Curved Mesh Generation for High-Order Finite Element Methods in the Context of Turbulent Flow Simulation

Seminar Thesis

by

Thomas Ludescher

Supervisor: Michael Woopen, M.Sc.

Rheinisch Westfälische Technische Hochschule Aachen

Aachen Institute for Advanced Study in
Computational Engineering Science

Aachen 2014

Contents

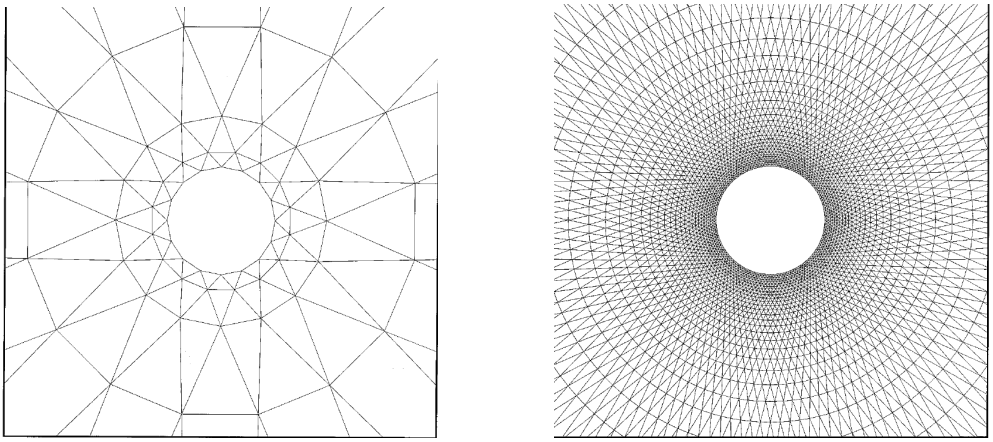
1 Introduction

High-order methods became quite popular in the area of computational fluid dynamics, to effectively resolve complex flow features using meshes which are reasonable for today's computers [9]. Whereupon the high accuracy is obtained with lower computational costs than lower order methods do [11], this is due to the fact that for the same accuracy lower order methods need a much finer mesh. State of the art methods in industry usually employ schemes relying on linear interpolation. This is mostly due to their favorable robustness properties. This means that one usually obtains a result although it may not be very accurate [9]. A method of choice in the high-order community is the discontinuous Galerkin method, which is very flexible in terms of arbitrary triangulation and boundary conditions [9]. Reasons for why high-order methods are rarely used are for example the higher memory requirements, the lower robustness due to the much reduced numerical dissipation and that robust high-order mesh generators are not readily available [11].

Motivated by the last reason, this paper presents a method to generate a mesh of second order where a linear mesh is provided as an input. Especially meshes for turbulence modeling are considered due to their strong requirements on grid resolution and thus stretched elements which are difficult to handle. The paper starts with a section showing the necessity of using high-order meshes in combination with high-order methods based on an example of subsonic flow around a circle. Afterwards the idea of generating a curved mesh with a given linear mesh through the elasticity analogy is described. Thereafter, a section is devoted to the mathematical backgrounds of linear elasticity. Then, the modifications of the equations are described which are helpful to gain more control over the element quality after deformation. The section is followed by an illustration of the implementation into the open source general purpose finite element library NGSolve. Then, results of described methods are presented for a sample problem and an application case. The paper ends with a summary and an outlook for further extensions and future topics.

2 Motivational Example - Flow around a Circle

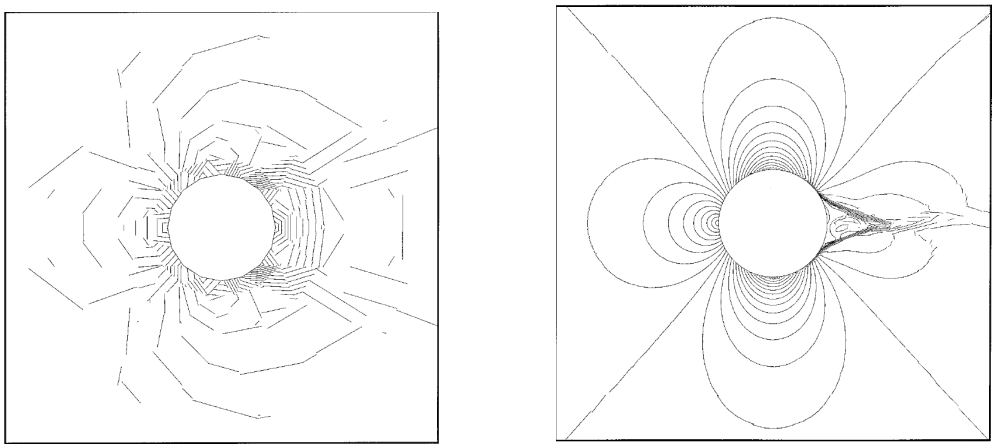
In this section an example from [1] is illustrated to show the need of high-order geometrical approximation in presence of high-order computations. The model problem is a subsonic flow around a circle, the two dimensional compressible Euler equations for steady state are used as a mathematical model. The freestream Mach number is chosen to be Ma = 0.38. A discontinuous Galerkin method is chosen as a discretization scheme. This combines the benefit of the finite element method, namely high-order polynomial interpolation within an element and that of the finite volume method, namely the discontinuous approximation at the element boundaries and therefore the solution of a Riemann problem. In this paper, only the coarsest and the finest mesh from [1] are discussed, see Figure 1.



(a) Coarse Mesh 16x4. (b) Fine Mesh 128x32.

