Technische Universiteit Delft
Faculteit Elektrotechniek, Wiskunde en Informatica
Delft Institute of Applied Mathematics

An elastic model for meshfitting for the Level-set Method

(Nederlandse titel: Een elastisch model voor grid-aanpassing voor de Level-set Methode)

Verslag ten behoeve van het
Delft Institute of Applied Mathematics
als onderdeel ter verkrijging

van de graad van

BACHELOR OF SCIENCE
in
TECHNISCHE WISKUNDE

door

WILLEM MIJNDERT DEN HERTOG

Delft, Nederland
Juni 2019

TU Delft

BSc verslag TECHNISCHE WISKUNDE

"An elastic model for meshfitting for the Level-set Method"

(Nederlandse titel: "Een elastisch model voor grid-aanpassing voor de Level-set Methode")

WILLEM MIJNDERT DEN HERTOG

Technische Universiteit Delft

Abstract: The numerical errors involving the application of the level-set method to a problem can be minimized by fitting a mesh (usually a triangular mesh in two dimensions) to the zero level-set curve defined by the level-set function. These numerical errors depend on two things: the size and skewness of the triangles in the mesh. To fit the mesh, Timo Wortelboer derived a physical model using springs [14]. He also defined measures to quantify the quality of a fitted mesh. First, his model is analyzed, as well as the quality measures he defined. Then an introduction on linear isotropic elasticity is given, based on which an elastic model for meshfitting is derived. The Finite Element Method is used to solve the differential equations of the elastic model. Then an optimisation algorithm is introduced to change the Lamé parameters such that the quality of the mesh is improved even more. Lastly the spring model and the elastic model are compared, which results in the elastic model being better than the spring model.

Begeleider

Dr.ir. D. den Ouden-van der Horst

Overige commissieleden

Dr. J.L.A. Dubbeldam

Dr. B. van den Dries

Juni, 2019

Delft

Preface

This thesis is the culmination of my Bachelor of Science in Applied Mathematics research project at the Delft University of Technology, at the faculty of Electrical Engineering, Mathematics and Computer Science.
I want to express my gratitude towards my supervisor, Dennis den Ouden - van der Horst, for his exceptional guidance.

Contents

Chapter 1

Introduction

In (mathematical) physics, a lot of problems involve materials or substances that can change from one phase to another. Examples of this are the melting of ice [15], or the modelling of different phases of steel [5]. When modelling these problems, the level-set method [8] can be used. This method (for two-phase problems) defines one phase of the problem as the 'negative' part, and the other phase as the 'positive' part, and in that way it defines the boundary between those phases implicitly as the zero level-set curve.
To solve a two-phase problem, we often use the Finite Element Method. This method needs, as all numerical methods, a discretisation of the area we are working in. In this thesis we consider the Delaunay triangulation of the area $(x, y) = [-1, 1] \times [-1, 1]$, a square with center $(x, y) = (0, 0)$ and sides of length 2. When we consider a two-phase problem in this area, it defines a zero level-set curve inside this area. To maximise the accuracy of the Finite Element Method, we want the triangulation of the area to fulfill three requirements:

1. it needs to fit the zero level-set curve (the boundary between the two phases),
2. the triangles of the mesh should be as close to equilateral triangles as possible, and
3. the triangles of the mesh should be as equal in size as possible.[1]

Since the standard Delaunay triangulation generally does not fit a random zero level-set curve, we need to adjust the mesh in such a way that it fits that curve, but also that it still fulfills the second and third requirement. This is the subject of this thesis.

Previous research on this subject has been done by Etelvina Javierre Pérez [2], Dennis den Ouden - van der Horst [6], Thijs Verbeek [11] and Timo Wortelboer [14]. The focus of this thesis is deriving an elastic model for fitting a mesh to a zero level-set curve (this is called 'meshfitting').

Chapter 2 introduces the necessary mathematical background information needed for this research, including the mesh quality measures introduced in [14]. In Chapter 3 we derive a numerical model for linear isotropic elasticity, and the results of this model are treated in Chapter 4. That chapter also contains the comparison of the elastic model with the spring model derived in [14]. In the last chapter we introduce some ideas for future research.

[1] Although this constraint can be replaced by the demand that changes in size should be gradual, thereby allowing locally refined meshes.

Chapter 2

Mathematical background

This chapter provides the mathematical background needed to understand the potential applications of this research. This is done in the first two sections. The third section describes the terminology we will use, and the last two sections are a summary of the work Timo Wortelboer did [14]. This thesis is a continuation of (parts of) his thesis, hence it is necessary to give this summary.

2.1 The finite-element method

The finite-element method (FEM) is a numerical integration method to solve (systems of) (partial) differential equations [10]. One of the advantages of the finite-element method is that it does not need a simple domain of integration (for example: a rectangle or a sphere), it also works for complicated domains. This is the reason FEM is used a lot in for example crash simulations or weather predictions [9].
For this research we consider a regular triangular mesh and we will use P1 FEM. This means that the basis functions are piecewise linear functions that equal one on one vertex and zero on the other vertices in the mesh. The solution of the differential equations will be a linear combination of these basis functions. Important to note is that the quality of the solution is dependent on the regularity of the mesh. This is the reason we will introduce measures to quantify the regularity of the mesh (in Section 2.4).

2.2 The level-set method

The level-set method [8] is a method used in problems with multiple phases. An example of such a problem is the investigation of heat flow through a plate consisting of two different areas (phases): the first phase is the iron region, and the second phase is the copper region (see Figure 2.1).

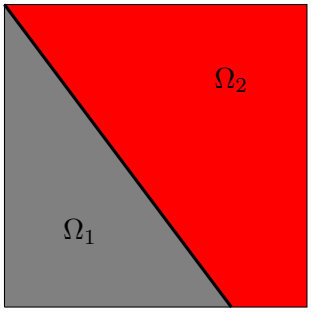

Figure 2.1: A plate consisting of iron (Ω_1) and copper (Ω_2).

To make it easier to talk about these problems, the level-set function $\phi(\mathbf{x})$, $\mathbf{x} \in \mathbb{R}^3$ has been introduced. In this example we say that the iron phase is the region $\Omega_1 = \{\mathbf{x} : \phi(\mathbf{x}) < 0\}$ and the copper phase is the region $\Omega_2 = \{\mathbf{x} : \phi(\mathbf{x}) > 0\}$. If we want to investigate the heat flow through this plate using FEM, we have to divide the area in triangles. With this division in triangles, it is preferred that no triangles consist of both metals. In other words: we want the interface between the two phases to be equal to a set of edges of the triangles. In that way, no triangle will consist of two different metals. Since we called the iron region the 'negative' region, and the copper region the 'positive' region, we will call the interface between the two phases the zero level-set curve $\Gamma_0 = \{\mathbf{x} : \phi(\mathbf{x}) = 0\}$ (the thick line between the grey and red area in Figure 2.1).

2.3 Terminology

To make it easier to talk about the mesh and all of its properties, we will introduce some terminology. We will use the same terminology as in [14]. As we can see in Figure 2.2, the mesh consists of triangles. We call the set of triangles T. A triangle consists of three edges and three vertices (points). We call the set of edges E, and the set of vertices P.

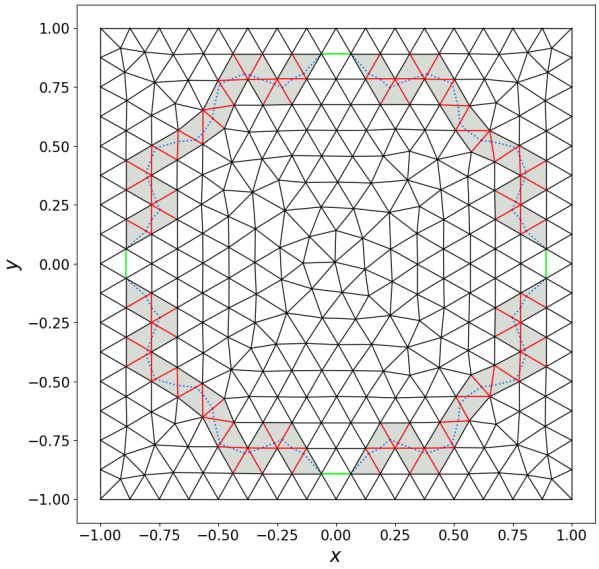

Figure 2.2: A mesh before meshfitting is applied. Retrieved from [14].

There are three different types of edges:

1. Boundary edges, these are the edges on the boundary of the mesh.

2. Level-set edges, these are the edges of the piecewise-linear approximation of the zero level-set curve (also called: the level-set function). In Figure 2.2 these are depicted in blue. Note: these edges are not necessarily edges of the triangles. As you can see in the picture, they are usually not edges of the triangles. If the level-set edge and the edge of a triangle are the same, it will be coloured green.

3. Zero edges, the red lines in Figure 2.2. These are the edges that are crossed by the level-set function.

We will consider four different types of vertices:

1. Boundary vertices, the vertices contained in the boundary of the mesh.

2. Corner vertices, the vertices in the corner of the mesh.

3. Zero vertices, the vertices that are crossed by the level-set function.

4. Interpolated zero level-set vertices, the vertices that are on the intersections of the level-set function and the edges of the mesh.

There is only one special kind of triangle: the triangles that have a zero edge as one of its edges. Those triangles are called: zero elements.

2.4 Mesh quality [14]

To analyse the quality of the mesh, we need to compare the properties of the fitted mesh with a 'perfect mesh'. A perfect mesh in this case is a mesh with equilateral triangles which are all the same size. Therefore we test the fitted mesh on two aspects: the skewness and the size.

2.4.1 The skewness of the triangles

We define the following formula for the skewness of a triangle Δ:

$$\text{Skew}(\Delta) = 1 - \frac{3\theta_{\min}}{\pi}, \tag{2.1}$$

where $\theta_{\min}$ is the smallest angle of the triangle. $\text{Skew}(\Delta) = 0$ when Δ is an equilateral triangle, because $\theta_{\min} = \frac{\pi}{3}$. Conversely, a 'triangle' with angles $(\pi, 0, 0)$ in radians has a skewness of 1. Based on this skewness of a triangle, we construct three measures of skewness of a mesh. In these measures, T is the whole set of triangles of the mesh.

1. The maximum skewness of the triangles in the mesh:

$$\text{Maxskew}(T) = \max\left\{\text{Skew}(\Delta) : \Delta \in T\right\}. \tag{2.2}$$

2. The average skewness of the triangles in the mesh:

$$\text{Avgskew}(T) = \frac{1}{\#T} \sum_{\Delta \in T} \text{Skew}(\Delta). \tag{2.3}$$

3. The standard deviation[1] of the skewness in the mesh:

$$\text{Stdskew}(T) = \sqrt{\frac{1}{\#T}\sum_{\Delta\in T} |\text{Skew}(\Delta) - \text{Avgskew}(T)|^2}. \tag{2.4}$$

For all these three measures holds: the lower, the better, because Equation (2.1) gives us that for a 'perfect mesh', the skewness of every triangle is 0. This tells us that in the perfect case, all three measures are 0.

2.4.2 The size of the triangles

The second property of the mesh we want to analyse is the size of the triangles in the mesh. To calculate the size of a triangle Δ, we will use the claim that the size of a parallelogram generated by two edges of Δ is twice the size of Δ. This is made more clear in Figure 2.3.

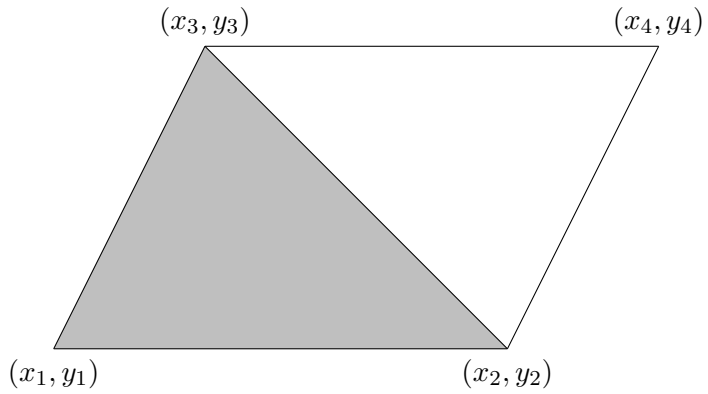

Figure 2.3: Visualisation of the claim that the size of the (grey) triangle Δ is half the size of the parallelogram.

The size of the parallelogram $\parallelogram$ in Figure 2.3 is equal to [4]:

$$\begin{aligned} \text{Size}_{\parallelogram} &= \left|\det \begin{bmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{bmatrix}\right| \\ &= |(x_2 - x_1)(y_3 - y_1) - (x_3 - x_1)(y_2 - y_1)| \end{aligned} \tag{2.5}$$

By definition of a parallelogram, the opposing edges are of the same length, and they are parallel. This means that the grey and white triangles in Figure 2.3 are of the same size. This implies that the size of triangle Δ with vertices $(x_1, y_1), (x_2, y_2), (x_3, y_3)$ is:

$$\begin{aligned} \text{Size}(\Delta) &= \frac{1}{2}\text{Size}_{\parallelogram} \\ &= \frac{1}{2}|(x_2 - x_1)(y_3 - y_1) - (x_3 - x_1)(y_2 - y_1)| \end{aligned} \tag{2.6}$$

We will construct the same kind of measures for the size of the triangles as for the skewness:

[1] This is the exact value of the standard deviation, because all the Skew(Δ) are known. If we consider this as a sample of the triangles, we need to apply a finite population correction to the standard deviation.

1. The maximum size of the triangles in the mesh:

$$\text{Maxsize}(T) = \max\{\text{Size}(\Delta) : \Delta \in T\}. \tag{2.7}$$

2. The minimum size of the triangles in the mesh:

$$\text{Minsize}(T) = \min\{\text{Size}(\Delta) : \Delta \in T\}. \tag{2.8}$$

3. The standard deviation[2] of the size in the mesh:

$$\text{Stdsize}(T) = \sqrt{\frac{1}{\#T}\sum_{\Delta\in T}|\text{Size}(\Delta) - \text{Avgsize}(T)|^2}, \tag{2.9}$$

where $\text{Avgsize}(T)$ is defined as

$$\text{Avgsize}(T) = \frac{1}{\#T}\sum_{\Delta\in T}\text{Size}(\Delta). \tag{2.10}$$

The triangles need to be as equal in size as possible. When all the triangles are of the same size (in the 'perfect mesh'), the Minsize and the Maxsize are equal. Since we want the fitted mesh to be as close to the 'perfect mesh' as possible, the smaller the Maxsize, the better the quality of the fitted mesh, and also: the bigger the Minsize, the better the quality of the mesh. The standard deviation should be as close to 0 as possible.

2.5 Meshfitting [14]

After defining the quality of the mesh, the mesh can be fitted to the zero level-set curve. For that we first move points of the mesh that are close to the zero level-set curve, onto the zero level-set curve. After that, the triangles close to the zero level-set curve can have a really bad shape (the measures of the last section will be really bad). To increase the quality of the mesh, we need to change the shape of a lot of triangles in the mesh in such a way that the total quality improves.

