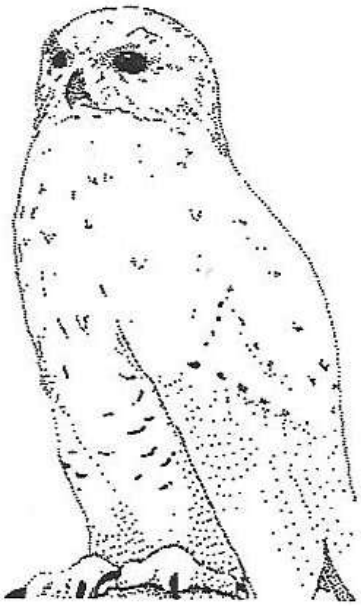
How to spot



order in

financial markets

A short course in applied chaos theory

by Hans Uhlig

Instruction manual - Contents

Overview

What you can expect from this manual

Here we will show, how you can make practical use of chaos theory. Applied to financial markets it allows to give much better prognoses and consistently so than forecasts based on conventional linear statistics, like auto regression of moving averages (ARMA), which are the basis of 'technical analysis'.

This guide is devided into four parts. The first part is a very brief introduction into the world of chaos. We will explain why and how the discovery of deterministic chaos has changed the modern data analysis procedures, the concepts of modelling dynamic systems, and our attitude towards long term predictability of events.

The second part will lead you through the whole process of data analysis and modelling using the tools developed over the years by physicists and mathematicians in the field of chaos research. However, we will not treat chaotic model systems here, like the logistic equation or the Mackey-Glass functions. Instead we will apply the theory to real market data. Our working example will be the Dow Jones Industrial Average.

The third part is a course in how to establish and use a 'nearest neighbour' predictor, which unlike many other predictors uses only local information for its predictions. What is meant by local will be explained more thoroughly below.

Finally we have a short discussion of the chaos theoretical approach as compared to traditional methods, like fundamental analysis and 'technical analysis'. We will focus on the main differences between these three.

There is an appendix which lists the markets for which data are available and the sources for these data

Part I - THE WORLD OF CHAOS

Random walks and chaotic trajectories

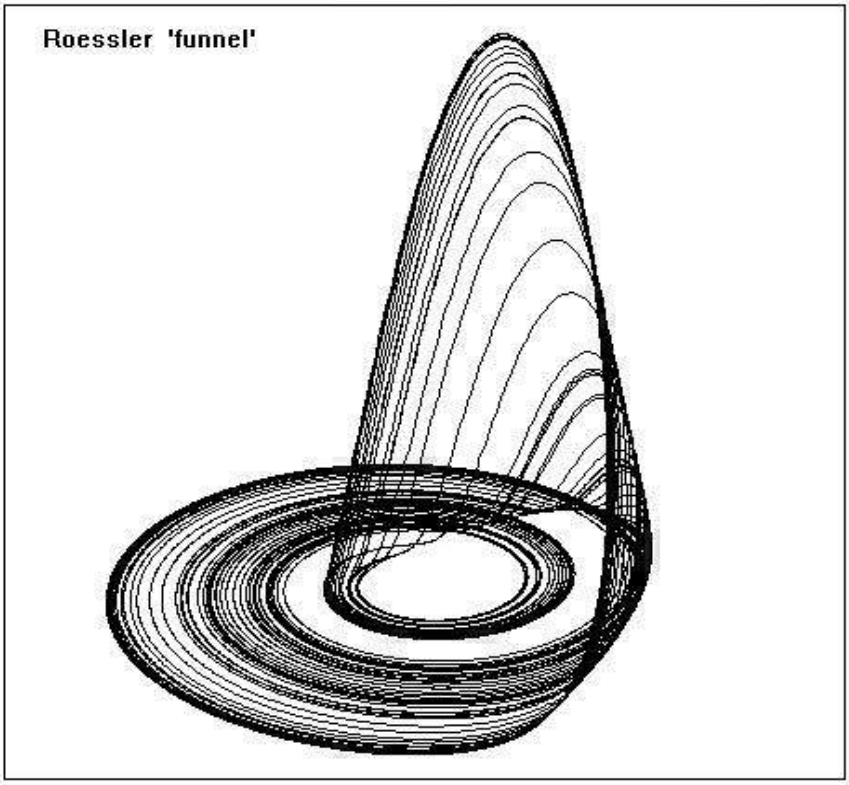
Statistical forecasts rely on probabilities. They are ideal in cases where single future events are unpredictable and only probabilities can be given for the various possible outcomes. In other words they are ideal for estimating random events.

For a long time market movements were regarded as random events, and this is still a popular opinion. Hence the success of the 'random walk' theory, which has many followers among scholars of economics. This theory was suggested in the nineteen sixties following statistical analysis of empirical market data. And the statements of this theory were supported by similar analyses many times later on.

Then, what made people change their mind concerning the 'random walk' theory? Basically it was the discovery of deterministic chaos.

Physicists and mathematicians recognized that quite simple dynamic systems made up of only three components could show seemingly unpredictable behaviour: Chaos. And this could occur although interrelations between the components were strictly coded in mathematical equations, which did not involve any random element. Not all coupled three component systems were capable of evolving this kind of behaviour. A necessary condition for chaos to appear was that at least one of the three interrelations was nonlinear.

A prominant example of a chaotic system with a single non linear coupling is the Roessler 'funnel', shown in the figure. Below we give the formulas which describe how the three components of the system change with time. Note that a,b,c are constant values. Non linearity arises in the third component from the product of two variables (z*x).



Roessler model

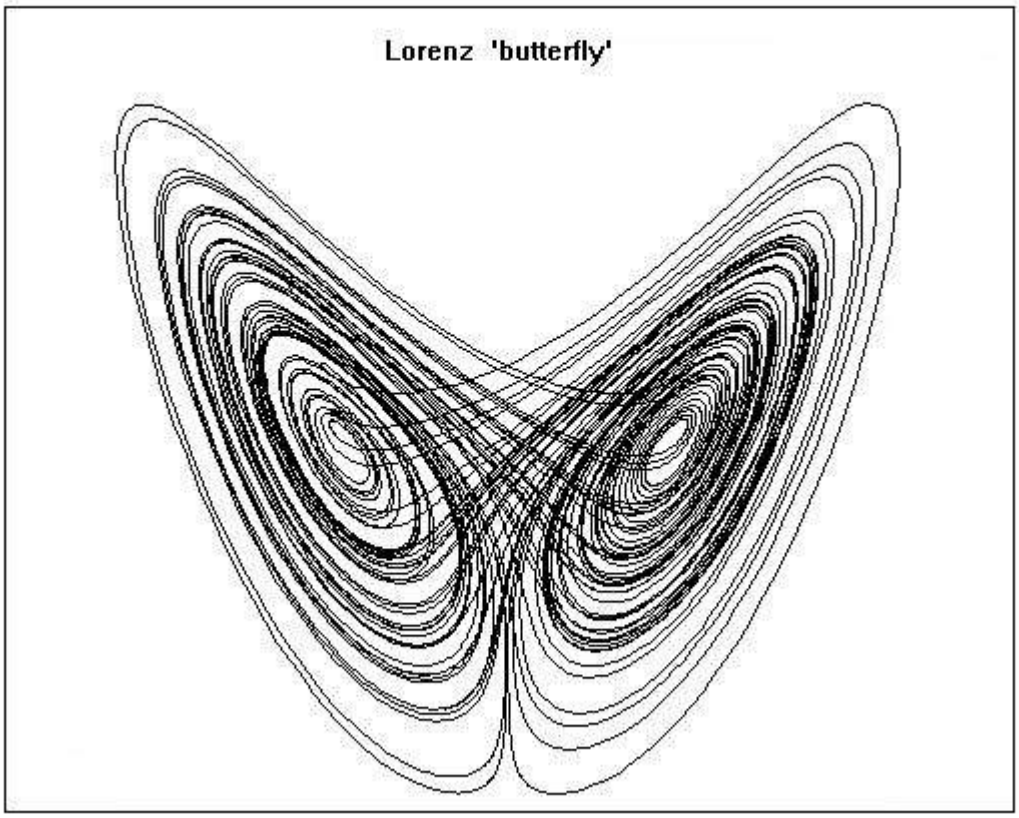
$$\frac{dx}{dt} = -y - z$$

$$\frac{dy}{dt} = x + a * y$$

$$\left(\frac{dz}{dt} = b + z * [x - c] \right)$$ non linear term

a, b, c = constants

Better known to the public is the Lorenz 'butterfly', see figure, which is also a three component system, but has two nonlinear interrelations, one more than the Roessler 'funnel'. Again we give the formulas below. They describe how the three components of the system change with time. Note that here b,r and sigma are constant values. Non linearity arises in the second component from the product (x*z) of two variables and in the third component from another product of two variables (x*y).



Lorenz model

$$\frac{dx}{dt} = \text{sigma} * (y - x)$$

$$\left.\begin{aligned} \frac{dy}{dt} &= x * (r - z) - y \\ \frac{dz}{dt} &= x * y - b * z \end{aligned}\right\} \text{non linear terms}$$

b, r, sigma = constants

Chaotic time series (trajectories) look like random events not only to the naked eye. Linear statistical tests for auto correlation or Fourier analysis cannot recognize differences between chaotic and random data and neither can wavelet analysis. This observation indicated that tests of randomness were not reliable if carried out with these methods. And since these tests have been applied to a broad range of problems they could probably have misclassified events as random which really were not.

Fortunately tests are available which can unambiguously distinguish ordered events from randomness, for example the well known Chi² - test of independence and the somewhat less known test of conditional entropy. Using these tests we have investigated data from many markets and compared them to matched pseudo-random data. While the pseudo-random data could not be distinguished from random events, all the original market data were at least significantly ($p > 0.95$ %) different from random events, most differences were even highly significant ($p > 99\%$). These results suggest that the 'random walk' hypothesis should be rejected. Markets behave reflexively in other words they are self referential systems. But beware, their dynamics are not as simple as practitioners of 'technical analysis' assume.

If market movements are not random events then there exist better methods than ARMA to forecast the future development. But before we can make use of these methods we have to discover them. This will be dealt with in the next part.

Part II - DATA ANALYSIS AND MODELLING

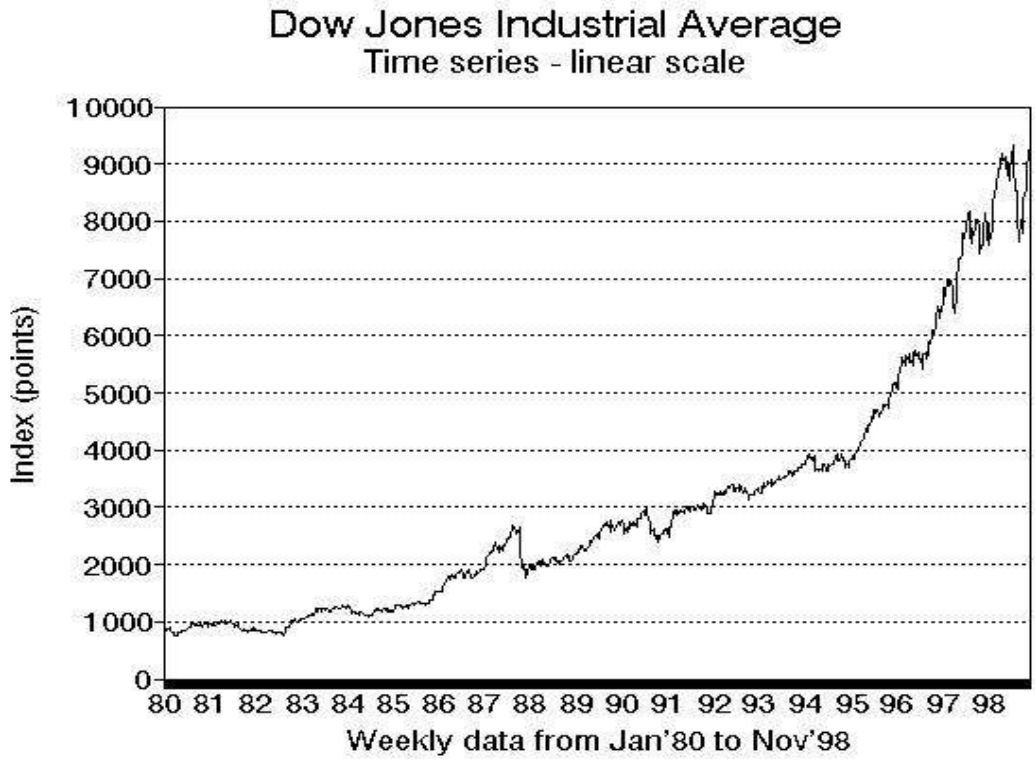
In the following we will develop a model of the DOW JONES industrial average as a dynamical system. The model will be created on the basis of a time series of weekly closes from January 1980 to November 1998.

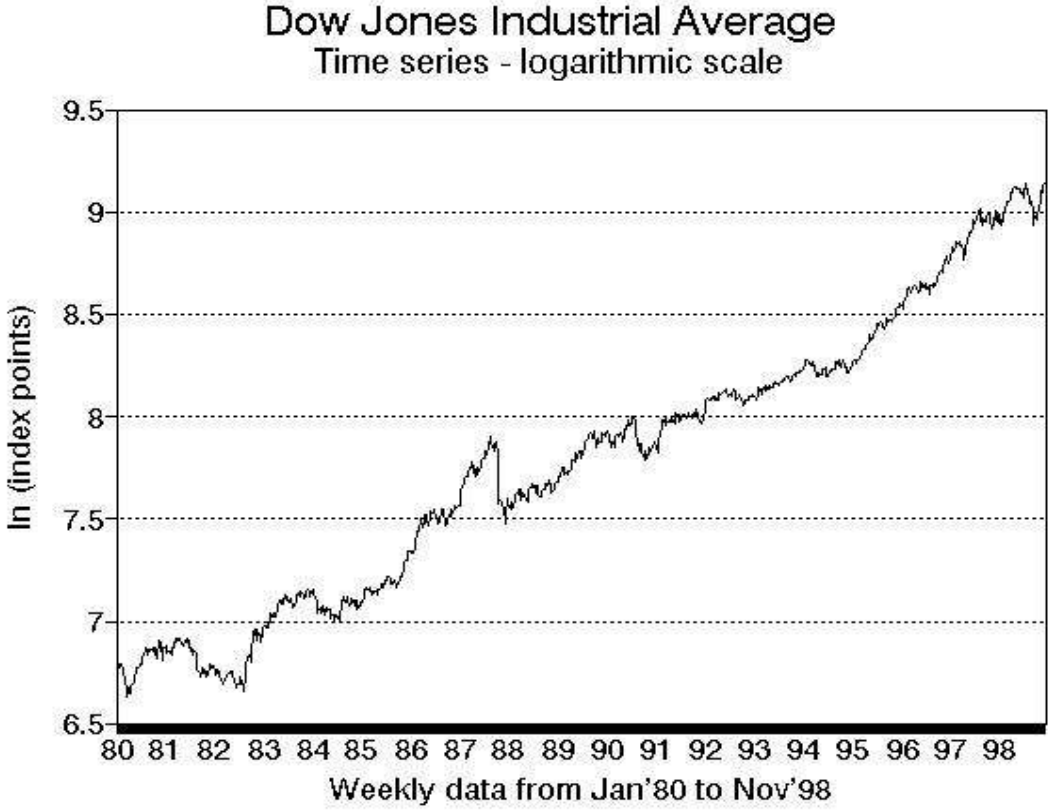
In principle the same type of analysis can be carried out with other markets. Of course one must not blindly take over all parameters found suitable for the DOW JONES index to the modelling of other markets. But despite of the fact that the individual conditions can be quite different from one market to another, the methods learned from the DOW JONES example can be applied to any market. For those who would like to try some experiments with other markets, a disk with data from various markets will be supplied together with this manual. More information concerning the data can be found in the appendix.

