u on the corresponding triangles (see Fig. 2.4.1):

q_1 (on triangles: $0, i, ia, ib, k, kb$), q_2 (on triangles: $0, i, ia, ib, j, ja$),
q_3 (on triangles: $0, j, ja, jb, i, ib$), q_4 (on triangles: $0, j, ja, jb, k, ka$),
q_5 (on triangles: $0, k, ka, kb, j, jb$), q_6 (on triangles: $0, k, ka, kb, i, ia$).

For each quadrature point (x^G, y^G), we would like to find the linear weights

intervals. The two linear polynomials p_s, where p_1 agrees with u on cell averages in the intervals I_0 and I_1, and p_2 agrees with u on cell averages in the intervals I_1 and I_2, give the following two second order approximations to the value of u at the point $x_{\frac{3}{2}}$ (the boundary of I_1 and I_2):

$$-\frac{1}{2}\bar{u}_0 + \frac{3}{2}\bar{u}_1, \qquad \frac{1}{2}\bar{u}_1 + \frac{1}{2}\bar{u}_2. \tag{2.7.4}$$

The quadratic polynomial p^2, which agrees with u on cell averages in the intervals I_0, I_1 and I_2, gives the following third order approximation to the value of u at the point $x_{\frac{3}{2}}$:

$$-\frac{1}{6}\bar{u}_0 + \frac{5}{6}\bar{u}_1 + \frac{1}{3}\bar{u}_1. \tag{2.7.5}$$

We would like to find γ_s such that

$$\gamma_1\left(-\frac{1}{2}\bar{u}_0 + \frac{3}{2}\bar{u}_1\right) + \gamma_2\left(\frac{1}{2}\bar{u}_1 + \frac{1}{2}\bar{u}_2\right) = -\frac{1}{6}\bar{u}_0 + \frac{5}{6}\bar{u}_1 + \frac{1}{3}\bar{u}_1 \tag{2.7.6}$$

for arbitrary $\bar{u}$'s. This leads to the following three equations:

$$-\frac{1}{2}\gamma_1 = -\frac{1}{6}, \qquad \frac{3}{2}\gamma_1 + \frac{1}{2}\gamma_2 = \frac{5}{6}, \qquad \frac{1}{2}\gamma_2 = \frac{1}{3},$$

for the two unknowns γ_1 and γ_2. It looks like an over-determined system but is in fact rank 2 and has a unique solution

$$\gamma_1 = \frac{1}{3}, \qquad \gamma_2 = \frac{2}{3}.$$

The reason can be understood if we ask for the validity of the equality (2.7.6) in the cases of $u = 1$, $u = x$ and $u = x^2$. Clearly if (2.7.6) holds in these three cases then it holds for arbitrary choices of $\bar{u}$'s. The crucial observation is that (2.7.6) holds for both $u = 1$ and $u = x$ as long as $\gamma_1 + \gamma_2 = 1$, as all three expressions in (2.7.4) and (2.7.5) reproduce linear functions exactly. Hence the

whose weighted average (linear combination) will give the same result as the quadratic reconstruction p^2 at each quadrature point (the weights are different for different quadrature points).

Using Fig. 2.4.1, we first build the following 9 linear polynomials by agreeing with the cell averages of u on the following stencils: p_1 (on triangles: $0, j, k$), p_2 (on triangles: $0, k, i$), p_3 (on triangles: $0, i, j$), p_4 (on triangles: $0, i, ia$), p_5 (on triangles: $0, i, ib$), p_6 (on triangles: $0, j, ja$), p_7 (on triangles: $0, j, jb$), p_8 (on triangles: $0, k, ka$), and p_9 (on triangles: $0, k, kb$).

For each quadrature point (x^G, y^G), we want to find the linear weights γ_s, which are constants depending only on the local geometry of the mesh, such that the linear polynomial obtained from a linear combination of these p_s

$$R(x,y) = \sum_{s=1}^{9} \gamma_s p_s(x,y) \tag{2.7.1}$$

satisfies

$$R(x^G, y^G) = p^2(x^G, y^G) \tag{2.7.2}$$

where p^2 is defined before using the least squares procedure, for arbitrary choices of cell averages

$$\{\bar{u}_0, \bar{u}_i, \bar{u}_j, \bar{u}_k, \bar{u}_{ia}, \bar{u}_{ib}, \bar{u}_{ja}, \bar{u}_{jb}, \bar{u}_{ka}, \bar{u}_{kb}\}. \tag{2.7.3}$$

Since both the left side and the right side of the equality (2.7.2) are linear in the cell averages (2.7.3), for the equality to hold for arbitrary $\bar{u}$'s in (2.7.3) one must have all 10 coefficients of the $\bar{u}$'s to be identically zero (when all terms are moved to one side of the equality), which leads to 10 linear equations for the nine weights γ_s. This looks like an over-determined system, but is in fact under-determined of rank 8, allowing for one degree of freedom in the choice of the nine γ_s.