2.5.1 Moving points to the zero level-set curve

To choose which points of the mesh we want to move to the zero level-set curve, we will give every edge e of the triangles in the mesh a weight corresponding to how similar that edge is compared to the zero level-set curve e_{lvset} closest to it. The measure here is the Euclidean norm between the midpoints of two vectors. The midpoint of vector e is denoted with $\mathbf{m}(e)$. So e_{lvset} is the edge of the zero level-set curve such that:

$$||\mathbf{m}(e) - \mathbf{m}(e_{\text{lvset}})||_2 = \min_{e_0\in\{\text{zero level-set curve}\}}\{||\mathbf{m}(e) - \mathbf{m}(e_0)||_2\}. \tag{2.11}$$

We define the weight of an edge e to a level set edge e_{lvset} as follows:

$$w(e) = (1 + \Delta\alpha(e, e_{\text{lvset}})) \cdot (1 + ||\Delta\mathbf{m}(e, e_{\text{lvset}})||_2) - 1. \tag{2.12}$$

Here, $\Delta\alpha(e, e_{\text{lvset}})$ is the angle the edge e makes with the level-set edge e_{lvset}, and $||\Delta\mathbf{m}(e, e_{\text{lvset}})||_2$ is the length of the vector between the midpoints of vector e and vector e_{lvset}. Indeed, we see

[2] This is the exact value of the standard deviation, because all the Size(Δ) are known. If we consider this as a sample of the triangles, we need to apply a finite population correction to the standard deviation.

that if the edge e exactly follows the zero level-set curve (so the angle between those vectors is zero, and the length of the vector between the midpoints of those vectors is zero as well), the weight of the vector is $w(e) = (1 + 0) \cdot (1 + 0) - 1 = 0$.
After giving all edges close to the zero level-set curve a weight, we can choose a start point and an end point, and calculate the shortest path (over those edges) between those points. The choice of the begin- and end point is dependent on the zero level-set curve. Choosing these points is easy when the zero level-set curve crosses the boundary of the mesh, because you can take the places where the zero level-set curve intersects the boundary (or the two neighbours of this point of intersection) as start points and end points. If the zero level-set curve is a closed curve (i.e. it does not cross the boundary of the mesh), we can also define start and end points, but that is a little more complicated. In-depth information on this can be found in [14].
By calculating the shortest path over the mesh edges, we have calculated the set of edges in the mesh which are the closest fit to the zero level-set curve. Now we have to move the vertices of the shortest path to the zero level-set curve. This is done by an orthogonal projection on the zero level-set curve. After this projection the triangles close to the zero level-set curve are deformed. Since those decrease the quality of the mesh, we have to find a way to improve this quality. A simple improvement is moving the projected vertices over the zero level-set curve in such a way that the skewness and size measures of the mesh improves. For a more in-depth explanation of these methods, see [14].

2.5.2 Mesh relaxation

After moving points to the zero level-set curve, there is still a lot of deformation in the triangles close to the zero level-set curve. This problem can be solved by 'relaxing' the mesh. Timo Wortelboer already made such a relaxation by modelling the mesh as a mesh of springs. After forcing vertices on the zero level-set curve (in the way described in Section 2.5.1), the mesh relaxation is found by numerically finding the lowest-energy state of the mesh of springs. This solution is a big improvement to the mesh quality. Since we will not use this spring model in this thesis (we will only use the results found with this model), there is no need to go in depth on this model. Also for this model, more in-depth information can be found in [14].

Chapter 3

Mathematical model

In this chapter we will derive the differential equations for our model, which we later will have to solve numerically. The derivation will be done in the first section. The second section describes the steps necessary to convert these equations into a form that can be numerically solved by a computer. We will use the Finite Element Method (FEM) for this conversion. Finally, the obtaining of the initial conditions and a short overview of the algorithm will be described.

3.1 Derivation of the differential equations

The idea is that we want to model the behaviour of an elastic two-dimensional material (rubber for example) when we apply forces to it. As the title of the section says, we will derive differential equations that describe this behaviour. We get to these equations in two steps: firstly, we will introduce and define the concept elastic deformation in a mathematical way. We use [7] for this. We will use this concept in our second step, the application of the definitions of elastic deformation to our model. This will give us the differential equations.

3.1.1 Elastic deformation

In physics and mathematics, deformation refers to the change of shape a material has after applying a system of forces (loads). This deformation is called elastic if the material returns to its original shape (in other words: its reference configuration) after removing these loads. If the shape of the material does not return to its reference configuration, it is called plastic deformation. An example of a material that deforms in an elastic way is rubber, and putting a dent in an iron plate is an example of plastic deformation. This thesis only considers elastic deformation.

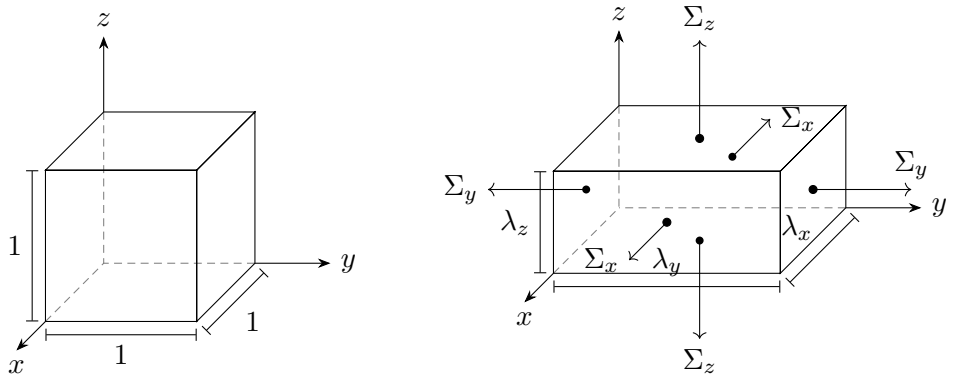

Figure 3.1: Deformation of the unit cube to a block with dimensions λ_x, λ_y and λ_z.

We can see in Figure 3.1 that we have an evenly distributed load Σ_x on the faces normal to the x-axis and loads Σ_y and Σ_z on the faces normal to the y-axis and z-axis respectively. The traction (the force acting over any unit of area) in the x-direction is equal to $\frac{\Sigma_x}{\lambda_y \lambda_z}$, and the traction in the y- and z-direction are $\frac{\Sigma_y}{\lambda_z \lambda_x}$ and $\frac{\Sigma_z}{\lambda_x \lambda_y}$ respectively.
In Figure 3.1, the tractions are normal to the surface of the block. Generally that is not the case. This is the reason why we split the traction vectors into stress-components. Stress is traction in the direction of or perpendicular to the surface normal. We resolve the traction in direction j, with $j \in \{x, y, z\}$ as follows:

$$\mathbf{T}_j = \sigma_{xj}\mathbf{e}_x + \sigma_{yj}\mathbf{e}_y + \sigma_{zj}\mathbf{e}_z. \tag{3.1}$$

With the Einstein Summation Convention we can also write

$$\mathbf{T}_j = \sigma_{ij}\mathbf{e}_i. \tag{3.2}$$

Here σ_{jj}, with $j \in \{x, y, z\}$ is the normal stress in the $\mathbf{e}_j$ direction, and σ_{ij}, with $i, j \in \{x, y, z\}$, $i \neq j$ are shear stresses acting perpendicular to the normal stress σ_{jj}. We now have defined nine stress components (normal stresses in three directions, and two stress components perpendicular to the normal stresses). An important property of stresses is that $\sigma_{ij} = \sigma_{ji}$, with $i, j \in \{x, y, z\}$. Combining all this information, we can write the (symmetric) stress tensor $\underline{\underline{\sigma}}$ as follows:

$$\underline{\underline{\sigma}} = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{xy} & \sigma_{yy} & \sigma_{yz} \\ \sigma_{xz} & \sigma_{yz} & \sigma_{zz} \end{bmatrix}. \tag{3.3}$$

One of the last things we need to define is the displacement. Every material point P in the reference configuration moves to a point P' because of traction. Let $P = X_x\mathbf{e}_x + X_y\mathbf{e}_y + X_z\mathbf{e}_z = \mathbf{X}$. Essentially, $\mathbf{X}$ is a label for material point P. At time t, the point P will be at $P' = x_x\mathbf{e}_x + x_y\mathbf{e}_y + x_z\mathbf{e}_z = \mathbf{x}(\mathbf{X})$. We define the displacement as

$$\mathbf{u} = \mathbf{x} - \mathbf{X} = \mathbf{u}(\mathbf{X}). \tag{3.4}$$

So for every point P, the displacement is the difference between the displaced position $\mathbf{x}$ and the label $\mathbf{X}$. We will only consider small displacements in an isotropic material, so our analysis only has to cover terms that are linear in the displacement components $u_i = x_i - X_i$. This form of elasticity is called Isotropic Linear Elasticity.
Lastly we will define the linear strain and connect that definition to the linear stress. The linear strain tensor is defined as:

$$\underline{\underline{\varepsilon}} = \frac{1}{2}\left(\mathbf{u}\nabla^T + \nabla\mathbf{u}^T\right). \tag{3.5}$$

Hooke's stress-strain law gives us the relationship between the linear stress and strain:

$$\underline{\underline{\sigma}} = \lambda \mathrm{Tr}(\underline{\underline{\varepsilon}})I + 2\mu\underline{\underline{\varepsilon}}, \tag{3.6}$$

where λ and μ are material constants: the bulk modulus and shear modulus respectively. Since we only look at static problems (it would not make sense if the mesh would still move after fitting it to the zero level-set curve), we will look at the equilibrium equation for linear elasticity. For this, we look at the resultant force acting on the material in a region $\mathcal{R}$.
There are two different forces acting on this material: body forces and tractive forces. Body forces are forces measured per unit volume. An example of a body force is gravity. We denote those body forces with $\mathbf{b}$, with components $b_i,\ i \in \{x, y, z\}$ (having the same directions as $\mathbf{e}_i,\ i \in \{x, y, z\}$). Combining this with Equation (3.1) gives us a force balance for every direction $i \in \{x, y, z\}$:

$$\iint_{\delta\mathcal{R}} \sigma_{ij}\mathbf{e}_j \, dS + \iiint_{\mathcal{R}} b_i \, dV = 0,\ j \in \{x, y, z\}. \tag{3.7}$$

In other words: for every direction $i \in \{x, y, z\}$ the sum of the tractive forces and body forces in that direction vanishes. Because this holds for every direction, we apply the divergence theorem to Equation (3.7). This gives us:

$$\iiint_{\mathcal{R}} \left(\nabla \cdot \underline{\underline{\sigma}} + \mathbf{b}\right) = 0, \tag{3.8}$$

where we write as a row vector. This has to hold for every region $\mathcal{R}$. From this, the equilibrium equation for static linear elasticity follows [7]:

$$-\nabla \cdot \underline{\underline{\sigma}} = \mathbf{b}. \tag{3.9}$$

3.1.2 The model

In our model we are only working in the xy-plane, so we only consider two dimensions. We will assume that there are no strains and deformations in the z-direction. So we can consider 2×2 stress- and strain matrices. We also assume there are no body forces. This implies that Equation (3.9) can be rewritten to

$$\nabla \cdot \underline{\underline{\sigma}} = 0. \tag{3.10}$$

To simplify notation, we introduce $X_x = x$ and $X_y = y$.
In our model, the strain matrix is given by

$$\underline{\underline{\varepsilon}} = \begin{bmatrix} \frac{\partial u_x}{\partial x} & \frac{1}{2}\left(\frac{\partial u_y}{\partial x} + \frac{\partial u_x}{\partial y}\right) \\ \frac{1}{2}\left(\frac{\partial u_y}{\partial x} + \frac{\partial u_x}{\partial y}\right) & \frac{\partial u_y}{\partial y} \end{bmatrix}. \tag{3.11}$$

Now from Equation (3.6) follows that

$$\underline{\underline{\sigma}} = \begin{bmatrix} (\lambda + 2\mu)\frac{\partial u_x}{\partial x} + \lambda\frac{\partial u_y}{\partial y} & \mu\left(\frac{\partial u_y}{\partial x} + \frac{\partial u_x}{\partial y}\right) \\ \mu\left(\frac{\partial u_y}{\partial x} + \frac{\partial u_x}{\partial y}\right) & \lambda\frac{\partial u_x}{\partial x} + (\lambda + 2\mu)\frac{\partial u_y}{\partial y} \end{bmatrix}. \tag{3.12}$$

We now get to a system of differential equations:

$$\begin{cases} \frac{\partial}{\partial x}\left[(\lambda + 2\mu)\frac{\partial u_x}{\partial x} + \lambda\frac{\partial u_y}{\partial y}\right] + \frac{\partial}{\partial y}\left[\mu\left(\frac{\partial u_y}{\partial x} + \frac{\partial u_x}{\partial y}\right)\right] = 0 \\ \frac{\partial}{\partial x}\left[\mu\left(\frac{\partial u_y}{\partial x} + \frac{\partial u_x}{\partial y}\right)\right] + \frac{\partial}{\partial y}\left[\lambda\frac{\partial u_x}{\partial x} + (\lambda + 2\mu)\frac{\partial u_y}{\partial y}\right] = 0 \end{cases}. \tag{3.13}$$

This system of differential equations needs some boundary conditions. We define four different boundary conditions. Let Γ be the boundary of our domain Ω. Let $\Gamma_1 \subseteq \Gamma$ be the points for which a deformation in the x-direction is imposed (it is possible that the deformation is 0, in this case there is effectively no deformation, but we treat it in the same way), $\Gamma_2 \subseteq \Gamma$ the points for which a deformation in the y-direction is imposed, $\Gamma_3 \subseteq \Gamma$ the points for which a force in the x-direction is described, and $\Gamma_4 \subseteq \Gamma$ the points for which a force in the y-direction is described. This gives the boundary conditions:

$$\begin{aligned}
u_x(x,y,t) &= u_1(x,y) && \text{for } (x,y) \in \Gamma_1,\ u_1 \in \mathbb{R}, \\
u_y(x,y,t) &= u_2(x,y) && \text{for } (x,y) \in \Gamma_2,\ u_2 \in \mathbb{R}, \\
\left(\underline{\underline{\sigma}}(x,y) \cdot \mathbf{n}\right)_x &= -f_x(x,y,t) && \text{for } (x,y) \in \Gamma_3, \\
\left(\underline{\underline{\sigma}}(x,y) \cdot \mathbf{n}\right)_y &= -f_y(x,y,t) && \text{for } (x,y) \in \Gamma_4.
\end{aligned} \tag{3.14}$$

Note that $\Gamma_1 \cup \Gamma_3 = \Gamma_2 \cup \Gamma_4 = \Gamma$, and $\Gamma_1 \cap \Gamma_3 = \Gamma_2 \cap \Gamma_4 = \emptyset$. This means that on every point on the boundary there is just one boundary condition in both the x- and y-direction. For every direction, either the deformation or the force is described, never both of them. Combining Equations 3.13 and 3.14 gives us the complete model of two-dimensional elastic deformation:

$$\begin{aligned}
\frac{\partial}{\partial x}\left[(\lambda+2\mu)\frac{\partial u_x}{\partial x} + \lambda\frac{\partial u_y}{\partial y}\right] + \frac{\partial}{\partial y}\left[\mu\left(\frac{\partial u_y}{\partial x} + \frac{\partial u_x}{\partial y}\right)\right] &= 0 && \text{for } (x,y) \in \Omega, \\
\frac{\partial}{\partial x}\left[\mu\left(\frac{\partial u_y}{\partial x} + \frac{\partial u_x}{\partial y}\right)\right] + \frac{\partial}{\partial y}\left[\lambda\frac{\partial u_x}{\partial x} + (\lambda+2\mu)\frac{\partial u_y}{\partial y}\right] &= 0 && \text{for } (x,y) \in \Omega, \\
u_x(x,y,t) &= u_1(x,y) && \text{for } (x,y) \in \Gamma_1,\ u_1 \in \mathbb{R}, \\
u_y(x,y,t) &= u_2(x,y) && \text{for } (x,y) \in \Gamma_2,\ u_2 \in \mathbb{R}, \\
\left(\underline{\underline{\sigma}}(x,y) \cdot \mathbf{n}\right)_x &= -f_x(x,y,t) && \text{for } (x,y) \in \Gamma_3, \\
\left(\underline{\underline{\sigma}}(x,y,t) \cdot \mathbf{n}\right)_y &= -f_y(x,y,t) && \text{for } (x,y) \in \Gamma_4.
\end{aligned} \tag{3.15}$$

