Algorithm for modeling crack propagation in three dimensional finite element models of planar geometries

S.M. Kleinendorst (Sandra)
MT 11.15

June 2011
Abstract

In order to predict the consequences of crack growth, it is important to model the crack propagation. The direction and length of an increment of crack propagation can be calculated with finite element software. The program described in this report manipulates the geometry in such a way that a new mesh is created on which calculations can be performed again to determine the next increment of crack propagation.

The geometry is divided in elements, tetrahedra, by the program TetGen. The elements are stored in three matrices: one for the nodes and their coordinates, one for the triangular face elements with the corresponding node numbers and one for the three dimensional volume elements with the four corresponding node numbers. The program manipulates these matrices in order to change the geometry. First the sides of the crack are defined by the program and then the new crack surfaces are inserted. When viewed from the side, the crack tip is surrounded by elements. The program searches for the elements that are cut by the propagated crack, adds new nodes and splits the elements by changing node numbers and adding new elements. The algorithm used for doing so is explained.

Furthermore applications of the program are given, such as the Arcan test.

Finally recommendations for improving the program are given.
Contents

1 Introduction 5

2 The algorithm of crack propagation 7
  2.1 Introduction .................................................. 7
  2.2 Crack propagation ............................................. 8
  2.3 The algorithm of crack propagation on the sides ............... 9
    2.3.1 Adding new nodes ........................................ 10
    2.3.2 Splitting elements ...................................... 11
    2.3.3 Different situations and their corresponding actions .... 12
  2.4 Insertion of the new crack surface ............................. 14
    2.4.1 Nodes in the crack tip ................................ 15
    2.4.2 Defining the crack surface .............................. 16

3 Applications of the crack propagation program 19
  3.1 An irregular crack in a simple specimen ..................... 19
  3.2 Arcan test .................................................. 20

4 Conclusion 23

Bibliography 24

A Sorting of the nodes on the edge 27

B Other possibilities of defining propagation situations 29
Chapter 1

Introduction

In manufacturing products a lot of different operations are carried out. The methods of manufacturing depend for a great deal on the material of the desired product. For example polymers are processed to the desired form in liquid state. Examples of processes related to polymer processing are injection molding, rotational molding and extrusion. To deform metals in liquid state high temperatures are required, much higher than the temperatures needed for manufacturing plastics. For large-scale manufacturing a lot of energy is needed, so it is expensive. This is why metals are processed when they are relatively cold. Examples of operations on metal products are rolling and deep drawing. During rolling large deformations occur, however the materials stays continuous: no holes or cracks appear. Other operations on metals include blanking and score forming. These operations are based on fracturing the material. Large pressure is applied to a small piece of material, so that the material eventually starts to crack. When the machine continues to exert force the crack propagates through the material. In which direction the crack is propagating depends on the damage of the material. When the material is deformed voids appear on a microscopic scale in the material. The number of voids in a certain volume can be described by a certain 'damage variable'. The larger this number is, the more likely it is that the crack propagates in that direction. The direction of the crack is important to know, because the crack forms the edge of the product. When this edge is not smooth, other operations are needed to smoothen the surface of the product. This can be avoided when the influence of the operation conditions on the crack surface is known. Then the operation can be manipulated in such a way that the crack surface is as smooth as possible.

Therefore it is important to study and model the crack propagation in a material under certain circumstances. Commercial software exists to perform finite element analysis to calculate stresses and strains on arbitrary geometries. However this software is useful only for continua. In case of crack propagation the software cannot be used. The problem is that the geometry changes during deformation, because the crack has propagated. Therefore to make a model of crack propagation an algorithm in mathematical software has to be written in order to change the mesh every step. In the model described in this report the mathematical software package Matlab is used along with a software package that creates tetrahedral elements in a given geometry, TetGen. In a first step of making a model the forces and stresses are not important. The distance of crack propagation per increment of deformation is considered to be known. A Matlab algorithm manipulates the file that is given to TetGen every step, so that the geometry is updated. Furthermore the history of the deformation stays intact.

The geometry considered is divided into tetrahedral finite elements for the mechanical analysis, as can be seen in figure 1.1a. In figure 1.1b a cut-out view of the discretised geometry is shown. The method used to model the crack propagation is based on splitting the elements of which the geometry consists. The elements are stored in three matrices: one for the nodes and their coordinates, one for the triangular face elements and their nodes and one for the tetrahedral three dimensional elements and their nodes. Our algorithm manipulates these matrices in such way that when we make a new mesh using TetGen, the crack has propagated.

5
Chapter 2 describes the algorithm that manipulates the matrices in order to change the geometry of the specimen. At first the front and back surface of the specimen are treated. The sides of the crack are observed from this point of view. A so-called crack vector is defined that describes the direction and length of the crack propagation. On the surface the crack tip is surrounded by triangular elements. The program searches all elements that are crossed by the crack vector and splits them. Section 2.3 describes how the program finds the elements that are cut, where new nodes are inserted and how the elements are split.

Once the sides of the propagated crack have been defined, the next step is to create the three-dimensional crack surfaces. Section 2.4 explains what method is used to insert the propagated crack surfaces. When the algorithm has manipulated the matrices in such a way that the crack has propagated, the new mesh can be used for new stress and strain calculations to determine the direction and length of a new increment of crack propagation. The program can be used again to model the new mesh.

In chapter 2 only the specimen of figure 1.1 is used to illustrate the algorithm and only one increment of crack propagation is discussed. Chapter 3 shows that the program can be used repeatedly to obtain a more realistic crack. Also a real example is shown: the Arcan test. The specimen of the Arcan test has a specific geometry and by varying the direction of the load on this specimen different crack paths occur. The results have been predicted in two dimensions already and now they are modeled in three dimensions. In the conclusion recommendations to improve the program are given.

