Chaotic aspects of the dripping faucet

Opleiding
Wiskunde voor de Industrie
Eindhoven

STUDENT REPORT 93-09

CHAOTIC ASPECTS OF THE DRIPPING FAUCET

H. Bonekamp
K. Friso
G. Pruis
A. Mirle

April - July, 1993

Chaotic aspects of the dripping faucet

H. Bonekamp
K. Friso
G. Pruis
A. Mirle

supervisor: J. Molenaar

april - july, 1993

Chaotic aspects of the dripping faucet

Contents

1 Introduction

This is a small study of the chaotic behaviour of a dripping faucet. The idea to study this system stems from Shaw[1984]. It is easily shown in a measurement that the system can be in both periodic and chaotic modes. A model for the dripping faucet is given by Molenaar[1992]. A first analysis of this model is reported by Noack[1992]. The model is based on a simple mass-spring system. The governing equations are linear apart from the action of dripping which is formulated as an instantaneous, non-linear event.

In this project we analyze the chaotic behaviour of time series of the dripping faucet by using different methods. The key issue is to find a reliable value of the largest Lyapunov exponent, which is a measure of chaotic behaviour. Some of the used methods are based on the idea of reconstruction. In section 2 we will shortly explain some important terms from chaos theory. The model of the dripping faucet is presented in section 3. For this model we calculated the largest Lyapunov exponent by three different methods, i.e. the "Wolf method", as is described in Wolf[1985], a direct method, and a method based on Wales[1991]. These methods can be found in section 4.

In all three methods it was explicitly used that the system is considered to be chaotic. However, it could be that the system is not (or weak) chaotic. Therefore we also considered a stochastic prediction method which is described in section 5.

Another aspect of the project is to investigate the behaviour of the model as a function of the flow velocity. It is known that for small flow velocity the system is not chaotic whereas for a higher velocity it is. In section 6 we consider what happens in between.

Finally in section 7 we present our conclusions and give some recommendations for future research.

2 Definitions

In this paragraph we will explain some terms which are frequently referred to in this report such as chaotic system, Lyapunov exponent, (chaotic) attractor and reconstruction. The explanation will be kept short. For more details we refer to Molenaar[1992].

The Lyapunov exponents (LE) λ_i, $i = 1,..n$, of an n-dimensional system give an indication of the strength of the divergence (if any) on an object X in $\mathbb{R}^n$. The sum of the i largest λ_i is a measure for the magnification of an arbitrarily chosen i-dimensional volume while it evolves in time. For example, if λ_1 (the largest eigenvalue) is positive, the length of intervals will on average be stretched. The dynamics on X is then called chaotic and X is called a strange attractor.

Consider a one-dimensional system. If ε_0 is the length of an interval the interval length at time t has changed into

$$\varepsilon(t) = \varepsilon_0 e^{\lambda_1 t}$$

on average. From here it follows that the Laypunov exponent is equal to

$$\lambda_1 = \frac{1}{t} \ln \frac{\varepsilon(t)}{\varepsilon_0}$$

In many practical situations no appropriate model for the system under consideration is known. One does often not know the phase space variables needed to fully describe the dynamics, and even the number of the relevant degrees of freedom is often uncertain. Only measurements are available. One would like to have at one's disposal measurements of the complete state vector $(x_1, x_2, ...)$. In practice only one or a few components are measured. The measured quantity is referred to as the readout function. We shall denote it by $x(t)$. In general x is a function of all state variables, so $x(t) = x(x_1(t), x_2(t), ...)$. How could one determine properties of the complete system, and in particular of the (possible) attractor, from these incomplete data? There is fortunately a partial answer to this question. The standard references are Packard et al.[1980] and Takens[1980]. We call this approach reconstruction.

The key idea is to embed the scalar series $x(t_i)$ into a higher dimensional space with dimension d_e, say. First we have to choose an embedding dimension $d_e \in \mathbb{N}$

and a delay $k \in \mathbb{N}$. Then, we construct a series of d_e-dimensional vectors by the procedure:

$$\begin{aligned} y_0 &= (x_0, x_{0+k}, x_{0+2k}, \ldots, x_{0+d_e \cdot k}) \\ y_1 &= (x_1, x_{1+k}, x_{1+2k}, \ldots, x_{1+d_e \cdot k}) \\ y_2 &= (x_2, x_{2+k}, x_{2+2k}, \ldots, x_{2+d_e \cdot k}) \\ &\vdots \end{aligned} \tag{1}$$

The points $y_i, i = 1, 2, \ldots$ lie on an artificial attractor Y in the artificial d_e-dimensional phase space, and perform there evolutions. A very important observation is (see the standard references mentioned before), that the dynamics on Y has the same characteristics as the dynamics on X. For example, X and Y have identical dimensions and Lyapunov exponents.

In practice d_e is not known in advance. To estimate a reliable value for d_e one applies the embedding procedure for increasing d_e values: $d_e = 1, 2, \ldots$. The theoretical lower bound on d_e is given by $d_e = 2d + 1$, where d is the (capacity or correlation) dimension of Y, thus X. For an explanation of the term dimension, we refer to Molenaar[1992]. However, a short impression can be given as follows: If the attractor is for instance a point then the dimension is 1. If the attractor is a line then the dimension is 2. For chaotic atttactors also a fractal dimension is possible.

The choice of k is not very critical. The value of the time interval $k\Delta t$, where Δt is a small period of time, should not be too small, because then the components of the y_i are nearly identical. If $k\Delta t$ is too large, i.e., much larger than the information decay time determined by the largest Lyapunov exponent, then there is no dynamical correlation between the points.

3 The dripping faucet

Shaw[1984] modelled the dripping faucet in terms of a vibrating mass point. The mass point is attached to the faucet by means of a spring. Its mass $m(t)$ increases at a constant rate, so one of the model equation reads:

$$\frac{\partial m}{\partial t} = \beta. \tag{2}$$

Due to gravity the mass point will move downwards. When it reaches a certain level, part of its mass is instantly cut off at a rate proportional to its velocity. Due to the spring force the mass point will jump upwards. After that the process will repeat itself more or less. The system is one-dimensional, because the motion is only in the vertical direction. The position of the mass is $y(t)$, and its velocity $v(t) = dy/dt$. The second law of Newton reads in this case:

$$\frac{\partial mv}{\partial t} = -mg - ky - \gamma v. \tag{3}$$

We rewrite this as a first order ODE system:

$$\begin{aligned} \frac{\partial y}{\partial t} &= v \\ \frac{\partial v}{\partial t} &= -g - \frac{ky}{m} - \frac{(\gamma + \beta)v}{m} \\ \frac{\partial m}{\partial t} &= \beta. \end{aligned} \tag{4}$$

When $y(t)$ passes y_0 from above, the mass is instantly reduced:

$$m(t) \rightarrow m(t) * (1 - e^{-\alpha/|v(t)|}) \tag{5}$$

The parameters of the system are:

- k: spring constant;
- α: parameter determining the cutting rate;
- β : rate of mass increas representing the faucet flow velocity;

- γ : coefficient of the friction.

In the following sections the parameters will be set to $(k, \alpha, \beta, \gamma) = (10, 0.1, 0.6, 1)$. Only in section 6 we will change the value of β.

The ODE system (5) can be integrated yielding m, v and y as functions of time.

4 Calculation of the largest Lyapunov exponent

In this section we describe three methods to estimate the largest Lyapunov exponents. Some of these methods have been tested by applying them to systems with known Lyapunov exponents. One of thes systems is the tent map, the use of which is presented in some detail.

4.1 The tent map

The tent map is a well known discrete time system which features chaotic behaviour. Time series produced by the tent map are often used to demonstrate chaotic behaviour. We apply it also as a test for the calculation of the first Lyapunov exponent using a prediction method, see Wales[1991] and section 4.4.

$$\begin{aligned} x_{n+1} &= f(x_n) \\ f(x_n) &= z\left(1 - 2|\frac{1}{2} - x_n|\right) \end{aligned} \tag{6}$$

The system is chaotic for $z > 1/2$ and it has a Lyapunov exponent of $\lambda_1 = \ln 2z$. The name tent map becomes clear from Figure 1.

From the positive Lyapunov exponent of the tent map we know that we have to face the loss of significant decimals. Direct implementation of the described model yielded only zeros after approximately 100 iterations due to the finite computer precision. A number is written as a finite number of bits by a computer. If this number is multiplied by 2 in binary code it means the bits are shifted one to the left. It appeared that the computer always writes a 0 behind this string of bits. If the bit string is n bits long, after n multiplications the whole string consists of bits equal to 0. Multiplication from now on only returns zero's. To overcome this problem we use the following trick. We replace equation (7) by

$$x_{n+1} = z\left(1 - \frac{2}{A_n}|\frac{A_n}{2} - A_n x_n|\right) \tag{7}$$

where $A_n > 0$ has a random value in every iteration . In fact this trick

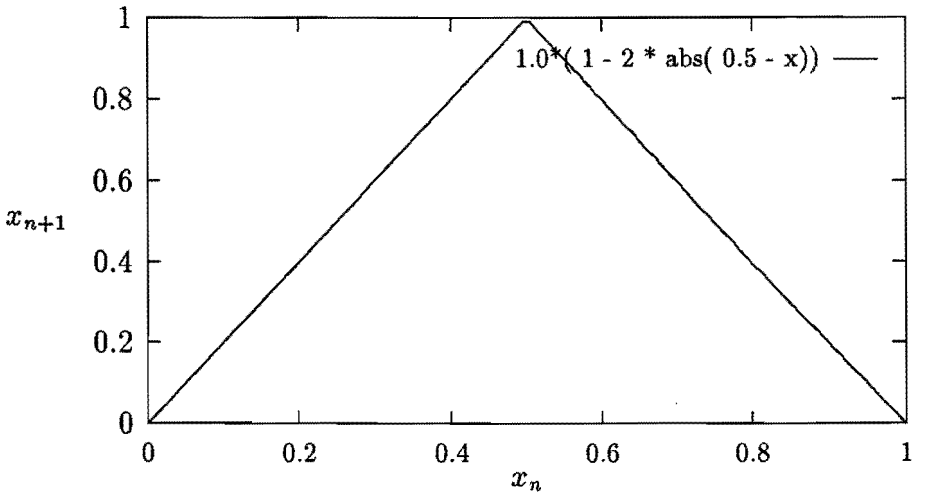

Figure 1: Tent map for $z = 1.0$

means that the starting value x_0 is written as a bit string with infinitely many, arbitrarily chosen decimals.

4.2 The Wolf package

One way to determine the largest Lyapunov exponent is by following two neighbouring trajectories for a certain number of time steps. This approach is worked out in a software package by Wolf[1985]. The approach used in this package is the following:

In a reconstructed state space (see section 2) we choose a point that is called reference point and a point close to it. The orbits through these points are followed and after some time steps the distance between the orbits is calculated. If the distance has become too large (much divergence) then a new trajectory is sought in the neighbourhood of the reference trajectory and the procedure is repeated. The average divergence yields a measure for the Lyapunov exponent.

Several parameters have to be chosen to run the program. First the embedding dimension and the time delay have to be specified. Furthermore an evolution time has to be set. This value says how many time steps the two points will be followed. If the distance between these points has become larger than the parameter *maximum seperation* a new neighbour of the reference trajectory is chosen. Otherwise the process continues with the two trajectories.

During running the trajectories of the points for *evolution time* time steps are plotted on the screen. From these pictures you can see if the points keep close to each other or not. This can be a bit confusing because you see a two-dimensional plot of the trajectories while you are working in a three or four dimensional state space. So, two points which seem to be close to each other can be far apart in reality. But if it happens a lot of times that you see orbital divergence it is better to change the chosen parameters. For example, reduction of the evolution time or the maximum seperation. During running the program the current estimate of the Lyapunov expnent is printed on top of the screen. In this way it is easily checked if this value converges or not.

First we checked if the Wolf package really does what we want. Therefore we made a time series for the following function:

$$x(t) = 2\sin(t) + 5\sin(3t)$$

Of course in this series there is no chaos present and the estimated Lyapunov exponent was indeed almost zero. Another test was the tent map described in section 4.1 for which it is known that the Lyapunov exponent is equal to $\ln 2$. The estimate from the Wolf program was $0.923 * \ln 2$. This was enough evidence that the Wolf package can make reliable estimates.

