A new doorsill design for the BMW 6 series cabriolet

H.W.J.A. Schepens

DCT 2006.065

Traineeship report

Coach(es):  Ing. H. Schneeweiss
            Dr.-Ing. L. Cremers
            Dr.-Ing. M. Kroiss

Supervisor:  Prof. dr. H. Nijmeijer

Technische Universiteit Eindhoven
Department Mechanical Engineering
Dynamics and Control Group

Eindhoven, September, 2006
## Contents

1 Introduction .................................................. 7

2 Cross-section description for optimization .................. 9
   2.1 Introduction ............................................ 9
   2.2 Problem definition .................................... 9
   2.3 The cross section representation ....................... 10
      2.3.1 PBAR ............................................. 11
      2.3.2 PBARL ........................................... 11
      2.3.3 PBRSECT ......................................... 12
   2.4 Summary ................................................ 15

3 Design of the Medial axis algorithm ....................... 17
   3.1 Introduction ............................................ 17
   3.2 Yang approximately medial axis method ................ 17
   3.3 The linear medial axis algorithm ...................... 19
   3.4 Summary ................................................ 21

4 Description of the measurements ............................ 23
   4.1 Introduction ............................................ 23
   4.2 The Frequency Response Function ....................... 23
      4.2.1 FRF estimation .................................... 23
      4.2.2 Frequency limitations ............................ 25
      4.2.3 The coherence function .......................... 27
   4.3 Measurement Equipment ................................ 27
      4.3.1 Mounting platform ................................. 28
      4.3.2 Response transducer .............................. 28
      4.3.3 Shaker ............................................ 30
   4.4 Standard masses ........................................ 31
   4.5 Summary ................................................ 32
5 Results
5.1 Introduction ............................................. 33
5.2 Model results ............................................ 33
5.3 Measurement .............................................. 34
5.4 Summary .................................................. 36

6 Conclusion and recommendations 39

A Sections PBARL 41

B The linear medial axis algorithm walk-through 47
B.1 introduction .............................................. 47
B.2 Calculating the medial line ............................... 47
B.3 Search for flange and inner loops. ...................... 47
B.4 Calculating the medial axis points ...................... 49
B.5 Two loop geometries ..................................... 50
B.6 One loop geometry ....................................... 51
B.7 Connecting the separated parts ......................... 54

C Area calculation medial axis method 55

D Approximated Yang algorithm 59

E Linear medial axis algorithm 65
E.1 Medial ....................................................... 65
E.2 Medial line ............................................... 76
E.3 Test 1 ....................................................... 78
E.4 Test 2 ....................................................... 79
E.5 Test 3 ....................................................... 80
E.6 Test 4 ....................................................... 82
E.7 Test 5 ....................................................... 83
E.8 Test 6 ....................................................... 84
Chapter 1

Introduction

Looking at the history of automobiles, it can be seen that in the beginning the automobiles were all cabriolets. Because of practice later on a detachable roof was delivered together with the car. But slowly when cars became reachable for the common people, the cabriolet became less popular as passenger car, although in race competitions the ‘open’ coachwork was till the ’50 still the most used variant. In the 1957 the cabriolet gained some of its popularity back, because Buick introduced the hardtop, as known today, and Ford introduced the legendary linen electric rooftop. From that time cabriolets became more a separate coachwork design.

Development of a cabriolet takes a lot of extra effort contributing to raise the stiffness. Because a cabriolet does not have a stiff roof structure, a lot of stiffness in the middle part of the car is lost. Less stiffness results, in this case, in lower eigenfrequencies. This will subsequently result in less comfort and worse car handling. To overcome these problems extra stiffness has to be added. Most of this extra stiffness is added by the doorsill and the roll-over-bar. The roll-over-bar is usually placed more to the back of the car, and has less contribution in stiffness for the middle part of the car. Therefore the doorsills are from crucial importance for the stiffness in the middle part of a cabriolet.

In this report the first steps toward the successor doorsill design of the BMW 6-serie cabriolet is discussed. The doorsills in the existing model has a very complex structure which is hard to build and therefore expensive. For the successor it is desired to make this structure less complex.

A new prototype is developed and modelled with help of the FEM-package Msc Nastran. To be sure the doorsill is optimal constructed it will be optimized. The optimization is parametrical, and different methods within Msc Nastran are available to transform the model into parameters for optimization. These methods are short discussed in Chapter 2.

The doorsills are modelled into an existing 6 series cabriolet, and simulated. This is done because later on the doorsill will be measured in the laboratory to verify the model results. We are not interested in the results of the doorsill itself, what the influence of the doorsill is on the eigenfrequencies of the coachwork. A short description on how the measurements are done is included in Chapter 4. Details on the new design are company confidential, so no details or exact values of the design are included in the report. Therefore the results of the model and
measurement are given relative to each other and not in absolute measures. The results are included and discussed in Chapter 5. In the last chapter a conclusion is drawn and recommendations are given.
Chapter 2

Cross-section description for optimization

2.1 Introduction

Because a car is very complex, many expertises meet each other by development. In case of the doorsill only the department dynamics and vibration is involved directly. The department dynamics and vibrations has three main tasks:

- Predict dynamical behaviour with help of models.
- Measuring concepts structures.
- Measuring competitor cars.

The new doorsill is developed with help of a FE-model and later on it is optimized within given boundary conditions. In these boundary conditions the minimum requirements are given, which the car must fulfil. These boundary conditions do not only concern dynamical and vibration boundaries, but also vehicle safety boundaries. Setting up these boundary conditions involves other departments indirectly in the design of the new doorsill. If the model results are satisfactorily, the doorsill is build into a coachwork and measured to control the model results.

2.2 Problem definition

Because all stiffness in the middle section of a cabriolet has to be delivered by the doorsills, these are important parts of the structure. The current design of the sill of the BMW 6-serie cabriolet, has a very complex structure. With as consequence that it is not only difficult to produce but also expensive. The doorsill is build from different shells as can be seen in figure 2.1. The length shell exists over the complete length of the doorsill. The width shells are on 7 different locations in the doorsill and are perpendicular to the length shell. The
assignment given is to develop a less complex doorsill with equal or higher eigen-frequencies of the coachwork as the current model. From a previous study it was concluded that it was not possible to construct a doorsill without inner structure within the boundary conditions that fulfil the requirement. These conditions are not only the eigenfrequencies, but also for example mass, dimensions, rollover and crash safety. In this report an attempt is made to design a doorsill without length shells. It is chosen to remove the length shells because it brings less stiffness in proportion to the width shells.

2.3 The cross section representation

In this case a beam-shell model is used for optimization. These types of models have the advantage of faster optimization with satisfactory results compared with FE-models. A beam-shell model is constructed, by transforming a FE model with a special toolkit developed at BMW and implemented into Msc Nastran.

An example of a beam-shell model can be seen in figure 2.2. The model is build up of 'blocks'. A block is a linear representation of the properties and geometry of the corresponding piece of the FEM model. The area of the cross-section and the length perpendicular to this area describes a block. Together with the material properties it is possible to describe all physical properties needed.

In Msc Nastran the cross-section of a block can be represented in three different ways, known as PBAR, PBARL and PBRSECT. In the next subsections these methods are described more in detail. Which routine is chosen for cross-section repre-
Figure 2.2: Beam-shell model of the 6-series cabriolet with the doorsill pictured in grey, from which one block is enlarged.

representation, has influence on the optimization. Because the optimization is parametrical, for every method there are different numbers of parameters to change, which can result in different solutions.

2.3.1 PBAR

The PBAR program converts the cross section to a rectangular profile. This is shown in figure 2.3. The rectangular profile matches the physical properties of the given cross-section as good as possible. In our case that is hard because nearly all cross-section exist of hollow profiles. The only parameter to change during optimization is the area of the cross-section itself. With only the area as parameter all geometrical information is lost. This gives the main problem that the solution of the optimization provides only the optimal parameters. The solution given is parametrically maybe optimal, but contains little information about the optimal solution of the originally profile itself.

2.3.2 PBARL

In case of the PBAR method the cross-section could only be transformed into a rectangular. The PBARL method has a few more choices. The geometry can be translated into 20 different type of profiles which are shown in Appendix A. Therefore the PBARL method is able to represent the geometric properties better, but still in most cases geometric information is lost because the given profiles have to match one of the 20 different types of cross-section. In case of a rectangular pro-
Figure 2.3: PBAR defines a given cross-section into a simple beam element.

Figure 2.4: PBARL defines a given cross-section.

file the cross-section in PBARL is described with four parameters, as can be seen in figure 2.4. Because the optimization is parametric, as mentioned before, it is possible to change more parameters during the optimization. This can result in a more optimal solution.

2.3.3 PBRSECT

The advantage of representing the cross-section in given profiles (PBARL) or in the area (PBAR), is that the optimization itself goes fast. Few parameters can be changed and all physical properties, like the moment of inertia, are easy too calculate. The disadvantage is the lost of geometrical information. By taking the real cross-section, the calculation time will rise, but no geometrical information will be lost.
In the PBRSECT algorithm the cross-section is given as a list of points. By connecting the points in the given order the geometry is represented linear. Representing the cross-section in this way is called general-section. It is also possible to represent this cross-section with the medial axis, which is also possible within PBRSECT.

**General section**

In case of general section the linear cross-section is exactly represented, because it use the same points as the cross-section. No loss in geometrical information occurs. It has the disadvantage of changing only 2 design variables; the height and width, which can be seen in figure 2.5. Because no more parameters can be changed the wall thickness changes proportional with changes in height and width. From the optimized cross-section the wall thickness is therefore unknown and extra effort is needed to find it. Also the wall thickness is uniform, which is in some case not optimal.

**Medial axis**

The other way to describe a cross-section in PBRSECT is with help of the medial axis. If you have a circle which has the following properties:

- It is entirely within the object.
- It is tangent to the object at more than one point.

Then the locus of centres of all such circles together is the medial axis, and each centre point itself is called medial axis point.

For the given cross-section the medial axis method is represented in figure 2.6. This representation has two advantages. One is that the cross-section can be de-
scribed well. As can be seen in figure 2.7 the description is a little worse than the general section method, because with the medial axis information, the corners are not correctly represented. But the real cross-section is still recoverable. The ‘error’ made in the representation of the corners does not have influence on the area of the cross-section, which is proved in Appendix C.

The second advantage is that the thickness of every line is now a design variable. It is possible to tune with more variables, which is especially by minimizing the mass a great advantage. Now the thickness of every line itself can be changed, instead of the complete height or width, which can give better results.

Figure 2.6: PBRSECT medial axis representation.

Figure 2.7: Example of mismatch medial axis method
2.4 Summary

If we line-up all methods for cross-section representation, it can be seen that PBAR and PBARL have the disadvantage of losing geometrical information in nearly all cases, which is not wanted. The optimized solution may be optimal but no information on the real profile is known. The best solution is in this case the PBRSECT method. The PBRSECT method is divided into the general section and the medial axis method. Both can be used, but the preference goes to the medial axis method. In case of the general section the wall thickness goes proportional with changes in length and with. If the cross-section is represented with the medial axis method the wall thickness is known and can be changed by optimization. This has the advantage that the optimized solution can be better as the optimized solution of the general section. However, a disadvantage of the medial axis method is that more effort is needed to represent the real optimized cross-section. Although the medial axis method is the most promising, no such program is available in Msc Nastran. Therefore a program for extracting the medial axis from a given geometry is developed. This is discussed more in detail in Chapter 3.
Chapter 3

Design of the Medial axis algorithm

3.1 Introduction

From the previous chapter it is concluded, that the medial axis method is the best alternative for representing the cross-section area. Because no program exists within Msc Nastran for extracting the medial axis from a given geometry, there is need for such program. Different algorithms are already studied, and the method of Yuandong Yang is found to be the most promising.

3.2 Yang approximately medial axis method

Here only the global idea of the Yang method will be explained. For more details you will be referred to the paper of Yang [2]. The Yang method make use of circles to find medial axis points by fitting an as large as possible circle in a geometry. The advantage of circles is that all intersecting the edge of the circle are at equal distance. If an as large as possible circle, intersecting at more then 1 point within the boundaries of the geometry the centre point of the circle is a medial axis point.

Now it will be explained step by step how the algorithm works. It starts with a point \( m \) within solid \( D \). From this point the distances perpendicular to the boundaries of the geometry are calculated. The minimum distance \( \delta(m) \) from all the distances is taken and a circle with \( N \) points, \( p_1,...,p_n \), is constructed with middle point \( m \) and radius \( \delta(m) \). For each point \( p_i, i = 1\text{ to } n \) the distance perpendicular to the nearest geometry boundary \( \delta(d_i) \) and its direction \( \vec{v}_i \) is calculated. If the direction \( \vec{v}_i \) differs from direction \( \vec{v}_{i+1} \), a medial axis point can be found between point \( p_i \) and \( p_{i+1} \), because a medial axis point has a minimal equal length in at least 2 directions. The approximated medial axis (ama) point is estimated by taking the mean between the positions \( p_i \) and \( p_{i+1} \). Increasing the number of point’s \( N \) will therefore result in a better approximation of the medial axis points. Every ama point is saved together with the mean length of distances. The found ama point with the greatest distance is the new starting point.
The algorithm is also shown in figure 3.1. The starting point is $O_1$. Around point $O_1$ a circle is constructed with $n$ points (not all are pictured). Between the points $p_i$ and $p_{i+1}$ where $\vec{v}_{i+1}$ change of direction a medial axis point can be found. These ama points are for points $(p_1, p_2)$, $(p_3, p_4)$ and $(p_5, p_6)$ respectively $q_1$, $q_2$ and $q_3$. The point $q_3$ has the largest mean distance and becomes the new start point. This repeats till the complete surface of the profile is covered with circles. This algorithm is implemented in a Matlab script, and included in Appendix D.

Testing the script on different cross-sections the algorithm does not fulfil our expectations, because the computation time is too high. There are two main reasons why the algorithm failed: the high accuracy and the cross-sections.

The first reason has to do with the algorithm itself. A circle is build up of $N$ points and the angle $\Delta \theta$ between 2 points is then equal to \( \frac{2\pi}{N} \). The distance between 2 neighbouring points is given as,

$$d = 2r \sin\left(\frac{\Delta \theta}{2}\right) = 2r \sin\left(\frac{\pi}{N}\right)$$

(3.1)

Therefore the maximum error that can occur is equal to,

$$\varepsilon_a = r \sin\left(\frac{\pi}{N}\right)$$

(3.2)

Consequently given a maximum error $\varepsilon_a$ the number of samples needed is given as a function of the radius $r$ of the circle,

$$N = \frac{\pi}{\arcsin\left(\frac{\varepsilon_a}{r}\right)}$$

(3.3)

For an accurate approximation, many samples $N$ are needed. This leads to a high computational time.

The second reason is introduced by the cross-sections. The structure of the vehicle is build from thin plated steal tubes. Taking the cross-section of such tube will result in a profile where the length of the surrounding will be much larger compare with the thickness. Because the Yang methods fits circles inside the geometry,
the radius of these circles are very small, and the maximum distance over which
the starting point can translate every step is the radius. An increasing number of
circles result in an increased computational time.

The example given in figure 3.2a shows a simple rectangular cross-section, with
its medial axis. This cross-section is calculated with the Yang algorithm (figure
3.2b). For this example every circle exists of 83 points, for $\varepsilon_a = 1e-3$ mm and
$r=1.5$ mm. To calculate an ama point $83 \times 4$ calculations have to be done and in total
22 ama points are needed to calculate the medial axis.

As can be seen in figure 3.2c there are only 2 points needed for the exact rep-
resentation of the medial axis line. The reason this is possible is that the given
cross-section is linear. A linear profile has the advantage that the medial axis does
not change the slope, if the profile does not change its slope. It is possible to rep-
resent the medial axis exact and with much less points. The cross-sections given
from Msc Nastran are all linear, which makes it possible to use this advantage. Be-
cause of the high CPU time of the Yang algorithm, a new algorithm is developed
which uses this advantage. The algorithm is described in the next section.

3.3 The linear medial axis algorithm

Using the advantage of linear profiles, a new algorithm is developed. The idea
behind it is to find the same medial axis points as shown in figure 3.2c. Before
explaining further first some terms are introduced.

The profiles at BMW are linear, which is a special type of profile. A profile
is build with closed rings, such ring will be called boundary loop. For example a
cross-section of a tube, exists of 2 boundary loops.

If the number of points in a boundary loop are reduced to a minimum, only
points remain where a boundary loop change of slope. The remaining points will
be called corner points.
Finding the closest medial axis point of every corner point, and connect them in the right way will give the medial axis. That is possible because between corner points the medial axis is also a line for linear profiles.

A medial axis point is found by calculating the medial line. The medial line is the medial axis of a corner point with the two corresponding lines of the boundary loop. As can be seen in figure B.1, there are 2 possibilities for the medial line. With information of the slopes of the 2 boundary lines and the coordinates of the 3 involved points it is possible to calculate the right medial line. The points \(1 - 2 - 3\) will give medial line 1 and the points \(1 - 2 - 1\) and \(3 - 2 - 3\) will give medial line 2. A medial axis point has to lie on a medial line. To find this point, first the intersection point with another medial line is found. The second medial line is found by searching the most nearby useable point. The definition useable is different for one and two boundary loop geometries. Connecting all medial axis points, in the right order will give the medial axis.

Finding the second point, the most nearby useable point, is different for one and two loop boundary geometries. In the case of two-boundary loop geometries, it is the most nearby point of the other boundary loop, except in case of flanges. Flanges deliver problems and therefore are searched and treated separately. The search method implemented works well, if there are not an equal number of flange on each loop.

With one-boundary loop geometries finding the right point is less obvious. It is possible that every point of that boundary loop is the right point. Therefore extra effort has to be made to find this point. To avoid extra calculations, a different algorithm is made for one and two boundary loops.

A geometry can exist of \(n\) boundary loops. Because of the different algorithms, a \(n\) boundary loop geometry is split into parts of one and two-boundary loops. Each part is treated separately and later on connected together.

