# Implementation and comparison of models to create a basic profile on a arbitrary road surface

# Implementation and comparison of models to create a basic profile on a arbitrary road surface.

**S.H.M. Kersjes**

**DCT 2005.131**

**Oktober 10, 2005**
**Eindhoven University of Technology (TU/e)**

**S.H.M. Kersjes** **s0550169(TU/e)**

**Supervisor: Dr. Ir. A. Schmeitz (TU/e)**
**Prof. Dr. H. Nijmeijer (TU/e)**

# Samenvatting

Gedurende de afgelopen 5 jaar is er [door Schmeitz aan de TU Delft] onderzoek verricht naar bandmodellen die de responsie van een autoband op oneffen wegdekken kunnen beschrijven. Om het quasi-statische ‘enveloping’-gedrag van een autoband te beschrijven is het zogenaamde ‘tandemmodel met elliptische nokken’ ontwikkeld. Met behulp van dit semi-empirisch contactmodel kan de quasi-statische responsie van een band goed beschreven worden op zowel tweedimensionale als ook driedimensionale wegdekoneffenheden. Nu het tandem model volledig ontwikkeld is, dient het zo efficiënt mogelijk in software geïmplementeerd te worden. Om de meest efficiënte methode te bepalen is echter nog verder onderzoek nodig.

Het doel van dit onderzoek is om verschillende methodes om het tandemmodel op een willekeurig wegdek te gebruiken, te implementeren en te onderzoeken, zowel tweedimensionaal als driedimensionaal. Hierna deze methodes te vergelijken en de meest efficiënte methode te bepalen.

Drie modellen zijn geïmplementeerd: het basic function model, het discrete model en het pre-evaluation model. Simulaties laten zien dat alle modellen bruikbaar zijn maar dat ieder model ook zijn beperkingen heeft hoe het gebruikt kan worden. Er is ook aangetoond dat interpolatie een negatief effect heeft op de resultaten, hoewel het discrete model hier niet zonder kan.

## Abstract

In the past 5 years research has been performed [Schmeitz at the TU Delft] on tire models that describe the response of a car tire on uneven road surfaces. In order to describe the quasi-static enveloping behaviour of a car tire, he developed the so-called tandem model with elliptical cams. With this semi-empirical contact model, the quasi-static behaviour can be described rather well. Now the tandem model is fully developed and validated, it has to be implemented in software as efficiently as possible. To do this, further research is necessary.

The aim of this research is to implement and investigate different methods to use the tandem model with elliptical cams on arbitrary road surfaces in the two-dimensional case as well as in the three-dimensional case. Comparing these different methods, and selecting the most efficient method.

Three models are implemented and investigated: the basic function model, the discrete model and the pre-evaluation model. Simulations show that the models are all valid but that each model has its limitations in which it can be used. It is also shown that interpolation has a negative influence on the results, although the discrete model cannot work without it.

# List of symbols

| Symbol | Definition | Unit |
|---|---|---|
| A | Storage matrix | [-] |
| $a_e$ | Ellipse width | [m] |
| B | Input matrix | [-] |
| $b_e$ | Ellipse height | [m] |
| $c_e$ | Ellipse power | [m] |
| h | Step height | [m] |
| $L_b$ | Length basic curve | [m] |
| R | Tire radius | [m] |
| X | Step coordinate | [m] |
| $X_b$ | Begin coordinate of the basic curve | [m] |
| $X_e$ | End coordinate of the basic curve | [m] |
| $X_{ellipse}$ | ½ of the valid search are of the ellipse | [m] |
| $X_{step}$ | Sample distance (X axis) | [m] |
| $\psi$ | Angle | [rad] |

| Indices | Definition |
|---|---|
| b | Begin |
| e | End |
| i | Step-number |
| * | used for the evaluation of intersecting or overlap |

# 1 Introduction

## 1.1 History of the tire model

In the past 5 years Schmeitz has performed research on tire models that describe the response of a car tire on uneven road surfaces [1]. In order to describe the quasi-static enveloping behaviour of a car tire, he developed the so-called tandem model with elliptical cams. With this semi-empirical contact model, the quasi-static behaviour can be described rather well. Now the tandem model is fully developed and validated, it has to be implemented in software as efficiently as possible. To do this further research is necessary.

## 1.2 Aim / subject

Subject:

- Implementing the elliptical cam model, in Matlab / Simulink and investigating different methods to use the elliptical cam model on arbitrary road surfaces 2D as well as 3D.
- Comparing different methods and computing time
- Selecting the most efficient method

Aim:

- Finding a good model which can efficiently describe the basic profile, needed for the elliptical cam model.

## 1.3 Approach of this research

The tandem model works with a basic profile as input to create an effective road profile. To do this it uses formulas. These formulas, and how they should be used, are already developed by Schmeitz. Now research has to be done on how this basic profile can be generated efficiently when driving on an arbitrary road surface. As indicated before the model uses an elliptical cam to create the basic profile. So far two methods to use this cam on a road surface are available, the analytical method [2] and the discrete method [1]. The analytical method uses analytical formulas, which give the trajectory of the cam sliding over a square obstacle, in combination with steps to create parts of the basic profile, so-called basic curves. Therefore the road profile has to be divided in small steps as displayed in *Figure 1-1*a. After this the basic curve to a single step can be created with the analytical formulas. These basic curves are parts of the total profile used by the tandem model. In *Figure 1-1*b one can see how the basic profile is built up of the individual basic curves.

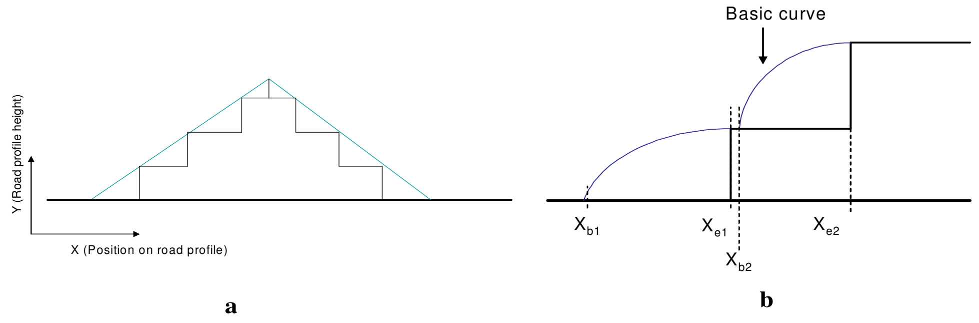


***Figure 1-1*** *Analytical method, a) the road profile is divided in steps, b) the basic profile is created out of basic curves belonging to each step.*

The discrete method was also developed by Schmeitz [1] and differs from the analytical method in that it discretises the elliptical cam, further called ellipse, and the road. To obtain the height of the ellipse, the discretised area is evaluated. This is done by summing the height of the ellipse and the height of the road, at each of these discretised points. At the point where this sum is largest, the ellipse touches the ground. With that information the height of the ellipse can be determined, and so the basic profile can be generated. In *Figure 1-2*, the principle of the discrete method is displayed. After a road point is evaluated the ellipse moves a place further on the road profile, and a new evaluation is started. Hence it is a relatively accurate way, however not fast, and thus not efficient enough.

In addition to this discrete method, another approach, proposed by Schmeitz, will be used. This approach does not discretise the ellipse or the road profile. Instead it looks which points of the road profile fall in the search area of the ellipse, after which it only evaluates the heights at these points. In the following chapters it will be explained how these methods are implemented in algorithms, beginning with the analytical approach. Because of the robustness and accuracy of the discrete model of Schmeitz, this model will be used to validate the other two approaches used in this report.

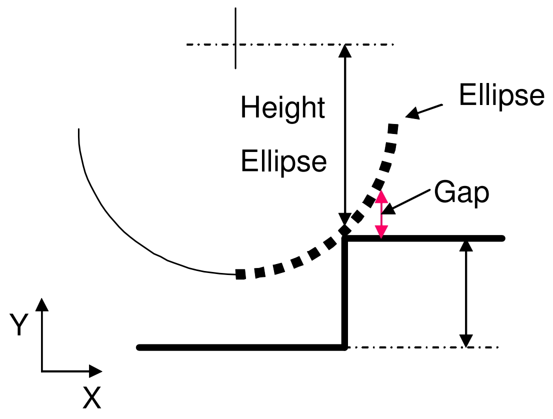


***Figure 1-2*** *Discrete method: at each dotted point the height of the road and ellipse are summed*

## 1.4 Organisation of this report

Chapter 2 will explain how the analytical model works, and how this is implemented in Matlab. The model will be validated, and the approach for working with a measured road profile is discussed. The three-dimensional approach is discussed at the last part of this chapter. Chapter 3 discusses the discrete methods for obtaining the final profile, starting with the discrete model of Schmeitz. These models will also be validated. Hereafter the three-dimensional models will be discussed. Chapter 4 compares the models with each other on accuracy and calculating time. First the two-dimensional models will be compared, after that the three-dimensional models. Finally chapter 5 gives the conclusions from this research and provides some recommendations.

# 2 Analytical method for obtaining the basic profile

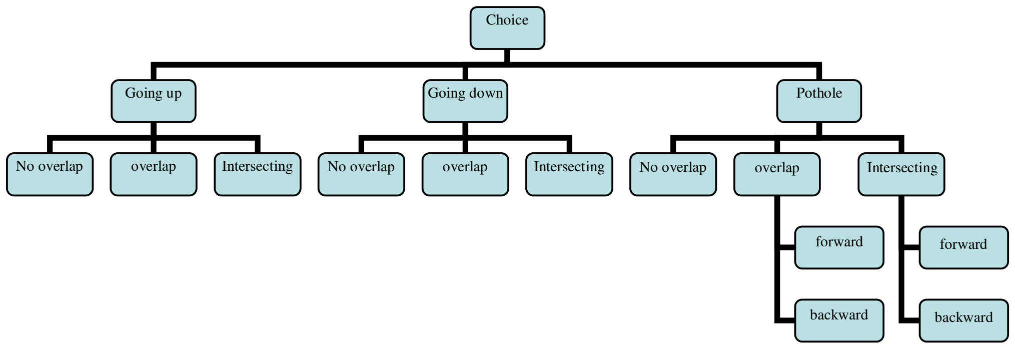
As mentioned in chapter one, the basic profile can be created by using basic curves. These basic curves are generated analytically, which promises a big advantage in calculating time. The disadvantage of this approach is that the individual curves, corresponding to each step, influence each other, see *Figure 1-1*. This is caused by the fact that the distance between the several steps and the length of the created basic curves are not the same. So, an algorithm has to be created that registers where a basic curve of a single step begins and where it ends. Finally all these basic curves are summed together, thereby creating a total basic profile.

In this chapter, the algorithm which evaluates each step and its influence will be explained, first on a two-dimensional scale and later in paragraph 2.6 on a three-dimensional scale.

## 2.1 The basic function model

The algorithm of this model is based on selections. These selections evaluate each step on the influence it has on its neighbours. For this a storage matrix is created, which includes the positions of the steps, their heights, begin and end positions, and the lengths of the basic curves. In order to make these selections in Matlab, several m-files were created.

In the following paragraphs these selections and the different influences will be explained. To give a quick overview on how the selections correlate to each other, the main sequence has been given in the flowchart below. Appendix A includes all m-files created for going upwards as will be explained in the following paragraphs.



***Flowchart 2-1*** *Main overview of the basic function model. Showing the correlation between the selections.*

## 2.2 Storage of data

Before going to the selections, the storage matrix will be discussed; herein the data to evaluate each step will be stored and adapted when going through the selections. The used parameters are:

| | |
|---|---|
| • Coordinate where the actual step takes place: | $X$ |
| • Coordinate where the basic curve begins: | $X_b$ |
| • Coordinate where the basic curve ends: | $X_e$ |
| • Step height: | $h$ |
| • length basic curve: | $L_b$ |

When a road profile is created or given, only the height of each measuring point is known, which means that the storage matrix has to be completed. To do this, several guidelines are made so that always the same approach is used. These guidelines are listed below. The missing parameters can be calculated with formula (2.1) and some basic relationships as displayed in *Figure 2-1*. [1].

1) There will be worked from front to back
2) Going up the X coordinate of the measurement is the same as $X_e$
3) Going down the X coordinate of the measurement is the same as $X_b$

$$L_b = a_e\left(1-\left(1-\frac{|h|}{b_e}\right)^{c_e}\right)^{\frac{1}{c_e}} \tag{2.1}$$

In the main m-file this is automated, so one has only to give the X coordinate, the definition that $X=X_e$ or $X=X_b$ and the step height. The missing parameters are automatically calculated and stored. The storage matrix stores the parameters as is displayed in Table 2-1. Every row in this table represents a step in the road profile. The second row for example contains an upward step of 5 mm. In Table 2-1 one can also see that a fixed interval between the steps is not required which, makes the total matrix compact. How one can interpret these parameters is displayed in Figure 2-1, where the parameter i is the row number (step number) in the storage matrix.

| X | h | $X_b$ | $X_e$ | $L_b$ |
|---|---|---|---|---|
| -0.2500 | 0 | -0.2500 | -0.2500 | 0 |
| -0.0650 | 0.0050 | -0.1105 | -0.0650 | 0.0455 |
| -0.0150 | 0.0100 | -0.0813 | -0.0150 | 0.0663 |
| 0 | 0 | 0 | 0 | 0 |
| 0.0150 | -0.0100 | 0.0150 | 0.0813 | 0.0663 |
| 0.0650 | -0.0050 | 0.0650 | 0.1105 | 0.0455 |
| 0.2500 | 0 | 0.2500 | 0.2500 | 0 |

***Table 2-1*** *Overview of the manner of storage in the storage matrix.*

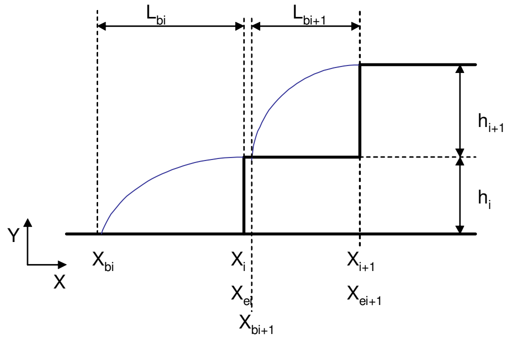


***Figure 2-1*** *Definition of the used parameters.*

## 2.3 Selections of the basic function model

As discussed before, the basic curves influence each other, and selections will be used to define the coordinates of the basic curve. In this paragraph the selections and the influences on which they are based, will be discussed. This will be done with some basic selections. The principals of all the other selections are the same as the selections discussed here.

### 2.3.1 Road profile, going up or going down

This selection evaluates if the road profile is going upwards or downwards. This is done by evaluating the step heights of the actual step and the following step. When both are positive the road profile is going upwards (going up), and when both are negative, the profile is going downwards (going down). In general 5 situations can occur as listed below.