Before explaining this, we first look at a simpler but illustrative one dimensional example. Let us denote I_j, $j = 0, 1, 2$, as three equal sized consecutive

$$c_{50}=1,\quad c_{51}=\bar{\xi}_{ib},\, c_{52}=\bar{\eta}_{ib},\, c_{53}=\overline{(\xi^2)}_{ib},\, c_{54}=\overline{(\eta^2)}_{ib},$$

$$c_{55}=\overline{(\xi\eta)}_{ib},\, c_{56}=\overline{(\xi^3)}_{ib},\, c_{57}=\overline{(\eta^3)}_{ib},\, c_{58}=\overline{(\xi^2\eta)}_{ib},\, c_{59}=\overline{(\xi\eta^2)}_{ib};$$

$$c_{60}=1,\quad c_{61}=\bar{\xi}_{ja},\, c_{62}=\bar{\eta}_{ja},\, c_{63}=\overline{(\xi^2)}_{ja},\, c_{64}=\overline{(\eta^2)}_{ja},$$

$$c_{65}=\overline{(\xi\eta)}_{ja},\, c_{66}=\overline{(\xi^3)}_{ja},\, c_{67}=\overline{(\eta^3)}_{ja},\, c_{68}=\overline{(\xi^2\eta)}_{ja},\, c_{69}=\overline{(\xi\eta^2)}_{ja};$$

$$c_{70}=1,\quad c_{71}=\bar{\xi}_{jb},\, c_{72}=\bar{\eta}_{jb},\, c_{73}=\overline{(\xi^2)}_{jb},\, c_{74}=\overline{(\eta^2)}_{jb},$$

$$c_{75}=\overline{(\xi\eta)}_{jb},\, c_{76}=\overline{(\xi^3)}_{jb},\, c_{77}=\overline{(\eta^3)}_{jb},\, c_{78}=\overline{(\xi^2\eta)}_{jb},\, c_{79}=\overline{(\xi\eta^2)}_{jb};$$

$$c_{80}=1,\quad c_{81}=\bar{\xi}_{ka},\, c_{82}=\bar{\eta}_{ka},\, c_{83}=\overline{(\xi^2)}_{ka},\, c_{84}=\overline{(\eta^2)}_{ka},$$

$$c_{85}=\overline{(\xi\eta)}_{ka},\, c_{86}=\overline{(\xi^3)}_{ka},\, c_{87}=\overline{(\eta^3)}_{ka},\, c_{88}=\overline{(\xi^2\eta)}_{ka},\, c_{89}=\overline{(\xi\eta^2)}_{ka};$$

$$c_{90}=1,\quad c_{91}=\bar{\xi}_{kb},\, c_{92}=\bar{\eta}_{kb},\, c_{93}=\overline{(\xi^2)}_{kb},\, c_{94}=\overline{(\eta^2)}_{kb},$$

$$c_{95}=\overline{(\xi\eta)}_{kb},\, c_{96}=\overline{(\xi^3)}_{kb},\, c_{97}=\overline{(\eta^3)}_{kb},\, c_{98}=\overline{(\xi^2\eta)}_{kb},\, c_{99}=\overline{(\xi\eta^2)}_{kb}.$$

For most triangulations, (2.6.2) has a unique solution.

2.7 Linear Combination of Polynomials

From the q-point Gaussian integration formula (2.2.3), we know that the result of a linear scheme depends only on the reconstruction values at those Gaussian quadrature points. So if two reconstructions have same values at those points, they should give same finite volume linear scheme. We want to construct a linear combination of lower order polynomials such that the resultant polynomial has same value at certain (quadrature) point as some higher order polynomial.

2.7.1 Combination for the third order scheme

A key step to build a third order WENO scheme based on linear polynomials is carried out in the following. We want to construct several linear polynomials

$$
\begin{pmatrix}
c_{00} & c_{01} & c_{02} & c_{03} & c_{04} & c_{05} & c_{06} & c_{07} & c_{08} & c_{09} \\
c_{10} & c_{11} & c_{12} & c_{13} & c_{14} & c_{15} & c_{16} & c_{17} & c_{18} & c_{19} \\
c_{20} & c_{21} & c_{22} & c_{23} & c_{24} & c_{25} & c_{26} & c_{27} & c_{28} & c_{29} \\
c_{30} & c_{31} & c_{32} & c_{33} & c_{34} & c_{35} & c_{36} & c_{37} & c_{38} & c_{39} \\
c_{40} & c_{41} & c_{42} & c_{43} & c_{44} & c_{45} & c_{46} & c_{47} & c_{48} & c_{49} \\
c_{50} & c_{51} & c_{52} & c_{53} & c_{54} & c_{55} & c_{56} & c_{57} & c_{58} & c_{59} \\
c_{60} & c_{61} & c_{62} & c_{63} & c_{64} & c_{65} & c_{66} & c_{67} & c_{68} & c_{69} \\
c_{70} & c_{71} & c_{72} & c_{73} & c_{74} & c_{75} & c_{76} & c_{77} & c_{78} & c_{79} \\
c_{80} & c_{81} & c_{82} & c_{83} & c_{84} & c_{85} & c_{86} & c_{87} & c_{88} & c_{89} \\
c_{90} & c_{91} & c_{92} & c_{93} & c_{94} & c_{95} & c_{96} & c_{97} & c_{98} & c_{99}
\end{pmatrix}
\cdot
\begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \\ a_8 \\ a_9 \end{pmatrix}
=
\begin{pmatrix} \bar{u}_0 \\ \bar{u}_i \\ \bar{u}_j \\ \bar{u}_k \\ \bar{u}_{ia} \\ \bar{u}_{ib} \\ \bar{u}_{ja} \\ \bar{u}_{jb} \\ \bar{u}_{ka} \\ \bar{u}_{kb} \end{pmatrix},
\tag{2.6.2}
$$

