Numerical PDE II HW 4 : Grid Generation part II

Dhruv Balwada

April 6, 2015

1 Statement of Problem

Generate two more grids for the nozzle of homework 3 with and without using control functions.

2 Description of the Mathematics

2.1 Elliptic grid generation

We start by using the result shown in problem 2 of homework 3.

$$\nabla^2 f = \sum_{i,j=1}^{3} g^{ij} f_{\zeta^i \zeta^j} + \sum_{i=1}^{3} \nabla^2 \zeta^i f_{\zeta^i} \tag{1}$$

Now choosing $f = \vec{X}$ we get $\nabla^2 \vec{X} = 0$.

In 2D this gives

$$\alpha \vec{X}_{\zeta\zeta} - 2\beta \vec{X}_{\zeta\eta} + \gamma \vec{X}_{\eta\eta} = -J^2 (P\vec{X}_\zeta + Q\vec{X}_\zeta) \tag{2}$$

Where $\alpha = \vec{X}_\eta^2 + \vec{Y}_\eta^2$, $\beta = \vec{X}_\zeta \vec{X}_\eta + \vec{Y}_\zeta \vec{Y}_\eta$ and $\gamma = \vec{X}_\zeta^2 + \vec{Y}_\zeta^2$. Also $P = \nabla^2 \zeta$ and $Q = \nabla^2 \eta$ are the control functions.

We can solve this system numerically to find $\vec{X}(\zeta, \eta)$.

For the numerical solution we will discretize the system as follows :

$$\alpha_{ij} = (\delta_\eta^o X_{ij})^2 + (\delta_\eta^o Y_{ij})^2$$

$$\beta_{ij} = (\delta_\zeta^o X_{ij})(\delta_\eta^o X_{ij}) + (\delta_\zeta^o Y_{ij})(\delta_\eta^o Y_{ij})$$

$$\gamma_{ij} = (\delta_\zeta^o X_{ij})^2 + (\delta_\zeta^o Y_{ij})^2$$

$$J = (\delta_\zeta^o X_{ij})(\delta_\eta^o X_{ij}) - (\delta_\zeta^o Y_{ij})(\delta_\eta^o Y_{ij})$$

$$\alpha_{ij}\delta_\zeta^+ \delta_\zeta^- \vec{X}_{ij} - 2\beta_{ij}\delta_\zeta^o \delta_\eta^o \vec{X}_{ij} + \gamma \delta_\eta^+ \delta_\eta^- \vec{X}_{ij} = -J_{ij}^2 (P_{ij}\delta_\zeta^o \vec{X}_{ij} + Q_{ij}\delta_\zeta^o \vec{X}_{ij})$$

This is solved using SOR scheme.

Boundary conditions:

The boundary conditions can be set as Dirichlet or Neumann. The Dirichlet conditions are set on the wavy boudaries(like the top line of the nozzle) where we know the value of X or Y. Sometimes we want to just set the orientation of the grid to the boundary and not worry about the actual value. In that case we set Neumann conditions, which is essentially defining the covariant basis vectors.

$$\vec{a_i} = \frac{\partial \vec{X}}{\partial \zeta_i}$$

So for setting the grid to be orthogonal at the left boundary we want $\vec{a_\zeta}.\hat{j} = \partial y/\partial \zeta = 0$. We can also set the covariant basis vector to be at an angle to the boundary. If we wanted the covariant basis vector(in the direction of ζ) to be at an angle θ to the boundary we ccould set that be setting $(\partial y/\partial \zeta)/(\partial x/\partial \zeta) = tan(\theta)$. We also choose $X_{N,j} = X_{max}$ as a Dirichlet condition. So we have $Y_{N,j} = Y_{N-1,j} + tan(\theta)(X_{max} - X_{N-1,j})$. We will use this to set the boundary condition at the right boundary and vary angle linearly from 0 to 15.

The control functions are chosen to be as follows:

$$P(\zeta, \eta) = -asign(\zeta - \zeta_o)e^{-b(\zeta - \zeta_o)} \tag{3}$$

Where a and b are parameters and $\zeta = \zeta_o$ is the line around which we want to concentrate the grid points. Similarly Q can be defined for focussing points in the η direction. Slight modifications can be made to the control functions to focus the grid around a point etc.

Steger and Sorenson(JCP 1979) Vol 33 describe an algorithm to control both the spacing and the angle of the grid using control functions. Suppose we want to impose both the grid spacing $\triangle s = constant$ and orthogonality at $\eta = 0$. We start with the form

$$P(\zeta, \eta) = P_1(\zeta)e^{-b\eta}$$

$$Q(\zeta, \eta) = Q_1(\zeta)e^{-b\eta}$$

Condition 1:

$$\triangle s = \sqrt{\triangle x^2 + \triangle y^2} = constant$$

In the limit

$$\frac{ds}{d\eta} = \sqrt{X_\eta^2 + Y_\eta^2}$$

Condition 2:

$$\nabla \zeta . \nabla \eta = 0$$

In 2D

$$\zeta_x \eta_x + \zeta_y \eta_y = 0$$

Under transformation

$$-(Y_\zeta Y_\eta + X_\zeta X_\eta) = 0$$

So

$$\frac{Y_\zeta}{X_\zeta} = -\frac{X_\eta}{Y_\eta}$$

$$\frac{ds}{d\eta} = Y_\eta\sqrt{X_\eta^2/Y_\eta^2+1} = Y_\eta\sqrt{Y_\zeta^2/X_\zeta^2+1} = Y_\eta/X_\zeta\sqrt{Y_\zeta^2+X_\zeta^2}$$

Then solve for η derivatives

$$Y_\eta = \frac{s_\eta X_\zeta}{\sqrt{X_\zeta^2+Y_\zeta^2}}$$

$$X_\eta = -\frac{s_\eta Y_\zeta}{\sqrt{X_\zeta^2+Y_\zeta^2}}$$

So along $\eta = 0$

$$\vec{R} = (P_1\vec{X_\zeta} + Q_1\vec{X_\eta})$$

$$R_1 = \frac{\alpha X_{\zeta\zeta} - 2\beta X_{\zeta\eta} + \gamma X_{\eta\eta}}{-J^2}$$

$$R_2 = \frac{\alpha Y_{\zeta\zeta} - 2\beta Y_{\zeta\eta} + \gamma Y_{\eta\eta}}{-J^2}$$