3.2 Numerical method

In this section, the numerical scheme of our problem will be derived. We will use the Finite Element Method with linear basis functions.
For the first equation in Equation (3.15), we will multiply the equation with a test function v_x, with $v_x = 0$ on Γ_1, and we will integrate over the domain Ω.

$$\begin{aligned}
\int_\Omega \left(\frac{\partial \sigma_{xx}}{\partial x} + \frac{\partial \sigma_{yx}}{\partial y}\right) v_x \, d\Omega &= \int_\Omega \left(\begin{bmatrix}\frac{\partial}{\partial x} \\ \frac{\partial}{\partial y}\end{bmatrix} \cdot \begin{bmatrix}\sigma_{xx} \\ \sigma_{yx}\end{bmatrix}\right) v_x \, d\Omega \\
&= -\int_\Omega \left(\begin{bmatrix}\sigma_{xx} \\ \sigma_{yx}\end{bmatrix} \cdot \begin{bmatrix}\frac{\partial v_x}{\partial x} \\ \frac{\partial v_x}{\partial y}\end{bmatrix}\right) d\Omega + \int_\Gamma \begin{bmatrix}\sigma_{xx} \\ \sigma_{yx}\end{bmatrix} \cdot \mathbf{n} v_x \, d\Gamma \\
&= -\int_\Omega \sigma_{xx}\frac{\partial v_x}{\partial x} + \sigma_{yx}\frac{\partial v_x}{\partial y} \, d\Omega + \int_{\Gamma_3} -f_x v_x \, d\Gamma \\
&= -\int_\Omega \left((\lambda+2\mu)\frac{\partial u_x}{\partial x} + \lambda\frac{\partial u_y}{\partial y}\right)\frac{\partial v_x}{\partial x} + \mu\left(\frac{\partial u_y}{\partial x} + \frac{\partial u_x}{\partial y}\right)\frac{\partial v_x}{\partial y} \, d\Omega \\
&\quad - \int_{\Gamma_3} f_x v_x \, d\Gamma.
\end{aligned} \tag{3.16}$$

Here, the second equation is allowed because of the Divergence Theorem, and f_x in the third equation comes from the boundary condition imposed on Γ_3. Repeating this procedure for the second equation in Equation (3.15) (now with a test function v_y, with $v_y = 0$ on Γ_2), we derive the system of equations

$$\begin{aligned}\int_\Omega \left((\lambda+2\mu)\frac{\partial u_x}{\partial x} + \lambda\frac{\partial u_y}{\partial y}\right)\frac{\partial v_x}{\partial x} + \mu\left(\frac{\partial u_y}{\partial x} + \frac{\partial u_x}{\partial y}\right)\frac{\partial v_x}{\partial y}\,d\Omega &= -\int_{\Gamma_3} f_x v_x\,d\Gamma \\ \int_\Omega \mu\left(\frac{\partial u_y}{\partial x} + \frac{\partial u_x}{\partial y}\right)\frac{\partial v_y}{\partial x} + \left(\lambda\frac{\partial u_x}{\partial x} + (\lambda+2\mu)\frac{\partial u_y}{\partial y}\right)\frac{\partial v_y}{\partial y}\,d\Omega &= -\int_{\Gamma_4} f_y v_y\,d\Gamma. \quad (3.17)\end{aligned}$$

Up until here all the calculations are exact.
After dividing Ω into a triangular grid, we define for every gridpoint $i = 1, \ldots, n$ in the grid:

$$\phi_i(\mathbf{x}) = a_i^0 + a_i^x x + a_i^y y, \qquad a_i^0, a_i^x, a_i^y \in \mathbb{R} \text{ constant per element.} \quad (3.18)$$

We define ϕ_i for every element which contains the gridpoint i, and we restrict the domain of ϕ_i to the element we are considering when calculating the constants. This is the linear basis function we talked about in the introduction in this section. We define $\phi_i(\mathbf{x})$ equal to one in gridpoint i, and equal to zero in every other gridpoint. Between gridpoint i and its neighbours, we have an linearly decreasing function. A typical basis function looks like this:

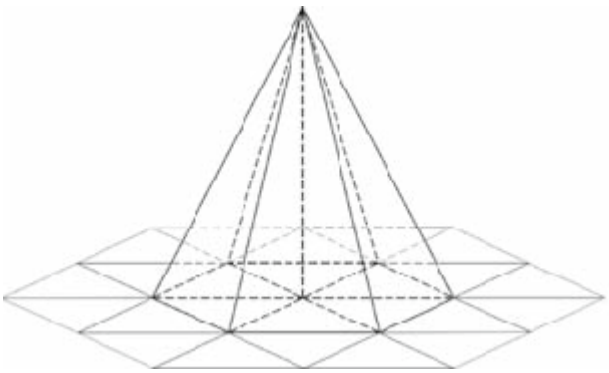

Figure 3.2: A linear basis function. Retrieved from [1]

We will use this function to approximate the solutions u_x and u_y. We will approximate u_x and u_y with a linear combination of the basis functions.

$$\begin{aligned}u_x(\mathbf{x}) &= \sum_{j=1}^{n} u_{x,j}\phi_j(\mathbf{x}) \\ u_y(\mathbf{x}) &= \sum_{j=1}^{n} u_{y,j}\phi_j(\mathbf{x}), \quad (3.19)\end{aligned}$$

and we set $v_x = v_y = \phi_i,\ i = 1, \ldots, n$. Combining Equations (3.17) and (3.19) gives us:

$$\begin{aligned}\int_\Omega \left((\lambda+2\mu)u_{x,j}\frac{\partial\phi_j}{\partial x} + u_{y,j}\lambda\frac{\partial\phi_j}{\partial y}\right)\frac{\partial\phi_i}{\partial x} + \mu\left(u_{y,j}\frac{\partial\phi_j}{\partial x} + u_{x,j}\frac{\partial\phi_j}{\partial y}\right)\frac{\partial\phi_i}{\partial y}\,d\Omega &= -\int_{\Gamma_3} f_x\phi_i\,d\Gamma \\ \int_\Omega \mu\left(u_{y,j}\frac{\partial\phi_j}{\partial x} + u_{x,j}\frac{\partial\phi_j}{\partial y}\right)\frac{\partial\phi_i}{\partial x} + \left(\lambda u_{x,j}\frac{\partial\phi_j}{\partial x} + (\lambda+2\mu)u_{y,j}\frac{\partial\phi_j}{\partial y}\right)\frac{\partial\phi_i}{\partial y}\,d\Omega &= -\int_{\Gamma_4} f_y\phi_i\,d\Gamma. \quad (3.20)\end{aligned}$$

We can write this system as

$$\begin{bmatrix} S_{xx} & S_{xy} \\ S_{yx} & S_{yy} \end{bmatrix} \begin{bmatrix} u_x \\ u_y \end{bmatrix} = \begin{bmatrix} q_x \\ q_y \end{bmatrix}, \tag{3.21}$$

with:

$$\begin{aligned}
(S_{xx})_{ij} &= \int_\Omega (\lambda + 2\mu) \frac{\partial \phi_j}{\partial x} \frac{\partial \phi_i}{\partial x} + \mu \frac{\partial \phi_j}{\partial y} \frac{\partial \phi_i}{\partial y} \, d\Omega \\
(S_{xy})_{ij} &= \int_\Omega \lambda \frac{\partial \phi_j}{\partial y} \frac{\partial \phi_i}{\partial x} + \mu \frac{\partial \phi_j}{\partial x} \frac{\partial \phi_i}{\partial y} \, d\Omega \\
(S_{yx})_{ij} &= \int_\Omega \mu \frac{\partial \phi_j}{\partial y} \frac{\partial \phi_i}{\partial x} + \lambda \frac{\partial \phi_j}{\partial x} \frac{\partial \phi_i}{\partial y} \, d\Omega \\
(S_{yy})_{ij} &= \int_\Omega \mu \frac{\partial \phi_j}{\partial x} \frac{\phi_i}{\partial x} + (\lambda + 2\mu) \frac{\partial \phi_j}{\partial y} \frac{\partial \phi_i}{\partial y} \, d\Omega \\
(q_x)_i &= - \int_{\Gamma_3} f_x \phi_i \, d\Gamma \\
(q_y)_i &= - \int_{\Gamma_4} f_y \phi_i \, d\Gamma.
\end{aligned} \tag{3.22}$$

From [3] follows that $\int_{\Omega_e} g(\mathbf{x}) \, d\Omega \approx \frac{|\Delta|}{6}(g(\mathbf{x}_1) + g(\mathbf{x}_2) + g(\mathbf{x}_3))$, with Ω_e the area of one triangle and $\mathbf{x}_i$ the vertices of the triangle. Also on the boundary we have that $\int_{x_1}^{x_2} \phi_i \, dx = \frac{L}{2}$, with $L = |x_2 - x_1|$. With this information we define the element matrices and vectors as:

$$\begin{aligned}
(S^e_{xx})_{ij} &= \int_{\Omega_e} (\lambda + 2\mu) a^x_j a^x_i + \mu a^y_j a^y_i \, d\Omega = \frac{|\Delta|}{2}((\lambda + 2\mu) a^x_j a^x_i + \mu a^y_j a^y_i) \\
(S^e_{xy})_{ij} &= \int_{\Omega_e} \lambda a^y_j a^x_i + \mu a^x_j a^y_i \, d\Omega = \frac{|\Delta|}{2}(\lambda a^y_j a^x_i + \mu a^x_j a^y_i) \\
(S^e_{yx})_{ij} &= \int_{\Omega_e} \lambda a^x_j a^y_i + \mu a^y_j a^x_i \, d\Omega = \frac{|\Delta|}{2}(\lambda a^x_j a^y_i + \mu a^y_j a^x_i) \\
(S^e_{yy})_{ij} &= \int_{\Omega_e} \mu a^x_j a^x_i + (\lambda + 2\mu) a^y_j a^y_i \, d\Omega = \frac{|\Delta|}{2}(\mu a^x_j a^x_i + (\lambda + 2\mu) a^y_j a^y_i) \\
(q^b_x)_i &= - \frac{L}{2} f_x(x_i, y_i) \\
(q^b_y)_i &= - \frac{L}{2} f_y(x_i, y_i)
\end{aligned} \tag{3.23}$$

The last thing we need to describe is how we get the stiffness matrix S from Equation (3.21). We will use the element matrices for this.

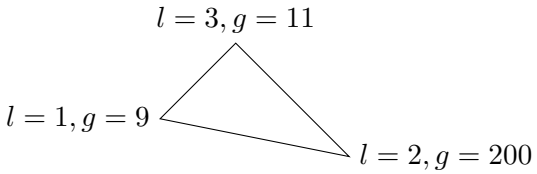

Figure 3.3: Example of one of the triangles in a mesh. l denotes the local index and g denotes the global index.

One of the element matrices from Figure 3.3 is given by

$$S^e_{xx} = \frac{|\Delta|}{2}\begin{bmatrix} (\lambda+2\mu)(a^x_1)^2+\mu(a^y_1)^2 & (\lambda+2\mu)a^x_2a^x_1+\mu a^y_2a^y_1 & (\lambda+2\mu)a^x_3a^x_1+\mu a^y_3a^y_1 \\ (\lambda+2\mu)a^x_1a^x_2+\mu a^y_1a^y_2 & (\lambda+2\mu)(a^x_2)^2+\mu(a^y_2)^2 & (\lambda+2\mu)a^x_3a^x_2+\mu a^y_3a^y_2 \\ (\lambda+2\mu)a^x_1a^x_3+\mu a^y_1a^y_3 & (\lambda+2\mu)a^x_2a^x_3+\mu a^y_2a^y_3 & (\lambda+2\mu)(a^x_3)^2+\mu(a^y_3)^2 \end{bmatrix}. \tag{3.24}$$

The other three element matrices are found in the same way, using the equations in Equation (3.23). The idea is that we add the element matrix to the stiffness matrix, and convert the local indices to global indices in that process. For example: the value $(S^e_{xx})_{11}$ from Equation (3.24) will be added to $(S_{xx})_{9,9}$ in the stiffness matrix, and the value $(S^e_{xx})_{23}$ from the element matrix will be added to $(S_{xx})_{200,11}$ in the stiffness matrix. Calculating all the element matrices for every element in the mesh and adding them to the stiffness matrix, gives us the desired stiffness matrix.
The Dirichlet boundary conditions on Γ_1 and Γ_2 (the imposed deformations in the x- and y-direction) can be processed into the stiffness matrix. We will clarify this with an example. Suppose that $u_x(\mathbf{x}_3) = 5$. This implies that $\mathbf{e}_3^T\mathbf{u}_x + \mathbf{0}^T\mathbf{u}_y = 5$, and we can process this easily into the stiffness matrix. Also note that on the boundary we have 2×1 element vectors, because the boundary elements are lines, which only have 2 vertices.

3.3 The internal conditions and the meshfitting algorithm

The model is not finished without internal conditions. The internal conditions are deformations in the interior of the area we are considering. We want the gridpoints that are close to the zero level-set curve to move to the zero level-set curve. These movements define the initial conditions of the grid. To calculate these conditions, we use the 'Shortest Path Method with Redistribution' algorithm [14]. This algorithm does exactly what we want: move gridpoints to the zero level-set curve, and it also redistributes those gridpoints over the zero level-set curve. More information about that algorithm can be found in [14].
From now on, the area we will consider is a square area with length 2 and centre $(x, y) = (0, 0)$. Our starting mesh will always be a Delaunay triangulation [12] of this square, but we can still vary the size of the triangles.
In Figures 3.4 and 3.6 we can see two different examples of a starting grid, one has a gridsize of 0.125, the other has a gridsize of 0.0625, with imposed deformations (the initial conditions) for a linear zero level-set curve and a circular zero level-set curve (more on those different zero level-set curves in Section 4.1, for now it just serves as an illustration to how we set the initial conditions).

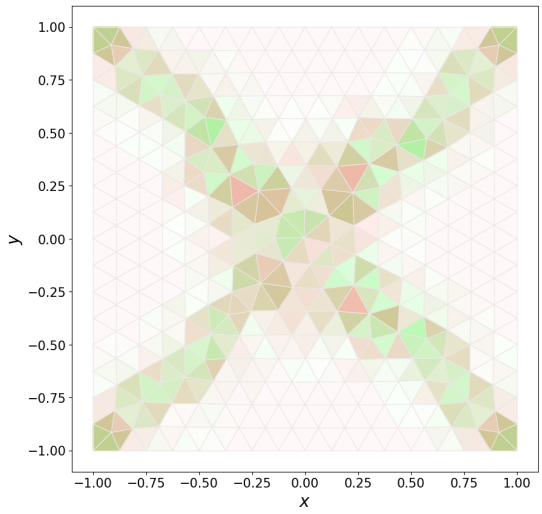

Figure 3.4: The standard grid with gridsize 0.125.

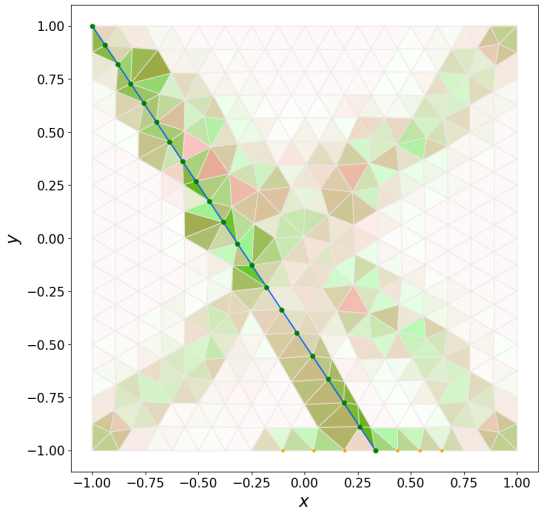

Figure 3.5: The initial deformation for a linear zero level-set curve with gridsize 0.125.