- The road profile is going up
- The road profile is going down
- Nothing changes to the height of the road profile
- A top occurs in the road profile
- A pothole occurs in the road profile

After the actual step is evaluated, the algorithm continues with the chosen m-file containing the next selection. These selections will then evaluate the step further. In the case that there is no change in the height of the road profile, or when a top occurs (in the road profile), the algorithm continues with the next step, because the basic curve will then not influence another basic curve. As an example of this selection, *Figure 2-2* shows the case that both the actual step and the following step have positive heights, therefore the road profile is going up. In this figure the basic curves do not influence each other. Appendix A.2 shows how this is implemented in Matlab.

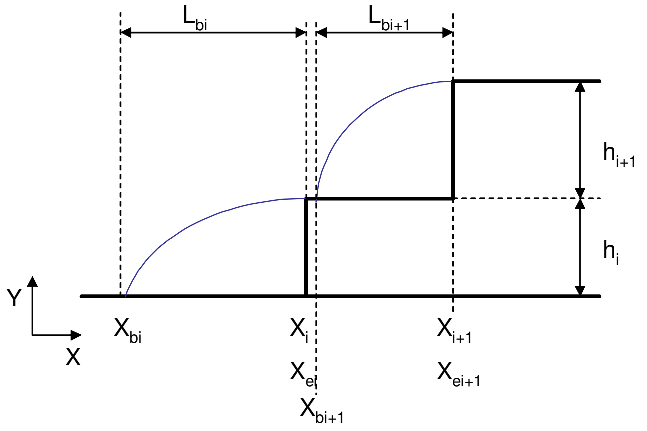


***Figure 2-2** Overview of what happens if the profile is going upwards.*

## 2.3.2 Basic curves, overlap or intersecting

When a "choice" has been made between the five situations discussed before, the steps can be examined in more detail. To create a better understanding of what happens, this report will only consider "going up". Going down or being in a pothole works basically in the same way. If there are big differences it will be discussed at the end of the paragraphs.

When going up, two things can happen: 1) the next step has no influence on the actual step or 2) does have influence on the actual step as displayed in *Figure 2-3*b and c. If the next step influences the actual step two different kinds of influence can occur. The first one is that the basic curve of the next step overlaps the basic curve of the actual step (*Figure 2-3*b). The second one is that the basic curves intersect (*Figure 2-3*c). To see whether there is influence, the begin coordinates $X_{b2}$ and $X_b$* are considered. $X_{b2}$ is the begin coordinate of the basic curve from the next step. When this coordinate is situated between the begin coordinate and end coordinate of the actual basic curve, the two basic curves intersect. This is displayed in *Figure 2-3*c. The coordinate $X_b$* is used to determine if the actual step is being overlapped by the next step. When $X_b$* is smaller than the begin coordinate of the actual step, the actual step (step1) is overlapped by the next step (step 2). This is also displayed in *Figure 2-3*b, where the dotted basic curve overlaps the solid basic curve. The value of $X_b$* is calculated by subtracting the basic curve length ($L_b$*) from the end coordinate of step 2. The basic curve length $L_b$* is calculated with formula (2.1) and uses h* for the step height. The height h* consists of the sum of the two step heights.

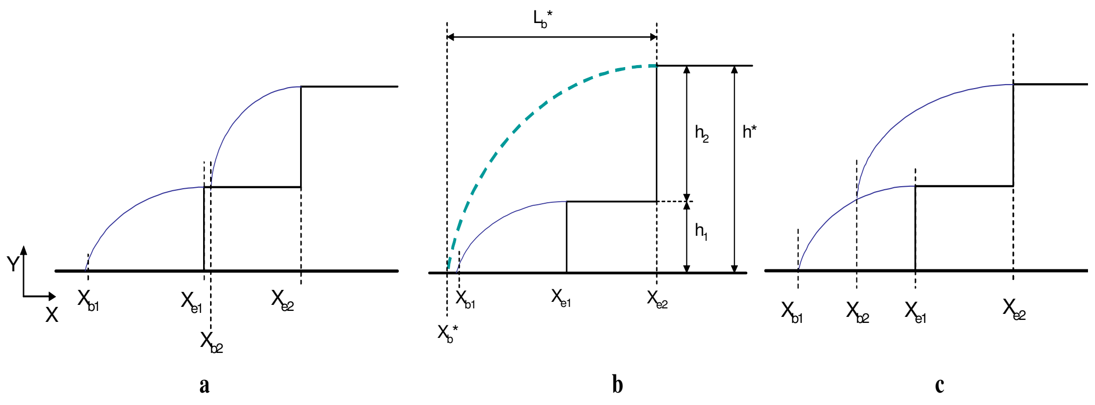


***Figure 2-3*** *Different influences of the basic curves, a) no influence, b) overlap, c) intersecting.*

**Differences in other scripts:**
When a pothole occurs, it is first selected whether it is a backward, forward or equally formed pothole, as displayed in. This is done because these forms change the coordinates of the basic curves, which the algorithm has to compare. For example: When a pothole is formed forward the second basic curve could be overlapped by the first basic curve, while as the pothole is formed backward the first basic curve could de overlapped by the second (see *Figure 2-4*).

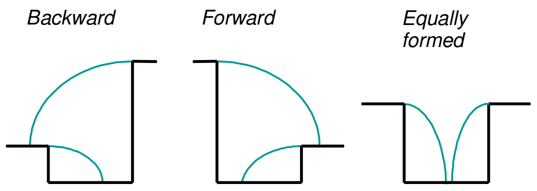


***Figure 2-4*** *Different forms of potholes.*

## 2.3.3 No overlap of the basic curves

No overlap means that there is no influence of the next step on to the actual step. Consequently nothing will happen to the data in the storage matrix, and the algorithm continues with evaluating the next step. This is illustrated in *Figure 2-5*.

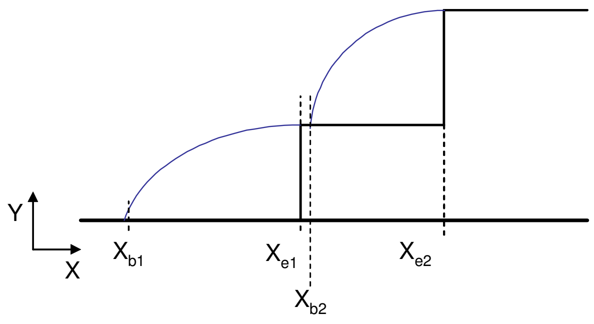


***Figure 2-5*** *No overlap, the basic curves do not influence each other.*

### 2.3.4 Overlap by basic curves

If there is overlap the actual step is totally overlapped by the next step, indicated with h* and $X_b$*. Because the actual step is overlapped, and therefore has no contribution, this step must be removed from the storage matrix. The m-file created for this does that in the following way. First the parameters in the storage matrix are updated. Secondly the overlapped step is removed. This leads to a smaller storage matrix and is as a result more efficient.
See for example *Figure 2-6*, here $X_{b2}$ will be replaced by $X_b$* and $h_2$ will be replaced by h*. After that, step 1 will be removed because it is overlapped by step 2. For the m-file see appendix A.4.

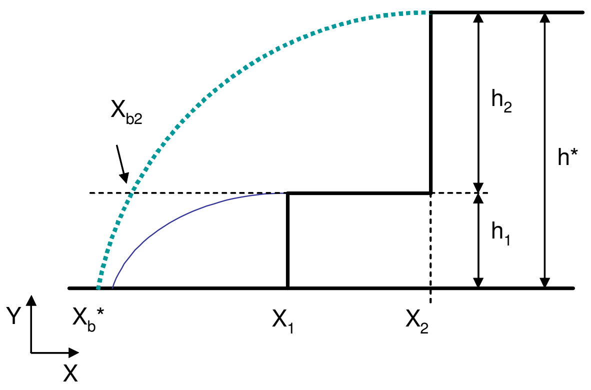


***Figure 2-6*** *Overlap, the dotted basic curve overlaps the continuous basic curve.*

### 2.3.5 Intersecting of the basic curves

When two basic curves intersect it means that the previous step is partly overlapped by the actual step, as displayed in *Figure 2-7*. Therefore, the position of the intersection has to be determined to be sure that the begin coordinate and end coordinate of the actual and next step are correct. To do this formula (2.2) through (2.5) from [1] are combined to formula (2.6). This formula is only dependent on $h_2$*, because the other parameters in this formula are constant.

$$h_2 + h_1 - h_2^* = h_1^* \tag{2.2}$$

$$Z = h - b_e + \left| b_e \left( 1 - \left( \frac{\left| x^* - x \right|}{a_e} \right)^{c_e} \right)^{\frac{1}{c_e}} \right| \tag{2.3}$$

$$x^* + L_b^* = x_e \tag{2.4}$$

$$L_b^* = a_e \left( 1 - \left( 1 - \frac{\left| h^* \right|}{b_e} \right)^{c_e} \right)^{\frac{1}{c_e}} \tag{2.5}$$

$$\mathrm{h}_2 = -\mathrm{b}_\mathrm{e} + \left| \mathrm{b}_\mathrm{e} \left( 1 - \left( \frac{Q}{\mathrm{a}_\mathrm{e}} \right)^{\mathrm{c}_\mathrm{e}} \right)^{\frac{1}{\mathrm{c}_\mathrm{e}}} \right| + \mathrm{h}_2^* \tag{2.6}$$

where

$$Q = \left| \left( \left( x_{e2} - a_e \left( 1 - \left( 1 - \frac{|h_2^*|}{b_e} \right)^{c_e} \right)^{\frac{1}{c_e}} \right) - x_{e1} \right) \right| \tag{2.7}$$

One is interested in $h_2$* so the formula has to be rewritten. Unfortunately this cannot be done. Therefore another approach has to be used. This new approach involves minimizing the distance between the two basic curves. At the point where this distance is zero the two basic curves intersect. For this formula (2.8) is created by combining formula (2.2) with (2.3) (note that $h_1$* = Z, height of the basic curve, for x*). This formula is only dependent of $h_2$*, hence by minimizing formula (2.8) one can find $h_2$*. With the value of $h_2$* one can find also the other parameters ($X_{b2}$, $X_{e1}$ etc.) by using the relationships between them. Hence the starting and exiting points can be updated, so that these are correct. To minimize formula (2.8) the Matlab command Fminbound is used. This will search for the minimum of the formula, using the technique of sectioning and bracketing in a given search area [3]. A termination tolerance of 1.0e-10 is used in this minimization. A higher tolerance did not improve the answer much, moreover it made the calculating time longer. See appendix A.5 A for the full script.

$$r = \left| -b_e + \left| b_e \left( 1 - \left( \frac{Q}{a_e} \right)^{c_e} \right)^{\frac{1}{c_e}} \right| + h_2^* - h_2 \right| \tag{2.8}$$

with Q the same definition as formula (2.7).

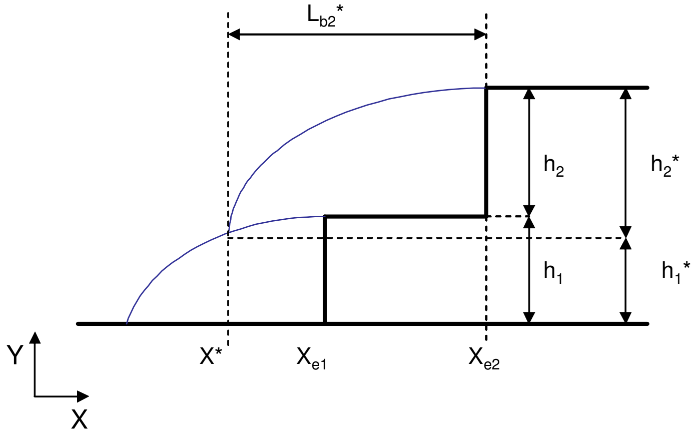


***Figure 2-7*** *Intersecting, the basic curve calculated with h2* intersects the other basic curve.*

**Differences in other scripts:**
In the situation of a pothole a different approach is used because this was more convenient, and for the simulations it does not matter. Here formula (2.3) is used twice which results in formula (2.9).

$$r=\left|\left\{|h_1|-b_e+\left|b_e\left(1-\left(\frac{|x-x_b|}{a_e}\right)^{c_e}\right)^{\frac{1}{c_e}}\right|\right\}-\left\{|h_2|-b_e+\left|be\left(1-\left(\frac{|x-x_e|}{a_e}\right)^{c_e}\right)^{\frac{1}{c_e}}\right|\right\}\right| \tag{2.9}$$

## 2.4 Creating the final (basic) profile by summing basic curves

Now all the steps have been evaluated, the final basic curve can be created. For this a script used by Schmeitz to sum the basic curves is modified. *Figure 2-8* illustrates what is done. At the begin coordinate (first contact with step) the curve is generated, and after the end coordinate (the ellipse is completely on to the step) the height is kept constant. The final profile is created by summing al these curves (the solid with the dotted line) [2]. To do this, a script is written (which is implemented in appendix A.1, last part). First a domain is created in which the final profile is made. Then every step is evaluated in this domain. A function (eltandem_2, see appendix A.6) then calculates the heights of the basic curves. This function uses the formulas created by [1] to do this. In this function (the script) another function, (bound) is called. This function bound evaluates from where the basic curve has to be created seen from the position on the global X axis. For this evaluation it uses the domain, [begin, end] and step coordinate of the curve.

After the heights are calculated, a parameter is used to compensate the offset of the basic curves, in the z direction (*Figure 2-9*). This offset is formed because the begin coordinate of the basic curve ($X_b$) is not similar to the step coordinate ($X_{step}$), in the case of intersecting basic curves. See section 2.3.5. Consequently the calculated height at $X_b$ is not the desired value. This is best explained with *Figure 2-9*, which is a downwards step. One will sum the heights of the usable part to the heights of the previous basic curves. At point $X_b$ one wants to sum a height of zero because at that point the usable part starts. One can see in *Figure 2-9* that the height at point $X_b$ is not zero but has already got a negative value because the total basic curve starts before the usable part (the height of the basic curve at $X_{step}$ is zero). When calculating the basic curve heights this is compensated by summing this offset to the calculated height of the usable part of the basic curve.

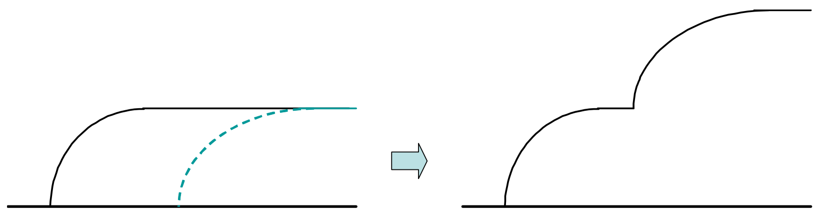


***Figure 2-8*** *Summing two basic curves to get a final profile.*

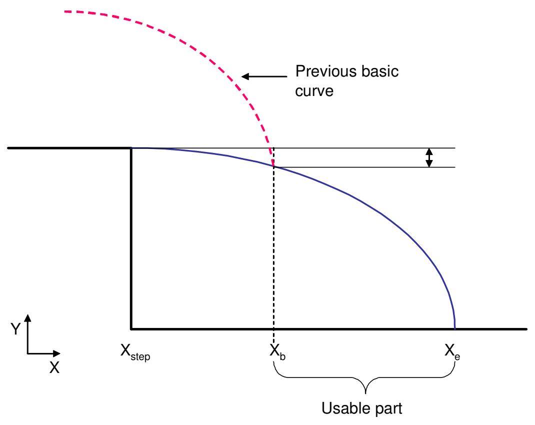


***Figure 2-9*** *Offset of the usable part due to difference between $X_b$ and $X_{step}$.*

## 2.5 Validation

To validate if the answers obtained from this model are correct, they have been compared with answers obtained from the discrete model created by Schmeitz. See paragraph 1.3. In chapter 4, the accuracy and calculating time will be compared. So here we only investigate if the model results agree with what is expected.