The algorithm in this report limits itself to planar geometries. It can be extended to be useful for geometries with curved surfaces by projecting the curved surface on a flat surface. However the report is not extended to that.
Chapter 2

The algorithm of crack propagation

2.1 Introduction

This chapter explains the algorithm that is used to model crack propagation in a specimen of a planar geometry. The algorithm consists of two parts: first the surface elements on the front and back of the specimen are split and then the extended crack surfaces are inserted. Splitting the face elements is done in several steps. Section 2.2 explains the method that is used to split these elements. Section 2.3 explains the algorithm that realizes the desired result. The algorithm is based on the method of section 2.2. In section 2.4 the extended crack surfaces (the upper and lower part of the notch) are inserted.

The specimen used in this chapter is a rectangular thick plate, as was seen in figure 1.1. However, other

![Figure 2.1: Example geometry; colors indicate different boundary markers](a) Marked surfaces (b) Legend
geometries can also be manipulated by the program, as can be seen in chapter 3. The program TetGen divides the geometry into a large number of elements. These elements are tetrahedra and thus the elements on the surfaces are triangles, see figure 1.1. For modeling the crack propagation the elements should be split. This splitting occurs by a Matlab program that manipulates the matrices in which the coordinates and numbers of all nodes and elements are stored. When these matrices are given to TetGen after manipulation a new mesh is made with the crack propagated slightly. Calculations to determine the new direction of the crack can be performed on this mesh. Then the matrices can be manipulated again and a new mesh can be made again.

The first step in the algorithm to change the matrices is to manipulate the surface elements. Therefore TetGen is forced to give so called 'boundary markers' to the face elements on a certain surface of the geometry, so that Matlab can recognize the different surfaces of the specimen. The different boundary markers are visualized in figure 2.1. The program starts by splitting the surface elements on surface 1, the red surface in figure 2.1a, and the back surface. These two dimensional operations use the same algorithm that is explained in this chapter.

### 2.2 Crack propagation

Before manipulating the element, face and node matrices, the direction and length of the crack path have to be known. The crack propagation can be represented by a vector. This vector changes every time step and can be calculated by stress and strain calculations on the geometry taking into account the boundary conditions. However, for this study these calculations are not performed and the crack path is considered to be known. The crack vector, as it will be called from now on, starts at the old crack tip. For the first step the number of the corresponding node on the crack tip has to be found manually. For all next time steps the new crack tip node can be used, which is the newest added node in the node matrix. It is not hard to find which node number the node on the original crack tip has, because the geometry, including the prescribed notch, is described by the user. The crack propagation vector might look like sketched in figure 2.2.

![Figure 2.2: Crack propagation. The green spot indicates the start of the crack, the old crack tip. The purple arrow indicates the crack vector. The red spots indicate where new nodes have to be placed.](image)

The green spot indicates the old crack tip and the purple arrow indicates the crack vector. At the
end of the crack vector the new crack tip is found. The red spots indicate the places where the crack vector intersects element edges and new nodes have to be added. The algorithm has to find the places where the crack vector intersects an element boundary. In section 2.3.1 it is explained how to find them. Section 2.3.2 explains the actual splitting of the surface elements. The crack vector determines how many elements are crossed and where the new crack tip appears. When an element is crossed there are several possibilities for what will happen to that element. These possibilities and the actions that have to be done in every situation are explained in section 2.3.3.

2.3 The algorithm of crack propagation on the sides

In this section an overview of the routine to model the crack propagation is given. It is important to emphasize the fact that the crack vector always starts at the crack tip node. This node is surrounded by a number of elements, see figure 2.3. The first step of the algorithm is to find these elements and put them in a matrix. Then for each of these elements the program checks what will happen. There are several possibilities, which will be explained in section 2.3.3. However it is most important to distinguish between the crack vector crossing the element and the crack vector not passing the element. When the element through which the crack propagates has been found, a new node has to be added at the edge of this element. Furthermore the element should be split.

![Figure 2.3: The crack tip node (red spot) is surrounded by several elements. The element through which the crack propagates has to be found.](image)

Next the element that is adjacent to this element, i.e. the element through which the crack propagates next, has to be split into two, see figure 2.4.

![Figure 2.4: The element next to the just evaluated element has to be split to obtain the same situation as before. The green vector \( \vec{c} \) indicates the crack vector, which had started in the crack tip node. Now the vector has been reduced and starts in the new crack tip node: the red spot.](image)
This is because then the same situation as before occurs: the crack vector starts from the crack tip node and that node is surrounded by elements. In this case there are only two elements through which the crack can possibly propagate: the two elements just defined by splitting the neighbouring element.

The steps are repeated, until an element is found in which the crack vector stops. This is one of the special situations to which section 2.3.3 pays attention. When this element is found the algorithm stops. Then a new crack vector can be given to the program and the routine starts all over again.

2.3.1 Adding new nodes

Around the crack tip node several elements are located. Every element can be described by two vectors starting in the crack tip node and a line $\ell$ connecting these two vectors. This can be seen in figure 2.5. In this figure also the crack vector is shown. Vector $\vec{a}$ describes the place where the new node has to be added. Vector $\vec{a}$ reaches to line $\ell$, since the new node is located on this line.

Therefore there are two equations describing this vector:

$$\vec{a} = \alpha \vec{s}_1 + (1 - \alpha) \vec{s}_2$$  \hspace{1cm} (2.1a)
$$\vec{a} = \beta \vec{c}.$$  \hspace{1cm} (2.1b)

where $\alpha$ and $\beta$ are constants which are to be determined. When $\alpha$ equals 0 the crack vector overlaps segment vector $\vec{s}_2$ and when $\alpha$ equals 1 the crack vector overlaps segment vector $\vec{s}_1$.

The situation is two dimensional, so when equations 2.1a and 2.1b are elaborated for the two dimensions, two equations and two unknowns are obtained: $\alpha$ and $\beta$. This can be solved for $\alpha$ and $\beta$ and then the place of the new node can be described by $\vec{a} = \beta \vec{c}$.

Now the new node with the corresponding coordinates can be added to the node matrix.