Figure 1: Meshes for Computation. [1]

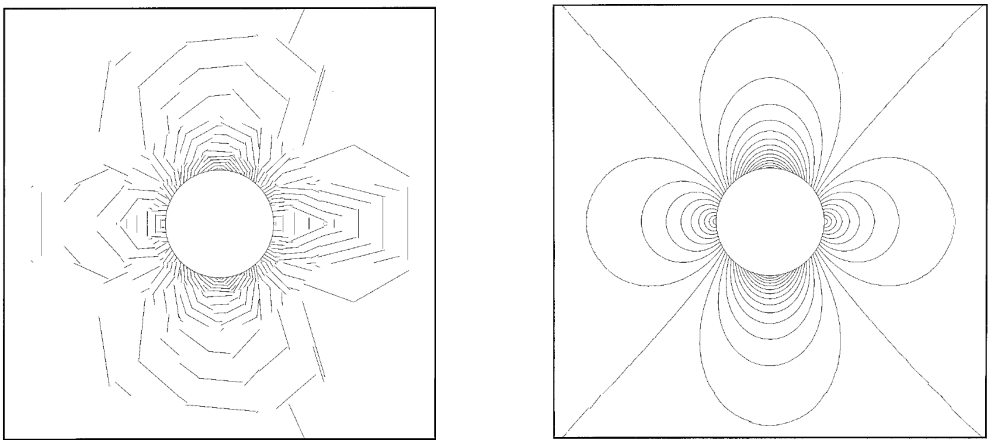
In the following different elements are denoted by P_kQ_m, where k denotes the order of polynomial interpolation for the unknown and m denotes the order for the geometrical approximation. Figure 2 shows the Mach isolines for P_1Q_1 elements.



(a) Coarse Mesh. (b) Fine Mesh.

Figure 2: Mach isolines P_1Q_1. [1]

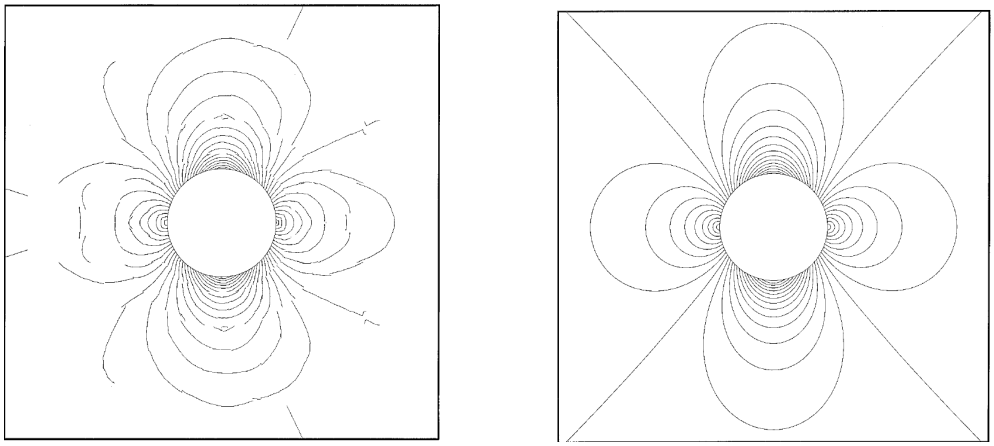
When solving the Euler equations for flow around a symmetric body, also a symmetric velocity profile, hence Mach isolines are expected. This is clearly not the case for the isoparametric linear interpolation. The usual remedy of clustering grid points around regions of interest in order to resolve physics more accurately does not help in this case. Even on the fine mesh a non-physical boundary layer and wake is present. The computations do not converge to a steady state solution due to the unsteadiness of the wake. Spurious entropy production near the solid walls are present which is unphysical as the Euler equations conserve entropy along streamlines of smooth flow (i.e. no shock) [7]. In the next case superparametric P_1Q_2 elements are used. The name superparametric since $m > k$. Figure 3 shows the corresponding Mach isolines.



(a) Coarse Mesh. (b) Fine Mesh.

Figure 3: Mach isolines P_1Q_2. [1]

A much more symmetric solution can be observed, even for the coarse mesh. The fine mesh seems to be perfectly symmetric. Going one step further to isoparametric P_2Q_2 elements, see Figure 4, the results get even better.



(a) Coarse Mesh. (b) Fine Mesh.

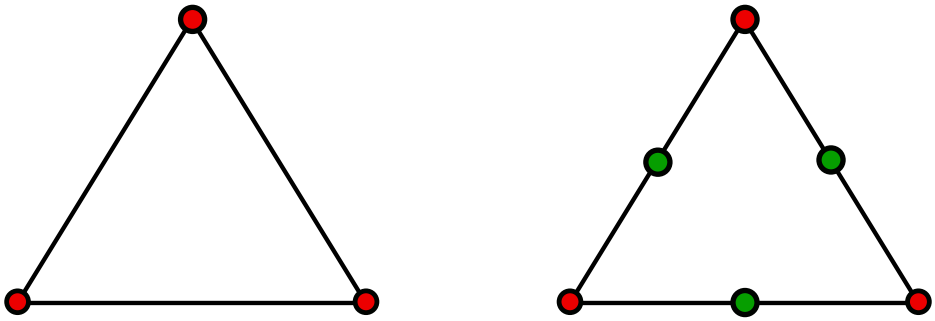
Figure 4: Mach isolines P_2Q_2. [1]

As a conclusion from this model problem one can see that a geometric representation

that takes into account at least the curvature (quadratic interpolation) of the boundary is mandatory to obtain meaningful numerical solutions. Therefore, high-order computations are only reasonable if high-order geometrical representations are available.

3 Generation of Curved Meshes with a Linear Input Mesh

The generation of high-order meshes in this paper require a linear mesh as a starting point. As seen in the previous section, elements of second order are already satisfactory. Hence, this paper is limited to the generation of second order meshes. Having a linear mesh at hand, one realizes that additional degrees of freedom are required to represent the elements. Hence, additional points are added to the existing element, see Figure 5.