The algorithm of the discrete model already contained several different cleats, the so called standard cleats. These cleats are predetermined shapes where the tire envelops about. Nevertheless, in order to validate whether the model is robust, some additional cleats have been made, called created cleats. In addition a road profile has been used, for which it is assumed that every possible situation occurs. For example overlap or intersecting of the basic curves.

The parameters used for the evaluations are displayed in *Table 2-2* and *Table 2-23*. *Table 2-2* displays the standard parameters of the tire and ellipse, for an explanation of these parameters is referred to [1]. X_step is the step size on the x axis when creating the final basic profile, and X_ellipse is half the search area on the ellipse. Because the elliptic cam model is valid for cleats up to 30 [mm] it is not necessary to evaluate the total width of the ellipse, but only from –X_ellipse to X_ellipse. X_ellipse can be calculated with formula (1) using the maximum step height. The number of discretization points is taken rather high, but this will only have a positive effect on the accuracy.

| **Tire parameters:** | | |
|---|---|---|
| R | 312e-3 | [m] |
| Paraac | 1.0325 | [-] |
| parabc | 1.0306 | [-] |
| parac | 1 | [-] |
| **Ellipse parameters:** | | |
| $a_e$ | paraac*R | [m] |
| $b_e$ | Parabc*R | [m] |
| $c_e$ | Parac | [-] |

***Table 2-2*** *Standard parameters, used in every evaluation.*

| **Basic function model parameters:** | | |
|---|---|---|
| X_step | 0.005 | [m] |
| **Discrete model parameters:** | | |
| Disc.points ellipse | 599 | [-] |
| X_ellipse | 0.120 | [m] |
| X_step | 0.005 | [m] |

***Table 2-3*** *Model parameters, used in this evaluation.*

### 2.5.1 Validation, Standard cleats

As said at the start of this paragraph, these standard cleats are already available. They are generated with a separate script (cleats_2 as shown in appendix B). The cleats used to validate are listed below. *Figure 2-10* displays the shape of these cleats. For the sake of simplicity not al the graphs of the validation cleats are displayed here. And because the graphs in *Figure 2-11* represent the main results obtained from the model, only these results are discussed.

From *Figure 2-11* it can be seen that the basic profiles, calculated with both models agree well. This is of course no surprise for cleats 2, 3 and 4 (step shaped cleats) because these consist of only one basic curve for going up, and one for going down. For that reason they cannot influence each other. For cleat 1 (trapezoid), the steepness of the trapezoid leads to overlap of the basic curves, hence the trapezoid is replaced by a single step. Although this occurs, one can see in *Figure 2-11*a, that the basic curves agree well.

For cleat 6, the basic curves at the downward side intersect each other, so the algorithm has to find the point of intersection. In *Figure 2-11*d one can see the results for this obstacle. Where the basic profile of the discrete model consists out of a smoothly curved line, the basic profile of the basic curve model consists out of two basic curves (see the dip in the line). Changing tolerances will not influence this in a positive way. A solution is to divide the road profile into more steps, however this will of course influence the calculating time. Despite this inaccuracy, which can be reduced, one can say that overall the models agree well. Thus the basic curve model is valid. In paragraph 4.1.2 the details of the model and its inaccuracies will be discussed further.

- Cleat 1: trapezoid
- Cleat 2: step 15 [mm] up and down
- Cleat 3: step 20 [mm] only upwards
- Cleat 4: step 30 [mm] only upwards
- Cleat 5: pothole
- Cleat 6: triangle

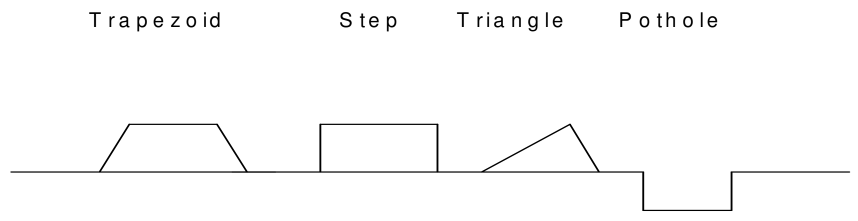
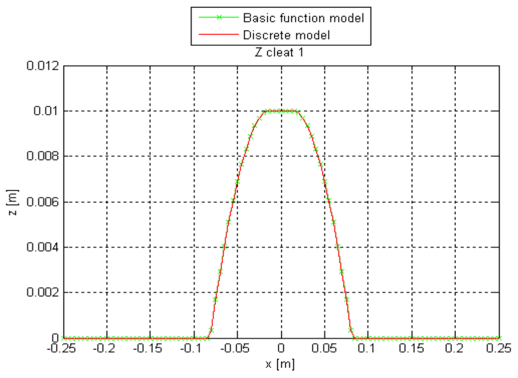
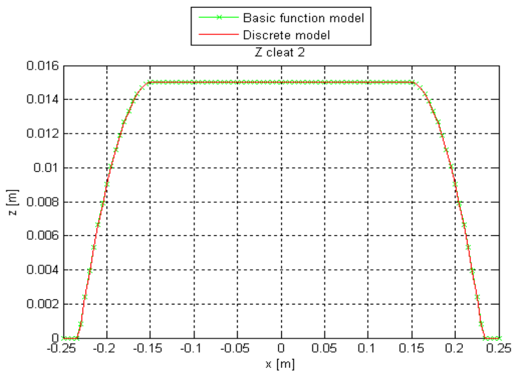


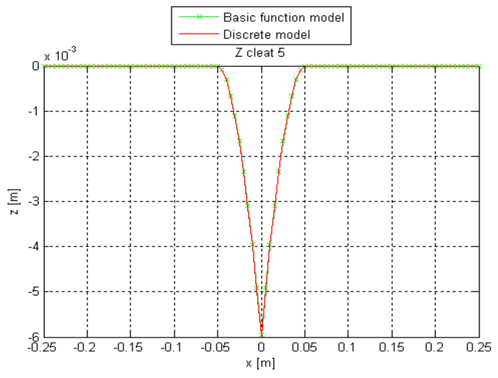
***Figure 2-10*** *Cleat shapes.*



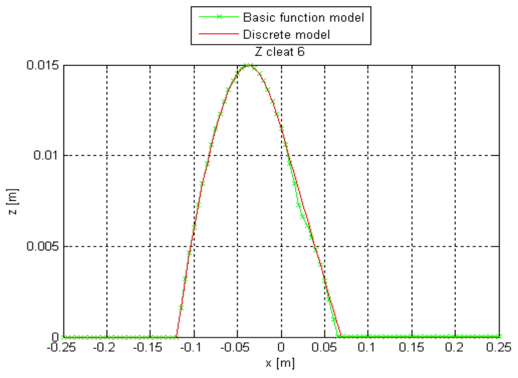
a



b



c



d

***Figure 2-11*** *Validation results, comparison of basic profiles for standard cleats, see figure 12 for the shapes of the cleats.*

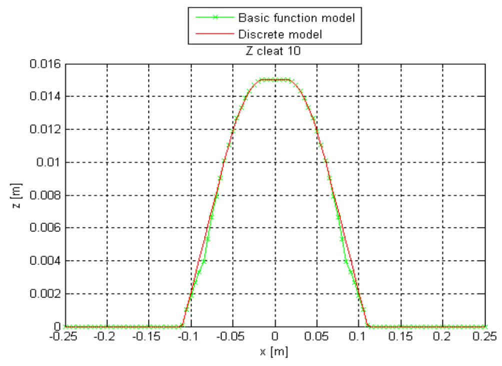
### 2.5.2 Validation, created cleats

The created cleats are shaped in the same way as the trapezoid cleats used before (*Figure 2-10*). The difference is that some basic functions of the algorithm have to be used, when evaluating these cleats. This is accomplished by changing the height and steepness of the trapezoid. For example, a cleat is created where two basic curves intersect and that the intersection point has to be found (cleat 11). Al these cleats use two steps a side, as input of the basic function model, to get the desired height and slope. This will lead to the same inaccuracy as with cleat 6, discussed in paragraph 2.5.1. However, it is instructive to see what actually happens.

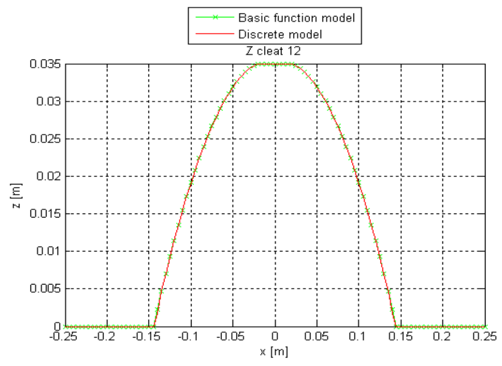
*Figure 2-13* again shows cleat 10, but now four steps have been used. Here it can be seen that the curves agree well when the number of steps increases. The same holds for cleat 13. For cleat 12 and 14, the same thing happens as for cleat 1, see paragraph 2.5.1 for an explanation.

Concluding, one can say that the model is also valid for the created cleats.

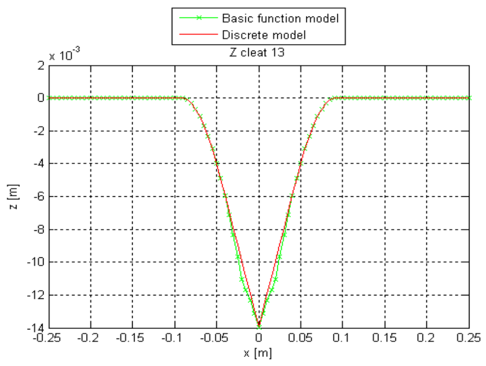
- Cleat 10: trapezoid (intersecting)
- Cleat 11: trapezoid (overlapping)
- Cleat 12: trapezoid (overlapping)
- Cleat 13: pothole trapezoid shaped (intersecting)
- Cleat 14: pothole trapezoid shaped (overlapping)



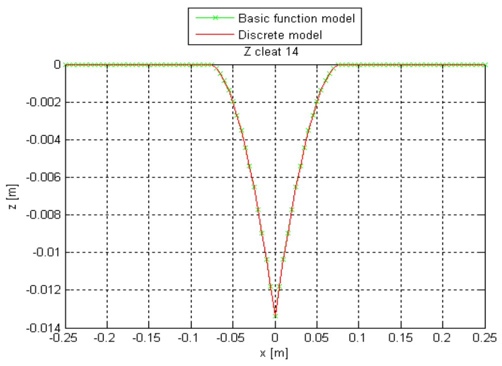
**a**



**b**



**c**



**d**

***Figure 2-12*** *Validation results, comparison of basic profiles for created cleats.*

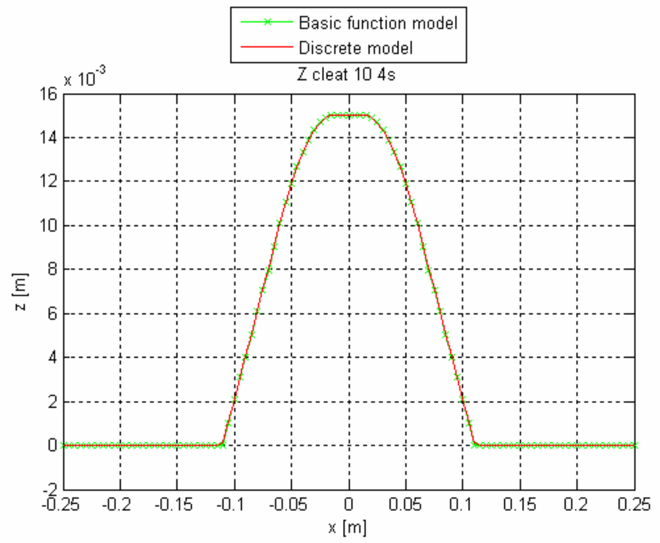


***Figure 2-13*** *Validation result****,*** *Comparison of basic profiles for cleat 10 using 4 steps to get the desired height and slope.*

## 2.5.3 Validation, road profile

For working with a road profile, a script has to be designed which automatically creates a matrix in the form of the storage matrix, from the road measurements. Otherwise stated, the road profile has to be divided into steps. How this is done is displayed in *Figure 2-14*.

As said in paragraph 2.2, not all the parameters have to be known to complete the storage matrix. Also some guidelines where given in this paragraph. These can now become convenient, because the data we have from the road profile only consists of a matrix with X coordinates and the measured heights. To create the storage matrix out of the measurements, a script is created (see appendix B.1). Herein the road measurements are put in an input matrix (B) and a storage matrix (A) is generated. To do this, a selection is made which determines if the road profile is going up, going down etc., by looking at the height of the previous and next measurement point (later referred to as: mpoint). Five situations can occur which are discussed below.

1) The profile is going up, meaning that height of the next mpoint is higher than the actual height and the height of the previous mpoint is lower than the actual height. The m-file then places the X coordinate (road profile) as the step coordinate and end coordinate of the basic curve ($X_e$) in the storage matrix A. The height at this step is the height of the previous mpoint extracted from the height of the actual mpoint.
2) The profile is going down, the same approach as in 1) is used but now in downward direction.
3) The mpoint does not change heights, relative to its neighbours. In this case exactly the same method, as discussed in 1) is used, resulting in a step with a step height of zero.
4) A pothole occurs in the road profile. Because a pothole consists of three steps namely, going down, a step in the pothole and going up, one only has to look at the step in the pothole. For the two other steps, the situations as discussed before can be used (situation 1 and 2). In the pothole the X coordinate is taken as step coordinate and begin coordinate, and the height is kept zero.
5) A top occurs in the road profile (see *Figure 2-14*). This means that there is a step going up and a step going down at the same X coordinate. Hence two steps have to be generated in the storage matrix.

To be able to generate two steps, the script works with two counters namely: i and k. the counter i is the row number (measurement point i) in the input matrix B, whereas k is the row number in the created storage matrix. When a top occurs in the road profile, first a step upwards is created in the storage matrix. After that, the counter k goes further but the counter i stays the same. Secondly a step going downwards is generated in the storage matrix. For this the same measurement point is used (counter i is the same) but it is placed at a new row in the storage matrix (counter k is set one row further). See for example *Figure 2-15*.