The algorithm is implemented into a Matlab script, which can be found in Appendix E. A further step-by-step approach of the algorithm is implemented into Appendix B. Running the algorithm results in a much lower computational time compared with the Yang algorithm. Therefore the algorithm is used to describe the cross-section of the blocks.
3.4 Summary

A new algorithm is designed and implemented in Matlab. It is much faster as the Yang algorithm, but only suitable for linear 2-D problems. It still has problems with two loop geometries with even flange on each loop. A new subroutine has to be found to avoid this problem. This algorithm will be used to describe the cross-sections of the beam and shell models in the PBRSECT medial axis method.
Chapter 4

Description of the measurements

4.1 Introduction

To verify the FE-model it is necessary to measure the doorsill. In this section it is shown how these measurements are taken. By obtaining results there are a lot of items that should be taken into account for collecting correct data. A closer look is taken on these items and explained how they will influence the measurements, and how to avoid errors.

Not the results of the doorsill only are wanted but the performance of the doorsill within the coachwork. Implementing the boundary conditions on a doorsill to simulate the doorsill is within the car is very hard and therefore it is build into a coachwork of the existing 6-series cabriolet. It has to be build within an existing coachwork, because the successor is not available yet.

4.2 The Frequency Response Function

The Frequency Response Function (FRF) is needed to obtain the dynamical properties of the structure. The first 5 eigenfrequencies and the corresponding eigenmodes are desired. To obtain these results measurements are done. The problem with measurements is the occurrence of errors/noise. Therefore the FRF has to be estimated, which is described in the next section.

4.2.1 FRF estimation

Measurements are taken to obtain the first 5 eigenfrequencies and corresponding eigenmodes. To obtain these results it is necessary to make a FRF, known as the system description $H(\omega)$. As can be seen in figure 4.1 the FRF can be described as,

$$H(\omega) = \frac{X(\omega)}{F(\omega)}$$ (4.1)
Figure 4.1: System description.

It represents, as function of the frequency $\omega$, the complex ratio between the output $X(\omega)$ and input $F(\omega)$. The function $H(\omega)$ has a magnitude $|H(\omega)|$ and a phase $\angle H(\omega)$. This means physically that a sinusoidal force signal with frequency $\omega_1$ will result in an output displacement with the same frequency and an amplitude that is multiplied with the $|H(\omega_1)|$. The output signal has a phase shift between input and output of $\angle H(\omega_1)$.

The FRF can also be defined as the cross-spectrum of excitation and response divided by the auto-spectrum of the excitation. Therefore it is possible to describe the FRF as:

$$H_1(\omega) = \frac{S_{FX}(\omega)}{S_{FF}(\omega)}$$  \hspace{1cm} (4.2)

Here, $S_{FX}(\omega)$ is the cross-spectrum of the excitation and response and $S_{FF}(\omega)$ is the auto-spectrum of the excitation. With help of spectral analysis it has been found that the FRF also can be estimated from the ratio between the auto-spectrum of the excitation $S_{XX}(\omega)$ and the cross-spectrum of excitation and response $S_{XF}(\omega)$:

$$H_2(\omega) = \frac{S_{XX}(\omega)}{S_{XF}(\omega)}$$  \hspace{1cm} (4.3)

The system represented in figure 4.1 is an ideal representation of reality, and in this case $H(\omega) = H_1(\omega) = H_2(\omega)$. In reality a FRF measurement cannot be free of noise. Figure 4.2 shows the measured force $\hat{F}(\omega)$ which is a combination of the real force $F(\omega)$ and noise from the input $M(\omega)$ and the measured response $\hat{X}(\omega)$ containing the true response $X(\omega)$ and output noise $N(\omega)$. The input and output signal and its corresponding noise signal are normally inseparable in time but they do not correlate with each other, so $S_{MX}(\omega) = 0$ and $S_{NX}(\omega) = 0$. By using the signals $\hat{X}(\omega)$ and $\hat{F}(\omega)$ the FRF can be estimated as:

$$\hat{H}_1(\omega) = \frac{\hat{S}_{XF}(\omega)}{\hat{S}_{FF}(\omega)} = H(\omega) \left( 1 + \frac{S_{MM}(\omega)}{S_{FF}(\omega)} \right)^{-1}$$  \hspace{1cm} (4.4)

$$\hat{H}_2(\omega) = \frac{\hat{S}_{XX}(\omega)}{\hat{S}_{XF}(\omega)} = H(\omega) \left( 1 + \frac{S_{NN}(\omega)}{S_{XX}(\omega)} \right)$$  \hspace{1cm} (4.5)

Neither of both estimators accounts for an accurate FRF. More insight can be found if $\hat{H}_1(\omega)$ and $\hat{H}_2(\omega)$ are examined around resonances and anti-resonances. Here the most dramatically changes in signal to noise ratio occur.

At a resonance, where input forces decline dramatically because of impedance mismatch between the structure and the input source, it allows the measurement noise to dominate the input end, while high level of responses on the outside ensure a
large signal to noise ratio. This results in a significant $S_{MM}(\omega)$ compared with $S_{FF}(\omega)$ and a $S_{NN}(\omega)$ of no importance compared to $S_{XX}(\omega)$. Consequently, the FRF estimator $\hat{H}_1(\omega)$ underestimates the true FRF and $\hat{H}_2(\omega)$ estimates accurately. At an anti-resonance, the response from the structure usually is insignificant compared with the force input, which allows measurement noise to dominate the output, while a high level of force produces a large signal to noise ratio at the input side, resulting in a significant $S_{NN}(\omega)$ compared with $S_{XX}(\omega)$ and a small $S_{MM}(\omega)$. In case of an anti-resonance $H_2(\omega)$ over-estimates the true FRF and $H_1(\omega)$ estimates it accurately.

### 4.2.2 Frequency limitations

The maximum frequency of the FRF is limited by the maximum frequency of the input signal or it is limited by the sampling frequency. So the input signal should hold for all the frequencies, ($-\infty < \omega < \infty$) and the sampling frequency should be $\infty$. The input signal is limited by the dynamical properties of the input system, in this case an shaker.

The maximum frequency can also be limited by the sampling frequency. For measuring a sinusoid there are minimally 2 point per period necessary, as shown in figure 4.3. So if fewer points are taken from a sinusoid, this results in a sinusoid of lower frequency. This phenomenon is called aliasing. The maximal frequency without aliasing is also known as Nyquist frequency:

$$f_{Nyq} \leq \frac{f_{sample}}{2} \quad (4.6)$$

Aliasing is not the only item that should be taken into account. The other is signal leaking. Signal leakage is caused by the fact that by taking a sample a signal is truncated before damped out. For Fourier analyses that sample will be copied many times and connected to each other. If the start value and end value of a sample are not the same these are connected to each other which introduces extra frequencies.
Figure 4.3: Limitation by sampling frequency.

Figure 4.4: Example of wrong time interval causing signal leaking.
4.2.3 The coherence function

The frequency domain model is a linear model. The coherence function gives the degree of linearity between the input and the output signals. The coherence function is defined as:

\begin{equation}
\gamma(\omega)^2 \equiv \frac{|S_{FX}(\omega)|^2}{S_{XX}(\omega) \cdot S_{FF}(\omega)}
\end{equation}

Where,

\begin{equation}
0 \leq \gamma(\omega)^2 \leq 1
\end{equation}

The bounds for the coherence function are 0 and 1 for respectively pure noise in the measurement and no noise in the measurement. The coherence function is used to indicate the quality of the measurement.

4.3 Measurement Equipment

To make a correct FRF estimation it is important to collect and measure the right signals as accurate as possible. Now we pay some attention to the placement of equipment and the equipment itself. The set-up of the measurements, as taken by BMW, is shown in figure 4.5.

![Figure 4.5: Schematic representation of the measurement set-up.](image)

27
4.3.1 Mounting platform

The measurements done at BMW can be divided roughly into 3 different groups: complete cars, coachworks, and components. For measuring the doorsill complete cars and components are not of interest and will not be treated here.

Setting up a measurement, a coachwork is placed on top of 4 standards. On top of each standard a rubber ball is fixed, accordingly not stiff connected to the ground is made.

In this case ground is relative, because the complete measurement place, see figure 4.6 is mounted on a platform that is supported by dampers and springs. This is done to minimize the amount of ground resonance, which can be seen as input noise in the measurements. The coachwork is placed on top the measuring place.

![Figure 4.6: Schematic representation of the measurement place.](image)

This top is on a equal level compared with the floor. For mounting stiffness bars, mounting the shaker, and placing accelerometers more easier beneath the car, a well is constructed inside the measurement place. Mounting stiffness bars is still done on feeling, but it is preferable to mount these each time with the same force with a moment spanner. In this matter no error can be introduced by different mounting moments, and exclude errors from 'loose' contacts, due to not tighten nuts.

4.3.2 Response transducer

As response transducers accelerometers are used, because of the many advantages:

- Good linearity
- Low weight
- Wide frequency range
- High environmental resistance
- Low transverse sensitivity
- Simple mounting methods
The mounting of the accelerometer is very important. A weak connection between transducer and structure will result in an acceleration output of the transducer itself instead of the structure. The best solution is to use a threaded steel stud, which in this case not usable because it would damage the vehicle. Instead of a threaded steel stud, a thin layer of beeswax is used. This gives also good results, although lowers the useful frequency of the accelerometer. Lowering the useful frequency causes no problems because the frequency lowers till approximately 29 KHz, as can be seen in figure 4.7, and the modes of interest are in the frequency range 15 till 50 Hz.

By clean surfaces beeswax is useable for fixing accelerometers till acceleration levels of about $100 \text{ m/s}^2$ on the condition that the maximum temperature is below 40°C. Above this temperature beeswax becomes soft and will have negative results on the measurement.

It is necessary to obtain scaled mode shapes by modal testing. To provide this a driving-point measurement is needed. Therefore a transducer should be mounted nearby the point of excitation or on the opposite site of the structure. By mounting the transducer the structural loading must be taken into consideration. Loading a structure may alter the mass, damping or stiffness. The dynamic mass loading produced by the transducer is depended on the local structural dynamic properties. To avoid these problems the accelerometers are mounted on stiff parts of the coachwork or on the supported structure. Because of the high stiffness the influence of the dynamical mass of the transducer is minimized.

The standard measurement contains 28 measuring points, as can be seen in figure 4.8. On each measuring point a tri axial accelerometers is placed. Still the positions of the sensor locations are equal for all models in standard measurements. The positions are determined with help of experience. With a method described in Kraker [1] it is possible to estimate the best $n$ sensor positions out of the Mass and Stiffness matrix. First the rotational, internal and inaccessible degrees of freedom (dof) are removed, giving $m$ remaining dof’s. Then the $\varepsilon$ relevant eigenmodes are selected, giving the sub matrix of selected modes and remaining dof’s $U_{me}$. With this information the so-called Fisher information Matrix $F_{ee}$ can be calculated:

$$F_{ee} = U_{me}^T U_{me}$$  \hspace{1cm} (4.9)
Knowing $F_{ee}$ the matrix $G_{mm}$ can be formed:

$$G_{mm} = U_{me} F_{ee}^{-1} U_{me}^T$$  \hspace{1cm} (4.10)

The diagonal terms of the matrix $G_{mm}$ represent the partial contribution of each measurement dof to the rank of $G_{mm}$, and hence to the linear independence of the $\varepsilon$ chosen modes.

### 4.3.3 Shaker

For excitation an electric hydraulic shaker is used. The input force is measured with a piezoelectric force transducer. Because these transducers only measure in one direction, also the input force should work in one direction. To achieve this a thin rod is used. This has a high stiffness in longitudinal direction comparing with the other directions. Therefore the force will only be guided in one direction. If forces work in other directions the rod will bend. In figure 4.9 a schematic representation is given.

The rod provides forces only in one degree of freedom (dof). In other direction the structure is free to move. Studs with weak rubbers on top provide this. Therefore the noise introduced into the structure caused by the shaker is minimized. To guide the force without losses into the structure, a stiff connection is necessary between the rod and the structure. To provide it a plate is glued on the structure where the rod and force transducer can then be screwed on.
4.4 Standard masses

If the coachwork is measured, some masses (SMB) are added. This is done to make the difference bigger between the local modes and the global modes. The disadvantage is that the global modes become less stable. The eigenfrequencies is proportional with the root of the stiffness divided by the mass:

$$\omega_0 = \sqrt{\frac{k}{m}}$$  \hspace{1cm} (4.11)

Placing a solid metal weight will increase the mass and stiffness disproportionally. The mass will be very stiff, but does not contribute much to the stiffness of the car, with results in lower eigenfrequencies.

For every coachwork the same masses, on given positions, are used. These are pictured in figure 4.10. As can be seen the total mass is 150 Kg. This is approximately 40% extra weight in case of an coachwork, which is not insignificant. But these masses are not placed every time identical. The misplacement of the masses
is not equal for every position, and in every direction. The masses 1 and 4 have only an error in the y direction. The misplacement of these masses can be maximal around 2 cm. There is no misplacement in the x direction, because the masses are placed against the inner structure.

Sometimes the masses 2 are especially made for the car. Normally the chair is mounted on this place, and the same holes are used for screwing the mass plate into place and no mismatch will occur. If no special masses are available, universal masses are used and they are glued. By gluing the error made is approximately maximal 1.5 cm in x and y direction.

The biggest error arise by placing the masses 3. An error of maximum 6 cm can be made in the y-direction and no error in x-direction because the mass is placed against a shell in x-direction.

Because we are dealing with insignificant extra masses, and errors in placement it is useful to investigate the influence of the misplacement of the masses. Improvements in the design usually gives a difference in frequency in the order of 0.1-1 Hz. A possible method to investigate the problem is the adjoin method. This method is described in Kraker [1]. This method is based on the differentiation of the eigenvalue problem. The eigenvalue problem is differentiated with respect to influencing parameter. In this case it is the extra mass edit by the smb’s. The masses are edited on the diagonal of the mass matrix by the appending dof. Adding a weight distribution on the extra masses can simulate shifting the masses.

4.5 Summary

The measurements are done according a common FRF measurement. The input is provided by an electro hydraulic shaker which a piezoelectric force transducer on top. This force transducer only measures the input force in one direction. To make sure that the force is only working in one direction, a thin rod is placed between the shaker and the piezoelectric force transducer. A thin rod provides much more stiffness in longitudinal direction then in the other directions. The accelerations are measured with accelerometers, which are mounted with beeswax. The beeswax provides a stiff connection of the accelerometer, in the frequency range of interesse, preventing noise in the output signal. Also the mounting platform provides noise in the output. The mounting platform is fix with a dempers and springs, which are configured to reduce environmental noise as much as possible. In the mounting platform a well a well is constructed, to reach the bottom of the car easier. For measurements of coach-works stiffness bars are mounted, because they have a significant contribution. Therefor it is preferable to mount the bars with a known moment. In this case it is sure that all bars are stiff connected, and every time with the same forces. Also the influence of the SMB’s have to be investigated. There are made big errors by placing the masses, but it is unknown what the influence of these misplacement is on the first eigenfrequencies. A method is included to investigate the influence analytical. It is important to investigate both recommendations, because changes in concept are in the order of 0.1-1 Hz. If the influence of the misplacement of the SMB’s and the difference of the fixing moment of the stiffness bars are also in the order of 0.1-1 Hz no conclusions can be drawn on the measurements.
Chapter 5

Results

5.1 Introduction

In the previous chapters, the medial axis algorithm is explained and the parts that are important for measuring. In this chapter the results from the model and measurements are shown. With help of the medial axis algorithm the medial axis of the cross-sections in the doorsills are calculated. Then the doorsill is optimized. The optimized doorsill is hand build in the workshop and welded in the place of the old doorsill. Finally the coachwork is measured to look if the predictions of the model were right.

5.2 Model results

Before building the new doorsill first models are made and calculated. In this section the results, as can be seen in table 5.2, of the models are discussed. The developments of the doorsill have gone in a few steps, which will be explained here.

Because the doorsill has to be measured, it will be build into an existing coachwork of a BMW 6 series cabriolet. Therefore a reference measurement is needed, to check whether no damage or failure is in the coachwork. The results of the model with the existing doorsill are also implemented into table 5.2. The difference between the model and the reference measurement is -0.4 Hz for the first torsion and 0.6 Hz for the first bending.

The next step is to implement the new doorsill. This is the doorsill without the length shell. As expected the results of this model are worse then the reference. As can be seen in table 5.2 the frequency of the first torsion mode drops with 0.7 Hz and the first bending mode even with 1.0 Hz compared with the reference measurement. This means that the first torsion differs -0.3 Hz and the first bending even with 1.6 Hz compared with the reference model.

Because the eigenfrequencies for the doorsills are lower as the reference, it does not fulfil the target. To meet the target the doorsill is optimized. For optimization not only dynamical constraints are given for the doorsills but also constraints for statical load, crash safety and rollover have to be implemented. The optimization
Table 5.2: Results model.

<table>
<thead>
<tr>
<th>Convert</th>
<th>Δ first torsion mode</th>
<th>Δ first bending mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reference</td>
<td>-0.4</td>
<td>0.6</td>
</tr>
<tr>
<td>Doorsill without length Shell</td>
<td>-0.7</td>
<td>-1.0</td>
</tr>
<tr>
<td>Optimized doorsill</td>
<td>1</td>
<td>1.9</td>
</tr>
</tbody>
</table>

is mass oriented, which means that the mass should be as low as possible without violating constraints. As can be seen in figure 5.1, after the 5th cycle the doorsill fulfil the target values. In this case the doorsill its height is grown with 80 mm and it’s width with 30 mm. As can be seen in figure 5.1, on iteration cycle 5 the first torsion mode is increased with 1.5 Hz and the first bending mode even with 2.5 Hz. With these results the new doorsill design fulfils the requirements.

![Figure 5.1: Frequency shift of the first torsion and first bending as function of the iteration step, together with the target values.](image)

5.3 Measurement

To verify the model results the doorsill is build in steps into an existing 6 series cabriolet. First a reference measurement is taken, which is necessary to check if the first eigenfrequencies are equal to previous measured 6 series cabriolets. If large differences occur, the coachwork can be damaged. The reference measurement is also done to see how much the new parts will really bring. Placing the smb’s and components like stiffness bars will all influence the values of the eigenfrequencies. Comparing the results of the new doorsill with the reference measurement instead of taking old measurement data, will exclude the misplacement of the SMB’s and give the results of the new doorsill more accurate. In figure 5.2 the reference mea-
surement of the coachwork is shown. Here the placement of the accelerometers and the fine measurement of the doorsill can be seen. After the reference measurement, the coachwork is send to the workshop to replace the doorsill. The doorsill is measured in more steps then the model. The doorsill without length shell is measured in two different steps, without and with width shells welded into place. This is respectively convert 1 and convert 2 as can be seen in figure 5.3. The reason to do it in steps is to gain extra information over the contribution of the width shells in the stiffness.