Solve for P_1 and Q_1

$$P_1(\zeta) = (Y_\eta R_1 - X_\eta R_2)/J \tag{4}$$

$$Q_1(\zeta) = -(Y_\zeta R_1 - X_\zeta R_2)/J \tag{5}$$

So P_1 and Q_1 depend on the grid. Solve with the grid:

$$P_1^{k+1} = P_1^k + \omega[(Y_\eta R_1 - X_\eta R_2)/J - P_1^k] \tag{6}$$

$$Q_1^{k+1} = Q_1^k + \omega[-(Y_\zeta R_1 - X_\zeta R_2)/J - Q_1^k] \tag{7}$$

with small ω.

3 Description of the Algorithm

Here I present the algorithms and describe the main code segments that contribute to the method.

3.1 nozzle elliptic

The main function that calls other functions and carries out the skeleton of the elliptical grid generation.

Initialize ζ and η.
Initialize the initial X and Y using the output of the transfinite interpolation(from homework 3).
Call boundary subroutine to set the proper boundary conditions.
Call forcing or Sorenson and Steger subroutine to set control functions P and Q.
for k=1 to kmax **do**
  RESMAX =0

for j=1 to N-1 **do**
for i = 1 to N-1 **do**
Define α and γ.
$D = -2 * (\alpha + \gamma)/h^2$
Call residual function to calculate the residual.
$\vec{X}(i,j) = \vec{X}(i,j) + 1/D * \omega * residual$
resmax = MAX(ABS(residual),resmax)
end for
Call boundary subroutine to update boundary values incase of neumann boundary conditions.
If we do SSOR then just run the above loop in reverse
If resmax is less than some tolerance then exit.
Update control functions if required.
end for
end for

3.2 boundary

This subroutine sets the boundary conditions as explained in the previous section. The top boundary was specified as a Dirichlet boundary. The left and bottom boundaries were specified as Neumann to make sure that the grid was orthogonal. At the right boundary the angle of the grid was varied from 0-15degrees using a Neumann condition.

3.3 forcing

This subroutine calculates the control functions based on the equation introduced in the previous section. In the initial runs this was set to zero. Later we chose

$$P = -0.2 sign(\zeta - 0.5) e^{(-0.7(\zeta - 0.5))}$$

$$Q = -1.1 sign(\eta - 1) e^{(-0.7(\eta - 1))}$$

These P and Q were chosen to focus grid points along $\zeta = 0.5$(approximately near the throat) and $\eta = 1.0$(top boundary). Experimentation was done with the parameters but here we only show one case where the effect of control functions is obviously visible.

3.4 residual

This subroutine calculates the LHS and RHS of equation 2 shown in previous section. Then the residual is defined as the RHS - LHS.

3.5 Steger and Sorenson

There are two subroutines that are written to calculate the control functions as described in Steger and Sorenson(1979). The first routine `initial_control_functions_ss.f90` calculates the initial P and Q based on equation 4 and 5 and using the exponential spreading in

the η direction. The initial grid generated by transfinite interpolation is used for the initial condition. The second routine `control_functions_ss.f90` updates the control functions as the iterations progress. This calculates the updated control functions bases on equation 6 and 7. For every step of the grid the control functions are updated.

4 Results

Results for elliptically generated grids are shown in the figures. The key results are that the grids are smoother as they are harmonic(or solutions to poisson equation).

The derivatives inside the domain for transfinite interpolation grids depended linearly on the derivatives near the boundary. So a non-smooth boundary will give a non-smooth grid inside most of the domain for transfinite interpolation. However for the elliptic grids the derivatives on the boundary are concentrated near the boundary and decay away from the boundary within a few grid points. The higher derivatives that were problematic near the throat propagated all the way inside the domain for transfinite but for elliptic grids they are limited to and close to the boundary. This decay is faster(spatially) in finer grids as can be seen coming figures 2 and 4. This implies that as the grid gets finer the non-existent derivatives can be limited very close to the boundary and not affect much of the interior domain.

The grids generated with control functions show the properties that were expected. The parameters have been chosen so that the grid points are focussed near the top and near the throat as can be seen. The runs with control functions did change the structure of the derivatives as would be expected. Lower values of $dX/d\zeta$ can be seen along the line where the grid was concentrated near the throat(slightly to the left of the throat). It was noticed that if the control functions were tuned to make the mesh size too small(very concentrated) the solver would not make a proper grid and some grid points will jump outside the domain.

Higher order derivatives are presented for comparison among different grids.

5 Extra results

The Steger and Sorenson approach to generating a grid was also implemented. Here I show results from one test case where the control functions force the grid to be orthogonal and have a constant spacing near the top boundary of the nozzle. The decay scale of the exponential was chosen to be 1. The results are presented in Figure 9. One would expect the grid spacing to be more uniform and a grid that is more orthogonal than the one without control functions. Figure 9(c) and (d) compare the $ds/d\eta$ and Figure 9(e) and (f) compare the term $\nabla\zeta.\nabla\eta$ for the grids generated with and without control. It is seen clearly that the grid spacing is much more uniform with control. However the bottom edge of the grid, which is farther away from the top boundary where the control is applied, has a more inhomogeneous grid spacing. Also the grid has been made uniform near the top edge but worse in terms of orthogonality near the bottom edge.

The grid is doing what is expected but the "No free lunch theorem" applies.

6 Conclusions

Grids were generated using elliptic grid generation technique in a nozzle. The grids are smoother than the ones presented in the previous homework assignment using transfinite interpolation and allow control of increasing(or decreasing) the resolution in a certain area of the domain. Also as a side note control functions to apply constant grid spacing and orthogonality were implemented.

