A collection of modelling problems carried out in the academic year 1991-1992 by the ECMI-students at the Eindhoven University of Technology

Opleiding Wiskunde voor de Industrie Eindhoven

Student Report 94-01

A Collection of Modelling Problems

1991 - 1992

Anjet den Boer
Hans Bonekamp
Peter Brand
Richard du Croo de Jongh
André Hendriksen
Aimé Herczog
Gerard Pruis
Paul Tijink

December 1993

A collection of Modelling Problems carried out in the academic year 1991 - 1992 by the ECMI-students at the Eindhoven University of Technology

Anjet den Boer

Hans Bonekamp

Peter Brand

Richard du Croo de Jongh

André Hendriksen

Aimé Herczog

Gerard Pruis

Paul Tijink

Table of contents

THE GUTTER PROBLEM

1 Introduction and Problem Description

In this report we handle the question of how large a gutter should be so that it will not overflow, given a certain amount of rainfall during a given period of time. With a certain amount of rainfall we mean a heavy shower One intuitively understands that this problem depends on the size and the angle of the roof.
Now a producer of gutters wants to have tables which he can use to recommend sizes of a gutter, given the size and the angle of the roof.
Modelling the income and outcome of water from a gutter we make the following assumptions:

- A roof has two leaning sides. The slope of a side varies between 30° and 60°.
- Every side is rectangular.
- A gutter hangs horizontally and has only one rain-pipe.
- A gutter can have two different shapes:
  -round
  -rectangular
- For every type of gutter there is a suitable type of rain-pipe.
- A gutter is hanged just below the edge of the roof.
- While it is raining there is no wind blowing, this means the rain falls vertically.

In the next section we give some results

2 Numerical Results

Using the model derived in the next section we can give some numerical results. (Using the fact that a heavy shower will be 1 mm of rainfall per minute, during 1 hour). The numbers in the tables represent the height of a gutter in centimeters and α is the angle of the roof.

Remark: we restrict ourselves to round gutters, because a hemispherical gutter with radius ρ and a rectangular gutter with height ρ and width $\frac{pi}{2}\rho$ give the same results.

Table I: size of gutters with $\alpha = 30°$

$b\backslash l$	10	25	50
5	0,0004	0,027	0,011
10	0,0017	0,011	0,043

Table II: size of gutters with $\alpha = 45°$

$b\backslash l$	10	25	50
5	0,0003	0,002	0,007
10	0,0011	0,007	0,029

Table III: size of gutters with $\alpha = 60°$

$b\backslash l$	10	25	50
5	0,0002	0,0009	0,0036
10	0,0006	0,0036	0,0143

Looking at these results we see that the values that occur remain small, even when the roof is big (i.e. 10 $\times$ 50 m). Assuming that a normal gutter will

have a height of about 15 cm, it is not nessecary for the producer to use tables to recommend a gutter.
However the question becomes important when the roof gets bigger, for example a roof of a large factory hall or when the rainpipe gets choked. In these cases one might need a gutter with a height of more than 15 cm. But then it will be better to reformulate the question in: "How many rain-pipes does one need, given a certain (very big) roof".
To see how this affects the results we have to look at the model. So in the next paragraph, we will derive the model.

3 Mathematical modelling

The question is of how large a gutter should be. So we have to find a minimum size of the gutter so that it will not overflow by a heavy shower. When the gutter is totally filled this means:
the amount of water coming into the gutter must be equal to the amount of water going out of the gutter.
We use the following notation:

- v : speed of the water at the beginning of the rainpipe [m/sec]
- g : gravitational acceleration [m/sec^2]
- d : diameter of the rainpipe [m]
- l : length of the roof [m]
- b : width of the roof [m]
- Φ_u : volume of the water going out of the gutter [m/sec^3]
- Φ_i : volume of the water coming into the gutter [m/sec^3]
- α : angle of the roof [rad]
- h : height of the gutter [m]
- N : the amount of rainfall [m/sec]

One easily sees that:

$$\Phi_i = blcos(\alpha)N$$

To derive Φ_u we consider the speed of the water at the beginning of the rainpipe. There holds:

$$v = \sqrt{2\,h\,g}$$

(This follows from: kinetic energy equals potential energy$\Longrightarrow \frac{mv^2}{2} = mgh$)
So Φ_u equals:

$$\Phi_u = Sv = \frac{\pi}{4} d^2 \sqrt{2\,h\,g}$$

where S is the surface of the rainpipe with diameter d.
Now we put $\Phi_i = \Phi_u$ which gives us the following equation for the minimal height of the gutter:

$$h = \frac{8 b^2 l^2 cos^2(\alpha) N^2}{\pi^2 d^4 g}$$

When we look at this formula we see that the relation between h and l is quadratic. This means a roof that is twice as small needs a gutter which is four times as small.
So when the formula says we need a gutter of 60 cm using one rain-pipe, we can also use a gutter of 15 cm and two rain-pipes.

ECOLOGICALLY JUSTIFIABLE INSECT PLAGUES PREVENTION

Contents

1 Description of the problem

In this report we consider a problem of a gardener who is growing vegetables in a glasshouse. During the growing cycle of the vegetables the plants suffer from insects which are eating their leaves. Since these insects (we will call them 'wammels') have a negative influence on the profit of the gardener, he wants to find a remedy for keeping the number of wammels as small as possible.
In this report we shall answer the following questions:

- How can we prevent an insect plague ?
- How can we keep the number of wammels (on average) as small as possible during the growing time ?
- How can we achieve that a very small number of wammels is still on the vegetables when they are being sold ?

It is clear that an insect plague is the first thing to avoid, since it ruins his vegetables.
The second question is due to the fact, that the vegetables must grow, without being affected by the insects. Finally the gardener prefers clean vegetables, when they are being sold.

We shall discuss two possibilities for answering these questions:

- The gardener can spray the vegetables with an insecticide. In this case we assume that only 1 % of the wammels stay alive after spraying.
- He can add other species of insects to the glasshouse environment, where he can choose out of two possible insect species, so called 'wimmels' and so called 'wummels'. These insects display the following behaviour

- wimmels(w_2) only eat wammels(w_1)
- wummels(w_3) eat wammels(w_1) as well as wimmels(w_2), but they strongly prefer wimmels.

We have modelled all the above mentioned possibilities. The results are given in the next section, while the mathematical model is discussed in section 3 and 4.

2 Results

We analysed the described possibilities to exterminate the wammels:

- The gardener can spray the vegetables with an insecticide. By assuming that only 1 % of the wammels stays alive, the wammels-population will recover in 7 days if the population has a doubling-time of one day. So keeping the amount of wammels small means spraying once a week. It is clear that this method is not ideal. Therefore we shall concentrate on biological remedies for the wammel-plague, in that way avoiding the use of environment damaging insecticides.

- By adding other species of insects to the population the gardener can control the number of wammels during the growing-cycle. He only has to know at what time and how many insects of each specie he has to add. In the tables, showed below, we give some numerical results. In these tables we only consider adding wimmmels, because in our opinion it is the best way to fight the wammels. The only assumption we make is, that the gardener has a rough estimate of the existing number of wammels in the glasshouse.
  Having this estimate, he can use Table I to determinate how many wimmels he should add to reach in a certain amount of time a situation in which the amount of wammels in the glasshouse is as small as possible.
  If the gardener has determinated, with the help of the first table, what amount of wimmels he is going to add, he can use Table II to get the time at which the amount of wammels becomes the same again as at the time at which he added wimmels, knowing that by adding wimmels the number of wammels in first instance decreases.
  Table III gives the gardener information about the growing cycle of the insect-populations. If he knows the amount of wammels and wimmels in the glasshouse, he knows how many days it takes to be in exact the same situation again.
  Figure 2.1 shows the periodic behaviour of the populations. (see paragraph 4.3)

- By adding wimmels and wummels, he has to be careful. It can either happen that all tree populations explode and keep growing (this is certainly not what he wants), or, in the other case, the wummels die out, but the wammels and wimmels form a biological balance as above.
  By doing experiments, he may find situations in which it can be lucrative adding wummels too.

Remark: The numbers in the tables are all divided by 1000 .

Table I

Time (in days) to reach minimal w_1

$w_2 \backslash w_1$	10	20	30	40	50
6	0.80	0.91	0.98	1.05	1.39
5	0.74	0.84	0.93	1.03	1.21
4	0.62	0.75	0.87	1.02	1.15
3	0.56	0.68	0.84	1.02	1.13
2	0.41	0.56	0.83	1.10	1.11

Table II

Time (in days) to reach same value of w_1

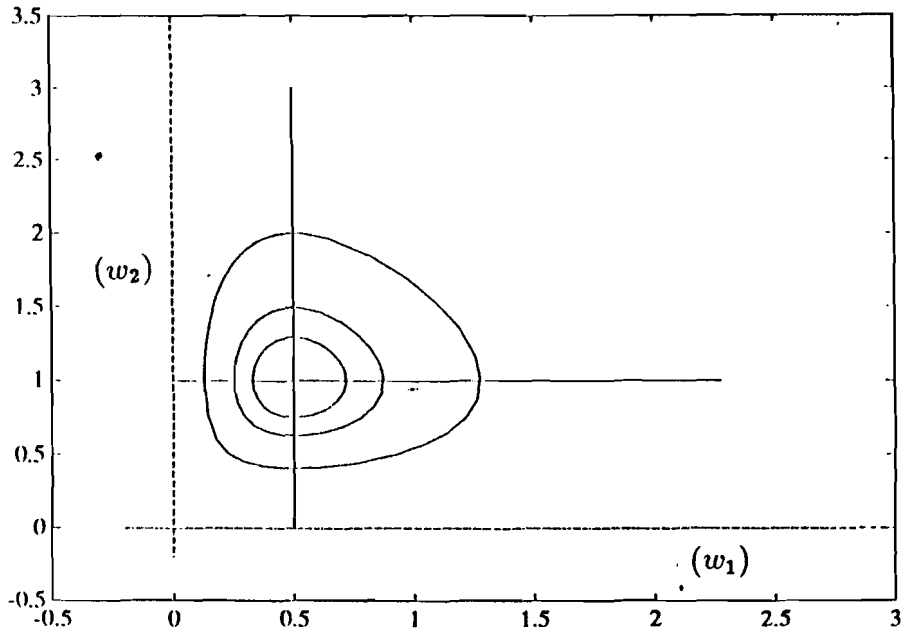
$w_2 \backslash w_1$	10	20	30	40	50
6	2.57	2.99	3.39	3.93	4.26
5	2.14	2.79	3.11	3.66	3.99
4	1.83	2.19	2.75	3.38	3.74
3	1.30	1.82	2.28	2.97	3.56
2	0.85	1.16	1.79	2.64	3.55

Table III

Period of curves (in days)

$w_2 \backslash w_1$	10	20	30	40	50
6	7.41	6.53	6.20	6.13	6.18
5	7.03	6.14	5.78	5.73	5.81
4	6.65	5.71	5.40	5.32	5.43
3	6.28	5.43	5.06	4.99	5.09
2	6.00	5.06	4.80	4.74	4.81

figure(2.1)



3 The Model

In the model described here we use the following notations:

wammels : w_1
wimmels : w_2
wummels : w_3

The model is based on the following assumptions and simplifications:

1. The w_1 eat only vegetables and there is always enough food (vegetables) for them.

2. The w_3 only eat w_2. In reality w_3 can eat w_1, but they prefer w_2 to w_1. So, if there are enough w_2, this assumption is justifiable.

3. Without the presence of w_2 and w_3, the growth of w_1 in a certain time period Δt is proportional to the existing number of w_1 in that (very small) period of time, so w_1 will grow exponentially. This is a biologicallly justifiable assumption.

4. Every single insect w_2 eats a fraction $(\beta w_1)\Delta t$ of the total amount of w_1 during a time period Δt of one day. Analogous remarks can be made about w_3 eating w_2.

Using the above assumptions we can derive the following balance- equations:

$$w_1(t+\Delta t) = w_1(t) + \Delta t(\alpha w_1(t) - \beta w_1(t) w_2(t))$$
$$w_2(t+\Delta t) = w_2(t) + \Delta t(-\kappa w_2(t) + \lambda w_1(t) w_2(t) - \mu w_2(t) w_3(t))$$
$$w_3(t+\Delta t) = w_3(t) + \Delta t(-\eta w_3(t) + \theta w_2(t) w_3(t))$$

All the parameters in these equations are positive. The dimension of t, [t], is T (in days). The dimension of the parameters α,....,θ is T^{-1} (in per day).

Taking $w_1(t)$, $w_2(t)$, and $w_3(t)$, to the left hand side in these balance equations, dividing all terms by Δt and further taking the limit for $\Delta t \to 0$, we get the following differential equations:

$$\dot{w_1}(t) = \alpha w_1(t) - \beta w_1(t) w_2(t)$$
$$\dot{w_2}(t) = -\kappa w_2(t) + \lambda w_1(t) w_2(t) - \mu w_2(t) w_3(t)$$
$$\dot{w_3}(t) = -\eta w_3(t) + \theta w_2(t) w_3(t)$$

The terms for example in the first equation can be explained as follows:

- $\dot{w_1} = \alpha w_1$
  This term causes an exponential grow of the w_1.
- $\dot{w_1} = -\beta w_1 w_2$
  This term tells us about the amount of w_2 eaten by w_1. For further comment see item 4 listed above.

The terms in the other equations can be explained in the same way.

4 Analysing the model

4.1 The one-dimensional case

First we analyse the situation in which $w_2 = w_3 = 0$. So the only equation left, is

$\dot{w}_1 = \alpha w_1$

and its solution is given by

$w_1(t) = w_1(0)e^{\alpha t}$.