From the models it is concluded that the doorsill was not stiff enough, and is optimized to meet the targets. As a result the doorsill became 80 mm higher and 30 mm wider. The optimized doorsill is also measured in steps. First an U-profile is welded beneath the doorsill. The U-profile have a height of 80 mm and a wall thickness of 3 mm. The doorsill with the U-profile welded beneath is named convert 3.

After convert 3 is measured an extra u-profile is welded to the side of the doorsill. The U-profile has a height of 30 mm and a wall thickness of 3 mm. The doorsill with this profile is named convert 4. The results of the measurements are shown in table 5.3. Only the relative difference compared with the reference measurement is shown. The exact values of the eigenfrequencies are company confidential and it is not allowed to publish them in this report. As can be seen the first convert already gives some improvement in comparison with the reference measurements. The first torsion mode lies 0.3 Hz higher as before. If the shells are welded into place, which is convert 2, it can be seen that all the eigenfrequencies raise except the first torsion mode, which stays equal. The first bending mode raised with 0.1 Hz and the second torsion mode with 0.4 Hz. In comparison with the model, see table 5.4, the first torsion and bending mode are both higher. This can be explained that the
doorsill which is build in has more like a box profile that is much stiffer. With the extra U-profile welded under the doorsill the first torsion and bending mode are even higher, and also the second torsion mode is higher as the reference. On the other hand the second torsion mode of convert 3 lowers from 0.4 till 0.1 Hz. This is even stronger by convert 4. The first torsion and bending mode improves, but the second torsion mode is even lower as the reference measurement. If convert 4 is compared with the results of the measurement as shown in table 5.4 the difference between model and measurement for the first torsion mode is only 0.1 Hz which is rather good. On the other hand the difference for the first bending mode is much higher, namely 1.3 Hz.

### 5.4 Summary

From the measurements it can be concluded that the optimized doorsill did not fulfil the targets. The first bending and torsion mode are increased significantly, but the second torsion mode dropped with 0.1 Hz. Hereby the box profile shape has to be taken in account, which delivers extra stiffness. By welding the U-profiles against the doorsill, two extra shells are introduced, which are not in the modelled
doorsill. These shells can drop the mass to stiffness ratio, which lowers the eigenfrequencies. It can be seen that the doorsill in state convert 2, before optimization, fulfilled the target by the measurements. This can be caused by the fact that the doorsill, which is built in, is a box-like profile with is much stiffer as the modelled doorsill.

Table 5.4: Comparison measurement and model data.

<table>
<thead>
<tr>
<th>Convert</th>
<th>Δ 1st torsion mode model [Hz]</th>
<th>Δ 1st torsion mode measurements [Hz]</th>
<th>Δ 1st bending mode model [Hz]</th>
<th>Δ 1st bending mode measurements [Hz]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reference</td>
<td>-0.4</td>
<td>-</td>
<td>0.6</td>
<td>-</td>
</tr>
<tr>
<td>Convert 2</td>
<td>-0.7</td>
<td>0.3</td>
<td>-1.0</td>
<td>0.1</td>
</tr>
<tr>
<td>Convert 4</td>
<td>1</td>
<td>1.1</td>
<td>2</td>
<td>3.3</td>
</tr>
</tbody>
</table>
Chapter 6

Conclusion and recommendations

For the first step toward a successor for the BMW 6 series cabriolet a prototype for a new doorsill is developed. The existing doorsill was very complex to build and therefore expensive to produce. For the new doorsill the target was to design a doorsill with a less complex structure that has the first 5 eigenfrequencies of the coachwork equal or even higher. The idea was to investigate how much it will bring to optimize the existing doorsill. For the optimization the PBRSECT medial axis representation was needed.

Some investigation was already done for medial axis algorithms. The Yang method looks the most promising and was implemented in Matlab. But the algorithm didn’t perform as well as expected. Because of the large difference between the proportions of the length and width, and the high accuracy it took very long to calculate the medial axis of a given cross-section.

A new algorithm was designed, that is suited for linear profiles. It only calculates the medial axis points of corner points. For linear profiles the medial axis between corner points is a line. The new algorithm is much faster, but has a problem if there is a flange in each loop. In this case the medial axis is not always correct in the neighbourhood of the flange.

With help of the medial axis algorithm the optimization was started. It was possible to find an optimal solution within the given boundary conditions. As a reference the doorsill is build into an existing 6 series cabriolet. This is done because it is very hard to simulate all boundary conditions of the doorsill for measurement. Not the results of the doorsill itself are of importance but the result of the coachwork with the doorsill itself.

The measurement is a standard set-up to make a modal analysis, except of putting extra weight to the coachwork. These so called smb’s are used to split global and local modes and predict the frequencies of a complete vehicle. But placing the smb’s is not as accurate as wanted. A shift of 3-4 cm is not abnormal. But what is the effect of that on the eigenfrequencies? It is recommended to start an investigation on the influence of the misplacement of the smb’s.
From the results of the measurements it can be concluded that the new doorsill design does not satisfy the targets. The first torsion and bending mode are higher as the reference but the second bending mode was lower. It can be seen that the measurement of convert 2 and 3 fulfil the target. Comparing the results of the model and measurement there are large differences. Most of it can be ascribed to the fact that the build doorsill looks more like a box profile. A box profile is a stiffer structure compared to the optimized geometry. Also by editing the U-profiles a difference compared with the model is made. Welding the profiles beneath and on the side, will deliver two extra shells, which are not included in the model. It is possible that more mass is added compared to stiffness. Also due to this fact convert 4 has not fulfilled the target. To investigate the influence of the box profile, and the extra shells, these should be implemented in the model and compared with the measurements.
Appendix A

Sections PBARL
Appendix B

The linear medial axis algorithm walk-through

B.1 Introduction

This appendix treats the linear medial axis program more in detail. The main idea of the algorithm is discussed, and the difference for the algorithm between two and one boundary loops is explained. The working of the program is further more explained with help of an example.

B.2 Calculating the medial line

To construct the medial axis points, the medial line of two lines is calculated. The medial line of the lines, A and B, is the line build from the points where the perpendiculars of the lines A and B have the same distance. In case of linear lines the medial line is half the enclosed angle. But with only the information of the slopes and y-intercept points there are two different medial lines possible, as can be seen in figure B.1.

The medial line is calculated by taking the information of the positions of points 1 and 3 and the angles the corresponding lines make. The script for this loop can be found in appendix E.2. Problems occur by vertical lines, because they have a slope of infinity. Instead of an slope of infinity this slope is set to $1 \cdot 10^8$. The error in angle that is made is $\varepsilon_{angle} = 5.73 \cdot 10^{-7}$ degree, with is within the error limits.

B.3 Search for flange and inner loops.

A medial axis point can be found by calculating the intersection point of medial lines from point $w_1$ and the point $w_2$. Finding the point $w_2$ is different for geometries with one loop and geometries with two loops. The easiest is to find point $w_2$ in a geometry with two loops because point $w_2$ has to be the most nearby point on the other loop, except for the flange, which will be treated later. For the one loop
geometry it is less obviously to find the right \( w_2 \) point because it could be each point in the neighbourhood. It is therefore necessary to design different algorithms to calculate the medial axis points for one-loop geometries.

But the program should also treat geometries existing out of more than one or two boundary loops. Therefore it makes use of the fact that a geometry with \( n \) loops can be separated into parts existing of one or two boundary loops. This algorithm is implemented before the search for medial axis points starts. The algorithm separates the geometry, find the medial axis points for every geometry part, and reunites all geometry parts together into the original geometry, connecting the medial axis lines.

How the algorithms work is explained with help of an example, see figure B.2. The first part of the program splits the geometry into parts of one or two boundary loops and the second part calculates the medial axis points of each part. Also the problems with flange are explained. As figure B.2 shows, this geometry exists out
of 3 loops. The first part of the program splits the geometry into two different geometries as can be seen in figure B.3. But the geometries existing of the filled line in figure B.3 still gives problems in finding medial axis points. The geometry existing of two loops has on the upper corners left and right flanges. Because this geometry exist of two loops from a starting point it can never find the right point on the other loop to estimate an medial axis point for that neighbourhood. Therefore it is needed to find the flange in geometry and separate these and threat them as one-loop geometries. Flanges give only problems by two loop geometries and are found by taking the absolute sum of every corner of a loop. This can be done because the geometries taken into account are builded up of plates. So no big differences in thickness are found. If there is a flange in the geometry the difference has to be bigger then 180 degrees. Is this not the case then there is no flange in the geometry. In figure B.4 the geometry can be seen as it comes out of the program. The geometry is separated into 3 different loops. The criteria for finding flange does not work always correct. When there are equal numbers of flanges in one loop. No big difference in the sum of angles is found, this can be seen in figure B.5. To avoid these problems another algorithm should be developed to find flanges. For the application at BMW this is not a problem, because these profiles are very rare. If the application is used on other profiles, this has to be taken into account.

B.4 Calculating the medial axis points

In most cases a medial axis point can be found at an intersection point of two medial lines. Because we are dealing with linear profiles with nearly always the same thickness. Therefore the algorithm searches the intersection point of two medial lines, if this intersection point exist. It is also possible that the intersection point is not always a medial axis point, see figure B.6, therefor some checks has to be made. Now a short description is given of the steps that will be taken to come to a medial axis point. In the first part it will be done for a two-loop geometry and after that the same is done for the one-loop geometries.
B.5 Two loop geometries

The first step taken is to determine which boundary is the longest. Because representing a corner is sometimes done in different number of points in each boundary loop. See for example figure B.6. The longest boundary loop will be called $L_1$ and the other $L_2$. The program starts with the first element of the list $L_1$, and goes on taking every element of this list.

Starting with element $k$ the most nearby point of the list $L_2$ is found by calculating the circle with smallest radius with middle point $(x_k, y_k)$,

$$distance = \sqrt{(x_n - x_k)^2 + (y_n - y_k)^2}$$ (B.1)

for $n = 1, 2, 3... \text{length}(L_2)$, the minimum position $h$ is found. The point on position $h$ in the list $L_2$ will be the second point. All the second points are listed in a vector called $w\text{-}loop\text{-}2$, to check later if all points in the list $L_2$ are used.

First the neighbouring points are determined (This is different in case of element number 1 and the last elements of every list). Knowing the numbers of the point. The slopes between the points of loop $L_1$ and $L_2$ can be found in a list. With help of these lists the medial lines are calculated, see figure B.1 as example.

Figure B.5: Example of geometry where flanges will be undetected.
Figure B.6: Example of geometry where intersection of medial lines does not deliver a medial axis point.

The first check is if the intersection point of the two medial lines is in the geometry. Is this not the case a new intersection point is calculated. This point is found by calculating the intersection point of the medial line of point \( k \) and the two lines corresponding to the second point. The lengths are calculated, and the minimum length is saved together with the coordinates of half the length, in the matrix called medial-axis.

If the intersection of the medial lines is in the geometry the distance of the intersection point and the geometry is checked. This is done by Test 1. The script can be found in appendix E.3. If it is a medial axis point distances have to be the same. If this is not the case, again the distance between the medial line and the two lines of the second point are calculated, and the minimum is saved together with the coordinates of half the distance.

If for all points in list \( L1 \) medial axis points are found, the program looks if points in loop \( L2 \) aren’t used. Is this the case than the length of the list \( w\text{-loop-2} \) is not equal to zero. The program reruns the algorithm with the points of \( L2 \) that are not used. These points will later on be fit in the right place in the medial-axis matrix.

B.6 One loop geometry

The problem by one-loop geometries is to find the right second point. This is not always obvious. Therefore a special algorithm is written.

The first part is nearly the same as for a two-loop geometry. First the most nearby point is searched with the same method as described in formula B.1. The neighbouring points are included and with the lines between them the medial lines are calculated. Then it’s checked whether the intersection point found is in the geometry.

Because of the many different possible roots to calculate a medial axis point, the program is split into different subroutines. These subroutines are called, Test \( i \) where \( i = 1...6 \). The routine Test 1 will not be treated here, because it is a part of the two-boundary loop algorithm. How the program is divided into subroutines
is shown in figure B.7. In the next part, a short description is given what every subroutine is doing. All the scripts of the subroutines can be found in appendix E.

**Subroutine Test 2**

The subroutine Test 2 search a new second point and calculates the new medial line. A new second point is found with help of the circle criterion, as shown in formula B.1. When this is known, a new intersection point is calculated, and it is checked if the intersection point is in the geometry. This is checked with help of the Matlab commando `inpolygon`. The input is a point, and a vector with the coordinates of the geometry. The function `inpolygon` gives the parameters `in` and `on` as output. The value of `in` can be 0 or 1, for respectively outside the geometry or not outside the geometry. The value for `on` can also be 1 and 0, for respectively on the boundary of the geometry and not on the boundary. In the case that `in = 1` and `on = 0` a point inside the geometry is found.

**Subroutine Test 3**

the algorithm Test 3 first checks if the medial lines are not the same. This is done by the following equation.

\[
p_{\text{vector}} - p_{\text{vector}^2} \leq \varepsilon_{\text{vector}}, \quad \frac{|b - b_2|}{abs(b)} \leq \varepsilon_b \tag{B.2}
\]

Here \(p_{\text{vector}}\) and \(b\) are coefficients of the line, given as \(y = p_{\text{vector}}x + b\). The index 2 refers to the medial line of the second point. If formula B.2 is true for the given
ε_vector and ε_b, the medial line is the mean of point k and the second point. The standard value for ε_vector and ε_b are 1 \cdot 10^{-1}. If the condition is not fulfilled, the algorithm searches for the most nearby points by searching all points in a range [Δ r_x, Δ r_y]. The values Δ r_x, Δ r_y are calculated by finding the largest step size in respectively x and y direction. For all points in the area of x_k±Δ r_x and y_k±Δ r_y the length, between the start point and the point where the medial line of the starting point intersect the line of the geometry, is calculated. The founded lengths are sorted. Starting with the point with the least length, it is checked if the distance to all lines are equal. If this is the case, it’s a medial axis point. If this is not the case, the next point is chosen, till the condition is fulfilled. When the condition of equal length is not fulfilled in the end, the mean between the point k and the second point is taken.

**Subroutine Test 4**

Subroutine Test 4 calculates the distances between the intersection point of the two medial lines, and the 4 lines which are needed to construct them, c_i for i = 1, 2, 3, 4 shown in figure B.8. The second point is represented in this figure by the parameter h. To find the distance between the intersection point and the perpendicular of the lines c_i first the coefficients of the line are calculated. The coordinates of the intersection point have indices start.

\[
a_{\text{perp},i} = -\frac{1}{a_i} \quad (B.3) \\
b_{\text{perp},i} = y_{\text{start}} - a_{\text{perp},i} \cdot x_{\text{start}} \quad (B.4)
\]

The intersection point between the lines c_i and a_{\text{perp},i}x + b_{\text{perp},i} can be found by solving the equation:

\[
c_i = a_i x + b_i = a_{\text{perp},i} \cdot x + b_{\text{perp},i} \quad (B.6)
\]

\[
x_{\text{int},i} = \frac{b_i - b_{\text{perp},i}}{a_{\text{perp},i} - a_i} \quad (B.7)
\]

\[
y_{\text{int},i} = a_i \cdot x_{\text{int},i} + b_i \quad (B.8)
\]

When the point int, i is known the distance can be calculated,

\[
\text{length}_i = \sqrt{(x_{\text{int},i} - x_{\text{start}})^2 + (y_{\text{int},i} - y_{\text{start}})^2} \quad (B.10)
\]

The value as is given as output of the subroutine and is 20% of the maximum of the lengths length_i. The output is used to filter points as shown in figure B.6.

**Subroutine Test 5**

The subroutine Test 5 checks only the difference in length. The lengths are calculated with help of the formulas shown in the section Test 4. If the difference is smaller as \(1 \cdot 10^{-3}\) a medial axis point is found. The point is saved in the matrix medial-axis together with the thickness. If this criterion is not satisfied the mean of point k and the second point is taken and the data is saved in medial-axis.
Subroutine Test 6

The subroutine Test 6 does nearly the same as Test 3. Only the check for the same direction vectors is not made. Because it is not necessary in that stage, but to keep the algorithm clear it is programmed as a subroutine.

B.7 Connecting the separated parts

If the geometry is separated into parts of 1 and 2 boundary loops, for each part a medial axis is calculated. At the end all the parts are joined together, and the medial axis are connected. The place where the medal axis has to be connected is found by elongation of the medial axis of 1-loop geometries. If these line intersect another medial axis within the distance of two times the thickness of the profile these medial axis are connected.
Appendix C

Area calculation medial axis method

Figure C.1: representation of the area representation of the medial axis method.

In figure C.1 is the connection illustrated between two profile parts in a corner. Here the proof is given that surface $A_1 = A_2$. The surface of a quadrangle, as pictured in figure C.2, is given by:

\[ O = \sqrt{(s - a)(s - b)(s - c)(s - d) - abcd \cos^2 \frac{1}{2}(A + C)} \]  

(C.1)

with,

\[ s = \frac{1}{2}(a + b + c + d) \]  

(C.2)
Every profile line has an angle $\alpha$ as can be seen in figure C.1. In medial axis representation $\alpha_1 = \alpha_2$ and $\alpha_3 = \alpha_4$. This because the profile is build from two parallel lines with the medial axis at a constant distance. These distances are equal, so $d_1 = d_4$ and $d_2 = d_3$. In figure C.3 a zoom in on the corner is given. With a green dot the angle $\phi$ is given. The value for $\phi$ changes by different $\alpha_1$ and $\alpha_3$ of the lines. If the surfaces $A_1$ and $A_2$ are the same, then $k_2 = k_4$ and $j_1 = j_3$. The complete length $k = k_1 + k_2$ and $j = j_1 + j_4$ is equal to:

$$k = \tan(\phi) \cdot d_3$$  \hspace{1cm} (C.3)
$$j = \tan(\phi) \cdot d_1$$  \hspace{1cm} (C.4)

From geometrical point of view the next equation holds:

$$\frac{j_1}{k_1} = \frac{k_2}{j_2}$$  \hspace{1cm} (C.5)

Substituting equations C.3 and C.4 into equation C.5 gives:

$$\frac{\tan(\phi) \cdot d_1 - j_2}{\tan(\phi) \cdot d_3 - k_2} = \frac{k_2}{j_2}$$  \hspace{1cm} (C.6)
The triangle $k_2, j_2$ gives a relation between $k_2$ and $j_2$:

$$\cos(\phi) = \frac{k_2}{j_2} \quad (C.7)$$

substituting this into equation C.6 gives:

$$\frac{\tan(\phi) \cdot d_1 - \frac{k_2}{\cos(\phi)}}{\tan(\phi) \cdot d_3 - k_2} = \frac{k_2}{\cos(\phi)} \quad (C.8)$$

Which give as result for $k_2$ as function of $\phi, d_1$ and $d_3$:

$$k_2 = \frac{-\tan(\phi) \cos(\phi)(-\cos(\phi) \cdot d_3 + d_1)}{\cos(\phi)^2 - 1} \quad (C.9)$$

The same expression can be found for the line $k_4$. For all angles the length $k_2 = k_4$. Then the length $j_1$ has to be equal to $j_4$. This can be easily shown by drawing a line from the intersection point $j_1, k_2$ and the intersection point of $j_4, k_4$. This is shown in figure C.4. If all lines and corners are filled in formula C.1 it can be seen that all corresponding input parameters are equal. Herby it is proved that the surfaces described by the linear medial axis method is equal to the real surface of the given cross-section.