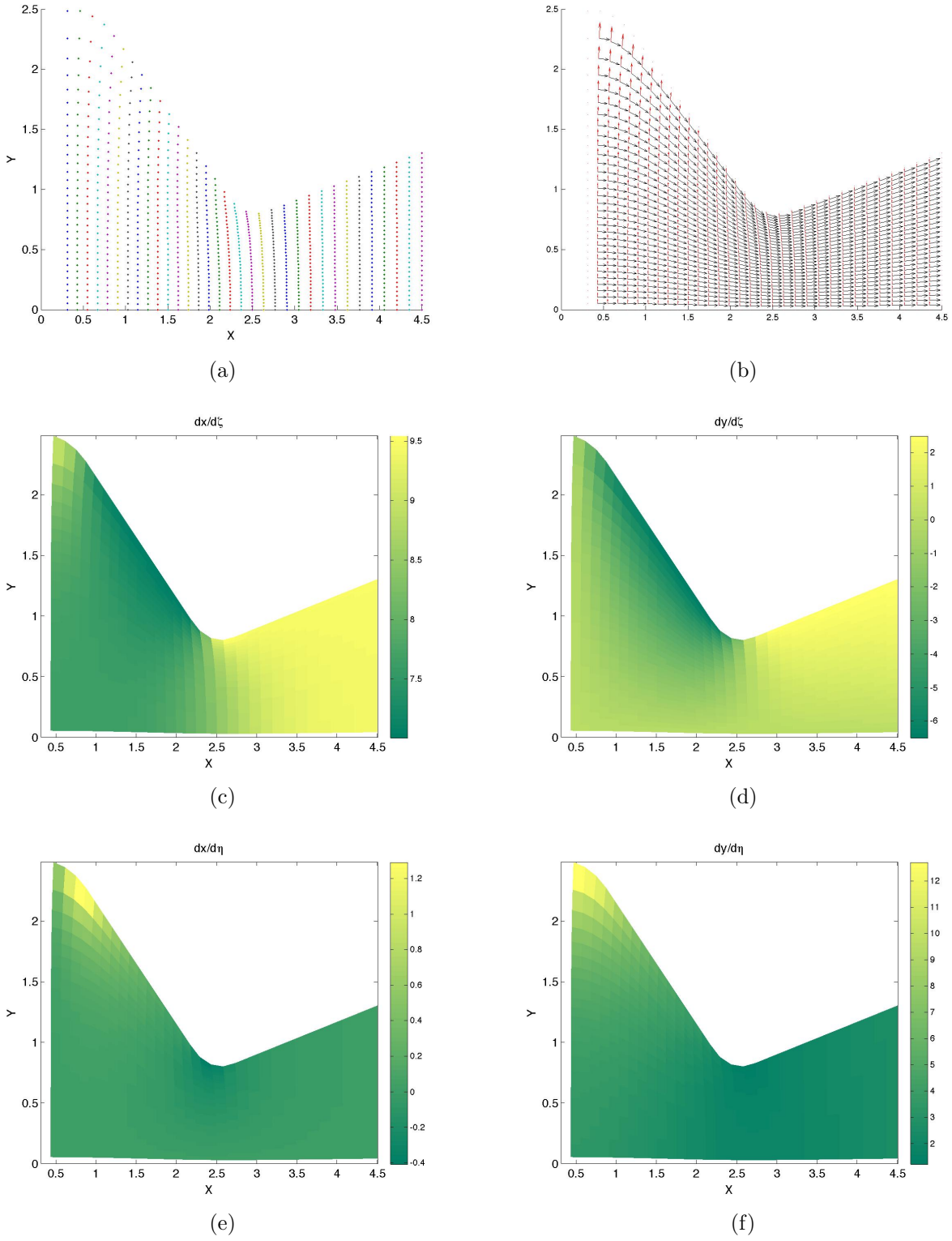
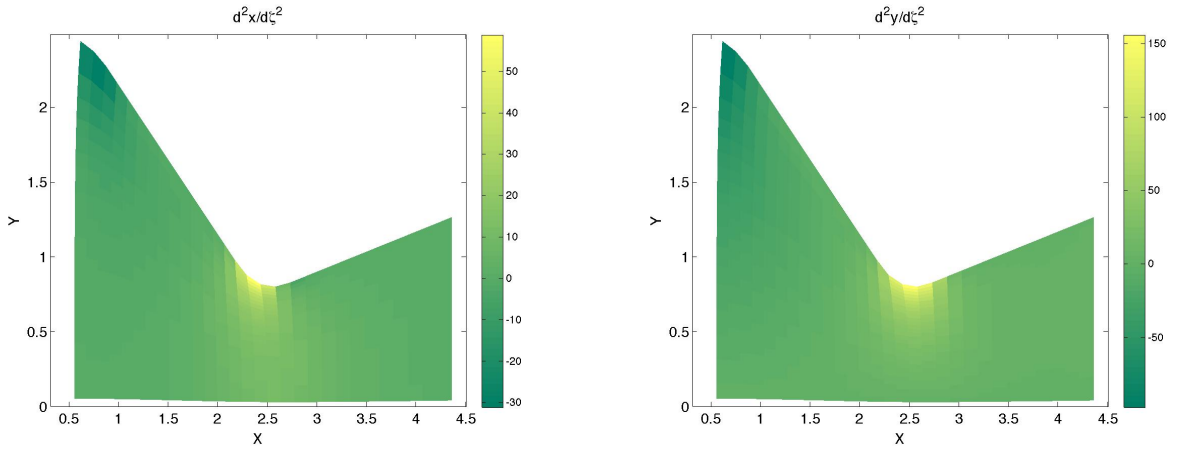
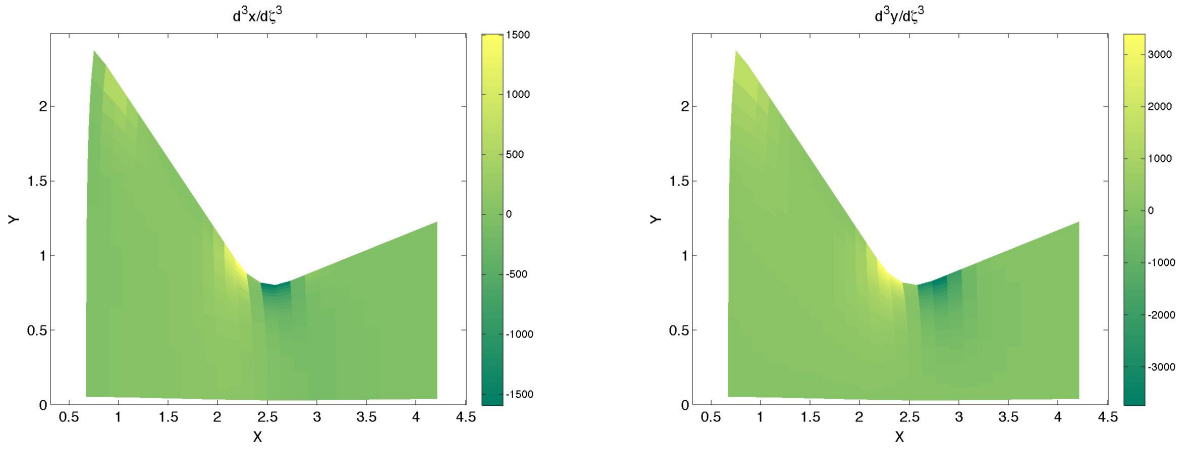