From the time series of the dripping faucet we obtain four different series:

- position data $y(t)$;
- velocity data $v(t)$;
- mass data $m(t)$;
- time between two drippings $\Delta T(t)$.

On average the time between two drippings is about 2 seconds. Because the time step in the data series is 0.4 seconds, 5 or 6 points will be measured between two drippings. For reconstruction it is the best to use points which are in between two drippings. Therefore the time delay should not be too large. In the runs we used 1 and 3 for the time delay. If the time delay would set larger there is no dynamical correlation between the points any more. The embedding dimension was set to 3 or 4 for the several runs.

In theory every function should be applicable for estimating the Lyapunov exponent. The function chosen is called the *readout function.* We used all four

Position data $y(t)$				
# data points	time delay (time steps)	embedding dim.	evolution time (time steps)	Lyapunov exponent (1/s)
5000	3	4	6	0.20
5000	3	4	4	0.23
5000	3	4	2	0.21
5000	3	4	1	0.21
5000	3	3	2	0.29
5000	3	3	1	0.27
5000	1	4	1	0.20
5000	1	5	1	0.21
10000	3	3	2	0.34
10000	3	3	1	0.33
32000	3	4	2	0.21

Table 1: Estimates for the Lyapunov exponent with position as readout function

Velocity data $v(t)$				
# data points	time delay (time steps)	embedding dim.	evolution time (time steps)	Lyapunov exponent (1/s)
5000	3	4	3	0.25
5000	3	4	5	0.24
5000	1	4	1	0.25
5000	1	5	1	0.24
10000	3	4	3	0.26
32000	3	4	2	0.25

Table 2: Estimates for the Lyapunov exponent with velocity as readout function

Mass data $m(t)$				
# data points	time delay (time steps)	embedding dim.	evolution time (time steps)	Lyapunov exponent (1/s)
5000	3	4	5	0.40
5000	2	4	5	0.42
32000	3	4	2	1.21

Table 3: Estimates for the Lyapunov exponent with mass as readout function

Time between drippings ΔT				
# data points	time delay (time steps)	embedding dim.	evolution time (time steps)	Lyapunov exponent (1/s)
2450	1	4	1	0.52
2450	1	3	1	0.52
2450	1	3	2	0.50
2450	1	4	2	0.48

Table 4: Estimates for the Lyapunov exponent with time between drippings as readout function

series mentioned above as a readout function. The results are shown in Tables 1 to 4, respectively.

We see that the estimates of the Lyapunov exponent using position data or velocity data coincide with each other. But if the mass or the times between drips are considered, the estimate for the Lyapunov exponent becomes much bigger. An explanation could be the following argument:

For two close points in the reconstructed state space it is calculated how the distance between their trajectories increases or decreases in a time interval. The smaller the starting distance and the time interval the better the estimate for the Lyapunov exponent. The function of the mass is discontinous in time, so at jumps the divergence can be very big. Also for the series of the times between jumps two close points can be far apart one iteration later. The position and velocity function are continous in time and are therefore the best to use.

From each table it can be seen that the estimate is quite insensitive for the parameter values. Also the length of the time series seems not to be of great importance. Of course there is a transient period but for this model 5000 data points seem to be enough. This is remarkable compared with the results of the other methods described further on in this section. In those methods longer time series give more accurate estimates.

From Table 1 it follows that an embedding dimension equal to 3 is too small because the largest Lyapunov exponent for those runs show a significant difference with runs with larger embedding dimension. This feature becomes more clear the longer the time series are.

4.3 Direct calulation of λ_1

4.3.1 Calculation with sorting of the points

The behaviour of the dripping faucet is described by a system of linear differential equations and a sudden mass decrease in the plain $y = y_{los}$. If we plot the plain $y = y_{los}$ just before and after the jump, the points of the attractor seem to form two lines. These lines are shown in Figure 2 and Figure 3 for a test series with 10000 data points (2453 jumps).

The state vector of the faucet follows a trajectory described by the differential equations until it reaches the plain $y = y_{los}$. Then it jumps from one line to the other and follows its trajectory again. The attracor is a kind of cutted Möbius band with a gap where the dripping takes place. The Figures 2 and 3

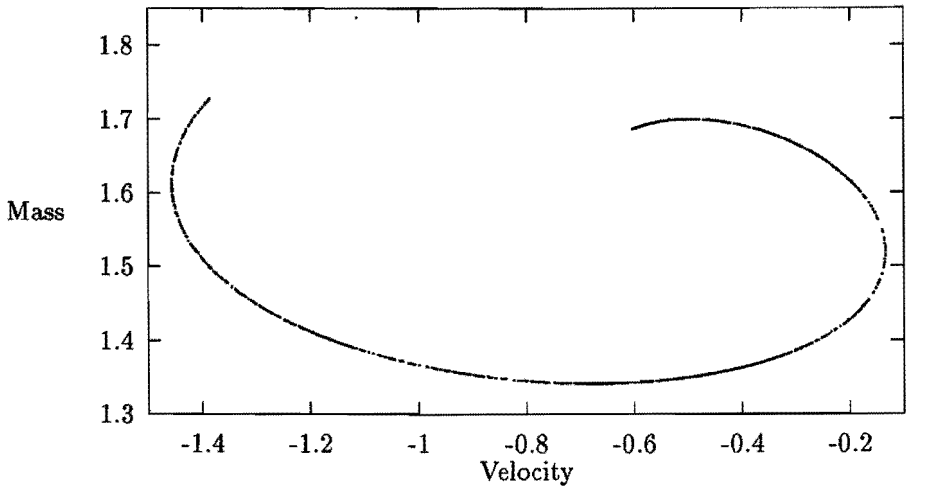

Figure 2: Cross section of the strange attractor with the plane $y = y_{los}$; the section before the jump.

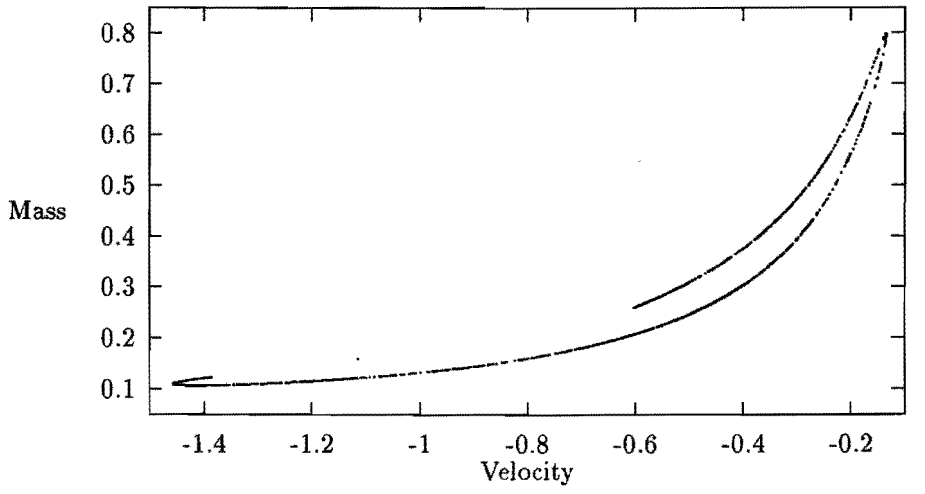

Figure 3: Same cross section as in Figure 2, but after the jump.

suggest that the band is completely flat, *i.e.*, it is two-dimensional. Later on this suggestion will be checked. In the first instance, the assumption is made, that the attractor is a two-dimensional band and the attention is directed to the calculation of the average stretching during jumping.

Sorting of the points

If we want to determine the stretching factor in a point, we need another point in the neighbourhood. Then we can determine the ratio of the distances of these points before and after the jump. In general calculations for finding a neighbouring point are time consuming. Under the assumption that the cross-sections in Figures 2 and 3 are really lines, we can order the points on the line before the jump and then consider two neighbouring points in these order.

Since the line is not a function, it is not possible to order all points at the same time. But by splitting the graph partial functions can be formed. These can be ordered after which they are put together again.

Calculation of the stretching factor during the jump

The program CHAOS_4 orders the points in Figures 2 and 3 and computes then from the sorted list the stretching factor. As described in section 2, the Lyapunov exponent can be numerically determined as the logarithm of the stretching factor of two points starting very close together. Therefore the program takes two neighbouring points of the list, computes the fraction of the distances of these two points before and after the jump and takes the logarithm of this value. To average these values, it adds the logarithms for all points in the list and divides by the number of the points. So it gets a weighted average for the Lyapunov exponent. The results are to be seen in Table 1.

The largest Lyapunov exponent is decreasing with increasing of the number of points. But if the difference between neighbouring points is very small, the results are not very exact since the accuracy of the data is only 5 decimals. So we can only conclude, that the largest Lyapunov exponent is a very small positive value or zero.

# points	# points of jump	stretching factor
1,000	244	0.07363
5,000	979	0.04698
10,000	2,453	0.03032

Table 5: Results of program CHAOS_4

The stretching factor of the regular motion

The dynamics on the band is described by linear differential equations. This does not imply that no stretching takes place. Since the stretching factor of the jump is very small, we compute the stretching during the regular motion.

We look at two neighbouring points after one jump and at the same points just before the next jump. The pascal program CHAOS_5 computes like in the previous part the logarithm of the fraction of the distances and averages this value. The results of CHAOS_5 are summarized in Table 6.

# points	# points of jump	stretching factor
1,000	244	0.33155
5,000	979	0.33350
10,000	2,453	0.35732

Table 6: Results of program CHAOS_5

These values are nearly the same and clearly greater than zero. The interesting aspect of this result is, that the stretching during the regular motion is a clearly present. However, since we consider neither the influence of jump, nor the dependence on time the values are not an estimate for the Lyapunov exponent. They only show, that there is stretching.

The stretching factor of the jump and the regular motion

The pascal program CHAOS_6 tries to determine an estimate for λ_1 taking into account the dynamics of both the band and the jump. Under the assumption of

a two-dimensional band, we observe two neighbouring points before one jump and the same points before the next jump. So, the trajectories are followed so long that exactly one jump is taken into account. The results of CHAOS_6 are shown in Table 3.

# points	# points of jump	stretching factor
1,000	244	0.10771
5,000	979	0.11010
10,000	2,453	0.12652

Table 7: Results of program CHAOS_6

This value is much smaller than the results of the WOLF package (see section 5.1). There are two possible reasons:

- Since the time between two considered successive points is nearly 2 seconds, they evolve faster apart than in the WOLF package. Therefore the result is not very exact.
- The curves may look like a line, but they are not lines.

To find out, if the second reason is true, we analyse the structure of the attractor more exactly in the next part.

4.3.2 Properties of the attractor

To check, if the cross-sections shown in Figures 2 and 3 are really one-dimensional lines, we can look at points after one jump and the same points before the next jump. Since the system of differential equations describing the faucet is autonomous, the trajectories can not cross. But if we look at the ten first points of a series, we find that these points change their order. They not only reverse their order like on an Möbius band, it is also possible, that at the beginning of the regular motion a point lies between two other points but afterwards is not. This is shown in Figure 4 and Figure 5. As far the points with the numbers 2, 6 and 9, the point 6 starts between point 2 and point 9, but after the regular motion the point 9 lies in the middle.