(a) Linear element. (b) Quadratic element.

Figure 5: Degrees of freedom for linear and quadratic elements (green nodes are additional).

A possible procedure for generating a curved mesh is to take the geometric information (e.g. from a CAD file) and move the new nodes (green nodes in Figure 5) at boundary elements to the location corresponding to the geometry. Figure 6 shows such an example.

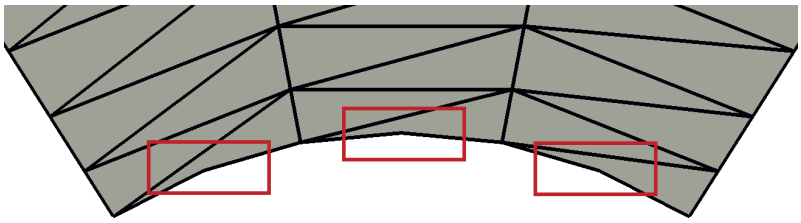
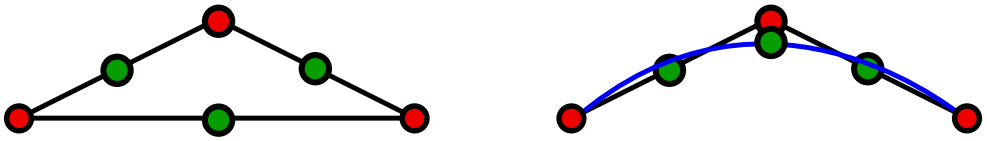


Figure 6: Adapted boundary elements.

This procedure seems to be uncomplicated and to require only small computational effort. The big problem, however is that a certain element quality is required, otherwise one obtains self-intersecting elements as in Figure 7.



(a) Quadratic element. (b) Quadratic element - distorted.

Figure 7: Distorted quadratic element (blue line intersecting).

Such elements lead to invalid triangulations and negative volumes and cannot be used for computations, hence it has to be avoided to generate those elements. Unfortunately, anisotropic elements as in Figure 7 are quite frequent. Especially in the case of turbulence modeling a high grid resolution near walls is required to resolve the viscous sublayer [2]. In order to keep the overall element count low, the elements are only refined normal to the wall, resulting in high aspect ratios. Hence, a solution is sought where also the inner nodes are moved such that no intersections are present.

A possible solution strategy is motivated by [6], where an elasticity analogy is applied. There, the geometry of the domain to be meshed is represented as an elastic solid. The undeformed configuration is the initial, linear mesh which is provided as an input. The external loading results from prescribing a boundary displacement such that it coincides with the curved geometry. Hence, only Dirichlet boundary conditions are present, all determined from the geometry. Finally, an equilibrium solution is sought which then provides the displacement for the individual nodes. The method presented in [6] uses a nonlinear elasticity analogy, where in this paper only a linear approach is presented. The following section provides details about the mathematical background of linear elasticity and its application to grid deformation within the finite element context.

4 Mathematical Background

Since one is interested in the final deformation it is sufficient to consider the problem of elastostatics only. The equations are given as [3]

$$-\nabla \cdot \boldsymbol{\sigma} = \mathbf{f} \tag{1}$$

where $\boldsymbol{\sigma}$ is the Cauchy stress tensor and $\mathbf{f}$ is the body force.

In linear elasticity the linear Saint Venant-Kirchhoff material law is used together with geometrical linearity yielding Hooke's law [12]

$$\boldsymbol{\sigma} = \lambda \operatorname{tr}(\boldsymbol{\epsilon})\mathbf{I} + 2\mu\boldsymbol{\epsilon} \tag{2}$$

with Lamé parameters λ and μ and the linear strain tensor

$$\boldsymbol{\epsilon} = \frac{1}{2}\left(\nabla\mathbf{u} + \nabla\mathbf{u}^T\right) \tag{3}$$

with displacement $\mathbf{u}$. Using Voigt notation, $\boldsymbol{\sigma}$ and $\boldsymbol{\epsilon}$ can be rewritten as

$$\boldsymbol{\sigma}_V = \begin{bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{33} \\ \sigma_{23} \\ \sigma_{13} \\ \sigma_{12} \end{bmatrix}, \quad \boldsymbol{\epsilon}_V = \begin{bmatrix} \epsilon_{11} \\ \epsilon_{22} \\ \epsilon_{33} \\ 2\epsilon_{23} \\ 2\epsilon_{13} \\ 2\epsilon_{12} \end{bmatrix} \tag{4}$$

Due to symmetry of $\boldsymbol{\sigma}$ and $\boldsymbol{\epsilon}$ this yields identical inner products

$$\boldsymbol{\sigma} : \boldsymbol{\epsilon} = \boldsymbol{\sigma}_V \cdot \boldsymbol{\epsilon}_V \tag{5}$$

Rewriting equation (2) into Voigt notation yields

$$\begin{bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{33} \\ \sigma_{23} \\ \sigma_{13} \\ \sigma_{12} \end{bmatrix} = \begin{bmatrix} \lambda+2\mu & \lambda & \lambda & 0 & 0 & 0 \\ \lambda & \lambda+2\mu & \lambda & 0 & 0 & 0 \\ \lambda & \lambda & \lambda+2\mu & 0 & 0 & 0 \\ 0 & 0 & 0 & \mu & 0 & 0 \\ 0 & 0 & 0 & 0 & \mu & 0 \\ 0 & 0 & 0 & 0 & 0 & \mu \end{bmatrix} \begin{bmatrix} \epsilon_{11} \\ \epsilon_{22} \\ \epsilon_{33} \\ 2\epsilon_{23} \\ 2\epsilon_{13} \\ 2\epsilon_{12} \end{bmatrix} \tag{6}$$