Figure 1: 32X32 grid generated for a nozzle using elliptic grid generation without control functions.



(a) (b)



(c) (d)

Figure 2: Higher derivatives on the 32X32 grid.

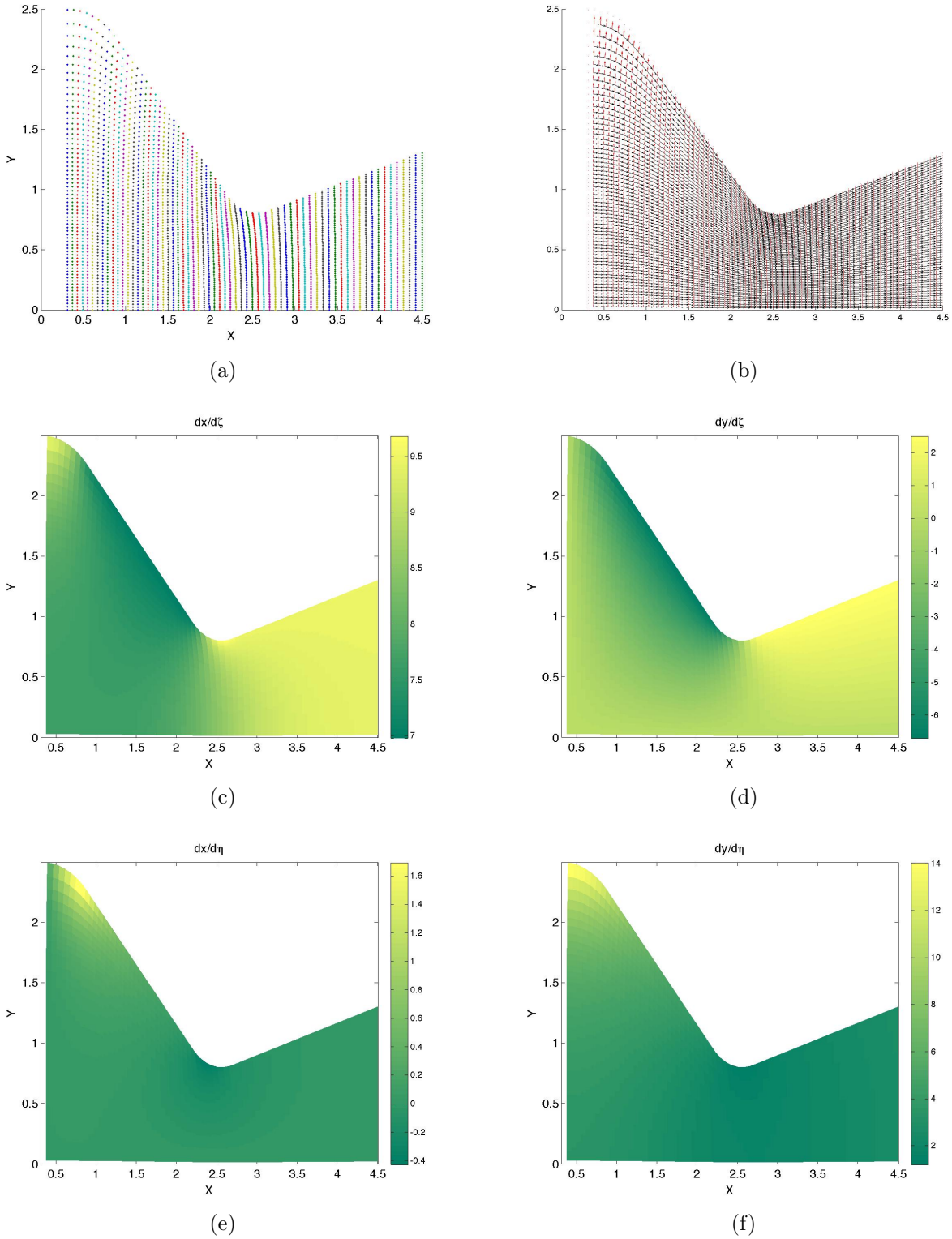
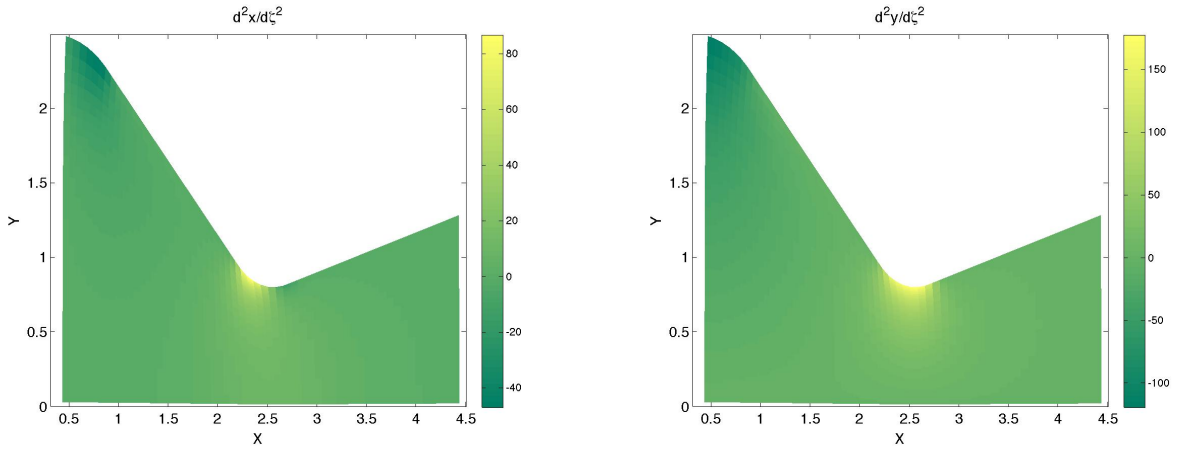
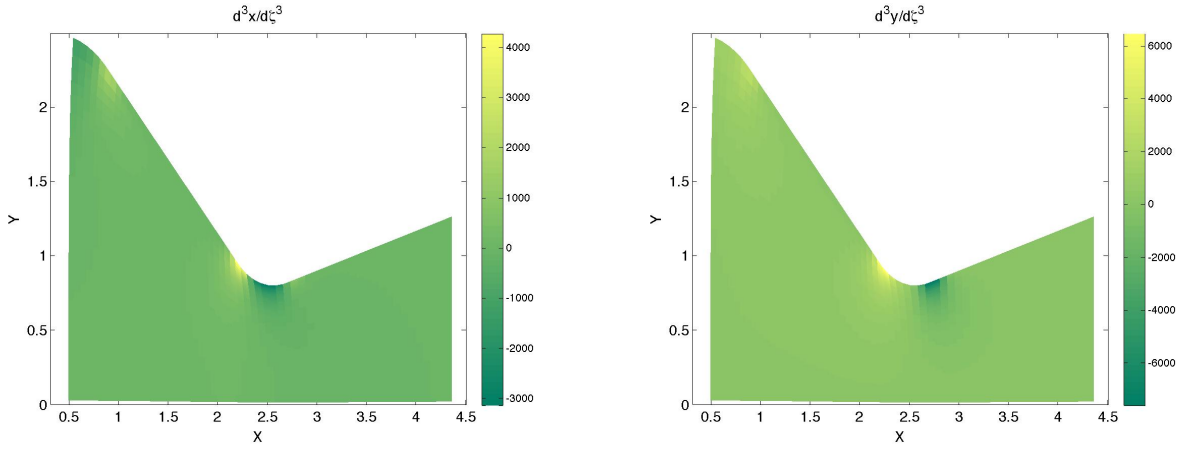