Therefore we analysed the attractor more exactly. We splitted an ordered output of DRIPS into very little parts of only ten points and looked at these

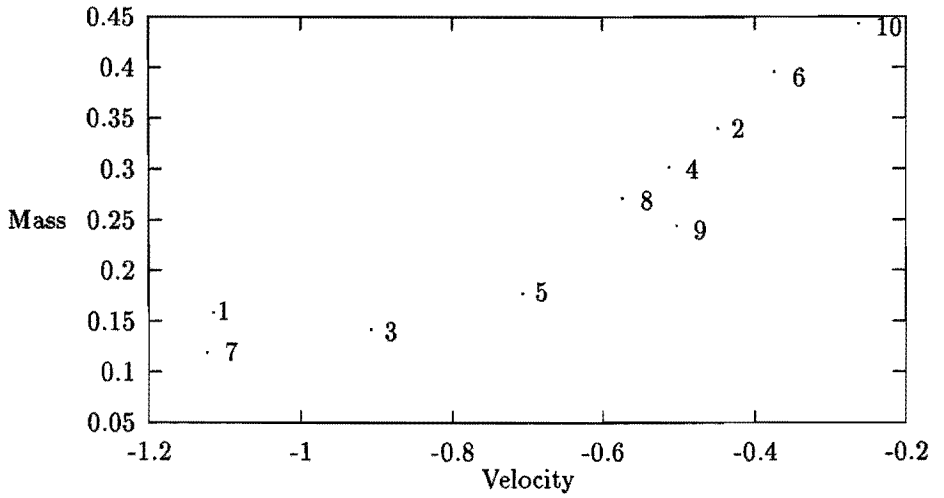

Figure 4: 10 points after the jump

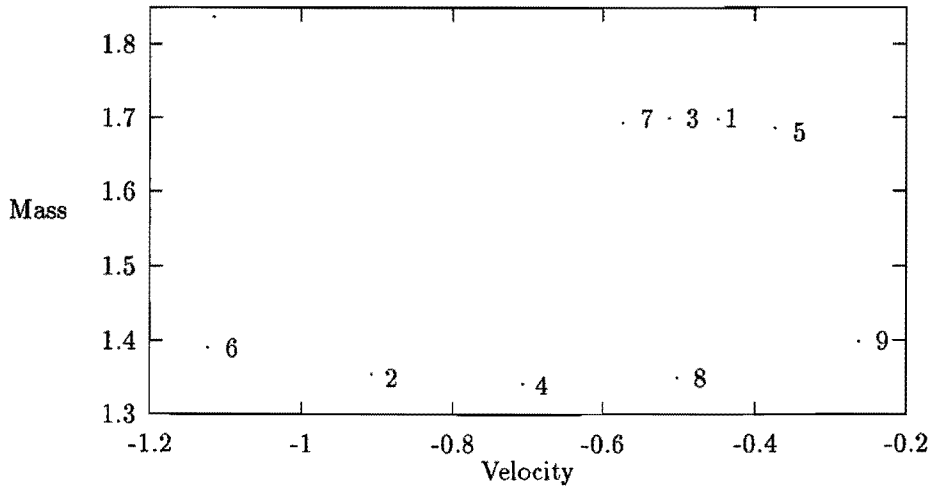

Figure 5: 10 points before the next jump

points after the jump and their position before the next jump to find the exact irregular behaviour. But these small parts showed regular behaviour, the points had before the next jump the same order like after the previous jump.

Then we considered 24 points nearly uniformly distributed on the line just after one jump and determined their position just before the next jump. From Figures 6 and 7 one can see, that the two parts of the folded line after one jump come together. This means that the trajectories don't cross, but the pictures are not really lines, there are a lot of lines very close together.

If you look at a very small part of the line before the jump you can see, that there are two different lines very close together (see Figure 8). The jump brings these lines so close together, that after the jump only one line is observable seen (see Figure 9). There are two lines, but you can see them only in even smaller parts of the whole line (see Figure 10). Before the next jump there are these two lines too, and they are splitted in two lines as result of the folding during the jump. So we can say that there are a lot of lines very close together, but this can only be seen if you zoom in.

Finally we can conclude that the attractor has the dough like structure which is characteristic for chaotic attractors. It consists of infinitely many layers on top of each other. The stretching takes place in these layers. The orbits in each layer diverge from each other. The orbits mainly tend to visit the upper part of the line just before the dripping. The whole structure can be thought of as being constructed by stretching a sheet during the regular motion and folding it during the dripping.

4.3.3 Analytical methods to determine λ_1

From the above we know that the attractor band is three-dimensional, although it is very thin. The dynamics of the jump discrete from $\mathbb{R}^2$ into $\mathbb{R}^2$.

The jump occurs only once between two parts of regular motion. So we can consider the Jacobian of this mapping only in the points just before the jump, we don't have to follow the trajectories.

The regular motion is described by a system of differential equations. So we can follow the motion along the trajectory and determine the greatest possible stretching with the method described in section 4c1 of Molenaar[1992].

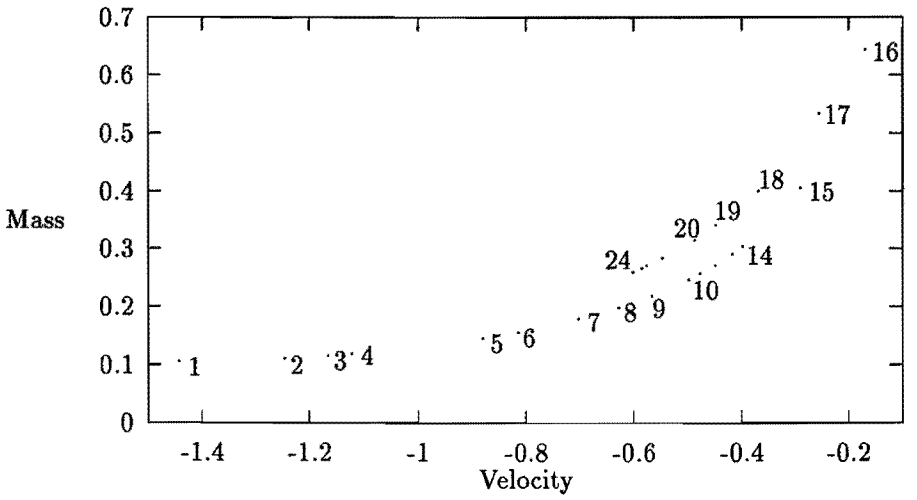

Figure 6: 24 points after one jump

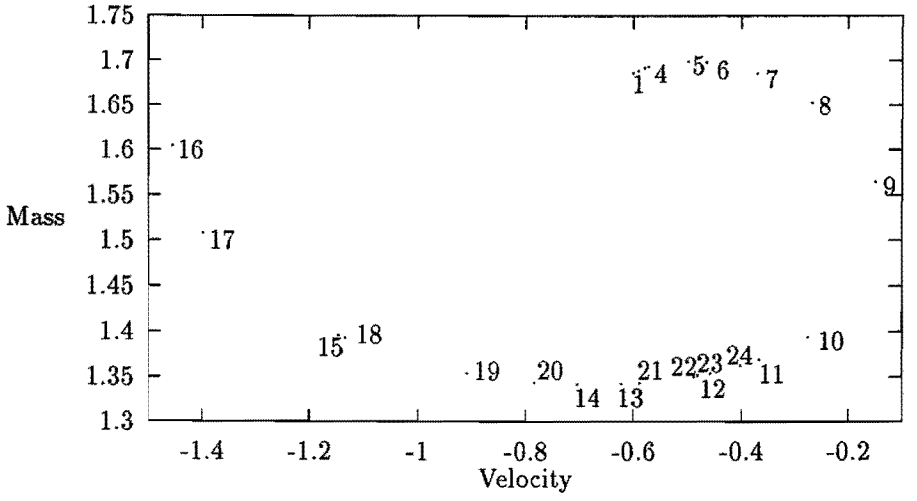

Figure 7: 24 points before the next jump

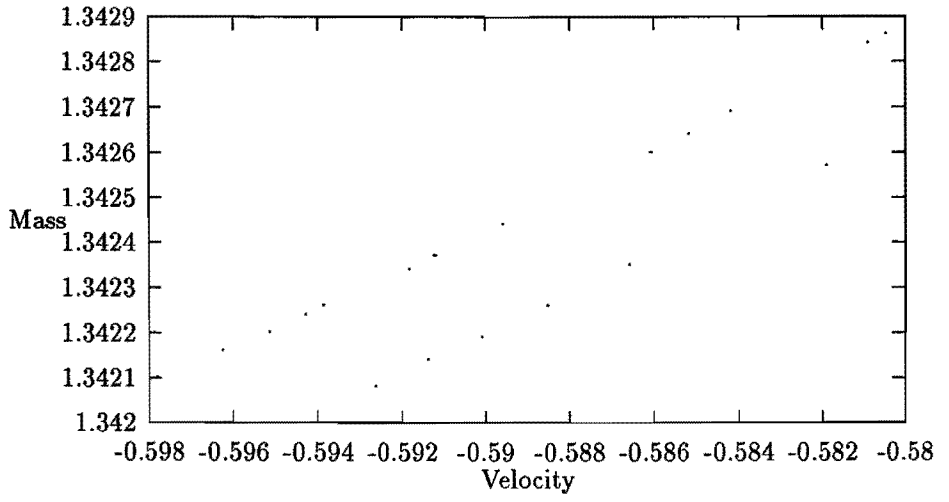

Figure 8: Two lines before the jump

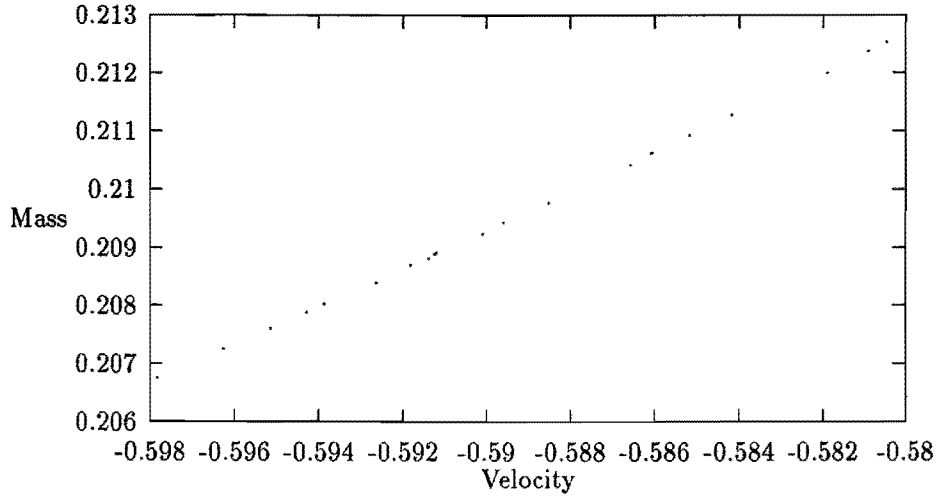

Figure 9: Only one line after the jump?

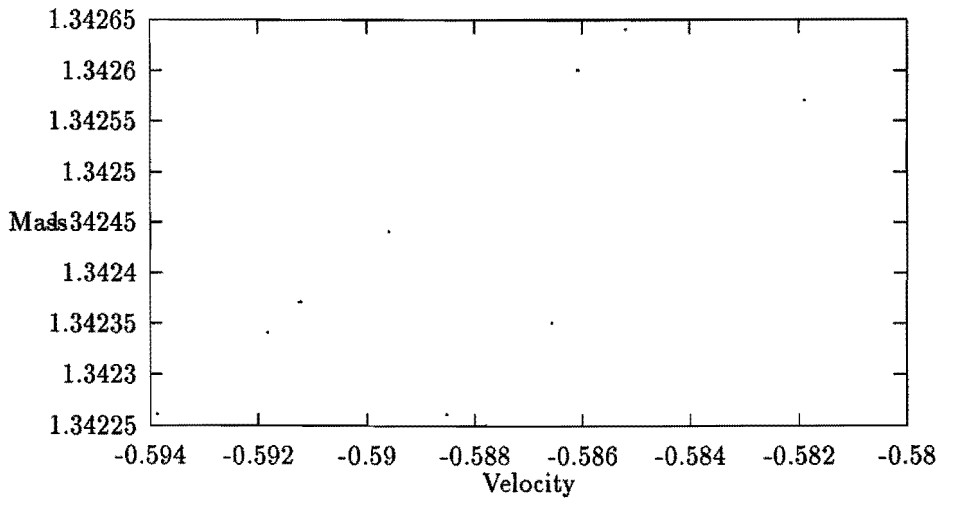

Figure 10: Also two lines after the jump

Eigenvalues of the Jacobi matrix of the jump

The dripping is modelled by a mapping

$$\begin{cases} m_{after} &= (1 - \exp(\alpha / v_{before}))\, m_{before} \\ v_{after} &= v_{before} \end{cases}$$

This can be written as a function from $\mathbb{R}^2$ in $\mathbb{R}^2$:

$$f \begin{pmatrix} v \\ m \end{pmatrix} = \begin{pmatrix} v \\ (1 - e^{\frac{\alpha}{v}})m \end{pmatrix}$$

The Jacobi matrix of this mapping is

$$\begin{pmatrix} 1 & 0 \\ \frac{\alpha}{v^2} m e^{\frac{\alpha}{v}} & 1 - e^{\frac{\alpha}{v}} \end{pmatrix}$$

This is a matrix with all zeros above the main diagonal, and therefore it is easy to determine the eigenvalues of this matrix: these are the elements in the main diagonal. One of the eigenvalues is one and the other is smaller than

one. The logarithms of the eigenvalues are

$$\begin{aligned} \lambda_1 &= 0 \\ \lambda_2 &= \ln(1 - e^{\frac{\alpha}{v}}) \end{aligned}$$

Since the argument of the logarithm in λ_2 is smaller than one, λ_2 is a negative value. The program CHAOS_2 computes for a test series made by DRIPS the weighted average of λ_2. The result for a test series with 1000 measure points (244 jumps) was

$$\lambda_2 = -1.83313$$

Because the greatest eigenvalue of the Jacobian is one, there is no stretching during the jump in all points of the attractor just before the jump. But one can easily see that this cannot be correct. There are points on the attractor with stretching during the jump. For instance, two points with the same mass but different velocities just before the jump do not change their velocities, but have different masses after the jump. So there is stretching in some points during the dripping. Therefore, the greatest eigenvalue can not contain the whole information about the stretching factor. The reason for this fact is that the eigenvalues only give the stretching in the direction of the eigenvectors, whereas the greatest stretching can occur in any direction.