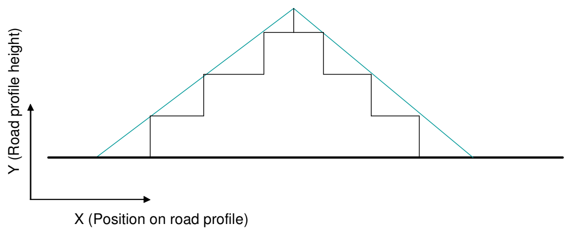


***Figure 2-14*** *Road profile divided into steps.*

```
A =
    0    0    0    0    0
    1    0    0    1    0
    2    1    0    2    0
    3    2    0    3    0
    4    1    0    4    0
    4   -1    4    0    0
    5   -1    5    0    0
    6    0    6    0    0
```

***Figure 2-15*** *Example of a top in the storage matrix A, see also Table 1.*

With the use of this script and the basic function model, as well as with the discrete model, an evaluation is made of the road profile. *Figure 2-16* displays a small part of the generated basic profile for the two models. Although this is only a small part of the total basic profile, it shows the differences between the models quite well. One can see clearly that the basic curve model generates the basic profile out of steps, while the discrete model uses interpolation.

It appears that there are big differences in the calculated heights at some points in the graph. This can be reduced by interpolating the road profile to a finer grid, thereby taking more steps to generate the storage matrix. Hence, one can say that the models in general agree.

However, one can also notice the two peaks (downwards), which is the result of interpolation of the road profile on a wrong scale. *Figure 2-17* shows the result if the road profile is not interpolated at all. One can see that the two peaks do not intersect the profile of the basic function model. The drawback is that the difference between the models is bigger. The effects of interpolation will be further discussed in paragraph 4.1.2.

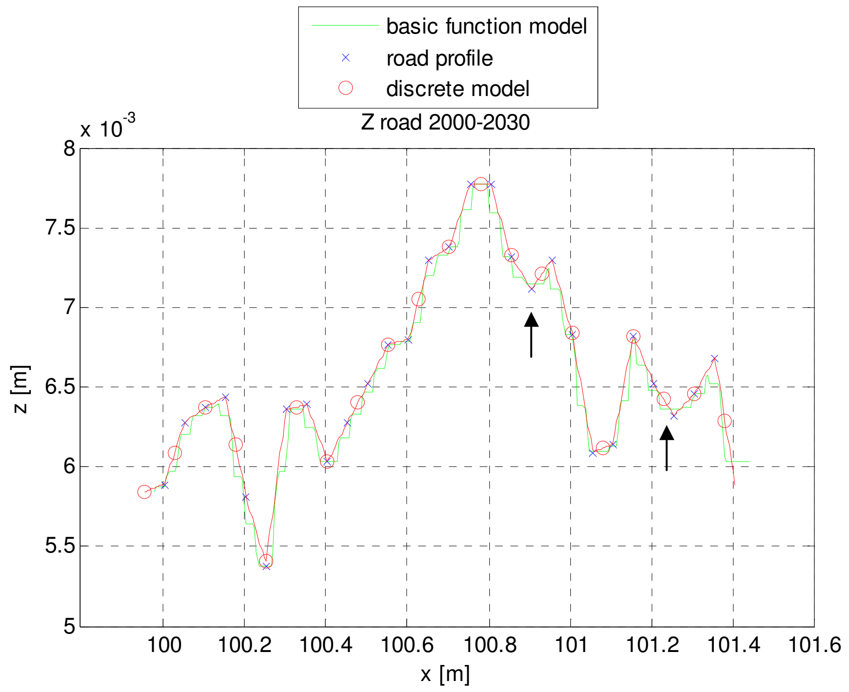


***Figure 2-16*** *Validation result, comparison of basic profiles for a piece of road profile.*

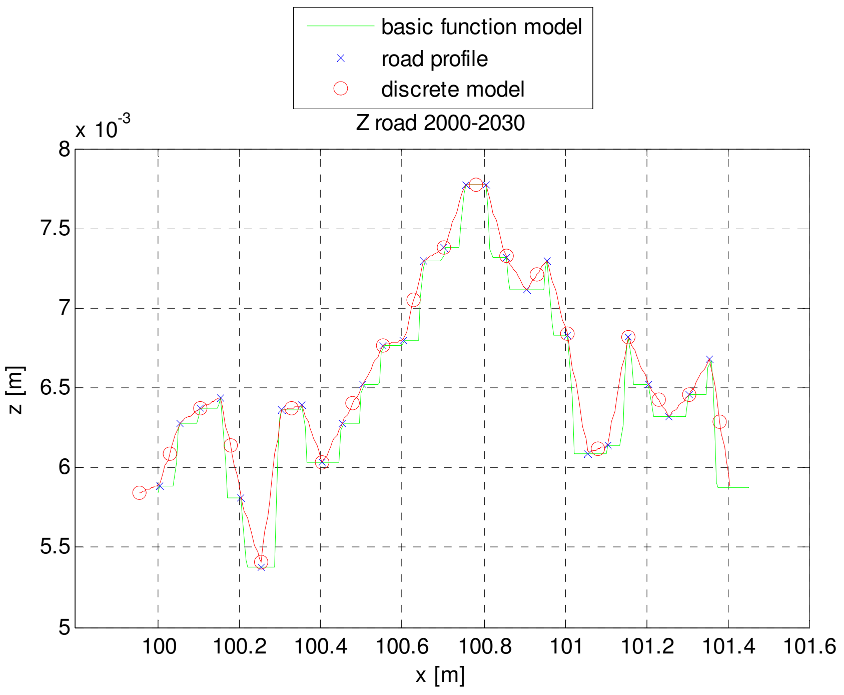


***Figure 2-17*** *validation result it the road profile is not interpolated to a finer grid.*

## 2.6 Analytical method, three-dimensional model

This paragraph will discuss how the three-dimensional elliptical cam model is constructed. The two-dimensional (2-D) elliptical cam model is used as the basis for the three-dimensional elliptic cam model. This can be explained by looking at the way the three-dimensional (3-D) elliptical cam model is created [1]. This is done by putting several 2-D tandem models (tracks) next to each other. Simple formulas are then used to calculate the effective height [1].

Because the aim of this assignment is to create the input for this model, only several inputs have to be delivered for the tracks in this 3-D model. Hereto the storage matrix A is extended with one dimension (third). Each “layer” in this third dimension is used for one 2-D tandem as a track in the 3-D model. For the calculation of the basic curves each “layer” has to be calculated individually after which the output is stored in an output matrix Z. Note: the model does not use a lateral coordinate, only a longitudinal one for the basic curves. As one can see it is not difficult to create a multiple track model out of a model with a single track. Hence in the rest of this report will be spoken of a single track model, driving over a three-dimensional road surface.

There where several approaches evaluated on how the basic function model could work with a three dimensional road surface. Nevertheless, none of these approaches seemed to have an advantage with regard to the three-dimensional discrete model. Therefore these approaches will not be discussed in this report.

# 3 Discrete methods for obtaining the basic profile

As said in chapter 1, a second method to obtain the basic profile is available. In addition to this already available approach, a different method to calculate the profile will be discussed in section 3.2. This different method is very similar to the discrete model used by Schmeitz. Therefore, the discrete model will be first discussed in section 3.1. Finally section 3.4 deals with the three-dimensional models.

## 3.1 The discrete model

The discrete method was developed by Schmeitz, and differs from the analytical method in that it discretises the ellipse and the road for the evaluation of the contact points. To know what the height of the ellipse is, this discretised area is evaluated. This is done by summing the height of the ellipse and the height of the road, at each of these discretised points. At the point where this sum is largest, the ellipse touches the ground and so the height of the ellipse can be determined. In *Figure 3-1* the principle of the discrete method is displayed.

In Matlab this model is implemented in the following way (see appendix A.7). First, a search area is created: this has the size in which the elliptical cam model is valid. After that this area is divided in a number of dicretisation points (599 in this case). It is important to check that this is an odd number otherwise there would not be a discretisation point (later referred to as dpoint) in the middle of the search area. This will lead to an offset in the calculated height, when the ellipse stands on a flat surface. Hereafter a vector is created containing the heights of the ellipse at every dpoint. After this vector is created it is summed with the road profile and the maximum value of the summed height is searched. At this dpoint the 'gap', see *Figure 3-1*, is the smallest. This maximum value is the height of the centre of the ellipse. By extracting half the height of the ellipse ($b_e$) from this centre height, the height of the ellipse at the evaluated point can be found (this point lies in the middle of the search area at the bottom of the ellipse see *Figure 3-1*.the height of this point is then stored and the ellipse moves a position further on the road profile, where the whole routine starts over again. After the whole road profile is evaluated, the stored heights can be plotted to give the basic profile. One can expect that this method is not always very efficient, due to the large number of evaluations that have to be done.

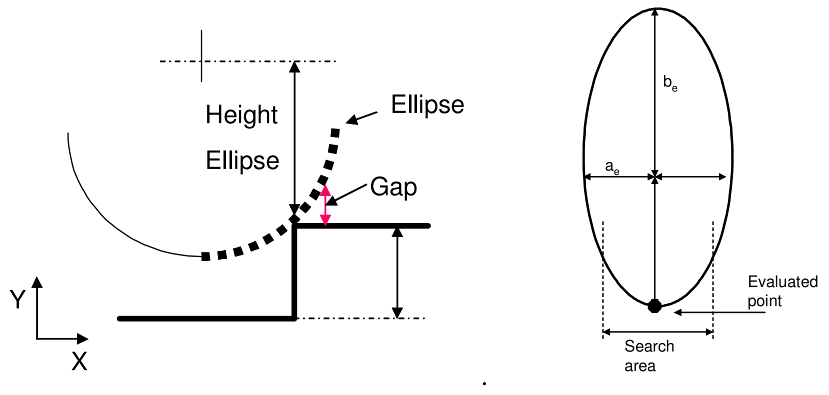


**Figure 3-1** *Working principle of the discretised model. And the used ellipse for the evaluations.*

## 3.2 The pre-evaluation model

Although the basis of this new model is the same as the discrete model, another approach of implementing is used. The idea is not to discretise the search area on the ellipse, but to look which road points fall in this search area, as is diplayed in *Figure 3-2*. Only these road points are evaluated, in the same way as is done by the discrete model. This approach has as advantage that always the same points on the ellipse are evaluated (assume a constant grid of the road surface), and thus can be determined in advance (pre-evaluation). The ellipse area does not have to be discretised, and the road surface does not need interpolation (to be able to sum the heights of the ellipse and road profile). This will result in a faster algorithm. The drawback is that a continuously and known grid has to be used or created for the road surface. In appendix A.8 this approach is implemented.

To work with a road profile, it is first interpolated to a predetermined distance so that it is assured that the sample interval is known. After that a vector is created, which contains the height of the ellipse at the points that will be evaluated. The length of this vector depends on the amount of points that fit in the size of the searching area. The method to evaluate where the ellipse touches the ground and what the corresponding height of the ellipse will be is the same as discussed in the previous paragraph.

Because always the same points on the ellipse are used to evaluate the height, it can occur that some of these points fall outside the road profile. See for example *Figure 3-2*. If the road profile starts by the cross and goes further to the right, the point on the left of the cross falls outside the road profile. To deal with this, a selection is made which investigates this. When a point falls outside the road profile (evaluation domain) the algorithm goes further with the next point to evaluate. This is under the assumption that points outside the evaluation domain do not influence the points in the evaluation domain.

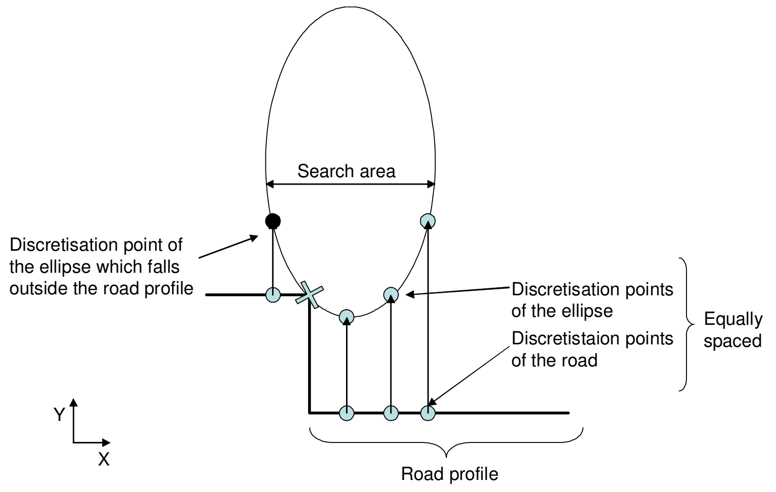

Search area
Discretisation point of the ellipse which falls outside the road profile
Discretisation points of the ellipse
Discretistaion points of the road
Equally spaced
Y
X
Road profile

***Figure 3-2*** *Working principle of the pre-evaluation model.*

## 3.3 Validation of the discrete models

The validation is done in the same way as for the analytical model, using the discrete model of Schmeitz. Also the same cleats and road part are evaluated. The parameters that are used in these evaluations are displayed in *Table 3-1*. The parameters for the tire and ellipse are the same as used before, and can be found in *Table 2-2*.

When looking at *Figure 3-3*, one can see that the basic profiles agree well, with exception for *Figure 3-3*d. Here one can see that the result of the pre-evaluation model does not agree with the discrete model. The cause of this, is that the cleat is discretised with an interval of 0.005 [m], while the top of the cleat is located at -0.0375 [m] with a height of 0.015 [m]. Hence no discretised cleat point will fall on this top, and so the cleat is thought to be lower. *Figure 3-4* demonstrates the top in the case that a sample interval of 0.0005 [m] is used. One can see clearly that the discrete model also has its inaccuracies due to the location of the top. This will also be explained in paragraph 4.1.2 and is called the first effect.