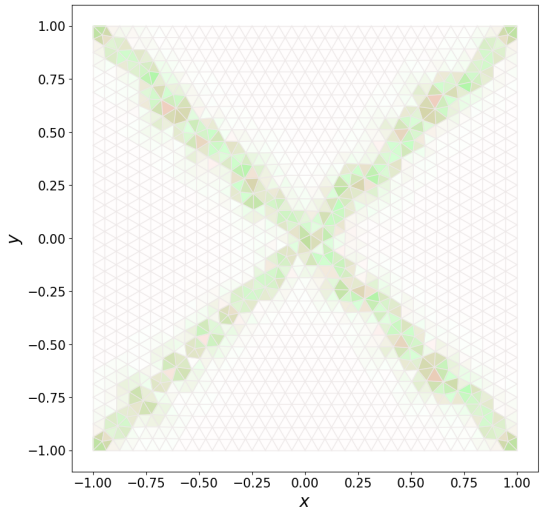

Figure 3.6: The standard grid with gridsize 0.0625.

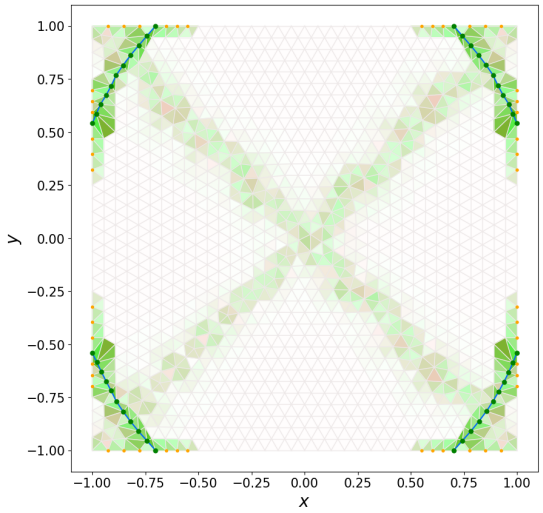

Figure 3.7: The initial deformation for a circular zero level-set curve with gridsize 0.0625.

In Figures 3.4-3.7, the intensity of the red and green in the elements depicts the mismatch in size and skewness respectively, compared to an equilateral triangle with the gridsize as edge length. So, if we consider a mesh with gridsize 0.125, for the colour coding we compare the triangles in the mesh with an equilateral triangle (the standard triangle) with edge length 0.125 (the standard triangle will get a white colour). If there exists an equilateral triangle with edge length 0.2 in the fitted mesh, it will be intensely red (because it is a lot bigger than the standard triangle), and if there exists a very skewed triangle with the same area as a standard triangle (say with angles $(\frac{\pi}{2}, \frac{\pi}{3}, \frac{\pi}{6})$ in radians), it will be intensely green, because it has the same area, but is very skewed compared to the standard triangle. The blue line in Figures 3.5 and 3.7 are the fitted zero level-set curves, and the blue dots on this line are the gridpoints which are on the blue line. The orange dots on the boundary of these figures are the begin- and endpoints

considered in the 'Shortest Path Method with Redistribution' algorithm. The difference between the standard grid and the grid with the internal deformation is the imposed internal deformation. The meshfitting algorithm works as follows:

- For every element in the original undeformed mesh:
  - Calculate the three (different) linear basisfunctions on this element,
  - Calculate the element matrices (Equation (3.23)),
  - Map those element matrices to the stiffnessmatrix S.
- Substitute the boundary conditions and internal conditions to S and $\mathbf{q}$ (the same S and $\mathbf{q}$ as in Equation (3.21)).
- Solve the linear system $S\mathbf{u} = \mathbf{q}$.

After solving the linear system, we find the vector $\mathbf{u}$ of all the deformations in the $x-$ and $y-$direction for every point in the grid.

Chapter 4

Numerical results

In this chapter we will discuss the results of the numerical model derived in Chapter 3. Before we get to the actual results, we first need to do a parameter investigation. In the first section we will introduce the test functions we will use to compare the two models. We already saw two of those functions in Chapter 3, and they are also used in [14]. In the second section, we will see what the influence is of the parameters λ and μ on the results of the model. Based on this, we will introduce an algorithm which optimizes the choices of the parameters. This will be described in the third section. The fourth section will compare the elastic model with the model derived in 2018 by Timo Wortelboer [14]. In the last section, the conclusion of the comparison is discussed.

4.1 Test functions

In this section we will briefly introduce the functions we will use to test our model, and compare it with the spring model. From these functions, we will only use the zero countour line, as if it is a level-set function. We will use the same functions as in [14], except for the 'cosine-times-sine' function. We have four different test functions: a linear function, a circular function, a dumbbell function and a star function. For every function, we will give the function definition, a countour plot (with the zero contour being a thick black line), and we briefly will talk about the difficulties every function has.

4.1.1 Linear test function

The definition of the linear test function is as follows:

$$f(x, y) = 3x + 2y + 1, \tag{4.1}$$

with the contour plot in Figure 4.1.

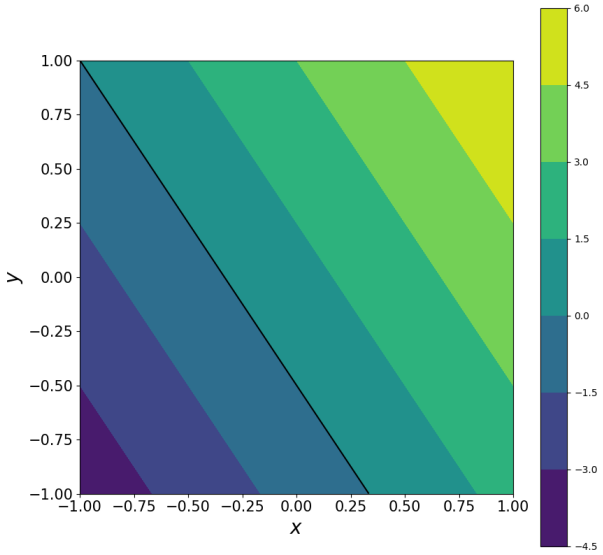

Figure 4.1: The contour plot of the linear test function.

This function has the easiest zero level-set curve, and we can fit the mesh really well to this zero level-set curve, because it is a straight line. The difficulty is the point in the corner, which cannot move, and possibly can create very skewed triangles.

4.1.2 Circular test function

The definition of the circular test function is as follows:

$$f(x, y) = \left(\frac{x}{1.1}\right)^2 + \left(\frac{y}{1.3}\right)^2 - 1, \tag{4.2}$$

with the contour plot in Figure 4.2.

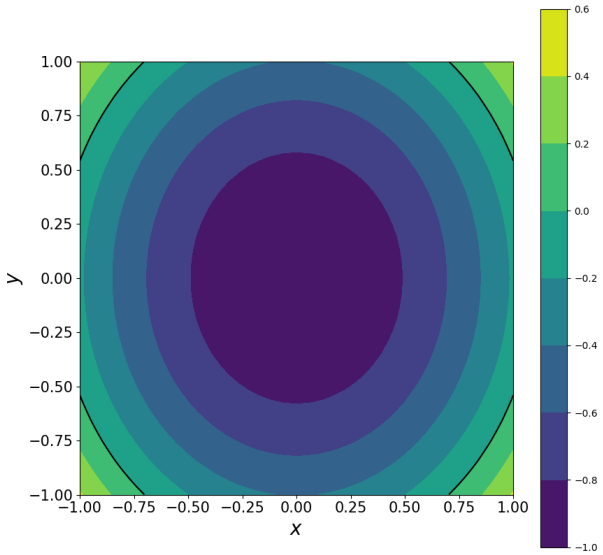

Figure 4.2: The contour plot of the circular test function.

This function is actually an ellips, but it is almost circular, so we still will call it the circular test function, with a circular zero level-set curve. The difficult part of this function is the fact

it crosses the boundary of the square eight times. This is difficult, because the gridpoints on the boundary can only move in the $x-$ or $y-$direction (in such a way that the area remains a square). This can also cause very skewed triangles.

4.1.3 Dumbbell test function

The definition of the dumbbell test function is as follows:

$$f(\mathbf{x}) = \max\left\{\frac{3}{10} - \left\|\mathbf{x} - \left(-\frac{1}{2}, 0\right)\right\|_2, \frac{1}{4} - \left\|\mathbf{x} - \left(\frac{1}{2}, 0\right)\right\|_2, \frac{1}{10} - |y|\right\}, \tag{4.3}$$

where $\mathbf{x} = (x, y)$ and $\|\cdot\|_2$ is the Euclidean norm. The contour plot is given in Figure 4.3.

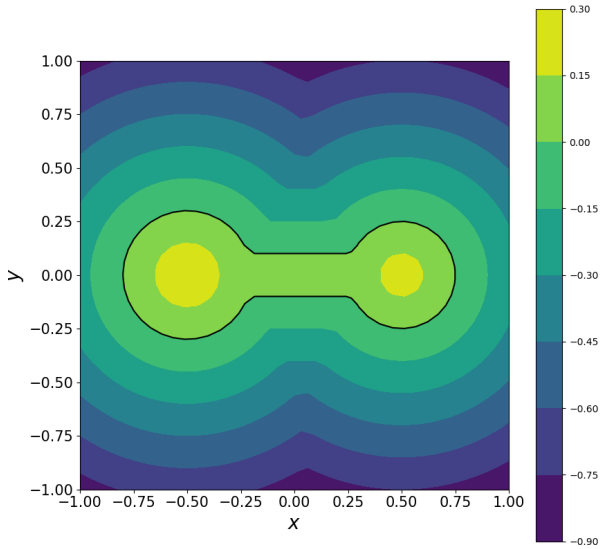

Figure 4.3: The contour plot of the dumbbell test function.

The easy thing about this function is that all the gridpoints near the zero level-set curve have the freedom to move in both the $x-$ and $y-$direction. The hard thing is, if the horizontal parts of the zero level-set curve in the middle are very close together, the choice whether to seperate the two circular regions, or keep them connected. The problem with the two horizontal parts being close together, is that the mesh can only fit to the zero level-set curve with very skewed, small and/or big triangles.

4.1.4 Star test function

The definition of the star test function is as follows:

$$f(x, y) = \frac{3}{5} + \frac{1}{4}\cos\left(6\arctan\left(\frac{y}{x}\right)\right)^2 - (x^2 + y^2), \tag{4.4}$$

with the contour plot in Figure 4.4.

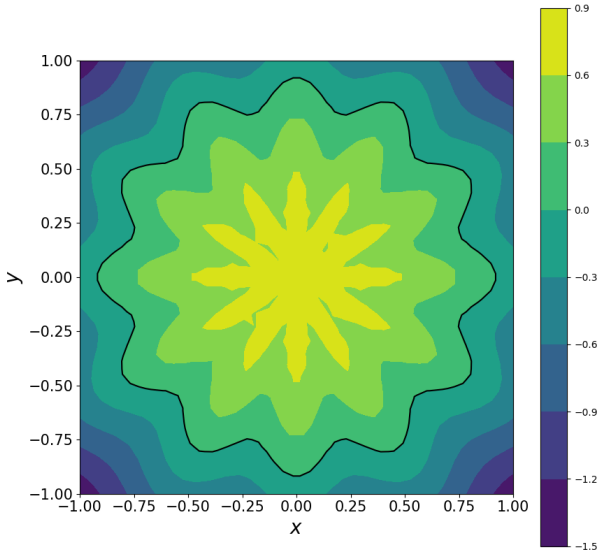

Figure 4.4: The contour plot of the star test function.

The difficulty with this zero level-set curve is the fitting the details of the oscillation of the cosine function. If we choose the gridsize too big, it will be impossible to fit the mesh in a good way to the zero level-set curve. The easy thing, however, is the fact it does not cross the boundary, so every gridpoint close to the zero level-set curve has all freedom to move.

4.2 Parameter investigation

In the numerical derivation of the model in Chapter 3, we have not chosen the parameters λ and μ yet. Before we can choose them wisely, it is necessary to know what the role is of those parameters. They actually have a big impact on how good our model performs. λ, the bulk modulus, defines the 'resistance' of the material to changes in size after applying a force to that material. The shear modulus, μ, is the parameter that defines the 'resistance' of the material to shear deformations. This implies that the ratio of λ and μ defines the 'division' of the stress into bulk deformations and shear deformations. If λ is small compared to μ, the bulk deformations will be big compared to the shear deformations and vice versa, although the magnitude of λ and μ still influences the deformation: suppose we have an elastic material M_1 which has the parameters $\lambda = 1$ and $\mu = 1$, and another elastic material M_2, with parameters $\lambda = 1000$ and $\mu = 1000$. If we apply a force F onto M_1, it will deform a lot more than material M_2 under the same force F, even though the ratio between λ and μ is 1 in both materials.
However, the good thing about our model is that we do not impose forces on boundaries or interior points, but we impose deformations on points. This means that only the ratio of λ and μ is of importance in our model. After all, imposing a deformation on an interior point, will make sure that this interior point does not move, whatever happens to the rest of the grid. This makes the parameter investigation a lot easier, because we will from now on set $\mu = 1$, and vary λ to find the best-fitting mesh.
The first thing we will have a look at, is the way the variation of λ influences the mesh quality, which is described in Section 2.4. For this we will look at the linear and circular zero level-set curve for grid sizes (h) equal to 0.125 and 0.0625.

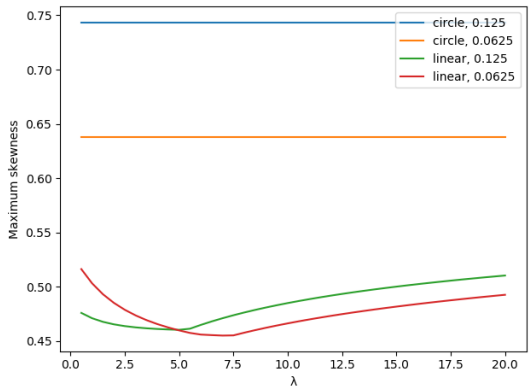

Figure 4.5: The maximum skewness.

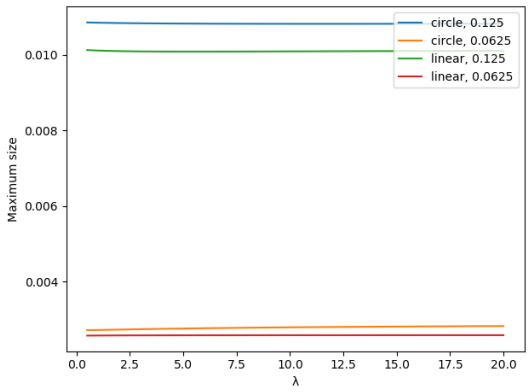

Figure 4.6: The maximum size.

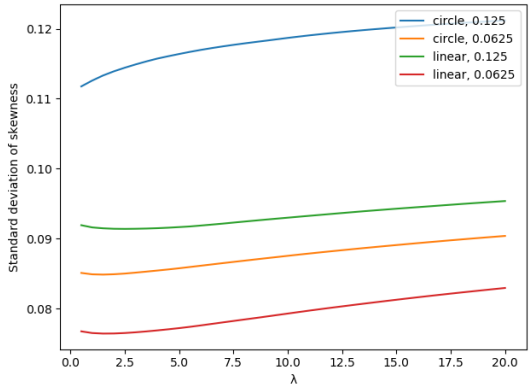

Figure 4.7: The standard deviation of the skewness.

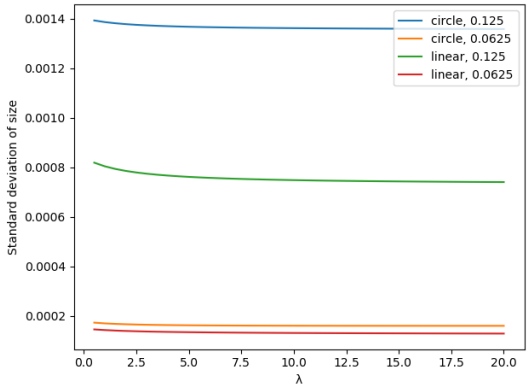

Figure 4.8: The standard deviation of the size.