Operator norm of the Jacobi matrix of the jump

The whole information about the largest stretching of a mapping contains the operator norm of the Jacobi matrix J, that means the greatest eigenvalue of $J^T J$. This matrix has the following form :

$$J^T J = \begin{pmatrix} 1 + (\frac{\alpha}{v^2} m e^{\frac{\alpha}{v}})^2 & (\frac{\alpha}{v^2} m e^{\frac{\alpha}{v}})(1 - e^{\frac{\alpha}{v}}) \\ (\frac{\alpha}{v^2} m e^{\frac{\alpha}{v}})(1 - e^{\frac{\alpha}{v}}) & (1 - e^{\frac{\alpha}{v}})^2 \end{pmatrix}$$

If we write the Jacobi matrix J in short notation, the structure of $J^T J$ becomes more clear:

$$J = \begin{pmatrix} 1 & 0 \\ a & b \end{pmatrix} \Rightarrow J^T J = \begin{pmatrix} 1 + a^2 & ab \\ ab & b^2 \end{pmatrix}$$

The eigenvalues of this matrix are :

$$\mu_{1/2} = \frac{a^2 + b^2 + 1}{2} \pm \sqrt{\frac{(a^2 + b^2 + 1)(a^2 + b^2 + 1)}{4} - b^2}$$

The greatest of these two values is :

$$\mu_1 = \frac{a^2+b^2+1}{2} + \sqrt{\frac{(a^2+b^2+1)(a^2+b^2+1)}{4} - b^2}$$

This value gives the greatest stretching factor of the considered mapping, and $\lambda_1 \equiv \log \mu_1$ is the greatest Lyapunov exponent of the mapping. The program CHAOS_3 computes for a test series made by DRIPS the weighted average of λ_1. The results of this program are given in Table 4.

# points	# points of jump	stretching factor
1,000	244	0.15734
5,000	979	0.17524
10,000	2453	0.17582

Table 8: Results of program CHAOS_3

The results for 5000 and 10000 points are nearly the same, so we can say that the real value is between 0.175 and 0.176.

Stretching during the regular motion

The differential equations describing the regular motion are

$$\begin{aligned} \frac{dy}{dt} &= v \\ \frac{dv}{dt} &= -g - \frac{k}{m}y - \frac{\beta+\gamma}{m}v \\ \frac{dm}{dt} &= \beta \end{aligned}$$

In a shorter way we can write this as

$$\dot{\underline{x}} = \mathrm{f}(\underline{x})$$

In a discrete dynamical system $\underline{x}_{n+1} = \mathrm{f}(\underline{x}_n)$ we can consider the Jacobian of the function f to get the stretching factor. This method is described in

Molenaar[1992]. But in the continuous case the function f determines the time deravitive and not the next point in a time series. So we cannot take the eigenvalues of the Jacobi matrix of f to get the stretching factor. Therefore we need a description of a mapping, which attaches to a given point the next point in a discrete time series resulting from the differential equations.

A point $\underline{x} = (y, v, m)$ moves in the time dt to $(y + dy, v + dv, m + dm)$. We can describe this motion by:

$$\phi(\underline{x}) \approx \underline{x} + f(\underline{x})dt = \begin{pmatrix} y + vdt \\ v - (g + \frac{k}{m}y + \frac{\beta+\gamma}{m}v)dt \\ m + \beta dt \end{pmatrix}$$

The Jacobi matrix of this mapping has the form

$$J = \begin{pmatrix} 1 & dt & 0 \\ -\frac{k}{m}dt & 1 - \frac{\beta+\gamma}{m}dt & \frac{k}{m^2}ydt + \frac{\beta+\gamma}{m^2}vdt \\ 0 & 0 & 1 \end{pmatrix}$$

To compute the Lyapunov exponent in $\underline{x}_0$ we have to follow the trajectory starting in the point $\underline{x}_0$. We can start with an arbitrary distance vector $\underline{\epsilon}_0$ between $\underline{x}_0$ and a neighbouring point $\underline{x}_0 + \underline{\epsilon}_0$ and observe the stretching of the interval $[\underline{x}_0, \underline{x}_0 + \underline{\epsilon}_0]$. After one iteration we get

$$\|\epsilon_1\| = \|\phi(\underline{x}_0 + \underline{\epsilon}_0)\| = \|J(\underline{x}_0)\underline{\epsilon}_0\|$$

The approximation with the Jacobian only holds if $dt \to 0$. Repeating the iterations we get

$$\underline{\epsilon}_n = J(\underline{x}_{n-1})J(\underline{x}_{n-2})...J(\underline{x}_0)\underline{\epsilon}_0$$

The value $\ln(\frac{\|\underline{v}_n\|}{\|\underline{v}_0\|})$ gives the stretching in n time steps. To get the average stretching in one time step we have to divide this value by n. So we get a sequence

$$\lambda_1(n) = \frac{1}{n}\ln(\frac{\|\underline{\epsilon}_n\|}{\|\underline{\epsilon}_0\|}) \tag{8}$$

The definition of λ_1 is then given by

$$\lambda_1 = \lim_{n\to\infty} \frac{1}{n} \ln\left(\frac{\|\underline{\epsilon}_n\|}{\|\underline{\epsilon}_0\|}\right) = \lim_{n\to\infty} \lambda_1(n)$$

We have to take two limits: first $dt \to 0$ and then $n \to \infty$.

To get a trajectory describing the whole attractor, we need an output file of DRIPS containing a lot of jumps. The time step between two output points can not be so small, that we can describe the behaviour for $dt \to \infty$ exactly.

We have two possibilities to compute the Lyapunov exponent :

1. We linearize the trajectory between two output points of DRIPS and get an error resulting from the relative large time between these two points.
2. We linearize the trajectory only with a very small time step dt. But we have no points on the trajectory between the output points of DRIPS to compute the Jacobian. So we must take the same Jacobian on the whole trajectory between two output points. Therefore we get here an error, too.

Since the jump has nothing to do with the Jacobian of the regular motion we can not apply the linearization at a measure point if it is directly followed by a jump. So we can follow a trajectory at most until the next jump. That is another source of error.

The pascal programs LYAPUN_1 and LYAPUN_2 compute the Lyapunov exponent with the two respective methods mentioned above. With a short time series of only 1000 points and a time step of 0.1 we tested the influence of the parameters

accuracy	:	The program computes $\lambda_1(n), n = 1, 2, \ldots$ and stops if $\|\lambda_1(n) - \lambda_1(n+1)\| <$ accuracy, since the series $\lambda_1(n)$ shall converge.
ddt	:	The program splits every input time interval dt into smaller parts of length ddt for the second method

to get the best parameters for the computations with larger files. The test results for the two programs are shown in Tables 5 and 6. The value for convergence steps gives on average how many steps were neccesary to reach the given accuracy for the convergence.

Program LYAPUN_1 shows no convergence. We can only get an accuracy of 10^{-2}. If we want to have a better accuracy, then we reach the next jump.

accuracy	λ_1	convergence steps
10^{-3}	0.86891	10.35964
10^{-2}	1.20908	7.66034

Table 9: Test results of program LYAPUN_1

accuracy	ddt	λ_1	convergence steps
10^{-3}	0.010	1.01691	5.64635
10^{-4}	0.010	1.30177	15.80220
10^{-5}	0.010	0.25933	71.09391
10^{-6}	0.010	0.03264	101.96603
10^{-5}	0.001	0.21905	4.27173
10^{-6}	0.001	0.23721	20.69930

Table 10: Test results of program LYAPUN_2

This we can see from the number of convergence steps of more than 10. Since the time between two jumps is nearly 2 seconds, there are 20 points between two drops. So the average number of following points until the next jump is 10. If the number of convergence steps is nearly 10, then all the following points were used for the computations, but no convergence occurred before the next jump. So the program LYAPUN_1 can not be applied to time series with time step 0.1.

The program LYAPUN_2 with time step 0.01 for computing the Jacobian converges until an accuracy of 10^{-5}. If the accuracy is 10^{-6}, then the number of convergence steps is greater than 100, and from the same reason like above, now with a time step of 0.01 and there is no convergence. But since the accuracy of the input data is 10^{-5}, this accuracy is enough for the Lyapunov exponent. So we can say, this program can be applied to time series with time step 0.1 with an accuracy of 10^{-5}.

If we take a smaller time step for the Jacobian, then the convergence speed increases, since the Jacobian is nearly the unit matrix for very small time intervals and the considered Jacobian remains constant during a lot of steps. If the Jacobian changes, then also the directions of the eigenvectors change, and the differences between the computed eigenvectors and the real eigenvectors increase. The results are nearly the same for different time steps. We conclude

that this program can be used for computing Lyapunov exponents.

We applied the program LYAPUN_2 on ten different time series with time step 0.1 and 10,000 points and got

$$\begin{aligned} \text{ddt} = 0.01 \quad &, \text{accuracy} = 10^{-5} \quad \rightarrow \lambda_1 = 0.265 \pm 0.01 \\ \text{ddt} = 0.001 \quad &, \text{accuracy} = 10^{-5} \quad \rightarrow \lambda_1 = 0.223 \pm 0.04 \end{aligned}$$

If ddt is smaller, the linearization is more reliable, but then the influence of the errors resulting from the assumption of the Jacobian being constant increasing. We take the average of both values to get an estimate for the greatest Lyapunov exponent:

$$\lambda_1 \approx 0.24$$

It makes no sense to compute λ_1 with the same program for a time series with smaller time step, because then the series must be very long to describe the whole attractor, and the computing time will consequently increase dramatically.

A possibility to improve the programs is to consider also the exact time point of a jump and apply the Jacobian of the jump in this point. Then the trajectory can be followed further, and the convergence would be better.

4.4 Calculation of λ_1 using the Wales method

By calculating the tent map series we saw that chaotic behaviour is seen as a loss of information due to the fact that small errors can blow up. The rate of this loss is a measure of the chaotic behaviour and related to the Lyapunov exponents. In Wales[1991] a theoretical description is given based on this idea. This article uses the concept of the Kolmogorov entropy K, which is equivalent to the mean loss of information per unit time. The Kolmogorov entropy K is equal to the sum of the positive Lyapunov exponents and in our dripping faucet model, where we have only one positive Lyapunov exponent, K is equal to λ_1.

The "Wales" method works as follows: First, the time series is divided in two parts. The first part is used as a data-base for a prediction method to be described here after. The values in the second part are used as starting points for predictions. Second, the correlation r between the predicted values and the real values is calculated as a function of the number of prediction steps. Third, K and thus λ_1 is given by half the initial slope of the plot of $\ln(1-r)$

against the number of predictions steps in time. The initial slope is the slope of the graph at prediction step zero.