![Figure 2.5: Face element defined by segment vectors $\vec{s}_1$ and $\vec{s}_2$. Linear interpolation defines line $\ell$ in the face element.](image)

Also the crack tip node has to be split and the existing crack tip node has to be moved up slightly and the duplicated node down slightly. Therefore two new vectors are defined perpendicular to the crack vector, but in the same plane. The first vector, $\vec{p}_1$, is attached to segment vector $\vec{s}_1$. The vector is determined by calculating the cross product of the crack vector with the first segment vector $\vec{s}_1$ and then taking the cross product of this resulting vector with the crack vector.

The second vector, $\vec{p}_2$, is attached to $\vec{s}_2$ and is determined by $(\vec{c} \times \vec{s}_2) \times \vec{c}$. This can be seen in figure 2.6. The nodes are moved along these vectors by a pre-determined value.

Concluding, two new nodes are added: one on line $\ell$ of the element and one as the duplication of the crack tip node. Furthermore the coordinates of the existing crack tip node are changed by moving it along a vector perpendicular to the crack vector.
2.3.2 Splitting elements

After adding a new node on line $\ell$ the face element has to be split in two. In case of the element of figures 2.5 and 2.6 this looks like figure 2.7.

Figure 2.7: An element with the newly added node and the split crack tip node. The element has to be split in two.

The existing face element has three nodes. The easiest way to split the element is to change the end node of segment $\tilde{s}_2$ into the newly added node. This produces the upper element in figure 2.7. Then a new face element has to be defined in the face matrix with the end node of segment $\tilde{s}_2$, the newly added node and the duplicated crack tip node. This is the lower element in figure 2.7.

After the splitting, some of the elements that contained the crack tip node are above the crack
and others are below the crack. So it is important to find all elements that border on the new element, i.e. the element just defined containing the duplicated crack tip node. In them the old crack tip node number has to be replaced by the number of the duplicated node. This is realized by taking the newly added element containing the duplicated crack tip node and the new crack tip node. The third node of this element is called ‘border node’, this is the node at the end of vector \( \vec{s}_2 \) in figure 2.8. The element adjacent to this element, element 3 in figure 2.8, ‘shares’ the old crack tip node, because this number has not been replaced yet, and the ‘border node’. Then in this element the old crack tip node is replaced by the duplicated one and the third node of this element becomes the ‘border node’. Then the program searches for the next element, which again contains the ‘border node’ and the old crack tip node. Again the old crack tip node is replaced by the new one and the third node is the new ‘border node’. The program continues like this, until it cannot find an element which contains the ‘border node’ and the old crack tip node: then all elements on one side of the crack have been adjusted so they contain the duplicated crack tip node.

Figure 2.8: The elements 1, 2 and 3 contained the crack tip node. Now the crack tip element is split, the crack tip node in these elements has to be replaced by the duplicated crack tip node.

2.3.3 Different situations and their corresponding actions

In the algorithm for each element it is checked what happens to that element, with the main goal to find out whether the crack vector crosses the element or not. When the crack vector does cross the element there are several possibilities, which are schematically shown in figure 2.9. When \( \alpha \) and \( \beta \) are calculated, their values give information about the situations that occurs for that element. When \( \alpha \) lies between 0 and 1, the crack vector crosses the element. When \( \alpha \) is larger than 1 or smaller than 0, the crack does not pass the element. Also when \( \beta \) is negative the crack does not cross the element. When \( \beta \) is smaller than 1, the crack vector goes entirely through the element. When \( \beta \) is larger than 1, the crack vector stops inside the element.
Figure 2.9: An element and the different possible locations of a new node, indicated by the red spots.

When the crack vector is close to one of the boundaries of the element, it is desirable to place the new node exactly on that boundary, to avoid extremely small elements. Therefore a certain tolerance is defined. When the crack vector is close to the line within that tolerance the program decides to place the node on the line. The same is true for the corners of the element: when a new node should be placed close to one of the existing nodes on the corners, no new node is placed, but simply the existing node is used. Figure 2.10 shows the different areas in which the crack vector can end inside the element. Note that for situations 1, 3, 4 and 5 as sketched in figure 2.9 the crack vector ends outside the element and these situations are thus dealt with in another element.

Situation 2: The crack vector is too small.
Situation 6: The crack vector ends inside the element.
Situation 7: The crack vector is close to segment vector 1 and stops on this segment.
Situation 8: The crack vector is close to segment vector 2 and stops on this segment.
Situation 9: The crack vector stops close to line $\ell$.
Situation 10: The crack vector stops near the end of segment vector 1.
Situation 11: The crack vector stops near the end of segment vector 2.

Figure 2.10: The different situations that can occur when the crack vector ends inside the element have been indicated by different colors. The radius of the circles around the corners of the element equals $tol$. Also the width of the bands alongside the borders of the element is $tol$.

The areas have to be described mathematically for the program to distinguish between the different situations. All of the situations can be distinguished by the values of $\alpha$ and $\beta$.
In situation 1 the crack vector does not cross the element. This happens when $\alpha$ is negative or...
larger than 1. It also occurs when $\beta$ is negative. In this case the program skips the element and goes to the next one and checks what happens with that element.

Situation 2, where the crack vector is too small, occurs when $\beta$ is large. The area for situation 2 is defined as all values of $\alpha$ between 0 and 1 and all values of $\beta$ larger than $1/tol$. When this situation occurs the program has to stop and no new node has to be placed.

When $\beta$ is smaller than 1, the crack vector crosses the element. Then three more situations can happen. First, the crack vector is (almost) overlapping segment vector 1. Then no new node has to be added, but simply the node at the end of $s_1$ is used. The crack vector then has to be reduced by $s_1$. A similar situation occurs in the second case, where the crack vector is (almost) overlapping the second segment vector. Also then no new node has to be added. The node at the end of vector $s_2$ is used and the crack vector is reduced by $s_2$. The third case is when the crack vector is not close to any of the segment vectors. Then a new node has to be placed on line $\ell$ and the crack vector is reduced by $\vec{a} = \beta \vec{c}$ (see section 2.3.1). The element is split in two according to section 2.3.2. The program continues by splitting the next element and then repeating all steps.