where

$$c_{00} = 1, \quad c_{01} = \bar{\xi}_0,\ c_{02} = \bar{\eta}_0,\ c_{03} = \overline{(\xi^2)}_0,\ c_{04} = \overline{(\eta^2)}_0,$$

$$c_{05} = \overline{(\xi\eta)}_0,\ c_{06} = \overline{(\xi^3)}_0,\ c_{07} = \overline{(\eta^3)}_0,\ c_{08} = \overline{(\xi^2\eta)}_0,\ c_{09} = \overline{(\xi\eta^2)}_0;$$

$$c_{10} = 1, \quad c_{11} = \bar{\xi}_i),\ c_{12} = \bar{\eta}_i),\ c_{13} = \overline{(\xi^2)}_i,\ c_{14} = \overline{(\eta^2)}_i,$$

$$c_{15} = \overline{(\xi\eta)}_i,\ c_{16} = \overline{(\xi^3)}_i,\ c_{17} = \overline{(\eta^3)}_i,\ c_{18} = \overline{(\xi^2\eta)}_i,\ c_{19} = \overline{(\xi\eta^2)}_i;$$

$$c_{20} = 1, \quad c_{21} = \bar{\xi}_j,\ c_{22} = \bar{\eta}_j,\ c_{23} = \overline{(\xi^2)}_j,\ c_{24} = \overline{(\eta^2)}_j,$$

$$c_{25} = \overline{(\xi\eta)}_j,\ c_{26} = \overline{(\xi^3)}_j,\ c_{27} = \overline{(\eta^3)}_j,\ c_{28} = \overline{(\xi^2\eta)}_j,\ c_{29} = \overline{(\xi\eta^2)}_j;$$

$$c_{30} = 1, \quad c_{31} = \bar{\xi}_k,\ c_{32} = \bar{\eta}_k,\ c_{33} = \overline{(\xi^2)}_k,\ c_{34} = \overline{(\eta^2)}_k,$$

$$c_{35} = \overline{(\xi\eta)}_k,\ c_{36} = \overline{(\xi^3)}_k,\ c_{37} = \overline{(\eta^3)}_k,\ c_{38} = \overline{(\xi^2\eta)}_k,\ c_{39} = \overline{(\xi\eta^2)}_k;$$

$$c_{40} = 1, \quad c_{41} = \bar{\xi}_{ia},\ c_{42} = \bar{\eta}_{ia},\ c_{43} = \overline{(\xi^2)}_{ia},\ c_{44} = \overline{(\eta^2)}_{ia},$$

$$c_{45} = \overline{(\xi\eta)}_{ia},\ c_{46} = \overline{(\xi^3)}_{ia},\ c_{47} = \overline{(\eta^3)}_{ia},\ c_{48} = \overline{(\xi^2\eta)}_{ia},\ c_{49} = \overline{(\xi\eta^2)}_{ia};$$

$$c_{82} = \overline{(\eta - \bar{\eta}_0)_{ka}},\, c_{83} = \overline{(\xi^2 - \overline{(\xi^2)}_0)_{ka}},\, c_{84} = \overline{(\eta^2 - \overline{(\eta^2)}_0)_{ka}},\, c_{85} = \overline{(\xi\eta - \overline{(\xi\eta)}_0)_{ka}};$$

$$c_{91} = \overline{(\xi - \bar{\xi}_0)_{kb}},\, c_{92} = \overline{(\eta - \bar{\eta}_0)_{kb}},\, c_{93} = \overline{(\xi^2 - \overline{(\xi^2)}_0)_{kb}},\, c_{94} = \overline{(\eta^2 - \overline{(\eta^2)}_0)_{kb}},$$

$$c_{95} = \overline{(\xi\eta - \overline{(\xi\eta)}_0)_{kb}};$$

and

$$b_1 = \bar{u}_i - \bar{u}_0, \quad b_2 = \bar{u}_j - \bar{u}_0, \quad b_3 = \bar{u}_k - \bar{u}_0, \quad b_4 = \bar{u}_{ia} - \bar{u}_0, \quad b_5 = \bar{u}_{ib} - \bar{u}_0,$$

$$b_6 = \bar{u}_{ja} - \bar{u}_0, \quad b_7 = \bar{u}_{jb} - \bar{u}_0, \quad b_8 = \bar{u}_{ka} - \bar{u}_0, \quad b_9 = \bar{u}_{kb} - \bar{u}_0.$$

We can solve this system as a linear least square problem; and with a least-square solution $(a_1, a_2, a_3, a_4, a_5)$, we can get a_0 by (2.5.2). The resultant p^2 is a third order approximation to u on $\triangle_0$.

2.6 Fourth Order Reconstruction

To construct a fourth order linear scheme, we need a cubic polynomial reconstruction, which has 10 degrees of freedom. We thus only consider the case where ia, ib, ja, jb, ka, kb are distinct in the stencil (see Fig. 2.4.1). We construct the cubic polynomial p^3 by requiring that its cell average agrees with that of u on each triangle in the 10-triangle stencil shown in Fig. 2.4.1. It seems that for most triangulations this reconstruction is possible.