![Figure C.4: Proof $j_1 = j_4$.](image)
Appendix D

Approximated Yang algorithm

```matlab
function [MA]=medial_axis_4(points,error_max,startpunten)

warning off
format short g;

clear all, clc, close all
MA=[0 0];

error_max = 0.05;
startpunten = [2] %[2, 4, 8, 9, 10 ];

punt_start = 2;
%punt_start = [26, 40];
%punt_end = [26, 40];
%punt_end = [26, 40];

delta_x = 1;
delta_y = 1;

figure(1);
plot( points(:,1) , points(:,2) )
%
% aanmaken lopende variable
zoek_root_matrix = [];
loop = 0;
zoek = 0;
contras = 0;
round = 0;
st = 0;
tst = 0;
telt = 1;
number = 1;

for i = 1;
zoek_root_matrix = [];

k2_lijst = [];
r_lijst = [];
af_lijst = [];

zoek_richting_matrix= [];

for z = 1:w-1
if (punt1(1) ~= points(d ,1)) | (punt1(2) ~= points(d,2))
   p(c).point = polyfit( points(k:k+1,1) ,points(k:k+1,2), 1);
y(c).lijn = p(c).point(1)*x+p(c).point(2);
y(c).a = -1/p(c).point(1);
y(c).point(1) = p(c).point(1);

if abs(p(c).point(1)) > 1e8
   y(c).a = 0;
y(c).point(1) = 1e8;
end
end
end
```

59
y(c).point(2) = points(k,2) + sign(p(c).point(1))*1e8*points(k,1);
y(c).lijn = p(c).point(1)*x+p(c).point(2);
end
if abs(p(c).point(1)) < 1e-8
y(c).a = 1e8;
end
matrix = [ matrix ; points(c,:)];
punten_lijst = [ Punten_lijst , k];
k = k+1;
d = d+1;
c = c+1;
else
if d < w
punt = points(k,:);
start = start + 1;
p(c).point = polyfit( points(k:k+1,1) , points(k:k+1,2), 1);
y(c).lijn = p(c).point(1)*x+p(c).point(2);
y(c).a = -1/p(c).point(1);
y(c).point(1) = p(c).point(1);
andere_kring = [andere_kring , k];
if abs(p(c).point(1)) > 1e8
y(c).a = 0;
y(c).point(1) = 1e8;
y(c).point(2) = points(k,2) + sign(p(c).point(1))*1e8*points(k,1);
y(c).lijn = p(c).point(1)*x+p(c).point(2);
end
if abs(p(c).point(1)) < 1e-8
y(c).a = 1e8;
end
matrix = [ matrix ; points(c,:)];
punten_lijst = [ Punten_lijst , k];
k = k+2;
d = d+2;
c = c+1;
elseif d == w
p(c).point = polyfit( points(k:k+1,1) , points(k:k+1,2), 1);
y(c).lijn = p(c).point(1)*x+p(c).point(2);
y(c).a = -1/p(c).point(1);
y(c).point(1) = p(c).point(1);
if abs(p(c).point(1)) > 1e8
y(c).a = 0;
y(c).point(1) = 1e8;
y(c).point(2) = points(k,2) + sign(p(c).point(1))*1e8*points(k,1);
y(c).lijn = p(c).point(1)*x+p(c).point(2);
end
if abs(p(c).point(1)) < 1e-8
y(c).a = 1e8;
end
matrix = [ matrix ; points(c,:)];
punten_lijst = [ Punten_lijst , k];
k = k+1;
else
d=d+1;
end
end
end
% searching the starting point
% maximale dikte van het blik is 0.6 mm
% dit wordt gebruikt om het begin punt te zoeken
if sign( y(punt_start-1).point(1) ) == sign( y(punt_start).point(1) )
hook = ( atan(y(punt_start-1).point(1) ) + atan(y(punt_start-1).point(1) ) ) / 2;
p_vector = tan(hook);
disp('haha')
elseif sign( y(punt_start-1).point(1) ) == sign( y(punt_start-1).point(1) )
hook = ( atan(y(punt_start-1).point(1) ) + atan(y(punt_start-1).point(1) ) ) / 2;
p_vector = tan(hook);
disp('haha')
elseif sign( y(punt_start-1).point(1) ) == 0
hook = ( atan(y(punt_start-1).point(1) ) + atan(y(punt_start-1).point(1) ) ) / 2;
p_vector = tan(hook);
disp('haha')
elseif sign( y(punt_start-1).point(1) ) == 0
hook = ( atan(y(punt_start-1).point(1) ) + atan(y(punt_start-1).point(1) ) ) / 2;
p_vector = tan(hook);
disp('haha')
elseif sign( y(punt_start-1).point(1) ) == sign( y(punt_start-1).point(1) )
hook = ( atan(y(punt_start-1).point(1) ) + atan(y(punt_start-1).point(1) ) ) / 2;
p_vector = tan(hook);
disp('haha')
end
b_vector = points(punt_start,2)-p_vector*points(punt_start,1);
drie_vlak = polyfit( [ points(punt_start-1,1) , points(punt_start+1,1) ] , [ points(punt_start-1,2) , points(punt_start+1,2) ], 1 );
x_intersect = ( b_vector - drie_vlak(2) ) / ( drie_vlak(1) - p_vector );
yst = sqrt( 0.6^2 / ( 1+p_vector^2 ) );
y_mean = mean( points(:,2));
if points(punt_start,2) < y_mean
if ( y_mean - points(punt_start,2) ) < ( y_mean - yst)
p_start = [ points(punt_start,1) - x_intersect points(punt_start,2) - p_vector*x_intersect ];
end

60
else
    p_start = [ points(punt_start,1) + xst points(punt_start,2)+p_vector * xst ];
end
else
    if ( y_mean - points(punt_start,2) ) < ( y_mean - yst)
        p_start = [ points(punt_start,1) + xst points(punt_start,2)+p_vector * xst ];
    else
        p_start = [ points(punt_start,1) - xst points(punt_start,2)+p_vector * -xst ];
    end
end
figure(1)
hold on
plot(p_start(1),p_start(2),'or')
% zoeken naar het eerste MA punt
% zoeken naar straal
b_startpunt = p_start(2)-(points(punt_start-1).a * p_start(1));
% lijn y_startpunt = y(1).a*x + b_startpunt
% moet snijden met de lijn y(1).lijn
x_intersect2 = ( b_startpunt - p(punt_start-1).point(2) ) / ( p(punt_start-1).point(1) - y(punt_start-1).a );
y_intersect2 = y(punt_start-1).a*x_intersect2 + b_startpunt;
% de straal is de lengte van het lijnstuk tussen deze twee punten
r = sqrt( ( x_intersect2 - p_start(1) ) ^2 + ( y_intersect2 - p_start(2) ) ^2 );
afstand = r;
afstand lijst = [afstand lijst afstand];
r lijst = [r lijst r];
% aanmaken van de eerste zoek richting
while loop < 1
    [eind1 , eind2 ] = find( MA(end,1) >= punt_eind(1) - delta_x & MA(end,1) <= punt_eind(1) + delta_x);
    [eind3 , eind4 ] = find( MA(end,2) >= punt_eind(2) - delta_y & MA(end,2) <= punt_eind(2)+ delta_y);
    if (length(eind1) == 0) || (length(eind3) == 0)
        if afstand > 0.2
            while zoek < 1;
                test=test+1
                N = pi/(asin( error_max / r) );
                N = round(N);
                if N < 25;
                    N = 25;
                end
                for e = 1:(N+1);
                    p(e).circle=p_start+r*[ cos( 2*pi*(e)/N) sin( 2*pi*(e)/N) ];
                    for tel=1:w-aantal;
                        j = punten_lijst(tel);
                        b = p(e).circle(2)-y(tel).a*p(e).circle(1); % berekenen van b van y=a*x+b
                        min_x = min ( [points(j,1) points(j+1,1)] )-2*r;
                        max_x = max ( [points(j,1) points(j+1,1)] )+2*r;
                        min_y = min ( [points(j,2) points(j+1,2)] )-2*r;
                        max_y = max ( [points(j,2) points(j+1,2)] )+2*r;
                        if min_x < p_start(1) && p_start(1) < max_x
                            if min_y < p_start(2) && p_start(2) < max_y
                                if y(tel).a == 0
                                    xnew(tel) = ( p(tel).point(2)-b )/ ( y(tel).a-p(tel).point(1) );
                                    ynew(tel) = p(e).circle(2);
                                else
                                    xnew(tel) = ( p(tel).point(2)-b )/ ( y(tel).a-p(tel).point(1) );
                                    ynew(tel) = subs( y(tel).lijn,xnew(tel) );
                                end
                                L(tel) = sqrt( ( xnew(tel)-p(e).circle(1) )^2 + ( ynew(tel)-p(e).circle(2) )^2 )
                            else
                                L(tel) = NaN;
                                xnew(tel) = NaN;
                                ynew(tel) = NaN;
                            end
                        else
                            L(tel) = NaN;
                            xnew(tel) = NaN;
                            ynew(tel) = NaN;
                        end
                    end
                    Len(e,1:w-aantal) = [ L ];
                    x_punt(e,1:w-aantal) = xnew;
                    y_punt(e,1:w-aantal) = ynew;
                    end
                end
            end
        end
        end
    end
end
end
clear L
p_1(e)=p(e).circle(1);
figure(1);
hold on;
set(gca,'DataAspectRatio',[1 1 1]);
plot(p_1,p_2,'r')

start = 0;
[val,pos]=min(Len(:,1));
breakpoint = diff(pos);\nk1_lijst = Len;
end = val;

[li,kz] = find( breakpoint ~= 0 );
k2_lijst(end) = length(k1_lijst);
s = length(k1);

if a > 1

start = 0;
for t = 1:a;

val2 = min( Len( kr(t),:) ) + min( Len( kr(t+1,:), :) );

end

while search < 1

while controle < 1

[qq,ww] = find( lengte == Lengte( end-start ) );

length_w = length(ww)+1;

end

if search < 1

for t = 1:a;

val(t) = min( Len( kr(t),:) ) + min( Len( kr(t+1,:), :) );

end

while search < 1

end

start = 0;
controle = 0;
if search < 1

s = 0;

end

else

if (length(rr)) > 0 && (length(rr)) == 0

start = start+1

length_w = length(ww)+1;

end

if search < 1

end

else

end

end

end

end

end

end
\[ r = \frac{\text{sorteer}(1) + \text{sorteer}(2)}{2} \]
\[
\text{afstand} = \text{sorteer}(1);
\]
\[
dx = \text{test_punt}(1)-\text{p}(\text{rr}+1).\text{circle}(1);
\]
\[
dy = \text{test_punt}(2)-\text{p}(\text{rr}+1).\text{circle}(2);
\]
\[
p_{\text{start}} = [\text{test_punt}(1)-dx, \text{test_punt}(2)-dy];
\]
\[\text{figure}(1);\]
\[\text{hold on;}\]
\[\text{plot}(p_{\text{start}}(1),p_{\text{start}}(2),'ok')\]
\[\text{MA} = [\text{MA}; p_{\text{start}}];\]
\[\text{r_lijst} = [\text{r_lijst}, r];\]
\[\text{afstand_lijst} = [\text{afstand_lijst}, \text{afstand}];\]
\[
\text{end}
\]
\[
\text{search} = 1;
\]
\[
\text{zoek} = 1;
\]
\[
\text{end} % \text{einde search loop}
\]
\[
\text{else} % \text{als a} > 1
\]
\[
tt = t;
\]
\[
\text{afstand} = r;
\]
\[
\text{r_lijst} = [\text{r_lijst}, r];
\]
\[
[tt_1, tt_2] = \text{find}(\text{max}(\text{val}) == \text{Len});
\]
\[
p_{\text{start}} = \text{p}(\text{tt_1}(1)).\text{circle};
\]
\[\text{MA} = [\text{MA}; p_{\text{start}}];\]
\[
\text{if} \; \text{tt} > 3
\]
\[
\text{search} = 1;
\]
\[
\text{zoek} = 1;
\]
\[
\text{loop} = 1;
\]
\[
\text{disp}'(\text{del})';
\]
\[
\text{disp}'(\text{einde simulatie})';
\]
\[
\text{end}
\]
\[
\text{end} % \text{einde loop a} > 1
\]
\[
\text{search} = 0;
\]
\[
\text{end} % \text{einde zoek loop}
\]
\[
\text{tt} = 1;
\]
\[
\text{zoek} = 0;
\]
\[
\text{rend} = 0;
\]
\[
\text{else} % \text{als afstand} > 0.2
\]
\[
\text{for} \; g = 1:\text{andere_kring}
\]
\[
\text{af} = \sqrt{(p_{\text{start}}(1)-\text{points}(g,1))^2 + (p_{\text{start}}(2)-\text{points}(g,2))^2};
\]
\[
\text{af_lijst} = [\text{af_lijst}; \text{af}];
\]
\[
[\text{po1}, \text{po2}] = \text{min}(\text{af_lijst}(1:1:2:end,1));
\]
\[
\text{if} \; \text{po1} > \text{punt_start}
\]
\[
\text{punt_start} = \text{po2}+1;
\]
\[
\text{else}
\]
\[
\text{punt_start} = \text{po2};
\]
\[
\text{end}
\]
\[
\text{af_lijst} = \emptyset;
\]
\[
\text{if} \; \text{sign}(\text{y}(\text{punt_start}-1).\text{point}(1)) == \text{sign}(\text{y}(\text{punt_start}).\text{point}(1))
\]
\[
\text{hoek} = \text{atan}(\text{y}(\text{punt_start}).\text{point}(1)) + \text{atan}(\text{y}(\text{punt_start}-1).\text{point}(1));
\]
\[
\text{p_vector} = \text{tan}(\text{hoek});
\]
\[
\text{elseif} \; \text{sign}(\text{points}(\text{punt_start},1)-\text{points}(\text{punt_start}-1,1)) == 0
\]
\[
\text{hoek} = \text{atan}(\text{y}(\text{punt_start}).\text{point}(1)) + \text{atan}(\text{y}(\text{punt_start}-1).\text{point}(1));
\]
\[
\text{p_vector} = \text{tan}(\text{hoek});
\]
\[
\text{elseif} \; \text{sign}(\text{points}(\text{punt_start},1)-\text{points}(\text{punt_start}-1,1)) == 0
\]
\[
\text{hoek} = \text{atan}(\text{y}(\text{punt_start}).\text{point}(1)) + \text{atan}(\text{y}(\text{punt_start}-1).\text{point}(1));
\]
\[
\text{p_vector} = \text{tan}(\text{hoek});
\]
\[
\text{elseif} \; \text{sign}(\text{points}(\text{punt_start},1)-\text{points}(\text{punt_start}-1,1)) == 0
\]
\[
\text{hoek} = \text{atan}(\text{y}(\text{punt_start}).\text{point}(1)) + \text{atan}(\text{y}(\text{punt_start}-1).\text{point}(1));
\]
\[
\text{p_vector} = \text{tan}(\text{hoek});
\]
\[
\text{end}
\]
\[
\text{b_vector} = \text{points}(\text{punt_start},2)-\text{p_vector}.*\text{points}(\text{punt_start},1);
\]
\[
\text{drievlak} = \text{polyfit}([\text{points}(\text{punt_start},1) \; \text{points}(\text{punt_start}-1,1)];[\text{points}(\text{punt_start},2) \; \text{points}(\text{punt_start}-1,1)];1);
\]
\[
\text{x_intersects} = (\text{b_vector} - \text{drievlak}(2)) / (\text{drievlak}(1) - \text{p_vector});
\]
\[
\text{y_intersects} = \text{points}(\text{punt_start},1) - \text{p_vector}.*\text{x_intersects};
\]
\[
\text{y_mean} = \text{mean}(\text{y_intersects});
\]
\[
\text{if} \; \text{points}(\text{punt_start},2) < \text{y_mean}
\]
\[
\text{if} \; (\text{y_mean} - \text{points}(\text{punt_start},2)) < (\text{y_mean} - \text{y_intersects})
\]
\[
\text{p_{\text{start}} = [\text{points}(\text{punt_start}) - \text{y_intersects} \; \text{p_vector}.*\text{y_intersects}];}
\]
\[
\text{else}
\]
\[
\text{p_{\text{start}} = [\text{points}(\text{punt_start}) - \text{y_intersects} \; \text{p_vector}.*\text{y_intersects}];}
\]
\[
\text{end}
\]
\[
\text{else}
\]
\[
\text{if} \; (\text{y_mean} - \text{points}(\text{punt_start},2)) < (\text{y_mean} - \text{y_intersects})
\]
\[
\text{p_{\text{start}} = [\text{points}(\text{punt_start}) - \text{y_intersects} \; \text{p_vector}.*\text{y_intersects}];}
\]
\[
\text{else}
\]
p_start = (points(start_point,1) - xst * points(start_point,2) + p_vector * -xst);

b_startpunt = p_start(2) - y(start_point - 1).a * p_start(1);
% lijn y_startpunt = y(1).a*x + b_startpunt
% moet snijden met de lijn y(1).lijn
x_intersect2 = (b_startpunt - p(start_point-1).point(2)) / (p(start_point-1).point(1) - y(start_point-1).a);
y_intersect2 = y(start_point-1).a * x_intersect2 + b_startpunt;
% de straal is de lengte van het lijnstuk tussen deze twee punten
r = sqrt((x_intersect2 - p(start_point-1).point(1))^2 + (y_intersect2 - p(start_point-1).point(2))^2);
afstand2 = r;
afstand_lijst = [afstand_lijst afstand2];
lijk = (lijk 2);