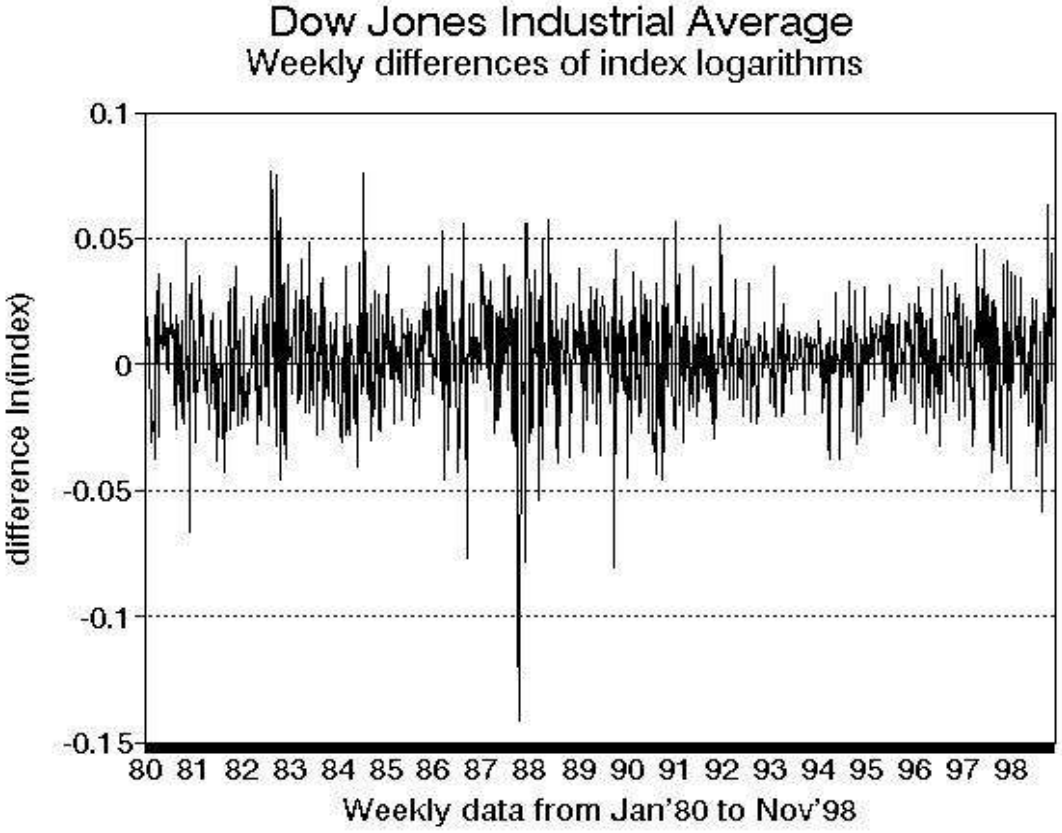
Data preprocessing

Detrending the data

The original data series has a strong upward trend. This is shown in the two figures, one of which uses a linear vertical scale, the other a logarithmic scale. With logarithmic scaling the trend fits much better to a straight line than with linear scaling. Anyway, the trend must be eliminated first. One way to achieve this is to build the logarithm of the data and then take the differences from week to week. Written down as a series of weekly differences of their logarithms the data are effectively detrended as you can learn from the corresponding figure. Note that we use natural logarithms (ln) here and not log10.




Dow Jones Industrial Average
Time series - logarithmic scale
ln (index points)
9.5
9
8.5
8
7.5
7
6.5
80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98
Weekly data from Jan'80 to Nov'98


Dow Jones Industrial Average
Weekly differences of index logarithms
difference ln(index)
0.1
0.05
0
-0.05
-0.1
-0.15
80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98
Weekly data from Jan'80 to Nov'98

Measuring the entropy

There is nothing particular about one week differences. Other (moving) differences between two or more weeks are also possible. One can easily show that the one, two and three week differences for example are distributed differently. The more uniform the distribution the greater our uncertainty would be to forecast a particular outcome. A measure for this uncertainty is the entropy. Our uncertainty obviously depends on the precision required. It is much easier to say if the next move is either up or down, which are two possible outcomes, a one bit decision or to classify the next move into one of four possible outcomes, a two bit decision. The entropy is usually given as uncertainty/bit. It can aquire values between 0 and 1. The entropy is minimum (zero) if one event is certain and all others are impossible. The entropy is maximum (one) if all events are equally likely. This is true only for uniform distributions. Normally distributed events have a lower entropy. The entropy usually rises with increasing resolution.

We have developed a computer program ENTROPRE.EXE that calculates the entropies for the various differences between one week and fifty two weeks (a year). The program will write the values as a table on the screen. You can easily obtain a hard copy with your printer if you give the PRINTSCREEN command. Usually there is a key named such on your keyboard. The program works with the original data and automatically calculates the logarithms and differences.

The program ENTROPRE.EXE cannot work with data in spreadsheet format, but requires an ASCII type data file. In order to put the program to work, the data column from the spreadsheet has to be saved as an ASCII file to a directory that holds the program ENTROPRE.EXE. More details are given in the user instructions for analysis programs.

```
Entropy time profile (weeks)  - djiaorig.prn

Kolmogoroff-Sinai entropy [S]: 32 bins = 5 bits,  987 data
-------------------------------------------------------------------------------
week  [ 1]  S = 3.600    week  [ 2]  S = 3.253    week  [ 3]  S = 3.267
week  [ 4]  S = 3.550    week  [ 5]  S = 3.601    week  [ 6]  S = 3.685
week  [ 7]  S = 3.779    week  [ 8]  S = 3.682    week  [ 9]  S = 3.545
week  [10]  S = 3.606    week  [11]  S = 3.762    week  [12]  S = 3.667
week  [13]  S = 3.737    week  [14]  S = 3.794    week  [15]  S = 3.829
week  [16]  S = 3.850    week  [17]  S = 3.975    week  [18]  S = 4.052
week  [19]  S = 4.139    week  [20]  S = 4.099    week  [21]  S = 4.117
week  [22]  S = 4.131    week  [23]  S = 4.168    week  [24]  S = 4.211
week  [25]  S = 4.244    week  [26]  S = 4.315    week  [27]  S = 4.363
week  [28]  S = 4.352    week  [29]  S = 4.337    week  [30]  S = 4.385
week  [31]  S = 4.365    week  [32]  S = 4.339    week  [33]  S = 4.418
week  [34]  S = 4.462    week  [35]  S = 4.376    week  [36]  S = 4.378
week  [37]  S = 4.330    week  [38]  S = 4.349    week  [39]  S = 4.332
week  [40]  S = 4.375    week  [41]  S = 4.375    week  [42]  S = 4.467
week  [43]  S = 4.533    week  [44]  S = 4.493    week  [45]  S = 4.510
week  [46]  S = 4.540    week  [47]  S = 4.569    week  [48]  S = 4.591
week  [49]  S = 4.578    week  [50]  S = 4.622    week  [51]  S = 4.615
week  [52]  S = 4.573
```

tau	Chi²	Error Prob.	Cramer-V	tau	Chi²	Error Prob.	Cramer-V
0	12818.00	0.00000000	1.0000	1	1730.12	0.00000000	0.3676
2	685.40	0.00000000	0.2315	3	148.23	0.87345164	0.1077
4	213.40	0.01171461	0.1293	5	152.75	0.80982449	0.1094
6	549.10	0.00000000	0.2076	7	683.20	0.00000000	0.2317
8	448.44	0.00000000	0.1878	9	254.41	0.00002352	0.1415
10	255.56	0.00001913	0.1419	11	154.68	0.77806888	0.1105
12	155.98	0.75516140	0.1110	13	123.57	0.99651800	0.0988
14	145.28	0.90645265	0.1072	15	142.72	0.92980744	0.1063
16	240.89	0.00023457	0.1382	17	228.60	0.00154175	0.1347
18	193.59	0.09454085	0.1240	19	119.32	0.99859199	0.0974
20	140.43	0.94682868	0.1057	21	144.74	0.91175879	0.1074
22	360.12	0.00000000	0.1695	23	351.20	0.00000000	0.1675
24	303.02	0.00000000	0.1557	25	208.39	0.02111110	0.1292
26	242.02	0.00019519	0.1393	27	117.22	0.99913415	0.0970
28	116.40	0.99928766	0.0967	29	137.48	0.96385740	0.1051
30	133.02	0.98120066	0.1035	31	123.77	0.99637111	0.0998
32	196.52	0.07239121	0.1259	33	327.55	0.00000000	0.1626
34	260.95	0.00000714	0.1452	35	108.42	0.99991577	0.0936
36	141.40	0.93998806	0.1070	37	117.62	0.99904872	0.0976
38	128.98	0.99035982	0.1023	39	189.84	0.13013641	0.1242
40	305.16	0.00000000	0.1575	41	158.32	0.71121887	0.1135
42	139.95	0.94995105	0.1068	43	145.90	0.90008961	0.1091

distance=1 n_bins=16 degrees of freedom=169 n_data=942 djiaorig.prn

What can we learn from that table? First of all we will observe that there are considerable differences over time. Second we can see that there is no monotonous increase or decrease. Though there is a tendency to higher values with time there can be islands of somewhat lower entropy anywhere. Often there are such islands around thirteen weeks difference. For the DOW there is an island at 12 weeks. So this could be a first rough estimate of a suitable reference time.

We can also analyse the time series with another program, called CHISQUAR.EXE which will perform a Chi² test of independence. This test could give us further hints about a suitable reference time. If we use the program to look at log(index) differences for 1 time step as shown in figure 5, it will show that the individual differences are significantly associated somehow over a period of ten weeks, which is over in the eleventh week. This is about the same time span as found above. We will meet the number eleven again later, when we look at the conditional entropy and mutual information.

Looking for the right reference time

If we look at the time series of twelve weeks differences of logarithms of indices,

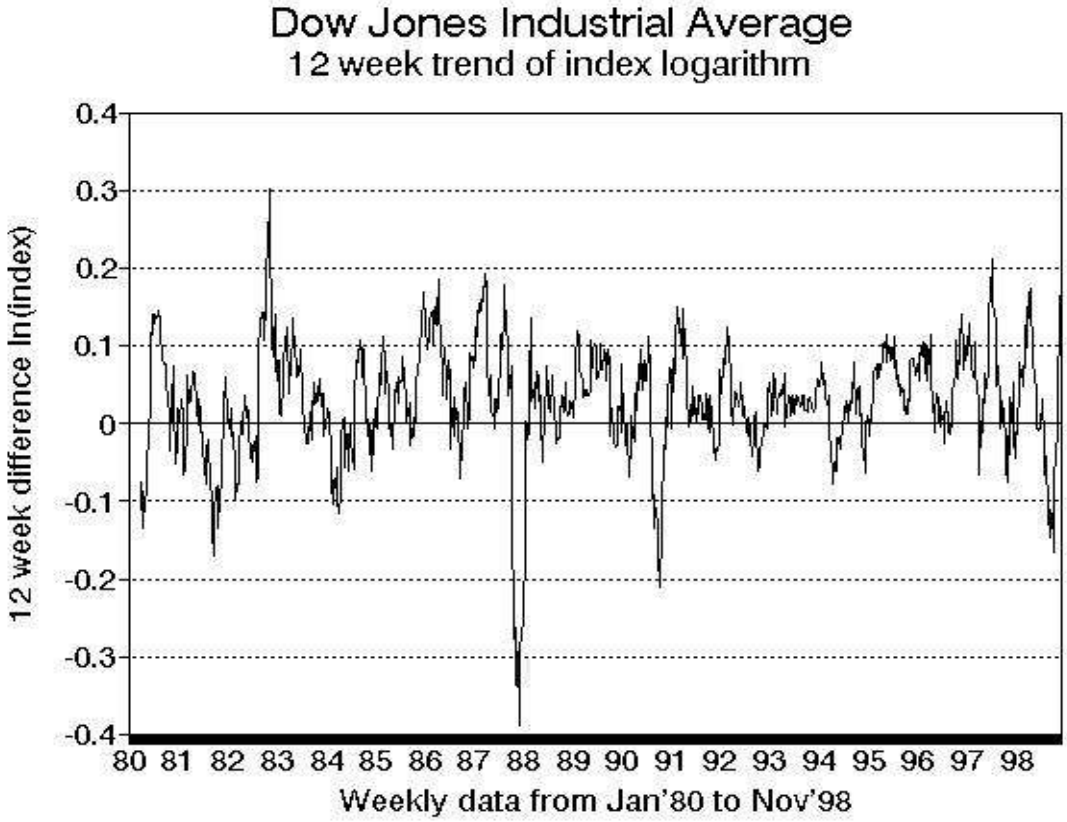
@LN(B10)-@LN(B22)....@LN(B984)-@LN(bB96)

there is much variation, which makes the curve bumpy. This variation can be reduced considerably if we do not compare the actual value to a single value in the past, but to an average over several weeks. We have observed that seven weeks averages in the past are quite good as a point to start from. The optimum may be a somewhat smaller range. This would mean that instead of calculating the twelve weeks difference we would calculate the difference to an average taken over the range of nine to fifteen weeks.

@LN(B10)-@LN(@AVG(B20..B26))...
@LN(B980)-LN(@AVG(B990..B996)).

The values we thus obtain are market trend values for the past quarter. But what is the true trend, if the values are moving up and down from one week to the next. We have some reason to assume that the true trend would keep its direction. We can approach the true trend if we minimize the changes in direction. An easy way to do this, is smoothing of the data with a simple four weeks moving average. Other averages are also possible, i.e. exponential, or stepwise linear. They are mentioned here only for completeness, but will not be discussed any further. All averages named so far would be calculated from complete, not interrupted, series of values and this causes trouble for prognoses. If we give a four week prognosis of a four weeks average, we cannot back calculate the individual value from the average. It is impossible, since we do not know the values of one, two and three weeks ahead. Therefore we have to be content with prognosing averages. This is no disadvantage if compared to ARMA procedures, because these would also give us averages. And the averages correlate well with the unsmoothed trends. The highest correlation is reached with one or two time steps delay at least with simple averages. In principle we could, back calculate individual values if we computed the averages differently. This does not make sense, however, with noisy data like non smoothed trends. We will come back to this subject later.

Smoothing with averages always works, but it means to sacrifice resolution. It is usually more rewarding to try out different reference times first, as we will see.



	A	B	C	D	E	F
1				change	successor	direction
2		U.S.A.		in	higher= 1	changes
3	date	DJIA	ln(DJIA)	12 weeks	lower= -1	counted
4						
5						
6		changes:		523		
7		data: 987	987	974		
8						
9						
10	981128	9333.08	9.14132	0.200135	1	523
11	981121	9159.15	9.122509	0.128873	1	523
12	981114	8919.59	9.096005	0.044233	-1	522
13	981107	8975.46	9.102249	0.063291	1	521
14	981031	8592.1	9.058598	-0.00069	1	521
15	981024	8452.29	9.042193	-0.04973	1	521
16	981017	8416.76	9.03798	-0.06002	1	521
17	981010	7899.52	8.974557	-0.16729	-1	520
18	981003	7784.69	8.959914	-0.15675	-1	520
19	980926	8028.77	8.990787	-0.117	1	519
20	980919	7895.66	8.974069	-0.12473	-1	518

We usually count the changes in direction for several reference times. Besides the 10..16 weeks range already mentioned we test 8..14, 9..15, 11..17 weeks. For this purpose we use a spreadsheet formula, called COUNTCD.WK1 which simply counts changes in direction. If the provisional optimum is found, further improvements can be expected by narrowing the reference range say from 10..16 to 11..16 or 10..15 and so on.

If the trend series are in column (C1..C1000), the spreadsheet is copied into cell (D1) next to it on the right side.

The formula consists of two columns (D1..E1000). Cell (D1) contains the formula giving the trend direction: @if(C1>C2,1,-1) In words it means if the trend increased as compared to the week before, the output should be one, if not, the output should be minus one. Since identical trends up to six digits almost never occur, we denote upward moving trends with +1 and downward moving trends with -1.

Cell (E1) contains the counting formula @if(D1=D2,E2,E2+1). In words it means that if the trend direction in cell (D1) has the same sign as the one in cell (D2) the counter should stay as it was one time step before. If the signs are different, which happens if the trend direction changes, it should add 1 to the counting sum. The reference time with the least changes is the one of choice. The next figure shows what it looks like in the spreadsheet.

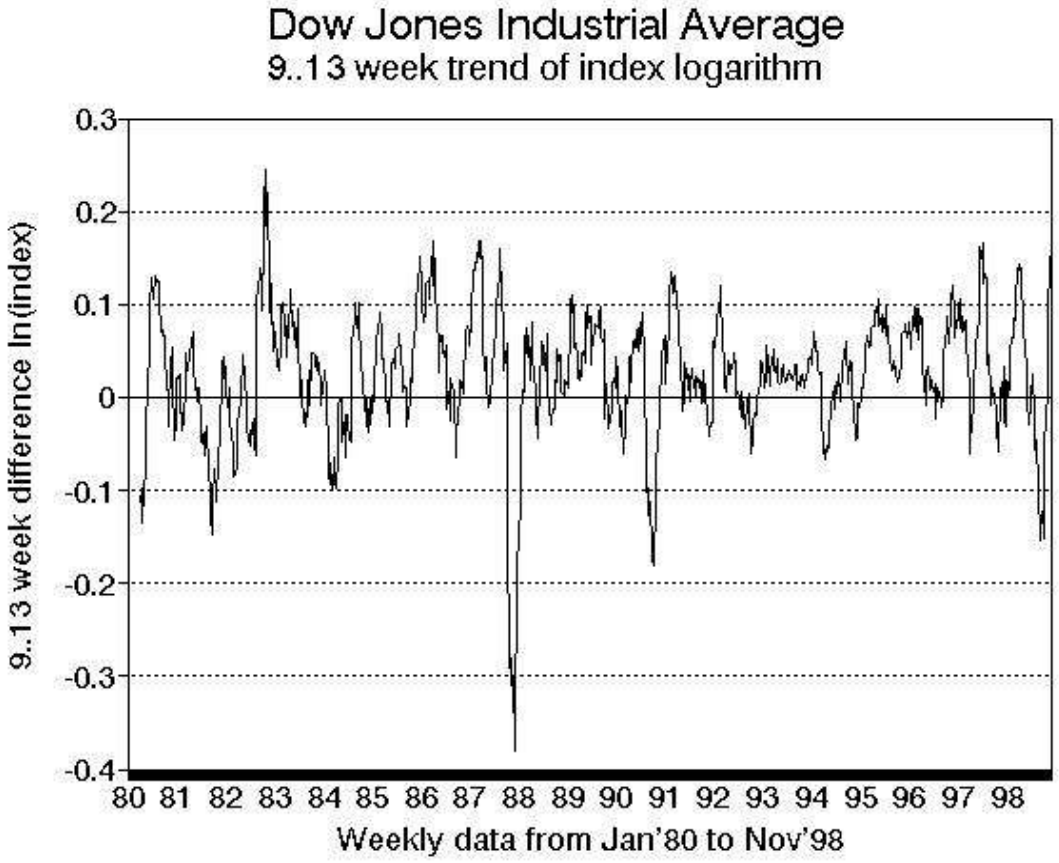
If we apply that formula to the 12 weeks differences of the DOW JONES data, we count 521 changes in direction. The difference to a 7 weeks range has considerably less changes 9..15 or 8..14 give 467 changes 7..13 give 479 and 10..16 give 473 changes. So far we cannot decide, which range is best. This will become clear if we narrow the range. Range 9..14 gives 459 and 8..13 gives 465 changes. Further narrowing yields the optimum at 9..13 with 457 changes. While range 10..13 gives 461 changes. Further narrowing to 9..12 results in 459 changes, which is slightly suboptimal. Therefore range 9..13 is the one of choice.