Let

$$\begin{aligned} p^3(x,y) = a_0 \;\; &+ \;\; a_1\xi + a_2\eta + a_3\xi^2 + a_4\eta^2 + a_5\xi\eta \\ &+ \;\; a_6\xi^3 + a_7\eta^3 + a_8\xi^2\eta + a_9\xi\eta^2, \end{aligned} \tag{2.6.1}$$

we have a 10×10 linear system for 10 unknowns $(a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9)$:

Now the requirement that p^2 matches the cell averages of u on the triangles $\triangle_i$, $\triangle_{ia}$, $\triangle_{ib}$, $\triangle_j$, $\triangle_{ja}$, $\triangle_{jb}$, $\triangle_k$, $\triangle_{ka}$, $\triangle_{kb}$ will give us an over-determined 9×5 system for 5 unknowns $(a_1, a_2, a_3, a_4, a_5)$:

$$\begin{pmatrix} c_{11} & c_{12} & c_{13} & c_{14} & c_{15} \\ c_{21} & c_{22} & c_{23} & c_{24} & c_{25} \\ c_{31} & c_{32} & c_{33} & c_{34} & c_{35} \\ c_{41} & c_{42} & c_{43} & c_{44} & c_{45} \\ c_{51} & c_{52} & c_{53} & c_{54} & c_{55} \\ c_{61} & c_{62} & c_{63} & c_{64} & c_{65} \\ c_{71} & c_{72} & c_{73} & c_{74} & c_{75} \\ c_{81} & c_{82} & c_{83} & c_{84} & c_{85} \\ c_{91} & c_{92} & c_{93} & c_{94} & c_{95} \end{pmatrix} \cdot \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \\ b_8 \\ b_9 \end{pmatrix}, \tag{2.5.4}$$

where

$$c_{11} = \overline{(\xi - \bar{\xi}_0)}_i, c_{12} = \overline{(\eta - \bar{\eta}_0)}_i, c_{13} = \overline{(\xi^2 - \overline{(\xi^2)}_0)}_i, c_{14} = \overline{(\eta^2 - \overline{(\eta^2)}_0)}_i,$$

$$c_{15} = \overline{(\xi\eta - \overline{(\xi\eta)}_0)}_i; c_{21} = \overline{(\xi - \bar{\xi}_0)}_j, c_{22} = \overline{(\eta - \bar{\eta}_0)}_j, c_{23} = \overline{(\xi^2 - \overline{(\xi^2)}_0)}_j,$$

$$c_{24} = \overline{(\eta^2 - \overline{(\eta^2)}_0)}_j, c_{25} = \overline{(\xi\eta - \overline{(\xi\eta)}_0)}_j; c_{31} = \overline{(\xi - \bar{\xi}_0)}_k, c_{32} = \overline{(\eta - \bar{\eta}_0)}_k,$$

$$c_{33} = \overline{(\xi^2 - \overline{(\xi^2)}_0)}_k, c_{34} = \overline{(\eta^2 - \overline{(\eta^2)}_0)}_k, c_{35} = \overline{(\xi\eta - \overline{(\xi\eta)}_0)}_k; c_{41} = \overline{(\xi - \bar{\xi}_0)}_{ia},$$

$$c_{42} = \overline{(\eta - \bar{\eta}_0)}_{ia}, c_{43} = \overline{(\xi^2 - \overline{(\xi^2)}_0)}_{ia}, c_{44} = \overline{(\eta^2 - \overline{(\eta^2)}_0)}_{ia}, c_{45} = \overline{(\xi\eta - \overline{(\xi\eta)}_0)}_{ia};$$

$$c_{51} = \overline{(\xi - \bar{\xi}_0)}_{ib}, c_{52} = \overline{(\eta - \bar{\eta}_0)}_{ib}, c_{53} = \overline{(\xi^2 - \overline{(\xi^2)}_0)}_{ib}, c_{54} = \overline{(\eta^2 - \overline{(\eta^2)}_0)}_{ib},$$

$$c_{55} = \overline{(\xi\eta - \overline{(\xi\eta)}_0)}_{ib}; c_{61} = \overline{(\xi - \bar{\xi}_0)}_{ja}, c_{62} = \overline{(\eta - \bar{\eta}_0)}_{ja}, c_{63} = \overline{(\xi^2 - \overline{(\xi^2)}_0)}_{ja},$$

$$c_{64} = \overline{(\eta^2 - \overline{(\eta^2)}_0)}_{ja}, c_{65} = \overline{(\xi\eta - \overline{(\xi\eta)}_0)}_{ja}; c_{71} = \overline{(\xi - \bar{\xi}_0)}_{jb}, c_{72} = \overline{(\eta - \bar{\eta}_0)}_{jb},$$

$$c_{73} = \overline{(\xi^2 - \overline{(\xi^2)}_0)}_{jb}, c_{74} = \overline{(\eta^2 - \overline{(\eta^2)}_0)}_{jb}, c_{75} = \overline{(\xi\eta - \overline{(\xi\eta)}_0)}_{jb}; c_{81} = \overline{(\xi - \bar{\xi}_0)}_{ka},$$