% end
end % einde afstand loop

clear lengte qq breekpunt x_punt y_punt p_1 p_2 p.circle sorteer Len k1 k2 % pos val
else % als punt te dicht bij eindpunt ligt
loop = 1;
end
			t = 0;
end % einde loop loop

TT = [MA(1,:),lijk];
Medial_axis = {1};
for t = 1:length(TT)
if TT(t,3) > (max([delta_x, delta_y])/2)
Medial_axis = [Medial_axis; TT(t,:)];
end
end

figure
plot(points(:,1), points(:,2))
hold on
plot(Medial_axis(:,1), Medial_axis(:,2), 'r', Medial_axis(:,1), Medial_axis(:,2), 'k')
Appendix E

Linear medial axis algorithm

E.1 Medial

function [ medial_axis ] = medial(points)

% calculating the polynoms between the points, by using polyfit.
% separate the boundary loops, and deleting the double points

punt1 = points(1,:);

k = 1;
c = 1;
matrix = [];
punten_lijst = [];
andere_kring = [];
con_lijst = [];

w = length(points);
Max_dis_x = 0.5;
Max_dis_y = 0.5;
informatie = [];

if y(point(1)) > point(2) x_max x_min y_max y_min

matrix = [];

while k < length(points);

if (punt1(1) ~= points(d,1)) | (punt1(2) ~= points(d,2));

ppoint(c,1:2) = polyfit( points(k:k+1,1) ,points(k:k+1,2), 1);
ypoint(c,1) = ppoint(c,1);
ypoint(c,2) = ppoint(c,2);

end

informatie = [informatie; [ ypoint(c,1) ypoint(c,2) max(points(k:k+1,1))
min(points(k:k+1,1)) max(points(k:k+1,2)) min(points(k:k+1,2)) ]];

k = k+1;
c = c+1;
d = d+1;
end

elseif (punt1(1) == point(d,1)) KB (point(2) == point(d,2));

matrix = [matrix ; points(c,:)];
punten_lijst = [ punten_lijst , c];

informatie = [informatie; [ ypoint(c,1) ypoint(c,2) max(points(k:k+1,1))
min(points(k:k+1,1)) max(points(k:k+1,2)) min(points(k:k+1,2)) ]];

k = k+1;
c = c+1;
d = d+1;
end

elseif (punt1(1) == point(d,1)) KB (point(2) == point(d,2));

matrix = [matrix ; points(c,:)];
punten_lijst = [ punten_lijst , c];

informatie = [informatie; [ ypoint(c,1) ypoint(c,2) max(points(k:k+1,1))
min(points(k:k+1,1)) max(points(k:k+1,2)) min(points(k:k+1,2)) ]];

k = k+1;
c = c+1;
d = d+1;
end

end

if abs(ppoint(c,1)) > 1e-8

yy(c) = 0;
ypoint(c,1) = 1e8;
ypoint(c,2) = points(k,2)+sign(ppoint(c,1))*1e8*points(k,1);
end

if abs(ppoint(c,1)) < 1e-8

yy(c) = 1e8;
ypoint(c,1) = 0;
ypoint(c,2) = points(k,2);
end

matrix = [matrix ; points(c,:)];
punten_lijst = [ punten_lijst , c];
informatie = [informatie; [ ypoint(c,1) ypoint(c,2) max(points(k:k+1,1))
min(points(k:k+1,1)) max(points(k:k+1,2)) min(points(k:k+1,2)) ]];

k = k+1;
c = c+1;
d = d+1;
end

medial_axis = [];

end
ypoint(c,1) = 1e8;
ypoint(c,2) = ppoints(k,2)+ sign(ppoint(c,1))*1e8*points(k,1);
end
if abs(ppoint(c,1)) < 1e-8
ypoint(c,1) = 0;
ypoint(c,2) = ppoints(k,2);
end
matrix = [matrix ; ppoints(c,:)];
punten_lijst = [ punten_lijst , k];
informatie = [ informatie; [ ypoint(c,1) ypoint(c,2) max(points(k:k+1,1))
min(points(k:k+1,1)) max(points(k:k+1,2)) min(points(k:k+1,2))]]; k = k+2;
a = a+2;
c = c+1;
end
ander = [ -1 ; andere_kring ; length(points)-1 ];
for j = 1:length(ander)-1
figure(2)
hold on
plot( points((ander(j)+2):(ander(j+1)+1),1) , points((ander(j)+2):(ander(j+1)+1),2),'b')
grid on
set(gca,'DataAspectRatio',[1 1 1]); % plot setting to set the scaling of x,y and z axis identical
end
% reduction of points, by deleting unnecessary points
pt = [points(1,:)];
yt = [ ];
info = [ ];
s = 1;
a = 1;
ma = [ 1 andere_kring length(points)-length(andere_kring) ];
punto = ppoints(punten_lijst,:);
andere_kring = [ ];
yt(j) = 1;
for k = 1:length(pt)-1
while s < an(k+1)-1
if abs(ypoint(s,1) - ypoint(s+e,1)) < 2e-2 & abs(ypoint(s,2) - ypoint(s+e,2)) < 1e-1
s = s+1;
if (s+e) >= an(k+1)
if an(k+1) == an(end)
yt = [yt ; ypoint(end,:) s e];
info = [info; ypoint(s,:) max([ points(s,1) points(1,1)]) min([ points(s,1) points(1,1)])
 max([ points(s,2) points(1,2)]) min([ points(s,2) points(1,2)])];
s = an(end);
else
yt = [yt ; ypoint(s,:) s e];
info = [info; ypoint(s,:) max([ points(s,1) points(1,1)]) min([ points(s,1) points(1,1)])
 max([ points(s,2) points(1,2)]) min([ points(s,2) points(1,2)])];
s = s+1;
end
end
else
pt = [pt ; points(s+e,:)];
yt = [yt ; ypoint(s,:) s e];
info = [info; ypoint(s,:) max([ points(s,1) points(s+e,1)]) min([ points(s,1) points(s+e,1)])
 max([ points(s,2) points(s+e,2)]) min([ points(s,2) points(s+e,2)])];
s = s+1;
if (s+e) >= an(k+1)
if an(k+1) == an(end)
yt = [yt ; ypoint(end,:) s e];
info = [info; ypoint(s,:) max([ points(s,1) points(1,1)]) min([ points(s,1) points(1,1)])
 max([ points(s,2) points(1,2)]) min([ points(s,2) points(1,2)])];
s = an(end);
else
yt = [yt ; ypoint(s,:) s e];
info = [info; ypoint(s,:) max([ points(s,1) points(s+e,1)]) min([ points(s,1) points(s+e,1)])
 max([ points(s,2) points(s+e,2)]) min([ points(s,2) points(s+e,2)])];
s = s+1;
end
end
end
andere_kring = [ andere_kring ; length(pt)];
end
ypoint
points = pt;
ypoint = pt;
informatie = info
point = points;
y = -1./ypoint(:,1);
for e = 1:length(ya)
    if abs(ya(e)) == inf
        ya(e) = 1e8;
    elseif abs(abs(ya(e)) < 1e-5
        ya(e) = 0;
    end
end

figure(2)
hold on
plot(points(:,1) , points(:,2),'.b' )

if length(andere_kring) == 2;
    rand_loop = 1;
    while rand_loop = 1;
        length_1 = length( point(1:andere_kring,1) );
        length_2 = length( point(andere_kring+1:end,1) );
        if length_1 >= length_2
            w_loop_1 = 1:andere_kring(1);
            w_loop_2 = andere_kring(1)+1:length(point);
            kring = 1;
        elseif w_loop_1 = 1:andere_kring(1);
            w_loop_2 = andere_kring(1)+1:length(point);
            kring = 2;
        end
    test_loop = 1;
    while test_loop = 1
        while length(w_loop_1) ~= 0
            w = w_loop_1(1);
            punt_start = point(w,:);
            if kring = 1
                straal_gebied = (point(andere_kring(1)+1:end,1)-punt_start(1)).^2 + (point(andere_kring(1)+1:end,2)-punt_start(2)).^2;
                [q1 q2] = min(straal_gebied);
                tweede_punt = q2+andere_kring(1);
            else
                straal_gebied = (point(1:andere_kring(1),1)-punt_start(1)).^2 + (point(1:andere_kring(1),2)-punt_start(2)).^2;
                [q1 q2] = min(straal_gebied);
                tweede_punt = q2;
            end
            lijnen_test = 1;
            while lijnen_test = 1
                lijnen_test = 0;
                if w ~= 1 & tweede_punt = andere_kring(1) & tweede_punt = length(point) & v = andere_kring(1) & tweede_punt = 1 & v = anderen_kring(1)+1
                    punt1 = v-1;
                    punt2 = v;
                    punt3 = v+1;
                    punt4 = tweede_punt-1;
                    punt5 = tweede_punt;
                    punt6 = tweede_punt+1;
                elseif v = 1 & tweede_punt = anderen_kring(1)+1
                    punt1 = andere_kring(1);
                    punt2 = v;
                    punt3 = v+1;
                    punt4 = tweede_punt-1;
                    punt5 = tweede_punt;
                    punt6 = tweede_punt+1;
                elseif v = 1 & tweede_punt = length(point)
                    punt1 = anderen_kring(1);
                    punt2 = v;
                    punt3 = v+1;
                    punt4 = tweede_punt-1;
                    punt5 = tweede_punt;
                    punt6 = tweede_punt+1;
                elseif v = anderen_kring(1)+1 & tweede_punt = anderen_kring(1)+1
                    punt1 = length(point);
                    punt2 = v;
                    punt3 = v+1;
                    punt4 = tweede_punt-1;
                    punt5 = tweede_punt;
                    punt6 = tweede_punt+1;
                elseif v = anderen_kring(1)+1 & tweede_punt = length(point)
                    punt1 = length(point);
                    punt2 = v;
                    punt3 = v+1;
                    punt4 = tweede_punt-1;
                    punt5 = tweede_punt;
                    punt6 = tweede_punt+1;
        end
    end
end

76
punt3 = w+1;
punt4 = tweede_punt-1;
punt5 = tweede_punt;
punt6 = andere_kring(1)+1;
elseif u == andere_kring(1)+1 & tweede_punt == 1;
punt1 = length(point);
punt2 = w;
punt3 = w+1;
punt4 = andere_kring(1);
punt5 = tweede_punt;
punt6 = tweede_punt+1;
elseif u == andere_kring(1)+1
punt1 = length(point);
punt2 = w;
punt3 = w+1;
punt4 = andere_kring(1);
punt5 = tweede_punt;
punt6 = tweede_punt+1;
elseif u == andere_kring(1)
elseif w == andere_kring(1)+1 & tweede_punt == 1;
punt1 = w -1;
punt2 = andere_kring(1);
punt3 = 1;
punt4 = tweede_punt;
punt5 = tweede_punt+1;
punt6 = tweede_punt+1;
elseif u == andere_kring(1) & tweede_punt == length(point);
punt1 = w -1;
punt2 = andere_kring(1);
punt3 = 1;
punt4 = tweede_punt;
punt5 = tweede_punt;
punt6 = tweede_punt+1;
elseif w == andere_kring(1)
elseif w == andere_kring(1) & tweede_punt == andere_kring(1)+1
punt1 = w -1;
punt2 = andere_kring(1);
punt3 = 1;
punt4 = tweede_punt;
punt5 = tweede_punt;
punt6 = tweede_punt+1;
elseif w == andere_kring(1)
elseif w == andere_kring(1) & tweede_punt == length(point);
punt1 = w -1;
punt2 = andere_kring(1);
punt3 = 1;
punt4 = tweede_punt;
punt5 = tweede_punt;
punt6 = andere_kring(1)+1;
elseif w == andere_kring(1)
elseif w == andere_kring(1) & tweede_punt == 1;
punt1 = w -1;
punt2 = andere_kring(1);
punt3 = 1;
punt4 = tweede_punt;
punt5 = tweede_punt;
punt6 = andere_kring(1)+1;
elseif w == andere_kring(1)
elseif w == andere_kring(1) & tweede_punt == andere_kring(1)+1
punt1 = w -1;
punt2 = w;
punt3 = andere_kring(1)+1;
punt4 = tweede_punt;
punt5 = tweede_punt;
punt6 = tweede_punt+1;
elseif u == 1 & tweede_punt == andere_kring(1)+1
punt1 = w -1;
punt2 = w;
punt3 = andere_kring(1)+1;
punt4 = tweede_punt;
punt5 = tweede_punt;
punt6 = tweede_punt+1;
elseif u == 1 & tweede_punt == length(point);
punt1 = w -1;
punt2 = w;
punt3 = w;
punt4 = length(point);
punt5 = tweede_punt;
punt6 = tweede_punt;
elseif u == 1 & tweede_punt == 1;
punt1 = w -1;
punt2 = w;
punt3 = w;
punt4 = andere_kring(1);
punt5 = tweede_punt;
punt6 = andere_kring(1)+1;
else
punt1 = w -1;
punt2 = w;
punt3 = w;
punt4 = andere_kring(1);
punt5 = tweede_punt;
punt6 = tweede_punt+1;
end
y2point = ypoint( punt2 : );
y1point = ypoint( punt1 : );
if w==13
    y1point = ypoint(point(punt1,:));
    y2point = ypoint(point(punt4,:));
    [p_vector b] = medial_line_search(point(punt2,:), point(punt1,:), point(punt3,:), y1point, y2point);
    % Searching direction of second point.
    y1point = ypoint(point(punt5,:));
    y2point = ypoint(point(punt4,:));
    [p_vector_1 b_1] = medial_line_search(point(punt5,:), point(punt4,:), point(punt6,:), y2point, y1point);
    if abs((p_vector) - (p_vector_1)) < 1e-2
        x_snij = (max([point(punt2,1) point(punt5,1)]) - min([point(punt2,1) point(punt5,1)]))/2 + min([point(punt2,1) point(punt5,1)]);
        y_snij = (max([point(punt2,2) point(punt5,2)]) - min([point(punt2,2) point(punt5,2)]))/2 + min([point(punt2,2) point(punt5,2)]);
    elseif abs(p_vector) > 500 & abs(p_vector_1) > 500
        x_snij = (max([point(punt2,1) point(punt5,1)]) - min([point(punt2,1) point(punt5,1)]))/2 + min([point(punt2,1) point(punt5,1)]);
        y_snij = (max([point(punt2,2) point(punt5,2)]) - min([point(punt2,2) point(punt5,2)]))/2 + min([point(punt2,2) point(punt5,2)]);
    else
        x_snij = (b - b_1) / (p_vector_1 - p_vector);
        y_snij = p_vector * x_snij + b;
    end
    xv = [point(1:andere_kring(1),1); point(1,1)];
    yv = [point(1:andere_kring(1),2); point(1,2)];
    xv2 = [point(andere_kring(1)+1:end,1); point(andere_kring(1)+1,1)];
    yv2 = [point(andere_kring(1)+1:end,2); point(andere_kring(1)+1,2)];
    [in, on] = inpolygon(x_snij, y_snij, xv, yv);
    [in2, on2] = inpolygon(x_snij, y_snij, xv2, yv2);
    if in == 1 & on == 0 & in2 == 0
        % checking the lengths
        [x_loop_1, w, lijn_text, medial_axis, tweede_punt ] = Loop_1(ypoint, ya, ..,point, medial_axis, tweede_punt, w_loop_1, w_loop_2, x_snij, y_snij, punt1,punt2,punt3, punt4,..
                                        punt5, punt6, lijn_text );
    else
        if ypoint(punt1,1) == ypoint(punt4,1) & ypoint(punt2,1) == ypoint(punt4,1)
            if ypoint(punt1,2) == ypoint(punt4,2) & ypoint(punt2,2) == ypoint(punt4,2)
                tweede_punt = punt6;
                lijn_text = 1;
            elseif ypoint(punt1,2) == ypoint(punt5,2) & ypoint(punt2,2) == ypoint(punt5,2)
                tweede_punt = punt4;
                lijn_text = 1;
            end
        end
    end
    if lijn_text == 0
        b_a = point(punt2,2) - p_vector * point(punt2,1);
        x_s_1 = (ypoint(punt5,2) - b_a) / (p_vector - ypoint(punt5,1));
        y_s_1 = p_vector * x_s_1 + b_a;
        x_s_2 = (ypoint(punt4,2) - b_a) / (p_vector - ypoint(punt4,1));
        y_s_2 = p_vector * x_s_2 + b_a;
    if w == 13
        x_s_1
        y_s_1
        x_s_2
        y_s_2
        p_vector
        b_a
        points(punt1,:)
        points(punt2,:)
        points(punt3,:)
        points(punt4,:)
        points(punt5,:)
        points(punt6,:)
    end
    if x_s_1 <= max([point(punt1,1) point(punt6,1)]) & x_s_1 >= min([point(punt1,1) ..
                                        point(punt6,1)]) & y_s_1 <= max([point(punt1,2) point(punt6,2)]) &
                                        y_s_1 >= min([point(punt1,2) point(punt6,2)])
        lengte_1 = sqrt( (point(punt2,1) - x_s_1)^2 + (point(punt2,2) - y_s_1)^2 );
    elseif
        lengte_1 = inf;
    end
    if x_s_2 <= max([point(punt1,1) point(punt4,1)]) & x_s_2 >= min([point(punt1,1) ..
                                        point(punt4,1)]) & y_s_2 <= max([point(punt1,2) point(punt4,2)]) &
                                        y_s_2 >= min([point(punt1,2) point(punt4,2)])
        lengte_2 = sqrt( (point(punt2,1) - x_s_2)^2 + (point(punt2,2) - y_s_2)^2 );
    elseif
        lengte_2 = inf;
    end
    if lengte_1 < lengte_2
49
\[ x_{snij} = \frac{\text{max}(\text{point}(\text{punt2},1)) - \text{min}(\text{point}(\text{punt2},1))}{2} + \text{min}(\text{point}(\text{punt2},1)); \]
\[ y_{snij} = \frac{\text{max}(\text{point}(\text{punt2},2)) - \text{min}(\text{point}(\text{punt2},2))}{2} + \text{min}(\text{point}(\text{punt2},2)); \]
\[ b_punt = y_{snij} - y_{point}(\text{punt5}) \times x_{snij}; \]
\[ x_{punt} = \frac{y_{point}(\text{punt5},2) - b_punt}{y_{point}(\text{punt5}) - y_{point}(\text{punt5});} \]
\[ y_{punt} = y_{point}(\text{punt5}) \times x_{punt} + b_punt; \]
\[ \text{lengte} = \sqrt{(x_{snij} - x_{punt})^2 + (y_{snij} - y_{punt})^2}; \]
\[ \text{else} \]
\[ x_{snij} = \frac{\text{max}(\text{point}(\text{punt5},1)) - \text{min}(\text{point}(\text{punt5},1))}{2} + \text{min}(\text{point}(\text{punt5},1)); \]
\[ y_{snij} = \frac{\text{max}(\text{point}(\text{punt5},2)) - \text{min}(\text{point}(\text{punt5},2))}{2} + \text{min}(\text{point}(\text{punt5},2)); \]
\[ b_punt = y_{snij} - y_{point}(\text{punt5}) \times x_{snij}; \]
\[ x_{punt} = \frac{y_{point}(\text{punt5},2) - b_punt}{y_{point}(\text{punt5}) - y_{point}(\text{punt5});} \]
\[ y_{punt} = y_{point}(\text{punt5}) \times x_{punt} + b_punt; \]
\[ \text{lengte} = \sqrt{(x_{snij} - x_{punt})^2 + (y_{snij} - y_{punt})^2}; \]
\[ \text{end} \]
\[ \text{if } \text{in} == 1 \text{ and } \text{on} == 0 \text{ and } \text{in2} == 0 \text{ and } \text{on2} == 0 \text{ and } \text{medial_axis} == 0 \text{ end} \]
\[ \text{medial_axis} = [\text{medial_axis}; x_{snij} y_{snij} 2 \times \text{lengte} \text{ punt2 punt5}]; \]
\[ % \text{hold on} \]
\[ % \text{plot}(x_{snij}, y_{snij}, 'ok') \]
\[ % \text{adjust vector w_loop, w_loop is necessary} \]
\[ % \text{for checking if every point is used.} \]
\[ (x_{snij} = \text{find}(\text{w_loop}, 2) == \text{punt3}); \]
\[ \text{w_loop}_1 = \text{w_loop}_1(2:end); \]
\[ \text{if length(w_loop}_1) == 0 \]
\[ \text{w_loop}_2 = [\text{w_loop}_2; 1 \times 4 - 6 \times \text{w_loop}_2(2:4) \text{end}]; \]
\[ \text{else} \]
\[ \text{medial_axis} = [\text{medial_axis}; x_{snij} y_{snij} \times \text{length}(\text{w_loop}_2)]; \]
\[ \text{if length(w_loop}_1) == 0 \]
\[ \text{w_loop}_2 = [\text{w_loop}_2; 1 \times 4 - 6 \times \text{w_loop}_2(2:4) \text{end}]; \]
\[ \text{end} \]
\[ \text{if length(w_loop}_2) == 0 \]
\[ \text{w_loop}_1 = [\text{w_loop}_2]; \]
\[ \text{w_loop}_2 = []; \]
\[ \text{if kring} == 1 \]
\[ \text{kring} = 2; \]
\[ \text{end} \]
\[ \text{if kring} == 1; \]
\[ \text{medial_axis} = [\text{medial_axis}; 0 0 0 0 0]; \]
\[ \text{else} \]
\[ \text{medial_axis} = [\text{medial_axis}; \text{medial_axis}]; \]
\[ \text{rond_loop} = 0; \]
\[ \text{test_loop} = 0; \]
\[ \text{rond_loop} = 0; \]
\[ \text{end} \]
\[ \text{end} \]
\[ \text{if length(w_loop}_1) == 0 \]
\[ \text{w_loop}_2 = [\text{w_loop}_2]; \]
\[ \text{end} \]
\[ \text{w_loop}_2 = [\text{w_loop}_2]; \]
\[ \text{end} \]
\[ \text{if length(w_loop}_1) == 0 \]
\[ \text{w_loop}_2 = [\text{w_loop}_2]; \]
\[ \text{end} \]
\[ \text{end} \]
\[ \text{end} \]
\[ \text{medial_axis} = [\text{medial_axis}; \text{medial_axis}]; \]
\[ \text{for s = 1:1:0 \text{if length(w_loop}_1) == 0 \text{medial_axis}_1 = [\text{medial_axis}_1]; \text{end} \text{end} \}
\[ \text{if length(w_loop}_2) == 0 \text{medial_axis}_1 = [\text{medial_axis}_1]; \text{for s = 1:1:0 \text{end} \}