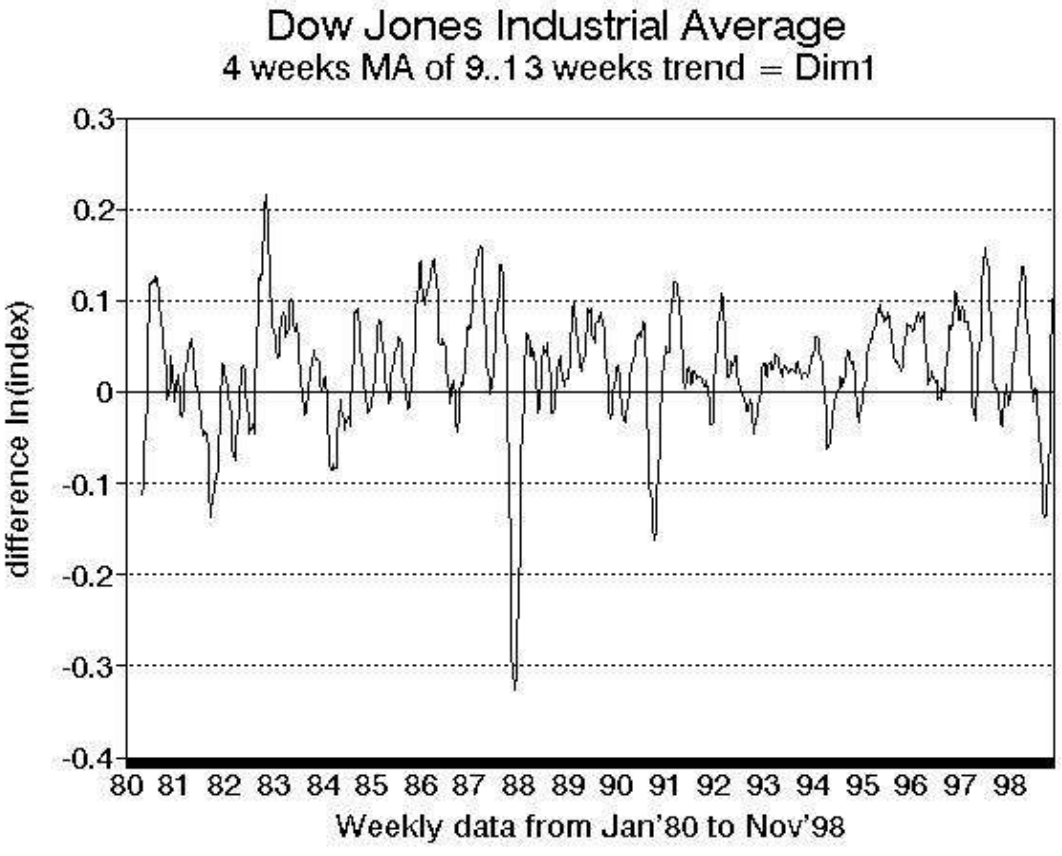
Smoothing with moving averages

Though 457 changes in direction were the best we could get so far, the number is still large. We can lower it considerably, by averaging over neighbouring values. If we use simple linear averages over four weeks, we make a compromise between smoothing the series and preserving the information contained in it. It is not optimal in either sense, but it is easy to use. If we apply a four weeks average to the 9..13 weeks trend of the DOW we obtain a notably smoothed curve. The number of changes is reduced to 205, which speaks for itself. The two graphs for the unsmoothed and smoothed series are given in the next figure.

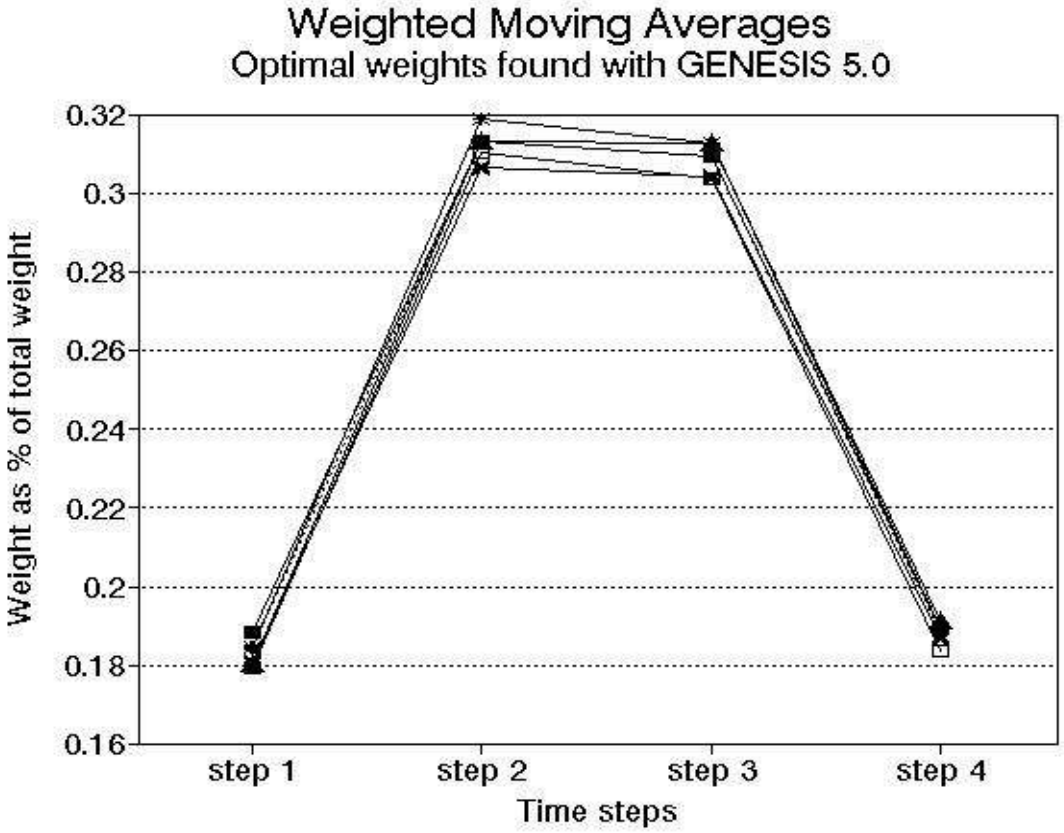
We would like to know, how close the average is to the true value. The correlation is 0.87 without delay, but with 1 week delay the correlation is 0.95 which is quite good. It means that the average will in most cases be closest to the true value not four weeks ahead, but three weeks ahead, provided our forecast is correct.

The curve can be smoothed considerably by assigning different weights to the factors. But smoothing will in many cases run counter to information content. And information loss will inevitably deteriorate our forecasts. Optimal or at least close to optimal smoothing factors can be found using genetic algorithms or other globally optimizing procedures, like simulated annealing methods.


Dow Jones Industrial Average
9..13 week trend of index logarithm
9..13 week difference ln(index)
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98
Weekly data from Jan'80 to Nov'98


Dow Jones Industrial Average
4 weeks MA of 9..13 weeks trend = Dim1
difference ln(index)
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98
Weekly data from Jan'80 to Nov'98

Usually many different combinations of four factors will give rise to the same number of direction changes. We should take the combination which preserves the maximum of information. An estimate of the relative information preserving capacity of smoothing factors can be obtained by multiplying the value of each factor with its corresponding mutual information, see below and adding up the products. The sum of products has to be normalized by deviding through the sum of factor values. The higher the ratio the more information is preserved. A common observation is that the second and third factors are nearly equal and higher than the first and fourth, which are also similar in size. Since the second value contains more information it is better if the second is the biggest factor. These hints are given for those who do not have access to genetic algorithms and want to try out some combinations by hand.



Genetic algorithms or simulated annealing methods will most probably not be available in your spreadsheet program. The former can be obtained as freeware programs like GENESIS or EVOLUTION MACHINE through the internet. Both programs require C-compilers, but that is not the real hurdle. UNIX systems are all equipped with them and excellent C compilers are available as public domain for DOS, for example the DJGPP port of the GNU compiler. However, some knowledge of the C programming language is indispensible, because the function to be optimized has to be coded by the user.

To give you an idea what genetic algorithms can accomplish we have tried to minimize the changes in direction by smoothing with a weighted four weeks moving average. The optimal weights were found with the program GENESIS version 5.0 from John Grefenstette. As written above many combinations of four factors perform equally well. A collection of which can be found in the table presented below. While a simple four weeks moving average had 205 changes in direction, the optimally weighted averages had only 167 changes, which is almost 20% less. Though the curve of the weighted average is even more smoothed, it correlates better with the unsmoothed value than the ordinary average. The rank correlation according to Spearman was

best with one week delay for both averages. For weighted averages it was rs=0.975 versus rs=0.971 for ordinary averages. The linear correlation according to Pearson was also best with one week delay in both cases. Again the weighted average was superior with Rý=0.962 versus Rý=0.957. Therefore with weighted averages we do not only obtain a smoothed curve by reducing the noise. We seem to do so selectively because we gain information.

SUITABLE FACTORS FOR WEIGHTED MOVING AVERAGES

1st week	2nd week	3rd week	4th week
2905	4835	4777	2925
2852	4835	4777	2910
2852	4924	4831	2925
2768	4791	4688	2832
2793	4733	4693	2886
2783	4835	4826	2950
2798	4688	4630	2832
2768	4733	4625	2832
2724	4688	4630	2832
2798	4816	4826	2950
2901	4973	4924	2989
2901	4973	4909	2989
2901	4973	4924	3048

Each row of the table above gives a suitable combination of weighting factors. The sum of the products has to be normalized, i.e. it must be divided by the sum of the factors. An example carried out with the underlined factors is found in the processing spreadsheet on disk. If the factors are normalized (sum of the four factors = 1), the differences between various combinations become marginal as can be learned from the next figure. It shows line graphs of the normalized first six rows from the table above.

How much information is contained in the data series?

Once the right reference time is chosen, we have a time series of trend values. This series represents the first dimension of the dynamic system (market) under study and all subsequent procedures will use it if not stated otherwise. First we have to determine the information content of the series. We will do this with non parametric tests, because they are distribution independent. This means their results will be valid not only for normally distributed data, but with data of any distribution.

The first test is Spearman's rank correlation test, which is the non parametric equivalent of Pearson's linear correlation test. This test measures monotonous dependency between individual data of a series. We have coded the procedure into a program named RANKCORR.EXE which will output a table printed on the screen giving the rank correlation of data which are 1 to 52 time steps apart, and the corresponding significance of the correlation value. This will give us a good estimate for the correlation time of the data. We will need it later.

The second test to perform is the entropy measurement. Again we have coded the procedure into a program. This program ENTRO32E.EXE classifies the data into 32 classes and determines the entropy, the conditional entropy and the mutual information of the classified data. The mutual information is an even more general measure of dependency between data than the rank correlation. It measures any kind of dependency. We will use the mutual information values later.

Extracting the dynamics by 'embedding'

We want to find the dynamic hidden behind and controlling the market data. This seems at first sight impossible, if we have only a single time series of market prices available. Certainly there are more forces driving the market than the actual price alone. Here again we can expect help from chaos theory. As was already mentioned above, in chaotic systems the components making up the system are coupled. This has the beneficial effect that each variable can deliver information about the others because of this coupling.

Time delay embedding

The term 'embedding' will be unfamiliar to many readers. This expression was coined by topologists, mathematicians who study spatial structures and their transformations. An embedding is the specialists expression of an equivalent transformation of structures, meaning that the structures in question can be smoothly transformed forth and back without loss of information.

Time delay embedding is one of the great inventions we owe to chaos theorists. It is an elegant procedure, which transforms time coordinates into space coordinates or if you prefer that metaphor serial into parallel information.

In order to characterize a certain state of a multi dimensional system, one needs depending on the number of dimensions of the system one, two, three or more independent measurements, one for each dimension. The brilliant idea of chaos theorists was that independence of the data was important but their origin was not, provided they came from the system under study. To obtain say three independent data for each time step from a single time series, they could not just take a block of three consecutive values. Why not? Because the data show more or less auto(self) correlation and thus are not independent. The data are in part redundant. Which means they contain the same information. But the autocorrelation declines with time and the next value with reasonably low correlation to the first selected value is a suitable candidate for the next (second) independent value. The delay time between the first and second values will be the same as that between the second and third values. It is called 'tau' (Greek for t). The following table shows schematically how the values are picked.

One value	time steps			
per dimension	1	2	....	n
1st dimension	x(1)	x(2)	....	x(n)
2nd dimension	x(1+tau)	x(2+tau)	....	x(n+tau)
3rd dimension	x(1+2*tau)	x(2+2*tau)	...	x(n+2*tau)
.....				
kth dimension	x(1+(k-1)*tau)	x(2+(k-1)*tau	...	x(n+(k-1)*tau)

To build a two dimensional data base one would take the value at time (t) for the first dimension and the value at time (t + tau) for the second dimension. For each value of the first dimension there would be a corresponding value for the second dimension which was 'tau' time units away. This can be easily carried over to three and more dimensions.

So far we have not offered judgement criteria for reasonable decorrelation. Indeed the decision is not clear cut. A broad range of values have been tried successfully. Here we will give two popular measures referring either to rank correlation or to mutual information. If the rank correlation is below 1/e = 0.3678.. then the values are reasonbly decorrelated (e = Euler's number = 2.7182..). If the mutual information has its first local minimum, then there is sufficient decorrelation. Usually the mutual information first declines for several time steps, but very often this monotonous decline

is interrupted by one or two higher values. The first local minimum is the last value in the row of monotonous decline. It is our experience that in practice there is not much difference between these two criteria. For further discussion of suitable correlation values see below.

In order to determine the rank correlation of the data, we have to write the column containing them to an ASCII-file. The data to be written are the four week averages of the logarithmic differences 9..13 weeks. They will be later referred to as the first dimension data. We have to write them to a file with the number of data given in the first row, followed by the data one per row, because this is the file format expected by all our programs.

The figure shows the output of the rank correlation program. We see that the inital correlation for one week is high, about 0.97 but it declines rapidly with time and the critical value of 0.3678 is passed in the 8th week, when the correlation is down at 0.299. We should keep this in mind because we will need this information later.

The entropy table for the DOW data shows that the entropy increases with time, we could also say our uncertainty increases and the mutual information of the data declines with time. After four weeks only one fifth of the original information is contained in the data. The mutual information is down at 0.21 from 1.0 at time zero. The first local minimum of the mutual information is reached at 11 weeks with 0.1159. It will be clear soon why we emphasize this point here. Besides the mutual information the table gives values for conditional entropy and conditional entropy per bit. The values for conditonal entropy and mutual information depend on the number of classes. While the entropy per bit gives the conditional entropy relative to the resolution (number of classes) in bits. Here we have 32 classes = 25 = 5 bit resolution.