2.5 Third Order Reconstruction

To construct a third order linear scheme (a scheme is called linear if it is linear when applied to a linear equation with constant coefficients), we need a quadratic polynomial reconstruction. It seems that one robust way is the least square reconstruction suggested by Barth and Frederickson [4]. For the control volume of triangle $\triangle_0$, see Fig. 2.4.1, let $\triangle_i$, $\triangle_j$, $\triangle_k$ be its three neighbors, and $\triangle_{ia}$, $\triangle_{ib}$ be the two neighbors (other than $\triangle_0$) of $\triangle_i$, and so on. We determine the quadratic polynomial p^2 by requiring that it has the same cell average as u on $\triangle_0$, , and also it matches the cell averages of u on the triangles in the set

$$\{\triangle_i,\ \triangle_{ia},\ \triangle_{ib},\ \triangle_j,\ \triangle_{ja},\ \triangle_{jb},\ \triangle_k,\ \triangle_{ka},\ \triangle_{kb}\},$$

in a *least-square* sense (as this is an over-determined system). Notice that some of the neighbors' neighbors ($\triangle_{ia}$, $\triangle_{ib}$, $\triangle_{ja}, \cdots$) may coincide. For example, $\triangle_{ib}$ might be the same as $\triangle_{ja}$. This, however, does not affect the least square procedure to determine p^2.

Let (x_0, y_0) be the barycenter of $\triangle_0$, define $\xi = (x - x_0)/h$, $\eta = (y - y_0)/h$, $h = |\triangle_0|^{1/2}$, then $p^2(x, y)$ can be written as:

$$p^2(x,y) = a_0 + a_1\xi + a_2\eta + a_3\xi^2 + a_4\eta^2 + a_5\xi\eta. \tag{2.5.1}$$

From the requirement that $p^2(x, y)$ has cell average $\bar{u}_0$, we have

$$a_0 = \bar{u}_0 - (a_1\bar{\xi}_0 + a_2\bar{\eta}_0 + a_3\overline{(\xi^2)}_0 + a_4\overline{(\eta^2)}_0 + a_5\overline{(\xi\eta)}_0), \tag{2.5.2}$$

thus

$$\begin{aligned} p^2(x,y) = \quad & a_1(\xi - \bar{\xi}_0) + a_2(\eta - \bar{\eta}_0) + a_3(\xi^2 - \overline{(\xi^2)}_0) + \\ & a_4(\eta^2 - \overline{(\eta^2)}_0) + a_5(\xi\eta - \overline{(\xi\eta)}_0)) + \bar{u}_0. \end{aligned} \tag{2.5.3}$$

$y_i)/h_i$, $h_i = |\triangle_i|^{1/2}$, then $p(x,y)$ can be written as:

$$p(x,y) = \sum_{j=0}^{k} \sum_{s+l=j} a_{sl} \xi^s \eta^l \tag{2.4.2}$$

K interpolation conditions given by

$$\bar{p}_j = \bar{u}_j, \qquad j = 1, 2, \cdots, K. \tag{2.4.3}$$

will produce a $K \times K$ linear system for K unknowns a_{sl}.

Remark 1. We use the normalized variables ξ, η to make the coefficients a_{sl} independent of mesh sizes.

Remark 2. For our third and fourth order WENO schemes, we always use a control volume as shown in Fig. 2.4.1 for triangle $\triangle_0$. The stencil consists of $\triangle_0$, its three neighbor triangles, and their other two neighbors (totally 10 triangles in general).

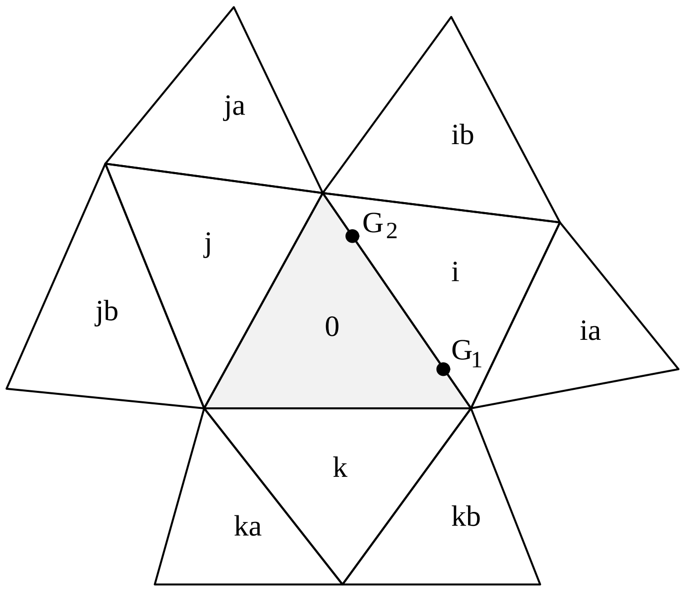

Figure 2.4.1: A typical stencil for triangle $\triangle_0$.

Remark. The fourth and fifth order TVD Runge-Kutta methods which are rather complicated can be found in [25]. For our schemes (both third and fourth order), we basically use the third order TVD Runge-Kutta method; while in examples of accuracy test for fourth order schemes , we use $\Delta t = (\Delta x)^{4/3}$ to achieve fourth order accuracy in the time direction.