In more detail the prediction works as follows (see also Molenaar[1992]) for more details): Consider a time series $\{x_i, i = 1, ..., M\}$. The procedure to get an estimate for x_{M+1} is based on finding local portions of the time series in the past, which closely resemble the last part and basing the predictions on what occurred immediately after these past events. It should be emphasized that no attempt is made to fit a function to the whole time series at once. The approach can quite simply be automated and may give striking results, if the attractor dimension is not too high and enough data is available. The procedure can be summarized as follows :

- Apply reconstruction. The procedure dealt with in section 2 will yield the embedding dimension d_e. Meanwhile the scalar series $x_i, i = 1, ..., M$ is transformed into a series $y_i, i = 1, ..., M'$ of d_e-dimensional vectors with $M' = M - (d_e\text{-}1)$. The problem is now to estimate $y_{M'+1}$.
- Search neighbours of $y_{M'}$ in the series y_i.
- Find the evolution of these neighbours after one iteration.
- Fit a (linear, quadratic, cubic, or ....) mapping to the one-step-ahead evolutions of the neighbours.
- Apply this map to y'_M. This yields an estimate for $y_{M'+1}$, and the last element of $y_{M'+1}$ is an estimate for x_{M+1}.

This algorithm is applied several times to obtain estimates for $x_{M+1}, x_{M+2}, ...$

4.4.1 Application on the tent map

In Wales[1991] the above described method is checked for the tent map (4.1). We repeated these calculations to verify these results using the chaotic predictor PREDCORS. PREDCORS is the implementation of the chaotic prediction method described in Molenaar[1992].

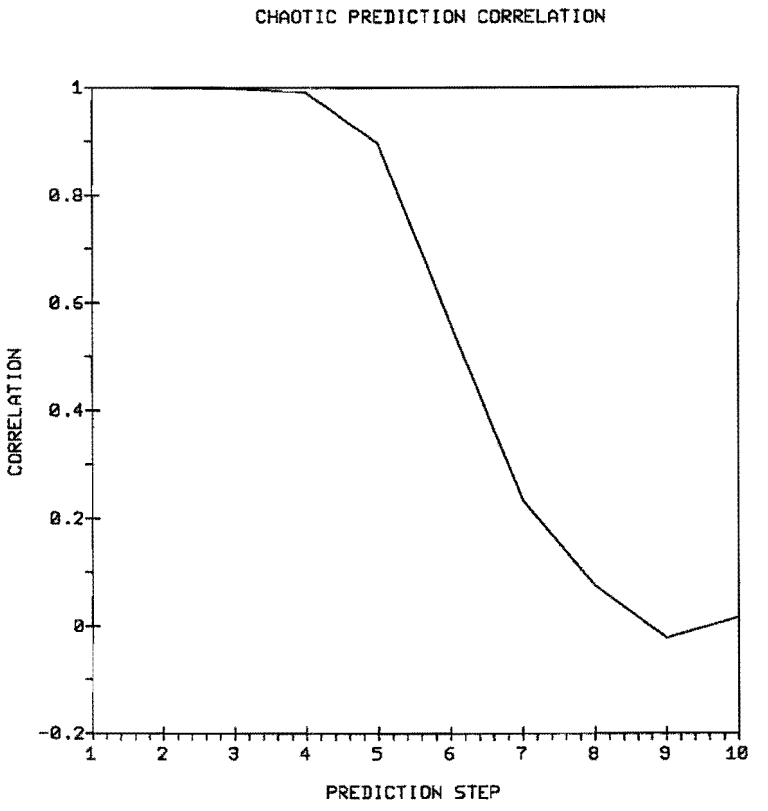

Figure 11: Correlation vs. prediction step using position data

Theoretically the value of λ_1 is $\ln 2 = 0.69315....$ In Wales[1991] the value $\lambda_1 = 0.69310$ was found for a serie of 1000 points in which the first 500 were used to fit the model. Figure 11 shows our results for the same parameters as used in Wales[1991]. We calculated the initial slope using a spline function (see also the next section). We obtained $\lambda_1 = 0.70$. This result is in good agreement with the real value.

4.4.2 Application to the dripping faucet

We have applied the method above on the position data produced by DRIPS consisting of 14000 time steps. The length of the time steps was 0.1 seconds. The first 13000 were used to search for neighbouring points. We used a long correlation interval, i.e. 500. We have predicted 65 timesteps of 0.1 seconds. This corresponds to about 3 drippings of the faucet. An important parameter in the prediction is the value of the embedding dimension. Theoretically, the value for the embedding dimension must be $d_e \geq 2d+1$ (Molenaar[1992], page 26) where d is the dimension of the chaotic attractor. For the dripping faucet d is probably slightly bigger than two. This means that an embedding dimension of 5 seems to be a reasonable choice. In the Figure 13 we give the correlation results for this case.

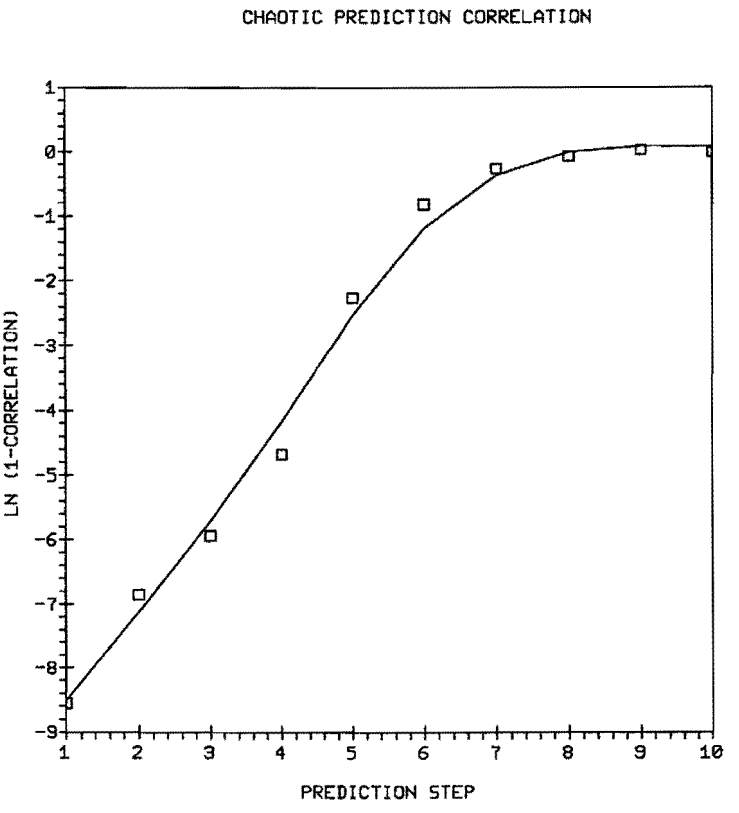

Figure 12: Fitted spline using tent map data

From the logaritmic plot in Figure 14 we have to calculate the initial slope. However, the plot is contaminated by noise. It seems to be that the length of the correlation interval was not long enough but because of memory problems with larger correlation intervals this is the best we can get. We have to smooth the graph to remove the effects of the noise. To do this we fitted a spline through the graph as can be seen in figure 14. Varying the stiffness of the spline, we find $0.09 \leq \lambda_1 \leq 0.33$. The fitted spline shown in the figure corresponds to $\lambda_1 = 0.16$. We like to stress here that using a spline is just one of the methods to remove noise. An estimate of the order of magnitude is the best result we can get.

Remark: The results above are obtained only for the position data of the dripping faucet. We have also made several calculations on the velocity and mass data. However, these results are worse compared to the results of the position data in the sense that larger wiggles were present and that the correlation decreases faster. This can also be seen at the following correlation plots for a series of 1000 points (see Figures 15-17). They denote the correlations as a function of time differences between two points. From the about first 50 points in the correlation plots one can see that the position data is on the average more correlated than the velocity or mass data. Consequently, predictions made by using position data will probably give better estimates.

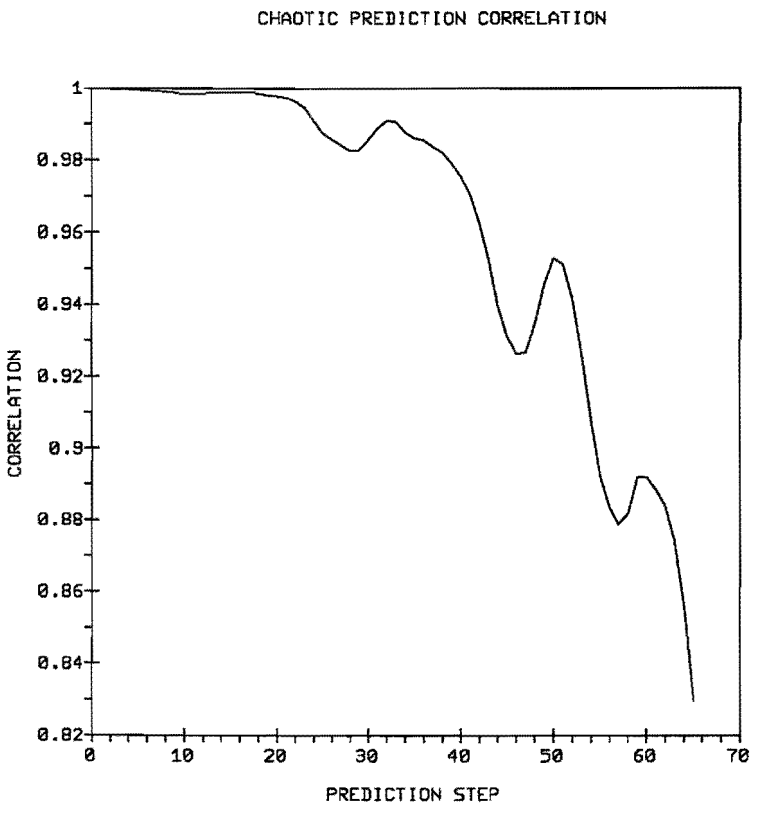

Figure 13: Correlation vs. prediction step using position data

Theoretically, the velocity and mass are proper read-out functions of the model and should provide us the same results.

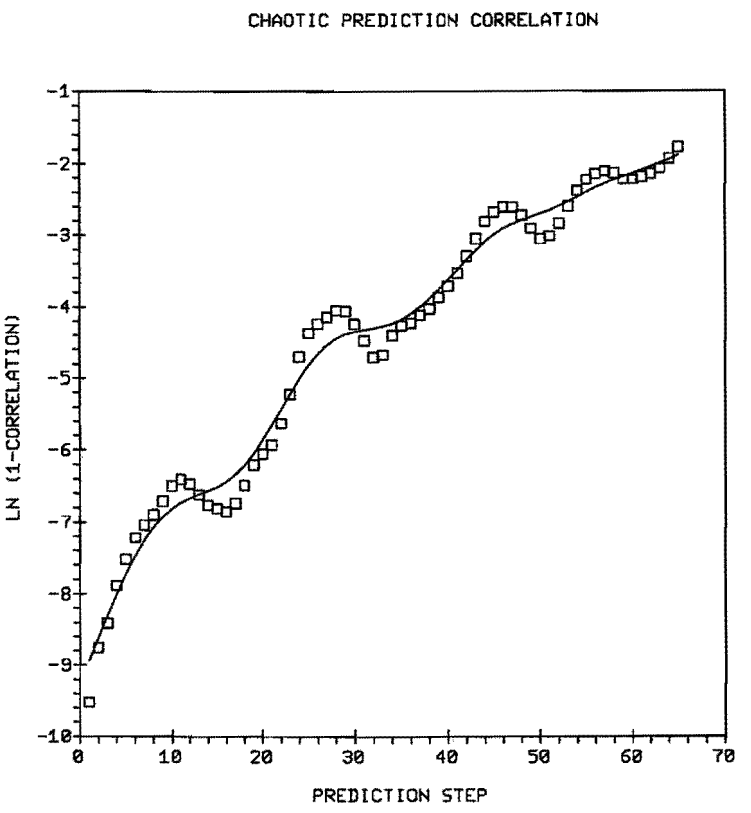

Figure 14: Fitted spline using position data

5 Stochastic Prediction

In the previous section we used a chaotic prediction method. In this section we shall apply a stochastic prediction method. Two programs will be used: Autobox, based on the Box-Jenkins model, and Linpred, based on an autoregressive model. First we introduce the Box-Jenkins model, second we introduce the Linpred predictor and finally we discuss the results obtained with both models.