Spearman rank correlation (rs)

file # 1: djia1dim.prn
file # 2: djia1dim.prn

d	rs	signif.	d	rs	signif.	d	rs	signif.
0	1.000	>99.9%						
1	0.971	>99.9%	2	0.900	>99.9%	3	0.804	>99.9%
4	0.698	>99.9%	5	0.595	>99.9%	6	0.494	>99.9%
7	0.394	>99.9%	8	0.299	>99.9%	9	0.209	>99.9%
10	0.130	>99.9%	11	0.067	96.1%	12	0.019	45.0%
13	-0.013	30.6%	14	-0.034	70.1%	15	-0.049	87.1%
16	-0.062	94.6%	17	-0.076	98.0%	18	-0.089	99.4%
19	-0.099	99.8%	20	-0.106	99.9%	21	-0.108	>99.9%
22	-0.105	99.9%	23	-0.097	99.7%	24	-0.088	99.3%
25	-0.077	98.3%	26	-0.065	95.4%	27	-0.049	86.8%
28	-0.029	63.1%	29	-0.008	19.2%	30	0.012	29.5%
31	0.029	62.9%	32	0.041	79.4%	33	0.049	86.8%
34	0.054	90.3%	35	0.059	92.9%	36	0.065	95.4%
37	0.073	97.4%	38	0.080	98.5%	39	0.084	99.0%
40	0.084	99.0%	41	0.080	98.5%	42	0.072	97.2%
43	0.061	93.6%	44	0.047	85.1%	45	0.032	66.4%
46	0.013	29.8%	47	-0.009	21.7%	48	-0.034	69.5%
49	-0.061	93.8%	50	-0.090	99.4%	51	-0.118	>99.9%
52	-0.141	>99.9%						

d = delay (time steps) signif. = significance

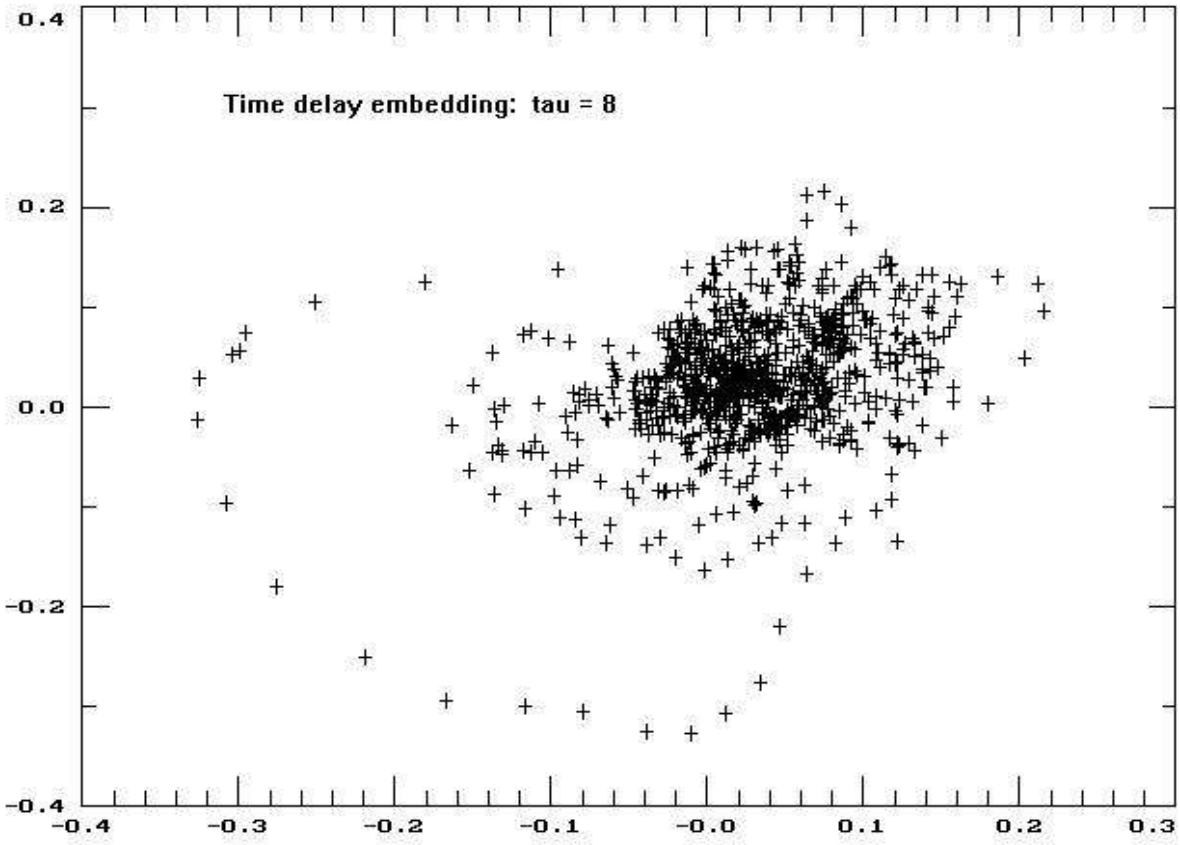
delay	entropy	conditional entropy	mutual information	entropy/bit
0	2.6307	0.0000	1.0000	0.0000
1	2.6286	1.1733	0.5536	0.2347
2	2.6264	1.6412	0.3751	0.3282
3	2.6241	1.9093	0.2724	0.3819
4	2.6225	2.0631	0.2133	0.4126
5	2.6212	2.1514	0.1792	0.4303
6	2.6215	2.2150	0.1551	0.4430
7	2.6222	2.2780	0.1313	0.4556
8	2.6225	2.2914	0.1263	0.4583
9	2.6225	2.3146	0.1174	0.4629
10	2.6214	2.3149	0.1169	0.4630
11	2.6207	2.3168	0.1159	0.4634
12	2.6199	2.3161	0.1159	0.4632
13	2.6190	2.3082	0.1187	0.4616
14	2.6182	2.3165	0.1152	0.4633
15	2.6173	2.3198	0.1137	0.4640
16	2.6163	2.3476	0.1027	0.4695
17	2.6154	2.3382	0.1060	0.4676
18	2.6144	2.3157	0.1143	0.4631
19	2.6132	2.3236	0.1108	0.4647
20	2.6132	2.3246	0.1104	0.4649
21	2.6131	2.3110	0.1156	0.4622

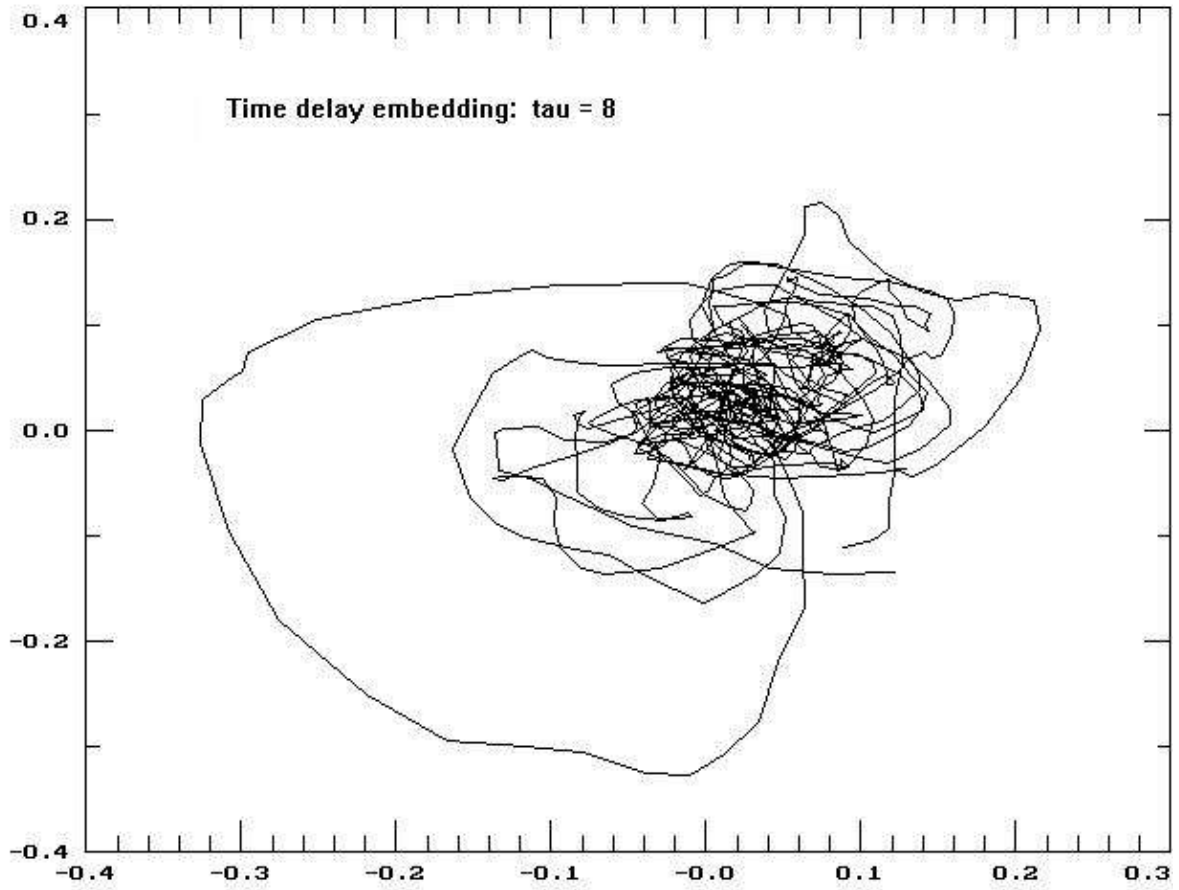
file = djia1dim.prn ndata = 950 to 971 Resolution = 5 bit

We can learn from the table that the conditional entropy reaches a plateau value after nine weeks, with about 0.46 per bit. The conditional entropy is smaller or equal to the (unconditional) entropy, but it can never exceed it. Here it is used as a measure of our uncertainty about the outcome at time zero if we know what happened at 1 or 2 or 3 .... time steps earlier. The number of classes chosen does not effect very much the outcome of the test. We would have a very similar picture, with respect to entropy, conditional entropy and mutual information, with 16 classes = 24 = 4 bit resolution. We would, however, sacrifice resolution. On the program disk there is a program which uses 6 bit resolution (64 classes). What is the case against using that program? In the statistics literature a number of classes close to the square root of the number of data is recommended, therefore we chose 32 classes here. If the number of data is much smaller or much larger than 1000 the smaller or the higher resolution should be used respectively.

Time delay embedding of the DOW JONES data

We have observed that with 8 time steps delay the rank correlation was smaller than 1/e. While the first local minimum of the mutual information was reached after 11 weeks and the conditional entropy changed only marginally after 9 weeks. This would suggest that the delay time (tau) for embedding should be in the region of 8 to 11 weeks. We will start with 8 weeks and show the embedding in two dimensions.

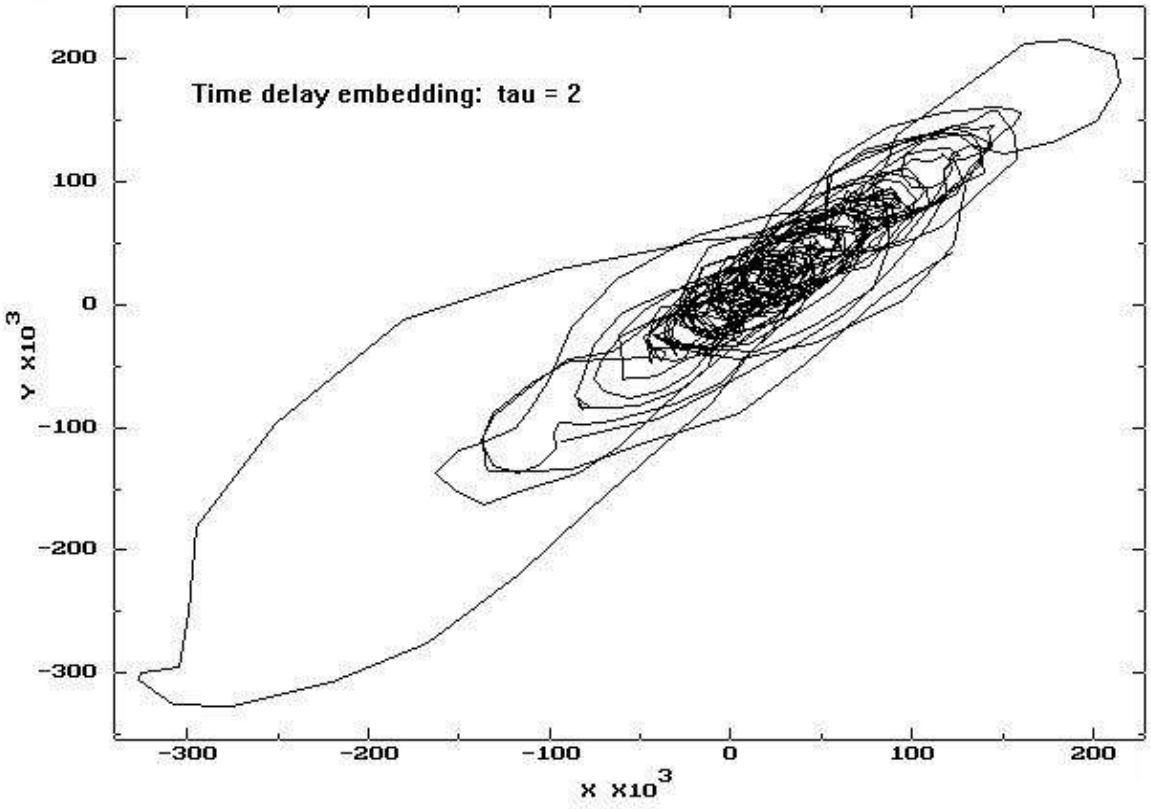

Time delay embedding: tau = 8
0.4
0.2
0.0
-0.2
-0.4
-0.4
-0.3
-0.2
-0.1
-0.0
0.1
0.2
0.3

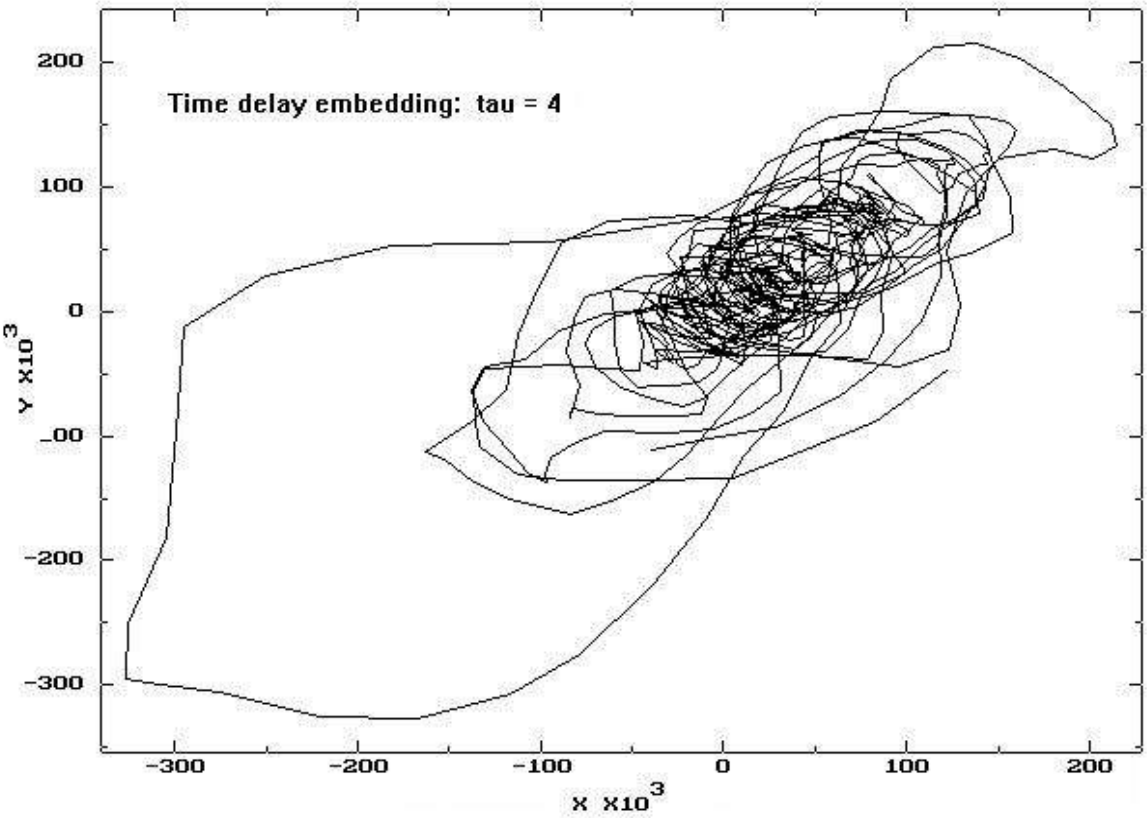

Time delay embedding: tau = 8
0.4
0.2
0.0
-0.2
-0.4
-0.4
-0.3
-0.2
-0.1
-0.0
0.1
0.2
0.3

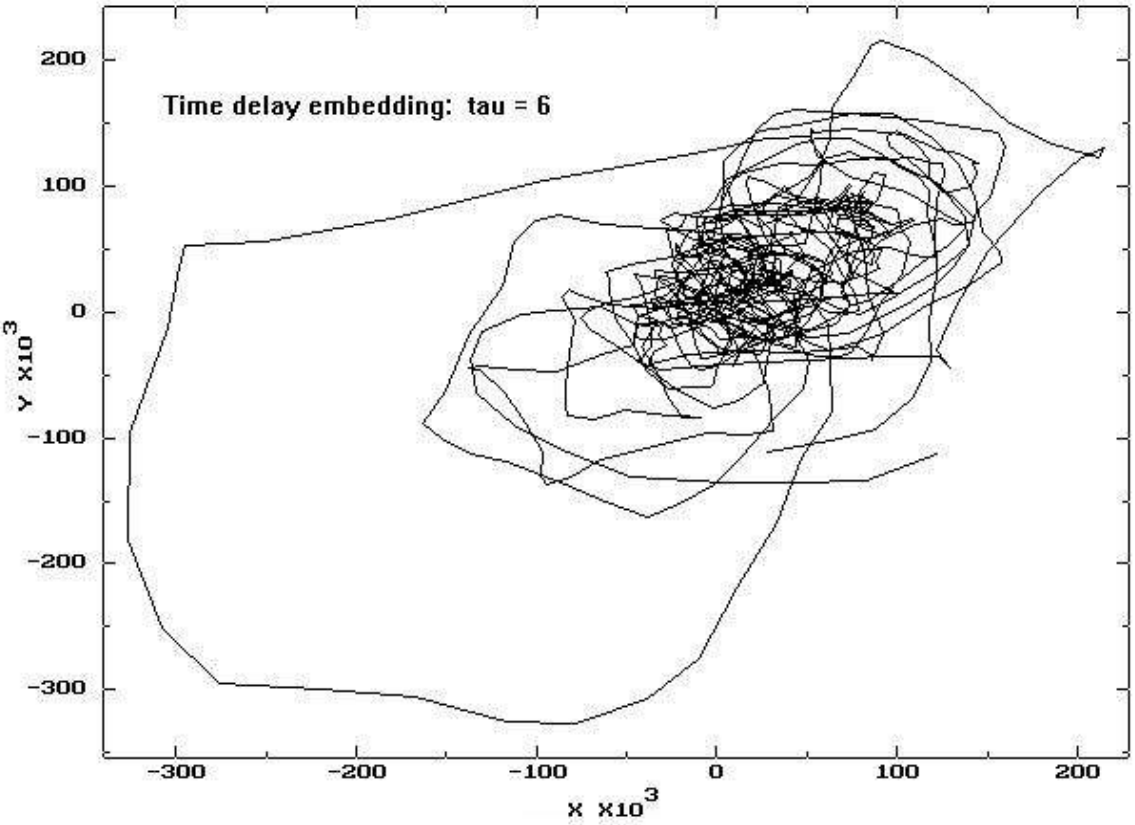
We show two graphs, which give you complementary information. The x,y scatter plot shows the known points in state space of the system. From this plot one can count for how many time steps the system stayed in certain regions of the state space, such as the zones near the extremes. And this plot shows us which points will be considered if we look for the nearest neighbours in state space. This task will be dealt with in part III. The line plot connects successive points and thus shows the path of the system in state space, which would not become clear from the scatter plot alone.