2.4 Reconstruction by Mean Values

Let P^k denote the set of two-dimensional polynomials of degree less than or equal to k. The reconstruction problem, from cell averages to point values, is as follows: given a smooth function u, and a triangulation with triangles $\{\triangle_0, \triangle_1, ..., \triangle_N\}$, we would like to construct, for each triangle $\triangle_i$, a polynomial $p(x, y)$ in P^k that has the same mean value as u on $\triangle_i$, and is an $(k+1)$-th order approximation to u on the cell $\triangle_i$. The mean value (cell average) of a function $u(x, y)$ on a cell $\triangle_i$ is defined as

$$\bar{u}_i \equiv \frac{1}{|\triangle_i|}\int_{\triangle_i} u(x, y)dxdy. \tag{2.4.1}$$

In order to determine $K = \frac{(k+1)(k+2)}{2}$ degrees of freedom in a k-th degree polynomial p, we need to use the information of at least K triangles. In addition to $\triangle_i$ itself, we may take its $K-1$ neighboring cells, and we rename these K triangles as $S_i = \{\Omega_1, \Omega_2, ..., \Omega_K\}$, S_i is called a stencil for the triangle $\triangle_i$. If we require that p has the mean value $\bar{u}_j$ on Ω_j for all $1 \leq j \leq K$, we will get a $K \times K$ linear system. If this linear system has a unique solution, S_i is called an *admissible* stencil. Of course, in practice, we also have to worry about any ill conditioned linear system even if it is invertible. For linear polynomials $k = 1$, a stencil formed by $\triangle_i$ and two of its neighbors is admissible for most triangulations.

Let (x_i, y_i) be the barycenter of $\triangle_i$, define $\xi = (x - x_i)/h_i$, $\eta = (y -$

the triangle (inside the neighboring triangle) at the Gaussian point.

Since we are constructing schemes up to fourth order accuracy, two point Gaussian $q = 2$ is used, which has $G_1 = cP_1 + (1-c)P_2$, $G_2 = cP_2 + (1-c)P_1$, $c = \frac{1}{2} + \frac{\sqrt{3}}{6}$ and $\omega_1 = \omega_2 = \frac{1}{2}$ for the line with end points P_1 and P_2.

Taking together above discretization and approximations, we can get the following spatially discretized equations for (2.2.2):

$$\frac{d}{dt}\bar{u}_i(t) = L(\bar{u}_i(t)). \tag{2.2.5}$$

2.3 TVD Runge-Kutta Methods

(2.2.5) defines a system of ordinary differential equations. The TVD Runge-Kutta methods developed in [49], see also [25], are used to solve this system. The second and third order versions are as follows: for solving a system of initial value problems of ODEs

$$u_t = L(u), \tag{2.3.1}$$

the second order TVD Runge-Kutta method is given by

$$\begin{aligned} u^{(1)} &= u^n + \Delta t L(u^n) \\ u^{n+1} &= \frac{1}{2}u^n + \frac{1}{2}u^{(1)} + \Delta t L(u^{(1)}) \end{aligned} \tag{2.3.2}$$

and the third order TVD Runge-Kutta method is given by

$$\begin{aligned} u^{(1)} &= u^n + \Delta t L(u^n) \\ u^{(2)} &= \frac{3}{4}u^n + \frac{1}{4}u^{(1)} + \frac{1}{4}\Delta t L(u^{(1)}) \\ u^{n+1} &= \frac{1}{3}u^n + \frac{2}{3}u^{(2)} + \frac{2}{3}\Delta t L(u^{(2)}) \end{aligned} \tag{2.3.3}$$

2.2 Finite Volume Formulation

Consider the following two-dimensional conservation law:

$$\frac{\partial u}{\partial t} + \frac{\partial f(u)}{\partial x} + \frac{\partial g(u)}{\partial y} = 0 \tag{2.2.1}$$

where $u : \Re^2 \times \Re^+ \to \Re^m$ and $f, g : \Re^m \to \Re^m$.

Hyperbolicity means that any real linear combination of $\alpha f'(u) + \beta g'(u)$ of the flux Jacobian should be diagonalizable with real eigenvalues.

In this chapter, we solve (2.2.1) using the finite volume formulation. Computational control volumes are simply triangles.

Taking the triangle $\triangle_i$ as our control volume, we formulate the semi-discrete finite volume scheme of equation (2.2.1) as:

$$\frac{d}{dt}\bar{u}_i(t) + \frac{1}{|\triangle_i|}\int_{\partial\triangle_i} F \cdot n\, ds = 0 \tag{2.2.2}$$

where $\bar{u}_i(t)$ is the cell average of u on the cell $\triangle_i$, $F = (f, g)^T$, n is the outward unit normal of the triangle boundary $\partial\triangle_i$, and $|\triangle_i|$ is the area of $\triangle_i$.

The line integral in (2.2.2) is discretized by a q-point Gaussian integration formula,

$$\int_{\partial\triangle_i} F \cdot n\, ds \approx |\partial\triangle_i| \sum_{j=1}^{q} \omega_j F(u(G_j, t)) \cdot n \tag{2.2.3}$$

where $|\partial\triangle_i|$ is the length of $\partial\triangle_i$, and $F(u(G_j, t)) \cdot n$ is replaced by a numerical flux. The simple Lax-Friedrichs flux is used for all our numerical experiments in the next chapter, which is given by

$$F(u(G_j,t))\cdot n \approx \frac{1}{2}\left[\Big(F(u^-(G_j,t)) + F(u^+(G_j,t))\Big)\cdot n - \alpha\left(u^+(G_j,t) - u^-(G_j,t)\right)\right] \tag{2.2.4}$$

where α is taken as an upper bound for the eigenvalues of the Jacobian in the n direction, and u^- and u^+ are the values of u inside the triangle and outside

WENO schemes. Numerical examples will be given in the next chapter, to demonstrate the accuracy and resolution of the constructed schemes.

accuracy should be $(2k-1)$ if all the $2k-1$ cells in the potential stencils are used. Still another problem is that ENO schemes use many local logical arguments in choosing stencils which is not very efficient on vector supercomputers such as CRAYs.