It is important to know how fast the w_1 is growing. One could for example measure the time T in which the amount of w_1 is doubled. We call T the 'doubling-time'. The value of T can be derived from the following equation:

$2w_1(0) = w_1(0)e^{\alpha T}$.

So $T = \frac{ln(2)}{\alpha}$

For small insects it can be realistic to take $T = 1$, meaning that the insect population doubles itself in one day. And so we get a value for α:
$\alpha = ln(2)$ per day.

From this estimation for α we conclude that if the gardener chooses to spray with an insecticide, instead of using a biological prevention-method, he has to spray at least once a week, otherwise the population of w_1 will still grow. For we did assume that spraying means killing 99 % of the insect-population, so after spraying, 1 % of the w_1 population is left, meaning that in 7 days the population doubles 7 times. And because $2^7 > 100$ it follows that a week after spraying the insect population will be greater than before.

4.2 Scaling the model

The next step will be the scaling of the parameters in the three-dimensional system of differential equations. Scaling is done in order to get a better comparison for the parameters in the model. We could get rid of 4 parameters, but instead, we choose a scaling in the following form:

$t' = \alpha t, \qquad$ with $\alpha = ln(2)$

$z_1 = \frac{\lambda}{\beta} w_1,\ z_2 = w_2,\ z_3 = \frac{\mu}{\theta} w_3.$

Which results in:

$$\dot{z_1} = z_1 - bz_1z_2$$
$$\dot{z_2} = -az_2 + bz_1z_2 - cz_2z_3$$
$$\dot{z_3} = -dz_3 + cz_2z_3,$$

with $a = \frac{\kappa}{\alpha}$, $b = \frac{\beta}{\alpha}$, $c = \frac{\theta}{\alpha}$, $d = \frac{\eta}{\alpha}$.

This system gives us a good impression of the original system provided that $\frac{\lambda}{\beta}, \frac{\mu}{\theta}, a, b, c, d$ are of O(1). (i.e. all events are of the same order)

4.3 The two-dimensional case

In this section we analyse the system in which only w_2 is added, so $w_3 = 0$. Using the scaled differential equations from section 4.2, we get:

$\dot{z_1} = z_1 - bz_1z_2$
$\dot{z_2} = -az_2 + bz_1z_2$

The matching equilibrium-points are:

(0,0) with eigenvalues $\lambda_1 = 1$ and $\lambda_2 = -a$,

$(\frac{a}{b}, \frac{1}{b})$ with eigenvalues $\lambda_1 = i\sqrt{a}$ and $\lambda_2 = -i\sqrt{a}$.

So (0,0) is an instable point. About the point $(\frac{a}{b}, \frac{1}{b})$ we can't say yet if it is stable or unstable. In the appendix we shall prove that $(\frac{a}{b}, \frac{1}{b})$ is a stable centerpoint. (Linearily it is stable, but this is not enough)

The integral-curves are easy to find. They satisfy

$$\frac{dz_2}{dz_1} = \frac{z_2(-a + bz_2)}{z_1(1 - bz_2)}$$

and so $ln(z_2) + aln(z_1) - b(z_2 + z_1) = \text{constant}$.

For different initial-values, some of these integral-curves are drawn in figure (4.3.1).
In the appendix we prove that these curves are closed.
By looking at the sign of $\frac{dz_2}{dz_1}$, it follows that their direction is counterclockwise.

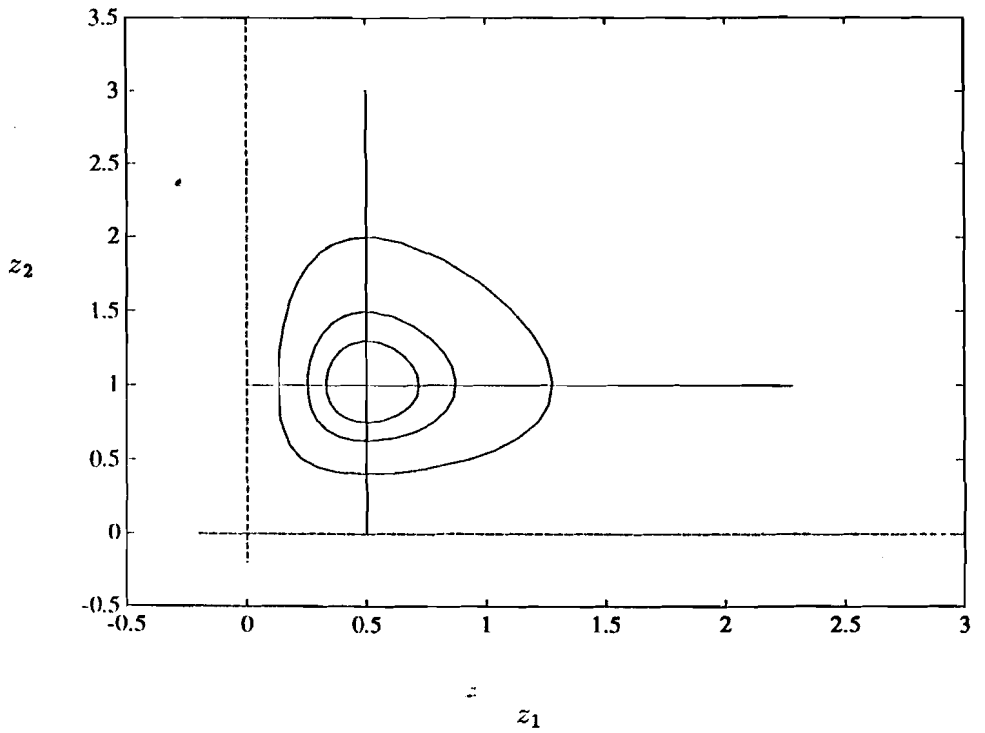


figure (4.3.1) $\qquad ln(z_2) + aln(z_1) - b(z_2 + z_1) = \text{constant.}$

The period of such a curve is difficult to calculate explicitly. In the neighbourhood of the critical point $(\frac{a}{b}, \frac{1}{b})$ we can find a linear approximation.
In the appendix this is done, by using perturbation series for Hamiltonian-systems, up to second order.
Linearizing the system around $(\frac{a}{b}, \frac{1}{b})$ gives the equation
$\dot{z} = Az$ with $A = \begin{bmatrix} 0 & 1 \\ a & 0 \end{bmatrix}$ The eigenvalues of A are $\pm i\sqrt{a}$

So the solution of this equation is

$z(t) = A_1 \cos(\sqrt{a}t) + A_2 \sin(\sqrt{a}t)$,

They have a period: $p' = \frac{2\pi}{\sqrt{a}}$.

So for the original system we obtain an estimation for the period p of the curves

$$p = \frac{1}{\alpha}p' = \frac{2\pi}{\alpha\sqrt{a}} = \frac{2\pi}{\sqrt{\kappa\alpha}}$$

In the appendix it is proved that, in going away from the critical point $(\frac{a}{b}, \frac{1}{b})$, the period p first decreases a bit. From numerical results we can deduce that further away from the critical point the period will grow again.
For example, choose initial points on the line:

$l : \{(z_1, z_2) | z_1 \in (0, \frac{a}{b}), z_2 = \frac{1}{b}\}$,

and $\alpha = \beta = ln(2)$ per day, $\kappa = \frac{ln(2)}{2}$ per day.

From this we can conclude: $p = 12.8$ days.
By taking some initial points we get the following estimations for the period:

$z(0) = (0.4, 1) \rightarrow$ period: 12.60 days
$z(0) = (0.3, 1) \rightarrow$ period: 12.70 days
$z(0) = (0.2, 1) \rightarrow$ period: 12.85 days

In Table III in section 2 there are some more results concerning the period of the curves.

4.4 The three-dimensional case

In this situation we consider the complete system, with w_1, w_2 and w_3. So we have the following (scaled) equations:

$\dot{z_1} = z_1 - bz_1z_2$
$\dot{z_2} = -az_2 + bz_1z_2 - cz_2z_3$
$\dot{z_3} = -dz_3 + cz_2z_3$

The matching equilibrium-points, when $d \neq c$, are

(0,0,0), $(0, \frac{d}{c}, \frac{-a}{c})$ and $(\frac{a}{b}, \frac{1}{b}, 0)$.

And in the special case that $d = c$ there is an extra line of critical points

$l : \{(\frac{a}{b} + \frac{c}{b}\lambda, \frac{1}{b}, \lambda) \in \mathbb{R}^3 | \lambda > 0\}$

The matching eigenvalues are

(0,0,0): $\lambda_1 = 1$, $\lambda_2 = -a$, $\lambda_3 = -d$ $\Rightarrow$ unstable point.

$(0, \frac{d}{c}, \frac{-a}{c})$: $\lambda_1 = 1 - \frac{b-d}{c}$, $\lambda_2 = \sqrt{ad}$, $\lambda_3 = -\sqrt{ad}$ $\Rightarrow$ unstable point.

$(\frac{a}{b}, \frac{1}{b}, 0)$: $\lambda_1 = \frac{c}{b} - d$, $\lambda_2 = i\sqrt{a}$, $\lambda_3 = -i\sqrt{a}$ $\Rightarrow$ if $\frac{c}{b} > d$ then this point is unstable, if $\frac{c}{b} \leq d$ we will prove it to be stable.

If $\frac{c}{b} = d$ the eigenvalues of the line l are:

$\lambda_1 = 0$, $\lambda_2 = i\sqrt{\frac{\lambda c^2}{b} + a + \lambda c}$, $\lambda_3 = -i\sqrt{\frac{\lambda c^2}{b} + a + \lambda c}$.

We shall also prove stability for these points.

A family of solutions we find immediately by comparing the first and the third equation:

$\dot{z_1} = z_1(1 - bz_2)$

$\dot{z_3} = z_3(-d + cz_2) \quad \Rightarrow$

$c\frac{\dot{z_1}}{z_1} + b\frac{\dot{z_3}}{z_3} = c - db \quad \Rightarrow$

$z_3 = \tilde{C} z_1^{-\frac{c}{b}} \exp\left(\frac{(c-db)t}{b}\right)$

In the appendix we prove stability for the point $(\frac{a}{b}, \frac{1}{b}, 0)$, if $\frac{c}{b} \leq d$, using this solution.
It is clear that if $\frac{c}{b} < d$, z_3 tends to zero if z_1 is bounded away from zero.
In the appendix it is also proved that if $c < db$,

$$\lim_{t\to\infty} z_3(t) = 0$$

holds for every initial condition $z(0) = (z_1(0), z_2(0), z_3(0))$,
and the curve converges to a closed curve in the z_1, z_2 plane, given by the equation:

$ln(z_2) + aln(z_1) - b(z_2 + z_1) =$ constant. (see figure (4.4.1)

If $c = db$ then all solutions are periodic, given by:

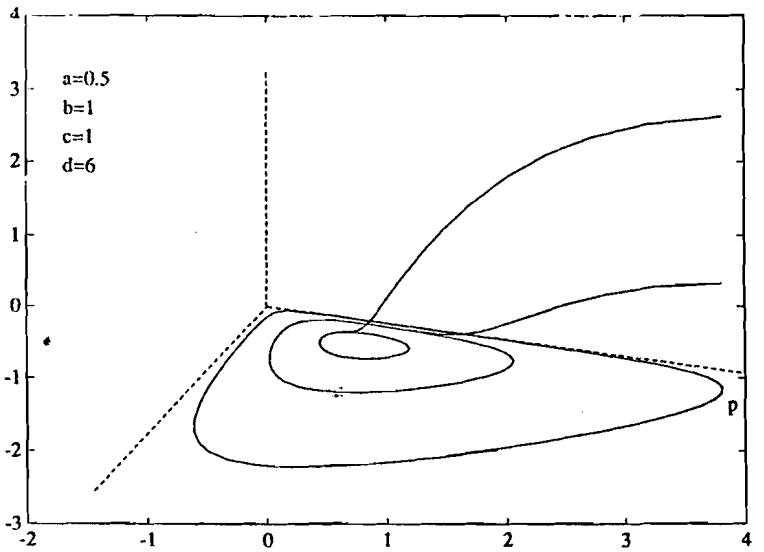
$ln(z_2) + aln(z_1) - b(z_2 + z_1) - \frac{c}{d} z_3 =$ constant $\quad \wedge \quad z_3 = \tilde{C} z_1^{-\frac{c}{b}}$

see figure (4.4.2)

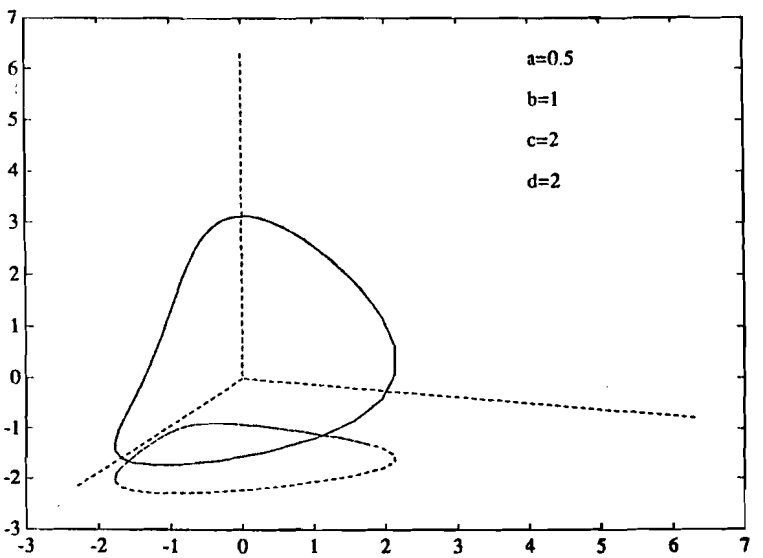
If $c > db$ then

$$\lim_{t\to\infty} z_3(t) = \infty$$

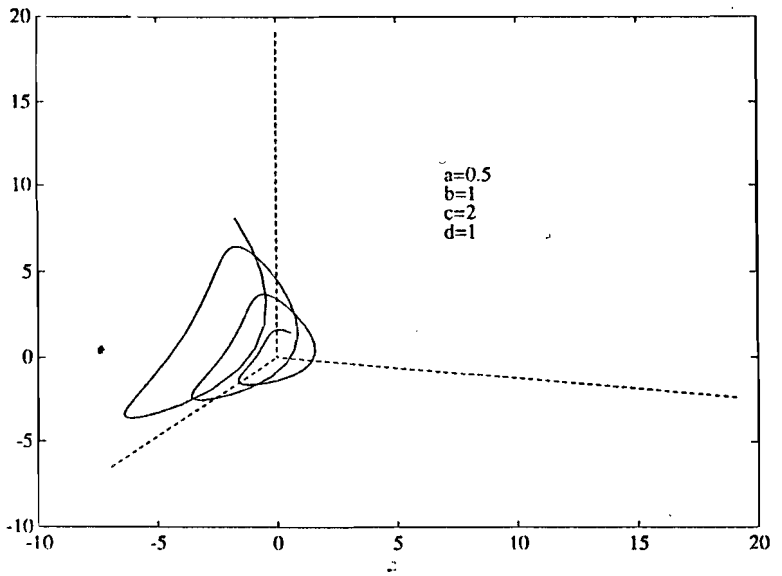
and we see in figure (4.4.3) that also z_1 and z_2 tend to infinity.
In all tree the pictures the ground plane is the z_1, z_2 plane.