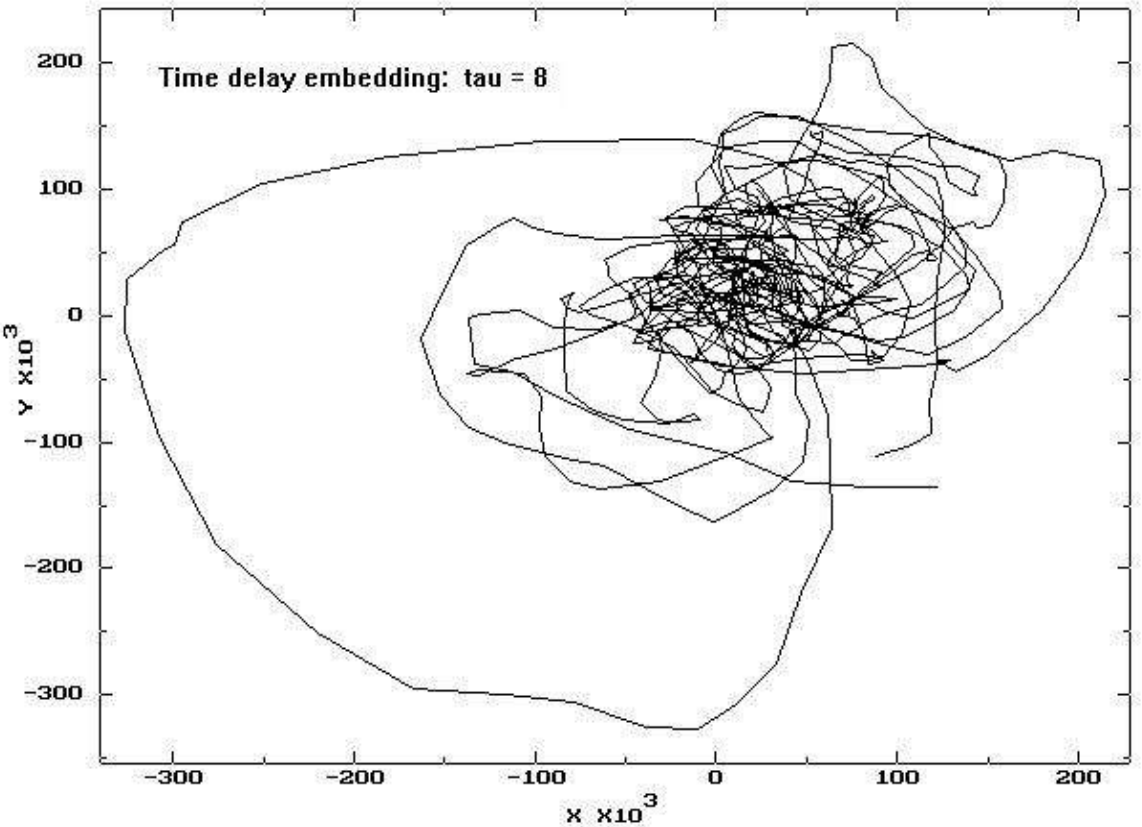
The weighted averages talked about in the context of genetic algorithms could likewise be used for time delay embedding. Rank correlation becomes smaller than 1/e at eight weeks delay which is the same as found for the ordinary moving average. However, the resulting two dimensional phase diagramm is not very different from that of the other average, therefore we skip it here.

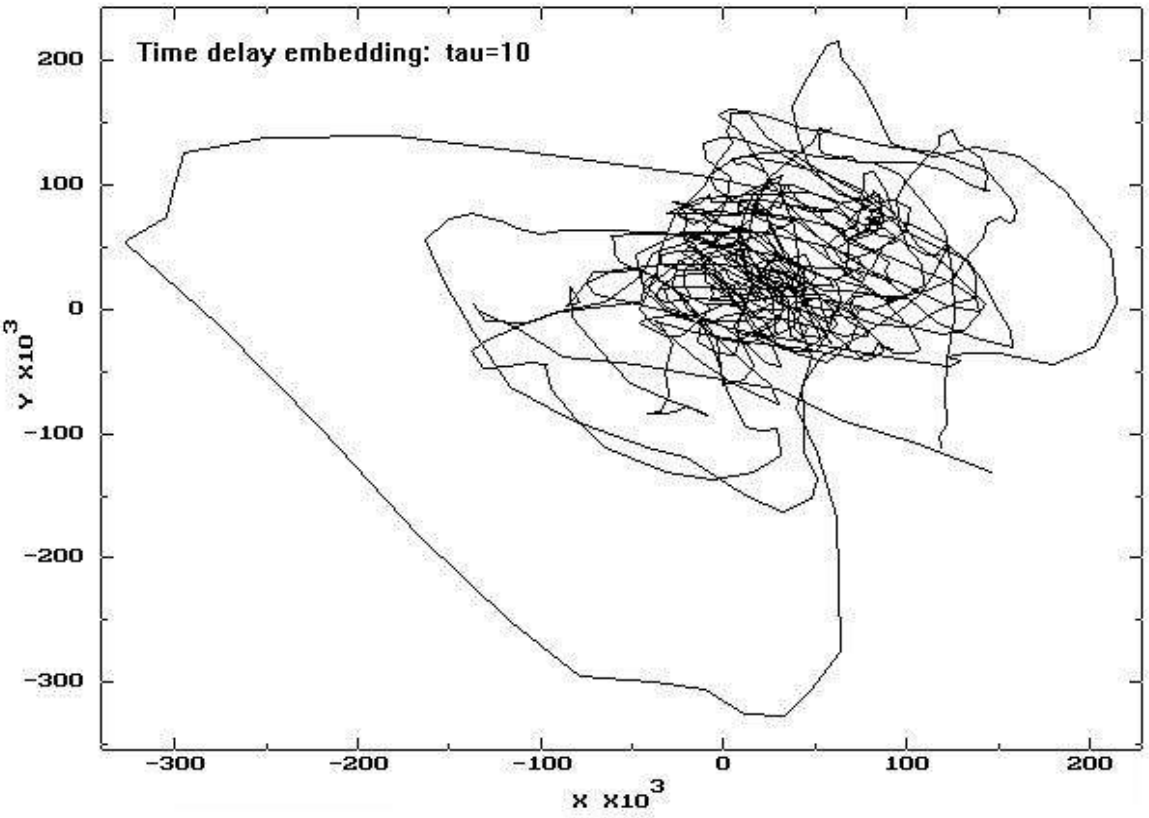
As already mentioned the choice of tau (time delay) is more or less arbitrary. If tau is chosen too short, the attractor will not be completely unfolded, and we would underestimate its dimension. If tau is too long the attractor will become distorted and order would be lost. The following figures show a series of two dimensional phase diagrams with increasing tau. The first coordinate (horizontal scale) is our usual 1st dimension. It is kept constant for all figures. As the second dimension (vertical scale) we use the same time series with the only difference that it is shifted by 2,4,6,8,10 and 12 time steps relative to the first dimension.

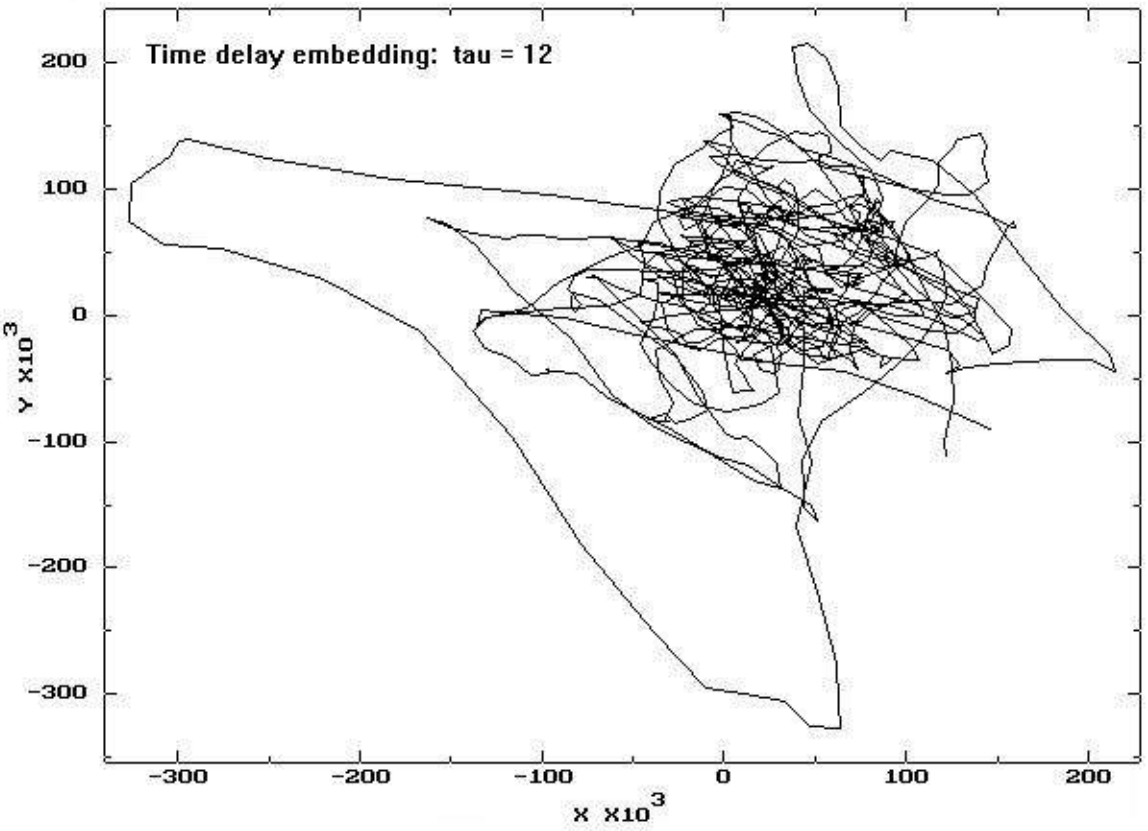

Time delay embedding: tau = 2
200
100
0
-100
-200
-300
Y X10^3
-300
-200
-100
0
100
200
X X10^3


Time delay embedding: tau = 4
200
100
0
-00
-200
-300
Y X10^3
-300
-200
-100
0
100
200
X X10^3


Time delay embedding: tau = 6
200
100
0
-100
-200
-300
Y X10^3
-300
-200
-100
0
100
200
X X10^3


Time delay embedding: tau = 8
200
100
0
-100
-200
-300
Y X10^3
-300
-200
-100
0
100
200
X X10^3


Time delay embedding: tau=10
200
100
0
−100
−200
−300
Y X10^3
−300
−200
−100
0
100
200
X X10^3


Time delay embedding: tau = 12
200
100
0
−100
−200
−300
Y X10^3
−300
−200
−100
0
100
200
X X10^3

You will easily recognize the order in the diagramm with tau=2 time steps . The curve uses certain paths repeatedly and thereby draws characteristic patterns. With increasing tau this order disappears gradually. It becomes more and more difficult to identify certain patterns. We know that the correlation between different points on the curve are limited to about 10 time steps. But the patterns emerge because in a non periodic way the curve reappears in the same regions. These recurrences are far apart in time and thus uncorrelated in the time domain. Therefore the order we see is not due to autocorrelation but due to an ordered dynamic which is an inherent property of the market.

Other embedding variants

Time delay embedding has become the classical embedding method for chaotic data, however, it is not the one and only way of embedding. It has been suitable for many chaotic model systems, for which almost endless and noise free data are available. In the case of noisy and limited data series it is not always the method of choice. Our experience is that it does work if the data series has been smoothed with appropriate factors selected by genetic algorithms.

Another embedding method utilizes first differences as second dimension and second differences for the third dimension, but this method requires very smooth data series. We have already tried it sucessfully with two dimensions and excessively smoothed data. A further method uses inverse exponentially weighted averages for the second dimension and first differences of the second dimension as third dimension. We use still another method which resembles the former, with respect to non linear weighting. But we weight the data according to their information content.

As already discussed above, averaging can reduce noise, but it also smears out information. We require a smooth curve for the second dimension, but we want to lose as little information as possible. To come close to this goal we can weight the data according to their information content. We know this already from the analysis of our data with the program ENTRO32.EXE which shows the mutual information of the data. The information declines with time, that is what one would expect. But how much of the information gets lost in each time step. This becomes apparent if we look at the mutual information of the data. It shows how much information earlier values of a time series provide about later values of the series. The latest value contains 100% information about itself, of course. The preceding value contains about 60% information about its successor. If we go back two time steps we see that it contains roughly 40% information about the latest value. Even earlier values contain about 30%, 25% etc about the far future. We see that there is an inverse exponential decline of information. In the first step we loose 40%, then 20%, 10% and 5%, soon approaching 0. These are only approximate values. One should not expect real world data to look exactly like that. But essentially the information decline resembles this example quite closely and the residual information often reaches a plateau.

Obtaining the second dimension

We want to use our data to forecast the future. Now we have to decide, how far into the future, i.e. for how many time steps, we want to make our forecast. And second we have to decide, in which way we want to make our forecasts, either as a one shot prognosis or as an iterated prediction, because that will determine how the data will be weighted.

How to predict, in one step or iteratively

A forecast leading say four time steps into the future may be carried out in several ways.

a) In one step - forecast over four time units.
b) In two steps - over two time units each.
c) In four steps - of one time unit each.

Procedure (b) and (c) are called iterated (repeatedly performed) predictions. For our example we will use the one step procedure, but we will briefly discuss the iterated prediction because in the chaos literature one finds strong arguments in its favour.

Iterated predictions use the predicted value of the first forecast as an input value for the second prediction and the result of this as an input for the next one and so forth.

One argument in favour of iterated predictions refers to the Lyapunov exponents. In short the Lyapunov exponents are measures of the sensitivity of a chaotic system to small changes in initial conditons. If a Lyapunov exponent is positive, initally small differences will grow with time. All chaotic systems possess at least one positive Lyapunov exponent. The smaller the exponent the more benign the system will behave. It was argued that in iterated predictions the prediction error would grow with the largest positive Lyapunov exponent, while in the one shot prediction the error would grow with the sum of all positive Lyapunov exponents. It is apparent that the sum yields the bigger value. However, this argument holds only for systems with more than one positive Lyapunov exponent. Chaotic systems with three dimensions have only one positive Lyapunov exponent. In the few cases where market dimensions were estimated no more than three dimensions were found. So this argument does not count for much in our case.

Chaotic model systems even if of low dimension could be predicted better with the iterative method. But those model data are free of noise and the underlying systems show strictly determined behaviour. In the case of noisy data series the iterative method will inevitably amplify the noise and this may result in predictions worse than those with the one shot method. Method (b), see above, is a compromise between the two extremes. However, we suggest to start with the one shot method, because it is easiest. If the data are not too noisy, method (b) could probably give better results. Only if method (b) is better than method (a) an attempt with method (c) seems justified.

One step prediction over four weeks

We want to forecast four weeks ahead in one step. To obtain the second dimension from the first one we proceed as follows:

For convenience we multiply all factors by 10000. The only reason for this is that this way we can spare some typing. The first value of the first dimension series is multiplied by 10000, the fourth value is multiplied by the corresponding mutual information times 10000, the fifth, sixth and following values downto the 13th value are each multiplied with the corresponding mutual information times 10000. All the products of first dimension value multiplied by the corresponding mutual information, times 10000 are summed up and devided by the sum of the mutual informations times 10000.