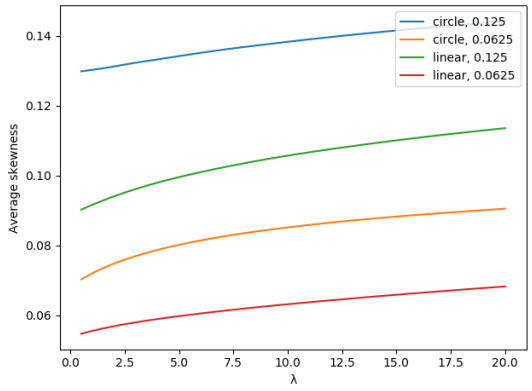

Figure 4.9: The average skewness.

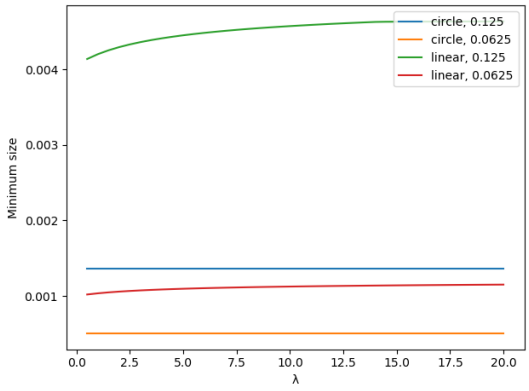

Figure 4.10: The minimum size.

As we can see in Figures 4.5-4.10, some measures of the mesh quality increase when λ increases, some measures decrease when λ increases, some measures have a minimum, and others stay almost the same. Because of that, we cannot give the best choice for λ at first glance. A good question we could ask ourselves right now is: what is the best choice of λ? That depends

on the aspects of the mesh quality we find important. Based on the aspects we find important, we can optimise λ. But first, we can give an estimate for the interval that contains the optimal λ based on the results depicted in Figures 4.5-4.10, to give an idea of the magnitude of λ.
First of all, we see that the maximum skewness of the triangles in the case of the circular test function cannot be influenced by varying λ. The maximum skewness can be minimized, however, in the case of the linear test function. The locations of the minima are $\lambda \approx 5$ and $\lambda \approx 7.5$ for the gridsizes 0.125 and 0.0625 respectively.
The maximum size of the triangles is not influenced by the choice of λ.
To minimise the standard deviation of the skewness, we should take $\lambda \approx 1$ in all cases, except for the circular test function with gridsize 0.125. In that case we should take λ as small as possible.
The standard deviation of the size decreases slightly for $0 \leq \lambda \leq 5$ for both the linear test function and the circular test function with gridsize 0.125, and for $\lambda \geq 5$ it almost stays constant. The same phenomenon happens for both test functions with gridsize 0.0625, but in this case for $\lambda \geq 2.5$ it almost stays constant.
The average skewness increases when λ increases, so to minimise that measure, λ has to be taken as low as possible for both the linear and circular test functions for both gridsizes.
Lastly, the minimum size is not influenced by varying λ for the circular test function. It is slowly increasing for the linear test function with both gridsizes. We want to maximise the minimum size, but we have to keep into account that it is not a very important measure, since the requirement for the size is that the local difference in sizes between the triangles is not very big (see the requirements in Chapter 1).
Based on the information we can get out of Figures 4.5-4.10 by just looking at the graphs, and taking all measures into account, we get the following estimates for the intervals that contain λ in the linear and circular test cases, with gridsizes 0.125 and 0.0625:

Test function	**Lower bound for λ**	**Upper bound for λ**
Linear ($h = 0.125$)	2	5
Linear ($h = 0.0625$)	2.5	5.5
Circular ($h = 0.125$)	0.5	2.5
Circular ($h = 0.0625$)	1.5	2.5

Table 4.1: Estimate of the interval that contains the optimal λ

4.3 Parameter optimisation

In this section we will explain the optimisation of the parameters. In the first subsection we define the optimisation function, followed by an explanation of the Golden-section search in Subsection 4.3.2. We end this section with the results of the optimisation.

4.3.1 Definition of the optimisation function

The aspects of the mesh we find important for the application of this model, are the size and the skewness. But we do not just consider the size and the skewness of the fitted mesh, we compare it to the size and skewness of the mesh we started with. This is a fair assumption, because starting mesh is a really good (in the sense that the variances in size and skewness of the triangles in the mesh are really low) triangulation of the area. Because of that, for every triangle in the fitted mesh, we want it to be as similar as possible to the triangle it started with.

We define the total quality of the mesh as follows:

$$Q(\lambda) = \sum_{e \in \text{elements}} \left(\left(1 + \left(\frac{\text{Size}(e_{\text{st}}) - \text{Size}(e_{\text{fit}})}{\text{Size}(e_{\text{st}})} \right)^2 \right) \cdot \left(1 + (\text{Skew}(e_{\text{st}}) - \text{Skew}(e_{\text{fit}}))^2 \right) - 1 \right). \tag{4.5}$$

Here, for all triangles e, we define e_{st} as the triangle before the meshfitting algorithm, and e_{fit} is the same triangle, but after running the meshfitting algorithm. The 1's are added to prevent adding zero to $Q(\lambda)$ when just the size or the skewness changes. In other words: if we would use

$$\left(\frac{\text{Size}(e_{\text{st}}) - \text{Size}(e_{\text{fit}})}{\text{Size}(e_{\text{st}})} \right)^2 \cdot (\text{Skew}(e_{\text{st}}) - \text{Skew}(e_{\text{fit}}))^2 \tag{4.6}$$

in our function for $Q(\lambda)$, if $\text{Skew}(e_{\text{st}}) = \text{Skew}(e_{\text{fit}})$, Equation (4.6) would equal zero, even if there would be a change in the size. Adding the ones prevents that from happening. The '-1' is to compensate for the added 1 for every triangle. Now, if $e_{\text{st}} = e_{\text{fit}}$, $Q(\lambda) = 0$.
We now have all the information we need to define a minimisation problem. We want to find the value of λ for which $Q(\lambda)$ is minimal. To find this λ, we will use the Golden-section search algorithm [13].

4.3.2 Golden-section search

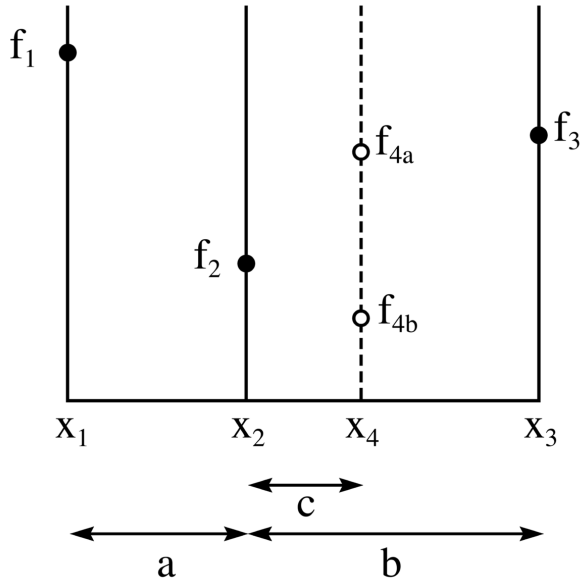

Figure 4.11: Image of a Golden-section search. Retrieved from [13].

In Figure 4.11 we see the variable x on the horizontal axis, and the function evaluations $f(x)$ on the vertical axis. In our case that would be λ on the horizontal axis and $Q(\lambda)$ on the vertical axis. The algorithm consists of a few steps.
The first step is to choose a lower and an upper boundary (in Figure 4.11, these boundaries are x_1 and x_3) in such a way that you are sure that $f(x)$ has a (local) minimum in that interval. The next step is to choose the next point where the function evaluation happens (x_2 in the figure). We choose x_2 using the golden ratio $\left(\frac{1+\sqrt{5}}{2}\right)$. The distance from x_3 to x_2 (b in the figure) divided by the distance from x_1 to x_2 (a in the figure) is equal to the golden ratio. After

choosing x_2, we evaluate $f_1 = f(x_1)$, $f_2 = f(x_2)$ and $f_3 = f(x_3)$. Since f_2 is lower than f_1 and f_3, we now know that the (local) minimum of $f(x)$ lies inside the interval $[x_1, x_3]$.
The next step is to choose x_4 in the biggest interval, namely in the interval (x_2, x_3). Again, we do that in such a way that $\frac{a}{c} = \frac{1+\sqrt{5}}{2}$, where c is the distance between x_2 and x_4.
The following step is to evaluate $f(x_4)$. If $f(x_4) = f_{4a}$, then we know that the minimum of $f(x)$ lies between x_1 and x_4. However, if $f(x_4) = f_{4b}$, we know that the minimum of $f(x)$ lies between x_2 and x_3.
The final step is to lower the upper boundary or increase the lower boundary. We lower the upper boundary to x_4 if $f(x_4) = f_{4a}$ and we increase the lower boundary to x_2 if $f(x_4) = f_{4b}$. We then take the new interval ($[x_1, x_4]$ or $[x_2, x_3]$) and repeat the process, until the length of the interval is smaller than a specific threshold. We then return the value $x_{\min}$ for which $f(x)$ is minimal in our calculations.
Note that this is not necessarily the true location of a (local) minimum of $f(x)$, it is just a value close to the true location of a (local) minimum, because we only know that the true location of a (local) minimum lies within the interval we ended with. In the case of our model, this Golden-section search algorithm returns the value of λ for which $Q(\lambda)$ is close to its true (local) minimum, where 'close' is defined by the stop condition of the algorithm (the size of the interval).

4.3.3 Results of the optimisation

Figures 4.12 and 4.13 show the quality of the fitted mesh for $0 \leq \lambda \leq 9.5$. We see that for every function and for every gridsize, there is a (local) minimum value for the quality.

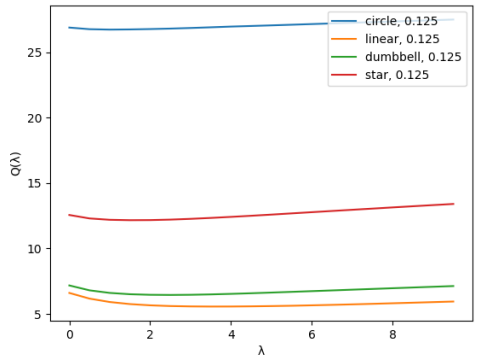

Figure 4.12: Graph of λ versus the quality of the fitted mesh $Q(\lambda)$. Gridsize 0.125.

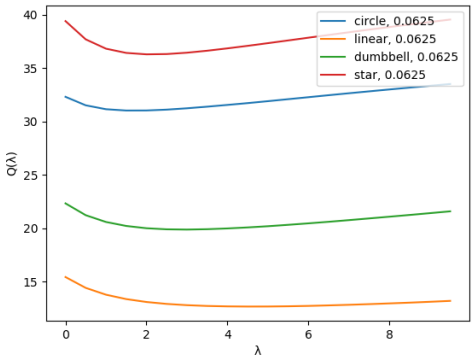

Figure 4.13: Graph of λ versus the quality of the fitted mesh $Q(\lambda)$. Gridsize 0.0625.

The calculated minima of the quality of the mesh and their locations can be found in Table 4.2.

	$h = 0.125$			$h = 0.0625$		
Test function	λ	$Q_{\text{FEM}}(\lambda)$	Q_{EFR}	λ	$Q_{\text{FEM}}(\lambda)$	Q_{EFR}
Linear test function	3.647	5.566	9.648	4.634	12.661	16.876
Circular test function	1.064	26.719	44.533	1.722	31.027	56.615
Dumbbell test function	2.458	6.453	8.506	2.912	19.877	24.445
Star test function	1.596	12.163	10.400	2.129	36.296	47.850

Table 4.2: The results of the optimisation.

In Table 4.2, Q_{FEM} and Q_{EFR} mean the quality of the mesh after applying the elastic model and the spring model respectively, measured with Equation (4.5) (in the case of Q_{EFR}, this is not a function of λ anymore, we just take e_{fit} in Equation (4.5) as the fitted element in the spring model). Note that: the lower Q, the better the quality, because $Q = 0$ means that the mesh is exactly the same as the standard Delauny triangulation of the area. Also, because of the definition of Q, the magnitude of Q tends to increase when the amount of elements increases (i.e. when the gridsize decreases). The amount of elements in the area with $h = 0.125$ is 578, and the amount of elements in the area with $h = 0.0625$ is 2334. Another thing to note is that the values in Table 4.2 are dependent on the starting interval and the stop condition of the Golden-section search algorithm. For the values in this table, we used the starting interval of $[0, 25]$, and we stopped when the size of the interval was lower than 0.25. This means that the values for λ in Table 4.2 are within 0.25 of the location of the true (local) minimum of $Q(\lambda)$. These results for the optimal λ are all inside the intervals we expected in Table 4.1.

4.4 Comparison of the meshfitting algorithms

In this section we find the results of the elastic model (called 'FEM' in the images) and the spring model (called 'Euler forward relaxation' or 'EFR' in the images). The first four subsections show the fitted grids of both models (one subsection per test function). In Subsections 4.4.5 and 4.4.6 we can compare the quality measures (defined in Chapter 2) for both models with those of the standard starting grid. In the images of the meshes (e.g. Figures 4.14, 4.15 etc.), the colour coding is the same as before: the amount of red depicts the size mismatch, and the amount of green depicts the skewness, compared with an equilateral triangle with sides equal to the gridsize. The blue line depicts the fitted zero level-set curve (with the blue points being the gridpoints that are on that curve). The orange dots in the images with the Euler forward relaxation are the gridpoints that have been moved compared to the standard grid. This means that the gridpoints that are neither orange nor blue in the meshes with the Euler forward relaxation have not been moved. All points of the grid in the elastic model, except the corner points, have moved, so it does not make sense to colour every moved gridpoint orange in the images of the elastic model.

4.4.1 Linear test function

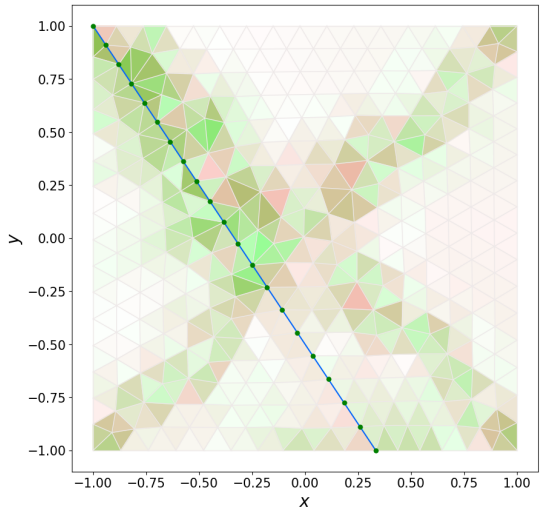

Figure 4.14: The fitted mesh for the linear test function with FEM, with gridsize 0.125.

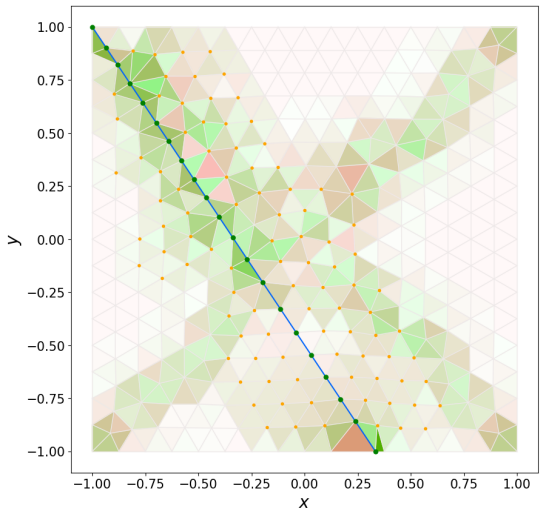

Figure 4.15: The fitted mesh for the linear test function with Euler forward relaxation, with gridsize 0.125.