When $\beta$ is larger than 1, but smaller than $tol$, the crack vector ends near line $\ell$. Again three situations can occur. First, the crack vector stops near the end node of segment vector $s_1$. This is situation 10 in figure 2.10. Then no new node is placed, the end node is taken as the new crack tip node and the program stops. Second, the crack vector can stop near the end node of vector $s_2$, situation 11. Then this node is taken as the new crack tip node and the program stops. When the crack vector is close to neither situation 9 occurs and a new node is placed on line $\ell$, the element and the element on the other side of line $\ell$ are split and the program stops.

When $\beta$ is larger than $tol$ and smaller than $1/tol$ the crack vector ends inside the element. Again the three situations described in the two paragraphs above can occur. In situation 7, where the crack vector is (almost) parallel to vector $s_1$ a new node is placed on this segment vector. This node is taken as the new crack tip node. The element and the neighbouring element which shares vector $s_1$ are split and the program stops. When the crack vector is close to segment vector 2, situation 8, a new node is placed on vector $s_2$. This node is the new crack tip node. The element and the neighbouring element which shares $s_2$ are split and the program stops. In the final situation, situation 6 in figure 2.10, the crack vector stops inside the element, not close to any of the borders. A new node is placed inside the element and the element is split into three. Then the program stops.

\section*{2.4 Insertion of the new crack surface}

By applying the algorithm of section 2.3 the crack can be defined on two sides of the geometry. Now the crack surfaces, the upper and lower part of the new crack segment, have to be defined. The structure of figure 2.11b will be used to generate the crack surfaces. Two surfaces lie above each other. The black dots indicate nodes that already exist and the red nodes have to be added to the node matrix in this stage of the program.

To create the structure of figure 2.11b the right nodes have to be assigned to the new elements in the element matrix. Therefore the order of the nodes on the sides of the crack surfaces is important. Appendix A explains how the nodes are placed in the right order. Besides the nodes on the side also the nodes in the old tip of the crack and the nodes in the new crack tip are needed. Also the nodes in the original crack tip have to be split and moved slightly. Section 2.4.1 explains how to obtain all these nodes and put them in the right order.

Finally, when all components for the two surfaces are present the surface has to be defined with the structure as shown in figure 2.11b. This is discussed in section 2.4.2.
2.4.1 Nodes in the crack tip

The nodes at the sides of figure 2.11a including the corner nodes, have to be determined and placed in the right order. How this is done can be found in appendix A. What remains is that the nodes in the old tip of the crack have to be sorted and the nodes in the new tip of the notch, i.e. the red nodes in figure 2.11b, have to be created.

The nodes in the old tip of the crack, i.e. the nodes at the bottom of figure 2.11a, are a part of elements above the crack as well as below the crack. By comparing the nodes in these two surfaces the nodes in the tip of the crack can be found.

To put these nodes in the right order we start with one of the original crack tip nodes. Then the program searches for an element that contains the crack tip node and one of the nodes in the tip of the crack. That node is the next node in the sequence. Then the program searches for the next element that this node and one of the other nodes in de tip of the crack share. Ultimately all nodes are in the right order, starting with the original crack tip node in the front surface and ending with the crack tip node of the back surface. Because the nodes at the corners are already included in the array of the nodes on the sides these nodes are removed from the array that contains the nodes in the old tip of the crack.

Now the red spots in figure 2.11b, i.e. the nodes in the new tip of the crack, have to be added to the node matrix. The nodes in the old tip have already been put in the right order. These nodes are used to create the new nodes. The new nodes can be generated by copying and moving the nodes in the old crack tip by the crack vector and adding them to the node matrix. Also other crack vectors can be used to create the red nodes, so a more realistic crack surface is created instead of a flat crack surface, see figure 2.12. These new nodes are also stored in a separate array, so that it is easy to retrieve and connect all these nodes in the crack surfaces.
One thing remains: the nodes in the old tip of the crack have to be split. This happens in the same way as nodes were split in section 2.3.1. In figure 2.6 it can be seen that for moving the node and the duplicated node the segment vectors are required. In case of the nodes that are split now no crack element exists like on the sides of the sample. Therefore the segment vectors of the first face element that was cut, i.e. the element containing the original crack tip node, are used to define the direction in which the notch tip nodes should move.

2.4.2 Defining the crack surface

All ingredients necessary for defining the two crack surfaces as shown in figure 2.11b are present: four arrays with the nodes of the edges of the crack, two arrays with the nodes in the old tip of the crack (one with the original nodes and one with the duplicated nodes) and one array with the nodes in the new crack tip. These arrays are only auxiliary, because all the nodes exist in the first place in the general node matrix.

Defining the two surfaces is simple. All elements have to be added to the general face matrix. The right nodes are connected to each other by several for-loops in the algorithm. It is important that the right boundary markers are assigned to the elements. They are no new boundary markers, for the upper and lower sides of the notch already existed: they are only extended by the new crack surfaces. The resulting crack surfaces are shown in figure 2.13.

The structure of the surface as shown in figure 2.11b has advantages and disadvantages. The greatest advantage and the reason why for this study is chosen to use this method, is that no new nodes have to be added to the inner area of the surface. The nodes on the sides are sufficient to create the surface. The most important disadvantage of this method is that the shape of the surface elements may be quite poor. The triangular elements are elongated, especially the elements on the sides of the geometry.

After all of the steps above the geometry can be given to TetGen again, resulting in a mesh like for example figure 2.14.
Figure 2.13: The final crack surfaces created by the program.

(a) The final crack surface of a crack that has propagated three times.
(b) A complex crack surface: for each original crack tip node a different crack vector is used.