(F10*10000+F14*2133+F15*1792+F16*1551+F17*1313+F18*1263+F19*1174+F20*1169+F21*1159+F22*1159+F23*1187)/(10000+2133+1792+1551+1313+1263+1174+1169+1159+1159+1187)

Provisional second dimension

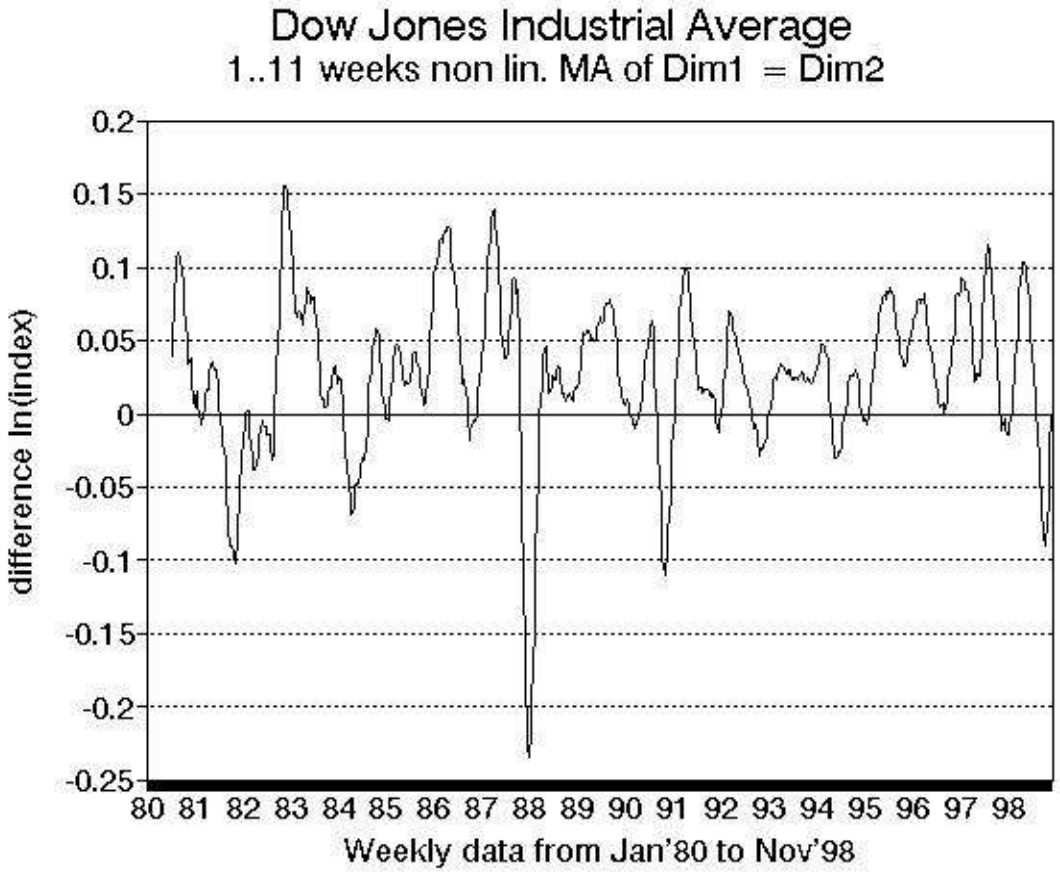
Above you see the formula as it is keyed into the spreadsheet. This transformation formula is copied into the column next to the data series of the first dimension, here it is the G column. It will calculate the values for the second dimension. These values are still provisional. The final

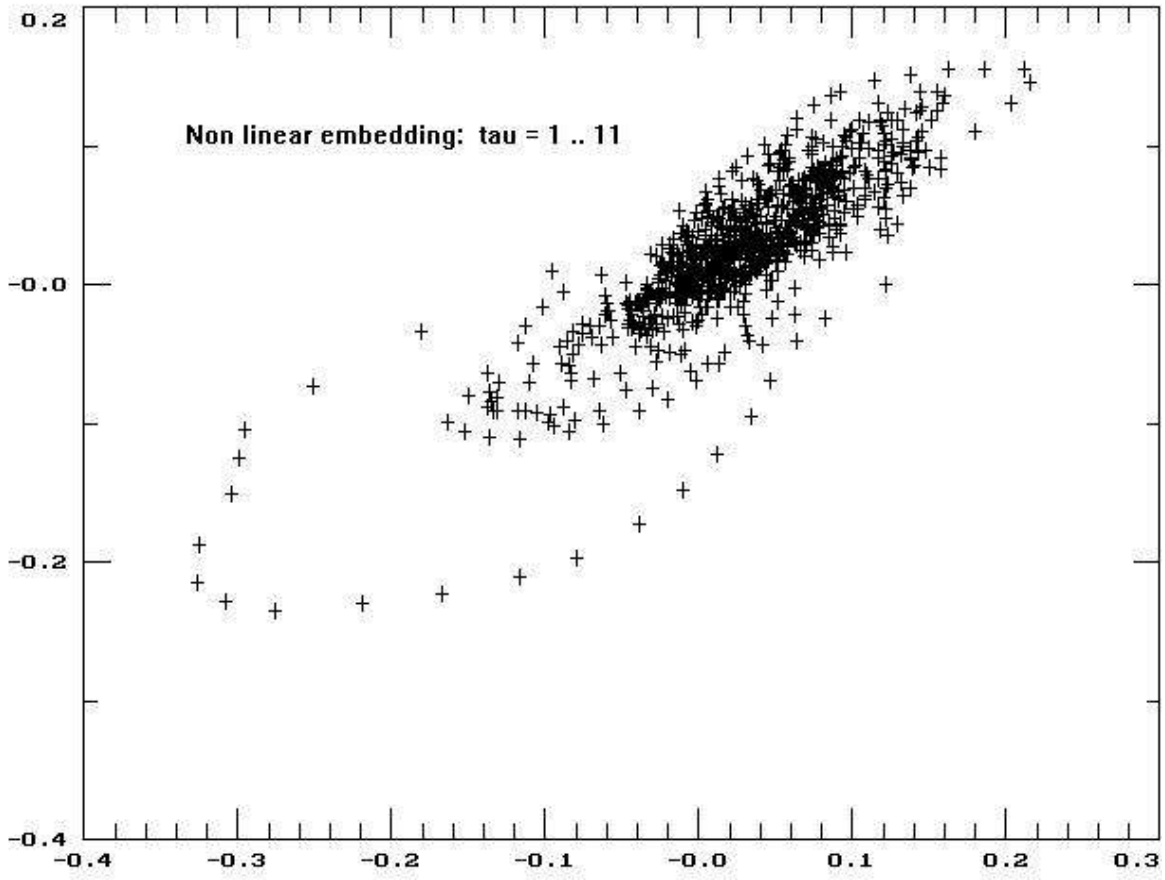
transformation will be determined by trial and error, by minimizing the direction changes of the second dimension. We start with factors 1,4,5,6,...13 and count the direction changes. Then we reduce the number of factors stepwise by leaving the last one out and counting the changes in direction each time. The number of factors which gives the minimum number of changes in direction is the number of choice. Once we have determined both dimensions we can develop a predictor based on the two dimensional projection of the system.

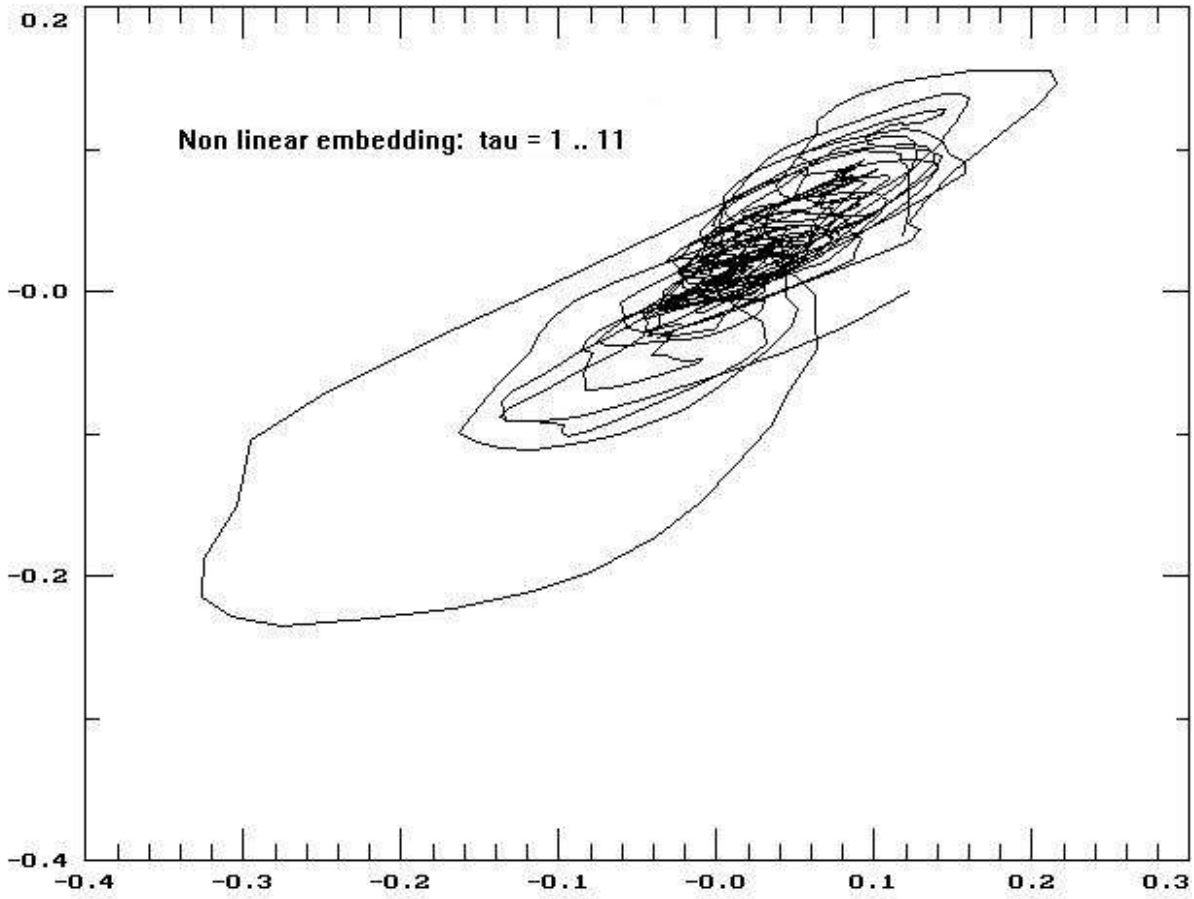
Final second dimension

In our example the formula as given above will lead to 169 changes in direction. Then we reduce the numerator by the last product and the denominator which contains the sum of factors, by the last factor. We obtain a shortened formula which gives us 171 changes. If we reduce the formula once more as just described, we will get 161 changes. Even shorter formulas will increase the number of changes to 163 and 167 and thus are not beneficial. Our optimal formula includes the factors down to 1159, which represents the first local minimum of the mutual information. This coincidence, between the optimal length of the formula and the first local minimum of the mutual information can be observed quite often, but not always.

The figures show the one dimensional curve for the second dimension, which is somewhat smoother than the first dimension.




Non linear embedding: tau = 1 .. 11
0.2
-0.0
-0.2
-0.4
-0.4
-0.3
-0.2
-0.1
-0.0
0.1
0.2
0.3


Non linear embedding: tau = 1 .. 11
0.2
-0.0
-0.2
-0.4
-0.4
-0.3
-0.2
-0.1
-0.0
0.1
0.2
0.3

If we plot the first dimension against the second dimension we obtain the two dimensional curve shown above.

How many dimensions are necessary?

For theoretical reasons at least three dimensions are necessary to characterize chaotic systems, since only systems consisting of three or more components can show chaotic behaviour. Some researchers have estimated how many dimensions markets would have. They came up with a fractal (broken) dimension of 2.2 for a currency market (US-$/Yen) and with 2.33 for a stock market index (S&P 500). In both cases these values are closer to two than to three dimensions. Nevertheless these data would mean that a third dimension would be necessary to characterize unambiguously a particular state of the system.

Suitable choices for the third dimension

We have tried out different types of predictors for the same markets. Some of which use only two dimensions, but in that case they take more than one value per dimension, in order to obtain vectors for each dimension. Others make use of three dimensions.

Then what is the third dimension we would select. For time delay embedding the situation seems to be clear. We just double the delay time to obtain the value for the third dimension. If the delay time is chosen too long there would be no interrelation any more between the first and the third dimension. But if we choose to embed the system in three dimensions we should select a time delay value which still preserves some association between the first and the third dimension. Otherwise we would loose a typical trait of a chaotic system. Thus to keep all dimensions of the system coupled one has to accept higher correlation values between neighbouring dimensions. For our example it would mean that we could use a time delay in the range of 4 to 6 weeks if we embed in three dimensions.

So much to the time delay embedding. It should have become clear meanwhile how to apply this method. The subsequent explanations will predominantly refer to non linear embedding. And the 'nearest neighbour' predictor will use that method. But certainly a predictor based on time delay embedding would also work. If you go thoroughly through the next part you will be able to set up a time delay based predictor on your own. If you do so and aim to compare the two predictors you have to consider that they will not predict the same thing. The time delay predictor will predict the first dimension, while the other one will predict the (smoothed) second dimension. Of course both will be compared to the statistical error for the relevant dimension, but it would still resemble comparing apples to pears. A fair comparison would have to be based on the same prediction, and fortunately this is not difficult to achieve. You will have to back calculate the first dimension from the second, which is no problem with the formula at hand. See below for back calculation.

Generating the third dimension for our data

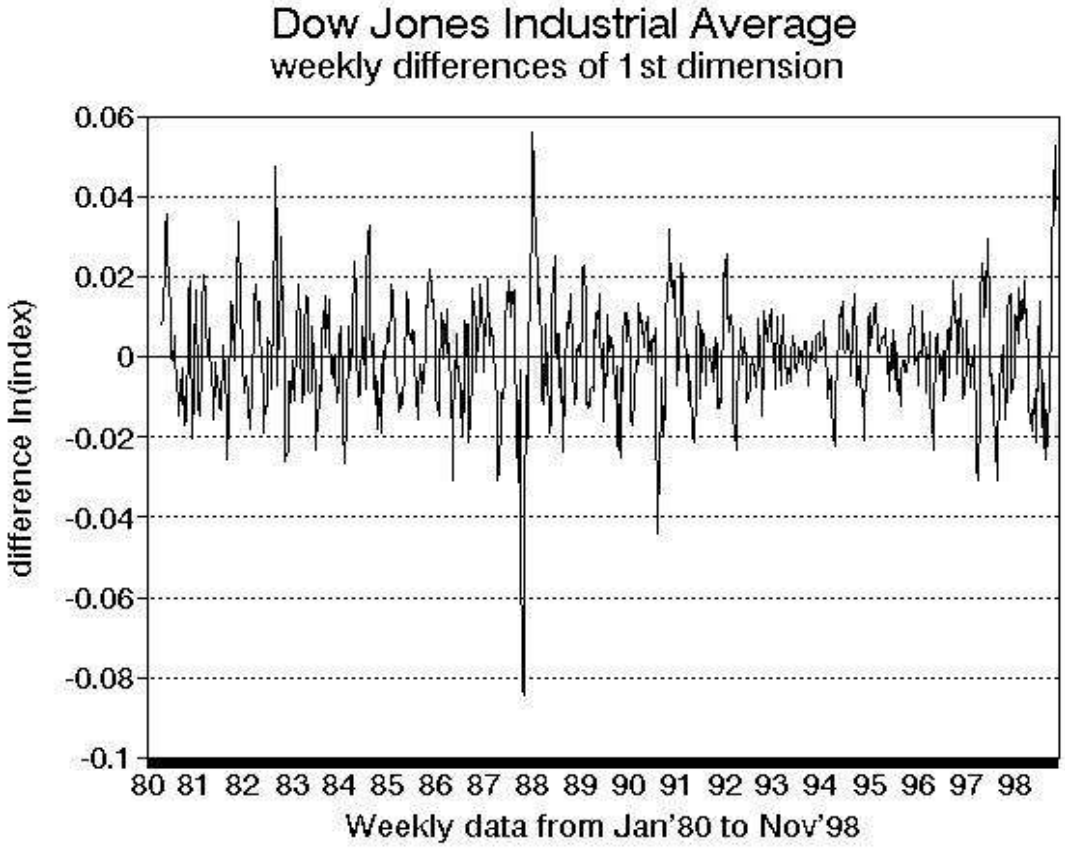
We could use first differences of the first or the second dimension to obtain the third dimension. In case of the time delay embedding we would be free to choose. But for the non linear embedding we could only use differences of the first dimension. The second dimension is already a short term integration of the first dimension. If we differentiate the second dimension, we would end up with something similar to the first dimension.