(figure (4.4.1) : $c < db$)



(figure (4.4.2) : $c = db$)



(figure (4.4.3) : $c > db$)

5 Conclusions

- To prevent a wammels(w_1) plague, adding only a large enough quantity of wimmels(w_1) is sufficient.

- To minimize the number of wammels(w_1), we have to follow a strategy of adding wimmels(w_2) at particular times, using the given tables.

- With a growing time taking several weeks, the adding of wummels(w_3) gives (on average) a smaller number of wammels(w_1) and wimmels(w_2) during this period.

- It is not wise to minimize the number of wammels(w_1) by adding wummels(w_3) because, it is difficult to calculate the right number to add, and making a small mistake can cause disastrous effects.

6 Appendix

We have a system differential equations in the following form:

$\dot{z_1} = z_1 - bz_1z_2$
$\dot{z_2} = -az_2 + bz_1z_2 - cz_2z_3$
$\dot{z_3} = -dz_3 + cz_2z_3$

By introducing new variables:

$x := \ln z_1, \qquad y := \ln z_2, \qquad z := \ln z_3$

We transform the system in the new variables:

$\dot{x} = 1 - be^y$
$\dot{y} = -a + be^x - ce^z$
$\dot{z} = -d + ce^y$

One solution is found directly by combining the first and third equation
$c\dot{x} + b\dot{z} = c - bd \quad \Rightarrow$

$e^z = A\exp\left((\frac{-cx}{b}) + (\frac{c-bd}{b})\ t\right)$, for certain constant A > 0.

Substituting this equation for e^z in the second differential equation gives:

$\dot{x} = 1 - b\exp(y)$
$\dot{y} = -a + b\exp(x) - cA\exp\left(\frac{-cx}{b} + (\frac{c-bd}{b})\ t\right)$

with A > 0 arbitrarily.

This is a time dependent Hamilton system, and the Hamiltonian is given by:

$$H = b(\exp(y) + \exp(x)) - y - ax + bA\exp\left(\frac{-cx}{b} + \left(\frac{c-bd}{b}\right)\ t\right)$$

where A $>$ 0 follows from the initial values.

such that $\dot{x} = -\frac{dH}{dy}$ and $\dot{y} = \frac{dH}{dx}$

Theorem 6.1

If $c - bd < 0$ then

$$\lim_{t\to\infty} z(t) = -\infty$$

and the Hamiltonian converges to the time-independent Hamiltonian for the system with the original $w_3 = 0$:

$b(e^y + e^x) - y - ax =$ constant.

If $c - bd > 0$ then

$$\lim_{t\to\infty} z(t) = \infty$$

If $c - bd = 0$ then all solutions are periodic, the projection in the $x - y$ plane of these curves are given by the integrals:

$b(e^y + e^x) - y - ax + bA\exp\left(\frac{-cx}{b}\right)=$ constant.

Proof

First we consider the case $c - bd < 0$.

Def. $I(x,y) := b(e^y + e^x) - y - ax \quad \Rightarrow$

$H = I + be^z$

We shall prove that

$$\lim_{t\to\infty} z(t) = -\infty$$

, with the consequence that

$$\lim_{t\to\infty} H(x,y,t) = I(x,y)$$

and that $I(x,y)$ is the Hamiltonian of the system with $z = 0$:

$\dot{x} = 1 - be^y$
$\dot{y} = -a + be^x$

Suppose

$$\lim_{t\to\infty} z(t) \neq -\infty$$

then since

$$\lim_{t\to\infty} \exp(\frac{-cx}{b} + (\frac{c-bd}{b})) \neq 0$$

follows necessary

$$\lim_{t\to\infty} x(t) \neq -\infty$$

In this case

$\exists T_1$ such that $\forall t > T_1 \quad : e^{x(t)} < \epsilon, \quad 0 < \epsilon \ll 1.$

But then with $\dot{y} = e^x - a - e^z \quad \Rightarrow \quad \dot{y} < \epsilon - a \quad \Rightarrow$

$y(t) < (\epsilon - a)t + c_1, \quad \forall t > T_1.$

But that would significate $y(t) \to -\infty$, however with $\dot{x} = 1 - e^y$ this would significate $\exists T_2$ s.t. $1 - e^y > 0, \quad \forall t > T_2$.

But this is a contradiction with the assumption, so

$$\lim_{t\to\infty} z(t) = -\infty$$

and

$$\lim_{t\to\infty} H(x, y, t) = I(x, y)$$

But $I(x, y)$ is the Hamiltonian of the 2-dimensional system:

$$\dot{x} = 1 - b\exp(y)$$
$$\dot{y} = -a + b\exp(x) - cA\exp\left(\frac{-cx}{b} + \left(\frac{c - bd}{b}\right)\ t\right)$$

Since $\dot{x} = -\dfrac{dI}{dy}$ and $\dot{y} = \dfrac{dI}{dx}$

Since $z(t) \to -\infty$ is equivalent with $w_3(t) \to 0$ in the original system, our theorem is confirmed by picture (4.4.1).

if $c - bd > 0$ then an analogous story can be hold, and it follows that $z(t) \to \infty$ and equivalent $w_3(t) \to \infty$, the curves diverge.

If $c - bd = 0$ then the Hamiltonian becomes time-independent, the system holds two degrees of freedom, and is integrable.
All solutions are periodic, given by:

$$b(e^y + e^x) - y - ax + bA\exp\left(\frac{-cx}{b}\right) = \text{constant.} \qquad \square$$

Now we look at the case $w_3 = 0$ and keep the following equations:

$$\dot{x} = 1 - be^y$$

$\dot{y} = -a + be^x$

$H = b(e^y + e^x) - y - ax$

Since eigenvalues of the unique critical point $(ln(\frac{a}{b}), ln(\frac{1}{b}))$ are purely complex, and the system is a Hamilton system, we can conclude that the critical point is a center point, and that all curves in the (x,y) plane are closed.

We look for an approximation for the period T close to the critical point

Define $\quad p := x - ln(\frac{a}{b}), \quad q := y - ln(\frac{1}{b})$

Substituted in the Hamiltonian gives:

$H = (e^q + ae^p) - q - ap -$ constant,

where the constant $(= ln(\frac{1}{b}) + aln(\frac{a}{b}))$ we can omit.

Now we develop H in a taylor series around $(p = 0, q = 0)$:

$H = (\frac{1}{2}q^2 + \frac{1}{6}q^3 + ...) + a(\frac{1}{2}p^2 + \frac{1}{6}p^3 + ...) + ...$

We scale in the neighbourhood of the critical point

$p = \epsilon\bar{p}, \quad q = \epsilon\bar{q}, \quad 0 < \epsilon \ll 1$

$\bar{H} := \frac{1}{2}(a\bar{p}^2 + \bar{q}^2) + \frac{1}{6}\epsilon(a\bar{p}^3 + \bar{q}^3) + \frac{1}{24}\epsilon^2(a\bar{p}^4 + \bar{q}^4) + ...$

The following step is introducing Action-Angle variables, defined by :

$\bar{p} := \sqrt{\frac{2j}{\sqrt{a}}} \sin\varphi$

$\bar{q} := -\sqrt{2j\sqrt{a}} \cos\varphi$

It's easy to prove that by this canonical change of variables the system re-

mains Hamiltonian, so by substituting φ, j for $\bar{p}, \bar{q}$ in H holds:

$$\dot{\varphi} = \frac{dH(\varphi, j)}{dj} \qquad \text{and} \qquad \dot{j} = -\frac{dH(\varphi, j)}{d\varphi}$$

By introducing a generating function $S(J, \varphi)$, transforming $(j, \varphi) \rightarrow (J, \phi)$ and applying Hamilton-Jacobi theorem , we can eliminate φ one order of ϵ . Doing this N times, you can get:

$$\dot{\varphi} = \frac{dH(\varphi, j)}{dj} = \omega(j) + O(\epsilon^N) \qquad \Rightarrow$$

$$\varphi(t) = (\omega(j)t + \varphi(0) + O(\epsilon^N)) \pmod{2\,\pi} \qquad \Rightarrow$$

$$T = \frac{2\pi}{\omega(j)} + O(\epsilon^N)$$

We will calculate the perturbation serie till N =3.

Substituting j and φ in H gives:

$$\bar{H} := \tfrac{1}{2}(a\bar{p}^2 + \bar{q}^2) + \tfrac{1}{6}\epsilon(a\bar{p}^3 + \bar{q}^3) + \tfrac{1}{24}\epsilon^2(a\bar{p}^4 + \bar{q}^4) + \ldots =$$

$$\sqrt{a}j + \epsilon\,(\tfrac{\frac{1}{3}\sqrt{2}}{a^{\frac{3}{4}}} j\sqrt{j}\cos^3\varphi - \tfrac{1}{3}\sqrt{2}a^{\frac{3}{4}}j\sqrt{j}\,\sin^3\varphi) + \epsilon^2\,(\tfrac{1}{6}j^2\cos^4\varphi + \tfrac{1}{6}aj^2\sin^4\varphi) +$$

$$O(\epsilon^3)$$

Now we define a canonical variable transformation $(j, \varphi) \rightarrow (J, \phi)$ by introducing a generating function

Def. $\qquad S(J, \varphi) = J\varphi + \epsilon\, S_1(J, \varphi) + \epsilon^2\, S_2(J, \varphi) + O(\epsilon^3)$

We choose S such that:

$$j = \frac{dS(J, \varphi)}{d\varphi} \qquad \text{and} \qquad \phi = \frac{dS(J, \varphi)}{dJ}$$

It can be proved again that $H(J, \phi)$ remains Hamiltonian.

For notation we define:

$V_1(j,\varphi) := \frac{1}{3}\sqrt{2}a^{\frac{3}{4}}j\sqrt{j}\cos^3\varphi - \frac{1}{3}\sqrt{2}a^{\frac{3}{4}}j\sqrt{j}\ \sin^3\varphi$

$V_2(j,\varphi) := \frac{1}{6}j^2\cos^4\varphi + \frac{1}{6}aj^2\sin^4\varphi$

Substituting $j = \dfrac{dS}{d\varphi}$ in H and setting the Hamilton-Jacobi equation gives:

$H = \sqrt{a}\dfrac{dS}{d\varphi} + \epsilon V_1(\dfrac{dS}{d\varphi},\varphi) + \epsilon^2 V_2(\dfrac{dS}{d\varphi}) =$

$K_0(J) + \epsilon K_1(J) + \epsilon^2 K_2(J) + \epsilon^3 K_3(J,\phi) \qquad \Rightarrow$

$H = \sqrt{a}\ (J + \epsilon\dfrac{dS_1(J,\varphi)}{\varphi} + \epsilon^2\dfrac{dS_2(J,\varphi)}{\varphi} + ...) + \epsilon V_1\ (J + \epsilon\dfrac{dS_1(J,\varphi)}{\varphi} + ..., \varphi) +$

$\epsilon^2 V_2(J + \epsilon\dfrac{dS_1(J,\varphi)}{\varphi} + ..., \varphi) + ... =$

$K_0(J) + \epsilon K_1(J) + \epsilon^2 K_2(J) + \epsilon^3 K_3(J,\phi)$

For comparing powers of ϵ, V_1 and V_2 have to be developed in Taylor series round J .
We want to eliminate φ till $O(\epsilon^2)$, so we have to resolve:

$O(\epsilon^0)$: $\quad K_0(J) = \sqrt{a}J$

$O(\epsilon^1)$: $\quad \sqrt{a}\dfrac{dS_1(J,\varphi)}{d\varphi} + V_1(J,\varphi) = K_1(J)$

$O(\epsilon^2)$: $\quad \sqrt{a}\dfrac{dS_2(J,\varphi)}{d\varphi} + \dfrac{dV_1(J,\varphi)}{dj}S_1(J,\varphi) + V_2(J,\varphi) = K_2(J)$

These are the equations to solve, since for $O(\epsilon^0)$ φ does not appear, we start for $O(\epsilon^1)$.