*Figure 3-5* shows the results for the created cleats. Almost every result in this figure seems to be correct but if one takes a closer look at *Figure 3-5*d (cleat 14), it can be seen that the basic curves do not fully correspond. To illustrate this better, *Figure 3-6* displays a small part of cleat 14. The left figure is created from the same simulation as is used for *Figure 3-5*d, the right figure is made with a sample interval of 0.0005 [m]. The effect one can see here is that the use of less points leads to a difference in height. This will also be explained in paragraph 4.1.2, and is called the third effect. For the road profile in *Figure 3-7* it is clear that the result agree well. Finally, despite the differences, one can say that the pre-evaluation model is valid for the standard cleats, because changing the sample interval gives the correct results.

| **Pre-evaluation model parameters:** | | |
|---|---|---|
| X_ellipse | 0.120 | [m] |
| X_step | 0.005 | [m] |
| | | |
| **Discrete model parameters:** | | |
| Disc.points ellipse | 599 | [-] |
| X_ellipse | 0.120 | [m] |
| X_step | 0.005 | [m] |

***Table 3-1*** *Parameters used in the simulations.*

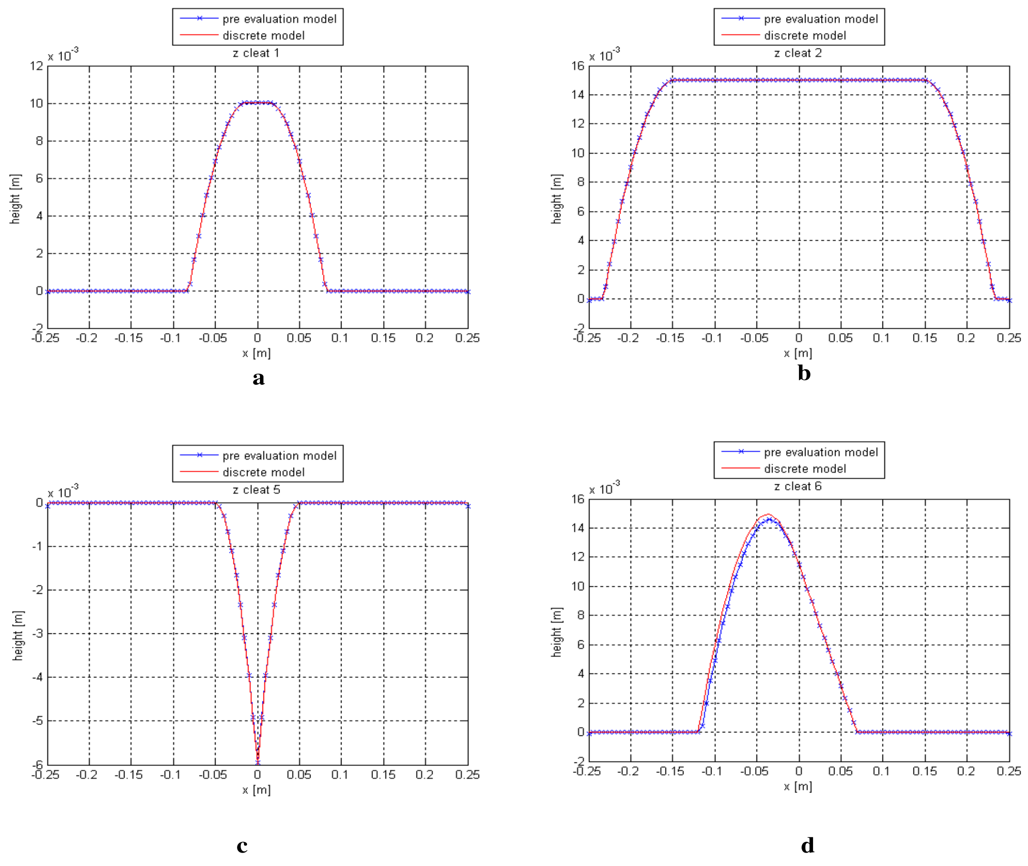


***Figure 3-3*** *Validation results, comparison of basic curves for standard cleats.*

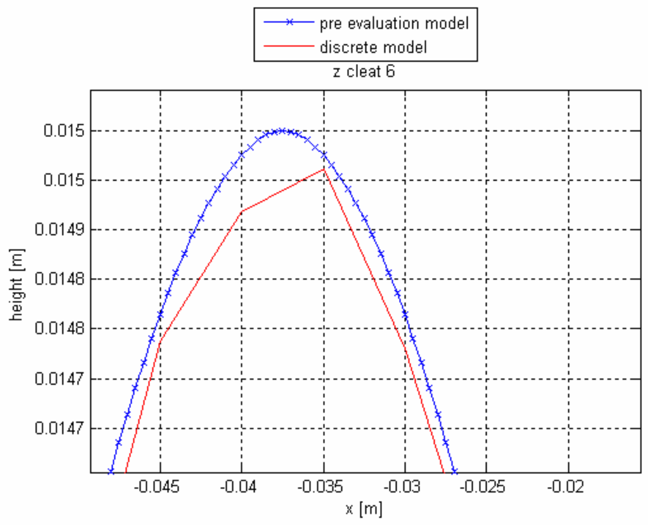


***Figure 3-4*** *Results of the top of cleat 6 with a sample interval of 0.0005 [m] for the pre-evaluation model, so that there is a discretisation point right on the top of the cleat.*

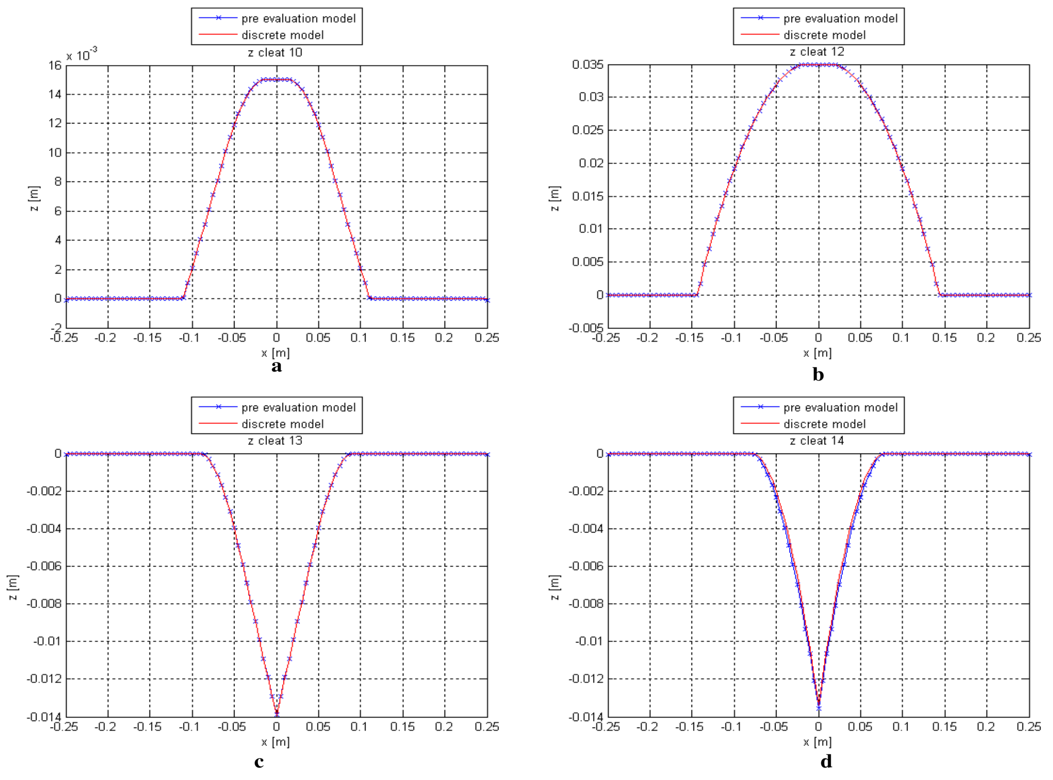


***Figure 3-5*** *Validation results, comparison of basic profiles for created cleats.*

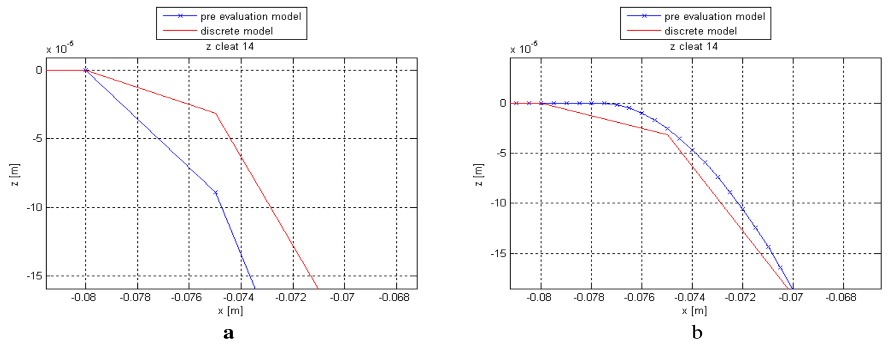


***Figure 3-6*** *Differences in heights due to the number of points. The discrete model uses the same sample interval for figure a, b. The pre-evaluation model uses in figure b a smaller sample interval (1/10 of the old interval) and has therefore more points to evaluate.*

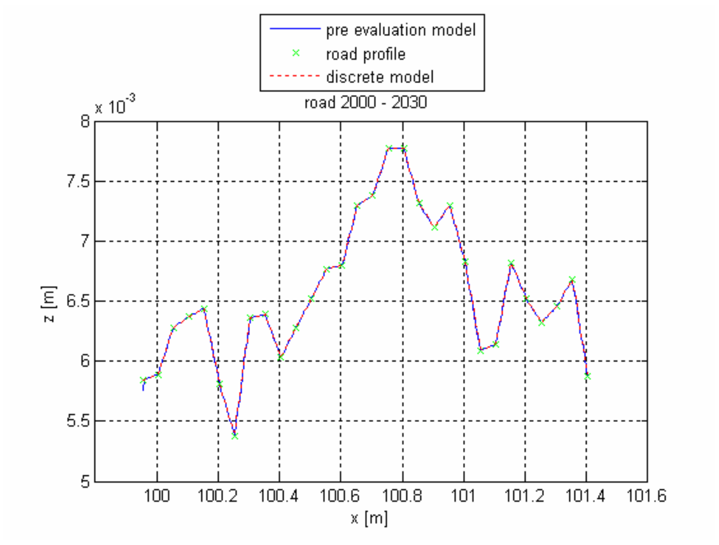


***Figure 3-7*** *Validation results, comparison of basic curves for a piece of road profile.*

## 3.4 Discrete method, three-dimensional models

As discussed before, a three-dimensional model is desired. This paragraph will discuss the three-dimensional discrete models. The models here will only consist of one track and not, as discussed in paragraph 2.6, of several tracks. This makes the models easier. And, as discussed in paragraph 2.6, when there is a model for one track, the model with more tracks can easily be generated. But before discussing the models, paragraph 3.4.1 will explain how the three-dimensional road surface is generated.

### 3.4.1 Generation of a three-dimensional road profile

Because of the lack of a real measured three-dimensional road profile, a road profile is created by hand. This is done with Matlab and the m-file for this can be found in appendix A.9. Keep in mind that this is an artificial road profile and it is therefore relatively small in size.

To create a road profile, the m-file uses a combination of a sine and cosine function to calculate the height at each road point. Hereafter the height is interpolated to get the desired grid. One can also use a smaller step size for x and y in the creation of the profile, but this will lead to a more irregular surface which is not desired at this moment. The road profile used for the simulations is: road_generated_3d22-mar-2005.mat. The profile is also displayed in *Figure 3-8*.

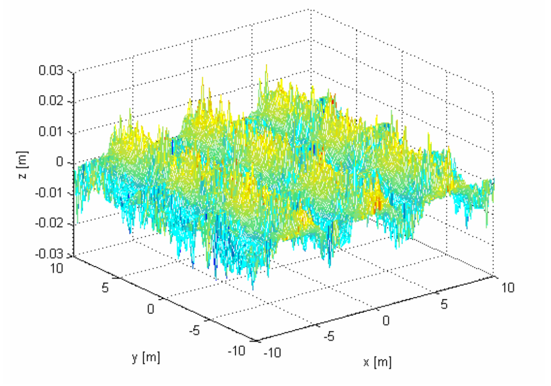


***Figure 3-8*** *Generated road profile.*

### 3.4.2 Three-dimensional discrete model

The three-dimensional discrete model has the same basis as the two-dimensional discrete model. It discretisies the search area and looks where the ellipse touches the ground to determine the height at the evaluation point of the ellipse. See paragraph 3.1.

To work in a three dimensional environment, use has been made of a local coordinate system which lies on the road point that is evaluated. This local coordinate system has the search area of the ellipse on its x axis, and nothing on its y axis. This is displayed in *Figure 3-9*. In this figure one can see where the local coordinate system is located on the driven trajectory. To be able to work with this local coordinate system the coordinates of the search area are transformed back to global coordinates using a rotation matrix. Hence the search area is described in global coordinates. Hereafter, the road surface can be interpolated in this search area, resulting in a vector that includes the heights. This makes it possible to determine the contact point using the method discussed in paragraph 3.1.

The path that will be driven on the created road surface will be a sine function, based on the global x coordinate. This will have as advantage that the trajectory is known, and that the ellipse makes all the possible steering angles, with respect to the road surface.
This approach is implemented (see appendix A.10) on the following way. First the trajectory is generated. Hereafter the height of the evaluation point of the ellipse (see *Figure 3-1*) is determined for each global x coordinate. This evaluation consists of five steps, which are listed below:

1) Deriving the angle ($\psi$) between the global and local coordinate systems for the rotation matrix.
2) Creating the search area in local coordinates.
3) Transforming it back to global coordinates.
4) Interpolating the road surface on to the desired search area in global coordinates.
5) Searching the contact point and storing the height of the evaluated point.

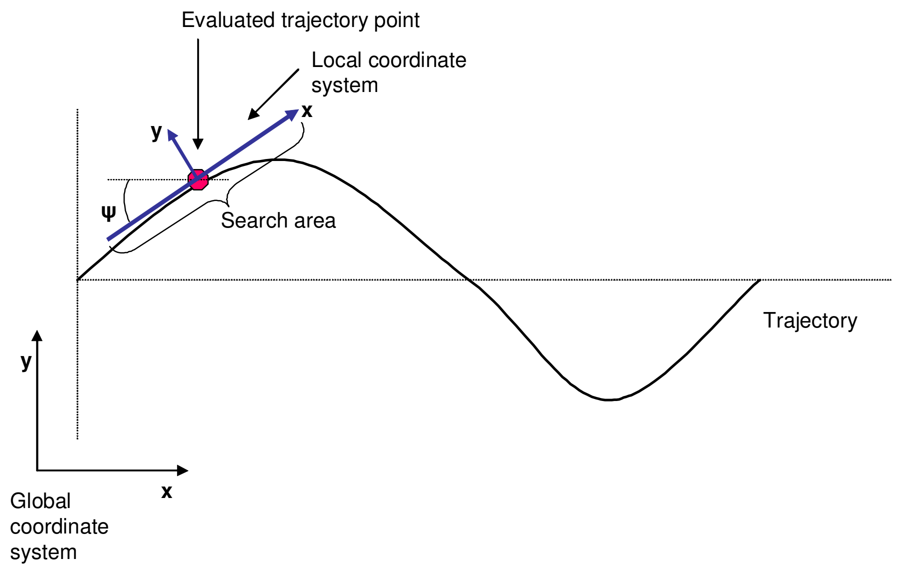


***Figure 3-9*** *Overview of the three-dimensional discrete model approach.*

### 3.4.3 Three-dimensional pre-evaluation model

This three-dimensional version of the pre-evaluation model has got the same basis as the two-dimensional model discussed in paragraph 3.2. The difference here is that multiple directions are considered, as is displayed in *Figure 3-10*. Hence the height of an evaluated road surface point is stored for multiple directions. This is done for the whole road surface leading to a matrix that contains the heights for every road surface point (multiple heights).

When driving over the road surface, one knows at which coordinates the tire is located and at which direction with respect to the global coordinate system. By using the heights and directions for the road surface points, one can find the corresponding height at an evaluated point, by interpolation. Therefore one only needs the matrix containing the heights of the road profile in the simulation. Hence this matrix can be determined once, for the whole road profile, before the simulation. So less work has to be done in the simulation.

This pre-evaluation of the road profile, and the calculation of the heights of the ellipse (see paragraph 3.2) before the simulations, are the main advantages of this model. In the implemented model (see appendix A.11) it is chosen to look at four directions with the angels 0, 45, -45 and 90 degree, to keep a simple algorithm. Hence the pre-evaluation part is almost the same as the two-dimensional model, only now it is placed at four directions and four heights are stored. To keep things simple, the model works with a known, equally spaced grid. Therefore no interpolation, between road points, is needed for the pre-evaluation, and everything can be done using counters. *Figure 3-11* shows the approach, the road surface here consists of 6 road points. At each of these points four directions are considered to evaluate the height of the road points. After the four heights are stored for all the six road points, the heights of the driven path can be determined. This will be further explained in the following pages.