Figure 2.14: After the geometry has been manipulated by the program TetGen creates the final result of this study. For this figure the crack vector $\vec{c} = 1.5\vec{e}_x + \vec{e}_z$ is used.
Chapter 3

Applications of the crack propagation program

In the previous chapter it was explained how the Matlab program used to model crack propagation works. In this chapter two applications of the program are shown. First the specimen chapter 2 is used. The program manipulates the specimen several times using artificially generated crack vectors and a crack consisting of several steps is created. Then a sample used for a so called ‘Arcan test’ is manipulated by the program. The crack vectors in this specimen are based on real test results.

3.1 An irregular crack in a simple specimen

The direction of a crack depends on the extent of damage in certain parts of the material. The crack is more likely to propagate through a more damaged part of the material, i.e. the part where more voids appear. Therefore the crack might look irregular. The crack can be modeled in several steps, using a different crack vector in every step. The program is called repeatedly with a new crack vector on the mesh generated in the previous step. The new mesh can be used for stress and strain calculations to determine the direction of the next part of the crack. However, in this study the crack vectors are chosen arbitrarily. Figure 3.1 shows the propagation of an irregular crack in a simple specimen in several steps.

Figure 3.1: Propagation of an irregular crack in a simple specimen in several steps.
3.2 Arcan test

A more realistic application of the crack propagation modeling program is the so called ‘Arcan test’. The test setup can be seen in figure 3.2. A specimen is clamped in two holders with a number of holes. Pins can be placed in these holes to exert a force in a certain direction on the specimen. Depending on the angle of the load the crack will propagate in a certain direction. In his thesis Mediavilla[1] has shown that the experiments of Amstutz et al.[2][3] can be modeled well by his simulations of the Arcan test. The experimental and simulated crack paths are shown in figure 3.3.

![Figure 3.2: The setup of the Arcan test[1].](image)

![Figure 3.3: Crack paths of the Arcan test: experimental (left) by Amstutz et al.[2][3] and simulated by Mediavilla [1].](image)

From the graphs of figure 3.3 the crack vectors that have to be used on the geometry can be determined. These are applied on the three dimensional geometry of the Arcan test sample. Mediavilla has simulated the crack propagation in this specimen in two dimensions and the results can be seen in figure 3.4.
Figure 2.23 — Finite element mesh and crack trajectory for different loading angles.

Numerical difficulties. Transfer errors together with large stress redistributions during the crack propagation can have a detrimental effect on the stability of the computations. It has been demonstrated that by separating and resolving these two instability sources, the robustness of the crack propagation algorithm can be improved considerably, enabling it to deal with coarse meshes and coarse crack discretisations. Transfer errors are addressed by restoring a consistent set of variables, followed by

Figure 3.4: The two dimensional simulation result of the Arcan test by Mediavilla [1].
Figure 3.5: The three dimensional geometries obtained by simulating the crack growth in the Arcan test.
Chapter 4

Conclusion

The goal of this study was to make a first step in modeling crack propagation in three dimensional finite element simulations. This study is limited to planar structures, however the program can be extended to usage for other geometries. A computer program called TetGen divides the geometry into multiple tetrahedral shaped elements. In the computer program Matlab these elements are stored in three matrices: one for the nodes, one for the surface elements, which are triangular, and one for the three dimensional tetrahedra.

For this study the crack direction and length is considered to be known. It generally depends on the forces that are exerted on the sample and on the material. Here we do not pursue this subject. Instead the crack direction and length are described by a given vector, the so called crack vector. This vector starts at a certain node: the tip of the existing crack. After the sample has been manipulated by the program the geometry can be presented to TetGen again to produce a new mesh based on the extended crack.

First the front and back surfaces of the specimen are adapted. The crack vector starts in a node surrounded by several elements. The program searches for the elements that are crossed by the crack vector and splits these elements. This is done by manipulating the matrices which describe the geometry: new nodes and elements are added to the node and face matrices, node coordinates are changed and node numbers of the face elements are adjusted.

Then the crack surface has to be defined in three dimensions. The two crack surfaces, i.e. the upper and lower part of the crack, are extended. Therefore new nodes in the tip of the crack are added, the nodes in the old crack tip are split and new elements forming the new crack surfaces are added to the face matrix.

Then the manipulated matrices can be transformed to a file TetGen can handle to create a new mesh. Stress and strain calculations can be performed on this new mesh to determine the direction of the next part of the crack. The three matrices can be generated again and a new crack vector can be used to again adapt the mesh.

The first step in modeling crack propagation is taken. However, many improvements can be made to the program to make it more appropriate for more complex geometries. Below several recommendations for improving the program are listed.

First the front and back surfaces of the sample need not be flat. They can be curved, three dimensional surfaces. Then all vectors can be projected on a flat, two dimensional surface and the same algorithm described here can be used to create the new crack. Finally the crack on the two dimensional surface has to be projected back on the curved surface.

In section 2.4.2 a great disadvantage of defining the crack surfaces the way described in this report was already mentioned. Because no new nodes are added in the center of the crack surfaces the elements defining these surfaces are rather elongated. TetGen can handle these elements, but will divide them into a great number of smaller, better shaped elements. Therefore more calculations are needed to manipulate the new mesh in Matlab. A great improvement to the program would therefore be to optimize the shape of the elements defining the crack surface.
A shortcoming of the program is that when the angle between the existing crack and the propagated crack reaches 90° or becomes even smaller, the corners of the crack surfaces are not aligned properly, see figure 4.1. Therefore the upper and lower surface of the notch can be intersecting. The situation that the angle is smaller than 90° is unrealistic, however in theory it should be possible. A solution to this problem might be to not split the elements on the sides yet by a certain distance and then inserting two crack surfaces, but to insert one crack surface and then duplicate this surface and shift the two surfaces as a whole. Another problem is that when the elements around the crack are small, the elements are flipped by separating the edges of the propagated crack by some distance. One could think of an algorithm that moves the entire elements above and below the crack on the sides and thereby compresses the elements that are located higher above and lower below the crack.