Replacing the Lamé parameters with [12]

$$\lambda = \frac{E\nu}{(1+\nu)(1-2\nu)} \tag{7}$$

$$\mu = \frac{E}{2(1+\nu)} \tag{8}$$

in equation (6) results in

$$\boldsymbol{\sigma}_V = \mathbf{D}\, \boldsymbol{\epsilon}_V \tag{9}$$

with

$$\mathbf{D} = \frac{E}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1-\nu & \nu & \nu & 0 & 0 & 0 \\ \nu & 1-\nu & \nu & 0 & 0 & 0 \\ \nu & \nu & 1-\nu & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1-2\nu}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1-2\nu}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1-2\nu}{2} \end{bmatrix} \tag{10}$$

$\boldsymbol{\epsilon}_V$ can be rewritten in terms of $\mathbf{u}$ as

$$\boldsymbol{\epsilon}_V = \begin{bmatrix} \partial_x u_1 \\ \partial_y u_2 \\ \partial_z u_3 \\ \partial_z u_2 + \partial_y u_3 \\ \partial_z u_1 + \partial_x u_3 \\ \partial_y u_1 + \partial_x u_2 \end{bmatrix} = \underbrace{\begin{bmatrix} \partial_x & 0 & 0 \\ 0 & \partial_y & 0 \\ 0 & 0 & \partial_z \\ 0 & \partial_z & \partial_y \\ \partial_z & 0 & \partial_x \\ \partial_y & \partial_x & 0 \end{bmatrix}}_{=:\mathcal{B}} \mathbf{u} \tag{11}$$

with the differential operator $\mathcal{B}$. A weak formulation for equation (2) is obtained by multiplying with a test function $\mathbf{v} \in \mathcal{V} := \mathcal{H}^1_{\Gamma_D}(\Omega) = \{\mathbf{v} \in \mathcal{H}^1(\Omega) \mid \mathbf{v} = \mathbf{0} \text{ on } \Gamma_D\}$ with $\mathcal{H}$ being a Hilbert space [4]

$$-\int_\Omega \mathbf{v} \cdot (\boldsymbol{\nabla} \cdot \boldsymbol{\sigma}) \, \mathrm{d}\Omega = \int_\Omega \mathbf{v} \cdot \mathbf{f} \, \mathrm{d}\Omega \tag{12}$$

using the chain rule

$$\boldsymbol{\nabla} \cdot (\boldsymbol{\sigma} \cdot \mathbf{v}) = \mathbf{v} \cdot (\boldsymbol{\nabla} \cdot \boldsymbol{\sigma}) + \boldsymbol{\nabla}\mathbf{v} : \boldsymbol{\sigma} \tag{13}$$

and using Gauss's theorem equation (12) can be rewritten as

$$\int_\Omega \boldsymbol{\nabla}\mathbf{v} : \boldsymbol{\sigma} \, \mathrm{d}\Omega = \int_\Omega \mathbf{v} \cdot \mathbf{f} \, \mathrm{d}\Omega + \int_{\Gamma_N} \mathbf{v} \cdot \mathbf{t}_n \, \mathrm{d}\Gamma \tag{14}$$

where Γ_N is the Neumann part of the boundary on which $\mathbf{v}$ does not vanish. $\mathbf{t}_n$ is the boundary traction defined through

$$\boldsymbol{\sigma} \cdot \mathbf{n} = \mathbf{t}_n \qquad \text{on } \Gamma_N \tag{15}$$

Due to symmetry of $\boldsymbol{\sigma}$ the inner product (double contraction) $\boldsymbol{\nabla}\mathbf{v} : \boldsymbol{\sigma}$ can be rewritten in Voigt notation as

$$\boldsymbol{\nabla}\mathbf{v} : \boldsymbol{\sigma} = \mathcal{B}\mathbf{v} \cdot \boldsymbol{\sigma}_V = (\mathcal{B}\mathbf{v})^T \boldsymbol{\sigma}_V \tag{16}$$

Together with equation (9) and equation (11) one obtains the following relation for equation (14)

$$\int_\Omega (\mathcal{B}\mathbf{v})^T \mathbf{D} \mathcal{B}\mathbf{u} \, \mathrm{d}\Omega = \int_\Omega \mathbf{v}^T \mathbf{f} \, \mathrm{d}\Omega + \int_{\Gamma_N} \mathbf{v}^T \mathbf{t}_n \, \mathrm{d}\Gamma \tag{17}$$

Equation (17) is the basis for our finite element discretization. We define the bilinear form

$$\mathbf{a}(\mathbf{u}, \mathbf{v}) := \int_\Omega (\mathcal{B}\mathbf{v})^T \mathbf{D} \mathcal{B}\mathbf{u} \, \mathrm{d}\Omega \tag{18}$$

and the linear form as

$$\boldsymbol{\ell}(\mathbf{v}) := \int_\Omega \mathbf{v}^T \mathbf{f} \, \mathrm{d}\Omega + \int_{\Gamma_N} \mathbf{v}^T \mathbf{t}_n \, \mathrm{d}\Gamma \tag{19}$$