5.1 The Box-Jenkins method

The B(ox)-J(enkins) method (Box & Jenkins[1976]) is a time series modeling process which describes a variable as a function of its past values (the AutoRegressive part) and the past values of the noise (the Moving Average part). The purpose of the B-J method is to find the linear model (or filter) which describes the underlying structure of the time series best. For the reduced time series (the series from which the effects of the underlying structure is "subtracted") it must hold that no structure is present. In other words, the reduced time series has to be white noise. After finding the model it can be used to forecast

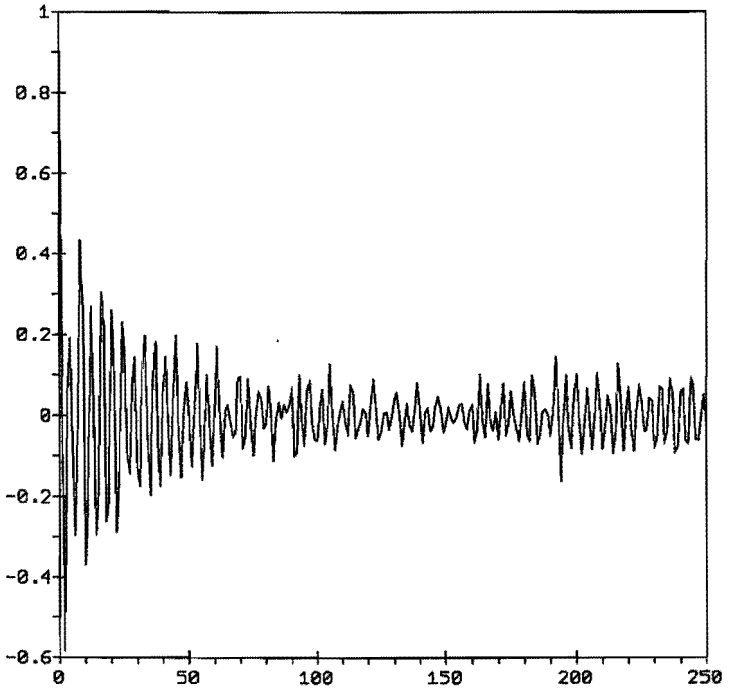

Figure 15: Position correlation plot

future values of the series.

The modeling procedure itself is a three stage iterative process:

- Identification: To make the time series stationary.
- Estimation and Diagnostic checks: To estimate the parameters in the identified model.
- Forecasting: To use the model to generate forecasts for future values of the time series.

Before discussing these points in more detail, it is necessary to introduce some terms. A B-J model, often referred to as an ARIMA-model, can be expressed in the following form:

$$\phi_p(B)w_t = \theta_0 + \theta_q(B)a_t, \tag{9}$$

where

$$w_t = (I - B^c)^d(z_t - \mu) \tag{10}$$

This form defines a so-called ARIMA(p,d,q)-model. The factors are summarized below:

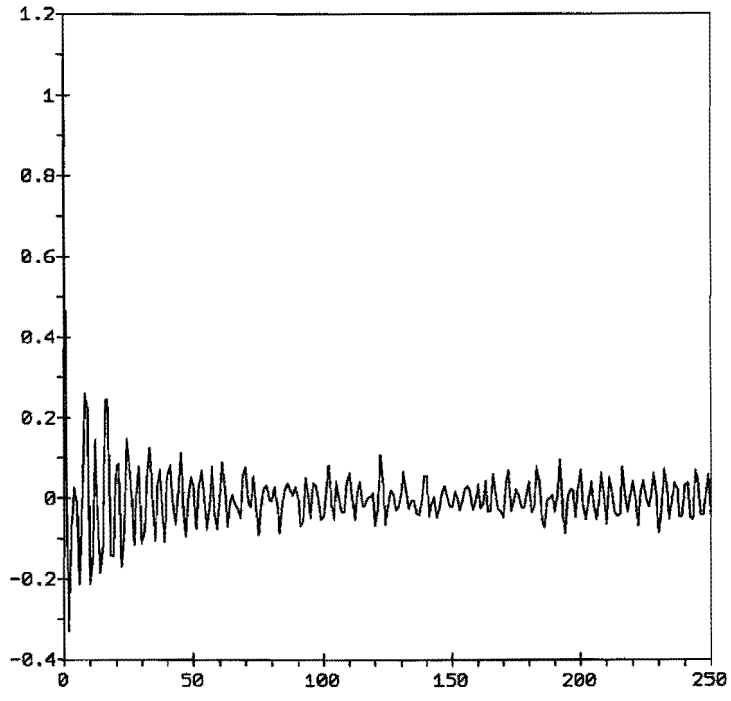

Figure 16: Mass correlation plot

- z_t = the dicrete time series.
- μ = the average value of z_t.
- ϕ_p = the autoregressive factor(s) of order p (AR(p)).
- a_t = the white noise at time step t.
- θ_0 = the deterministic trend (constant).
- θ_q = the moving average factor(s) of order q (MA(q)).
- w_t = the differenced time series of order d and degree c.
- B = the backshift operator, i.e. $Bz_t = z_{t-1}$.

It is possible that the time series includes also some seasonal effects, for instance an annual cyclus. In that case we can have more than one autoregressive, moving average and/or differencing factors. For instance, if we have monthly data and an annual effect, the model can be an $ARIMA_{12}(p_{12}, d_{12}, q_{12})\times ARIMA(p, d, q)$-model.

We will now further explain the various components of the above equation.

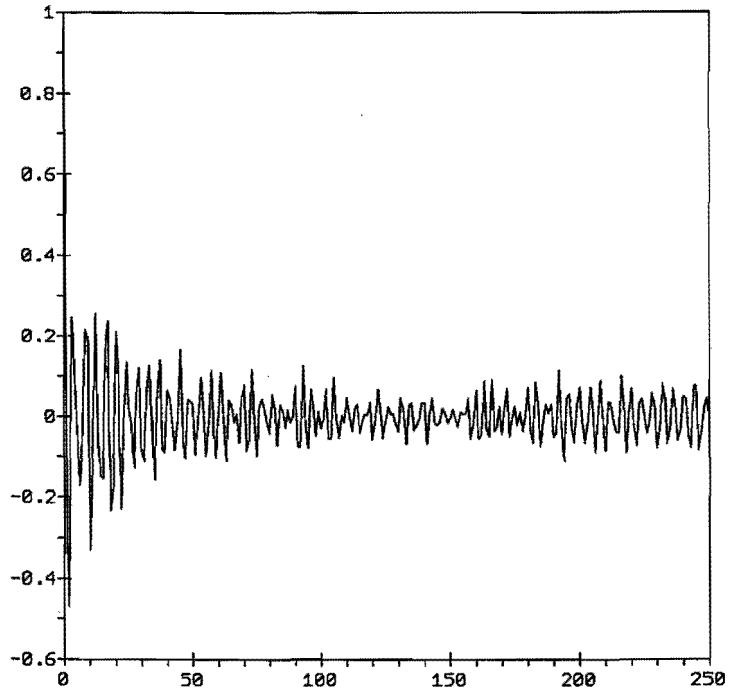

Figure 17: Velocity correlation plot

Differencing factor(s).
A model may have a number of differencing factors. Each differencing factor is a polynomial of the form $(I - B^c)^d$. For a model with one or more differencing factor(s), the expectance of z_t (μ) can be taken to be equal to zero. This remark also applies to the deterministic trend parameter (θ_0).

$\phi_p(B)$) Autoregressive factor(s)
A model may have a number of autoregressive factors. Each autoregressive factor is a polynomial of the form:

$$(1 - \phi_1 B^1 - \phi_2 B^2 - \phi_3 B^3 - ... - \phi_p B^p), \tag{11}$$

where $\phi_1, ..., \phi_p$ are the (possibly vanishing) parameter values of the polynomial.

$\theta_q(B)$ Moving average factor(s)
A model may have a number of moving average factors. Each moving average factor is a polynomial of the form

$$(1 - \theta_1 B^1 - \theta_2 B^2 - \theta_3 B^3 - ... - \theta_q B^q), \tag{12}$$

where $\theta_1, ..., \theta_p$ are the (possibly vanishing) parameter values of the polynomial.

We now describe the three-stage modeling process in more detail.

5.1.1 Stage 1: Model Identification.

In the identification phase the time series is examined in order to choose a tentative model form. There are several key statistical tools used during this phase. The most important of these are the autocorrelations and the partial autocorrelation of the time series.

The first step in identification is to make the time series stationary. In a stationary series, the mean and the variance are constant over time. Differencing makes the mean constant over time. Looking at the autocorrelations of the time series give clues as to the appropriate level of differencing that is required. An autocorrelation function that starts out high (.9 or above) and decays slowly indicates the need for differencing. The order of the differencing is determined by the number of time periods between the relatively high autocorrelations. For example, if the autocorrelation function is high at lags of 12 (for annual data) and dies out very slowly, then this is an indication for the need of differencing of order 12 and degree greater or equal to 1.
Making the variance constant in time can be achieved by a transformation of the data. We will not consider this technique in detail. For the interested reader we recommend the book of Box & Jenkins[1976].

The step succeeding the inducement of stationarity is the tentative identification of the autoregressive/moving average structure. The autocorrelations and the partial autocorrelations of the appropriately differenced and transformed series have patterns which are associated with a particular model form. The analyst can make a conjecture about model form by examining the information in the (partial) autocorrelation functions. In general, decaying autocorrelations and a cut-off in the partial autocorrelation indicate an autoregressive structure and decaying partial autocorrelations and a cut-off in the autocorrelation indicate a moving average structure.

5.1.2 Stage 2: Model Estimation and Diagnostic Checking.

The second stage of the model building process is estimation of the coefficients in the tentatively identified model. This is done by a nonlinear least squares estimation procedure, which is fully described in Box & Jenkins[1976].

There are three basic diagnostic checks that must be performed on the estimated model. These tests are for necessity, invertibility and sufficiency. Each

parameter included in the model should be statistically significant (necessary) and each factor must be invertible. In addition, the residuals from the estimated model should be white noise (model sufficiency).

The test for necessity is performed by examining the t-ratios for the individual parameter estimates. Parameters with nonsignificant estimates should be deleted from the model.

Invertibility is determined by calculating the roots from each factor in the model. All of the roots must lie outside the unit circle. If one of the factors is noninvertible then the model must be adjusted. The appropriate adjustment is dictated by the type of the factor that is noninvertible. For example, a noninvertible autoregressive factor usually indicates under-differencing, while a noninvertible moving average factor may indicate over-differencing or the present of a deterministic factor.

Model sufficiency is tested in the same way as model identification is performed. If there are patterns in the residual autocorrelations and partial autocorrelations of the residuals, then there is the necessity to add parameters to the model.

If all of the parameters are necessary, if each factor is invertible and if the model is sufficient, then the ARIMA model is adequate and it can be used for forecasting.

5.1.3 Stage 3: Model Forecasting.

Model forecasting with the properly identified and estimated model is simply an algebraic process of applying the model form to the actual time series data and computing the forecast values from a given time origin. The confidence intervals give a measure of the uncertainty in the point forecasts.

Remark: Autobox performs the above described steps automatically.

5.2 The Linpred predictor

This program can be probably best explained by comparing it with the Autobox program:

- Linpred calculates only AR-factors and difference factors, whereas Autobox also calculates MA-factors. MA-factors are not that important for

forecasting, because they don not change the forecasts, but only influence the confidence interval of the forecasts.

- Linpred can only calculate one factor in the autoregressive model. This is not a very serious restriction, because seasonal effects can always be captured by choosing p in the AR(p)-model large enough.
- The size of the dataset is restricted to 300 in the Autobox program, whereas Linpred can handle much larger datasets (in our case 13000, with the points 13001 to 14000 used to calculate the forecast correlation).

5.3 Results of Autobox and Linpred

We applied Linpred to a dataset consisting of 14000 position points, with 0.1 seconds between two points and we applied Autobox on subsets of this dataset. On the average there is a drop every 21 points. However, the spreading is quite large, i.e. from 16 to 27.
We start our discussion with the results of Autobox. The model which fulfilled the necessity requirements was an $ARMA(2,0,1)$-model. No differencing was necessary. This was to be expected, because there is no trend in the data of the dripping faucet, i.e. the position is after each drip more or less the same. Moreover, the variance is stationary. The fraction of the data that could be explained by the model was 99% ($R^2 = 0.99$). All the autoregressive factors were invertible. Only 7 autocorrelations and 10 of the residuals remained significant (out of 100 calculated autocorrelations). The residuals were normally distributed with an extremely small probability level. Furthermore, they were close to zero except for some outliers which had an average lag between them of 21. This equals the average time between two drops and as a result it could be removed by extending the model with a ARIMA-component with lag 21. However, this showed not to be significant. The explanation for this could be that the series was not long enough (only 300 points). Furthermore, the lag is on the average 21, but ranges from 16 to 27. The influence of this phenomenon can also be seen in Figure 18, which represents an illustrative part (lag 241 to 300) of the plot of the fit (B) versus the actual values (A), where X denotes the case when the fit and the actual data point coincide.