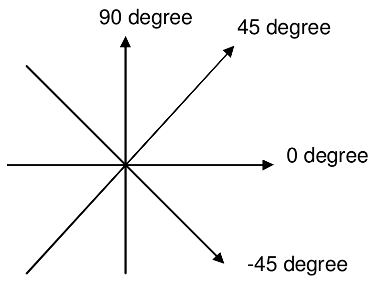


***Figure 3-10*** *Evaluated directions, four in the implemented model.*

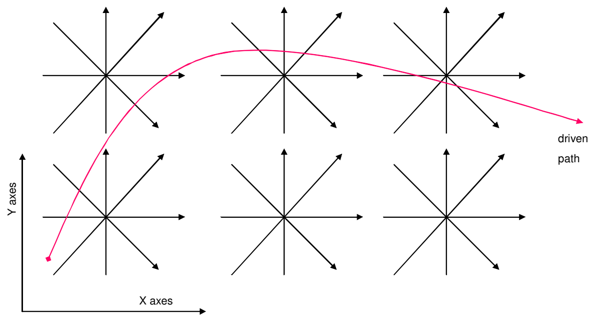


***Figure 3-11*** *Working principle of the pre-evaluation model, looking for every surface point in multiple directions. Coordinates are global coordinates.*

Also in this model a sine curved trajectory will be driven, based on the x axis. Because the road is already pre-evaluated (for some directions the heights are known at the road points), the only thing that has to be done is to use this data to calculate the appropriate height at the points on the driven trajectory. This is done in two stages.

Stage 1: The y coordinate of the driven trajectory is calculated with the sine function and is dependent on the x coordinate of the road surface. But this y coordinate is almost never the same as one of the defined points on the grid. So the first thing that has to be done is to determine the four different heights (multiple directions) at this calculated y coordinate. This is done by interpolating between the heights of the neighbour points, see *Figure 3-12*.

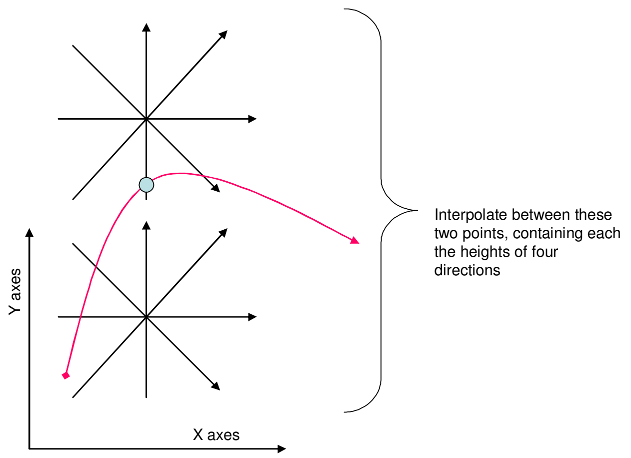

Interpolate between these two points, containing each the heights of four directions
Y axes
X axes

***Figure 3-12** Interpolating between two points on the y axes, the dot is the y coordinate of the trajectory.*

Stage 2: Now the four heights at the evaluated point of the trajectory are known, the correct height, with respect to the angle of the trajectory has to be determined. Because we make use of a sine curved trajectory, this angle of the trajectory is always known. But, this angle is almost never the same as one of the four directions evaluated. To get the right height one has to interpolate between the heights of the two nearest pre-evaluated directions. See *Figure 3-13*. A selection is used to determine which two directions have to be used.

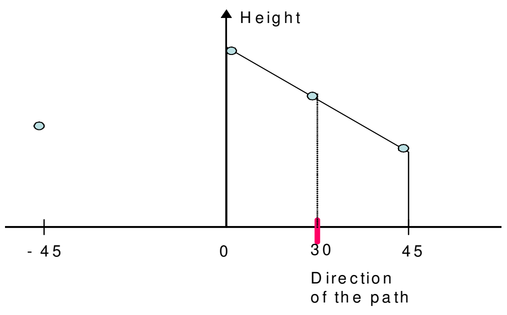


***Figure 3-13** Interpolating the height between two pre-evaluated directions.*

# 4 Comparison of the various models

In this chapter the different models will be compared on computing time and accuracy. First the two-dimensional models will be compared in paragraph 4.1. In paragraph 4.2 the three-dimensional models will be compared. Use has been made of "normal" parameters so that a good trade-of is made between accuracy and computing time. These parameters can be found in Table 2, 3 and 4. Comparing these models it is assumed that the influence of the total tire simulation model, of which the enveloping model is a part, can be neglected.

## *4.1 Comparison of the two-dimensional models*

### 4.1.1 Comparison of the computing time

To compare the models with regard to calculating time, all the models have evaluated the same cleats/road piece. For the shape of the cleats is referred to paragraph 2.5.

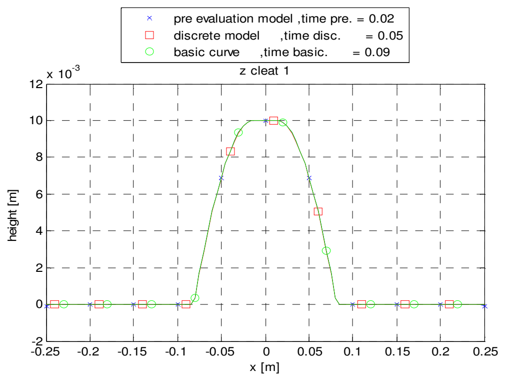
*Figure 4-1* displays the results. It can be seen that the pre-evaluation is faster than the two other models, and although not expected the discrete model is for the cleats faster than the basic curve model. The reason for this can be found by the input of the models. The discrete model uses a separate script with functions for the input of the cleats. Therefore it is not necessary to interpolate the profile, as discussed in paragraph 3.1. Hence one of the most time consuming parts of the discrete model is banned. In the situation of a measured profile this cannot be done, so the discrete model becomes slower. This is also displayed in *Figure 4-1*d.

When looking at *Figure 4-1*b and *4-1*c one can see that the basic curve model has a difference in computing time between cleat 10 and cleat 13. These cleats are both trapezoid shaped cleats in such a way that intersecting of the basic curves occurs. However cleat 13 has also two basic curves intersecting in the middle (X=0). The difference is also due to the fact that, for intersecting in a pothole a different formula is used, namely(2.9). In order to solve this formula in this case, a bigger search domain is used ($h < x$), which makes the calculation slower.

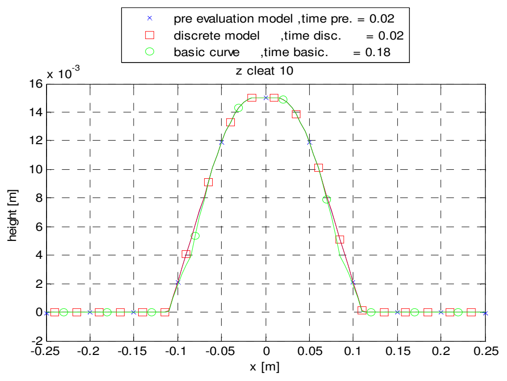
To evaluate whether the basic function model and pre-evaluation model hold their advantage with respect to the discrete model, different parts of the road profile where evaluated. To keep things simple, use has been made of the already available grid points of the road surface. 1000 grid points is about 50 [m] road profile. *Figure 4-2* and *Table 4-1* show the results. It is clear that the pre-evaluation model gives the best computing time. This is understandable when one looks at the models. The pre-evaluation model only evaluates at the grid points of the road profile, and uses no formulas (as the basic curve model) or interpolation (as the discrete model) for that. Hence this is the simplest model. The basic function model calculates which steps have been made in the road profile, and stores them in a matrix. This however, has to be converted into basic curves and with that a basic profile can be generated. This conversion is the big disadvantage of the model. In order to have the same smooth basic profile the heights of the basic curves have to be calculated at every grid point. This annihilates the advantage onto the other models, because still every point on the basic profile is "evaluated". The discrete model uses the road grid as well, but interpolates this at the desired "search area". It also uses formulas to calculate the height of the ellipse at the discretisation points of the ellipse. Hence this model has both disadvantages (interpolation and the use of formulas at each point) with respect to the computing time, and is consequently the slowest.

Concluding:

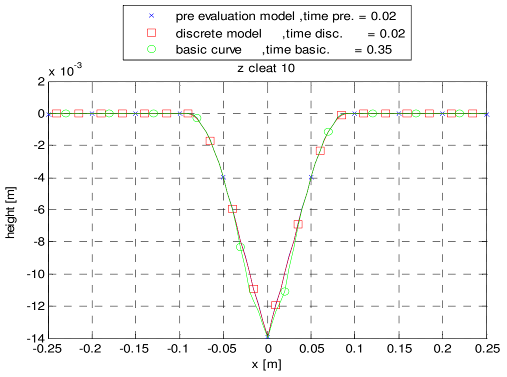
The basic curve model has an advantage with respect to the discrete model. But it is not as fast as the pre-evaluation model.



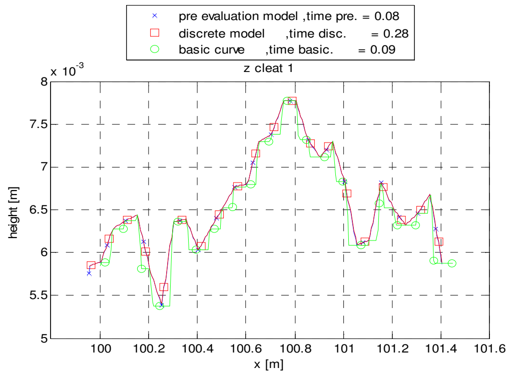
**a**



**b**



**c**



**d**

***Figure 4-1*** *Comparison of results and computing time (see legend) for the different models.*

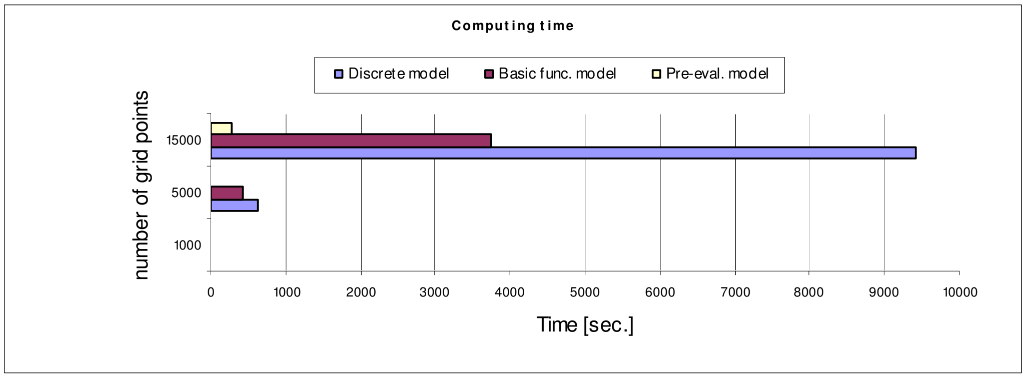


***Figure 4-2*** *Computing time with respect to the evaluated grid points.*

| Grid points | Discretised [s] | Basic func. [s] | Pre-eval. [s] |
|---|---|---|---|
| 2000-3000 | 31.40 | 17.30 | 1.60 |
| 3000-4000 | 30.50 | 17.60 | 1.60 |
| 5000-10000 | 631.90 | 419.00 | 33.50 |
| 5000-20000 | 9418.10 | 3739.70 | 276.00 |

***Table 4-1*** *Computing time with respect to the evaluated grid points.*

### 4.1.2 Comparison of the accuracy of the models and their error sources

As discussed and shown before, the models do not fully agree when evaluating a piece of road or a cleat. In paragraph 2.5 influence of the used number of steps and interpolation with respect to the created profile is discussed. In paragraph 3.3 the effect of the number of grid points is displayed. This paragraph will explain the reason of these deficiencies. This will be done by explaining the "effects" that are responsible for these inaccuracies.

**Effect 1:**

For all the models the number of measurement points of the road and their position is important. The effect caused by this is that for example the ellipse does not see the end of the cleat "clearly". Otherwise stated, the end position of the cleat is not the same as one of the measurement points of the road profile, as displayed in *Figure 4-3*.

Of course when one measures a road profile, one can say that the measurements represent the true profile. This suggests that the area between two measurement points will not influence the profile. But by interpolation the true measurement points can be lost. This is for example shown in paragraph 3.3, *Figure 3-4* where the top of the cleat is not seen.

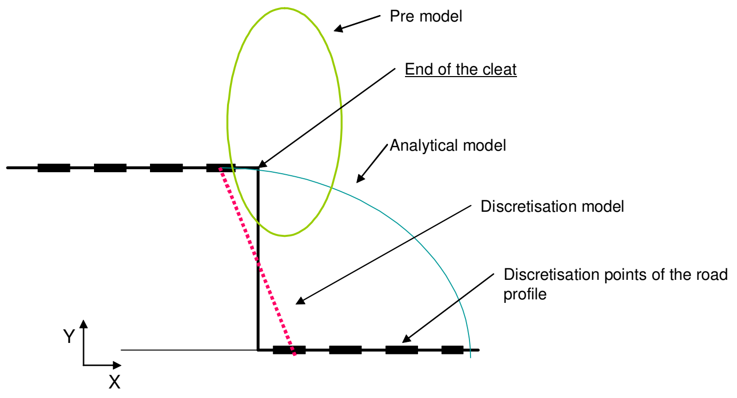


***Figure 4-3*** *The end of the cleat is not clear due to a too big measurement interval in the discretisation points of the road profile.*

**Effect 2:**

For the basic curve model, the road is approximated by steps, so that differences can occur when taking too large steps. This can for example be seen in *Figure 4-4*. Here one can see that the profile of the basic curve model and the line created by the discrete model differ. As shown in paragraph 2.5.2, increasing the number of steps will give a better agreement. However as shown in paragraph 2.5.3 and as discussed in the previous paragraph, this can lead to other deficiencies. One can also question if the approximation made by interpolation is correct. Consider for example a road profile consisting of square bricks, so that the road profile only consists of straight steps. Interpolating this profile will generate a change in steepness and so an incorrect evaluation. Hence one can say that it is not clear which model gives the correct profile, although one has to realize that the differences between the models are relatively small, and that the created profile by each model gives an overall global result.

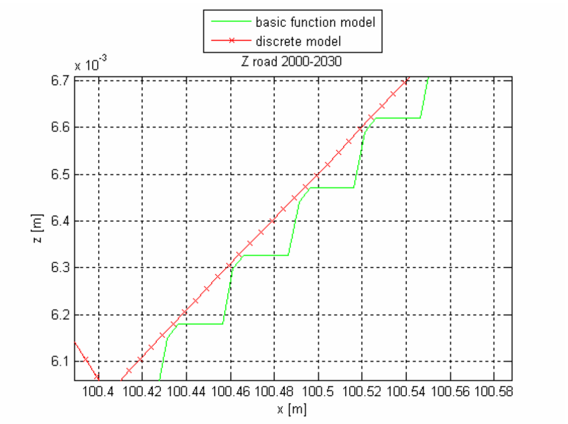


***Figure 4-4*** *Inaccuracy of the basic function model with respect to the discrete model due to too big steps.*

**<u>Effect 3:</u>**
As said before, the number of measurement points and interpolation play an important role. With the discrete model another thing appears. Namely the number of discretisation points of the search area. Using a small number of discretisation points, it is possible that a different height will be measured as is displayed in *Figure 4-5*. This is partly the same reasoning as with the first effect, with the difference that here the deficiency comes from the model itself. *Figure 4-5* shows what happens. The model calculates that the ellipse touches the cleat at the cross but it should calculate it as the dotted ellipse.