When changing into the discrete setting, the functions $\mathbf{u}$ and $\mathbf{v}$ become vectors with entries corresponding to the nodes of the mesh $\mathbf{u}^h$ and $\mathbf{v}^h$ and together with the shape function matrix $\mathbf{N}$ they can be expressed as:

$$\mathbf{u} \approx \mathbf{N}\mathbf{u}^h \tag{20}$$

$$\mathbf{v} \approx \mathbf{N}\mathbf{v}^h \tag{21}$$

The differential operator $\mathcal{B}$ has to be expressed by the derivatives of the shape functions and thus becomes a matrix

$$\mathcal{B}\mathbf{v} \approx \mathbf{B}\mathbf{v}^h = \begin{bmatrix} \mathbf{B}_1, & \ldots, & \mathbf{B}_{\text{nshp}} \end{bmatrix} \mathbf{v}^h \tag{22}$$

with

$$\mathbf{B}_A = \begin{bmatrix} \frac{\partial N_A}{\partial x} & 0 & 0 \\ 0 & \frac{\partial N_A}{\partial y} & 0 \\ 0 & 0 & \frac{\partial N_A}{\partial z} \\ 0 & \frac{\partial N_A}{\partial z} & \frac{\partial N_A}{\partial y} \\ \frac{\partial N_A}{\partial z} & 0 & \frac{\partial N_A}{\partial x} \\ \frac{\partial N_A}{\partial y} & \frac{\partial N_A}{\partial x} & 0 \end{bmatrix} \tag{23}$$

where `nshp` denotes the number of shape functions.
Since $\mathbf{u}^h$ and $\mathbf{v}^h$ are vectors of constant coefficients, they are independent of the integration domain and can thus be pulled out of the integral. Thus equation (17) in its discretized form can be written as:

$$\mathbf{v}^{hT} \left(\underbrace{\int_\Omega \mathbf{B}^T \mathbf{D} \mathbf{B} \, \mathrm{d}\Omega}_{\mathbf{K}} \, \mathbf{u}^h = \underbrace{\int_\Omega \mathbf{N}^T \mathbf{f} \, \mathrm{d}\Omega + \int_{\Gamma_N} \mathbf{N}^T \mathbf{t}_n \, \mathrm{d}\Gamma}_{\tilde{\mathbf{f}}} \right) \tag{24}$$

Hence it breaks down to solve the linear system

$$\mathbf{K}\mathbf{u}^h = \tilde{\mathbf{f}} \tag{25}$$

$\mathbf{K}$ is obtained by so called BDB-integrators, the name giving is clear through the definition. Setting up $\mathbf{K}$ in a finite element context, one can make use of the additibility of integrals by splitting Ω into subdomains (elements) $\sum_e \Omega_e$ and the transformation to the reference domain Ξ. Then, $\mathbf{K}$ takes the following form

$$\mathbf{K} = \int_\Omega \mathbf{B}^T \mathbf{D} \mathbf{B} \, \mathrm{d}\Omega = \sum_e \int_{\Omega_e} \mathbf{B}^T \mathbf{D} \mathbf{B} \, \mathrm{d}\Omega = \sum_e \int_\Xi \mathbf{B}_{\text{ref}}^T \mathbf{D} \mathbf{B}_{\text{ref}} \, J^e \mathrm{d}\Xi \tag{26}$$

$$= \mathop{\mathsf{A}}_e \underbrace{\int_\Xi \mathbf{B}_{\text{el}}^T \mathbf{D} \mathbf{B}_{\text{el}} \, J^e \mathrm{d}\Xi}_{\mathbf{K}_e} \tag{27}$$

where J^e is the jacobian determinant of the transformation from the reference domain to the physical domain

$$J^e = \det\left(\frac{\partial \mathbf{x}_e}{\partial \boldsymbol{\xi}}\right) \tag{28}$$

$\mathbf{A}$ is the assembly operator, $\mathbf{K}_e$ is the elemental stiffness matrix and the elemental $\mathbf{B}$-matrix $\mathbf{B}_{\text{el}}$ depends only on the reference element and defined defined as

$$\mathbf{B}_{\text{el}} = \left[\mathbf{B}_{\text{el},1}, \quad \ldots, \quad \mathbf{B}_{\text{el,nelshp}}\right] \tag{29}$$

with

$$\mathbf{B}_{\text{el,A}} = \begin{bmatrix} \frac{\partial N_A}{\partial \xi} & 0 & 0 \\ 0 & \frac{\partial N_A}{\partial \eta} & 0 \\ 0 & 0 & \frac{\partial N_A}{\partial \zeta} \\ 0 & \frac{\partial N_A}{\partial \zeta} & \frac{\partial N_A}{\partial \eta} \\ \frac{\partial N_A}{\partial \zeta} & 0 & \frac{\partial N_A}{\partial \xi} \\ \frac{\partial N_A}{\partial \eta} & \frac{\partial N_A}{\partial \xi} & 0 \end{bmatrix} \tag{30}$$

`nelshp` is the number of shape functions per element.

4.1 Adaption for Grid Deformation

As described earlier the boundary conditions are of Dirichlet type only, hence the equation

$$\mathbf{a}(\mathbf{u}^h, \mathbf{v}^h) = \boldsymbol{\ell}(\mathbf{v}^h) \tag{31}$$

has a zero right-hand side $\boldsymbol{\ell}(\mathbf{v}^h) = \mathbf{0}$. Decomposing the solution vector $\mathbf{u}^h$ into

$$\mathbf{u}^h = \mathbf{u}_0^h + \mathbf{u}_D^h \tag{32}$$

with $\mathbf{u}_D^h$ being the known values on Γ_D and the inner nodes $\mathbf{u}_0^h \in \mathcal{V}^h \subset \mathcal{V}$ being from the same space as $\mathbf{v}^h$. This results in a new formulation

$$\mathbf{a}(\mathbf{u}_0^h, \mathbf{v}^h) = -\mathbf{a}(\mathbf{u}_D^h, \mathbf{v}^h) =: \tilde{\boldsymbol{\ell}}(\mathbf{v}^h) \tag{33}$$