$O(\epsilon^1)$: $\sqrt{a}\dfrac{dS_1(J,\varphi)}{d\varphi} + V_1(J,\varphi) = K_1(J) \qquad \Rightarrow$

$$\sqrt{a}\frac{dS_1(J,\varphi)}{d\varphi} + \frac{1}{3}\sqrt{2}a^{\frac{3}{4}}j\sqrt{j}\cos^3\varphi - \frac{1}{3}\sqrt{2}a^{\frac{3}{4}}j\sqrt{j}\sin^3\varphi = K_1(J)$$

We choose S_1 such that φ becomes eliminated.
This can be done by developing S_1 in his Fourier-serie, the details are omitted, but we arrive at:

$$K_1(J) = 0 \qquad \wedge$$

$$\sqrt{a}\,\frac{dS_1(J,\varphi)}{d\varphi} = -\frac{1}{12}\sqrt{2}a^{\frac{3}{4}}J\sqrt{J}(3\cos\varphi + \cos 3\varphi - 3a\sqrt{a}\sin\varphi + a\sqrt{a}\sin 3\varphi)$$

$$\Rightarrow$$

$$K_1(J) = 0 \qquad \wedge$$

$$S_1(J,\varphi) = -\tfrac{1}{12}\sqrt{2}a^{\frac{5}{4}}J\sqrt{J}\ (3\sin\varphi + \tfrac{1}{3}\sin 3\varphi + 3a\sqrt{a}\cos\varphi - \tfrac{1}{3}a\sqrt{a}\cos 3\varphi)$$

Since $K_1(J) = 0$ it is necessary to look at higher order terms:

$$O(\epsilon^2): \quad \sqrt{a}\frac{dS_2(J,\varphi)}{d\varphi} + \frac{dV_1(J,\varphi)}{dj}S_1(J,\varphi) + V_2(J,\varphi) = K_2(J) \qquad \text{, where}$$

$$V_1(J,\varphi) := \tfrac{1}{3}\sqrt{2}a^{\frac{3}{4}}J\sqrt{J}\cos^3\varphi - \tfrac{1}{3}\sqrt{2}a^{\frac{3}{4}}J\sqrt{J}\ \sin^3\varphi \qquad \Rightarrow$$

$$\frac{dV_1}{dJ} = \frac{1}{2}\sqrt{2}a^{\frac{3}{4}}\sqrt{J}\cos^3\varphi - \frac{1}{2}\sqrt{2}a^{\frac{3}{4}}\sqrt{J}\ \sin^3\varphi$$

We derive that:

$$S_1\frac{dV_1}{dJ} = -\frac{1}{12}\sqrt{2}a^{\frac{5}{4}}J\sqrt{J}\ (3\sin\varphi + \frac{1}{3}\sin 3\varphi + 3a\sqrt{a}\cos\varphi - \frac{1}{3}a\sqrt{a}\cos 3\varphi).$$

$$(\tfrac{1}{2}\sqrt{2}a^{\frac{3}{4}}\sqrt{J}\cos^3\varphi - \tfrac{1}{2}\sqrt{2}a^{\frac{3}{4}}\sqrt{J}\ \sin^3\varphi) =$$

$$\frac{1}{48a^2}J^2\ (3\sin\varphi + \frac{1}{3}\sin 3\varphi + 3a\sqrt{a}\cos\varphi - \frac{1}{3}a\sqrt{a}\cos 3\varphi)$$

$(3\cos\varphi - \cos 3\varphi + 3a\sqrt{a}\sin\varphi - a\sqrt{a}\sin 3\varphi)$ ... (*)

This expression is used in the equation we want to solve:

$$\sqrt{a}\frac{dS_2(J,\varphi)}{d\varphi} + \frac{dV_1(J,\varphi)}{dj}S_1(J,\varphi) + V_2(J,\varphi) = K_2(J)$$

Again we develop S_2 in it's fourier series, and derive:

$K_2(J) = \frac{1}{16}J^2(a+1)$

We remark that it is also possible calculating $K_2(J)$ directly, by using ...(*) and formulas like

$\cos^4\varphi = \frac{3}{8} + \frac{1}{2}\cos 2\varphi + \frac{1}{8}\cos 4\varphi$

$\sin^4\varphi = \frac{3}{8} - \frac{1}{2}\cos 2\varphi + \frac{1}{8}\cos 4\varphi,$

and leave out all the φ-dependent terms.

For the Hamiltonian, we have derived:

$H = \sqrt{a}J + \frac{1}{16}\epsilon^2 J^2(1+a) + O(\epsilon^3)$ and

$$\dot{\varphi} = \frac{dH}{dJ} = \sqrt{a} + \frac{1}{8}\epsilon^2 J(1+a) + O(\epsilon^3) \quad \Rightarrow$$

$\varphi = (\sqrt{a} + \frac{1}{8}\epsilon^2 J(1+a))t + O(\epsilon^3) =$

$\omega(J)t + O(\epsilon^3)$

For the period T we get the following approximation :

$$T = \frac{2\pi}{\omega(J)} + O(\epsilon^3) =$$

$$\frac{2\pi}{\sqrt{a} + \frac{1}{8}J\epsilon^2(1+a)} + O(\epsilon^3) =$$

$$\frac{2\pi}{\sqrt{a}} - \frac{1}{4}\pi J\epsilon^2(1+\frac{1}{a}) + O(\epsilon^3)$$

The conclusion is that close to the critical point , the period T decreases, if The radius J becomes bigger.

SIMULATION OF NEEDLING PROCESSES ON SURFACES

Contents

1 Description of the problem

In many chemical processes crystals can appear on the surfaces of substances, which cool down. These phenomenons have the following properties:

- A crystal has the form of a needle: They appear in a point (kernel) at a certain time and grow in opposite directions across a straight line. The position of such a kernel as well as the tangent of the corresponding line and the initial growing time are all random. (figure 2.1 page 4)
- The speed, under which a crystal grows, appears to be constant.
- An crystal-end stops growing, when it hits an other crystal.

We made a computer program Kristal2.pas (see section 5) , which simulates the crystal growing proces. In this report both the program and the results are demostrated. Furthermore the following question is answered:

- What can be said about the amount of active crystal ends per square unit at any time of the process? Herefore we also made a computer program, called Kristaleinden.pas given in section 5.

In section 2 are given the numerical results, while the working of Kristal2.pas is set out in section 3. Section 4 is reservated for conclusions and section 5 gives the two computer programs, written in Turbo Pascal.

2 Numerical results

For simulating the crystal growing process, we made a computer program, called Kristal2.pas, written in Turbo-Pascal. (In section 5 the program is given and in section 3 it's working is explained)
When the program is being runned, we get pictures like figure 2.1, which gives a situation of a certain crystal growing process, on a rectangle in $\mathbb{R}^2$.

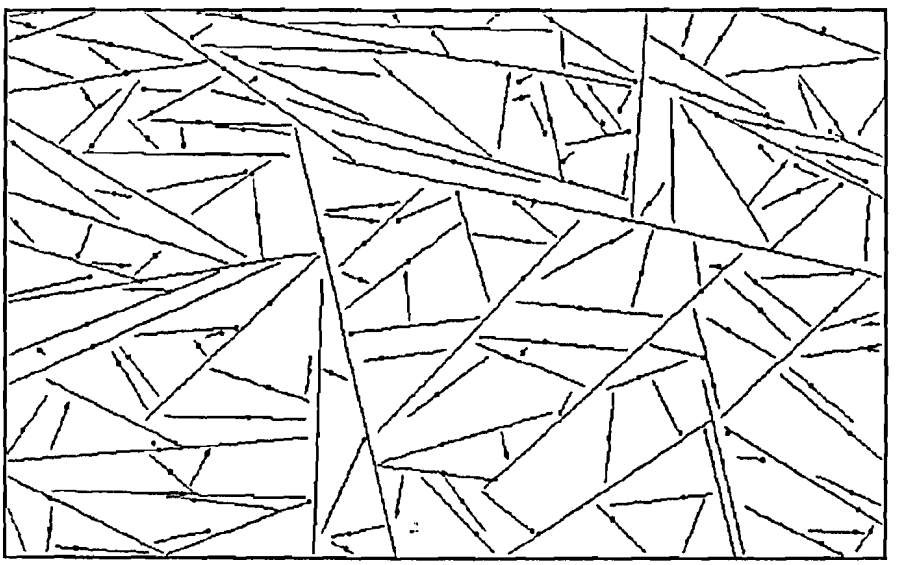


figure 2.1 (random crystal growing)

Since the processes are random, it is not possible to find an analitical solution for the number of active crystal ends at a certain time.
Therefore is made a program called Kristaleinden.pas, which calculates the number of active crystal ends at each time step for the running process. Figure 2.2 gives such a result. On the vertical axis is set out $N(t)$: the number of active ends as a function of time t. On the horizontal axis the time.
By running the program several times (for example 1000 times) and averaging $N(t)$, we get figure 2.3.

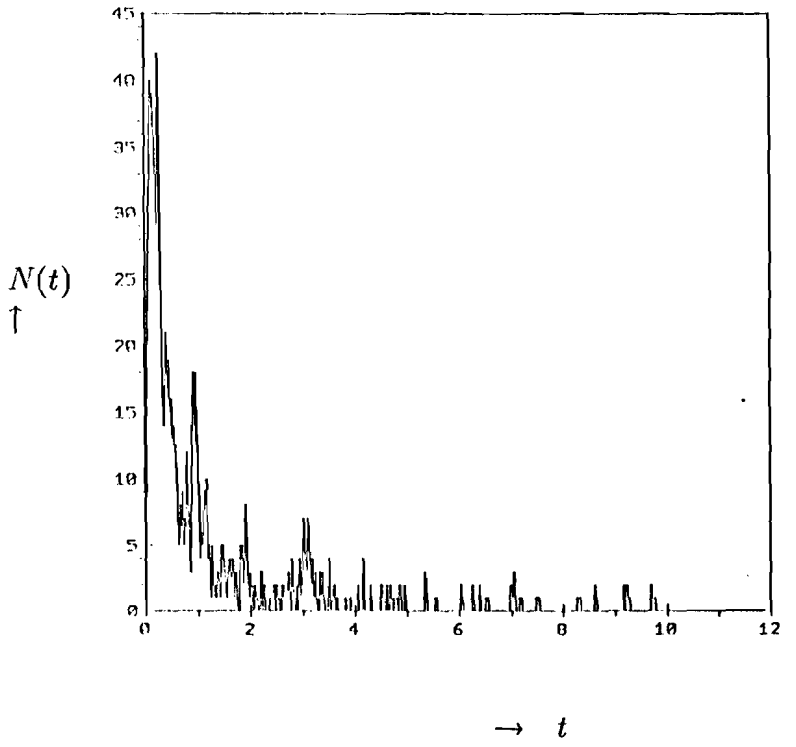


figure 2.2 ($N(t)$ for one proces)

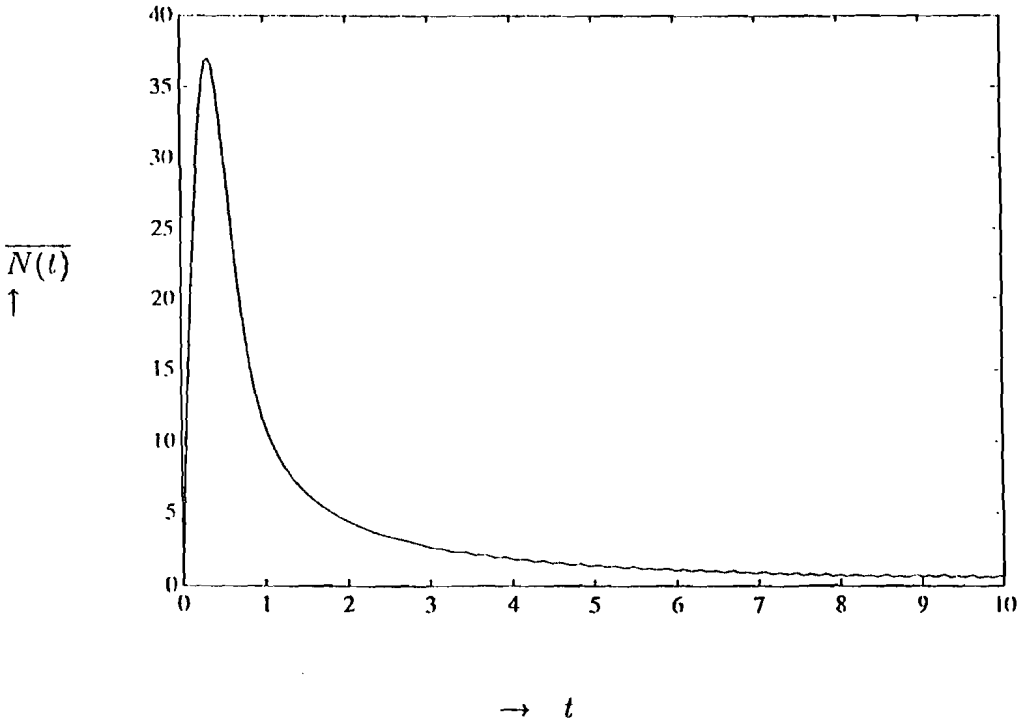


figure 2.3 (The average $N(t)$ of 1000 processes)