Figure 3: 64X64 grid generated for a nozzle using elliptic grid generation without control functions.



(a) (b)



(c) (d)

Figure 4: Higher derivatives on the 64X64 grid.

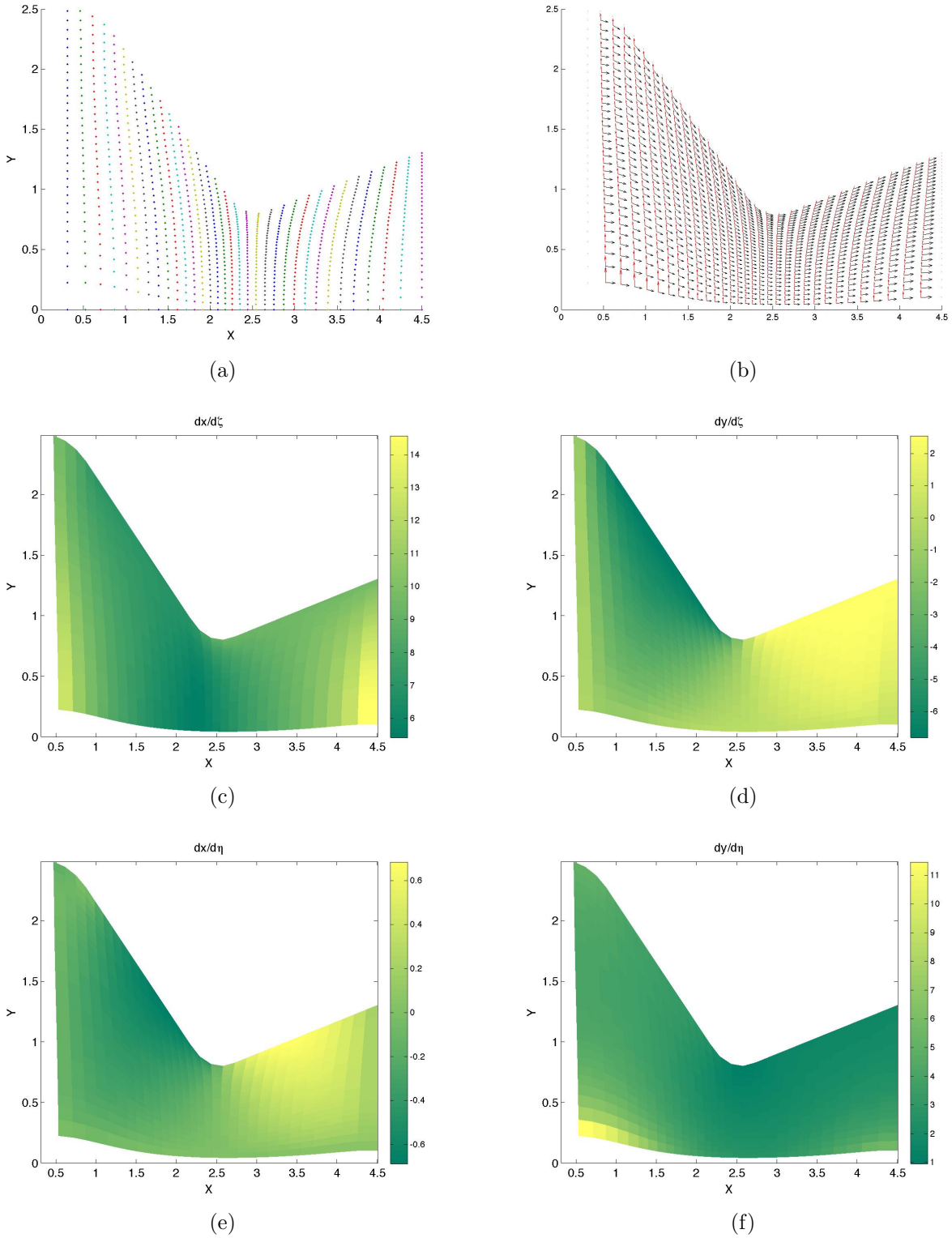
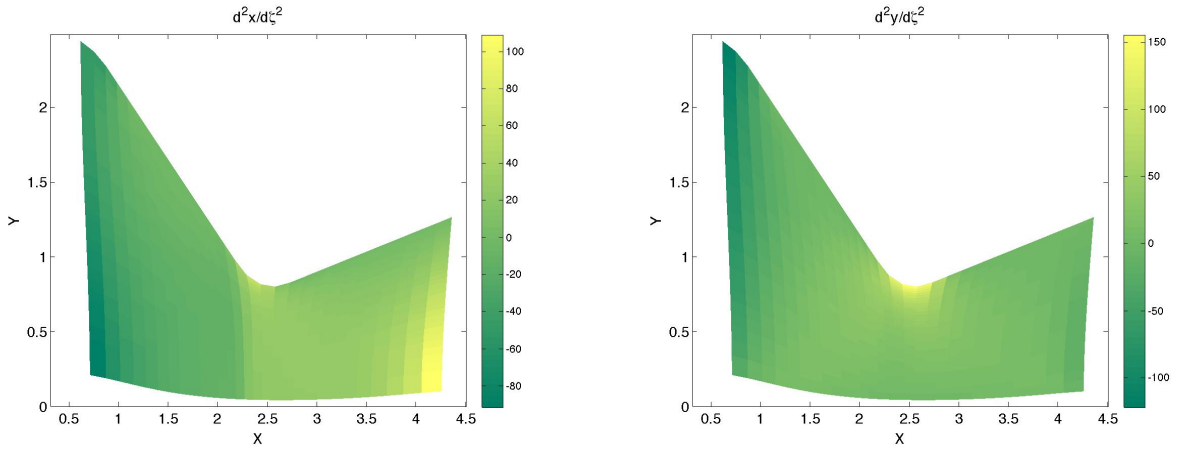
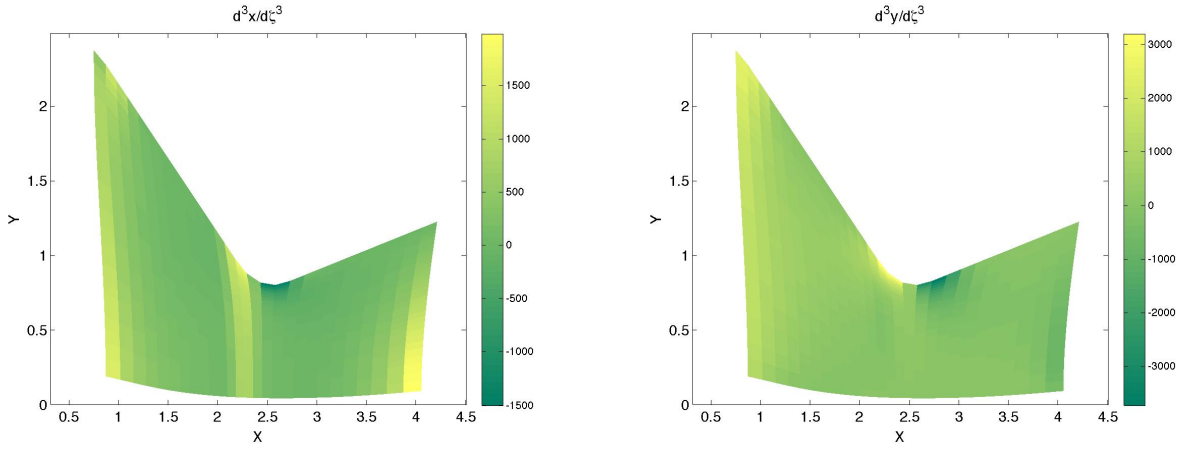


Figure 5: 32X32 grid generated for a nozzle using elliptic grid generation with control functions to to focus grid points near throat and upper wall.



(a) (b)



(c) (d)

Figure 6: Higher derivatives on the 32X32 grid.