if r1(1) == 1
    if medial_axis(s,4) < MA(q1-1,5)
        MA = [medial_axis(s,1:3) medial_axis(s,5:-1:4); MA(1:end,:)];
    else
        MA = [MA(1:r1(1),:);[medial_axis(s,1:3) medial_axis(s,5:-1:4)]; MA(r1(1)+1:end,:)];
    end
else
    if medial_axis(s,4) < MA(r1(1),5)
        MA = [MA(1:r1(1),:);[medial_axis(s,1:3) medial_axis(s,5:-1:4)]; MA(r1(1):end,:)];
    else
        MA = [MA(1:r1(1),:);[medial_axis(s,1:3) medial_axis(s,5:-1:4)]; MA(r1(1)+1:end,:)];
    end
end
else
    [r1 r2] = find(MA(:,5) == (medial_axis(s,4)-1));
    if r1(1) == 1
        if medial_axis(s,4) < MA(q1-1,5)
            MA = [medial_axis(s,1:3) medial_axis(s,5:-1:4); MA(1:end,:)];
        else
            MA = [MA(1:r1(1),:);[medial_axis(s,1:3) medial_axis(s,5:-1:4)]; MA(r1(1)+1:end,:)];
        end
    else
        if medial_axis(s,4) < MA(r1(1)-1,5)
            MA = [MA(1:r1(1)-1,:);[medial_axis(s,1:3) medial_axis(s,5:-1:4)]; MA(r1(1):end,:)];
        else
            MA = [MA(1:r1(1),:);[medial_axis(s,1:3) medial_axis(s,5:-1:4)]; MA(r1(1)+1:end,:)];
        end
    end
end
medial_axis = MA(1:(medial_total-1),:);
else
    medial_axis = [medial_axis; medial_axis(1,:)];
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% If geometry exist of only one boundary loop
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if length(andere_kring) == 1
    rond_loop = 0;
    andere_kring = length(points);
    for w = 1:andere_kring;
        clear x_p y_p lengte tweede_punt lijnet d k;
        LL = [];
        cos_lijn=[];
        sin_lijn=[];
        punt_start = points(w,:);
        straal_gebied = (points([1:w-1 w+1:end],1)-punt_start(1)).^2 + (points([1:w-1 w+1:end],2)-punt_start(2)).^2;
        [q1 q2] = min(straal_gebied);
        if q2 >= w
            q2 = q2+1;
        end
        tweede_punt = q2
        if w == 1
            if tweede_punt == andere_kring;
                punt0 = andere_kring-1;
                punt1 = andere_kring;
                punt2 = 1;
                punt3 = q1;
                punt4 = tweede_punt;
                punt5 = tweede_punt-1;
                punt6 = 1;
            else
                punt0 = andere_kring;
                punt1 = andere_kring-1;
                punt2 = 1;
                punt3 = q1;
                punt4 = tweede_punt;
                punt5 = tweede_punt-1;
                punt6 = 1;
            end
        elseif w == andere_kring;
            if tweede_punt == 1;
                punt0 = andere_kring;
                punt1 = andere_kring-1;
                punt2 = 1;
                punt3 = q1;
                punt4 = tweede_punt;
                punt5 = tweede_punt-1;
                punt6 = 1;
            elseif w == andere_kring-1;
                punt0 = andere_kring-2;
                punt1 = andere_kring-1;
                punt2 = 1;
                punt3 = q1;
                punt4 = tweede_punt;
                punt5 = tweede_punt-1;
                punt6 = 1;
            end
        end
    end
punt4 = tweede_punt - 1;
punt5 = tweede_punt;
punt6 = tweede_punt + 1;
end

else
if tweede_punt == andere_kring;
punt0 = w - 2;
punt1 = w - 1;
punt2 = w;
punt3 = w + 1;
punt4 = tweede_punt - 1;
punt5 = tweede_punt;
punt6 = tweede_punt + 1;
else
punt0 = w - 2;
punt1 = w - 1;
punt2 = w;
punt3 = w + 1;
punt4 = tweede_punt - 1;
punt5 = tweede_punt;
punt6 = tweede_punt + 1;
end

end

y2point = ypoint( punt2, : );
y1point = ypoint( punt1, : );

[p_vector b] = medial_line_search( points(punt2,:), points(punt1,:), points(punt3,:), y2point, y1point )
y2point = ypoint( punt5, : );
y1point = ypoint( punt4, : );

[p_vector_1 b_1] = medial_line_search( points(punt5,:), points(punt4,:), points(punt6,:), y2point, y1point )
x_p = ( b - b_1 ) / ( p_vector_1 - p_vector);
y_p = p_vector*x_p + b;

xv = [points(:,1); points(1,1)];
yv = [points(:,2); points(1,2)];
[in, on] = inpolygon(x_p, y_p, xv, yv);

if in == 0 | on == 1 % Check if intersection point is in the geometry.
if tweede_punt == punt1 || tweede_punt == punt3 % Check if second point is neighboring point.
[in, on, x_p, y_p, p_vector_2, b_2, tweede_punt, punt4, punt5, punt6] = Loop_2(points, w, punt_start, ypoint, punt4, punt5, punt6, andere_kring, b, p_vector);
if in == 0 | on == 1 % Intersection point is not in the geometry
[medial_axis] = Loop_5(points, punt0, punt1, punt2, punt3, punt5, medial_axis, p_vector, p_vector_2, informatie, ypoint, ya, w, b, b_2, xv, yv);
else % point is in the geometry.
[as, Lengte] = Loop_6(ya, points, ypoint, punt1, punt2, punt4, punt5, y_p, x_p);
if min(Lengte) > as % minimal length has to be at least 20% of maximal length
afstand = max( Lengte);
straal_gebied2 = (points(:,1) - x_p).^2 + (points(:,2) - y_p).^2;
[q1, q2] = find( straal_gebied2 <= afstand^2 );
e1 = find( q1 ~= w & q1 ~= tweede_punt);
if length(e1) ~= 0 % check if point is in enclosed circle NO
[medial_axis] = Loop_5(points, punt0, punt1, punt2, punt3, punt5, medial_axis, p_vector, p_vector_2, informatie, ypoint, ya, w, b, b_2, xv, yv);
else % check if point is in enclosed circle YES
[medial_axis] = Loop_7(Lengte, points, medial_axis, punt2, punt5, y_p, x_p);
end
else % length less then 20%.
[medial_axis] = Loop_5(points, punt0, punt1, punt2, punt3, punt5, medial_axis, p_vector, p_vector_2, informatie, ypoint, ya, w, b, b_2, xv, yv);
end
end
end

else % Second point is no neighboring point.
[medial_axis] = Loop_5(points, punt0, punt1, punt2, punt3, punt5, medial_axis, p_vector, p_vector_1, informatie, ypoint, ya, w, b, b_1, xv, yv);
end
end

else % point is in the geometry
[as, Lengte] = Loop_6(ya, points, ypoint, punt1, punt2, punt4, punt5, y_p, x_p);
if min(Lengte) > as % minimal length has to be at least 20% of maximal length
afstand = max( Lengte);

72
straal_gebied2 = (points(:,1)-x_p).^2 + (points(:,2)-y_p).^2;
q1 q2 = find(straal_gebied2 <= afstand^2);
e1 = find( q1 ~= w & q1 ~= tweede_punt);
if length(e1) ~= 0 % Check if there are other points in the enclosed circle with radius length.
    if tweede_punt == punt1 || tweede_punt == punt3 % Check if second point is neighboring point
        [in, on, x_p, y_p, p_vector_2, b_2, tweede_punt, punt4, punt5, punt6] = Loop_2(points,...
            w,punt_start, ypoint, punt4, punt5, punt6, andere_kring, b, p_vector);
        if in == 0 | on == 1 % Point is not in the geometry
            [medial_axis]=Loop_5(points,punt0,punt1,punt2, punt3, punt5, medial_axis, p_vector,p_vector_2,...
                informatie, ypoint, ya, w, b, b_2, xv, yv);
        else % Point is part of the geometry.
            [as,Lengte] = Loop_6(ya,points, ypoint,punt1,punt2,punt4,punt5,y_p,x_p);
            if min(Lengte) > as % Minimal length has to be at least 20% of the maximal length.
                afstand = max( Lengte);
                straal_gebied2 = (points(:,1)-x_p).^2 + (points(:,2)-y_p).^2;
                q1 q2 = find(straal_gebied2 <= afstand^2);
                e1 = find( q1 ~= w & q1 ~= tweede_punt);
                if length(e1) ~= 0 % Check if intersection point is in the enclosed circle.
                    [medial_axis]=Loop_7(Lengte,points, medial_axis, punt2, punt5, medial_axis,...
                        p_vector,p_vector_2, informatie, ypoint, ya, w, b, b_2, xv, yv);
                else % Check if point is in the enclosed circle
                    [medial_axis]=Loop_5(points,punt0,punt1,punt2, punt3, punt5, medial_axis,...
                        p_vector,p_vector_2, informatie, ypoint, ya, w, b, b_2, xv, yv);
                end
            else % length is less than 20% of the maximal length
                [medial_axis]=Loop_5(points,punt0,punt1,punt2, punt3, punt5, medial_axis,...
                    p_vector,p_vector_2, informatie, ypoint, ya, w, b, b_2, xv, yv);
            end
        end
    else % second point is no neighboring point
        [medial_axis]=Loop_5(points,punt0,punt1,punt2, punt3, punt5, medial_axis, p_vector,p_vector_1,...
            informatie, ypoint, ya, w, b, b_1, xv, yv);
    end
else % No points in the enclosed circle
    [medial_axis]=Loop_7(Lengte,points, medial_axis, punt2, punt5,x_p, y_p);
end
end
else % Minimal length is smaller as 20% of the maximal length
    if tweede_punt == punt1 || tweede_punt == punt3 % Check if second point is a neighboring point.
        [in, on, x_p, y_p, p_vector_2, b_2, tweede_punt, punt4, punt5, punt6] = Loop_2(points,...
            w,punt_start, ypoint, punt4, punt5, punt6, andere_kring, b, p_vector);
        if in == 0 | on == 1 % Intersection point is no part of the geometry.
            [medial_axis]=Loop_5(points,punt0,punt1,punt2, punt3, punt5, medial_axis,...
                p_vector,p_vector_2, informatie, ypoint, ya, w, b, b_2, xv, yv);
        else % point is part of the geometry.
            [as,Lengte] = Loop_6(ya,points, ypoint,punt1,punt2,punt4,punt5,y_p,x_p);
            if min(Lengte) > as % Minimal length has to be at least 20% of the maximal length.
                afstand = max( Lengte);
                straal_gebied2 = (points(:,1)-x_p).^2 + (points(:,2)-y_p).^2;
                q1 q2 = find(straal_gebied2 <= afstand^2);
                e1 = find( q1 ~= w & q1 ~= tweede_punt);
                if length(e1) ~= 0 | length(e2) ~= 0 % Check if points are in the enclosed circle.
                    [medial_axis]=Loop_7(Lengte,points, medial_axis, punt2, punt5, medial_axis,...
                        p_vector,p_vector_2, informatie, ypoint, ya, w, b, b_2, xv, yv);
                else % Check if point is in the enclosed circle
                    [medial_axis]=Loop_5(points,punt0,punt1,punt2, punt3, punt5, medial_axis,...
                        p_vector,p_vector_2, informatie, ypoint, ya, w, b, b_2, xv, yv);
                end
            else % length less than 20% of the maximal length
                [medial_axis]=Loop_5(points,punt0,punt1,punt2, punt3, punt5, medial_axis,...
                    p_vector,p_vector_2, informatie, ypoint, ya, w, b, b_2, xv, yv);
            end
    else % second point is no neighboring point
        [medial_axis]=Loop_5(points,punt0,punt1,punt2, punt3, punt5, medial_axis, p_vector,p_vector_1,...
            informatie, ypoint, ya, w, b, b_1, xv, yv);
    end
else % length is less than 20% of the maximal length
    [medial_axis]=Loop_5(points,punt0,punt1,punt2, punt3, punt5, medial_axis,...
        p_vector,p_vector_2, informatie, ypoint, ya, w, b, b_2, xv, yv);
end
end
else % Minimal length is smaller as 20% of the maximal length
    if tweede_punt == punt1 || tweede_punt == punt3 % Check if second point is a neighboring point.
        [in, on, x_p, y_p, p_vector_2, b_2, tweede_punt, punt4, punt5, punt6] = Loop_2(points,...
            w,punt_start, ypoint, punt4, punt5, punt6, andere_kring, b, p_vector);
        if in == 0 | on == 1 % Intersection point is no part of the geometry.
            [medial_axis]=Loop_5(points,punt0,punt1,punt2, punt3, punt5, medial_axis,...
                p_vector,p_vector_2, informatie, ypoint, ya, w, b, b_2, xv, yv);
        else % point is part of the geometry.
            [as,Lengte] = Loop_6(ya,points, ypoint,punt1,punt2,punt4,punt5,y_p,x_p);
            if min(Lengte) > as % Minimal length has to be at least 20% of the maximal length.
                afstand = max( Lengte);
                straal_gebied2 = (points(:,1)-x_p).^2 + (points(:,2)-y_p).^2;
                q1 q2 = find(straal_gebied2 <= afstand^2);
                e1 = find( q1 ~= w & q1 ~= tweede_punt);
                if length(e1) ~= 0 % Check if intersection point is in the enclosed circle.
                    [medial_axis]=Loop_7(Lengte,points, medial_axis, punt2, punt5, medial_axis,...
                        p_vector,p_vector_2, informatie, ypoint, ya, w, b, b_2, xv, yv);
                else % Check if point is in the enclosed circle
                    [medial_axis]=Loop_5(points,punt0,punt1,punt2, punt3, punt5, medial_axis,...
                        p_vector,p_vector_2, informatie, ypoint, ya, w, b, b_2, xv, yv);
                end
            else % length less than 20% of the maximal length
                [medial_axis]=Loop_5(points,punt0,punt1,punt2, punt3, punt5, medial_axis,...
                    p_vector,p_vector_2, informatie, ypoint, ya, w, b, b_2, xv, yv);
            end
        end
    else % second point is no neighboring point
        [medial_axis]=Loop_5(points,punt0,punt1,punt2, punt3, punt5, medial_axis, p_vector,p_vector_1,...
            informatie, ypoint, ya, w, b, b_1, xv, yv);
    end
else % length is less than 20% of the maximal length
    [medial_axis]=Loop_5(points,punt0,punt1,punt2, punt3, punt5, medial_axis,...
        p_vector,p_vector_2, informatie, ypoint, ya, w, b, b_2, xv, yv);
end
end
% Filter out double points in the medial axis. This can happen by one boundary loop geometries.
if length(andere_kring) == 1
    a = 1:length(medial_axis)-1;
    [s1 s2] = find( medial_axis(a) == medial_axis(a+1));
    if length(s1) == 2
        medial_axis = medial_axis(s1+1:end,:)
    else
        if length(s1) == 1
            medial_axis = medial_axis(s1+1:end,:)
        end
    end
end
medial_axis = medial_axis(s1(1)+1:s1(2),:);
end
end
figure(2)
plot(points(:,1),points(:,2))
hold on
plot(medial_axis(:,1), medial_axis(:,2),'or')
set(gca,'DataAspectRatio',[1 1 1]);
MA_tussen = [];
for s = 1:(size(medial_axis,1)-1)
    p_1 = polyfit( medial_axis(s:s+1,1),medial_axis(s:s+1,2),1);
lengte = ( medial_axis(s,3) + medial_axis(s+1,3) )/2;
    MA_tussen = [ MA_tussen ; p_1(1) medial_axis(s,1:2) medial_axis(s+1,1:2) lengte medial_axis(s,4) ];
end
if MA_tussen(1) == size(medial_axis,1)
    MA_tussen = [ MA_tussen ; medial_axis(1,1:4) ];
else
    MA_tussen = [ MA_tussen ; medial_axis(1,1:4) ];
end