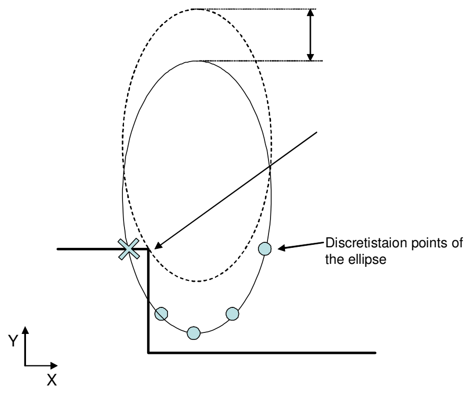


***Figure 4-5*** *Inaccuracy due to discretising the ellipse with a small number of discretisation points. These points are equally spaced.*

<u>Concluding:</u>
Interpolating the road profile should be prevented, independent of the model used. Interpolation will only give inaccuracies due to loosing curvature or missing true measurement points. Nevertheless, in all models created, interpolation is used when evaluating the road profile. This is done because the measured road profile uses a measurement interval which is too large, and because interpolation is part of the model (discrete model). The basic curve model has the benefit that no interpolating is needed, but to give a better agreement with the other models interpolation is used, in the case of a measured road profile. This plays a smaller role in the evaluated cleats, but if one evaluates the cleats exactly a difference in height can be spotted between the models. Hence one can say that a measured road profile must consist of a very fine grid. This will allow the basic function model and pre-evaluation model to work without interpolation, and so giving a result which is completely dependent on the input.

## 4.2 Comparison of the three-dimensional models

### 4.2.1 Comparison of the computing time

To compare the models with regard to computing time, all the models have evaluated the same generated road piece. This is the same as the one displayed in *Figure 3-8*. The trajectory driven is the same sine curved function as discussed before in paragraph 3.4.

As shown in paragraph 4.1.1, the two-dimensional pre-evaluation model is the fastest. This is not the case for the three-dimensional model, in contrast to what was expected. The explanation for this comes from the pre-evaluation model itself. The model first evaluates the whole road surface (every grid point), after which the trajectory is "driven" and evaluated. Hence not only the points which fall on the driven trajectory are evaluated, but also all the other points. As stated before, the big advantage of the pre-evaluation model would be that the road is pre-evaluated, and therefore only the stored data has to be used. This means that the pre-evaluation part of the model (so not the part that generates the profile) will not be taken into account with the computing time. The results are displayed in the table below. The time is also included with the pre-evaluation so that one can see what time it takes to pre-evaluate the road surface.

| | **Pre-evaluation model [sec.]** | **Discretised model [sec.]** |
|---|---|---|
| With pre-evaluation | 7.63 | 1.21 |
| Without pre-evaluation | 0.35 | 1.21 |

***Table 4-2*** *Computing times of the three-dimensional models.*

### 4.2.2 Comparison of the accuracy

Because both three-dimensional models are based on the two-dimensional models discussed before, the error sources (effect 1 – 3) of these models will play a role in the accuracy of the three-dimensional models. To compare the models, simulations with both models are made. The created basic profile is displayed in *Figure 4-6*. As can been seen in the figure, the two models appear not to agree very well. However, if one takes a side view one gets a plot like *Figure 4-7*. Here one can see that the figures do agree globally, except for the peaks (these are from the pre-evaluation model).

These differences are caused by the interpolation used. For example: *Figure 4-8* shows what happens with the discrete model. The discrete model interpolates to get the height at the search area. The line in this case is the interpolated search area of the model. One can see that the heights of the line differ from the actual height of the surface. Moreover the pre-evaluation model uses only four directions, and interpolating only between these directions will result in a less accurate height.

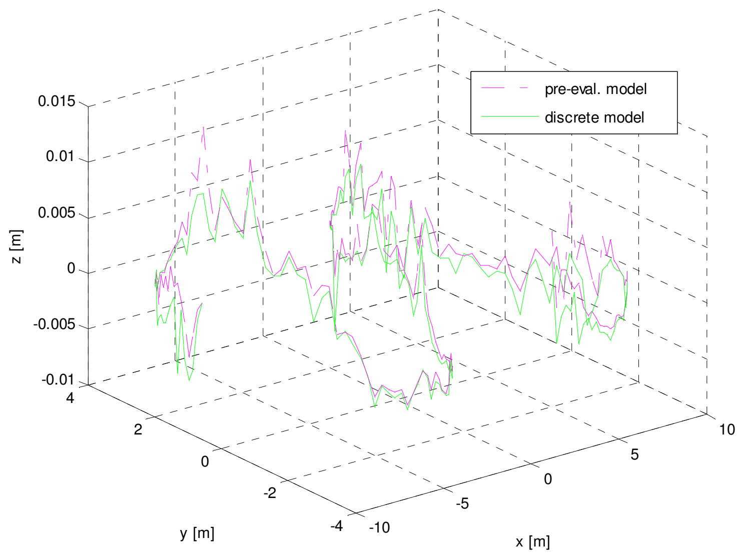


***Figure 4-6*** *Results for the , driving a sine curved path on the generated road.*

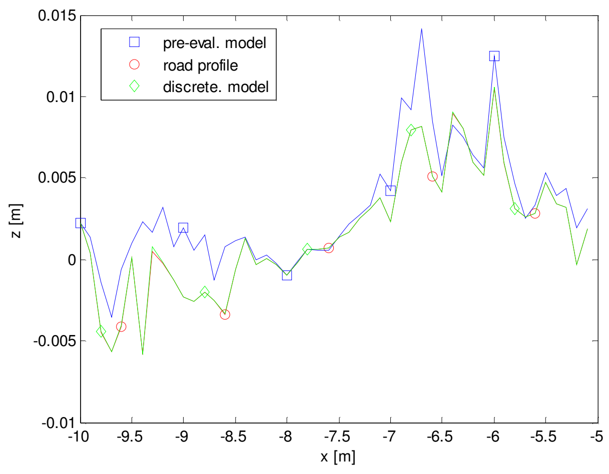


***Figure 4-7*** *Side view of the generated profile of Figure 4-6.*

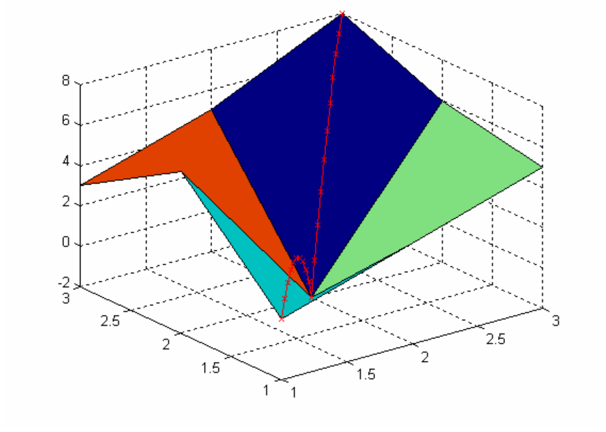


***Figure 4-8*** *Difference in height due to interpolation the marked line is the result of interpolating the height.*

# 5 Conclusion and recommendations

## 5.1 Conclusions

In the past 5 years Schmeitz has performed research on tire models that describe the response of a car tire on uneven road surfaces. In order to describe the quasi-static enveloping behaviour of a car tire, he developed the so-called tandem model with elliptical cams. Now the tandem model is fully developed and validated, it has to be implemented in software as efficiently as possible.

The subject of this research is to implement and investigate different methods to use the elliptical cam model on arbitrary road surfaces 2D as well as 3D and to compare the different methods and computing time. The aim is to find a good working model that can efficiently describe the basic profile, needed for the elliptical cam model.

Three models are implemented and investigated: the basic curve model, the discretised model and the pre-evaluation model. For the basic curve model it turned out to be impossible to create a three-dimensional model. Nevertheless the two-dimensional model shows some advantages with respect to the other models. For example, it is possible for the basic curve model to work with a rough measured road profile without using interpolation to refine the grid.

The discrete model is the slowest model, and although it is used to validate the other models, one has to be aware that this model also has its inaccuracies. Due to the used interpolation, differences can occur and using a small sample interval is therefore advised. The three-dimensional discrete model shows a good result, but validation is not possible because no other already validated model could be found. Instead of validating the model it is compared with the three-dimensional pre-evaluation model. The differences that are found between the models originate from the interpolation used.

The pre-evaluation model is the best model when one has a fine measured road profile with a known measurement interval. In that case no interpolation is necessary and the results are truly dependent on the measured road profile. One of the advantages of the three-dimensional pre-evaluation model is that the road profile can be pre-evaluated. Hence one can use the stored data to calculate the basic profile, which is more time efficient. Nevertheless the road profile has to be pre-evaluated, which consumes a lot of time, as shown in paragraph 4.2.1. One of the drawbacks of the pre-evaluation model is that the accuracy is dependent on how accurate the pre-evaluation is done. Using for example four directions, gives a simple algorithm but the accuracy could be improved by using more directions. Another drawback is that, to fully utilize the benefit of this model, the total road profile with grid has to be known before the simulation. Consequently, one cannot use the potential of the model in a real time simulation on an unknown road.

Interpolation of the road profile gives inaccuracies, and stimulates the effects as shown in chapter 4. One can also not justify the assumption that the area between two measurement points does not influence the basic profile. Hence an interpolated rough measured road profile can not be used in these kinds of simulations to get reliable results. It is therefore only used in this research to create and compare the models.

## 5.2 Recommendations

Although this research is done with best effort, recommendations can be made. The basic function model for example, uses the formulas from paragraph 2.3.5. These formulas are then solved with a minimization function of Matlab. One can investigate if there are other, better (faster) ways to solve this problem.

With respect to the pre-evaluation model, it is recommended to extend this model with more directions to see if the results become more like the results of the discrete model. It is also recommended to validate the models with real measured responses, and to determine the inaccuracies with these results. This will also give the possibility to validate the three dimensional models.

# References

1. Schmeitz, A.J.C., A Semi-Empirical Three Dimensional Model of the Pneumatic Tyre Rolling over Arbitrarily Uneven Road Surfaces. 2004 ISBN 90-9018380-9

2. Schmeitz, A.J.C., An efficient dynamic ride and handling tire model for arbitrary road unevennesses. VDI-berichte nr. 1632 pp. 173-199, VDI verlag GmbH, Dusseldorf, Germany, 2001.

3. Heath, M.T., Scientific Computing An Introductory Survey. McGraw-Hill, 2002. 2nd ed. ISBN 0-0-112229-x

# Appendix A M-files.

## A.1

```
% ********************************************************************
% basic curve model using ellipsoids / analytically created basic curves
% *******************************************************************
% c     =   ellipsoid power     [-]
% ac        =   ellipsoid width      [-]
% bc        =   ellipsoid height        [-]
% Fz0       =   vertical load           [N]
% *******************************************************************
% x     =   step position       [m]
% xb    =   begin position basic curve [m]
% xe    =   end position basic curve [m]
% h     =   step height [m]
% *******************************************************************
% Written by S.H.M. kersjes
% *******************************************************************
close all
clear
clc
% Tyre parameters
R=312e-3;   % tyre radius (including tread element height) [m]
parash  = 0.8773;   % Shift     [-]
paraac  = 1.0325;   % ac        [-]
parabc  = 1.0306;   % bc        [-]
parac       = 1.8230;   % c         [-]
% Model parameters
ac          =   paraac*R;
bc          =   parabc*R;
c           =   parac;
%************* road *************
%     X       h      Xb       Xe      Lb
A = [ -250    0      0        -700    0;
       -20    5      0        -20     0;
       -15    5      0        -15     0;
         0    0      0         0      0;
        15   -5      15        0      0;
        20   -5      20        0      0;
       250    0      0        700     0];
  A=A/1000;
% *** evaluation bij steps_28012005 ****
TIME0=cputime;
[A_begin,A_end] = steps_28012005(A,ac,bc,c);
A_begin
disp('       x          h       xb       xe       lb')
A_end
disp('       x          h       xb       xe       lb')
% *** calculating the effective road profile ****
x_begin=(A_eind(1,3));
t = length(A_eind(:,1));
x_end=A_eind(t,4);
x_step=0.005;
x=x_begin:x_step:x_end;
i = 1;
Z2 = 0;
while i < length(A_eind(:,1))
    hs = A_eind(i,2);    % step height
    x0 = A_eind(i,1);
    xb = A_eind(i,3);
    xe = A_eind(i,4);
[Z]=eltandem_2(x,c,ac,bc,hs,x0,xb,xe);
Z1 = Z;
Z2 = Z2+Z1;
i = i+1;
end
disp('calc. time')
TIME_tandem=cputime-TIME0;
```

## A.2

```
% selection algorithm for steps in a road surface 2-D
% S.Kersjes 28-1-2005
function [A_begin,A_end] = steps_28012005(A,ac,bc,c)
%**** ellips *****************
ae = ac;
be = bc;
ce = c;
%**** counters  ***************
i = 0;
%***************************************************
%          complete storage matrix A             %
%***************************************************
    i = 1;
    while i <= length(A(:,1)) ;
        A(i,5)= ae*( 1- (1-(abs(A(i,2))/(be)) )^ce )^(1/ce);
        i = i+1;
    end
    i = 1;
    while i <= length(A(:,1)) ;

        if A(i,2) > 0
        A(i,3)= A(i,1)-A(i,5);
        elseif A(i,2) < 0
        A(i,4)= A(i,3)+A(i,5);
        elseif A(i,2) == 0
        A(i,3) = A(i,1);
        A(i,4) = A(i,1);
        end
        i = i+1;
    end
%**** copy A *****************
A_copie = A
%***********************************
%       selection samples          %
%***********************************
i = 1; % you want to look one step in front and one step behind
while i < length(A(:,1));

    if A(i,2) > 0
        if A(i+1,2) >= 0
        [A,i] = going_up(A,i,ae,be,ce);
        elseif A(i+1,2) < 0
        disp(' top ')
        i= i+1;
        else
        disp('error steps')
        i = i+1
        end

    elseif A(i,2) < 0
        if A(i+1,2) <= 0
        [A,i] = going_down(A,i,ae,be,ce);
        elseif A(i+1,2) > 0
        [A,i] = puthole(A,i,ae,be,ce);
        else
        disp('error steps')
        i = i+1
        end

    elseif A(i,2) == 0
         disp('  no step   ')
         i = i+1;
    else
         disp('error steps')
         i = i+1
    end
end
A_begin = A_copie;
A_end = A;
```