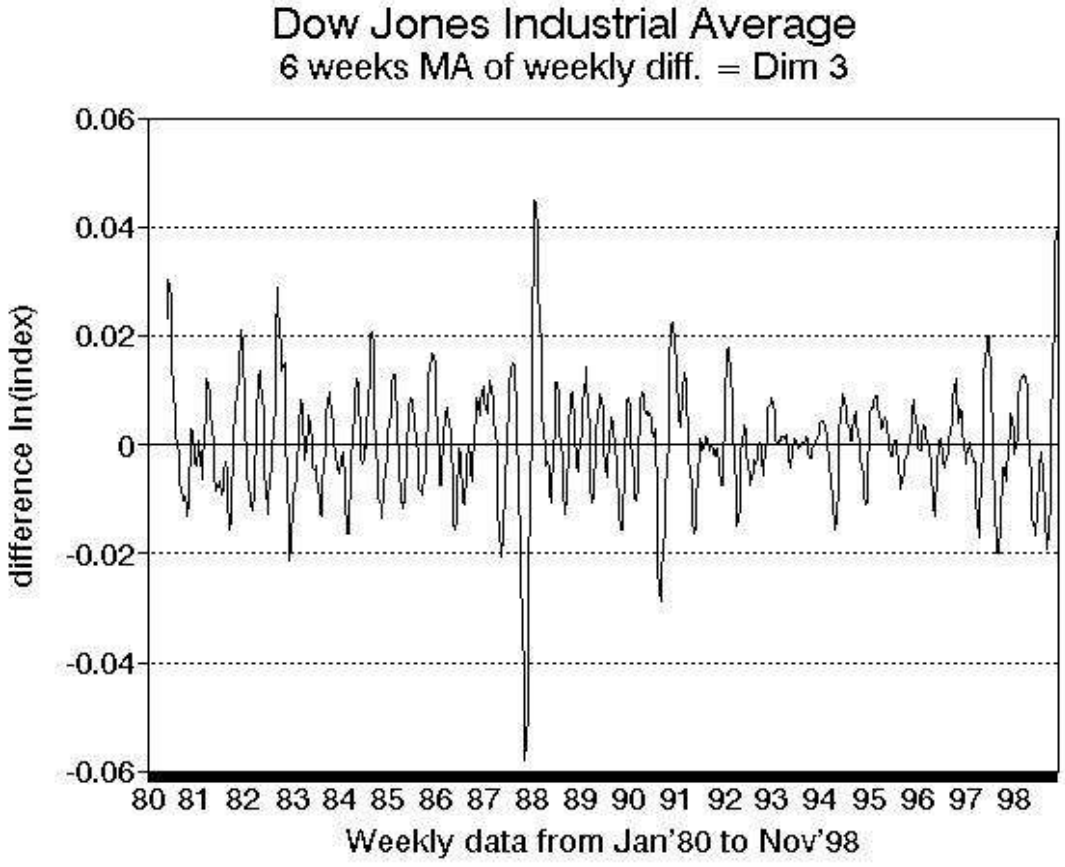
The data series of the weekly differences of the first dimension does not yield a smooth curve, but is rather erratic. If we apply the COUNTCD.WK1 formula to our example data, we find 473

changes in direction. This can be improved upon by averaging over several values. A linear moving average over six values was often found suitable in the case of four week prognoses. If we use this fist formula here we can reduce the changes in direction to 183. This series will be our third dimension.

This averaging does not only result in a smoother curve but also in a better correlation with the second dimension value four time steps ahead, which is the value we wish to forecast. The corresponding value for the first dimension can be back calculated from that for the second dimension. We will come to that later.

With the extracted market dynamics at hand we can try to develop prognosis instruments. Which values should be the basis for our prognoses and how many values are needed for optimal predictions? Obviously we should take those values which contain the most information about the future value. We can measure the information content with the mutual information. The program we used before ENTRO32.EXE worked with one data series and reported the mutual information of data within the same series. Now we would like to know the mutual information between different series. This can be done with program CONENT32.EXE which works with two data files. It reports how much information the second data series provides about the first. The program RANKCORR.EXE can also compare two different files, but the program output cannot be interpreted in terms of information content. It may however give some useful hints concerning the correlations between the different dimensions.


Dow Jones Industrial Average
weekly differences of 1st dimension
difference ln(index)
0.06
0.04
0.02
0
-0.02
-0.04
-0.06
-0.08
-0.1
80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98
Weekly data from Jan'80 to Nov'98


Dow Jones Industrial Average
6 weeks MA of weekly diff. = Dim 3
difference ln(index)
0.06
0.04
0.02
0
-0.02
-0.04
-0.06
80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98
Weekly data from Jan'80 to Nov'98

As described above for the first dimension, the data for the second and third dimensions have to be written to an ASCII file. The second dimension has 958 data and the third dimension has 965 data. The programs have no problem if data files to be compared have different lengths.

Looking for an optimal model

We will now load the program CONENT32.EXE to measure the cross dimension mutual information. The first file asked for by the program should be the file containing the second dimension data series. As second file we should try the first dimension series, second dimension and third dimension one after another. One has to look for high mutual information of the data from four and more time steps back. We will take no more than four values altogether. It is recommended to take one value more than the number of dimensions of the underlying system. If more values than necessary are used for prognoses the prediction does not improve, but to the contrary may even worsen. This was a common experience made by several researchers.

For the DOW JONES data the following picture presents: The mutual information of the first dimension about the value of the second dimension four weeks later is 0.4062 and for five weeks ahead it is 0.3728. The information of the second dimension about its future values four weeks ahead is 0.3170 and for five weeks it is 0.2727. The information of the third dimension about the second dimension value is highest six weeks in advance with 0.1566 and thus clearly above background. For the third dimension the maximum of the mutual information does not coincide with the maximum of the rank correlation. The latter is highest eight weeks in advance.

Estimating the total information to be gained

Which four values are optimal. Shall we simply use the four highest values? Definitely not! We have to look for the highest total information. If any two values are highly correlated the two values together do not contribute much more information than one of them alone. Therefore highly cross correlated values should be avoided.

The auto correlation of each of the three dimensions is quite high, whereas the interrelation is much smaller. Therefore it is recommended to take as the first three values the largest values of each dimension. Difficult becomes the decision for the fourth value, which could be from the first or the second dimension. A look at the rank correlation shows, that a five weeks prognosis based on the first dimension correlates much better with the second dimension (0.893) than the five weeks auto correlation of the second dimension (0.789). This finding would favour the following structure for a prognosis based on a three dimensional non linear embedding:

input values			output
1st dim	2nd dim	3d dim	second dim
t-4, -5	t-4	t-6	t

With the input values on the left side we want to forecast the output value on the right side. We will forecast the second dimension value. From this value we can however back calculate the corresponding value for the first dimension.

We have to keep in mind that rank correlation and mutual information are global measures, which are valid for global optimizers, see below. In the case of local optimizers this need not be true. We would therefore recommend to try several different combinations.

To choose a prediction method

The next step will be the decision for a certain type of predictor. It may be a neural net for example which is a popular choice these days. Several authors have shown that neural nets were superior to other predictors particularly if only small numbers of data were available, while with plenty of data available the differences vanished. Neural nets use different activation functions and depending on the type of function the data have to be normalized in different ways, before they can be fed to the nets. We use to develop three types of neural nets: Backpropagation networks after preselection with genetic algorithms and adaptive logical networks which both are global optimizers, or selforganizing maps as established by Teuvo Kohonen, which are local optimizers. These types of neural nets are mentioned here primarily for completeness, they will not be discussed in detail. However, some misconceptions about neural nets, should be briefly addressed here.

Neural nets are optimizing tools, most of which belong to the class of non linear optimizers. They are not automatic generators of optimal models for complex systems - and no other method can achieve this task. There exist mathematical theorems which state that it is impossible to construct an algorithm which will automatically find and extract all information contained in a data series.

Therefore, you should not ask too much from a neural net. If you select your model carefully, you can optimize it with neural nets and obtain a very good model. If you select an inferior model, a neural net will most probably not come to your rescue. If a bad model is optimized with neural nets, it will not become as bad as it could be, but it will never become a good model.

So much to neural nets, model selection and optimization. Now we will change our focus and have a closer look on so called 'nearest neighbour' predictors. This will be done in the next part.

Part III - NEAREST NEIGHBOUR PREDICTION

Introduction

First let us give a general introduction to the 'nearest neighbour' world. We will keep the matter simple.

Most forecasting programs are global optimizers. All these programs will be developed using a large number of so called 'representative' data, for training purposes. They apply a certain rule to all these data and look for the total error. The rule will be modified until it minimizes the overall error with test data, which are not included in the training data sets. This is true for linear multivariate regression programs and many neural net applications.

A different approach is used by local optimizers. They apply their rules only to a small number of selected data sets namely those, which are most similar. These are selected anew each time from a data base and are called the nearest neighbours. As the actual data change the nearest neighbours change as well.

Local optimizers compare favourably with global optimizers, particularly if both are linear predictors. If the data base is very large they even match the best neural nets.

Advantages of nearest neighbour methods over neural nets

Nearest neighbour methods offer a number of advantages even if compared with optimal neural nets.

- transparency
- low cost
- self improving capability
- no lengthy training

Transparency is a strong argument in favour of nearest neighbour predictors. One always knows which data are judged as similar and therefore used as nearest neighbours. There is no black box decision.

Low cost is another advantage of nearest neighbour methods. Basically the method involves a sorting algorithm, which requires a simple spreadsheet program.

The predictors are self improving. Since the data base will increase with time, it will become more and more likely to find very close nearest neighbours, on which to base a forecast.

No lengthy training phase is required to establish a nearest neighbour predictor.

How to find optimal nearest neighbours

Once the criteria for nearest neighbours are defined, the rest is easily done with a spreadsheet program.

Weighting of the reference values

The first decision is about the reference values. These are the values to which the nearest neighbours refer. Nearest neighbours will be selected, because they are similar to these values. Shall we treat all the values the same or is it better to weight the values differently according to their importance, i.e. information about the future. Remember what was said above about validity of mutual information for local optimizers. If you want to adapt the weighting to the mutual information you should know how it is calculated. It is important to know that the conditional entropy and mutual information measurements are performed with normalized values. All values are devided into classes, which spread evenly between the minimum and the maximum value. The calculations are performed with the classified data, not with the absolute values.

The average absolute differences between individual data of each dimension may differ considerably. Since in our example the second dimension is a smoothed first dimension the average difference from time step to time step is smaller than that of the first dimension. The third dimension will have the smallest average difference. If similarity is important and if original values are compared, then the third dimension values contribute only with a small fraction to the total difference. This can be compensated for by weighting the data with different factors. In a neural net this would be done automatically, but for this method it has to be done by hand.

To summarize, we can weight the data according to their mutual information and according to their spread between extreme values.

Choosing a distance measure

If the question of weightings is solved we have to decide about the distance measure we wish to apply. One common measure is the absolute deviation also known as Manhattan norm or city block norm. This norm summarizes the absolute differences, which means that all individual differences to the upside or to the downside receive a positive sign and are added. The total is the absolute difference.

The Euklidian norm is another distance measure. This norm first squares the individual differences, then builds the sum of squares and finally takes the root of the sum of squares. This norm is computationally more expensive than the above method. Which means that it takes more time (factor about 4 depending on computer language and compiler) to calculate the Euklidian distance than the absolute distance. If the databases are small and the computer is fast, it would not make much difference. If the data base is large and the computer is slow it might be an argument to favour another distance measure.

Less well known is the supremum norm. It calculates the absolute differences and takes the largest of these distances as total distance measure. This is the computationally fastest method.

In our example we will use the absolute difference as the distance measure.

If we only looked for the very nearest neighbour we could start now. In practice the very nearest neighbour alone is not the best choice. It rather pays to average over several nearest neighbours. We have tested between one and four nearest neighbours and found three neighbours best in most cases. It is no effort to try different numbers of neighbours. If we decide to average over several neighbours another question arises. Shall we weight the different neighbours uniformly or shall we weight them with respect to their distance. In our experience distance weighted neighbours were superior to evenly weighted neighbours. Again we have to choose a suitable weighting.

We have weighted the neighbours inversely proportional to their (absolute) distance, with good results. The formula given below was used for weighted averages of three nearest neighbours. It assumes that the reference values are placed in row 10, the output of the nearest neighbours are placed in column F and the distance calculation is performed in column G.

(F11/G11+F12/G12+F13/G13) / (1/G11+1/G12+1/G13)

Other weightings are conceivable as well. One could also try to weight the neighbours inversely proportional to the square of their distance.

Predictors can have different orders: 0th, 1st and 2nd order

Now the predictor is ready and we could start with the nearest neighbour prognosis. The type of predictor so far described is called a predictor of zero order. It means that the predicted value is taken as it is, without further modification. We will treat only this type of predictor here.

More sophisticated predictors based on nearest neighbours are feasible and we will mention them here without going too much into details. One refinement would be to draw a regression line through known values of present time and past to the forecasted value and then take the value on the regression line. If the regression curve applied is a straight line, then the predictor is called a local linear predictor. If the regression curve is a second order polynomial it is called a local second order predictor. Predictors of even higher order have no advantage. The local linear predictors are more suitable for iterative predictions over several small time steps. For one step

predictions over four time steps they are not ideal since the trajectory may have significant curvature within that time, nevertheless they are often the method of choice. Local second order predictors are theoretically better suited to that problem. However, they are less robust and sometimes give curious results.

Measures of error and prognostic gain

We want to forecast trend changes as compared to statistical prediction, see below. In case of the Dow Jones data the average change of the short/intermedite term trend was zero over three decimal digits for the last twenty years. This was true for periods of one to ten time steps at least. Therefore the best statistical assumption we can make is that the trend will remain unchanged. Common forecasting rules which rely on probabilities, like auto regression of moving averages (ARMA) cannot in the long run improve upon the statistical error. We can make better predictions, provided that three conditions hold:

a) the trend changes
b) we forecast the right direction of change
c) we do not forecast too much change

We will always compare our prediction error with the statistical error. It will only make sense to develop a predictor if it yields significantly and consistently better predictions than statistical forecasts. How can we measure the prognostic gain i.e. the advantage we take from using a predictor. A variety of measures have been suggested, which asymptotically are equal, but with small samples they give quite different results. We will describe them and discuss in short the pros and cons.

Absolute error measures

mean abs. gain (%) = [1-(abs. prognostic error / abs. statistical error)]* 100.

abs. prognostic error = absolute difference between forecasted and true trend.

abs. statistical error = absolute difference between actual and future trend

Correlation measures

correlation gain = correlation R^2 of forecast with true value as compared to statistical R^2 i.e. self correlation for forecasting period.

Root of mean squared error measures

rms gain (%) = [1-(rms error / rms difference)]* 100

rms = root of mean squares

rms(d) = root of mean squared differences (4 week differences)
rms(e) = root of mean squared error (actual minus forecast)

Rmsd is a measure of the statistical error.

Rmse is a measure of the error made by a forecasting procedure.

Pros and cons of the different measures

Correlation measures are only suitable for long data series. High correlation values denote relative similarity of two data sequences. One must not mistake a similar for a congruous sequence. Similarity is a measure of shape and is irrespective of size. It is not a good measure of error in the case of short time series. If the true value and the forecast differ only by a constant factor and/or a fixed amount the error may be very high even if the correlation is perfect.

The rms gain is a suitable measure for long time data series, but it is less well suited to short series since squaring of errors makes it very sensitive to outliers. Therefore it will change rapidly over time. In other words it is not a robust or stable measure.

Of all the measures of error and prognostic gain listed above the absolute gain is the most robust and reliable. It has the same advantage over the rms measure as the method of least median of squares, also known as LMS has over the least mean squares. The LMS method also minimizes absolute differences.

It is not much effort to calculate all three listed error measures. And we do that routinely. With these tools at hand we can make predictions and we can control if our endeavours were worthwhile.

Error distribution

It has been shown for chaotic systems, that the local Lyapunov exponents differed considerably depending on the regions of an chaotic attractor. A large Lyapunov exponent means that even initially very close trajectories will within a few time steps drift apart and become dissimilar. Since our prognoses are based on these similarities a large local Lyapunov exponent will lead to a large possible error and the opposite holds true for small Lyapunov exponents. If the regions with small and large Lypunov exponents are identified, one can at the time of prognosis give the size of the error to be expected.

The investigations just mentioned were carried out with the Lorenz model system. In that model system enough data were available to allow for thorough statistical analyses. This is usually quite different for empirical data. We have only a limited number of observations, from which we could draw conclusions. Can we obtain reliable information from small numbers of data? In general the error declines proportional to the square root of the number of data.

Improved error estimation

We could try to apply a bootstrap procedure on our error data, which means to repeatedly draw samples with replacement. The bootstrap is a method which makes efficient use of computing power. The method has been developed quite a time ago by Bradley Efron, but only recently it has been verified mathematically and accepted by the society of statisticians. The bootstrap is particularly valuable if applied to small sample data. If done carefully it can provide a good approximation of the true distribution. Theoretically we could take an infinite number of samples, but this is neither desirable nor necessary. Usually the estimate of the true distribution will converge, which means it will become more and more stable until only minimal fluctuations remain. We could draw as many samples as necessary until we have arrived at such a stable estimate. The absolute error values may not be very reliable, but the relative error distribution should come close to the true situation.

How to apply the nearest neighbour predictor

We can do all the preprocessing with a spreadsheet program. Let us call this the processing spreadsheet. It contains the formulas for the various calculations and transformations. The complete processing spreadsheet for the DOW JONES predictor will we provided together with this manual.

A	B	C	D	E	F	G
			1st dim	2nd dim		3rd dim
date	index	trend orig.	trend aver.	non lin. smooth.	1st diff. of 1st dim	averages of 1st diff

To the right of column G we will copy the parameters used for nearest neighbour determination.

(-------- input values ----------)				output value
dim1 (t-5)	dim1 (t-4)	dim2 (t-4)	dim3 (t-6)	dim2 (t)

We want to forecast trend changes four time steps ahead. Therefore the values on which we base our predictions must be at least four time steps away from the value to be predicted, see discussion above. In this example we will base our predictions on the four values, named input values. The predicted value should be as close as possible to the output value. The letters dim1, dim2 and dim3 refer to the data column from which the values are taken. while the letters in parentheses denote the number of time steps between input and output value. For example (t-5) denotes a value to be taken five time steps earlier than the value to be predicted. It is apparent that our input values contain the most recent values from each of the three dimensions plus one value from the preceding time step. We do this because these values contain the most information about the output value.