% calculating the thickness of every line, by taken the middle of the line
% T and calculate the intersection point of the perpendicular with the
% T geometry. In this way the lines of the geometry that are not parallel are
% T mediated.
MA = [];
s = 1;
if size(MA_tussen,1) == 1 % one boundary loop geometry
    pol = polyfit( [MA_tussen(s,2) MA_tussen(s,6)] ,[MA_tussen(s,3) MA_tussen(s,7)] ,1);
    y_midden = ( MA_tussen(s,4) + MA_tussen(s,8) ) / 2;
    m = min( [ MA_tussen(s,9) MA_tussen(s,10) ]);
    pol = polyfit( [MA_tussen(s,1) MA_tussen(s,3) ] ,[MA_tussen(s,2) MA_tussen(s,4) ] ,1);
    ym = -1/ pol(1);
    if abs(pol(1)) == 1e8
        y_int = y_midden;
        x_int = point(s,1);
        lengte = 10*sqrt( ( x_int -x_midden)^2 + ( y_int - y_midden )^2 )/10;
        lengte = round(lengte)/10;
    elseif pol(1) == 0;
        x_int = x_midden;
        y_int = point(s,2);
        lengte = 10*sqrt( ( x_int -x_midden)^2 + ( y_int - y_midden )^2 )/10;
        lengte = round(lengte)/10;
    else
        ym = -1/ pol(1);
        b_int = y_midden - ym*x_midden;
        x_int = ( ypoint(s,2) - b_int) / ( ym - ypoint(s,1) );
        y_int = ym * x_int + b_int;
        lengte = 10*sqrt( ( x_int -x_midden)^2 + ( y_int - y_midden )^2 )/10;
        lengte = round(lengte)/10;
    end
end
Medial = [ MA_tussen ];
else % two boundary loop geometry
    while s < size(MA_tussen,1)
        if abs( MA_tussen(s,1) - MA_tussen(s+1,1) ) < 2e-2 & abs( MA_tussen(s,6) - MA_tussen(s+1,6) ) < 1e-1
            s = s+1;
            if (s+1) == size(MA_tussen,1)
                MA = [MA ; [MA_tussen(s,2:5) MA_tussen(s+1,6:8) MA_tussen(s,7) MA_tussen(s+1,7) ] ];
                s = size(MA_tussen,1);
                end
        else
            MA = [MA ; [MA_tussen(s,2:5) MA_tussen(s,6:8) MA_tussen(s,7) MA_tussen(s,8) ] ];
            s = s+2;
            end
        end
    end
end
% searching the length Medial = [];
for s = 1:size(MA,1)
    % calculate the middle point of the line
    y_midden = ( MA(s,1) + MA(s,3) ) / 2;
    m = min( [ MA(s,2) MA(s,4) ]);
    pol = polyfit( [MA(s,1) MA(s,3)] ,[MA(s,2) MA(s,4)] ,1);
    ym = -1/ pol(1);
    if pol(1) == 1e8
        y_int = y_midden;
        x_int = point(s,1);
        lengte = 10*sqrt( ( x_int -x_midden)^2 + ( y_int - y_midden )^2 )/10;
        lengte = round(lengte)/10;
    elseif pol(1) == 0;
        x_int = x_midden;
        y_int = point(s,2);
        lengte = 10*sqrt( ( x_int -x_midden)^2 + ( y_int - y_midden )^2 )/10;
        lengte = round(lengte)/10;
    else
        ym = -1/ pol(1);
        b_int = y_midden - ym*x_midden;
        x_int = ( ypoint(s,2) - b_int) / ( ym - ypoint(s,1) );
        y_int = ym * x_int + b_int;
        lengte = 10*sqrt( ( x_int -x_midden)^2 + ( y_int - y_midden )^2 )/10;
        lengte = round(lengte)/10;
    end
end

74
\[ y_{\text{int}} = y_{\text{m}} \times x_{\text{int}} + b_{\text{int}}; \]

\[ \text{lengte} = 10 \times \sqrt{(x_{\text{int}} - x_{\text{midden}})^2 + (y_{\text{int}} - y_{\text{midden}})^2}; \]

\[ \text{lengte} = \text{round}(\text{lengte}) / 10; \]

end

Medial = [Medial; pol MA(s, i:4) 2 * lengte];

end
end
E.2 Medial line

function [p_vector, b ] = medial_line_search( punt2, punt1, punt3, y2point , y1point )
alpha1 = atan( y1point(1) );
a2 = atan( y2point(1) );
if abs(alpha1) < 1e-4
alpha1 = 0;
end
if abs(alpha2) < 1e-4
alpha2 = 0;
end
if (sign( alpha1) * alpha1) == (sign( alpha2 ) * alpha2)
if y1point(1) > 1e8
p_vector = 0;
elseif y1point(1) < 1e-8
p_vector = 1e8;
else
p_vector = (-1 / y1point(1));
end
elseif (abs(alpha1)+ abs(alpha2) ) > ( 0.5*pi - 0.0005 ) && (abs(alpha1) +abs(alpha2)) < ( 0.5*pi + 0.0005 )
phi = 0.25*pi;
if abs(alpha2) < 1e-3
hoek = phi*sign( punt3(1) - punt2(1) )*sign( punt1(2)-punt2(2) );
elseif abs(alpha1) < 1e-3
hoek = phi*sign( punt1(1) - punt2(1) )*sign( punt3(2) - punt2(2) );
else
alpha_min = min( [abs(alpha1) abs(alpha2)])
if alpha_min == abs( alpha1);
alpha = alpha1;
else
alpha = alpha2;
end
if sign( punt3(1) - punt2(1) ) == sign( punt1(1)-punt2(1) )
hoek = alpha - si*phi;
else
hoek = alpha + si*phi;
end
p_vector = tan(hoek);
elseif alpha1 == 0 || alpha2 == 0
if alpha1 == 0
if sign( punt2(1) - punt1(1)) == 1
if sign( punt3(2) - punt2(2) ) == 1
hook = (ps - alpha2)/2;
p_vector = tan( hook );
else
hook = alpha2/2;
p_vector = tan( hook );
end
elseif sign( alpha2 ) == -1
hook = (ps - alpha2)/2;
p_vector = tan( hook );
else
hook = alpha2/2;
p_vector = tan( hook );
end
else
if sign( punt3(2) - punt2(2) ) == 1
if sign( alpha2 ) == -1
hook = (ps - alpha2)/2;
p_vector = tan( hook );
else
hook = alpha2/2;
p_vector = tan( hook );
end
else
if sign( alpha2 ) == 1
hook = (ps - alpha2)/2;
p_vector = tan( hook );
else
hook = alpha2/2;
p_vector = tan( hook );
end
end
end
elseif alpha2 == 0
if sign( punt2(1) - punt3(1)) == 1
if sign( punt1(2) - punt2(2) ) == 1
hook = (ps - alpha1)/2;
p_vector = tan( hook );
else
hook = alpha1/2;
p_vector = tan( hook );
end
elseif sign( alpha1 ) == -1
hook = (ps - alpha1)/2;
p_vector = tan( hook );
else
hook = alpha1/2;
p_vector = tan( hook );
end
else
if sign( alpha1 ) == 1
hook = (ps - alpha1)/2;
p_vector = tan( hook );
else
hook = alpha1/2;
p_vector = tan( hook );
end
end
elseif alpha1 == 0 || alpha2 == 0
if alpha1 == 0
if sign( punt2(1) - punt1(1)) == 1
if sign( punt3(2) - punt2(2) ) == 1
hook = (ps - alpha2)/2;
p_vector = tan( hook );
else
hook = alpha2/2;
p_vector = tan( hook );
end
elseif sign( alpha2 ) == -1
hook = (ps - alpha2)/2;
p_vector = tan( hook );
else
hook = alpha2/2;
p_vector = tan( hook );
end
else
if sign( alpha2 ) == 1
hook = (ps - alpha2)/2;
p_vector = tan( hook );
else
hook = alpha2/2;
p_vector = tan( hook );
end
end
elseif alpha2 == 0
hoek = alpha1/2;
p_vector = tan( hoek );
end
else
  if sign( alpha1 ) == -1
    hoek = (pi - alpha1)/2;
p_vector = -tan( hoek );
  else
    hoek = alpha1/2;
p_vector = tan( hoek );
  end
end
else if sign( punt2(2) - punt2(2) ) == 1
  if sign( alpha1 ) == -1
    hoek = (pi - alpha1)/2;
p_vector = -tan( hoek );
  else
    hoek = alpha1/2;
p_vector = tan( hoek );
  end
end
else if sign( punt1(2) - punt2(2) ) == -1
  if sign( alpha1 ) == -1
    hoek = (pi - alpha1)/2;
p_vector = -tan( hoek );
  else
    hoek = alpha1/2;
p_vector = tan( hoek );
  end
end
if sign( punt2(1) - punt1(1) ) == sign( punt1(1) )
  if sign( punt2(2) - punt1(2) ) == sign( punt2(2) - punt3(2) )
    phi = max( [alpha1 alpha2] ) - min( [alpha1 alpha2] )/2;
    hoek = max( [alpha1 alpha2] ) - phi;
p_vector = tan( hoek );
  else
    phi = (alpha2-alpha1)/2;
    hoek = phi - sign(alpha2)*alpha2;
p_vector = tan( hoek );
  end
else
  phi = 0.5*pi - ( abs( alpha1) + abs( alpha2) )/2;
  if alpha2 < 0
    hoek = alpha2 - phi;
  else
    hoek = alpha2 + phi;
  end
  p_vector = tan( hoek );
end
else
  phi = 0.5*pi - ( abs( alpha1) + abs( alpha2) )/2;
  if alpha2 > 0.5*pi-0.05;
    hoek = alpha2 - phi;
  else
    if alpha2 < 0
      hoek = alpha2 + phi;
    else
      hoek = alpha2 - phi;
    end
  end
  p_vector = tan( hoek );
end
else
  phi = (abs(alpha2)+abs(alpha1))/2;
  if alpha2 < 0
    hoek = alpha2 + phi;
  else
    hoek = alpha2 - phi;
  end
  p_vector = tan( hoek );
end
else
  if sign( punt2(1) - punt1(1) ) == 0 || sign( punt2(1) - punt3(1) )== 0
    if sign( punt2(2) - punt1(2) ) == sign( punt2(2) - punt3(2) )
      phi = 0.5*pi - ( abs( alpha1) + abs( alpha2) )/2;
      if alpha2 > 0
        hoek = alpha2 - phi;
      else
        hoek = alpha2 + phi;
      end
      p_vector = tan( hoek );
    else
      if alpha2 < 0
        hoek = alpha2 + phi;
      else
        hoek = alpha2 - phi;
      end
      p_vector = tan( hoek );
    end
    else
      phi = 0.5*pi - ( abs( alpha1) + abs( alpha2) )/2;
      if alpha2 < 0
        hoek = alpha2 - phi;
      else
        hoek = alpha2 + phi;
      end
      p_vector = tan( hoek );
    end
end
\[ p_{\text{vector}} = \tan(\ \text{hoek}); \]

end

end

d = pant2(2) - p_{\text{vector}} \cdot pant2(1); 

E.3 Test 1

function \([w_{\text{loop}1}, w_{\text{loop}2}, {\text{l}ij}jen\_text, \text{medial\_axis},\text{tweed\_punt}] = \text{Loop}_1(ypoint, ya, point, medial\_axis,.. \ w_{\text{loop}1}, w_{\text{loop}2}, s_{\text{axis}}; y_{\text{axis}}; \text{punt}_2; \text{punt}_3; \text{punt}_4; \text{punt}_5; {\text{l}ij}jen\_text) \]
\[ b = \text{punt2}(2) - p_{\text{vector}} \cdot \text{punt2}(1); \]

end

end

if ypoint(punt1,1) == 1e8
  x_punt1 = point(punt1,1);
  y_punt1 = y_snij;
elseif ypoint(punt1,1) == 0
  x_punt1 = x_snij;
  y_punt1 = point(punt1,2);
else
  x_punt1 = (ypoint(punt1,2) - b_punt2) / (ya(punt1) - ypoint(punt1,1));
  y_punt1 = ya(punt1) * x_punt1 + b_punt2;
end

lengte_1 = sqrt((x_snij - x_punt1)^2 + (y_snij - y_punt1)^2);

end

b_punt5 = y_snij - ya(punt5) * x_snij;
if ypoint(punt5,1) == 1e8
  x_punt5 = point(punt5,1);
  y_punt5 = y_snij;
elseif ypoint(punt5,1) == 0
  x_punt5 = x_snij;
  y_punt5 = point(punt5,2);
else
  x_punt5 = (ypoint(punt5,2) - b_punt5) / (ya(punt5) - ypoint(punt5,1));
  y_punt5 = ya(punt5) * x_punt5 + b_punt5;
end

lengte_5 = sqrt((x_snij - x_punt5)^2 + (y_snij - y_punt5)^2);
if abs(lengte_2 - lengte_5) < 0.001
  medial_axis = [medial_axis; x_snij y_snij 2*lengte_2 punt2 punt5];
  hold on
  plot(x_snij, y_snij, 'ok');
  aanpassing van de vectoren w_loop
  \[ z3 z4 \] = find(w_loop_2 == punt5);
  w_loop_1 = w_loop_1(2:end);
  if length(z4) == 0
    z4 = 0;
  end
  w_loop_2 = \[ w_loop_2(1:z4-1) w_loop_2(z4+1:end) \];
  tweede_punt = punt5;
else
  % controleren of we met een verdikking te maken hebben
  if ypoint(punt1,1) == ypoint(punt4,1) | ypoint(punt2,1) == ypoint(punt4,1)
    if point(punt2,2) == point(punt4,2) | ypoint(punt2,2) == ypoint(punt4,2)
      tweede_punt = punt4;
      {\text{l}ij}jen\_text = 1;
    end
  elseif ypoint(punt1,1) == ypoint(punt2,1) | ypoint(punt2,1) == ypoint(punt5,1)
    if point(punt1,2) == point(punt2,2) | ypoint(punt1,2) == ypoint(punt2,2)
      tweede_punt = punt2;
      {\text{l}ij}jen\_text = 1;
    end
  end

  if {\text{l}ij}jen\_text == 0
    d1 = abs(point(punt2,1) - point(punt5,1));
    d2 = abs(point(punt2,2) - point(punt5,2));
    s_{\text{axis}} = max(point(punt2,1) point(punt5,1)) + 0.5*d1;
    y_{\text{axis}} = max(point(punt2,2) point(punt5,2)) + 0.5*d2;
    \[ p_{\text{vector}} = \text{ya}(punt5) - ya(punt5) \cdot s_{\text{axis}}; \]
    u_{\text{vector}} = \text{ya}(punt5) - ya(punt5) \cdot s_{\text{axis}};
    medial_axis = \{ medial_axis; s_{\text{axis}}; y_{\text{axis}}\};
    hold on
    plot(x_snij, y_snij, 'ok');
    tweede_punt = punt5;
  end

end

\[ w_{\text{loop}2} = \{ w_{\text{loop}2} \in d1 - w_{\text{loop}2}(2*d1) \}; \]

end

end
E.4 Test 2

```matlab
function [in, on, x_p, y_p, p_vector_2, b_2, tweede_punt, punt4, punt5, punt6, andere_kring, b, p_vector ];

straal_gebied = (points([1:w-2 w+2:end],1)-punt_start(1)).^2 + (points([1:w-2 w+2:end],2)-punt_start(2)).^2;

[q1 q2] = min(straal_gebied);
if q2 == w
    if w == 1;
        q2 = q2+2;
    else
        q2 = q2+3;
    end
end
tweede_punt = q2;
if tweede_punt == andere_kring;
    punt4 = tweede_punt-1;
    punt5 = tweede_punt;
    punt6 = 1;
elseif tweede_punt == 1;
    punt4 = andere_kring;
    punt5 = tweede_punt;
    punt6 = tweede_punt+1;
else
    punt4 = tweede_punt-1;
    punt5 = tweede_punt;
    punt6 = tweede_punt+1;
end

ypoint = ypoint( punt5,: );
y1point = ypoint( punt4,: );
[p_vector_2 b_2] = medial_line_search( points(punt5,:), points(punt6,:), points(punt5,:), y2point, y1point );

x_p = ( b - b_2 ) / ( p_vector_2 - p_vector);
y_p = p_vector*x_p + b;

[xv,yv] = impolygon(x_p,y_p,xv,yv);
```