To have more control over the mesh deformation and obtain meshes of better quality not the linear elasticity equations are solved but the modification according to [10] is applied. The mesh deformation in [10] is intended for fluid-structure interaction problems but it

can also be applied for our problem setting. This modification affects the stiffness matrix as follows:

$$\mathbf{K}_e = \int_\Xi \mathbf{B}_{\text{el}}^T \mathbf{D} \mathbf{B}_{\text{el}} \, J^e \mathrm{d}\Xi \rightsquigarrow \int_\Xi \mathbf{B}_{\text{el}}^T \mathbf{D} \mathbf{B}_{\text{el}} \, J^e \left(\frac{J^0}{J^e} \right)^\chi \mathrm{d}\Xi \tag{34}$$

where J^0 is an arbitrary scaling parameter and χ is the so-called stiffening power. The additional scalar factor $(J^0/J^e)^\chi$ can be regarded as being applied to the $\mathbf{D}$ matrix and thus acting on the elasticity modulus E. Hence, one obtains a modified elasticity modulus $\tilde{E} = (J^0/J^e)^\chi E$ which is different for each element. Depending on the value of χ one can interpret $\tilde{E}$ to stiffen or soften the element depending on its jacobian or its size, the deviation from the reference element to be more precise.

5 Implementation into NGSolve

NGSolve is a general purpose finite element library built on top of the mesh generator Netgen [8]. NGSolve already provided integrators for linear elasticity (BDB-integrators) which had to be adapted to assemble the modified form as shown in the previous section. The following flow chart illustrates the major changes in NGSolve.

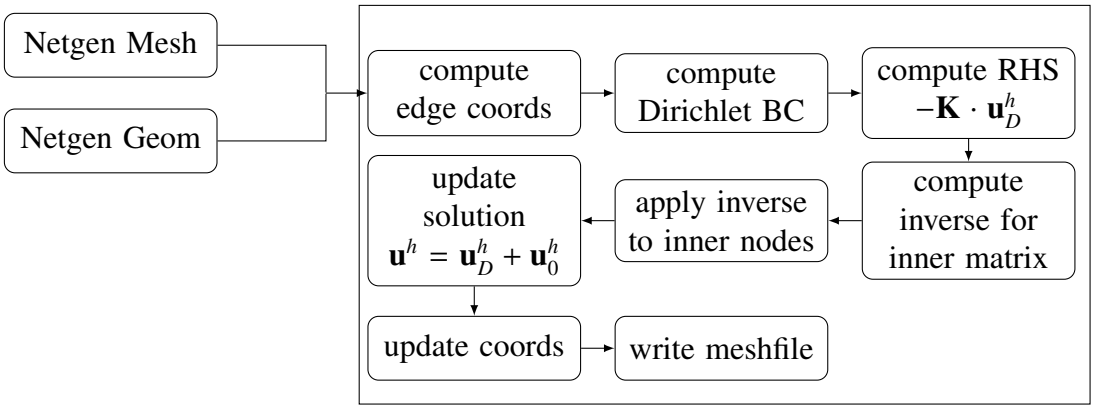


Figure 8: Workflow of numproc `GridDeform` in NGSolve.

A new so called numproc (numerical procedure) had to be created to handle the grid deformation, hence the name `GridDeform`. In addition to the linear mesh (given in the Netgen file format) a geometry (also in the Netgen file format) has to be provided as an input. Given that, the coordinates of the new degrees of freedom (green nodes in Figure 5) are computed. Then, based on the geometry file, the corresponding Dirichlet boundary conditions (displacements) are computed. This provides the vector $\mathbf{u}_D^h$, which is then multiplied with the assembled (quadratic) modified stiffness matrix to form the right-hand side of the equation. NGSolve did not provide a routine for imposing non-homogeneous Dirichlet boundary conditions strongly, hence all the described steps had to be implemented. Afterwards, a direct solver for the inner nodes ($\mathbf{u}_0^h$) is set up and applied to the right-hand side. Then, the displacement vector (solution from the equation) is updated by the Dirichlet nodes and the displacements are added to the coordinate vector and the new mesh file can be written.

The situation gets somewhat more complicated, if the mesh file is not available in the Netgen format. This is due to the fact that geometrical information is included in the mesh file, namely the value of the spline parameter of the boundary edges. This parameter ranges from 0, the starting point of the corresponding spline, to 1, the end point of the spline. In case this information is missing, another numproc `ReconstructSpline2D` has been implemented to overcome the problem.

If also the geometry is not present (at all or in the Netgen format), a tool `genSpline` has been developed which reconstructs the geometry from a given input mesh together with an analytical function and its derivative describing the boundary of the domain. Having these tools at hand, the restriction to the Netgen file format is not as severe as it might sound at the beginning.

6 Results

In this section the deformation of grids based on the geometry shown in Figure 9 is presented.

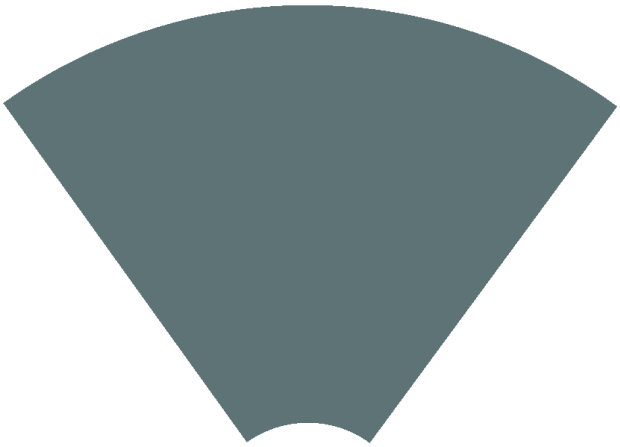
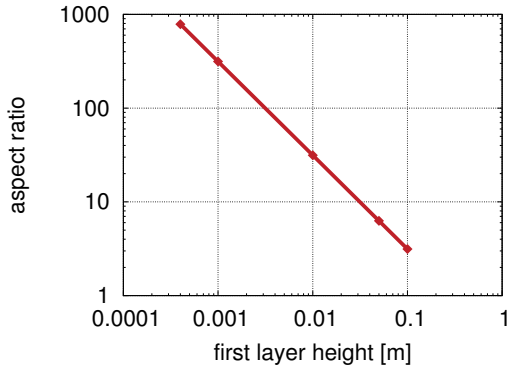