WENO (Weighted ENO) schemes were developed to improve upon ENO schemes, in Liu, Osher and Chan [41] and Jiang and Shu [34]. The key idea for WENO is that instead of using only one of the candidate stencils to form reconstruction, one uses a convex combination of all of them to achieve possible higher order accuracy. Advantages of WENO schemes over ENO include the smoothness of numerical fluxes, better steady state convergence, and better accuracy using the same stencils. Levy, Puppo and Russo [39] designed one dimensional finite volume WENO schemes based on a staggered grid and Lax-Friedrichs formulation.

For a review of ENO and WENO schemes, see [48].

Friedrich [23] constructed WENO schemes on unstructured meshes using a co-volume formulation as in Abgrall [1]. The WENO schemes in [23] only achieve the same order of accuracy as the corresponding ENO schemes when the same set of stencils is considered. This is not optimal as was known in Jiang and Shu [34] for structured meshes.

In the following sections, we present higher order WENO schemes on triangular meshes when using the same set of ENO stencils. By using a combination of two-dimensional linear polynomials, we can get third order schemes while keeping the solution essentially non-oscillatory near discontinuities. Furthermore, by using a combination of two-dimensional quadratic polynomials, we can get fourth order schemes while keeping the solution essentially non-oscillatory near discontinuities. We will first describe the procedure to construct the high order linear schemes. The formulation at this stage is important to accommodate nonlinear WENO weights later. We then describe the third and fourth order

lems, for example in simulating shock turbulence interactions, Shu and Osher [50], Shu, Zang, Erlebacher, Whitaker and Osher [51], and Adams and Shariff [3]; in the direct simulation of compressible turbulence, Shu, Zang, Erlebacher, Whitaker and Osher [51], Walsteijn [57], and Ladeinde, O'Brien, Cai and Liu [36]; in solving the relativistic hydrodynamics equations, Dolezal and Wong citeDo; in shock vortex interactions and other gas dynamics problems, Casper and Atkins [9] and Erlebacher, Hussaini and Shu [21]; in incompressible flow calculations, E and Shu [20] and Harabetian, Osher and Shu [26]; in solving the viscoelasticity equations with fading memory, Shu and Zeng [52]; in semiconductor device simulation, Fatemi, Jerome and Osher [22] and Jerome and Shu [30], [31]; in image processing and level set methods, Osher and Sethian [43], Sethian [47], and Siddiqi, Kimia and Shu [53]. The original ENO paper by Harten, Osher, Engquist and Chakravarthy [29] was for a one dimensional finite volume formulation. Later, this finite volume formulation of ENO schemes has been extended to two dimensional structured meshes by Harten [27] and by Casper [8], and to unstructured triangular meshes by Abgrall [1], Harten and Chakravarthy [28], and Sonar [55]. Finite volume ENO schemes based on a staggered grid and Lax-Friedrichs formulation were given in Bianco, Puppo and Russo [6]. Although finite difference versions of ENO schemes [49, 50] are more efficient for multidimensional calculations, finite volume schemes have the advantage of easy handling of complicated geometry by arbitrary triangulations.

ENO schemes also have certain drawbacks. One problem is the stencil might change even by a round-off error perturbation near zeros of the solution and its derivatives. It's obvious that this free adaptation of stencils is not necessary in smooth regions, and this may cause loss of accuracy. Another problem is that the adaptive stencil won't give the optimal order accuracy. In the stencil choosing process, k candidate stencil are considered, covering $2k-1$ cells, but only one stencil is actually used, resulting in $k-$th order accuracy. The optimal order

2.1 Some Technical Backgrounds

A major challenge in designing high-order finite difference and finite volume methods for the numerical solution of hyperbolic conservation laws has been how to eliminate or reduce spurious oscillations near discontinuities without destroying accuracy in smooth regions. There were mainly two ways to tackle this issue prior to 1987. One way was to add some artificial viscosity, which could be tuned so that it was large enough near the discontinuity to suppress, or at least to reduce oscillations, but was small elsewhere to maintain accuracy. One disadvantage of above approach is that fine tuning of the parameter controlling the size of the artificial viscosity is problem dependent. Another way was to apply limiters to eliminate the oscillations. In fact, one reduced the order of accuracy of the method near the discontinuity. by carefully designing such limiters, the TVD (total variation diminishing) property could be achieved for nonlinear scalar one dimensional problems. One disadvantage of this approach is that accuracy necessarily degenerates to first order near smooth extrema and except in trivial cases, any method that is TVD in two dimensions is at most first order accurate (see [24]).

The ENO (essentially non-oscillatory) idea proposed by Harten, Osher, Engquist and Chakravarthy in [29] was the first successful attempt to obtain a self similar (no mesh size dependent parameters), uniformly high order accurate, yet essentially non-oscillatory method. The vital idea of ENO schemes is to use the "smoothest" stencil among several candidates to approximate the fluxes at cell boundaries to a high order accuracy and at same time to avoid spurious oscillations near discontinuities. Various numerical examples show that ENO schemes are indeed uniformly high order accurate and resolve shocks with sharp and monotone transitions.