Sometimes other input values may be found superior. Often the values from two dimensions suffice. For example dim1(t-5), dim1(t-4), dim2(t-5), dim2(t-4) has been a good combination on many occasions.

If the final form of the predictor has been set up and only actual values have to be added regularly, one can extract a sub spreadsheet from the original processing spreadsheet (DJIAPROC.WK1), which contains only the last say forty data sets and all formulas. We can call this our update spreadsheet (DJIAUPDT.WK1). In this spreadsheet we can work with new data and perform all calculations of the values for the three dimensions including the back calculation. From this spreadsheet we can copy the new values and add them to our data base in the predictor spreadsheet (DJIAPRED.WK1), to be dealt with in the next section.

Creating a predictor spreadsheet

The predictor will be placed on a separate spreadsheet. For this purpose we copy only the v a l u e s , not the formulas of our input and output data into a new spreadsheet. We usually start with row 10 as first data row and we place the input data in column B..E and the output in column F.

We will provide the complete predictor spreadsheet (DJIAPRED.WK1) together with this manual. But in case you are in doubt what is what and where to find everything we will describe it here.

Column A is reserved for the numbering of the data sets. Our data are in descending order, which means that the most actual values are on top and the values of the distant past are at the bottom.

Our distance calculations will be performed in column G. Column H through L will contain the calculations of the nearest neighbours. We reserve column H for the very nearest neighbour, columns I and J contain linear averages of three and four nearest neighbours respectively. Columns K and L contain weighted averages of three and four nearest neighbours. Columns farther to the right are reserved for the calculation of error and prognostic gain.

We want to test our nearest neighbour predictor and find out how it would have performed if we had used it during the last half year.

Column A was left free. This columns is reserved for a list of data set numbers, which we will set up now. We write into cell (A10) the following formula: (A9+1). And a 1 will appear in the cell. We copy this formula into the range A10..A1000 and we get a column of numbers running from 1..1000. Then we transform the formulas into values by copying the values of A10..A999 into this range. The list may be longer, but it must not be shorter than the length of the data base.Into cell (G10) we write the following formula:

@ABS(B10-B10)+@ABS(C10-C10)+@ABS(D10-D10)+
@ABS(E10-E10)

The formula is copied into all cells in the range (G10..G999). What does this formula mean? We ask the program to compare the cells in columns B,C,D,E of all data sets to the set in row ten. The $-signs denote an absolute address, as compared to the relative addresses in most of the other formulas. The differences in B,C,D,E are summarized and copied into column (G) of the respective data set.

Sorting the data sets according to distance

Our reference data set is the one in (A10) ... (G10). Now we will make use of the data base function of the spreadsheet program. First we sort all data sets in the range A10..G999 according to their distance to the reference data set. Therefore we ask the program to use column (G10..G999) as first (and only) sorting key.

Please do not take these numbers too literally. We will probably not have 990 data sets and therefore part of the range will be empty i.e. hold no data. Therefore the actual sorting range has to be adapted to the size of the data base.

We will be asked in what order, ascending or descending we wish to have the data sorted. We choose ascending order, which means that the data sets with the smallest differences are on top and vice versa. The data set #1 will remain on top because its distance to itself is zero.

Computing the output of nearest neighbours

In row 11 we will find the very nearest neighbour, in row 12 the next one and so on. Column F contains the outputs. The desired output is of course the one in cell (F10). The outputs of the nearest neighbours will be computed in cells (H10)..(L10). The cell (H10) which is reserved for the output of the very nearest neighbour will simply take over the value from cell (F11). Cells (I10) and (J10) will calculate linear averages from cells (F11..F13) and (F11..F14) respectively. Cells (K10)

and (L10) will calculate the averages of the outputs (F11..F13) and (F11..F14) weighted by their inverse distances (G11..G13) and (G11..G14), see above for the weighting formula.

It is very important to pay attention that nearest neighbours which are less than four data set numbers apart from the reference set are excluded from the calculations of nearest neighbour outputs, because for real prognoses four weeks ahead these data sets would not be available.

If the calculations are valid we can copy the formulas of cells (H10)..(L10) into the next row, range (H11).. (L11). When this has been done we transform the formulas in cells (H10) .. (L10) into their values. The sequence of events is important.

Resorting the data sets according to data set number

When we started with our nearest neighbour calculations the data sets were already ordered according to their set numbers. Meanwhile they are ordered according to their distance to data set #1. Now we can calculate the nearest neighbours of data set #2. But first we have to reorder the whole database according to the database numbers. We have to change the range to be sorted into (A11..G999) and we have to change the sorting key into (A11..A999). Next we have to change the formula in cell (G11) because now the neighbourship to data set #2 has to be calculated. We only need to change the absolute address of the row from $10 to $11. This address occurs four times in the formula and we have to change them all. Then we have to copy the contents of cell (G11) into the range (G11..G999). The data are now sorted in original order and their distance to data set #2 has been calculated. The next step will be the sorting according to distance as described above.

One has to calculate the nearest neighbour predictions for many occasions in the past before a reliable estimate of their usefulness can be obtained. We do this for at least a half year period. Should the predictor give satisfactory results for the test phase it can be tried with new data. Cross model validations are necessary, but they can be biased if only a small number of data is collected. If the predictor is regularly applied it should be considered to alter the data set numbering to facilitate the addition of new data sets. One could number the sets in descending order from say 1000..1 or one could simply sort according to date. If the spreadsheet cannot sort according to date, we can code the date as integer, say 990122 would mean 22nd January 1999. There will be no problems in the year 2000, if we change from six to seven digits. We type 1000101 for 1st January 2000. If we write the date like this and sort according to date in descending order, the latest data set will always be on top. And if the data are sorted according to distance it would be quite informative to know the date of the nearest neighbours.

Back calculation of trend values

The back calculations will be performed with the updating spreadsheet (DJIAUPDT.WK1). In this spreadsheet we have removed some columns which were contained in the processing spreadsheet, because they are not needed for updating or back calculation of values. The columns missing in the update sheet are: column C of the processing sheet, which held the logarithm of the index, column D which contained the index changes within 12 weeks and column J which contained the weighted averages as optimized with genetic algorithms. Finally the formulas for counting direction changes were omitted. So that the update sheet looks just like the scheme on page 62.

The output value of our predictor is a trend, but what we really want to know is the index value (price, exchange rate). Therefore we have to back calculate the trend values into the desired form. A correct calculation would involve three steps.

a) back calculation from 2nd dimension to 1st dimension
b) back calculation from first dimension (smoothed trend) to
   the (unsmoothed) 9..13 weeks trend.
c) from the 9..13 weeks trend we can calculate the index.

The back calculation formula from second dimension to first dimension is given below.

```
(-(D14*2133+D15*1792+D16*1551+D17*1313+D18*1263
+D19*1174+D20*1169+D21*1159)+E10*(10000+2133+
1792+1551+1313+1263+1174+1169+1159))  / 10000
```

In the updating spreadsheet (DJIAUPDT.WK1) supplied together with this manual you can see that the back calculation of the first dimension from the second dimension as performed in cell H10 returns the right value. Contents of D10 and H10 are identical.

Back calculation of unsmoothed trend from smoothed trend comes next. We do this here because we need the value for the unsmoothed trend in the next step to show that the formula for back calculating the index value is correct. Note that I10 and C10 contain the same values.

Back calculation of (smoothed) index

In the next step we back calculate the index value. The formula to be used is:

@EXP(h10)*@AVG(b19..b23)

In the update spreadsheet you see that the values in J10 and B10 are identical. In both cases it is the true index value. But this is only a demonstration that the formula would return the right value, if we would feed it with the unsmoothed trend. For our prognoses we work with smoothed trends as already mentioned above. Therefore we can only back calculate sort of an average index. As mentioned at an earlier occasion averaging shifts the maximum correlation between the unsmoothed and smoothed value by one time step to the past. Therefore the actual average correlates best with the unsmoothed value of the previous week. The average index calculated in cell K10 correlates best with the index value in cell B11, as we expected and not with B10.

Example prediction

We have applied this crude predictor (one step, zero order, equal weighting of all inputs) to forecast the trend of the next four weeks, December 3 to December 24, 1998. Data sets used to forecast are numbered -1 ... -4 in DJIAPRED.WK1. The statistical forecast would be that the trend would remain constant, i.e. the trend in four weeks will be the same as the actual one. This statistical prognosis would result in the values given in column E6..E9 in the DJIAPRED.WK1 predictor spreadsheet. The true values are given in the update spreadsheet DJIAUPDT.WK1, also in column E6..E9.

Depending on how many nearest neighbours were considered and if they were distance weighted or not, different values were predicted. You find them in the prediction spreadsheet DJIAPRED.WK1 in the columns from H6..H9 for a single nearest neighbour up to L6..L9 using four distance weighted nearest neighbours. No matter which of the nearest neighbour prognoses you look at, all of them would have done better than the statistical forecast. We have not back calculated the index values for two reasons. First, it should be clear that if the trend forecast with the nearest neighbour method is better than the statistical forecast, the back calculation cannot but support this finding. Second, too many data in the spreadsheets would be confusing.

We know that this finding need not be representative of the over all performance. But the example is intended to show that this kind of prediction can work at least in principle, crude as it is, see above.

Let us now look which data sets were used as nearest neighbours for the different predictions. Note that the four predictions for one, two, three and four weeks ahead of Nov 28, 1998 are altogether examples of four week prognoses. Because for all these forecasts only such information was used which was at least four weeks old.

Date:	1203	1210	1217	1224	
Nearest	week	week	week	week	
Neighb.	# 1	# 2	# 3	# 4	
------	----	----	----	----	
1st	757	411	410	842	) data set
2nd	883	881	556	956	) numbers
3rd	756	882	557	740	) in sheet
4th	845	860	409	739	) DJIAPRED

The high numbers of the data sets indicate that the nearest neighbours in state space are far away in time. Since these are weekly data which start at number 1 for the most recent data set, the set number 410 is almost eight years back, and this is the neighbour closest in time. The second nearest neighbour for the prognosis of the week ending on 1224, was number 956 which is almost 19 years back in time.

You will find the nearest neighbours given in the table above, if you sort the data sets according to distance. The data sets listed in column of week #1 should appear as nearest neighbours of row B9..E9,. The sets listed in the column of week #4 should be the nearest neighbours of the data set in B6..E6. The other nearest neighbours should appear if sorted according to distance to B7..E7 and B8..E8 respectively. We give you the table to compare those data to the results of your efforts in following our instructions how to use the predictor.

This crude predictor certainly has its shortcomings. There is still room for improvement as regards to weighting of inputs, higher order (first or second) of predictor, and iterated prediction. Further improvements are possible with a longer data base.

The whole procedure of nearest neighbour prediction using a spreadsheet, as described here, is somewhat laborious. Though if one is accustomed to it the evaluation of a predictor with weekly data for a half year can be done within thirty minutes. This was our experience with a data base of about a thousand data sets and on an i386 without numerical co-processor. However, once the experimental phase is over it would be worth pondering if the whole procedure should be coded into a stand alone computer program.

Summary

We have explained briefly why statistical methods are not ideal for prognoses of market movements. And we described how methods developed for chaos research can be applied to discover order in seemingly random market data series. Procedures suitable for extracting the market dynamics were shown. And a local predictor based on 'nearest neighbours' was introduced. Once established the predictor can be applied using a spreadsheet program. The example shows that simple means suffice to set up a useful predictor. This is not the most elegant way to use it, but it works.

The methods were applied to real world data, i.e. the time series of weekly closes of the Dow Jones industrial average. Markets behave similarly which means that their complexities are of the same order of magnitude and the association between past and future data is comparable as well. The example discussed here in some detail is quite typical of markets in general. Therefore the experience gained with this market can facilitate efforts to develop predictors for other markets if desired. For which markets data are made available will be mentioned in the appendix.

Predictions with better than statistical accuracy are limited to short periods. Just how short is difficult to answer. As argued above, the prediction error is not uniformly distributed over the state space of a dynamic system like a market. The Chi²-Test shows that movements in our example market were non random for a period of 11 weeks. But this number is an average and does neither represent the lower nor the upper limit. As a rough estimate we would say that under favourable conditions a prediction over six months could be achieved. But we doubt that even optimal conditions would allow longer forecasts that deserve the name.

Brief discussion of the chaos theoretical approach versus fundamental analysis and 'technical analysis'.

How do predictions based on chaos theory compare to those based on fundamental or 'technical' considerations? The forecasting methods described above are based on thorough analysis. However, the type of analysis applied does not use any concept of value, which is indispensible for fundamental analysis. Furthermore this approach does not assume any rational behaviour of market participants as required by econometric models. And unlike fundamental analysis, economic and financial parameters other than the market price are disregarded in our analysis and modelling. In that respect it resembles 'technical analysis'.

The chaos theoretical approach is built around the concepts of information and complexity. Serial information is not only extracted from the available data, but also transformed into parallel information. Which means transforming time coordinates into space coordinates. This transformation is necessary since the self similarity of chaotic trajectories becomes only evident in space, but neither in the time domain, nor in the frequency domain. How much information in parallel is needed in each case for optimal information extraction, will depend on the complexity of the system. Several objective criteria are available for measuring information content and complexity. Despite the beauty of fractal graphics chaos research is serious and solid science. It is firmly rooted in topology and information theory.

'Technical analysis' on the other hand does not use any concept at all. It is an empirically more or less justified collection of statistical and geometrical tools, for the application of which there exist neither objective criteria nor a theoretical background. Therefore scientists of economics and finance criticized it as being subjective and not serious. 'Technical analysts' have always denied the 'random walk' theory. It seems that they pursued the right idea, however, their means were insufficient. Now with chaos theory and its sophisticated tools available there seems to be no need any more for 'technical analysis'.

Appendix

The time series data listed below will be made available on a disk, which we supply together with this instruction manual. You will find data for 15 stock market indices, 6 foreign exchange rates against the US-$ from which you may compute 15 different mutual exchange rates. There are 2 commodities and 2 financial futures, 5 short term interest rates (3 months) and 5 long term interest rates (10 years government bonds). Altogether these are more than 40 different individual markets, about 30,000 individual data.

Stock markets	Data (approx. numbers)		
S&P 500	1000		
DJIA	1000		
France	1000	6000	
DAX	1000		
FTSE100	1000		
Nikkei	1000		
Australia	800		
India	800		
Hongkong	800		
Korea	800		
Malaysia	800	7200	
Philippines	800		
Singapore	800		
Taiwan	800		
Thailand	800		
Currencies			
DM/$	1000		
Yen/$	1000		
French Franc/$	1000		
Swiss Frank/$	1000	6000	
$/Ecu(Euro)	1000		
Br. Pound/$	1000		
Commodities			
Gold	800		1600
Oil	800		
Financial Futures			
T-Bonds	800		1600
T-Bills	800		
Interest rates (3 m)			
DM	800		4000
French Franc	800		
Br.Pound	800		
Yen	800		
Swiss Frank	800		
Interest rates (long term government bonds)			
DM	800		
French Franc	800		
Br.Pound	800	4000	
Yen	800		
Swiss Frank	800		

Data sources

The data come from various sources, as listed below. Further information about these and other data of our data base may be obtained from the author: Hans.Uhlig@hamburg.de.

See also http:// www.hans-uhlig.de

1 - Economist
2 - Financial Times
3 - Barron's
4 - Far Eastern Economic Review
5 - Asian Wall Street Journal
6 - Finanz & Wirtschaft
7 - Frank Mella: "Dem Trend auf der Spur".
Verlag Boersenzeitung, Frankfurt/Main, 1988.
(calculated DAX values for the time before July 1988,
when it was officially recognized as the standard
stockmarket index for Germany)

For some countries the stock market indices changed over time. In those cases the indices were concatenated and transformed on the basis of overlapping data.

We cannot exclude typing errors and/or missing data. So if you use the data you should be careful. Anyway you will do so at your own risk. We refuse any responsibility for them being correct and/or complete.

References

Here we give two references for the complexity, i.e. number of dimensions of financial markets.

Matt Ridley: The Mathematics of Markets. A survey of the frontiers of finance. THE ECONOMIST, 9th October 1993. Reprints of this survey can be obtained from THE ECONOMIST Newspaper Ltd, 25 St. James Street, London SW1A 1HG, United Kingdom.

Reading of this survey is highly recommended. It deals with all kinds of financial markets and reviews a number of recent efforts to improve market predictions.

Paul De Grauwe, Hans Dewachter and Mark Embrechts: Exchange Rate Theory - Chaotic Models of Foreign Exchange Markets. Blackwell Publishers, Oxford, United Kingdom, 1993.

This book focuses on foreign exchange markets. It provides a good review of econometric market models, fundamental and chartist methods to predict the market. The methods of chaos theory and their scientific background are explained. Market dimensions are estimated and two kinds of market models are developed: a chaotic model and a periodic model with stochastic shocks.

We have coded these two models into C-programs. Executables and the source code of which are included on the program disk (Disk # 1) enclosed.

Bärbel Finkenstädt: Nonlinear Dynamics in Economics - A Theoretical and Statistical Approach to Agricultural Markets. Lecture Notes in Economics and Mathematical Systems Vol. 426, Springer Verlag Berlin, Heidelberg, 1995.

The book analyses commodity prices and looks for non linear dynamics. The author applied a simple nearest neighbour method to forecast commodity prices and showed its superiority over optimal conventional linear statistical methods (ARMA).

More references could be given of course, but here we mentioned only those which deal with financial markets and we tried to provide examples for different markets.