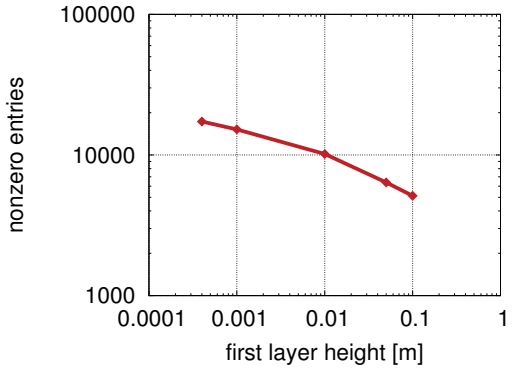
Figure 9: Sample Geometry.

The circle segment has an inner radius of 1 and an outer radius of 5 with an angle of 0.4π. The number of elements in circumferential direction is kept constant to be 4. The number of elements in radial direction is determined by the growth rate and first layer height, where the growth rate is kept constant at 1.2. Hence, the first layer height remains the only variable parameter. First layer heights up to $4 \cdot 10^{-4}$ with a corresponding aspect ratio of around 800 could be computed, as can be observed in Figure 10a. To obtain these results, the two decisive parameters were Poisson's ratio $\nu = 0.4$ and the stiffening power $\chi = -0.4$. The choice of those two parameters gave the maximum aspect ratios. The choice of the elasticity modulus E and scaling constant J^0 do not affect the result since they are a constant factor in front of the global matrix and thus can be canceled out as the right-hand side depends on the matrix only. Figure 10b shows the non-zero entries of the stiffness matrix $\mathbf{K}$, but only the lower triangular matrix, since it is symmetric. Figure 10c shows an estimation of the condition number based on the ∞-norm. For this not only

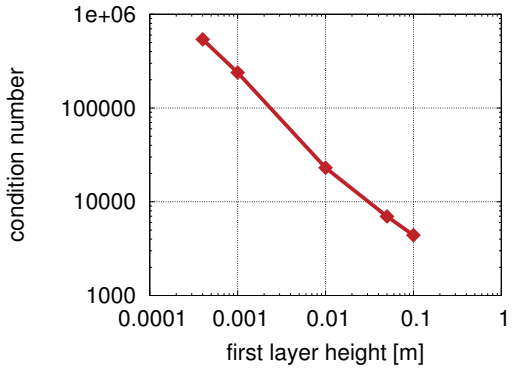
the lower triangular part of $\mathbf{K}$ is taken, but the full symmetric inner matrix. With inner matrix the part of the matrix corresponding to the inner nodes $\mathbf{u}_0^h$ is meant, since the entire matrix is singular due to the Dirichlet boundary condition and would result in a condition number of infinity. It can be observed that the condition number grows as the element size shrinks. The system gets more complicated to solve.



(a) Aspect ratios.



(b) Nonzero entries of $\mathbf{K}$ (lower triangular).



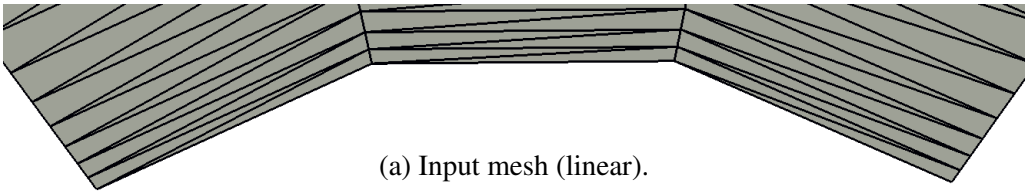
(c) Condition numbers of $\mathbf{K}$.

Figure 10: Properties of different meshes.

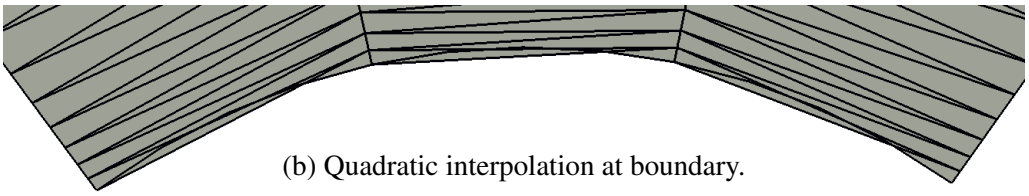
Figure 11 shows the transformation from the input mesh (a) to the curved mesh (c) obtained by solving the modified elasticity equations. It can be observed that the input mesh gives a very bad geometry representation and the circle segment can be barely recognized. But also the necessity of solving the entire domain is demonstrated, since modifying only the boundary nodes (b) leads to an invalid mesh.

Figure 12 shows an application case from Nasa[1] for turbulence modeling. The grids are provided (linear meshes) by the website and can be used to study turbulence models since validated solutions are also provided for comparison. The shown grid (cut-out) is of structured type and has 177x81 nodes with a domain length of 50 and height of 5. The first layer height is around $4 \cdot 10^{-6}$ and this results to an aspect ratio of around 7500 at the peak of the bump. Compared to the previous example the aspect ratio is increased by a factor of 10. This is due to the fact that the curvature was much higher for the circle segment and hence the the mesh became invalid more easily. For this application case, the tools described earlier had to be used in order to reconstruct the geometry and add the geometrical information to the mesh file.