3 Simulation method

In this section the working of Kristal2.pas is explained. A proces is simulated on a grid of size M*M in $\mathbb{R}^2$ and is considered as a discrete time process. For a better explanation we introduce the following notation:

t	: Time
$N(t)$	: Number of active ends
$H(t)$	: Number of marked elements (element = sqare of size $\frac{1}{M} * \frac{1}{M}$)
(x_i,y_i)	: Coordinates of the kernels
A_i	: Tangents of corresponding lines
T_i	: Origintimes of the kernels
M	: Grid size
K	: Number of kernels
$MT(i,j)$	: Marking time of grid element (i,j)
$MI(i,j)$	: Index for the kernel which enters element (i,j) first

K (number of kernels), (x_i,y_i) (coordinates of the kernels), A_i (tangents of corresponding lines) and T_i (Origintimes of the kernels) are determinated by a standard Randomizer of Turbo-Pascal, the first three from a uniform distribution and T_i from an exponential distribution with parameter λ. We consider λ as a linear function of t: $\lambda(t) = \lambda_0(1 + t)$
The program acquires from the user:

1) The grid size M (roosteromvang)
2) λ_0 for the exponential distribution (lambda-begin)
3) The end time of the process (eindtijd)

Then the program starts to simulate the process using a discrete time step $\frac{1}{M}$. This indicates that a horizontal and vertical movement in one time step is at most one grid element. (see figure 3.1)

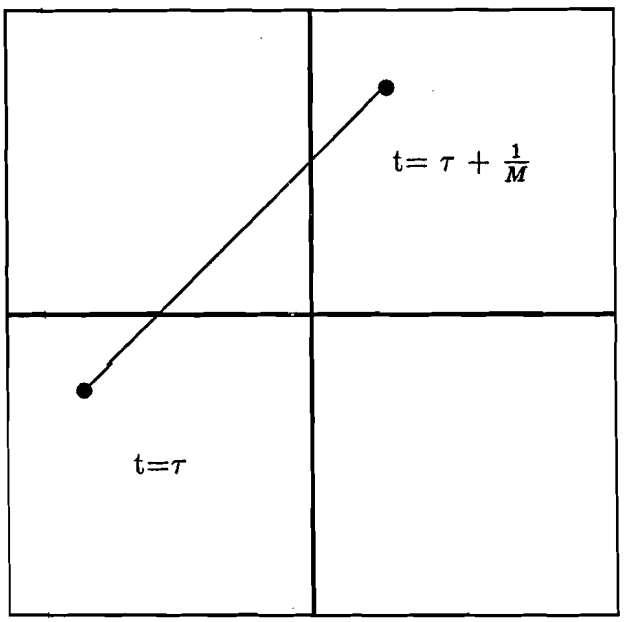


figure 3.1

Once an active end has reached a grid element, two situations can occur:

- The active end is the first end that reaches the element. The marking time and the marking index for the element are changed.
- An other crystal was in the element before. the active end stops growing.

In this way Kristal2.pas evaluates for each time step the situation, until the given end-time is reached.

4 Conclusions

We succeeded in simulating a crystal growing process, but could not find an analitical solution for $N(t)$: the number of active ends as a function of time t. Figure 2.3 tells us something about its behaviour on average, however more theoretical research can be done. We suppose that it might be possible to derive a differential equation for $\bar{N}(t)$, using Markov-chains.