Another improvement to the program might be in the definition of the different situation that can occur to an element. In section 2.3.3 an element was divided into several areas. The area determines the positions of the new node: when a node should be placed close to an element boundary the program decides to place the node on that boundary. The definition as described in this report is quite simple. This is a great advantage, since the program can calculate which situation occurs quickly. Another advantage of this definition is that the boundaries are relative to the size of the element. Therefore no problems with intersecting boundaries occur in small elements. However, there are also some disadvantages to this definition of the different areas. First of all the corner areas are not defined properly. The areas that belong to the situation where the crack vector ends close to one of the corners of the element are diamond shaped. Therefore there are spots that are actually closer to one of the borders than to the corner, but when the crack vector ends at such a place, the program acts like the crack vector ended near the corner. Furthermore the disadvantage of defining the tolerance relative to the size of the element is that when the element is small, the tolerance will be even smaller. Thus when the crack vector ends quite close to one of the border of the element, but not within the tolerance, a new node will be placed inside the element, resulting in three even smaller elements that probably will be poorly shaped. Other possibilities of defining the different areas inside an element are discussed in appendix B.

It can be concluded that the first step in modeling crack propagation in three dimensions is taken. However, many improvements are possible to make the program suited for more complex geometries and to improve the resulting meshes.
Bibliography


Appendix A

Sorting of the nodes on the edge

In section 2.3.2 it is explained that when an element is split, the crack tip node is split too. The existing crack tip node is moved slightly up or down, perpendicular to the crack vector, depending on the orientation of the segment vectors $\vec{s}_1$ and $\vec{s}_2$. When the next element is split, the same happens. Thus there is no guarantee that all existing crack tip node numbers and duplicated crack tip nodes are on the same side of the crack. Figure A.1 illustrates a possible violation. This situation complicates things, since it is not easy to find out which nodes are positioned above the crack and which below the crack.

![Diagram of nodes along the crack]

Figure A.1: An example of the origin of nodes along the crack. Not all of the existing crack tip nodes are necessarily on the same side.

To order the nodes on the edges of the crack two arrays are defined in the program: one for the nodes that are positioned above the crack and one for the nodes below the crack. In several steps each array is filled with the nodes. These steps are integrated in the program as described in chapter 2.

1. Both arrays start with the old crack tip node.

2. When the program discovers where the crack vector crosses the element boundary, the new node that is added there is also added to both arrays. For example we have:

Array 1: $[10 \ 939]$  
Array 2: $[10 \ 939]$. 


3. The next step in the algorithm is to duplicate the crack tip node. To arrange the nodes conveniently we decide to always keep the original crack tip node in the first array. Therefore after duplicating the crack tip node the first entry of the second array is changed into the duplicated crack tip node. So we get:

Array 1: [10 939]
Array 2: [940 939].

4. Next the program finds the place where the second element is crossed by the crack vector. The new node that is placed at this position is added to both arrays:

Array 1: [10 939 941]
Array 2: [940 939 941].

5. The crack tip node is duplicated again. Now the program needs to distinguish between the upper and lower array for the first time. To do so, the last ‘border node’ which was discussed in section 2.3.2 is used. Recalling that on the side of the crack where the duplicated crack tip node is positioned the old crack tip node in all the elements on that side should be replaced by the duplicated one. Therefore the so called ‘border node’ was used. The last ‘border node’ actually is the node that is positioned right before the node we now want to change in the array, see figure A.2. Therefore the ‘border node’ enables the program to find out in which array the node before the last one should be changed into the duplicated crack tip node. For example the arrays look as follows:

Array 1: [10 942 941]
Array 2: [940 939 941].

Figure A.2: The last ‘border node’ is what makes it possible for the program to discover in which array the node before the last one should be changed in the duplicated crack tip node.

Steps 4 and 5 are repeated until the end of the crack is reached. Then the arrays may look as follows:

Array 1: [10 942 944 945 461 949 948]
Array 2: [940 939 941 943 947 946 948].

This process is repeated on the back surface of the sample resulting in four arrays, two for each surface.
Appendix B

Other possibilities of defining propagation situations

In figure 2.10 it can be seen how the different situations that determine where a new node is located are defined. One of the advantages of this method is that there are no problems with small elements: the bands never overlap at other places than the corners of the element. Another advantage is that the mathematical description of the different areas is quite simple. However, there are also disadvantages. For small elements the bands are small as well. When the crack vector ends close to one of the bands, but not inside it, the program decides to place the new node inside the element, resulting in three even smaller (and probably poorly shaped) elements. Therefore another option of defining the situations is to define the tolerance as an absolute value instead of a relative one. Then the element is divided in the different areas as in figure B.1.

Situation 2: The crack vector is too small.
Situation 6: The crack vector ends inside the element.
Situation 7: The crack vector is close to segment vector 1 and stops on this segment.
Situation 8: The crack vector is close to segment vector 2 and stops on this segment.
Situation 9: The crack vector stops close to line $\ell$.
Situation 10: The crack vector stops near the end of segment vector 1.
Situation 11: The crack vector stops near the end of segment vector 2.

Figure B.1: Another way of defining the different areas in an element.

Unlike figure 2.10 a band to the right of the element is defined in figure B.1. This is because when the crack vector ends in this area, a new node is placed on the line $\ell$ and the crack vector is shortened. Then in the next step the crack vector appears to be too short and no new node is placed. To prevent the program from calculating this extra step, finding out that the crack vector is too short, the band to the right of the element is defined. The advantage of this method is that the areas are defined more systematically. In figure 2.10 the corners were diamond shaped, which means that part of that area is actually closer to one of the sides than to the corner itself. In the situation of figure B.1 points in the corner areas really are closest to the corner. The greatest disadvantage of this method, however, is that it is more complicated to describe the areas mathematically. For setting up mathematical relations the distances from the end of
the crack vector to the line segments have to be determined. Table B.1 gives the mathematical representation of the distances. Every distance is given a name to avoid having to use the relatively long mathematical expressions every time.