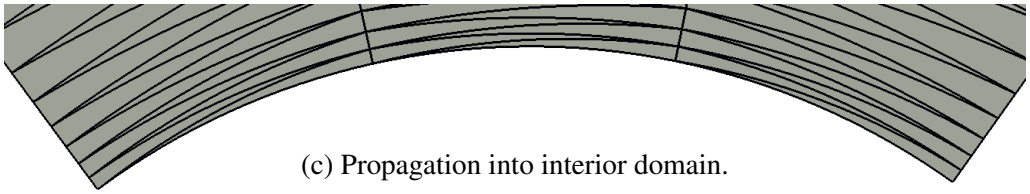
[1] http://turbmodels.larc.nasa.gov/bump_grids.html



(a) Input mesh (linear).



(b) Quadratic interpolation at boundary.



(c) Propagation into interior domain.

Figure 11: Grid deformations.

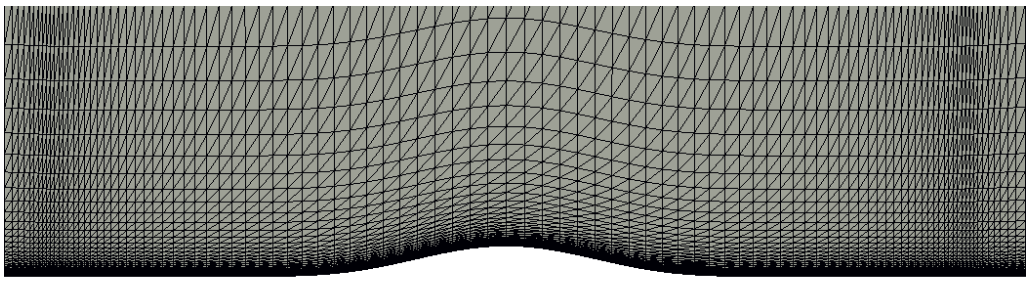


Figure 12: Bump 2D.

7 Summary and Outlook

This paper presented a method of high-order meshes needed for high-order schemes for computational fluid dynamics. Meshes of second order can be generated if a linear mesh and a corresponding geometry are provided as an input. The propagation of the deformation from the boundaries into the interior of domain is handled with the elasticity analogy. More precisely, a modified linear elasticity model is used which allows control of the element deformation based on its jacobian. Results were presented which showed the successful implementation of the model into the general purpose finite element library NGSolve. The applicability of the method to general meshes was shown.

Future work should include an extension to 3D. The assembly procedure would not change but the treatment of geometries has to be modified to account for three-dimensional models. Including load-stepping, which means to apply the boundary conditions stepwise instead of fully at once, would probably allow for handling of meshes with higher aspect ratios. This is due to the fact that the stiffness matrix would be gradually updated after

each loading step. Of coarse this solution procedure would be more costly. Also, one could think of replacing the linear elasticity model with a nonlinear one, for example a Neo-Hookean model as proposed in [6]. The method is expected to be more robust than the shown linear model. One could also step away from the elasticity analogy and use radial basis functions (RBFs) to propagate the deformations into the interior of the domain as proposed in [5]. That approach has the advantage of lower computational costs and the independency of mesh connectivities, therefore structured as well as unstructured and hybrid meshes would be treated equally. Unfortunately, the success of the method depends on the choice of the right basis function and its parameters.

References

[1] F. Bassi and S. Rebay. High-Order Accurate Discontinuous Finite Element Solution of the 2D Euler Equations. *Journal of computational physics*, 138(2):251–285, 1997.

[2] J. Blazek. *Computational Fluid Dynamics: Principles and Applications: (Book with accompanying CD)*. Elsevier Science, 2005.

[3] P.G. Ciarlet. *Three-Dimensional Elasticity*. Number Bd. 1 in Studies in mathematics and its applications. Elsevier Science, 1988.

[4] J. Donea and A. Huerta. *Finite Element Methods for Flow Problems*. Finite Element Methods for Flow Problems. John Wiley & Sons, 2003.

[5] S. Jakobsson and O. Amoignon. Mesh Deformation using Radial Basis Functions for Gradient-Based Aerodynamic Shape Optimization. *Computers & Fluids*, 36(6):1119–1136, 2007.

[6] P.-O. Persson and J. Peraire. Curved Mesh Generation and Mesh Refinement using Lagrangian Solid Mechanics. In *Proceedings of the 47th AIAA Aerospace Sciences Meeting and Exhibit*, 2009.

[7] A. Rizzi and L.-E. Eriksson. Computation of Flow around Wings based on the Euler Equations. *Journal of Fluid Mechanics*, 148:45–71, 1984.

[8] J. Schöberl. Netgen an advancing front 2d/3d-mesh generator based on abstract rules. *Computing and Visualization in Science*, 1(1):41–52, 1997.

[9] C. Shu. High-Order Finite Difference and Finite Volume WENO Schemes and Discontinuous Galerkin Methods for CFD. *International Journal of Computational Fluid Dynamics*, 17(2):107–118, 2003.

[10] K. Stein, T. Tezduyar, and R. Benney. Mesh Moving Techniques for Fluid-Structure Interactions with Large Displacements. *Journal of Applied Mechanics*, 70(1):58–63, 2003.

[11] Z. Wang, K. Fidkowski, R. Abgrall, F. Bassi, D. Caraeni, A. Cary, H. Deconinck, R. Hartmann, K. Hillewaert, H. Huynh, et al. High-Order CFD Methods: Current Status and Perspective. *International Journal for Numerical Methods in Fluids*, 72(8):811–845, 2013.

[12] P. Wriggers. *Nichtlineare Finite-Element-Methoden*. Springer Berlin Heidelberg, 2001.