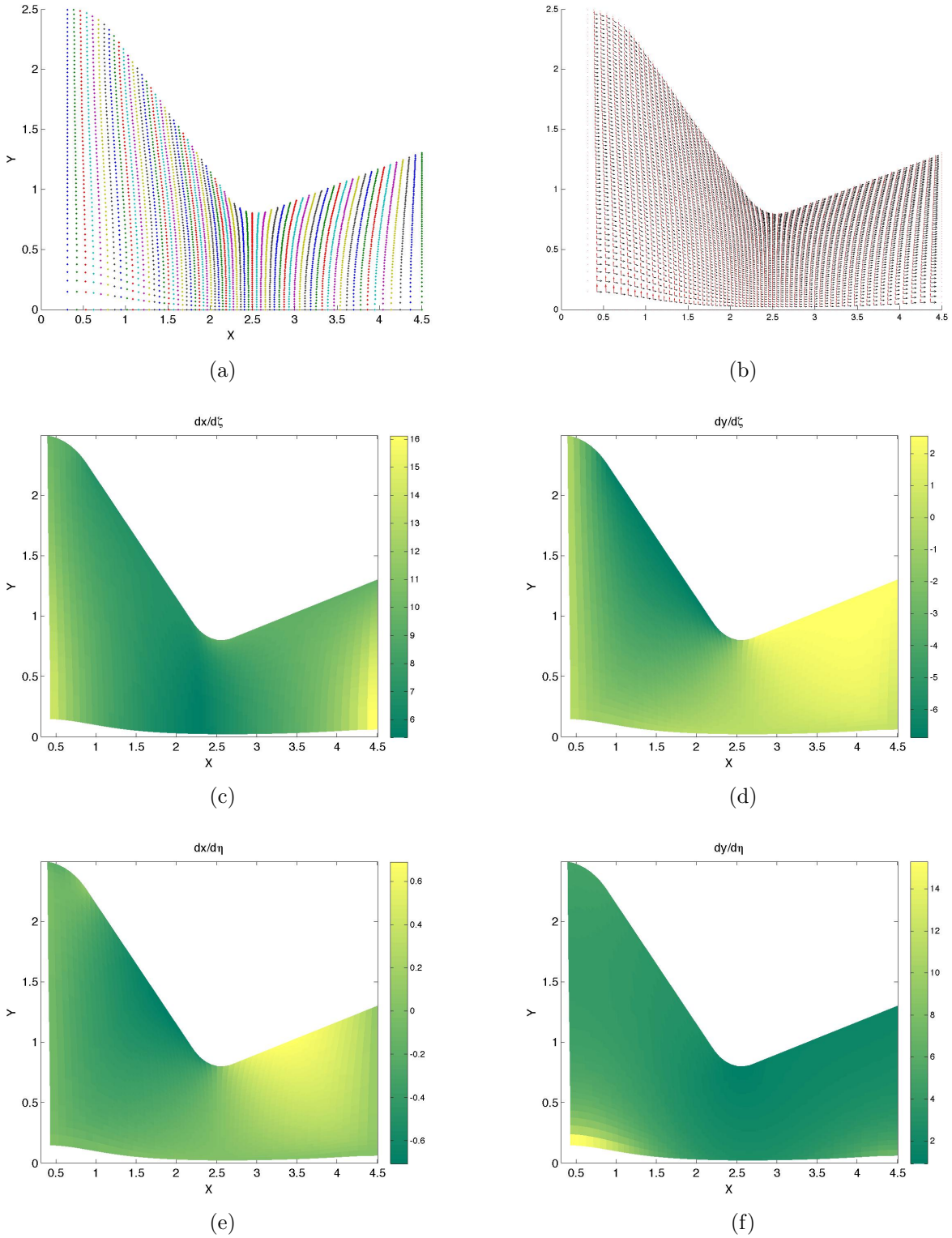


Figure 7: 64X64 grid generated for a nozzle using elliptic grid generation with control functions to to focus grid