5 Computer programs

```
PROGRAM Kristal2;

Uses
 Crt, Graph;

Const Nmax = 1000;
      Mmax = 103;

TYPE  arr = ARRAY [ 1 .. Nmax ] OF real;
      ari = ARRAY [ 1 .. Nmax ] OF integer;
      mat = ARRAY [ 0 .. Mmax , 0 .. Mmax ] OF real;
      matpointer = ^mat;
      mti = ARRAY [ 0 .. Mmax , 0 .. Mmax ] OF integer;
      mtipointer = ^mti;

VAR   i1, i2, j1, j2, k, M, N, code: integer;
      HokjesTeller, EindenTeller, KernenTeller, KernenErbij: integer;
      x, y, hst, vst, T, EindTijd, Lambda_begin: real;
      KernX, KernY, HoekKern: arr;
      Bx, By, 0x, 0y: arr;
      BeginTijdKern, EindBoven, EindOnder: ari;
      MarkeerTijd: matpointer;
      MarkeerIndex: mtipointer;
      uitv1, uitv2: text;
      Gd, Gm, LowMode: integer;
      Color: word;

FUNCTION Lambda ( T: real ): real;
BEGIN
 Lambda := Lambda_begin * T + Lambda_begin;
END;

PROCEDURE Tijdstippen ( VAR BeginTijdKern: ari; EindTijd: real;
                        M: integer; VAR N: integer);
```

```
VAR i: integer;
   x, interval: real;
BEGIN
 x := ( random ( 10000 ) / 10000 );
 WHILE ( x < 1E-30 ) DO x := ( random ( 10000 ) / 10000 );
 interval := - ln ( x ) * Lambda ( 0 );
 BeginTijdKern [ 1 ] := trunc ( M * interval );
 i := 2;
 WHILE ( ( i <= N ) AND ( BeginTijdKern [ i - 1 ] < EindTijd * M ) ) DO
 BEGIN
   x := ( random ( 10000 ) / 10000 );
   WHILE ( x < 1E-30 ) DO x := ( random ( 10000 ) / 10000 );
   interval := - ln ( x ) * Lambda ( BeginTijdKern [ i - 1 ] / M );
   BeginTijdKern [ i ] := BeginTijdKern [ i - 1 ] + trunc ( M * interval );
   i := i + 1
 END;
 N := i - 1;
END;

PROCEDURE Coordinaten ( VAR KernX, KernY: arr; Aantal: integer );
VAR i: integer;
BEGIN
 FOR i := 1 TO Aantal DO
 BEGIN
   KernX [ i ] := ( random ( 10000 ) / 10000 );
   KernY [ i ] := ( random ( 10000 ) / 10000 );
 END
END;

PROCEDURE Hoeken ( VAR HoekKern: arr; Aantal: integer );
CONST pie = 3.1415;
VAR i: integer;
BEGIN
 FOR i := 1 TO Aantal DO HoekKern [ i ] := pie * ( random ( 10000 ) / 10000 )
END;

PROCEDURE Initialisatie ( VAR MarkeerTijd: matpointer;
                          VAR MarkeerIndex: mtipointer;
                          Aantal1, Aantal2: integer;
```

```
                        KernX, KernY: arr; VAR Bx, By, Ox, Oy: arr;
                        VAR EindBoven, EindOnder: ari );
VAR i, j:integer;
BEGIN
 NEW ( MarkeerTijd );
 NEW ( MarkeerIndex );
 FOR i := 0 TO Aantal1 - 1 DO
   FOR j := 0 TO Aantal1 - 1 DO
   BEGIN
     MarkeerTijd^ [ i, j ] := 1000;
     MarkeerIndex^ [ i, j ] := 0
   END;
 FOR i := 1 TO Aantal2 DO
 BEGIN
   Bx [ i ] := KernX [ i ];
   By [ i ] := KernY [ i ];
   Ox [ i ] := KernX [ i ];
   Oy [ i ] := KernY [ i ];
   EindBoven [ i ] := 0;
   EindOnder [ i ] := 0;
 END
END;

FUNCTION Doorgaan ( HokjesTeller, M: integer; T, EindTijd: real ): boolean;
BEGIN
 Doorgaan := ( ( HokjesTeller < ( M * M ) ) AND ( T < EindTijd ) )
END;

FUNCTION RondAf ( Getal: real ): integer;
VAR teken: integer;
BEGIN
 IF ( Getal >= 0 ) THEN teken := 1 ELSE teken := -1;
 RondAf := teken * trunc ( abs ( Getal ) )
END;

PROCEDURE VeranderHokje ( Snijtijd, Tijd: real; i, j, k, Index, N: integer;
                          VAR Waarde, HokjesTeller, EindenTeller: integer;
                          VAR EindBoven, EindOnder: ari; VAR MarkeerTijd: matpoin
                          VAR MarkeerIndex: mtipointer );
```

```
BEGIN
 IF ( Snijtijd < Tijd )
    THEN BEGIN
           IF ( Tijd > 900 )
              THEN HokjesTeller := HokjesTeller + 1
              ELSE BEGIN
                     IF ( Index <= N )
                        THEN BEGIN
                               IF EindBoven [ Index ] = 1
                                  THEN EindenTeller := EindenTeller - 1;
                               EindBoven [ Index ] := 2
                             END
                        ELSE BEGIN
                               IF EindOnder [ Index - N ] = 1
                                  THEN EindenTeller := EindenTeller - 1;
                               EindOnder [ Index - N ] := 2
                             END;
                   END;
           MarkeerIndex^ [ i, j ] := k;
           MarkeerTijd^ [ i, j ] := Snijtijd
         END
    ELSE BEGIN
           Waarde := 2;
           EindenTeller := EindenTeller - 1
         END
END;

FUNCTION NieuweKernen ( Tijd: real; BeginTijdKern: ari;
                        M, N, KernenTeller: integer ): integer;
VAR k, som:integer;
   doorgaan: boolean;
BEGIN
 k := KernenTeller + 1;
 som := 0;
 doorgaan := ( k <= N );
 IF doorgaan THEN doorgaan := ( Tijd >= BeginTijdKern [ k ] / M );
 WHILE doorgaan DO
 BEGIN
   som := som + 1;
```

```
    k := k + 1;
    doorgaan := ( k <= N );
    IF doorgaan THEN doorgaan := ( Tijd >= BeginTijdKern [ k ] / M )
  END;
  NieuweKernen := som
END;


FUNCTION BuitenRooster ( i, j, M: integer ): boolean;
BEGIN
  BuitenRooster := ( ( i < 0 ) OR ( j < 0 ) OR ( i >= M ) OR ( j >= M ) )
END;


BEGIN {Hoofdprogramma}
  ASSIGN ( uitv1, 'Einden.uit' );
  REWRITE ( uitv1 );
  ASSIGN ( uitv2, 'Hokjes.uit' );
  REWRITE ( uitv2 );
  T := 0;
  HokjesTeller := 0;
  KernenTeller := 0;
  EindenTeller := 0;
  N := Nmax;
  writeln ( 'Geef roosteromvang:' );
  readln ( M );
  writeln ( 'Geef beginwaarde van exponentiele verdeling:' );
  readln ( Lambda_begin );
  writeln ( 'Geef eindtijd van proces:' );
  readln ( EindTijd );
  Randomize;
  Tijdstippen ( BeginTijdKern, EindTijd, M, N );
  Coordinaten ( KernX, KernY, N );
  Hoeken ( HoekKern, N );
  Initialisatie ( MarkeerTijd, MarkeerIndex, M, N,
                  KernX, KernY, Bx, By, Ox, Oy, EindBoven, EindOnder );
  Gd := Detect;
  InitGraph ( Gd, Gm, 'C:\BIN\TP60\BGI\');
  IF GraphResult <> 0 THEN Halt ( 1 );
```

```
SetGraphMode ( LowMode);
Color := GetMaxColor;
Rectangle ( 0, 0, GetMaxX, GetMaxY );
WHILE Doorgaan ( HokjesTeller, M, T, EindTijd ) DO
BEGIN
  KernenErbij := NieuweKernen ( T, BeginTijdKern, M, N, KernenTeller );
  FOR k := ( KernenTeller + 1 ) TO ( KernenTeller + KernenErbij ) DO
  BEGIN
    i1 := RondAf ( M * KernX [ k ] );
    j1 := RondAf ( M * KernY [ k ] );
    IF ( MarkeerTijd^ [ i1, j1 ] > 999 )
       THEN BEGIN
              HokjesTeller := HokjesTeller + 1;
              Bar ( trunc ( KernX [ k ] * GetMaxX - 1 ),
                    trunc ( KernY [ k ] * GetMaxY - 1 ),
                    trunc ( KernX [ k ] * GetMaxX + 1 ),
                    trunc ( KernY [ k ] * GetmaxY + 1 ) );
              EindenTeller := EindenTeller + 2;
              EindBoven [ k ] := 1;
              EindOnder [ k ] := 1;
              MarkeerTijd^ [ i1, j1 ] := T;
              MarkeerIndex^ [ i1, j1 ] := k
            END
       ELSE BEGIN
              EindBoven [ k ] := 2;
              EindOnder [ k ] := 2
            END
  END;
  KernenTeller := KernenTeller + KernenErbij;
  FOR k := 1 TO KernenTeller DO
  BEGIN
    IF ( EindBoven [ k ] = 1 ) THEN
    BEGIN
      x := Bx [ k ] + cos ( HoekKern [ k ] ) / M;
      y := By [ k ] + sin ( HoekKern [ k ] ) / M;
      i1 := RondAf ( M * Bx [ k ] );
      j1 := RondAf ( M * By [ k ] );
      i2 := RondAf ( M * x );
      j2 := RondAf ( M * y );
```

```
IF BuitenRooster ( i2, j2, M )
   THEN BEGIN
          EindBoven [ k ] := 2;
          EindenTeller := EindenTeller - 1;
          code := 0
        END
   ELSE code := abs ( i2 - i1 ) + 2 * abs ( j2 - j1 );
CASE code OF
  1: BEGIN
       hst := T + abs ( ( i2 / M - Bx [ k ] ) / cos ( HoekKern [ k ] ) );
       VeranderHokje ( hst, MarkeerTijd^ [ i2, j2 ], i2, j2, k,
                       MarkeerIndex^ [ i2, j2 ], N, EindBoven [ k ],
                       HokjesTeller, EindenTeller, EindBoven,
                       EindOnder, MarkeerTijd, MarkeerIndex )
     END;
  2: BEGIN
       vst := T + abs ( ( j2 / M - By [ k ] ) / sin ( HoekKern [ k ] ) );
       VeranderHokje ( vst, MarkeerTijd^ [ i2, j2 ], i2, j2, k,
                       MarkeerIndex^ [ i2, j2 ], N, EindBoven [ k ],
                       HokjesTeller, EindenTeller, EindBoven,
                       EindOnder, MarkeerTijd, MarkeerIndex )
     END;
  3: BEGIN
       hst := T + abs ( ( i2 / M - Bx [ k ] ) / cos ( HoekKern [ k ] ) );
       vst := T + abs ( ( j2 / M - By [ k ] ) / sin ( HoekKern [ k ] ) );
       IF ( hst < vst )
          THEN BEGIN
                 VeranderHokje ( hst, MarkeerTijd^ [ i2, j1 ], i2, j1,
                                 k, MarkeerIndex^ [ i2, j1 ], N,
                                 EindBoven [ k ], HokjesTeller,
                                 EindenTeller, EindBoven, EindOnder,
                                 MarkeerTijd, MarkeerIndex );
                 IF ( EindBoven [ k ] = 1 )
                    THEN VeranderHokje ( vst, MarkeerTijd^ [ i2, j2 ],
                                         i2, j2, k,
                                         MarkeerIndex^ [ i2, j2 ], N,
                                         EindBoven [ k ],
                                         HokjesTeller, EindenTeller,
                                         EindBoven, EindOnder,
```

```
                                                MarkeerTijd, MarkeerIndex )
                    END
                ELSE BEGIN
                       VeranderHokje ( vst, MarkeerTijd^ [ i1, j2 ], i1, j2,
                                       k, MarkeerIndex^ [ i1, j2 ], N,
                                       EindBoven [ k ], HokjesTeller,
                                       EindenTeller, EindBoven, EindOnder,
                                       MarkeerTijd, MarkeerIndex );
                       IF ( EindBoven [ k ] = 1 )
                          THEN VeranderHokje ( hst, MarkeerTijd^ [ i2, j2 ],
                                               i2, j2, k,
                                               MarkeerIndex^ [ i2, j2 ],
                                               N, EindBoven [ k ],
                                               HokjesTeller, EindenTeller,
                                               EindBoven, EindOnder,
                                               MarkeerTijd, MarkeerIndex )
                    END
          END
    END; {CASE}
    IF ( EindBoven [ k ] = 1 )
       THEN Line ( trunc ( Bx [ k ] * GetMaxX ), trunc ( By [ k ] * GetMaxY ),
            trunc ( x * GetMaxX ), trunc ( y * GetMaxY ) );
    Bx [ k ] := x;
    By [ k ] := y
END;
IF ( EindOnder [ k ] = 1 ) THEN
BEGIN
    x := Ox [ k ] - cos ( HoekKern [ k ] ) / M;
    y := Oy [ k ] - sin ( HoekKern [ k ] ) / M;
    i1 := RondAf ( M * Ox [ k ] );
    j1 := RondAf ( M * Oy [ k ] );
    i2 := RondAf ( M * x );
    j2 := RondAf ( M * y );
    IF BuitenRooster ( i2, j2, M )
       THEN BEGIN
              EindOnder [ k ] := 2;
              EindenTeller := EindenTeller - 1;
              code := 0
            END
```

```
   ELSE code := abs ( i2 - i1 ) + 2 * abs ( j2 - j1 );
CASE code OF
  1: BEGIN
       hst := T + abs ( ( i2 / M - Ox [ k ] ) / cos ( - HoekKern [ k ] ) )
       VeranderHokje ( hst, MarkeerTijd^ [ i2, j2 ], i2, j2, k + N,
                       MarkeerIndex^ [ i2, j2 ], N, EindOnder [ k ],
                       HokjesTeller, EindenTeller, EindBoven,
                       EindOnder, MarkeerTijd, MarkeerIndex )
     END;
  2: BEGIN
       vst := T + abs ( ( j2 / M - Oy [ k ] ) / sin ( - HoekKern [ k ] ) )
       VeranderHokje ( vst, MarkeerTijd^ [ i2, j2 ], i2, j2, k + N,
                       MarkeerIndex^ [ i2, j2 ], N, EindOnder [ k ],
                       HokjesTeller, EindenTeller, EindBoven,
                       EindOnder, MarkeerTijd, MarkeerIndex )
     END;
  3: BEGIN
       hst := T + abs ( ( i2 / M - Ox [ k ] ) / cos ( - HoekKern [ k ] ) )
       vst := T + abs ( ( j2 / M - Oy [ k ] ) / sin ( - HoekKern [ k ] ) )
       IF ( hst < vst )
          THEN BEGIN
                 VeranderHokje ( hst, MarkeerTijd^ [ i2, j1 ], i2, j1,
                                 k + N, MarkeerIndex^ [ i2, j1 ], N,
                                 EindOnder [ k ], HokjesTeller,
                                 EindenTeller, EindBoven, EindOnder,
                                 MarkeerTijd, MarkeerIndex );
                 IF ( EindOnder [ k ] = 1 )
                    THEN VeranderHokje ( vst, MarkeerTijd^ [ i2, j2 ],
                                         i2, j2, k + N,
                                         MarkeerIndex^ [ i2, j2 ],
                                         N, EindOnder [ k ],
                                         HokjesTeller, EindenTeller,
                                         EindBoven, EindOnder,
                                         MarkeerTijd, MarkeerIndex )
               END
          ELSE BEGIN
                 VeranderHokje ( vst, MarkeerTijd^ [ i1, j2 ], i1, j2,
                                 k + N, MarkeerIndex^ [ i1, j2 ], N,
                                 EindOnder [ k ], HokjesTeller,
```

```
                                        EindenTeller, EindBoven, EindOnder,
                                        MarkeerTijd, MarkeerIndex );
                        IF ( EindOnder [ k ] = 1 )
                           THEN VeranderHokje ( hst, MarkeerTijd^ [ i2, j2 ],
                                                i2, j2, k + N,
                                                MarkeerIndex^ [ i2, j2 ],
                                                N, EindOnder [ k ],
                                                HokjesTeller, EindenTeller,
                                                EindBoven, EindOnder,
                                                MarkeerTijd, MarkeerIndex )
                     END
               END
         END; {CASE}
         IF ( EindOnder [ k ] = 1 )
            THEN Line ( trunc ( Ox [ k ] * GetMaxX ), trunc ( Oy [ k ] * GetMaxY )
                 trunc ( x * GetMaxX ), trunc ( y * GetMaxY ) );
         Ox [ k ] := x;
         Oy [ k ] := y
      END
   END;
   T := T + 1 / M;
   Delay ( 10 );
   writeln ( uitv1, T:8:5, EindenTeller:5 );
   writeln ( uitv2, T:8:5, HokjesTeller:5 );
END;
readln;
CloseGraph;
DISPOSE ( MarkeerTijd );
DISPOSE ( MarkeerIndex );
CLOSE ( uitv1 );
 CLOSE ( uitv2 );
END
```

```
PROGRAM KristalEinden;

Const Nmax = 1000;
      Mmax = 103;
      Vitesse = 1;

TYPE  arr = ARRAY [ 1 .. Nmax ] OF real;
      ari = ARRAY [ 1 .. Nmax ] OF integer;
      art = ARRAY [ 1 .. 10 * Mmax ] OF real;
      mat = ARRAY [ 0 .. Mmax , 0 .. Mmax ] OF real;
      matpointer = ^mat;
      mti = ARRAY [ 0 .. Mmax , 0 .. Mmax ] OF integer;
      mtipointer = ^mti;

VAR   i1, i2, j1, j2, k, M, N, code, p, proces, aantal: integer;
      teller, HokjesTeller, EindenTeller, KernenTeller, KernenErbij: integer;
      x, y, hst, vst, T, EindTijd, Lambda_begin: real;
      KernX, KernY, HoekKern: arr;
      Bx, By, Ox, Oy: arr;
      BeginTijdKern, EindBoven, EindOnder: ari;
      AantalEinden, AantalHokjes: art;
      MarkeerTijd: matpointer;
      MarkeerIndex: mtipointer;
      uitv1, uitv2: text;

FUNCTION Lambda ( T: real ): real;
BEGIN
  Lambda := Lambda_begin * T + Lambda_begin;
END;

PROCEDURE Tijdstippen ( VAR BeginTijdKern: ari; EindTijd: real;
                        M: integer; VAR N: integer);
VAR i: integer;
    x, interval: real;
BEGIN
  x := ( random ( 10000 ) / 10000 );
  WHILE ( x < 1E-30 ) DO x := ( random ( 10000 ) / 10000 );
  interval := - ln ( x ) * Lambda ( 0 );
```

```
  BeginTijdKern [ 1 ] := trunc ( M * interval );
  i := 2;
  WHILE ( ( i <= N ) AND ( BeginTijdKern [ i - 1 ] < EindTijd * M ) ) DO
  BEGIN
    x := ( random ( 10000 ) / 10000 );
    WHILE ( x < 1E-30 ) DO x := ( random ( 10000 ) / 10000 );
    interval := - ln ( x ) * Lambda ( BeginTijdKern [ i - 1 ] / M );
    BeginTijdKern [ i ] := BeginTijdKern [ i - 1 ] + trunc ( M * interval );
    i := i + 1
  END;
  N := i - 1;
END;

PROCEDURE Coordinaten ( VAR KernX, KernY: arr; Aantal: integer );
VAR i: integer;
BEGIN
  FOR i := 1 TO Aantal DO
  BEGIN
    KernX [ i ] := ( random ( 10000 ) / 10000 );
    KernY [ i ] := ( random ( 10000 ) / 10000 );
  END
END;

PROCEDURE Hoeken ( VAR HoekKern: arr; Aantal: integer );
CONST pie = 3.1415;
VAR i: integer;
BEGIN
  FOR i := 1 TO Aantal DO HoekKern [ i ] := pie * ( random ( 10000 ) / 10000 )
END;

PROCEDURE Initialisatie ( VAR MarkeerTijd: matpointer;
                          VAR MarkeerIndex: mtipointer;
                          M, N: integer; KernX, KernY: arr;
                          VAR Bx, By, Ox, Oy: arr;
                          VAR EindBoven, EindOnder: ari );
VAR i, j:integer;
BEGIN

  FOR i := 0 TO M - 1 DO
```

```
    FOR j := 0 TO M - 1 DO
    BEGIN
      MarkeerTijd^ [ i, j ] := 1000;
      MarkeerIndex^ [ i, j ] := 0
    END;
  FOR i := 1 TO N DO
  BEGIN
    Bx [ i ] := KernX [ i ];
    By [ i ] := KernY [ i ];
    Ox [ i ] := KernX [ i ];
    Oy [ i ] := KernY [ i ];
    EindBoven [ i ] := 0;
    EindOnder [ i ] := 0;
  END
END;

FUNCTION Doorgaan ( HokjesTeller, M: integer; T, EindTijd: real ): boolean;
BEGIN
  Doorgaan := ( ( HokjesTeller < ( M * M ) ) AND ( T < EindTijd ) )
END;

FUNCTION RondAf ( Getal: real ): integer;
VAR teken: integer;
BEGIN
  IF ( Getal >= 0 ) THEN teken := 1 ELSE teken := -1;
  RondAf := teken * trunc ( abs ( Getal ) )
END;

PROCEDURE VeranderHokje ( Snijtijd, Tijd: real; i, j, k, Index, N: integer;
                          VAR Waarde, HokjesTeller, EindenTeller: integer;
                          VAR EindBoven, EindOnder: ari; VAR MarkeerTijd: matpoi
                          VAR MarkeerIndex: mtipointer );
BEGIN
  IF ( Snijtijd < Tijd )
     THEN BEGIN
            IF ( Tijd > 900 )
               THEN HokjesTeller := HokjesTeller + 1
               ELSE BEGIN
                      IF ( Index <= N )
```

```
                        THEN BEGIN
                               IF EindBoven [ Index ] = 1
                                  THEN EindenTeller := EindenTeller - 1;
                               EindBoven [ Index ] := 2
                             END
                        ELSE BEGIN
                               IF EindOnder [ Index - N ] = 1
                                  THEN EindenTeller := EindenTeller - 1;
                               EindOnder [ Index - N ] := 2
                             END;
                  END;
            MarkeerIndex^ [ i, j ] := k;
            MarkeerTijd^ [ i, j ] := Snijtijd
          END
     ELSE BEGIN
            Waarde := 2;
            EindenTeller := EindenTeller - 1
          END
END;

FUNCTION NieuweKernen ( Tijd: real; BeginTijdKern: ari;
                        M, N, KernenTeller: integer ): integer;
VAR k, som:integer;
    doorgaan: boolean;
BEGIN
  k := KernenTeller + 1;
  som := 0;
  doorgaan := ( k <= N );
  IF doorgaan THEN doorgaan := ( Tijd >= BeginTijdKern [ k ] / M );
  WHILE doorgaan DO
  BEGIN
    som := som + 1;
    k := k + 1;
    doorgaan := ( k <= N );
    IF doorgaan THEN doorgaan := ( Tijd >= BeginTijdKern [ k ] / M )
  END;
  NieuweKernen := som
END;
```

```
FUNCTION BuitenRooster ( i, j, M: integer ): boolean;
BEGIN
  BuitenRooster := ( ( i < 0 ) OR ( j < 0 ) OR ( i >= M ) OR ( j >= M ) )
END;


BEGIN {Hoofdprogramma}
  ASSIGN ( uitv1, 'Einden.uit' );
  REWRITE ( uitv1 );
  ASSIGN ( uitv2, 'Hokjes.uit' );
  REWRITE ( uitv2 );
  writeln ( 'Geef roosteromvang:' );
  readln ( M );
  writeln ( 'Geef beginwaarde van exponentiele verdeling:' );
  readln ( Lambda_begin );
  writeln ( 'Geef eindtijd van proces:' );
  readln ( EindTijd );
  writeln ( 'Geef aantal processen:' );
  readln ( proces );
  aantal := trunc ( M * EindTijd ) + 1;
  writeln ( 'aantal=', aantal );
  Randomize;
  FOR i1 := 1 TO aantal DO
  BEGIN
    AantalEinden [ i1 ] := 0;
    AantalHokjes [ i1 ] := 0
  END;
  NEW ( MarkeerTijd );
  NEW ( MarkeerIndex );
 FOR p := 1 TO proces DO
 BEGIN
  teller := 0;
  T := 0;
  HokjesTeller := 0;
  KernenTeller := 0;
  EindenTeller := 0;
  N := Nmax;
  Tijdstippen ( BeginTijdKern, EindTijd, M, N );
```

```
Coordinaten ( KernX, KernY, N );
Hoeken ( HoekKern, N );
Initialisatie ( MarkeerTijd, MarkeerIndex, M, N, KernX, KernY,
                Bx, By, Ox, Oy, EindBoven, EindOnder );
WHILE Doorgaan ( HokjesTeller, M, T, EindTijd ) DO
BEGIN
  KernenErbij := NieuweKernen ( T, BeginTijdKern, M, N, KernenTeller );
  FOR k := ( KernenTeller + 1 ) TO ( KernenTeller + KernenErbij ) DO
  BEGIN
    i1 := RondAf ( M * KernX [ k ] );
    j1 := RondAf ( M * KernY [ k ] );
    IF ( MarkeerTijd^ [ i1, j1 ] > 999 )
       THEN BEGIN
              HokjesTeller := HokjesTeller + 1;
              EindenTeller := EindenTeller + 2;
              EindBoven [ k ] := 1;
              EindOnder [ k ] := 1;
              MarkeerTijd^ [ i1, j1 ] := T;
              MarkeerIndex^ [ i1, j1 ] := k
            END
       ELSE BEGIN
              EindBoven [ k ] := 2;
              EindOnder [ k ] := 2
            END
  END;
  KernenTeller := KernenTeller + KernenErbij;
  FOR k := 1 TO KernenTeller DO
  BEGIN
    IF ( EindBoven [ k ] = 1 ) THEN
    BEGIN
      x := Bx [ k ] + cos ( HoekKern [ k ] ) / M;
      y := By [ k ] + sin ( HoekKern [ k ] ) / M;
      i1 := RondAf ( M * Bx [ k ] );
      j1 := RondAf ( M * By [ k ] );
      i2 := RondAf ( M * x );
      j2 := RondAf ( M * y );
      IF BuitenRooster ( i2, j2, M )
         THEN BEGIN
                EindBoven [ k ] := 2;
```

```
          EindenTeller := EindenTeller - 1;
          code := 0
        END
    ELSE code := abs ( i2 - i1 ) + 2 * abs ( j2 - j1 );
CASE code OF
  1: BEGIN
       hst := T + abs ( ( i2 / M - Bx [ k ] ) / cos ( HoekKern [ k ] ) )
       VeranderHokje ( hst, MarkeerTijd^ [ i2, j1 ], i2, j1, k,
                       MarkeerIndex^ [ i2, j1 ], N, EindBoven [ k ],
                       HokjesTeller, EindenTeller, EindBoven,
                       EindOnder, MarkeerTijd, MarkeerIndex )
     END;
  2: BEGIN
       vst := T + abs ( ( j2 / M - By [ k ] ) / sin ( HoekKern [ k ] ) )
       VeranderHokje ( vst, MarkeerTijd^ [ i1, j2 ], i1, j2, k,
                       MarkeerIndex^ [ i1, j2 ], N, EindBoven [ k ],
                       HokjesTeller, EindenTeller, EindBoven,
                       EindOnder, MarkeerTijd, MarkeerIndex )
     END;
  3: BEGIN
       hst := T + abs ( ( i2 / M - Bx [ k ] ) / cos ( HoekKern [ k ] ) )
       vst := T + abs ( ( j2 / M - By [ k ] ) / sin ( HoekKern [ k ] ) )
       IF ( hst < vst )
          THEN BEGIN
                 VeranderHokje ( hst, MarkeerTijd^ [ i2, j1 ], i2, j1,
                                 k, MarkeerIndex^ [ i2, j1 ], N,
                                 EindBoven [ k ], HokjesTeller,
                                 EindenTeller, EindBoven, EindOnder,
                                 MarkeerTijd, MarkeerIndex );
                 IF ( EindBoven [ k ] = 1 )
                    THEN VeranderHokje ( vst, MarkeerTijd^ [ i2, j2 ],
                                         i2, j2, k, MarkeerIndex^ [ i2,
                                         N, EindBoven [ k ],
                                         HokjesTeller, EindenTeller,
                                         EindBoven, EindOnder,
                                         MarkeerTijd, MarkeerIndex )
               END
          ELSE BEGIN
                 VeranderHokje ( vst, MarkeerTijd^ [ i1, j2 ], i1, j2,
```

```
                                    k, MarkeerIndex^ [ i1, j2 ], N,
                                    EindBoven [ k ], HokjesTeller,
                                    EindenTeller, EindBoven, EindOnder,
                                    MarkeerTijd, MarkeerIndex );
                    IF ( EindBoven [ k ] = 1 )
                       THEN VeranderHokje ( hst, MarkeerTijd^ [ i2, j2 ],
                                            i2, j2, k, MarkeerIndex^ [ i2,
                                            N, EindBoven [ k ],
                                            HokjesTeller, EindenTeller,
                                            EindBoven, EindOnder,
                                            MarkeerTijd, MarkeerIndex )
                  END
        END
  END; {CASE}
  Bx [ k ] := x;
  By [ k ] := y
END;
IF ( EindOnder [ k ] = 1 ) THEN
BEGIN
  x := Ox [ k ] - cos ( HoekKern [ k ] ) / M;
  y := Oy [ k ] - sin ( HoekKern [ k ] ) / M;
  i1 := RondAf ( M * Ox [ k ] );
  j1 := RondAf ( M * Oy [ k ] );
  i2 := RondAf ( M * x );
  j2 := RondAf ( M * y );
  IF BuitenRooster ( i2, j2, M )
     THEN BEGIN
            EindOnder [ k ] := 2;
            EindenTeller := EindenTeller - 1;
            code := 0
          END
     ELSE code := abs ( i2 - i1 ) + 2 * abs ( j2 - j1 );
  CASE code OF
    1: BEGIN
         hst := T + abs ( ( i2 / M - Ox [ k ] ) / cos ( - HoekKern [ k ] )
         VeranderHokje ( hst, MarkeerTijd^ [ i2, j1 ], i2, j1, k + N,
                         MarkeerIndex^ [ i2, j1 ], N, EindOnder [ k ],
                         HokjesTeller, EindenTeller, EindBoven,
                         EindOnder, MarkeerTijd, MarkeerIndex )
```

```
   END;
2: BEGIN
     vst := T + abs ( ( j2 / M - Oy [ k ] ) / sin ( - HoekKern [ k ] )
     VeranderHokje ( vst, MarkeerTijd^ [ i1, j2 ], i1, j2, k + N,
                     MarkeerIndex^ [ i1, j2 ], N, EindOnder [ k ],
                     HokjesTeller, EindenTeller, EindBoven,
                     EindOnder, MarkeerTijd, MarkeerIndex )
   END;
3: BEGIN
     hst := T + abs ( ( i2 / M - Ox [ k ] ) / cos ( - HoekKern [ k ] )
     vst := T + abs ( ( j2 / M - Oy [ k ] ) / sin ( - HoekKern [ k ] )
     IF ( hst < vst )
        THEN BEGIN
               VeranderHokje ( hst, MarkeerTijd^ [ i2, j1 ], i2, j1,
                               k + N, MarkeerIndex^ [ i2, j1 ], N,
                               EindOnder [ k ], HokjesTeller,
                               EindenTeller, EindBoven, EindOnder,
                               MarkeerTijd, MarkeerIndex );
               IF ( EindOnder [ k ] = 1 )
                  THEN VeranderHokje ( vst, MarkeerTijd^ [ i2, j2 ],
                                       i2, j2, k + N, MarkeerIndex^ [
                                       N, EindOnder [ k ],
                                       HokjesTeller, EindenTeller,
                                       EindBoven, EindOnder,
                                       MarkeerTijd, MarkeerIndex )
             END
        ELSE BEGIN
               VeranderHokje ( vst, MarkeerTijd^ [ i1, j2 ], i1, j2,
                               k + N, MarkeerIndex^ [ i1, j2 ], N,
                               EindOnder [ k ], HokjesTeller,
                               EindenTeller, EindBoven, EindOnder,
                               MarkeerTijd, MarkeerIndex );
               IF ( EindOnder [ k ] = 1 )
                  THEN VeranderHokje ( hst, MarkeerTijd^ [ i2, j2 ],
                                       i2, j2, k + N, MarkeerIndex^ [
                                       N, EindOnder [ k ],
                                       HokjesTeller, EindenTeller,
                                       EindBoven, EindOnder,
                                       MarkeerTijd, MarkeerIndex )
```

```
                    END
                END
            END; {CASE}
            Ox [ k ] := x;
            Oy [ k ] := y
          END
        END;
        T := T + 1 / ( M * Vitesse );
        teller := teller + 1;
        AantalEinden [ teller ] := AantalEinden [ teller ] + EindenTeller;
        AantalHokjes [ teller ] := AantalHokjes [ teller ] + HokjesTeller
      END;
      writeln ( ' proces nr. ', p );
    END;
      FOR i1 := 1 TO aantal DO
      BEGIN
        AantalEinden [ i1 ] := AantalEinden [ i1 ] / proces;
        AantalHokjes [ i1 ] := AantalHokjes [ i1 ] / proces;
        T := i1 / M;
        writeln ( uitv1, T:9:5, ' ', AantalEinden [ i1 ] );
        writeln ( uitv2, T:9:5, ' ', AantalHokjes [ i1 ] )
      END;
      CLOSE ( uitv1 );
      CLOSE ( uitv2 );
      DISPOSE ( MarkeerTijd );
      DISPOSE ( MarkeerIndex )
    END.
```