<table>
<thead>
<tr>
<th>Distance</th>
<th>Name</th>
<th>Mathematical representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vector representing the perpendicular projection of the crack vector on $s_1$.</td>
<td>$\beta \vec{c} - \vec{c}$</td>
<td>$(\beta \vec{c} - \vec{c})\left(\frac{s_1 - s_2}{|s_1 - s_2|}\right)$</td>
</tr>
<tr>
<td>Vector representing the perpendicular projection of the crack vector on $s_2$.</td>
<td>$\beta \vec{c} - \vec{c}$</td>
<td>$(\beta \vec{c} - \vec{c})\left(\frac{s_1 - s_2}{|s_1 - s_2|}\right)$</td>
</tr>
<tr>
<td>Projection of the difference vector $\beta \vec{c} - \vec{c}$ on the line $\ell$. See figure B.2.</td>
<td>$\beta \vec{c} - \vec{c}$</td>
<td>$\left(\beta \vec{c} - \vec{c}\right)\left(\frac{s_1 - s_2}{|s_1 - s_2|}\right)$</td>
</tr>
<tr>
<td>Distance from end of crack vector to $s_1$.</td>
<td>$\beta \vec{c} - \vec{c}$</td>
<td>$\left(\beta \vec{c} - \vec{c}\right)\left(\frac{s_1 - s_2}{|s_1 - s_2|}\right)$</td>
</tr>
<tr>
<td>Distance from end of crack vector to $s_2$.</td>
<td>$\beta \vec{c} - \vec{c}$</td>
<td>$\left(\beta \vec{c} - \vec{c}\right)\left(\frac{s_1 - s_2}{|s_1 - s_2|}\right)$</td>
</tr>
<tr>
<td>Distance from end of crack vector to line $\ell$. See figure B.2.</td>
<td>$\beta \vec{c} - \vec{c}$</td>
<td>$\left(\beta \vec{c} - \vec{c}\right)\left(\frac{s_1 - s_2}{|s_1 - s_2|}\right)$</td>
</tr>
</tbody>
</table>

Now the distances from the crack vector to the line segments are defined and have been given a name, the areas of the different situations can be described. Table B.2 gives the criteria for every situation. The numbering of the situations is such that for an element first is checked whether the crack vector passes the element or not (using $\alpha$ and $\beta$). When it does pass the element the program checks if the crack goes entirely through the element or stops inside. If the crack vector stops inside the element, the program checks which of the situations of figure B.1 occurs.
Table B.2: Criteria for checking which situation occurs.

<table>
<thead>
<tr>
<th>Nr.</th>
<th>Situation</th>
<th>Criteria</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The crack vector does not pass the element</td>
<td>( \alpha &lt; 0 ) ( \alpha &gt; 1 ) ( \beta &lt; 0 )</td>
<td>The crack vector is in opposite direction.</td>
</tr>
<tr>
<td>2</td>
<td>The crack vector is too small, so nothing will happen.</td>
<td>( \beta \geq 0 ) ( 0 \leq \alpha \leq 1 ) ( | \perp \vec{c} \vec{s}_1 | \leq tol ) ( | \perp \vec{c} \vec{s}_2 | \leq tol )</td>
<td>The crack vector is not in opposite direction. The projection of the crack vector on ( \vec{s}_1 ) lies within the defined tolerance from the crack tip node. The projection of the crack vector on ( \vec{s}_2 ) lies within the defined tolerance from the crack tip node.</td>
</tr>
<tr>
<td>3</td>
<td>The crack vector goes entirely through the element. A new node should be placed on line ( \ell ).</td>
<td>( 0 \leq \alpha \leq 1 ) ( 0 &lt; \beta &lt; 1 ) ( | \vec{c} - \vec{s}_1 | &gt; tol ) ( | \vec{c} - \vec{s}_2 | &gt; tol ) ( | \Delta \vec{c} | &gt; tol )</td>
<td>The crack vector is inside the element. The crack vector goes entirely through the element. The crack vector is not close to ( \vec{s}_1 ). The crack vector is not close to ( \vec{s}_2 ). Exclude the dark green area in figure B.1.</td>
</tr>
<tr>
<td>4</td>
<td>The crack vector crosses the element entirely and is close to ( \vec{s}_1 ). No new node is placed, the existing node at the end of vector ( \vec{s}_1 ) is used.</td>
<td>( \alpha \leq 1 ) ( 0 &lt; \beta &lt; 1 ) ( | \vec{c} - \vec{s}_1 | \leq tol ) ( | \Delta \vec{c} | &gt; tol )</td>
<td>The crack vector is close to ( \vec{s}_1 ).</td>
</tr>
<tr>
<td>5</td>
<td>The crack vector crosses the element entirely and is close to ( \vec{s}_2 ). No new node is placed, the existing node at the end of vector ( \vec{s}_2 ) is used.</td>
<td>( \alpha \geq 0 ) ( 0 &lt; \beta &lt; 1 ) ( | \vec{c} - \vec{s}_2 | \leq tol ) ( | \Delta \vec{c} | &gt; tol )</td>
<td>The crack vector is close to ( \vec{s}_2 ).</td>
</tr>
<tr>
<td>6</td>
<td>The crack vector stops inside the element.</td>
<td>( 0 \leq \alpha \leq 1 ) ( \beta &gt; 1 ) ( | \Delta \vec{c} | &gt; tol ) ( | \Delta \vec{c} \vec{s}_1 | &gt; tol ) ( | \Delta \vec{c} \vec{s}_2 | &gt; tol )</td>
<td>Exclude the yellow area. Exclude the light green area.</td>
</tr>
<tr>
<td>7</td>
<td>The crack vector stops inside the element and is close to ( \vec{s}_1 ). The crack vector is projected on ( \vec{s}_1 ) and the new node is placed on this location.</td>
<td>( \alpha \leq 1 ) ( \beta &gt; 1 ) ( | \Delta \vec{c} \vec{s}_1 | \leq tol ) ( | \Delta \vec{c} \vec{s}_1 | &lt; | \Delta \vec{c} \vec{s}_2 | ) ( | \Delta \vec{c} \vec{s}_2 | &lt; | \Delta \vec{c} | ) ( tol &lt; | \perp \vec{c} \vec{s}_1 | &lt; | \vec{s}_1 | - tol )</td>
<td>Include the yellow area. Exclude light green area at the left corner of figure B.1. Exclude the dark green area at the right upper corner of figure B.1. Taking care that the projection of the crack vector on ( \vec{s}_1 ) doesn’t end inside the circle with radius ( tol ) around the end of vector ( \vec{s}_1 ).</td>
</tr>
</tbody>
</table>
The crack vector stops inside the element and is close to $s_2$. The crack vector is projected on $s_2$ and the new node is placed on this location.