points near throat and upper wall.

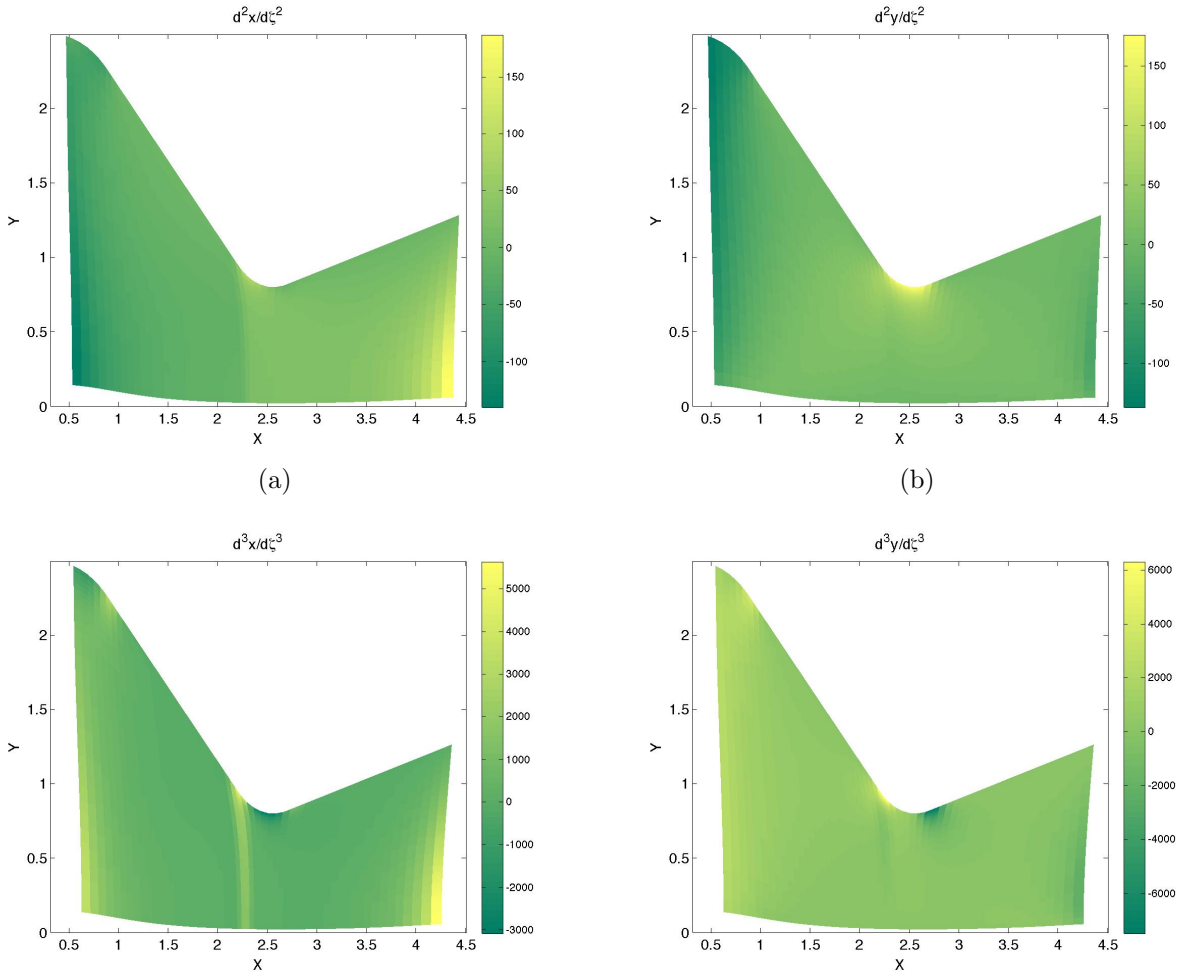
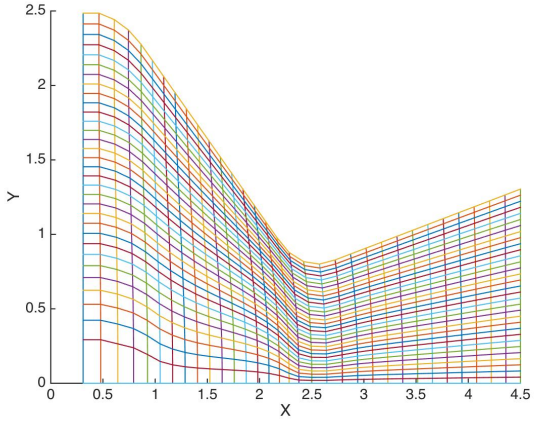
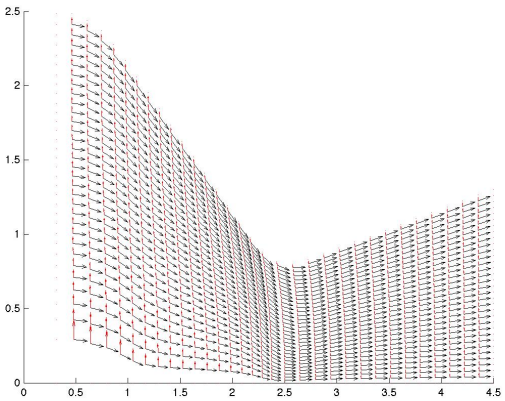