ENO schemes ([29], Shu and Osher [49, 50]) have been successfully applied to solve hyperbolic conservation laws and other convection dominated prob-

Chapter 2

Weighted Essentially Non-oscillatory Schemes on Triangular Meshes

In this chapter we construct high order weighted essentially non-oscillatory (WENO) schemes on two dimensional unstructured meshes (triangles) in the finite volume formulation. We construct third order schemes using a combination of linear polynomials, and fourth order schemes using a combination of quadratic polynomials.

our high-order WENO schemes in details with some numerical examples to demonstrate the accuracies and robustness of the methods for two dimensional shock calculations. Then, the method of discontinuous Galerkin finite element formulation for HJ equations will be discussed with some one and two dimensional numerical examples to illustrate its capability.

be strategically placed in the computational domain. Thus, these meshes are particularly effective in modeling irregular boundaries, multiscale geometries, and rapidly changing solutions.

This dissertation tries to design and analyze high-order numerical methods for hyperbolic equations on unstructured meshes. There are basically two categories of methods on unstructured meshes. One is finite element methods based on a discontinuous Galerkin formulation for hyperbolic problems. Another one is finite volume schemes based on the idea of updating cell averages by fluxes through the boundary of the cell. The discontinuous Galerkin formulation uses high-order local approximations on general geometry, and provide a natural framework for finite element approximations and for theoretical developments. The finite volume formulation uses high-order reconstruction procedures based on arbitrary triangulation or other mesh types to obtain point values needed in computing fluxes.

The first focus of this work is on high-order weighted essentially non-oscillatory (WENO) schemes on two dimensional unstructured meshes (triangles) in the finite volume formulation. We present third order schemes using a combination of linear polynomials in reconstructions, and fourth order schemes using a combination of quadratic polynomials. Essentially non-oscillatory property is achieved by adding some nonlinear weights to the combination coefficients.

The second focus of this work is on a discontinuous Galerkin finite element method for solving the nonlinear Hamilton-Jacobi (HJ) equations which are frequently encountered in control theory, differential games and computer image processing. The method is based on the Runge-Kutta discontinuous Galerkin finite element method for solving conservation laws. HJ equations are closely related to conservation laws in a way that the former can be treated as some "integrated version" of the latter, this observation dictates our method.

This dissertation will be organized in the following way. First, we describe

Chapter 1

Introduction

The numerical solution of hyperbolic systems is always an interesting subject both for engineers and scientists. People have been working on this issue for many decades, especially since computers were invented. Despite of huge achievements, people are aware that much remains to be done in this field. This is particularly true in the numerical analysis for hyperbolic conservation laws which include the Euler equations of gas dynamics and for Hamilton-Jacobi equations. Here, two questions are most important. How to numerically resolve shocks and discontinuities while simultaneously producing accurate, non-oscillatory results elsewhere in the computational domain? How to obtain the most accurate results for a fixed computational resource?

Traditionally, finite difference methods are used to solve hyperbolic equations. Since they are developed from one-dimensional cases, finite difference methods require a structured mesh for problems in two and three dimensions, hence are difficult to apply to problems involving realistic geometries or for adaptive mesh refinements. On the other hand, unstructured meshes are a powerful computational tool used in the numerical modeling of physical phenomena on complex, irregular domains. Instead of requiring a systematic (or ordered) distribution of grid points, unstructured meshes allow grid points to

List of Figures

List of Tables

Contents

Acknowledgments

I wish to express my sincere gratitude to my advisor and friend, Professor Chi-Wang Shu, for his patient and invaluable guidance throughout this work. I have learnt a lot from him, both in skills to do research and in attitudes to pursue my academic goal; this work would not have been possible without his encouragement and teaching.

I would like to thank Professor David Gottlieb for providing me an opportunity to come to Brown University in 1995. I also want to thank David for his great help through my study here, especially on spectral methods.

I want to give my thanks to Professor Jan Hesthaven for his advice and teaching on spectral methods. My thank should also go to Professor Wai Sun Don, I enjoy the cooperation with him on the project of multidomain shock calculations by spectral methods, I have learnt a lot from him, especially on good programming styles. I also want to thank Professor Paul Fischer for his advice and teaching on parallel computing.

Abstract of "Numerical Methods for Hyperbolic Equations on Unstructured Meshes," by Changqing Hu, Ph.D., Brown University, May 1999

In this thesis, two high order numerical methods for hyperbolic equations are presented. The first focus of this work is on high-order weighted essentially non-oscillatory (WENO) schemes for solving conservation laws on two dimensional unstructured meshes in the finite volume formulation. We construct third order schemes using a combination of linear polynomials in reconstructions, and fourth order schemes using a combination of quadratic polynomials. Essentially non-oscillatory property is achieved by adding some nonlinear weights to the combination coefficients. The second focus of this work is on discontinuous Galerkin finite element methods for solving the nonlinear Hamilton-Jacobi equations The methods are based on the Runge-Kutta discontinuous Galerkin finite element method for solving conservation laws. The methods have the flexibility of treating complicated geometry by using arbitrary triangulation, can achieve high order accuracy with a local, compact stencil.

Numerical Methods for Hyperbolic Equations on Unstructured Meshes

by

Changqing Hu

M.S., University of Pittsburgh, 1995
B.S., Peking University, China, 1986

Thesis

Submitted in partial fulfillment of the requirements for
the Degree of Doctor of Philosophy
in Division of Applied Mathematics at Brown University

May 1999