79
function (medial_axis)=Loop_5(points,punt0,punt1,punt2, punt3, punt5, medial_axis, p_vector,p_vector_2, informatie, ppoint, xe, yb, x2, y2, yv ) 

LL=[]; 
snij =[]; 
if (abs(p_vector) >= 100) & (abs(p_vector_2) >= 100)
   b = points(punt2,2) - 500*points(punt2,1); 
b_2 = points(punt5,2) - 500*points(punt5,1); 
p_vector = 100; 
p_vector_2 = 100; 
end 
if abs( p_vector - p_vector_2 )/abs(p_vector) < 1e-1 & abs(b - b_2)/abs(b) < 1e-1 % controleren of dat middenlijnen gelijk lopen en ongeveer zelfde beginpunt hebben
   x_p = (max( [points(punt2,1) points(punt5,1)] )-min( [points(punt2,1) points(punt5,1)] ))/2 + min( [points(punt2,1) points(punt5,1)] ); 
y_p = (max( [points(punt2,2) points(punt5,2)] )-min( [points(punt2,2) points(punt5,2)] ))/2 + min( [points(punt2,2) points(punt5,2)] ); 
   medial_axis = [ medial_axis; x_p y_p 0 punt2 punt5]; 
   figure(1) 
   plot( x_p , y_p ,'ok');
else % middenlijnen zijn niet gelijk
   KL = ( max(abs(diff(points(:,1)))) + max(abs(diff(points(:,2)))) )/2; 
   max_x = max(abs(diff(points(:,1)))) 
   max_y = max(abs(diff(points(:,2)))) 
   x_new = 0.01; 
   y_new = p_vector* x_new;
   [in, on] = inpolygon((points(punt2,1)+x_new),(points(punt2,2)+y_new),xv,yv);
in
   if in == 0 
      if p_vector < 0
         xmini = points(punt2,1) -KL; 
xmini = points(punt2,1); 
ymini = points(punt2,2) + KL; 
else if p_vector > 1.5
         xmini = points(punt2,1) - KL; 
xmini = points(punt2,1) + KL; 
ymini = points(punt2,2) + KL; 
else
         xmini = points(punt2,1) - KL; 
xmini = points(punt2,1); 
ymini = points(punt2,2) - KL*p_vector; 
end
   else
      if p_vector < 0
         xmini = points(punt2,1); 
xmini = points(punt2,1) + KL; 
ymini = points(punt2,2) + KL*p_vector; 
else if p_vector > 1.5
         xmini = points(punt2,1) - KL; 
xmini = points(punt2,1) + KL; 
ymini = points(punt2,2) - KL; 
else
         xmini = points(punt2,1); 
xmini = points(punt2,1) + KL; 
ymini = points(punt2,2) + KL*p_vector; 
end
   end
   [test1 test2 ] = find( ( xmini <= informatie(:,3) ) & (xmaxi >= informatie(:,4) )); 
   [test3 test4 ] = find( ( ymini <= informatie(:,5) ) & (ymaxi >= informatie(:,6) )); 
sd = 0; 
sd_lijst = []; 
for a1 = 1:length(test3)
   for a2 = 1:length(test1)
      if test1(a2) == test3(a1)
         if test1(a2) ~= punt2
            if test1(a2) ~= punt1
               if test1(a2) ~= punt0
                  if test1(a2) ~= punt3
                     sd = sd +1; 
                  end
               end
            end
         end
      end
   end
end

end

end

end

end

end

end

end

sd

sd_lijst

test1
test2
test3
test4

for c = 1:length( sd_lijst )
   lijn = sd_lijst(c);

80
if abs(p_vector) < 1e-6
    y_intersect = points(w,2);
    x_intersect = (y_intersect - b) / p_vector;
else
    if abs(p_vector) == 1e8
        x_intersect = points(w,2);
        y_intersect = ypoint(lijn,1)* x_intersect + ypoint(lijn,2);
    else
        if abs( ypoint(lijn,1)) == 1e8
            x_intersect = points(lijn,1);
            y_intersect = p_vector*x_intersect + b;
        elseif abs( p_vector ) < 1e-6
            y_intersect = points(lijn,2);
            x_intersect = (y_intersect - b) / p_vector;
        else
            x_intersect = ( ypoint(lijn,2) - b)/ ( p_vector - ypoint(lijn,1) );
            y_intersect = p_vector*x_intersect + b;
        end
    end
end
lengte = sqrt( (x_intersect-points( punt2,1 ) )^2 + ( y_intersect - points( punt2,2 ) )^2 );
if lengte < 1e-6
    lengte = inf;
eend
LL = [LL ;lengte];
snij= [snij; x_intersect , y_intersect];
end
zoek = 0;
while test < 1
    [x1 , x2] = find( LL == sorteer_lengte( zoek ) );
    x_p = min([ points(punt2,1) snij(x1,1)]) + max([points(punt2,1) points(punt1,1)])/2 + min([ points(punt2,1) points(punt1,1) ]);
    y_p = min([ points(punt2,2) snij(x1,2)]) + max([points(punt2,2) points(punt1,2)])/2 + min([ points(punt2,2) points(punt1,2) ]);
    medial_axis = [ medial_axis , x_p, y_p ,punt2 punt3];
    figure(1)
    plot( x_p, y_p, 'ok');
    elseif im == 1 & ni==1
        b_sni ,k , y_p = ypoint( sd_lijst(z1(1)) ) * x_p;
        u_1 = abs(( b_sni,k - ypoint( sd_lijst(z1(1))) ) / ( ypoint( sd_lijst(z1(1)) ) - ypoint( sd_lijst(z1(1)) ) ));
        y_1 = ypoint( sd_lijst(z1(1)) ) * u_1 + b_sni,k;
        lengte_l = sqrt( ( x_1 -x_p)^2 + ( y_1 -y_p)^2 );
        b_sni,k = ypoint( sd_lijst(z1(1)) ) / ( ypoint( sd_lijst(z1(1)) ) - ypoint( sd_lijst(z1(1)) ));
        u_2 = abs(( b_sni,k - ypoint( sd_lijst(z1(2))) ) / ( ypoint( sd_lijst(z1(2))) - ypoint( sd_lijst(z1(2)) ) ));
        y_2 = ypoint( sd_lijst(z1(2)) ) * u_2 + b_sni,k;
        lengte_k = sqrt( ( x_2 -x_p)^2 + ( y_2 -y_p)^2 );
    end
    else
        zoom = zoom+1;
        if zoom == length( sorteer_lengte) -1
            medal_axis = [ medial_axis , x_p, y_p , punt2 punt5 ];
            figure(1)
            plot( x_p, y_p, 'ok');
            test = 1;
        else
            zoom = zoom+1;
        end
    end
end
E.6 Test 4

```matlab
function [as,Lengte]= Loop6(ya,points, ypoint, punt1,punt2,punt4,punt5,y_p,x_p)
    punt_4 = [ punt1 punt2 punt4 punt5 ];
    for d = 1:4
        if ya( punt_4(d) ) == 1e8;
            x_1 = x_p;
            y_1 = points( punt_4(d),2 );
        elseif ya( punt_4(d) ) == 0;
            x_1 = points( punt_4(d),1 );
            y_1 = y_p;
        else
            b_a = y_p - ya( punt_4(d) ) * x_p;
            x_1 = ( ypoint( punt_4(d),2 ) - b_a ) / ( ya( punt_4(d) ) - ypoint( punt_4(d),1 ) );
            y_1 = ya( punt_4(d) ) * x_1 + b_a;
        end
        Lengte(d) = sqrt( ( x_1 -x_p )^2 + ( y_1 - y_p)^2 );
    end
end

% controleer of de afstanden verschillen niet te groot zijn.
as = 0.2*( max(Lengte) );
```

% controle loop controle gelijke middenlijnen
function [medial_axis]=Loop7(Lengte,points, medial_axis, punt2, punt5,x_p, y_p);

if abs(Lengte(1) - Lengte(3)) < 1e-3 | abs(Lengte(1) - Lengte(4)) < 1e-3
    medial_axis = [ medial_axis; x_p y_p 2*Lengte(1) punt2 punt5];
    figure(1)
    plot(x_p, y_p, 'ok');
else
    x_p = (max([points(punt2,1) points(punt5,1)]) - min([points(punt2,1) points(punt5,1)]))/2 + min([points(punt2,1) points(punt5,1)]);
    y_p = (max([points(punt2,2) points(punt5,2)]) - min([points(punt2,2) points(punt5,2)]))/2 + min([points(punt2,2) points(punt5,2)]);
    medial_axis = [ medial_axis; x_p y_p 0 punt2 punt5];
    figure(1)
    plot(x_p, y_p, 'ok');
end
function [medial_axis]=Loop8(points,punt0,punt1,punt2, punt3, punt5, medial_axis, p_vector,p_vector_2, informatie, ypoint, ya, w, b, b_2, xv, yv);

LL = [];
s = 1;

KL = ( max(abs(diff(points(:,1)))) + max(abs(diff(points(:,2)))) )/4;
x_new = 0.01;
y_new = p_vector * x_new;

[in, on] = inpolygon((points(punt2,1)+x_new),(points(punt2,2)+y_new),xv,yv);

if in == 0
    if p_vector < 0
        xmini = points(punt2,1) - KL;
xmaxi = points(punt2,1);
ymini = points(punt2,2);
        ymaxi = points(punt2,2) - KL*p_vector;
selif p_vector > 1.5;
xmini = points(punt2,1) - KL;
xmaxi = points(punt2,1) + KL;
ymini = points(punt2,2) - KL;
        ymaxi = points(punt2,2) + KL;
    else
        xmini = points(punt2,1) - KL;
xmaxi = points(punt2,1);
ymini = points(punt2,2) - KL*p_vector;
        ymaxi = points(punt2,2);
    end

else
    if p_vector < 0
        xmini = points(punt2,1);
xmaxi = points(punt2,1)+KL;
ymini = points(punt2,2)+ KL*p_vector;
xmaxi = points(punt2,2);
        ymaxi = points(punt2,2) + KL;
    elseif p_vector > 1.5;
xmini = points(punt2,1) - KL;
xmaxi = points(punt2,1) + KL;
ymini = points(punt2,2) - KL;
xmaxi = points(punt2,2) + KL;
    else
        xmini = points(punt2,1);
xmaxi = points(punt2,2) - KL;
ymini = points(punt2,2) - KL*p_vector;
        ymaxi = points(punt2,2) + KL*p_vector;
    end

end

[end test2 ] = find( ( xmini <= informatie(:,3) ) & (xmaxi >= informatie(:,4) ));
[end test4 ] = find( ( ymini <= informatie(:,5) ) & (ymaxi >= informatie(:,6) ));

sd = 0;
sd_lijst = [];

for a1 = 1:length(test3)
    for a2 = 1:length(test1)
        if test1(a2) == test3(a1)
            if test1(a2) ~= punt2
                if test1(a2) ~= punt0
                    if test1(a2) ~= punt1
                        if test1(a2) ~= punt3
                            sd = sd +1;
                            sd_lijst = [sd_lijst ; test1(a2)];
                        end
                    end
                end
            end
        end
    end
end

for c = 1:length( sd_lijst )
    lijn = sd_lijst(c);

    if abs(p_vector) < 1e-6
        x_intersect = points(w,1);
y_intersect = p_vector*x_intersect + b;
    elseif abs(p_vector) == 1e8
        x_intersect = points(lijn,1)* y_intersect + ypoints(lijn,2);
    else
        if abs( ypoint(lijn,1)) == 1e8
            x_intersect = points(lijn,1);
y_intersect = p_vector*x_intersect + b;
        elseif abs( p_vector ) < 1e-6
            y_intersect = points(lijn,2);
x_intersect = (y_intersect - b) / p_vector;
        elseif abs( ypoint(lijn,2) - b) / p_vector;
            y_intersect = ypoint(lijn,2);
            x_intersect = ypoint(lijn,1) - (ypoint(lijn,1) - b) / p_vector;
    end

    lengte = sqrt(( x_intersect-points(punt2,1) )^2 + ( y_intersect - points(punt2,2))^2);
end

if lengte < 1e-6

end

84
lengte = inf;
end
LL = [LL length];
snij= [snij; x_intersect , y_intersect];
end
sorteer_length = sort(LL);
test = 0;
zoek = 1;
while test < 1
    [z1 , z2] = find( LL == sorteer_length( zoek ) );
    x_p = min([ points(punt2,1) snij(z1(1),1)] ) + abs((snij(z1(1),1)-points(punt2,1))/2);
    y_p = min([ points(punt2,2) snij(z1(1),2)] ) + abs((snij(z1(1),2)-points(punt2,2))/2);
    for=[points(:,1); points(1,1)];
    y=[points(:,2); points(1,2)];
    (in, on) = inpolygon(x_p,y_p,xv,yv);
    if in == 0
        zoek = zoek + 1;
        if zoek == length( sorteer_length ) + 1
            test = 1;
        else
            b_snij_l = y_p - ya( sd_lijst(z1(1)) *x_p);
            x_l = abs(( b_snij_l - ypoint(( sd_lijst(z1(1)) ) ) / ( ypoint(( sd_lijst(z1(1)) ) ) - ya( sd_lijst(z1(1)) ) ) ));
            y_l = ypoint(( sd_lijst(z1(1)) ) ) *x_l + b_snij_l;
            lengte_l = sqrt( ( x_l -x_p)^2 + ( y_l -y_p)^2 );
            b_snij_k = y_p - ya( punt2 ) *x_p;
            x_k = abs(( b_snij_k - ypoint(( punt2 ) ) ) / ( ypoint(( punt2 ) ) - ya( punt2 ) ));
            y_k = ypoint(( punt2 ) ) *x_k + b_snij_k;
            lengte_k = sqrt( ( x_k -x_p)^2 + ( y_k -y_p)^2 );
            if abs( lengte_k - lengte_l) < 0.001
                medial_axis = [ medial_axis; x_p y_p 2*lengte_k punt2 punt5 ];
                figure(1)
                plot( x_p , y_p , 'ok');
                test = 1;
            else
                zoek = zoek +1;
                if zoek == length( sorteer_length ) + 1
                    test = 1;
                else
                    b_snij_l = y_p - ya( sd_lijst(z1(1)) *x_p);
                    x_l = abs(( b_snij_l - ypoint(( sd_lijst(z1(1)) ) ) / ( ypoint(( sd_lijst(z1(1)) ) ) - ya( sd_lijst(z1(1)) ) ) ));
                    y_l = ypoint(( sd_lijst(z1(1)) ) ) *x_l + b_snij_l;
                    lengte_l = sqrt( ( x_l -x_p)^2 + ( y_l -y_p)^2 );
                    b_snij_k = y_p - ya( punt2 ) *x_p;
                    x_k = abs(( b_snij_k - ypoint(( punt2 ) ) ) / ( ypoint(( punt2 ) ) - ya( punt2 ) ));
                    y_k = ypoint(( punt2 ) ) *x_k + b_snij_k;
                    lengte_k = sqrt( ( x_k -x_p)^2 + ( y_k -y_p)^2 );
                    if abs( lengte_k - lengte_l) < 0.001
                        medial_axis = [ medial_axis; x_p y_p 2*lengte_k punt2 punt5 ];
                        figure(1)
                        plot( x_p , y_p , 'ok');
                        test = 1;
                    else
                        zoek = zoek +1;
                        if zoek == length( sorteer_length ) + 1
                            test = 1;
                        else
                            b_snij_l = y_p - ya( sd_lijst(z1(1)) *x_p);
                            x_l = abs(( b_snij_l - ypoint(( sd_lijst(z1(1)) ) ) / ( ypoint(( sd_lijst(z1(1)) ) ) - ya( sd_lijst(z1(1)) ) ) ));
                            y_l = ypoint(( sd_lijst(z1(1)) ) ) *x_l + b_snij_l;
                            lengte_l = sqrt( ( x_l -x_p)^2 + ( y_l -y_p)^2 );
                            b_snij_k = y_p - ya( punt2 ) *x_p;
                            x_k = abs(( b_snij_k - ypoint(( punt2 ) ) ) / ( ypoint(( punt2 ) ) - ya( punt2 ) ));
                            y_k = ypoint(( punt2 ) ) *x_k + b_snij_k;
                            lengte_k = sqrt( ( x_k -x_p)^2 + ( y_k -y_p)^2 );
                            if abs( lengte_k - lengte_l) < 0.001
                                medial_axis = [ medial_axis; x_p y_p 2*lengte_k punt2 punt5 ];
                                figure(1)
                                plot( x_p , y_p , 'ok');
                                test = 1;
                            else
                                zoek = zoek +1;
                                if zoek == length( sorteer_length ) + 1
                                    test = 1;
                                else
                                    b_snij_l = y_p - ya( sd_lijst(z1(1)) *x_p);
                                    x_l = abs(( b_snij_l - ypoint(( sd_lijst(z1(1)) ) ) / ( ypoint(( sd_lijst(z1(1)) ) ) - ya( sd_lijst(z1(1)) ) ) ));
                                    y_l = ypoint(( sd_lijst(z1(1)) ) ) *x_l + b_snij_l;
                                    lengte_l = sqrt( ( x_l -x_p)^2 + ( y_l -y_p)^2 );
                                    b_snij_k = y_p - ya( punt2 ) *x_p;
                                    x_k = abs(( b_snij_k - ypoint(( punt2 ) ) ) / ( ypoint(( punt2 ) ) - ya( punt2 ) ));
                                    y_k = ypoint(( punt2 ) ) *x_k + b_snij_k;
                                    lengte_k = sqrt( ( x_k -x_p)^2 + ( y_k -y_p)^2 );
                                    if abs( lengte_k - lengte_l) < 0.001
                                        medial_axis = [ medial_axis; x_p y_p 2*lengte_k punt2 punt5 ];
                                        figure(1)
                                        plot( x_p , y_p , 'ok');
                                        test = 1;
                                    else
                                        zoek = zoek +1;
                                        if zoek == length( sorteer_length ) + 1
                                            test = 1;
                                        else

Bibliography