It can be seen that the inaccuracies occur in the neighborhood of the jump (the low values).
The problem discussed above has important influence on the accuracy of the forecasts as is shown in Figure 19.

The results get even worse after timestep 320. Figure 20 also suggests a poor

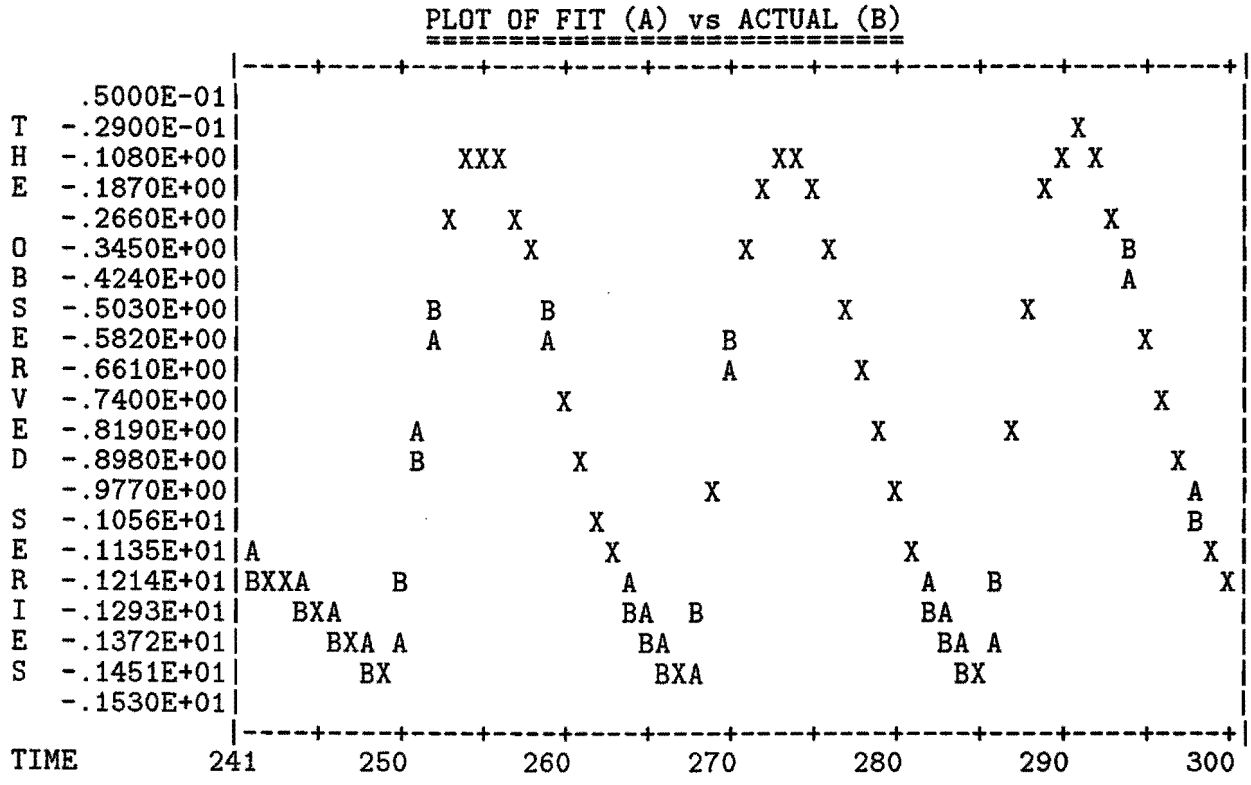
PLOT OF FIT (A) vs ACTUAL (B)
.5000E-01
T -.2900E-01
H -.1080E+00
E -.1870E+00
-.2660E+00
O -.3450E+00
B -.4240E+00
S -.5030E+00
E -.5820E+00
R -.6610E+00
V -.7400E+00
E -.8190E+00
D -.8980E+00
-.9770E+00
S -.1056E+01
E -.1135E+01
R -.1214E+01
I -.1293E+01
E -.1372E+01
S -.1451E+01
-.1530E+01
TIME
241
250
260
270
280
290
300

Figure 18:

TIME PERIOD	FORECAST VALUE	ACTUAL VALUE	ERROR	% ERROR
301	-.12934E+01	-.13218E+01	-.28447E-01	2.15
302	-.12798E+01	-.13790E+01	-.99198E-01	7.19
303	-.12143E+01	-.14197E+01	-.20546E+00	14.47
304	-.11118E+01	-.14479E+01	-.33608E+00	23.21
305	-.98976E+00	-.14674E+01	-.47763E+00	32.55
306	-.86549E+00	-.14457E+01	-.58019E+00	40.13
307	-.75426E+00	-.13423E+01	-.58807E+00	43.81
308	-.66769E+00	-.11947E+01	-.52700E+00	44.11
309	-.61283E+00	-.10348E+01	-.42198E+00	40.78
310	-.59197E+00	-.88705E+00	-.29508E+00	33.27
311	-.60298E+00	-.76825E+00	-.16527E+00	21.51
312	-.64018E+00	-.68849E+00	-.48310E-01	7.02
313	-.69553E+00	-.65220E+00	.43327E-01	6.64
314	-.75988E+00	-.65932E+00	.10056E+00	15.25
315	-.82427E+00	-.70650E+00	.11777E+00	16.67
316	-.88093E+00	-.78813E+00	.92802E-01	11.77
317	-.92410E+00	-.89730E+00	.26803E-01	2.99
318	-.95043E+00	-.10266E+01	-.76124E-01	7.42
319	-.95903E+00	-.11685E+01	-.20944E+00	17.92
320	-.95130E+00	-.13162E+01	-.36486E+00	27.72

Figure 19: Forecasts made with Autobox

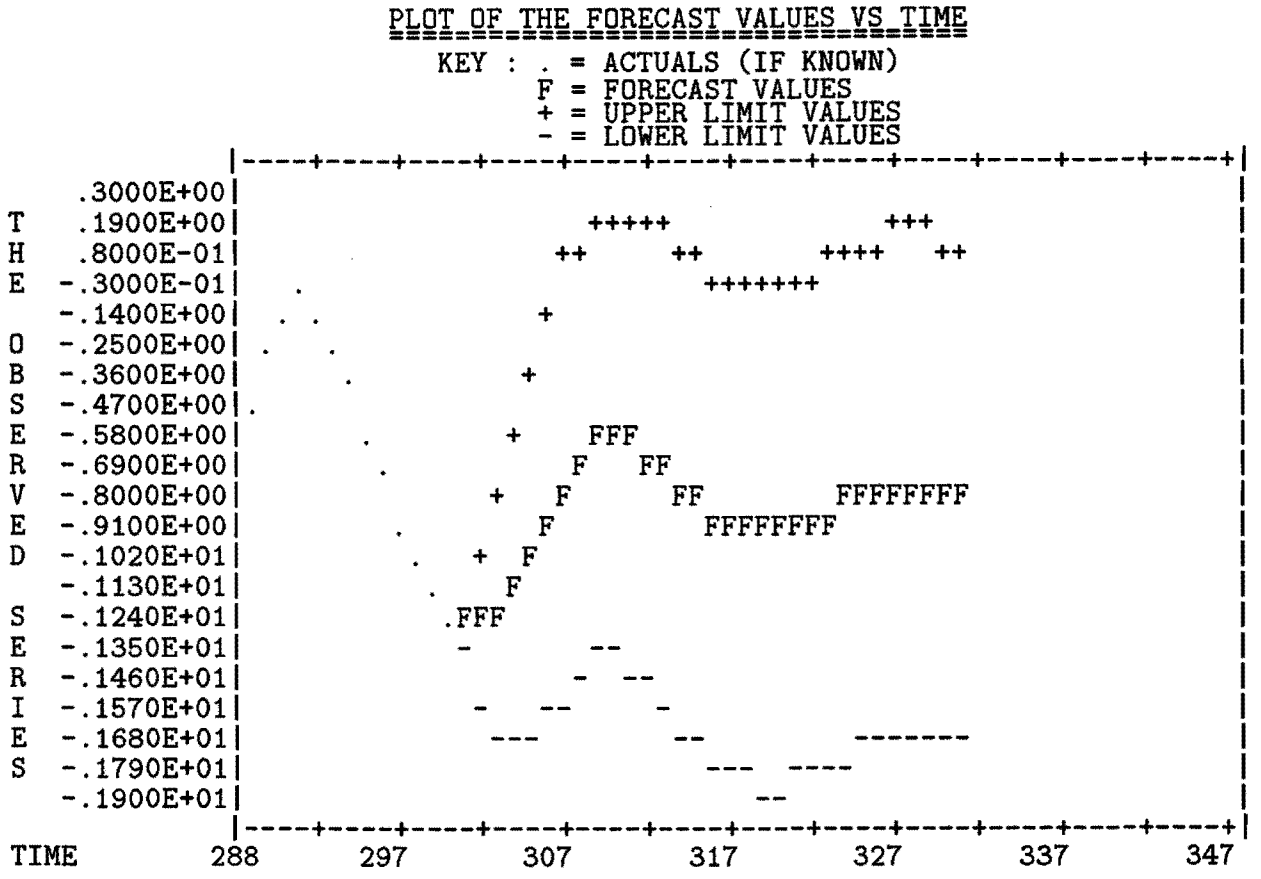

Figure 20: Forecasts made with Autobox

prediction capacity.

This poor prediction ability is probably due to the jump. This causes that the MA-factor is very important (the t-ratio is about 11, whereas to be included in the model it has to be at least 2). As discussed before, the MA-factor has no influence on the predicted value, but more on the width of the confidence interval.

As can be expected, the conclusions drawn from the Linpred-results do not differ very much from the conclusions drawn from the Autobox-model. Figure 21 and 22 obtain the results of two models, an AR(2)-model and an $AR(25)$-model. The first is similar to the model found above, except for the MA-factor (which is not important for the predicted values). The second is a result of a closer examination of the $AR(2)$-plot; we can see that there seems to be a period of about 21 in the series. This was also noted in the discussion of the

Autobox results. An $AR(2) \times AR_{21}(p)$ seems to be useful. This can be partially simulated by an $AR(25)$-model. The models are equal if the 3^{rd} until the 20^{th} and the 22^{th} until the 25^{th} coefficients in the $AR(25)$-model are zero. The resulting graph from the $AR(25)$-model is given in figure 22. The period of 21 seems to be removed. Moreover, the accuracy of the prediction is improved, but still much worse than that achieved using the chaotic prediction method and given in Figure 13.