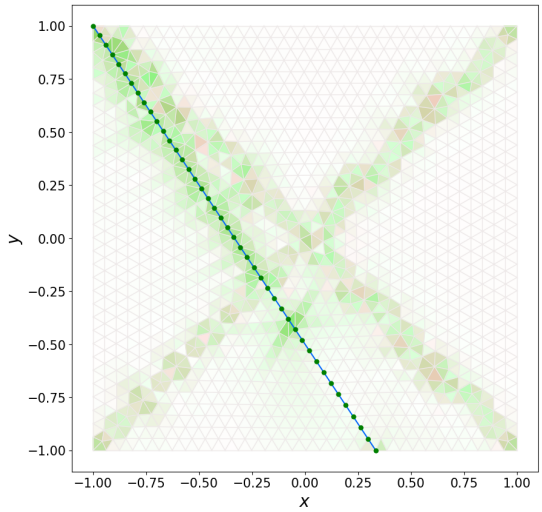

Figure 4.16: The fitted mesh for the linear test function with FEM, with gridsize 0.0625.

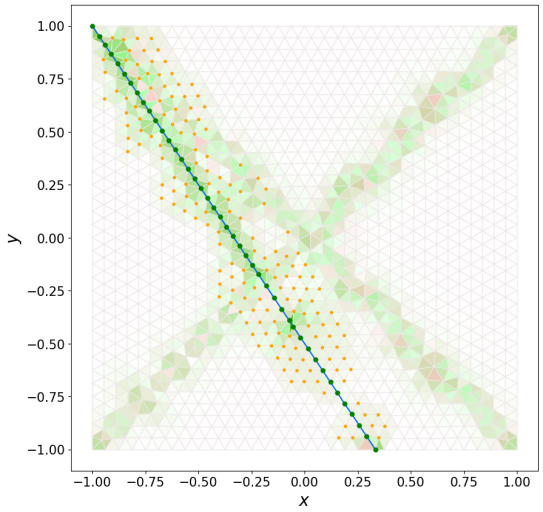

Figure 4.17: The fitted mesh for the linear test function with Euler forward relaxation, with gridsize 0.0625.

In every comparison of images of the elastic model and the spring model, we can see relatively big deformations close to the zero level-set curve in the spring model, while the deformations in the elastic model are a lot more spread out over the whole area. The biggest examples of this are the very big red triangle and the very small and skewed green triangles on the bottom boundary of the area in Figure 4.15. Both fits with gridsize 0.0625 (Figures 4.16 and 4.17) result in a (to the eye) good mesh, with not that many deformations.

4.4.2 Circular test function

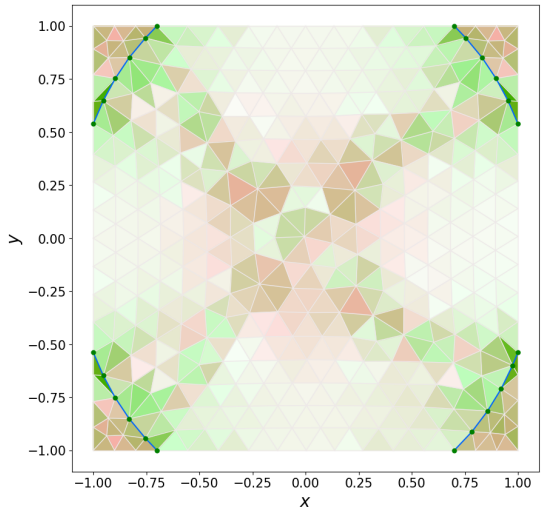

Figure 4.18: The fitted mesh for the circular test function with FEM, with gridsize 0.125.

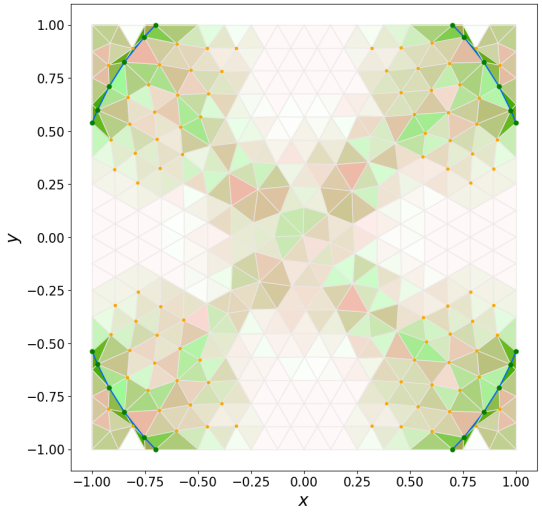

Figure 4.19: The fitted mesh for the circular test function with Euler forward relaxation, with gridsize 0.125.

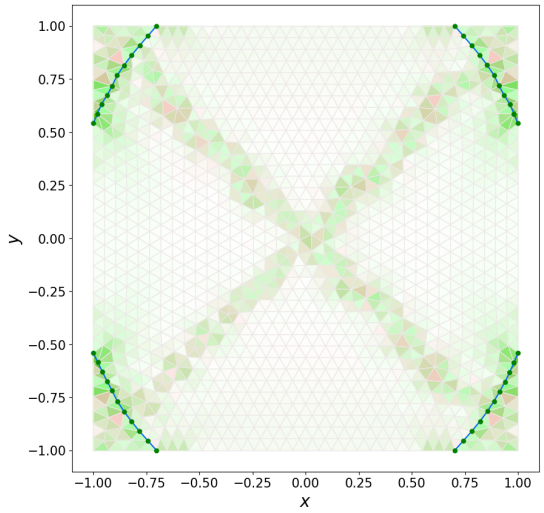

Figure 4.20: The fitted mesh for the circular test function with FEM, with gridsize 0.0625.

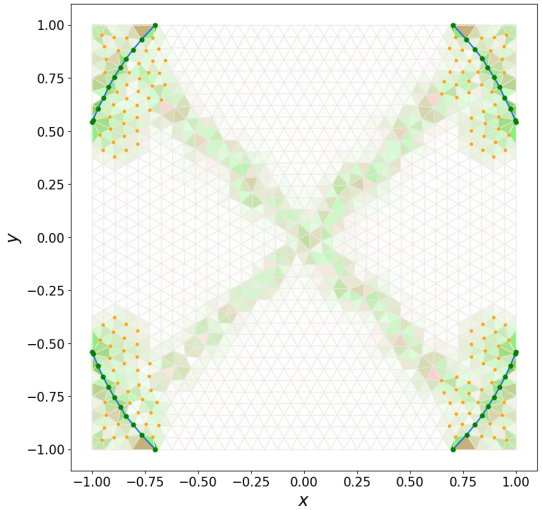

Figure 4.21: The fitted mesh for the circular test function, with Euler forward relaxation, with gridsize 0.0625.

The circular test function is one of the harder test functions to fit. It results in a relatively bad quality mesh. This is because the room in the corner areas of the square is not very big. This results in very small and skewed triangles in those areas. In Figure 4.19 there is one remarkability: all four parts of the mesh in the corner area, have a white triangle inside it. This white triangle is a(n) (almost) 'perfect' triangle, but it does not seem very optimal in this case, because it results in a lot of even more skewed triangles than in Figure 4.18. The inside of the circle is not deformed much in all cases.

4.4.3 Dumbbell test function

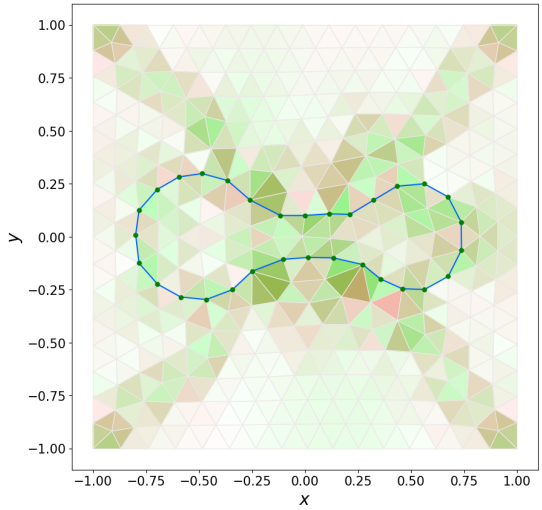

Figure 4.22: The fitted mesh for the dumbbell test function with FEM, with gridsize 0.125.

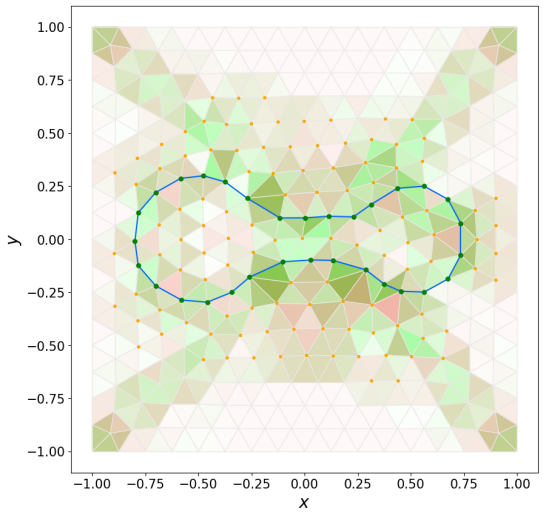

Figure 4.23: The fitted mesh for the dumbbell test function with Euler forward relaxation, with gridsize 0.0625.

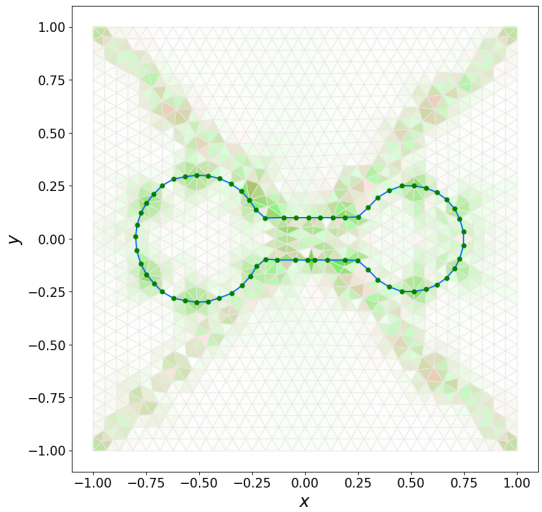

Figure 4.24: The fitted mesh for the dumbbell test function with FEM, with gridsize 0.125.

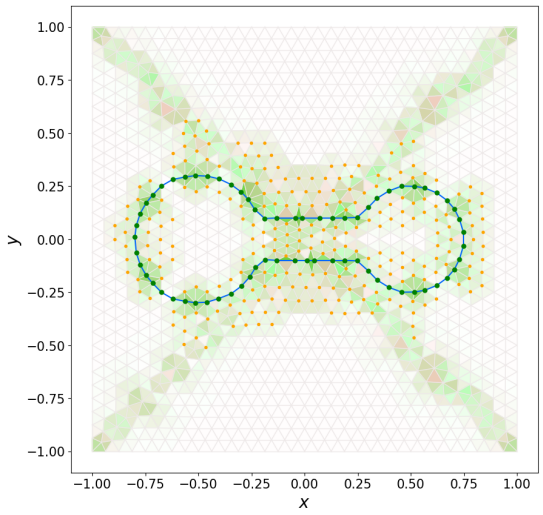

Figure 4.25: The fitted mesh for the dumbbell test function with Euler forward relaxation, with gridsize 0.0625.

For the elastic model, the dumbbell test function gives the second best fitted mesh, only after the linear mesh. This is the same for the spring model, but the fit for the dumbbell test function performs even better than the fit for the linear test function, for gridsize 0.125 (see Table 4.2). The differences between the results of the two models for the dumbbell test function are to the eye not very substantial.

4.4.4 Star test function

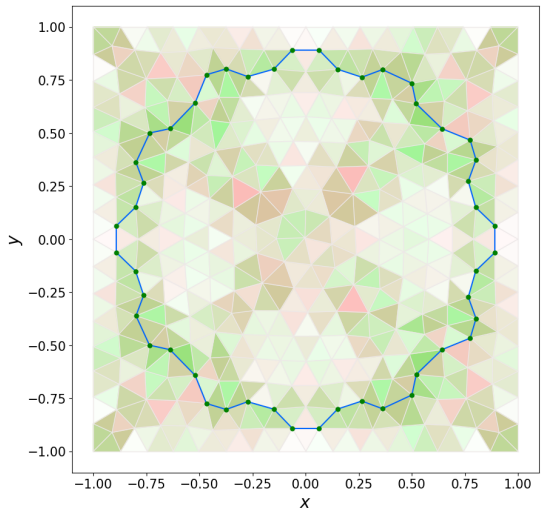

Figure 4.26: The fitted mesh for the star testfunction with FEM, with gridsize 0.125.

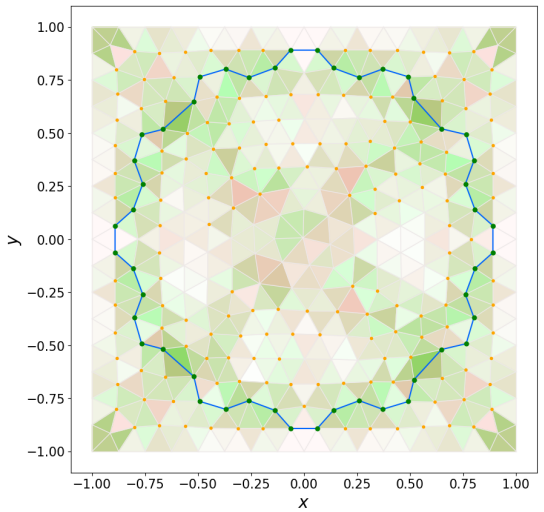

Figure 4.27: The fitted mesh for the star test function with Euler forward relaxation, with gridsize 0.125.

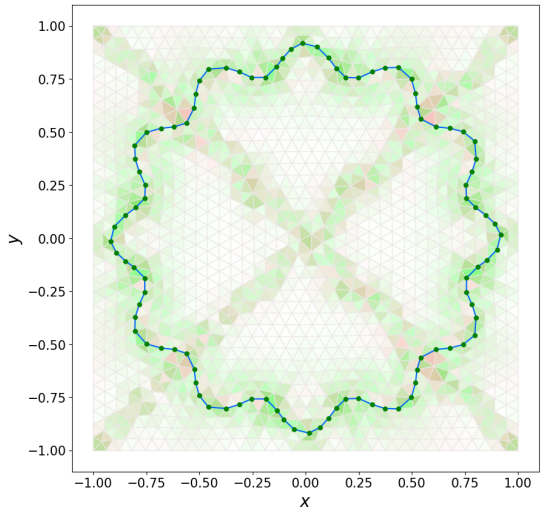

Figure 4.28: The fitted mesh for the star test function with FEM, with gridsize 0.125.

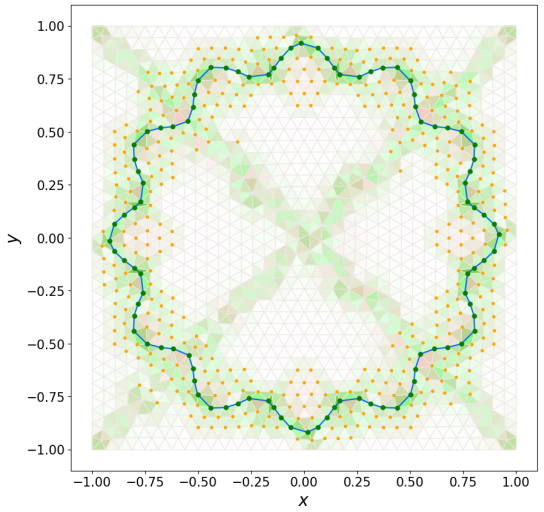

Figure 4.29: The fitted mesh for the star test function with Euler forward relaxation, with gridsize 0.0625.