& $\alpha \geq 0$  
& $\beta > 1$  
& $\|\Delta c s_2\| \leq tol$  
& $\|\Delta c s_1\| \leq \|\Delta c|l|$  
& $tol < \|\perp c s_2\| < \|s_2\| - tol$  
& Include the light green area.  
& Exclude yellow area at the left corner of figure B.1.  
& Exclude the dark green area at the right lower corner of figure B.1.  
& Taking care that the projection of the crack vector on $s_2$ doesn’t end inside the circle with radius $tol$ around the end of vector $s_2$.  

| The crack vector stops at the end of the element, near line $\ell$. A new node should be placed on this line. | $0 \leq \alpha \leq 1$  
& $\beta > 0$  
& $\|\Delta c\| \leq tol$  
& $\|\Delta c s_1\| \leq \|\Delta c s_1\|$  
& $\|\Delta c s_2\| \leq \|\Delta c s_2\|$  
& $\|\beta c - s_1\| - \perp (\beta c - c) \ell \geq tol$  
& $\|\beta c - s_2\| - \perp (\beta c - c) \ell \geq tol$  
& Include the dark green area on both sides of line $\ell$.  
& Exclude yellow area at right upper corner of figure B.1.  
& Exclude light green area at right lower corner of figure B.1.  
& Taking care that the projection of the crack vector on line $\ell$ does not end inside the circle with radius $tol$ around the end of vector $s_1$ (excluding the red area).  
& Taking care that the projection of the crack vector on line $\ell$ does not end inside the circle with radius $tol$ around the end of vector $s_2$ (excluding the orange area).  

| The crack vector stops at the end of the element, near line $\ell$ and is close to $s_1$. No new node has to be added. The existing node at the end of vector $s_1$ is used. | $\alpha \leq 1$  
& $\beta > 0$  
& $\|\perp c s_1\| \leq \|s_1\| - tol$  
& $\|\Delta c s_1\| \leq \|\Delta c s_1\|$  
& $\|\beta c - s_1\| - \perp (\beta c - c) \ell \leq tol$  
& $\|\Delta c\| \leq tol$  
& The projection of the crack vector on segment vector $s_1$ ends inside the circle with radius $tol$ around the end of $s_1$ (excluding yellow area).  
& Exclude dark green area.  
& The distance between the end of the crack vector and line $\ell$ is smaller than the tolerance.  

| The crack vector stops at the end of the element, near line $\ell$ and is close to $s_2$. No new node has to be added. The existing node at the end of vector $s_2$ is used. | $\alpha \geq 0$  
& $\beta > 0$  
& $\|\perp c s_2\| \geq \|s_2\| - tol$  
& $\|\Delta c s_2\| \leq \|\Delta c s_2\|$  
& $\|\beta c - s_2\| - \perp (\beta c - c) \ell \leq tol$  
& $\|\Delta c\| \leq tol$  
& The projection of the crack vector on segment vector $s_2$ ends inside the circle with radius $tol$ around the end of $s_2$ (excluding light green area).  
& Exclude dark green area.  
& The distance between the end of the crack vector and line $\ell$ is smaller than the tolerance.  

32
These complex mathematical descriptions are hard to program. A mistake is easily made and it is hard to see if all situations are defined unambiguously. Instead of programming a for-loop containing all these situations another option is to program if-statements. The scheme for the program can be seen in figures B.3 - B.5. The advantage of the if-statements is that all the situations are covered and there is no situation possible where none of the eleven situations occur. In programming for-loops it is possible, in theory, that a situation occurs which none of the for-statements covers. Another advantage is that in for-loops all the values of the complex mathematical descriptions of the areas have to be calculated. In the schemes below can be seen that for situation 1 only \( \alpha \) and \( \beta \) have to be calculated. The difficult calculations only follow for the more complex situations. However, this way of programming still bears the disadvantages of figure B.1. Difficult calculations have to be made and henceforth is chosen to use the more simple method of section 2.3.3.

0 \leq \alpha \leq 1
\beta > 0

Situation 1: not inside element. Check next element.

Situation 2: crack vector is too small. Stop program.

Situation 3: Place a new node on line \( \mathbf{\lambda} \). Crack vector stops inside element (Start in next scheme)

Situation 4: crack vector is close to segment vector 1. Place no new node.

Situation 5: crack vector is close to segment vector 2. Place no new node.

Figure B.3: Scheme 1
Situation 6: Crack vector stops in center of element. Place new node at the end of the crack vector, split element and stop program.

Figure B.4: Scheme 2
Situation 7: Crack vector stops near segment vector 1. Place new node on this vector, split elements bordering on vector and stop program.

Situation 8: Crack vector stops near segment vector 8. Place new node on this vector, split elements bordering on vector and stop program.

Situation 9: Crack vector stops near line $\ell$. Place new node on this line, split elements bordering on line $\ell$ and stop program.

Situation 10: Crack vector stops near end of segment vector 1. Place no new node and stop program.

Situation 11: Crack vector stops near end of segment vector 2. Place no new node and stop program.

Figure B.5: Scheme 3