HOW TO PLACE THE WIPER ON A WINDSCREEN

Contents

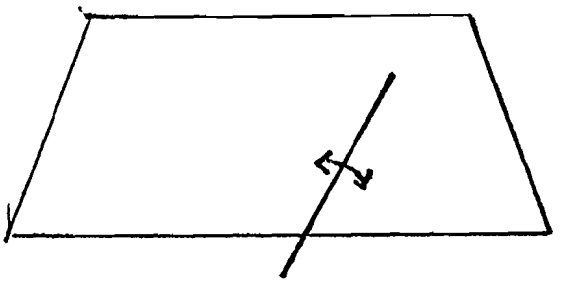
1 Foreword

This report was written as an exercise in applying mathematics with a real-life problem at the Technische Universiteit Eindhoven (NL). It is a part of the post-graduate program 'Mathematics for Industry'.

First we will give an introduction to the problem in plain english, followed by a mathematical formulation of the problem. Then we will give the results that we found. In the fifth section we will present the methods used and in the last section our recommendations for further investigation.

2 Introduction

We were asked to give some mathematically based advise on where to place a common wiper on the border of a windscreen of, for instance a car.



Figuur 1: Where to place the wiper.

Of course, this question is not complete. First, one has to decide what a 'good' wiper must do. Second, there will be some restrictions on the shape and size of both the wiper and the windscreen, given by the sponsor. Third, the place the wiper may be put can not be choosen arbitrary.

3 Mathematical introduction

The following assumptions are made to define the problem:

- The windscreen is flat (2D).
- The size of the windscreen is a polygon in general, but we will look only at a rectangle and a trapezium.
- The shape of the wiper is a (mathematical) linesegment or two connected linesegments with an angle in between.
- A 'good' wiper wipes as much space as possible.

The conclusions of these first assumptions is that the problem will be an optimization problem.

Optimization of the area wiped will be done according to:

- The length of a wiper or both the two lengths and the angle of the special wiper,
- The place of the wiper.

Certain standard techniques are available to do this optimization, but, since these require a lot of computing time, we will not use them and make our own.

4 Results

In this section we will present our results. The first model used to analyze the problem will use a rectangular windscreen and a simple wiper. The second model will use a trapezium shaped windscreen and a special wiper.

4.1 Results for Model I

For an introduction to the model, see the appropriate section.

- case 1 : The window is wider than high.

  We found that there exists a critical ratio between the heigth and the width of the window. If this is smaller than 0.568, then the wiper will be placed in the middle of the window and its length will be half of the width of the window. If on the other hand, the ratio between the heigth and the width is more than 0.568, the wiper will be placed at a distance x from the left corner where x is equal to the heigth. The length of the wiper is also equal to the heigth.

- case 2 : The window is higher than wide.

  We found that there exists a critical ratio between the heigth and the width of the window. If this is smaller than 1.414 then the wiper will be placed in the lefthand corner and its length will be the width of the window. On the other hand, if this ratio is more than 1.414 then the wiper will be placed too in the left hand corner but its length will be the heigth of the windscreen.

4.2 Results Model II

For an introduction to the model, see the appropriate section.