The star test function is the only test function that performs worse in the elastic model than in the spring model for gridsize 0.125 (in Table 4.2, the quality measure of the mesh in Figure 4.26 is higher than the mesh in Figure 4.27). The deformations in the mesh in Figure 4.28 are a lot more spread out over the whole mesh than in the mesh in Figure 4.29. Examples of very big deformations in Figure 4.29 which are not present in Figure 4.28 are the very skewed triangles near $(x, y) = (-0.75, -0.20)$ and $(x, y) = (-0.75, 0.20)$.

4.4.5 Quality measures for gridsize 0.125

For every quality measure treated in Section 2.4, we first give a histogram that depicts that measure for the fitted meshes (using both the elastic model and the spring model), for all four test functions, followed by a small discussion about the results.

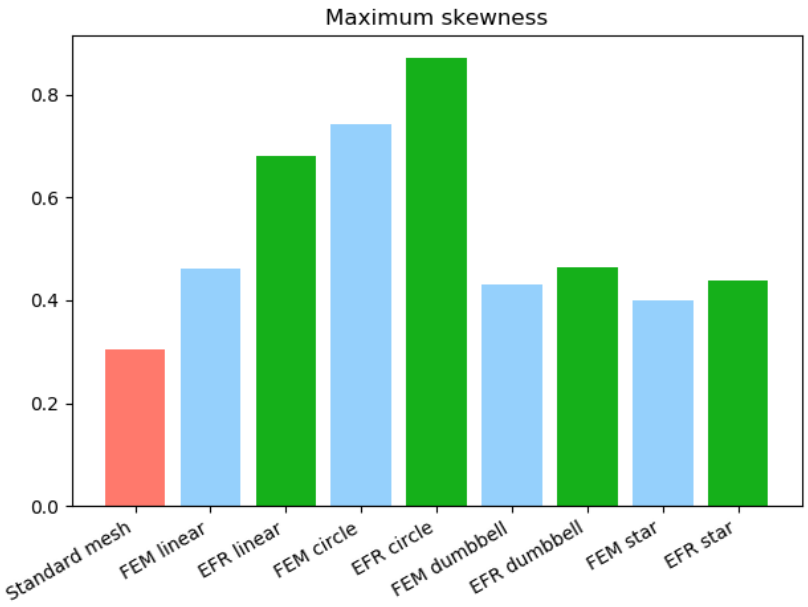

Figure 4.30: The maximum skewness measure for gridsize 0.125.

For every test function, the grid of the elastic model performs better with respect to the maximum skewness than the grid of the spring model.

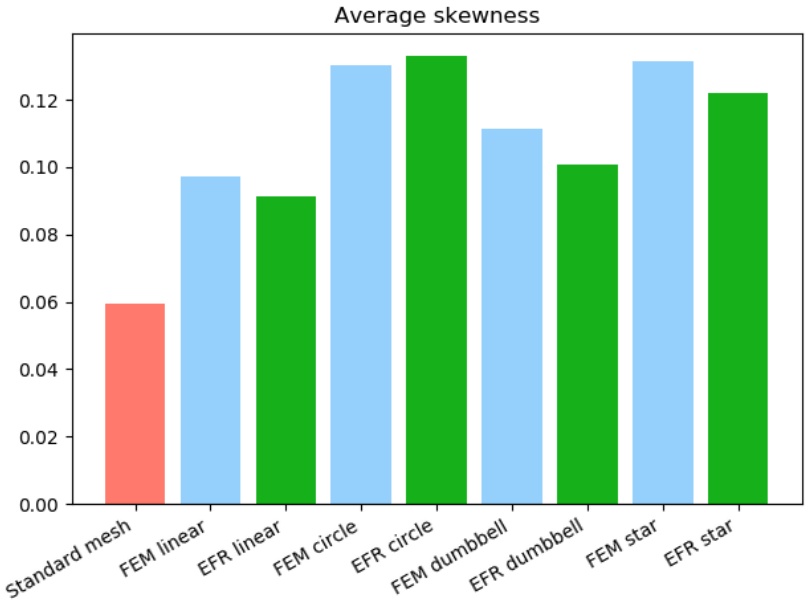

Figure 4.31: The average skewness measure for gridsize 0.125.

The average skewness is the measure the spring model generally performs better on than the elastic model. The reason for this is that, as a result of meshfitting, in the spring model only the triangles near the zero level-set curve move, which leaves most triangles in the mesh untouched (hence it leaves most triangles (almost) equilateral, and the skewness of equilateral triangles is 0), while the imposed deformation (the internal conditions from Section 3.3) 'spreads out' a lot more in the elastic model (this gives small deformations everywhere on the grid, hence the higher average skewness in most cases).

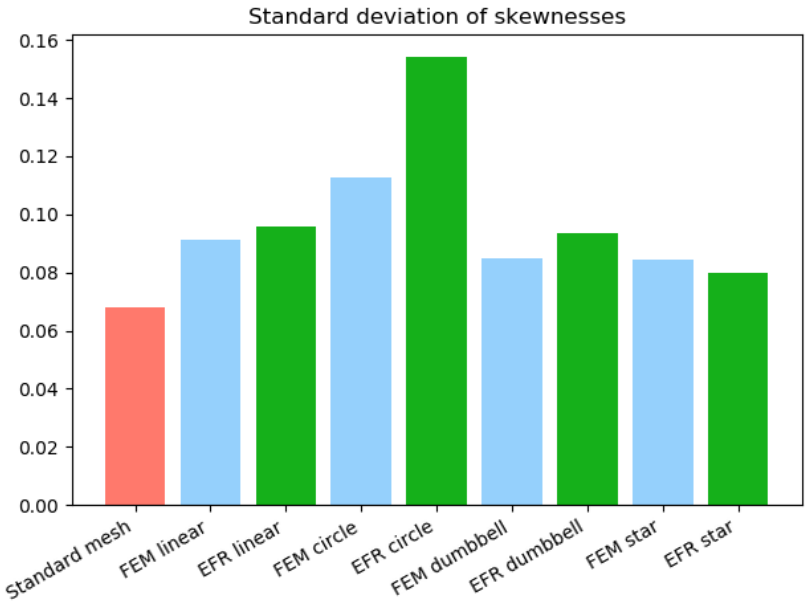

Figure 4.32: The standard deviation of the skewness measure for gridsize 0.125.

The 'spread' of the imposed deformation over the whole area after meshfitting in the elastic model generally results in a lower standard deviation of the skewness compared to the spring model (Figure 4.32). The exception to this is the star test function.

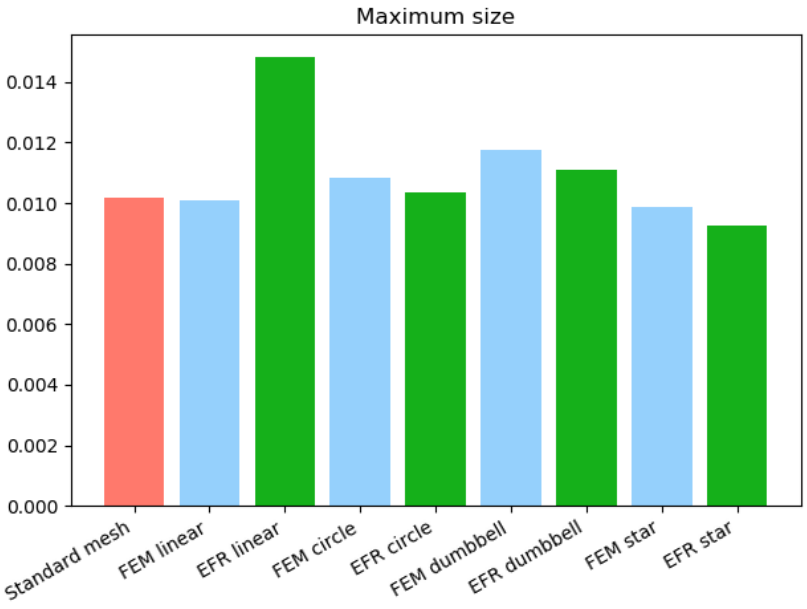

Figure 4.33: The maximum size measure for gridsize 0.125.

The spring model performs a little bit better than the elastic model on the maximum size measure with all test functions, except for the linear test function, where it performs a lot worse than the elastic model.

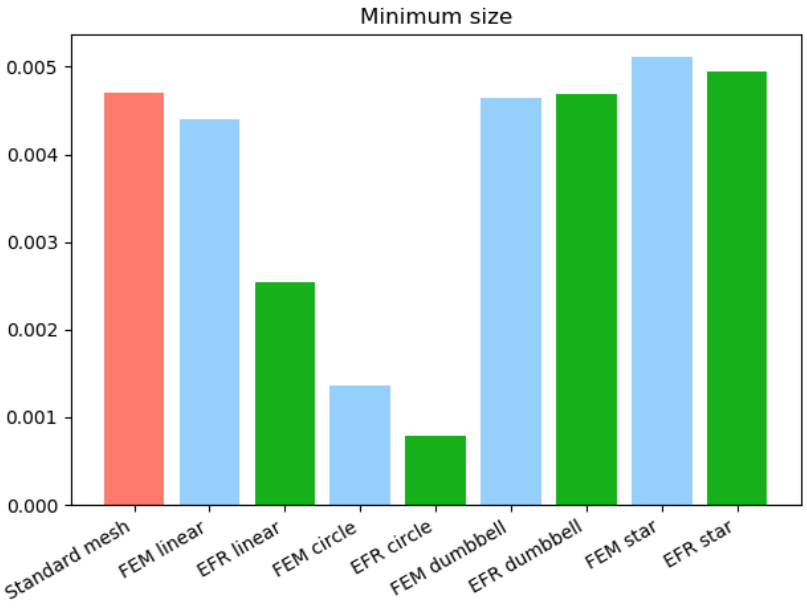

Figure 4.34: The minimum size measure for gridsize 0.125.

The minimum size is a measure which generally performs better on the elastic model than on the spring model.

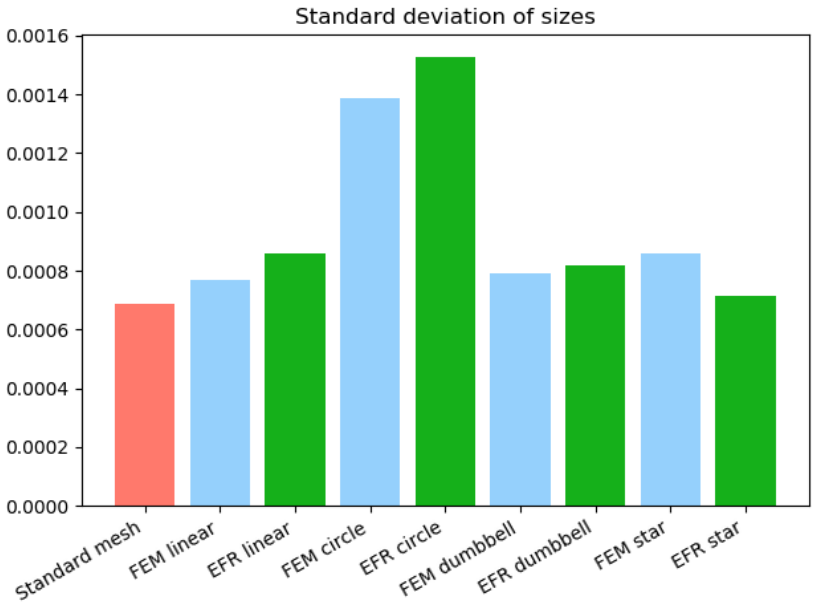

Figure 4.35: The standard deviation of the size measure for gridsize 0.125.

With the same reason as given in the discussion after Figure 4.32, the elastic model generally performs better than the spring model with respect to the standard deviation of the size.

4.4.6 Quality measures for gridsize 0.0625

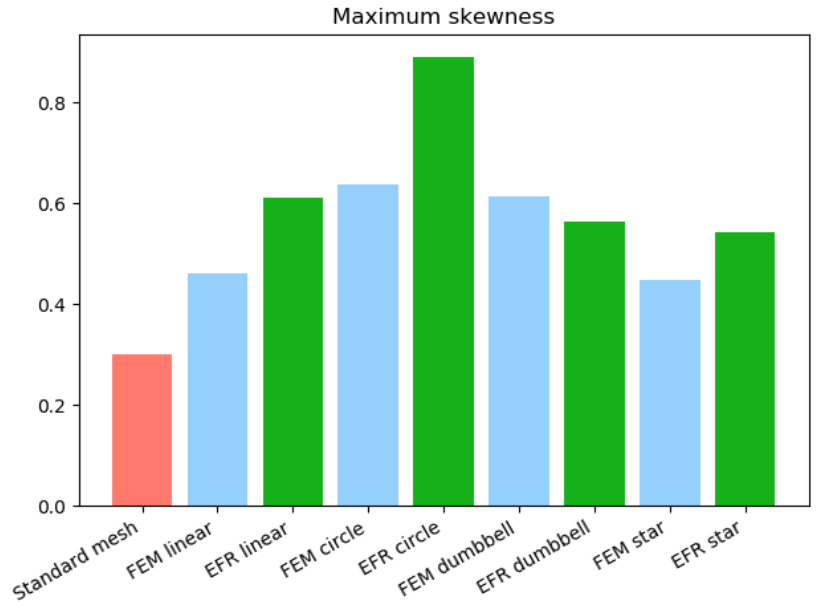

Figure 4.36: The maximum skewness measure for gridsize 0.0625.

The maximum skewness generally is a lot better in the elastic model than in the spring model. The maximum skewness of the fitted grid in the elastic model is slightly worse than the fitted grid in the spring model for the dumbbell test function.

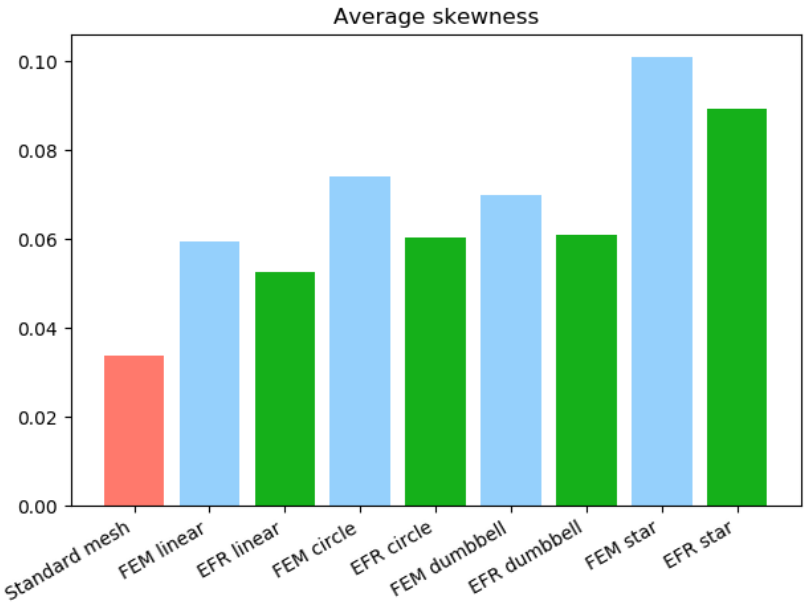

Figure 4.37: The average skewness measure for gridsize 0.0625.

The spring model performs better than the elastic model in the average skewness measure. Again, this is because of the way both models work (see the discussion about the results in Figure 4.31).