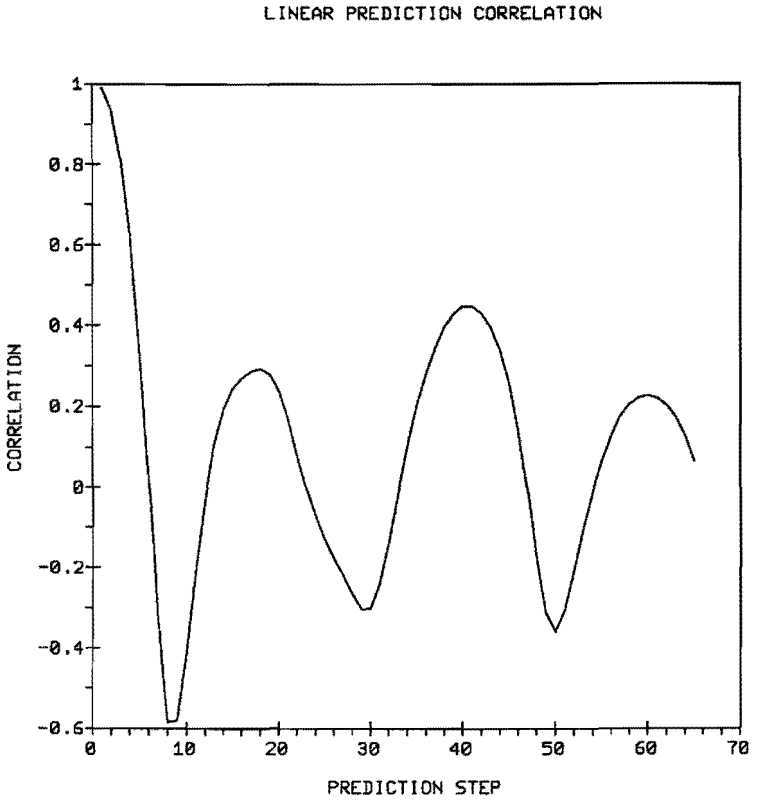

Figure 21: AR(2)-model

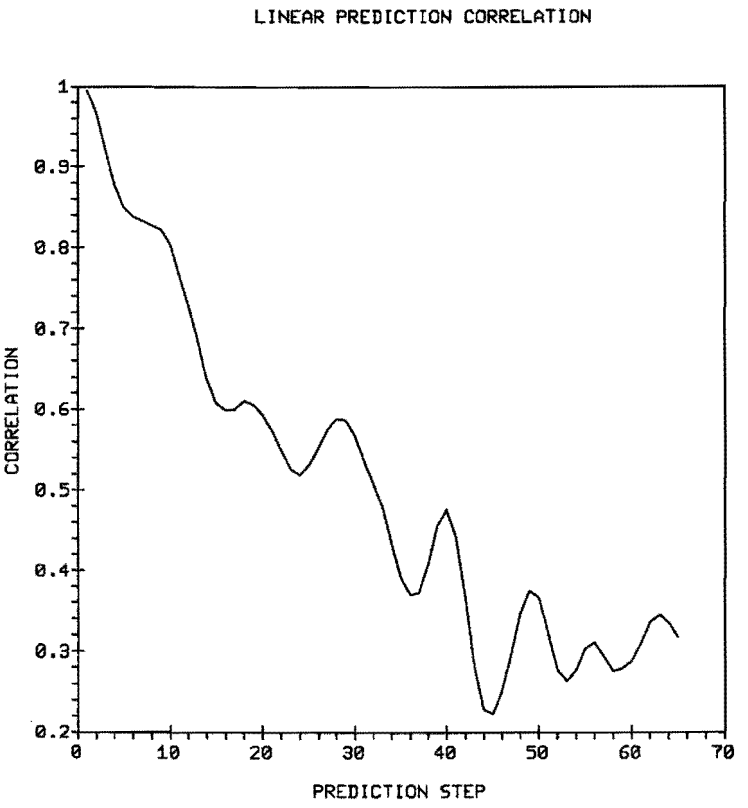

Figure 22: AR(25)-model

6 Towards chaotic behaviour of the dripping faucet

Until now, in this project we have always set the flow velocity β of the faucet equal to 0.6. It is known that the system is chaotic for this value. For small flow velocities no chaos is present. Here, we want to find out how the system becomes chaotic as a function of parameter β.

For a certain flow velocity we calculate values of the times between two drips $T_n, n = 1, 2, ...$ using the program DRIPS. From the plot of T_n versus T_{n+1} we can see if the system is chaotic or not for this flow velocity. If the plotted points are centered at n points this means that the system has period n. Otherwise the system is probably chaotic.

For 25 values for β in the interval $[0.5 - 0.6]$ we followed this procedure. We calculated 2000 points with a time step 0.2 and skipped the first 1000 points. This is to avoid a possible transient period. So, we obtained about 85 points of times between jumps. The results of the period for these values are given in Figure 23. For practical reasons chaotic behavior is indicated by a "0" although it corresponds to a period ∞. .

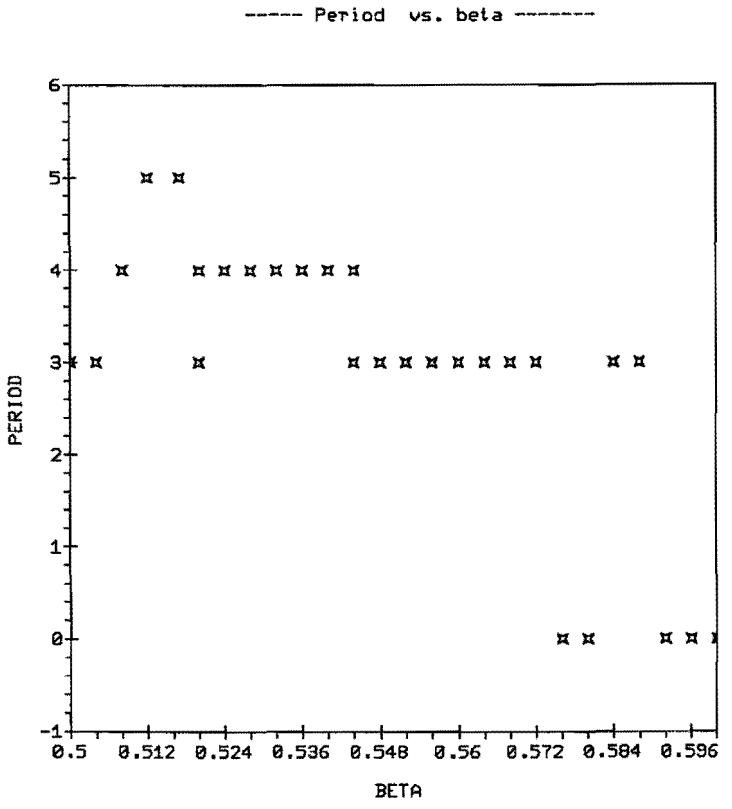

Figure 23:

We see that there is no period doubling as in the logistic map. For some values it is difficult to say if the period is equal to 3 or 4 (but that is not very important). Remarkable is that at the end (β between 0.576 and 0.6) the system is first chaotic, then periodic, and afterwards chaotic again. In the interval $[0.588 - 0.592]$ we made 10 runs to see what happens there. The result is shown in Figure 24. For β increasing from 0.590 it suddenly changes from periodic to chaotic behavior. The intervals can be made smaller and smaller but we did not do this because it takes too much computing time. In this run we took 5000 points of which we skipped the first 4000 points because the transient can be very long. For this last run the calculation took about 50 minutes for each β value.

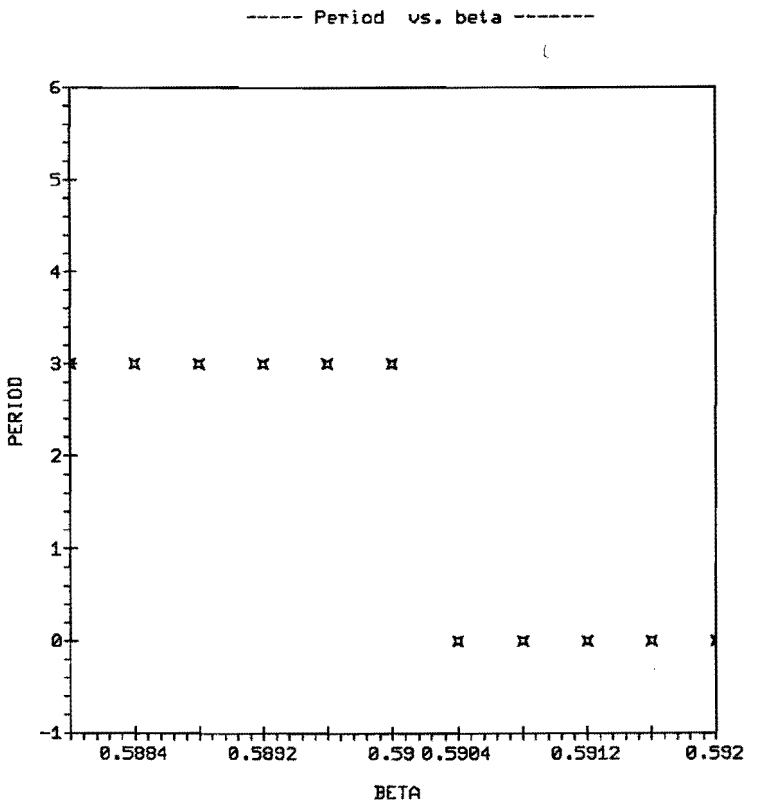

Figure 24:

7 Conclusions and recommendations

In this report we described several methods to calculate the Lyapunov exponent of the model of the dripping faucet. The outcomes are presented in Table 11.

method	Lyapunov exponent	lower bound	upperbound
WOLF package	0.23	0.20	0.25
Direct	0.24	0.22	0.26
Prediction	0.16	0.09	0.33

Table 11: Best values for the Lyapunov exponent

From this Table we may conclude that the largest Lyapunov exponent of the model of the dripping faucet is about 0.24. Consequently the dripping faucet is a weakly chaotic system.

It is hard to compare the used methods with each other. Looking at the length of the interval of estimates it seems that the direct method gives the

best results, followed by the Wolf package.

The following remarks should be made about the several methods:

Wolf package

It seems that the results depend on the used readout function. Using the time series of the dripping intervals or the mass series does not give reliable results. The reason probably is the discontinuity of these series. The position and velocity series give better results. This does not match with the theory which says that every function of the state variables can be used as a readout function (Takens[1981]).

Another aspect found is that the embedding dimension used for reconstruction should be bigger than 3.

Direct method

Since the attractor is not two-dimensional, it is not useful to order the points in the plain of the jump. A two-dimensional set can not be ordered like a line. So the results of the sorting gave only the hint to analyse the attractor more exactly, but have nothing to do with the largest Lyapunov exponent.

The method takes into account the behaviour of the system from its description by differential equations. So this must be the most exact method to determine the largest Lyapunov exponent, because it uses all information about the system. In practice it often happens that the exact description is not known, only a time series is given. Then the direct method is not applicable.

Prediction method

By the prediction method it is also found that the results depend on the used readout function.

The method of Wales produces estimates of the Lyapunov exponent which lie in the largest interval of the 3 methods. The reason could be that the time series were not long enough. This can be seen from the correlation plot, which still contains wiggles. Longer time series, however, introduce computability problems. Wiggles were not present in the correlation plot of the tent map. For this map the value found for the Lyapunov exponent coincides with the value in Wales[1991] and with the theoretical value.

We also applied a stochastic description. The underlying statistical model seems to be an $ARIMA(2,0,1)$-model (AUTOBOX) or perhaps an $ARIMA(2,0,1) \times ARIMA_{21}(2,0,1)$-model. In both cases, the predictions are bad. Probably due to the fact that the jumps in the data have a large influence on the

predictions.

If the flow velocity is equal to 0.5 the system is periodic. For a flow velocity equal to 0.6 it is chaotic. In the interval in between the system is sometimes periodic and sometimes chaotic. No period doubling has been observed, as is seen for the well-known logistic map.

In future research it would be interesting to look at the influence of the readout function. As we have seen there are big differences by using the position series or, for example, the series of times between jumps.

Another recommendation is that the stochastic model could be extended to a model which can handle jumps (see Box & Jenkins[1976]).

References

[*] G.E.P. Box & G.M. Jenkins, *Time series analysis: forecasting and control*, Holden-Day, San Fransisco, 1976.

[*] J. Molenaar, *Chaos theory and time series*, Report IWDE 92-05 T.U.E. Eindhoven, 1992.

[*] A. Noack, *Der tropfende Wasserhahn - ein Beispiel für ein System mit chaotischem Verhalten*, ISAM'92 Nonlinear Dynamics: Attractor Approximation and Global Behaviour, pp. 167-178, 1992

[*] N.H. Packard, J.P. Crutchfield, J.D. Farmer, R.S. Shaw, *Geometry from the time series*, Phys. Rev. Lett., vol. 45, pp. 712, 1980.

[*] R.S. Shaw, *The dripping faucet as a model chaotic system*, Aerial Press, Santa Cruz, 1984.

[*] G. Sugihara & R.M. May, *Nonlinear forecasting as a way of distinguishing chaos from measurement error in time series*, Nature, vol. 344, pp. 734-741, 1990.

[*] F. Takens, *Detecting strange attractors in turbulence*, in: Dynamical systems and turbulence, Warwick. Lecture Notes in Math 898 (Springer, Berlin, 1981), pp. 336-381.

[*] D.J. Wales, *Calculating the rate of loss of information from chaotic time series by forecasting*, Nature, vol. 350, pp. 485-488, 1991.

[*] A. Wolf et al, *Determining Lyapunov exponents from a time series*, Physica 16D, pp. 285-317, 1985.