We have calculated the following results for a few popular cars:

name	w	h	e	$\frac{\alpha}{\pi}$	(O_x, O_y)	l	Surf	%	$\frac{\phi_{max}}{\pi}$	$\frac{\delta(\phi)}{\pi}$
Panda	1.02	0.43	0.05	0.47	(0.51,0)	0.43	0.29	69.3	1	1
Kadett	1.10	0.67	0.09	0.46	(0.42,0)	0.67	0.47	70.1	0.67	0.67
Escort	1.11	0.70	0.09	0.46	(0.40,0)	0.70	0.50	70.2	0.65	0.65
Renault	0.35	0.51	0.13	0.42	(0.68,0)	0.51	0.41	65.4	1	1
Peugeot	1.17	0.62	0.10	0.50	(0.59,0)	0.58	0.52	78.9	1	1
Golf	1.17	0.50	0.12	0.43	(0.59,0)	0.50	0.39	74.4	1	1

where w is the width of the window, h the heigth, e the excentricity, (O_x, O_y) the place the wiper will be put, Surf the wiped area, % the percentage of wiped area, and the rest speaks for itself.

We found that the wiper will always be put at the border of the windscreen and not below, that there is no angle between the two linesegments and even that there are no two linesegments, but only one and thus that we are 'in casu' in the same situation as the ordinary wiper placed at the bottom of the window. Why this is the case? We suspect that it has something to do with only allowing a maximum of 10% of the heigth below the bottom of the window to be the place to put a wiper.

5 Mathematical Solution

We will introduce 2 models to discuss the problem. The first one is a rather simple one and can be analyzed analytically, the second is not so simple and will require a computer program.

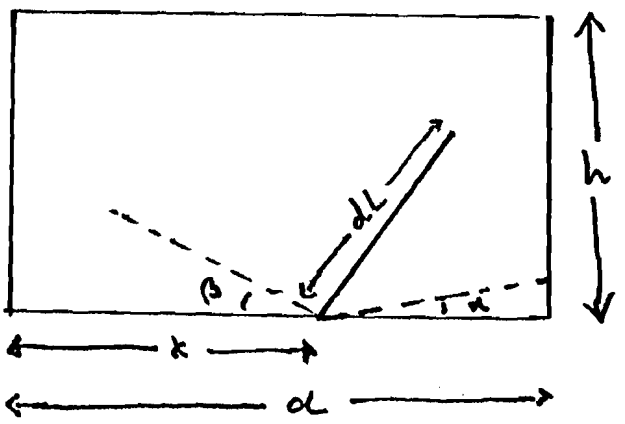
5.1 Model I

5.1.1 Parameters, Variables and Assumptions for Model I

In this model we look at the case that the windscreen is a rectangle and the wiper is a simple straigth linesegment.

The variables and assumptions in this case are :

- The width of the windscreen is d,
- The length of the wiper is dl,
- The heigth of the windscreen is dh,
- The wiper is placed at the border of the windscreen, at a distance x from the lefthand corner.
- The wiper starts under an angle α with the positive x-axis and ends with an angle β with the negative x-axis..



Figuur 2: Model I.

5.1.2 Constraints on Model I

This model has a few limitations. First, ofcourse, a windscreen is not flat. Second, a recrangle windscreen is not often used and third, the wiper is mostly not situated on the border of the windscreen.

5.1.3 Analytical Approach Model I

For a first introduction we assume

$$dl \leq dh \tag{1}$$

$$dh \leq d \tag{2}$$

We now can deduce

$$cos\alpha = \frac{d-dx}{dl} = \frac{1-x}{l} \tag{3}$$

$$cos\beta = \frac{dx}{dl} = \frac{x}{l} \tag{4}$$

or

$$x = \frac{cos\beta}{cos\alpha + cos\beta} \tag{5}$$

$$l = \frac{1}{cos\alpha + cos\beta} \tag{6}$$

The area wiped is

$$A = d^2 \frac{\pi - \alpha - \beta}{2} l^2 \tag{7}$$

$$= d^2 \frac{\pi - \alpha - \beta}{4(cos\alpha + cos\beta)^2} \tag{8}$$

By now, we can take into account the restrictions (1) and (2). The last one means that $h \leq 1$ and from (1) we deduce

$$\frac{1}{cos\beta + cos\alpha} \leq h \tag{9}$$

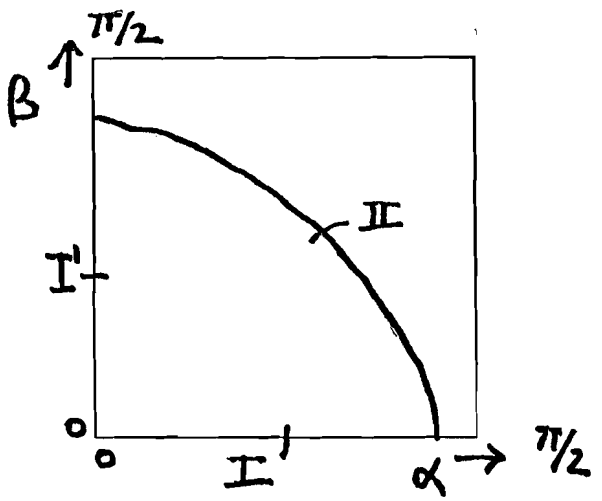
Since obvious

$$0 \leq \alpha \leq \frac{\pi}{2}$$

$$0 \leq \beta \leq \frac{\pi}{2}$$

we have the following domain for α and β:

The shaded area in figure (3) is the area that is allowed for α and β. Since the problem has no global maximum, the maxima will be found on the boundary. Thus, we will devide the boundary into 3 parts :



Figuur 3: Domain for α and β, for $h \leq 1$

- I and I' : Since the problem is symmetrical for α and β we will only consider the case $\beta = 0$,
- II : The curve is described by (9) and the obvious restrictions on α and β. Note that $0 \leq h \leq 1$ and thus the curve will remain in the lower left hand corner.

On II the area is

$$A = d^2 \frac{\pi - \alpha - \beta}{2} h^2$$

and since the curve II is convex, the maxima will be with $\alpha = 0$ or $\beta = 0$. Taking the last case, the value of α can be calculated from

$$\frac{1}{cos\alpha + 1} = h \tag{10}$$

On I the area is

$$A = A(\alpha) = d^2 \frac{\pi - \alpha}{2(cos\alpha + 1)}$$

Differentiating to α and equating to zero yields

$$cos\alpha + 1 = (\pi - \alpha) sin\alpha$$

This can not be solved analytically. With a simple computer program one yields

$$\alpha' \approx 0.81047028$$

This value of α is not a maximum but a minimum. So a maximum will be found on the endpoints of I:

$$A_{max} = \dot{max}(A(0), A(\alpha_h))$$

where

$$\frac{1}{cos\alpha_h + 1} = h$$

the same equation as (9) with $\beta = 0$.

The critical value of α_h is

$$\alpha_h \approx 0.70589947$$

which can be computed numerically easy and the corresponding h is

$$h(\alpha_h) \approx 0.56785003$$

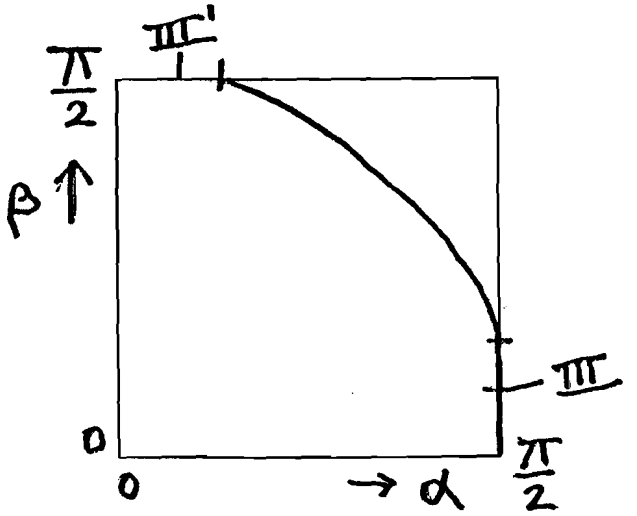
Thus, if $h \leq h(\alpha_h)$ then choose $\alpha = \beta = 0$ else choose $\alpha = arccos(\frac{1}{h} - 1)$ and $\beta = 0$.

Now consider instead of (1) and (2)

$$dl \leq dh \tag{11}$$

$$dh \geq d \tag{12}$$

This will give us the following domain for α and β:



Figuur 4: Domain for α and β, for $h \geq 1$

The maxima will now lie in curve III and III', but, since the problem is symmetrical, we will only consider curve III ($\alpha = \frac{\pi}{2}$). This will give rise to the same analysis and we will not write it out fully, but give only the results.

There exists again a critical value of β

$$\beta_h \approx 0.78539816$$

and the corresponding h is

$$h(\beta_h) \approx 1.41421356$$

So, if $h \leq h(\beta_h)$ then choose $\alpha = \frac{\pi}{2}$ and $\beta = 0$ else choose $\alpha = \frac{\pi}{2}$ and $\beta = arccos(\frac{1}{h})$

5.1.4 Numerical Approach Model I

The numerical work on Model I is very little. We have used a simple bisection method to find the values of α', α_h and β_h.

5.2 Model II

In this case we will look at a windscreen with a trapezium shape and a wiper which consists of two parts connected under a certain angle.

5.2.1 Parameters, Variables and Assumptions for Model II

The variables and assumptions in this case are :

- The width of the windscreen is b,
- The heigth of the windscreen is h,
- The excentricity of the windscreen is e,
- The length of the wiper is l and k, where k is the first part which does not wipe and l is the second part which does wipe.
- The wiper is placed at or under the border of the windscreen, with a maximum of $0.1h$
- The angle between the two parts of the wiper is β
- The lefthandcorner of the windscreen will be placed in the Origin and the wiper will be placed in (O_x, O_y)
- The minimum angle the wiper must make is ϕ_{min} and the maximum angle is ϕ_{max}

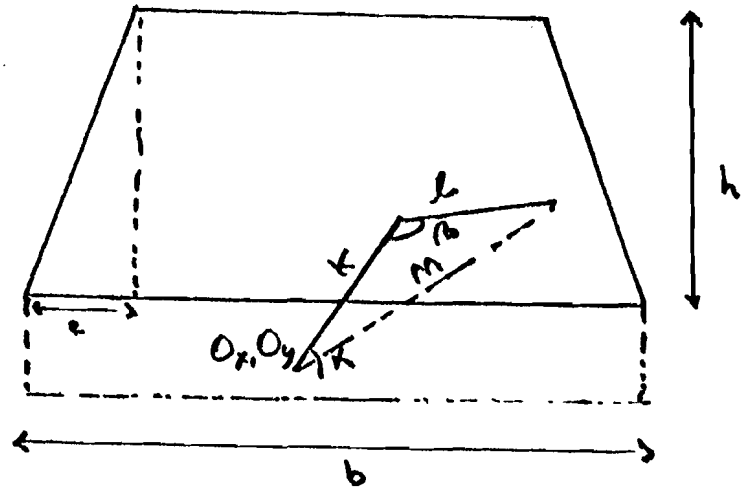
5.2.2 Constraints on Model II

There are again a few constraints for this model. The first is of course that a windscreen is not flat or a trapezium but a more general shape. The second is that the place the wiper may be put is limited rather arbitrairy.

5.2.3 Analytical Approach Model II

The analysis of this model is not an easy task. Therefore it will remain rather introductionary. However, a few things can be said.

A situation O_x, O_y, β, k, l, m, γ, α is allowed if the following equations hold:



Figuur 5: Variables and Parameters of Model II.

- α is the angle of the wiper with the positive x-axis,
- $m^2 = k^2 + l^2 - 2klcos\beta$
- $l^2 = k^2 + m^2 - 2kmcos\gamma$
- $\frac{\pi}{2} \leq \beta \leq \pi$
- $O_y + ksin(\alpha + \gamma) \geq 0$
- $O_y + ksin(\alpha + \gamma) \leq h$
- $O_y + msin(\alpha) \geq 0$
- $O_y + msin(\alpha) \leq h$
- $\frac{hO_x - eO_y}{eksin(\alpha+\gamma) - hkcos(\alpha+\gamma)} \leq 0$ or $\frac{hO_x - eO_y}{eksin(\alpha+\gamma) - hkcos(\alpha+\gamma)} \geq 1$
- $\frac{-O_x - eO_y + h(b-e) + eh}{eksin(\alpha+\gamma) - hkcos(\alpha+\gamma)} \leq 0$ or $\frac{-O_x - eO_y + h(b-e) + eh}{eksin(\alpha+\gamma) - hkcos(\alpha+\gamma)} \geq 1$
- $\frac{hO_x - eO_y}{emsin(\alpha) - hmcos(\alpha)} \leq 0$ or $\frac{hO_x - eO_y}{emsin(\alpha) - hmcos(\alpha)} \geq 1$
- $\frac{-O_x - eO_y + h(b-e) + eh}{emsin(\alpha) - hmcos(\alpha)} \leq 0$ or $\frac{-O_x - eO_y + h(b-e) + eh}{emsin(\alpha) - hmcos(\alpha)} \geq 1$
- $0 \leq O_x \leq b$
- $-0.1h \leq O_y \leq 0$
- $\alpha \geq \phi_{min}$ and $\alpha \leq \phi_{max}$

The object is now to find an α_1 and an α_2 such that these equations are met for all $\alpha_1 \leq \alpha \leq \alpha_2$

This, of course, is not an easy task, especially analytically.

5.2.4 Numerical Approach Model II

We have written a computer program that will give us for a certain set of variables (O_x, O_y, β, k, l) the maximum area that can be wiped with it. The idea behind this program is to intersect the wiper with the border of the windscreen to find the maximum and minimum α. With this procedure one can easy build a grid in R^5, evaluate this maximum in each gridpoint, refine the grid accordig to the maximum maximum found and repeat all this until the required precision is met. The only problem is that it takes a substancial amount of time.

6 Recommendations

Further investigation may be required for the same problem, but with a different shaped windscreen which does not neccesary have to be flat. Also, a better algorithm for determining the optimum is usefull. Third, a different wiper may be investigated.