## A.3

```
function [A,i] = going_up(A,i,ae,be,ce);
        lbs = ae*( 1- (1-(abs(A(i+1,2)+A(i,2))/(be)) )^ce )^(1/ce);
        hs  = A(i+1,2)+A(i,2);
        xbs = A(i+1,1) - lbs;
        %************** no overlap ************************
        if A(i+1,3) >= A(i,4)
        disp('   NO OVERLAP UP    ')
        i = i+1;
        %************** full overlap ******************
        elseif xbs <= A(i,3)
        [i,A] = overlap_going_up(A,i,hs,xbs);
        %************* intersecting *******************
        elseif (( A(i,4) > A(i+1,3) ) & ( A(i+1,3) > A(i,3) ))
        [i,A] = cross_going_up(A,be,ae,ce,i,lbs);
        else
        disp('error going up!!!!!!!!!!!!!!!!!!!!!!!')
        i = i+1;
        end
```

## A.4

```
function [i,A] = overlap_going_up(A,i,hs,xbs);
A(i+1,2) = hs;
A(i+1,3) = xbs;
A(i+1,5) = A(i+1,4)-A(i+1,3);
             k=i;
             B = [];
                while k < length(A(:,1));
                A(k,:) = A(k+1,:);
                k =k+1;
                end
    A = A(1:end-1,:)
    disp('   OVERLAP GOING UP   ')
    i = i - 1
```

## A.5

```
function [i,A] = cross_going_up(A,be,ae,ce,i,lbs);
% *** domain of h2s ****
     x1 = A(i+1,2)              %h2
     x2 = (A(i+1,2)+A(i,2)) ; %h2 + h1
% *** tolerance ****
     options = optimset('Display','iter','tolX',1.0e-10,'largescale','on');
     param=[];
% *** optimum ****
     [x,fval,exitflag] = FMINBND('fun' ,x1,x2,options,param,A,i,be,ae,ce,lbs )
% *** matrix A update ***
     hs = x;
     xs = A(i+1,1)-ae*(1- (1- (abs(hs)/be) )^ce)^(1/ce);
     lbs = ae*( 1- (1-(abs(hs)/(be)) )^ce )^(1/ce)
     A(i+1,2)    = hs;
     A(i,4)      = xs;
     A(i+1,3)    = xs;
     A(i+1,5)    = A(i+1,4)-A(i+1,3);
     A(i,5)      = A(i,4)-A(i,3);
% *** display to workspace ***
     disp('   CROSS GOING UP   ')
     i = i+1 ;
```

## A.5.1

```
function f = fun(x,param,A,i,be,ae,ce,lbs );
h1 = A(i,2);
h2 = A(i+1,2);
xe2 = A(i+1,1);
xe1 = A(i,1);
f = abs(  abs(h1)-be+ abs( be* (1- (  abs( (xe2-ae*(1- (1- abs(x)/be)^ce)^(1./ce))
-xe1)  /ae)   ^ce )^(1/ce) )    +abs(x) -abs(h1) -abs(h2)   );
```

## A.6

```
function [Z]=eltandem_2(dxm,c,ac,bc,hs,x0,xb,xe);
[mdxm,ndxm]=size(dxm);
if mdxm<ndxm,
   dxm=dxm';
end
% Basic curve
if hs>=0,
    Z=hs-bc+bc*( (1-(abs(-bound(dxm-x0,xb-x0,xe-x0))/ac).^c).^(1/c));
    Zs =hs-bc+bc*( (1-((x0-xb)/ac).^c).^(1/c));
    Z = Z - abs(Zs);
elseif hs<0,
    Z =-bc+bc*( (1-((bound(dxm-x0,xb-x0,xe-x0))/ac).^c).^(1/c));
    Zs =-bc+bc*( (1-(abs(xb-x0)/ac).^c).^(1/c));
    Z = Z + abs(Zs);%compensation offset in Z direction
end
```

## A.7

```
% **********************************************************************
% Discrete model
% *********************************************************************
% Xs            =   global position vector      [m]
% X0            =   global position obstacle    [m]
% c         =   ellipsoid power             [-]
% ac            =   ellipsoid width             [-]
% bc            =   ellipsoid height                [-]
% i_cleat   =   obstacle number             [-]
% *********************************************************************
% Rewritten by S.H.M. Kersjes
% *********************************************************************
clear
clc
close all
% domain
x0=0;
x_begin=-0.25;
x_end=0.25;
x_step=0.005; % 5 mm
sv=1;
x=x_begin:x_step:x_end;
R=312e-3;    % tyre radius (including tread element height) [m]
% Ellipsoid model parameters
% *********************************
parash  = 0.8773;   % Shift     [-]
paraac  = 1.0325;   % ac        [-]
parabc  = 1.0306;   % bc        [-]
parac       = 1.8230;   % c         [-]

for i_cleat=[1 2 3 4 5 6 7 8 10 11 12 13 14 15],
            TIME0=cputime;
            % Model parameters
            ac          =   paraac*R;
            bc          =   parabc*R;
            c           =   parac;
            [Z]=elmodel_Z(x,x0,c,ac,bc,i_cleat);
            TIME=cputime-TIME0;
            if sv==1,
                Testname=['eltandemc' num2str(i_cleat) 'l' num2str(round(4000/1000))
'lc'];
               eval(['save ',Testname,' x Z TIME']);
            end
      end
```

## A.8

```
% ********************************************************************
% pre evaluation model
% ********************************************************************
% Xs            =   global position vector      [m]
% X0            =   global position obstacle    [m]
% c         =   ellipsoid power             [-]
% ac            =   ellipsoid width             [-]
% bc            =   ellipsoid height                [-]
% x_ellipse     =   1/2 search area                 [m]
% ********************************************************************
% Written by S.H.M. Kersjes
% ********************************************************************
clc
clear
close all
% parameters
R=312e-3;
parash  = 0.8773;   % Shift     [-]
paraac  = 1.0325;   % ac        [-]
parabc  = 1.0306;   % bc        [-]
parac       = 1.8230;   % c         [-]
ac          =   paraac*R;
bc          =   parabc*R;
c           =   parac;
x_ellipse = 0.120;
% trapezoid cleat 1
X = [-0.25   0;
     -0.025 0;
     -0.015 0.01;
      0.015 0.01;
      0.025 0;
      0.25   0];
XI = [-0.25:0.005:0.25];
B = INTERP1(X(:,1),X(:,2),XI)
% ***** set time **************
TIME0=cputime;
%***** creating hellips*****
xh=[-x_ellipse:0.005:x_ellipse];
hellips=abs(-bc*(1-(abs(xh)/ac).^c).^(1/c));
%**************  2-D *****************
for i = 1:length(B);
    h  =[] ;
    k = 1;
        for l= i-24:i+24;
            if ((l >1) & (l < length(B)))
            h(k) = hellips(k)+B(l);
            else
            %disp('outside domain')
            end

            k = k+1;
        end
    z1=max(h);
    A(i) = z1-bc;
end
disp('calc. time')
TIME_pre=cputime-TIME0;
```

## A.9

```
%*******************************************
%   road surface generator                %
%*******************************************
% written by S.H.M. Kersjes               %
%*******************************************
close all
clear
clc
x = [-10:0.2:10];
y = [-10:0.2:10];
for i=1:length(x);
    for j=1:length(y);
    rp = random('Poisson',1:1,1,1);
z(i,j) = (1/(rp+1)*sin(x(i))+rp*cos(y(j)))/250;
    end
end
[x y] = meshgrid(-10:0.2:10);
[XI YI] = meshgrid(-10:0.1:10);
ZI = interp2(x,y,z,XI,YI);
figure
mesh(XI,YI,ZI)
xlabel('x [m]')
ylabel('y [m]')
zlabel('z [m]')
XX =XI
YY =YI
ZZ =ZI
Testname=['road' '_' 'generated' '_' '3d',date];
               eval(['save ',Testname,' XX YY ZZ']);
```

## A.10

```
%**************************************************
%   3-D discrete model                            %
%**************************************************
% written by S.H.M. Kersjes                       %
%**************************************************
clear
clc
close all
%**** parameters******
A = 3
a = pi/5
b = 0.2
R=312e-3;
paraac  = 1.0325;   % ac        [-]
parabc  = 1.0306;   % bc        [-]
parac       = 1.8230;
ac          =   paraac*R
bc          =   parabc*R
c           =   parac
%***** road *******
load('D:\blok5_vakken\stage\3d_pre_model\road_generated_3d22-mar-2005.mat')
X = XX;
Y = YY;
Z = ZZ;
% **** set time******
TIME0 = cputime;
%**** trajectorie coordinates ******
x =(-10:0.1:10);
y =A.*sin(a.*x+b);
for j=[1:length(x)];
    xg = x(j);
    dydx = A*a*cos(a*xg+b);
    phi = atan(dydx);
    %*** local coordinates *****
    x_ellipse = 0.120;
    x1=linspace(-x_ellipse,0,24);
    x2=linspace(0,x_ellipse,24);
    xl=[x1(1:end-1) x2];
    yl = 0;
    %***** rotation matrix ******
    R = [cos(phi)  sin(phi);
         -sin(phi) cos(phi)];
    %**** global coordinates ****
    for i=[1:length(xl)];
    D(i,:) = [xl(i) yl]*R+[x(j) y(j)];
    end
    %***** interpolating the height *****
    XI = D(:,1);
    YI = D(:,2);
    ZI= interp2(X,Y,Z,XI,YI);
    %***** calculating height ******
    hellips=abs(-bc*(1-(abs(xl)/ac).^c).^(1/c));
    h=hellips+ZI';
    [maxh,ii]=max(h);
    Zl(j)=maxh-bc;
end
```

## A.11

```
%***************************************************
%   3-D pre-evaluation model                       %
%***************************************************
% written by S.H.M. Kersjes                        %
%***************************************************
clc
clear
close all
AA = 3;
a = pi/5;
b = 0.2;
R=312e-3;
paraac  = 1.0325;   % ac        [-]
parabc  = 1.0306;   % bc        [-]
parac       = 1.8230;
ac          =   paraac*R;
bc          =   parabc*R;
c           =   parac;
QQ = 0; % 1 calculates the surface again
time0 = cputime;
if QQ == 1
%*********loading the generated road ************
load('D:\blok5_vakken\stage\3d_pre_model\road_generated_3d22-mar-2005.mat')
XX = XX;
YY = YY;
ZZ = ZZ;
B =ZZ;
% *** creating hellipse ***********
xl_1 = [];
xl_2 =[];
for c =1:7; % 7 road points in the ellipse
d = -0.3+c*0.1-0.1;
xl_1 =[xl_1 d];
end
hellips_1=abs(-bc*(1-(abs(xl_1)/ac).^c).^(1/c));
c = 1;
for c = 1:5
d = -0.2828+c*0.1414-0.1414
xl_2 =[xl_2 d];
end
hellips_2=abs(-bc*(1-(abs(xl_2)/ac).^c).^(1/c));
%******* evaluating the road for four angles *************
for i = 1:length(B(1,:));
    for j =1:length(B(:,1));
%**************  0 degree ******************
    h  =[];
    k = 1;
        for l= i-(length(xl_1)-1)/2:i+(length(xl_1)-1)/2;
            if l < 1;
            %disp('outside domain')
            elseif l > length(B(1,:));
            %disp('outside domain')
            else
            h(k) = hellips_1(k)+B(j,l);
            end
            k = k+1;
        end
    z1=max(h)-bc;
    A(j,i,1) = z1;
%*************** 90 degree*****************
    h = [];
    k = 1;
        for l= j-((length(xl_1)-1)/2):j+((length(xl_1)-1)/2);
            if l < 1;
            %disp('outside domain')
            elseif l > length(B(:,1));
            %disp('outside domain')
            else
            h(k) = hellips_1(k)+B(l,i);
            end
            k = k+1;
        end
    z2=max(h)-bc;
    A(j,i,2) = z2;
```

```
%**************** 45 degree *****************
    h = [];
    k = 1;
    p = j+((length(xl_2)-1)/2);
        for l= i-((length(xl_2)-1)/2):(i+(length(xl_2)-1)/2);
            if l < 1;
            %disp('outside domain')
            elseif l > length(B(1,:));
            %disp('outside domain')
            elseif p > length(B(1,:));
            %disp('outside domain')
            elseif p < 1;
            %disp('outside domain')
            else
            h(k) = hellips_2(k)+B(p,l);
            end
            k = k+1;
            p = p-1;
         end
    z3=max(h)-bc;
    A(j,i,3) = z3;
%*************** -45 degree *****************
    h = [];
    k = 1;
    p = j-((length(xl_2)-1)/2);
         for l= i-((length(xl_2)-1)/2):(i+(length(xl_2)-1)/2);
            if l < 1;
            %disp('outside domain')
            elseif l > length(B(1,:));
            %disp('outside domain')
            elseif p > length(B(:,1));
            %disp('outside domain')
            elseif p < 1;
            %disp('outside domain')
            else
            h(k) = hellips_2(k)+B(p,l);
            end
            k = k+1;
            p = p+1;
         end
    z4=max(h)-bc;
    A(j,i,4) = z4;
   end
end
```

# Appendix B Matlab scripts

## B.1

```
% *** conversion ****
i = 2;
k = i;
A =[];
while i < length(B)
    if ((B(i-1,2) <= B(i,2))& (B(i,2) <= B(i+1,2))) %going up
        A(k,2) = B(i,2)-B(i-1,2);
        A(k,1) = B(i,1);
        A(k,4) = B(i,1);
        A(k,5) = 0;

    elseif ((B(i-1,2) < B(i,2))& (B(i,2) > B(i+1,2))) %top
        A(k,2) = B(i,2)-B(i-1,2);
        A(k,1) = B(i,1);
        A(k,4) = B(i,1);
        A(k,5) = 0;
        k = k+1
        A(k,2) = B(i+1,2)-B(i,2);
        A(k,1) = B(i,1);
        A(k,3) = B(i,1);
        A(k,5) = 0;
        disp('extra point??????????????????//')

    elseif ((B(i-1,2) >= B(i,2))& (B(i,2) >= B(i+1,2))) %going down
        A(k,2) = B(i+1,2)-B(i,2);
        A(k,1) = B(i,1);
        A(k,3) = B(i,1);
        A(k,5) = 0;

    elseif ((B(i-1,2) > B(i,2))& (B(i,2) < B(i+1,2))) %pothole
        A(k,2) = 0;
        A(k,1) = B(i,1);
        A(k,4) = B(i,1);
        A(k,5) = 0;
        disp('pothole')
    else
        B(i,1)
        disp('error')

    end
    k = k+1;
    i = i+1;
end
```

# Appendix C Comments on the M-files and Matlab scripts

**M-file A.11**
In this m-file there are two vectors containing the heights of the point on the ellipse (hellipse). This is done because fewer points fit in the search area when the ellipse stands under an angle of 45 or -45 degree. Also some parts are automated so only the begin part has to be changed as the grid size changes.

**Matlab script B.1**
One can see that there is a contradiction in this script when all three heights at the mpoints are the same (B(i-1,2)=B(i,2)=B(i+1,2)). Attempts to rewrite the script did not succeed, because the resulted scripts did not cover al the possibilities. Fortunately Matlab will not block on this, and it makes no difference which selection extends the matrix A, so that this selection can be used.