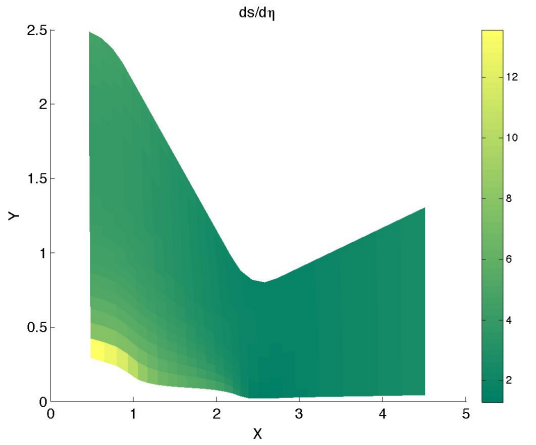
Figure 8: Higher derivatives on the 64X64 grid.



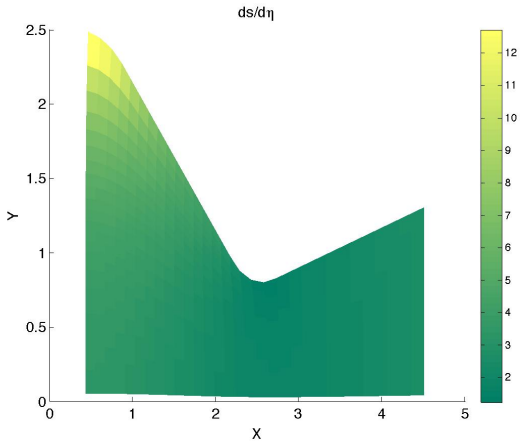
(a)



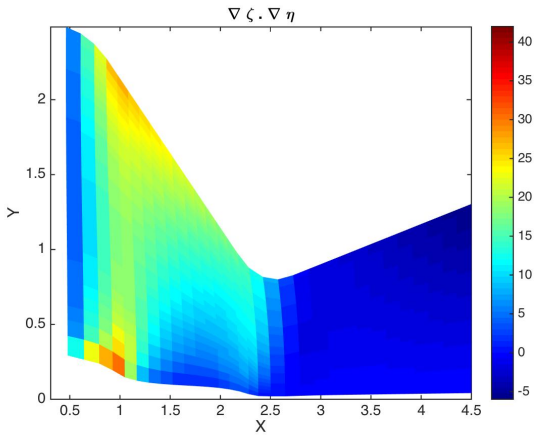
(b)



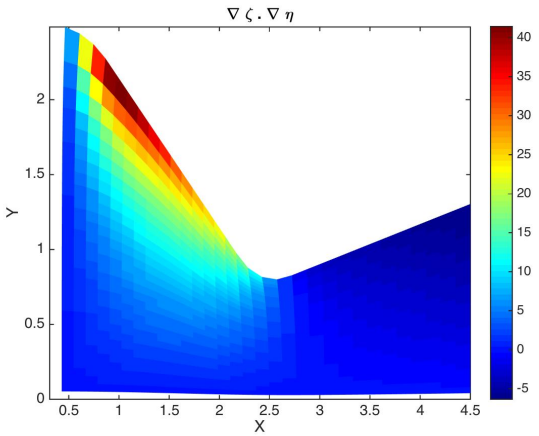
(c) With SS control



(d) Without any control functions



(e) With SS control



(f) Without any control functions

Figure 9: 32X32 grid generated for a nozzle using elliptic grid generation with control functions to control grid spacing and orthogonality near the top boundary.