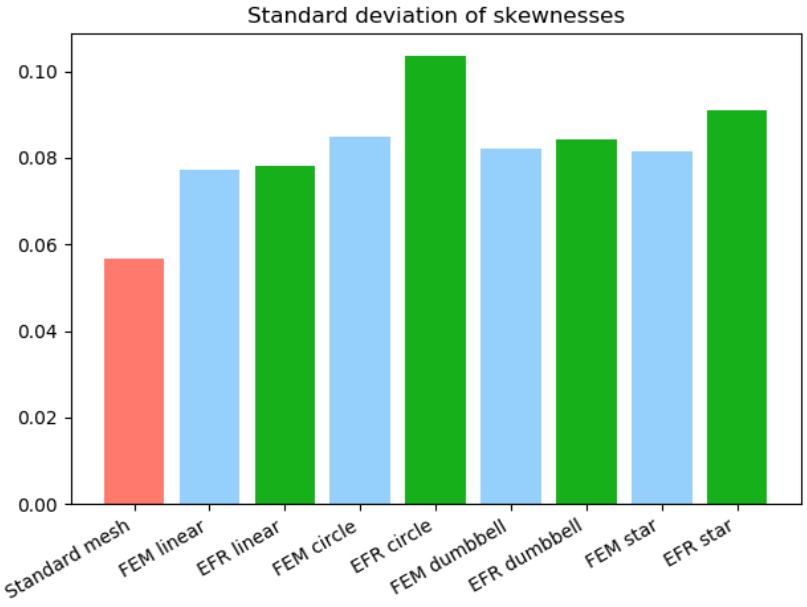

Figure 4.38: The standard deviation of the skewness measure for gridsize 0.0625.

The elastic model performs better than the spring model on all testfunctions, if we look at the standard deviation of the skewness.

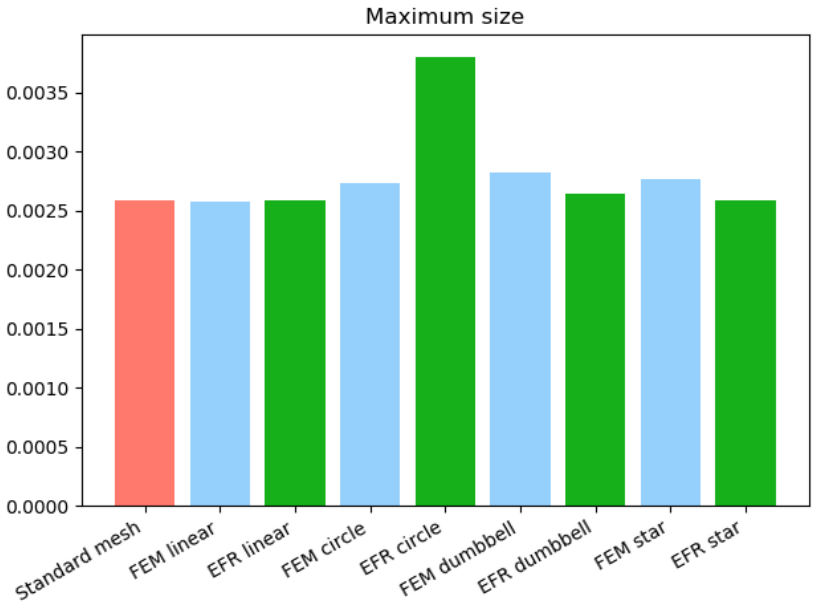

Figure 4.39: The maximum size measure for gridsize 0.0625.

The elastic model and the spring model have small differences if we look at the maximum size, except for the circular test function: the elastic model performs a lot better for that test function.

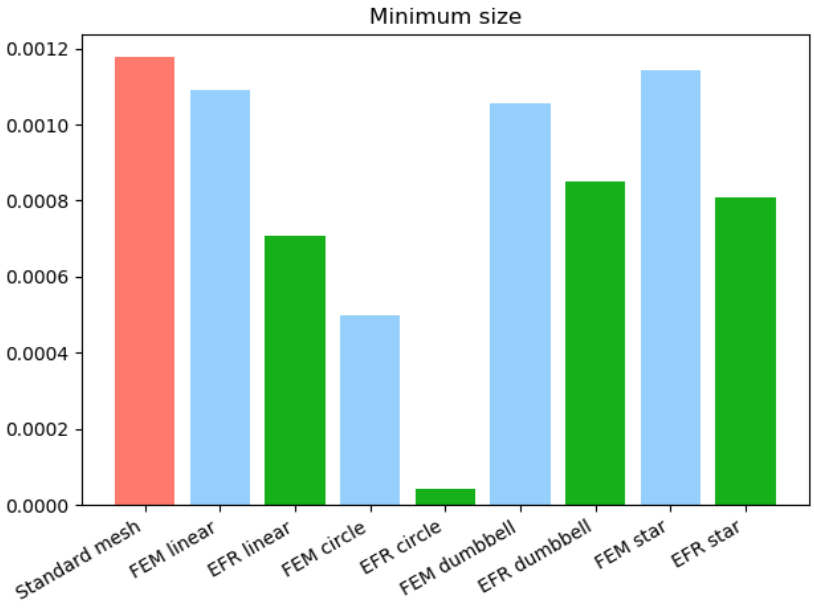

Figure 4.40: The minimum size measure for gridsize 0.0625.

For every test function, the mesh in the elastic model results in a better minimum size than the mesh in the spring model.

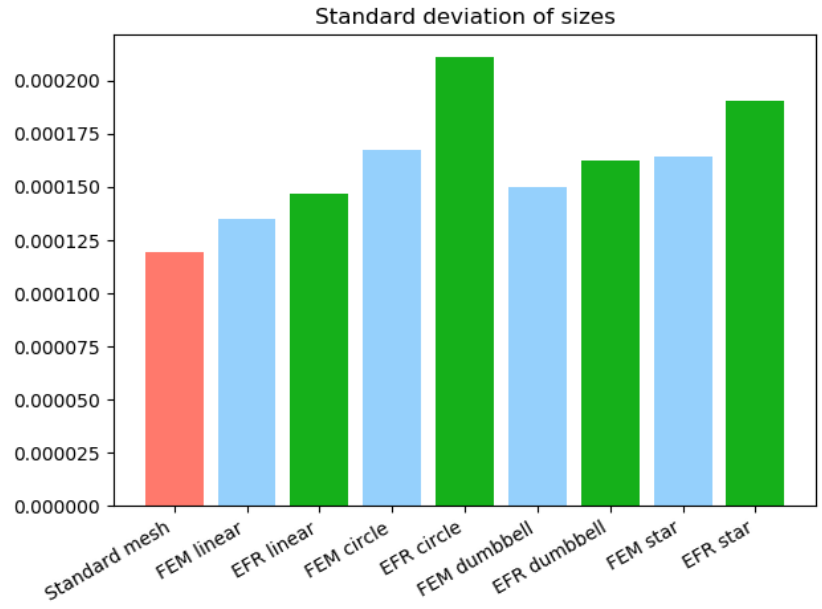

Figure 4.41: The standard deviation of the size measure for gridsize 0.0625.

The standard deviation of the size is better in the elastic model than in the spring model, for every test function.

4.5 Conclusion

In the quality measures of the size, the elastic model almost always seems to perform better than the spring model. Also, the standard deviations of the sizes and skewnesses seem to be better with the elastic model, in most cases. However, the spring model has a lower average skewness in most cases.
The problem with comparing both models is that the elastic model 'spreads' the imposed deformation by the zero level-set curve over the entire mesh, while the spring model 'spreads' the imposed deformation only over the elements close to the zero level-set curve, and it keeps the rest of the mesh untouched, which makes the local deformations close the the zero level-set curve a lot bigger in the spring model than in the elastic model. Combining the mesh qualities found in Table 4.2 and the results discussed in Section 4.4, we can say that the elastic model performs better than the spring model for random zero level-set curves.

Chapter 5

Discussion and future research

In this chapter we will go over some improvements that can be made on the elastic model, and we will give some advices/ideas for future research on the subject of meshfitting.

5.1 Applying the elastic model before the mesh redistribution

The first change we could make to the current elastic model would be to not use the starting deformations of the 'Shortest Path Method with Redistribution', like in Figures 3.5 and 3.7. By applying extra internal conditions, we could probably give the internal gridpoints more freedom to fit the zero level-set curve, which then might increase the performance of the elastic model. The extra internal conditions would be as follows: for some points $\mathbf{x}$ close to the zero level-set curve, we need that $\phi(\mathbf{x} + \mathbf{u}(\mathbf{x})) = 0$, with ϕ the zero level-set curve and $\mathbf{u}(\mathbf{x})$ the deformation of gridpoint $\mathbf{x}$. The words 'some' and 'close' of course have to better defined for this change in the elastic model. An advantage of this change might be that the gridpoints which are now fixed on the zero level-set curve, can move over the zero level-set curve, which might increase the quality of the mesh.

5.2 Testing the methods for multiple zero level-set curves in an area

Right now we only tested the models for the linear, star, dumbbell and circle zero level-set curves, but interesting might be to test both models for multiple zero level-set curves in an area. An example of such a level-set curve might be:

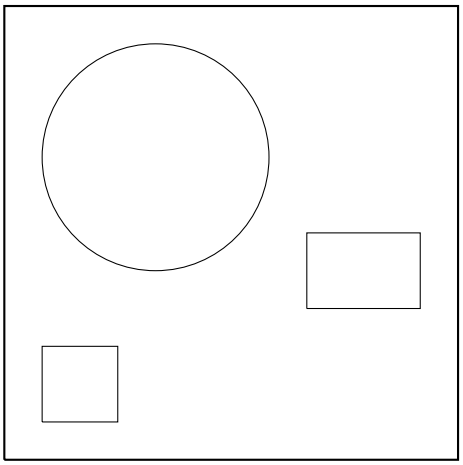

Figure 5.1: Possible zero level-set curve. Here, the circle, the small square and the rectangle depict the zero level-set curve. The big square is the total area of the mesh.

5.3 Modelling a non-homogeneous material

In the current elastic model, we use a homogeneous λ and μ. Something that could be an improvement over the current model, would be modelling a non-homogeneous material. We could maybe even use a form of the Golden-section search algorithm to optimise the values of λ and μ over the whole mesh. An advantage of this method is that we can increase λ and μ in places where we do not want deformations and decrease those parameters in places where we want a lot of deformation. In the current model, we just choose a λ and μ such that 'on average' the deformations are the best ones we can get.

5.4 Higher dimensions [14]

An obvious extension of the elastic model in two dimensions, would be an elastic model in three dimensions. The problem with this extension lies with the fact that Dijkstra's shortest path algorithm used in the 'Shortest Path Method with Redistribution', cannot be applied in three dimensions, because the zero level-set curve is a surface instead of a line. In [14], Timo Wortelboer introduces a solution using a dual graph. But if the change in the elastic model, described in Section 5.1, is implemented, and if the shortest path algorithm is not necessary anymore, the extension to three dimensions is doable.

5.5 Goodness of fit

In the Introduction (Chapter 1), the first requirement of the triangulation is: 'it [the triangulation] needs to fit the zero level-set curve'. We have not discussed much about this requirement in this thesis, but what we can see is that the fit of the mesh (both the elastic and the spring model) follows the zero level-set curve well in all the test functions we used. However, it is definitely a good idea to quantify how well this fit follows the zero level-set curve. An idea for the implementation of this Goodness of fit measure is the following: we will look at two aspects of the zero level-set curve: the difference in length of the zero level-set curve and the fitted mesh (or the circumference for a closed curve) and the area between the zero level-set curve and the

fitted mesh. The measure for the length is as follows:

$$L(M) = |\ell(\Gamma_0) - \ell(\Gamma_{0_{\text{fit}}})|, \tag{5.1}$$

where M is the mesh, Γ_0 the zero level-set curve, $\Gamma_{0_{\text{fit}}}$ the fitted zero level-set curve in the mesh, and $\ell(\cdot)$ is the length of a curve. This measure is zero if the mesh exactly follows the zero level-set curve, but might still be zero if it does not follow the zero level-set curve, because this measure does not look at the shape of the zero-level set curve and its fit. This is why we also need to look at the area between the zero level-set curve and the fitted mesh. An elegant way to do this is as follows:

$$A(M) = \int_0^1 ||\mathbf{x}_{\Gamma_0}(t) - \text{proj}_{\Gamma_{0_{\text{fit}}}} \mathbf{x}_{\Gamma_0}(t)||\, dt. \tag{5.2}$$

Here we parameterise the zero level-set curve with parameter t such that if we let t vary from 0 to 1, we walk over the entire curve, (with $\mathbf{x}_{\Gamma_0}$ the Cartesian coordinates of a point on the zero level-set curve). In the integral we take the difference between the zero level-set curve and its projection on the fitted mesh. This will give us a measure of the area between the zero level-set curve and the fitted mesh. The only problem with this measure is that it might be hard to implement, because of the parameterisation and the projection formulas needed. So we simplify it as follows:

$$A(M) = \int_\Omega |H(\phi(\mathbf{x})) - H(\phi_{\text{fit}}(\mathbf{x}))|\, d\Omega, \tag{5.3}$$

where $H(\mathbf{x})$ is the Heaviside step function and $\phi(\mathbf{x})$ is the level-set function.
If both $L(M)$ and $A(M)$ are equal to zero, the fitted mesh follows the zero level-set curve exactly, and if the fitted mesh does not follow the zero level-set curve exactly, at least one of the two measures will be greater than zero.

Bibliography

[1] Z. Fašková, R. Čunderlík, and K. Mikula. "Finite element method for solving geodetic boundary value problems". In: *Journal of Geodesy* 84 (Feb. 2009), pp. 135–144. DOI: 10.1007/s00190-009-0349-7.

[2] E. Javierre Pérez. *Numerical methods for vector Stefan models of solid-state alloys*. Delft: Technische Universiteit Delft, 2006.

[3] J. van Kan, A. Segal, and F. Vermolen. *Numerical Methods in Scientific Computing*. Delft: Delft Academic Press/VSSD, 2014.

[4] D. Lay. *Linear Algebra and Its Applications*. Fourth. Edinburgh Gate: Pearson Education Limited, 2014.

[5] Jianghua Li, Nikolas Provatas, and Geoffrey Brooks. "Kinetics of scrap melting in liquid steel". In: *Metallurgical and Materials Transactions B* 36 (Jan. 2005), pp. 293–302. DOI: 10.1007/s11663-005-0031-2.

[6] D. Den Ouden. *Mathematical Modelling of Nucleating and Growing Precipitates: Distributions and Interfaces*. Delft: Technische Universiteit Delft, 2015.

[7] D.F. Parker. *Fields, Flows and Waves: An Introduction to Continuum Models*. London: Springer-Verlag London, 2003.

[8] J.A. Sethian. "Curvature and the evolution of fronts". In: *Communications in Mathematical Physics* 101 (Dec. 1985), pp. 487–499. DOI: 10.1007/BF01210742.

[9] J. Steppeler. "A Galerkin finite element-spectral weather forecast model in hybrid coordinates". In: *Computers & Mathematics with Applications* 16.1 (1988), pp. 23–30. ISSN: 0898-1221. DOI: https://doi.org/10.1016/0898-1221(88)90021-1. URL: http://www.sciencedirect.com/science/article/pii/0898122188900211.

[10] E. Süli. *Lecture notes on Finite Element Methods for Partial Differential Equations*. Oxford: Mathematical Institute, University of Oxford, 2000.

[11] T. Verbeek. *Modelling and Simulating Three Phases of Steel*. Delft: Technische Universiteit Delft, 2018.

[12] Wikipedia. *Delaunay triangulation*. [Online; accessed 15 June, 2019]. URL: https://en.wikipedia.org/wiki/Delaunay_triangulation.

[13] Wikipedia. *Golden-section search*. [Online; accessed 13 June, 2019]. URL: https://en.wikipedia.org/wiki/Golden-section_search.

[14] T.E. Wortelboer. *Meshfitting for the Level-set Method*. Delft: Technische Universiteit Delft, 2018.

[15] Ysroslav Zasiadko et al. "A STUDY INTO ICE BUILD-UP AND MELTING ON VERTICAL COOLED PIPES". In: (Aug. 2016). DOI: 10.15673/ret.v52i3.117.