



Neuromodels of analytic dynamic systems

by

D. C. Dracopoulos and Antonia J. Jones

Department of Computing, Imperial College London

DEPARTMENT OF COMPUTING
IMPERIAL COLLEGE OF SCIENCE,
TECHNOLOGY AND MEDICINE
UNIVERSITY OF LONDON
180 Queen's Gate, London, SW7 2BZ
Telephone: 071-589 5 111 Ext: 5017/5096
Telefax: 071-581 8024
E-Mail: ajj@doc.ic.ac.uk

Date/version: 5 August 2008

To appear in: Neural Computing & Applications 1(4).

Copyright © 1993, D. Dracopoulos and Antonia J. Jones

D. Dracopoulos and Antonia J. Jones: *Neuromodels of analytic dynamic systems*.

Abstract. In contrast to recent work aimed at using neural networks for relatively 'long term' prediction of time series, this paper examines how neural networks designed for short term prediction can form very good approximation models, valid over a large region of the phase space, after having been trained on as few as 500 points *from a single trajectory* of the underlying dynamic system. This is illustrated using four dynamic systems of increasing complexity, including a simple chaotic system and a more realistic system describing rigid body rotation using orthogonal torques under a chaotic control regime.

Keywords. Neural Nets, Dynamic Systems, Modelling, Euler equations, Chaos, Prediction.

CONTENTS

Introduction 1

Simple pattern recognition 3

Locally predictive nets applied to simple dynamic systems 4

A two point attractor system 5

The Van der Pol system 5

A simple chaotic system 6

Rigid body rotation 7

Conclusions 8

References 8

FIGURES 10

List of Figures

Figure 1 The phase space, coloured black for points which converge to the right attractor and white for points which converge to the left attractor. 10

Figure 2 A 2-10-10-2 pattern recognising net attempting to predict the target attractor. 10

Figure 3 Two point attractor system. Attempting long range prediction by iteration of the 4-10-2 LPN. Trajectory through (-0.5, 0.5). 11

Figure 4 Two point attractor system. How well the 4-10-2 LPN learnt the training data. The two graphs are virtual indistinguishable. The network is predicting 1 sec ahead. 12

Figure 5 Two point attractor system. Generalisation capability of the 4-20-2 LPN on a trajectory from initial point (-0.5, 0.5), i.e. different from the training trajectory. Network predicting 1 sec ahead. 13

Figure 6 The vector field for the two point attractor system. 14

Figure 7 Two point attractor system. The vector field of the 4-20-10-2 LPN - network predicting 0.01 sec ahead. 15

Figure 8 Van der Pol system. How well the 4-10-2 LPN learnt the training data. The network is predicting 0.1 sec ahead. 16

Figure 9 Van der Pol system. Generalisation capability of the 4-10-2 LPN on trajectory starting at (0.01, 0.01), i.e. different from the training trajectory. The network is predicting 0.1 sec ahead. 17

Figure 10 Van der Pol system. A phase plane diagram in which two different trajectories are shown: one starting at (-3,2) which spirals inwards, and the other starts near the origin (0.01, 0.01) and spirals outwards. 18

Figure 11 Van der Pol system. The phase plane diagram for the 4-10-2 LPN for the trajectories starting at (-3, 2) and (0.01, 0.01) respectively. The network is predicting 0.1 sec ahead. 19

Figure 12 Van der Pol system. Iterating the 4-10-2 LPN (for the same trajectories). As expected the performance is poor, but it is interesting that the iterated LPN system also exhibits a limit cycle. 20

Figure 13 Simple chaotic system. Two trajectories with almost identical initial conditions. 21

Figure 14 Simple chaotic system. How well the 8-10-5-2 LPN learnt the training data. Network predicting 0.1 sec ahead. 22

Figure 15 Simple chaotic system. Generalisation capability of the 8-10-5-2 LPN on trajectory starting at (2.01, 3.01). Network predicting 0.1 sec ahead. 23

Figure 16 Simple chaotic system. A Poincaré section. 24

Figure 17 Simple chaotic system. A Poincaré section for the 8-10-5-2 LPN. Network predicting 0.1 sec ahead. 25

Figure 18 Euler system for the chaotic control regime. Two trajectories with almost identical initial conditions. 26

Figure 19 Euler system. How well the 12-10-5-3 LPN learnt the training data (only a part of the training trajectory is shown). Network

D. Dracopoulos and Antonia J. Jones: *Neuromodels of analytic dynamic systems.*

predicting 0.5 sec ahead. 27

Figure 20 Euler system. Generalisation capability of the 12-10-5-3 LPN on trajectory starting at (1.9, 1.9, 1.9) - x angular velocity. Network predicting 0.5 sec ahead. 28

Figure 21 Euler system. Generalisation capability of the 12-10-5-3 LPN on trajectory starting at (1.9, 1.9, 1.9) - y angular velocity. Network predicting 0.5 sec ahead. 29

Figure 22 Euler system. Generalisation capability of the 12-10-5-3 LPN on trajectory starting at (1.9, 1.9, 1.9) - z angular velocity. Network predicting 0.5 sec ahead. 30

Figure 23 Euler system. Iterative prediction of the 12-10-5-3 LPN on trajectory starting at (2.9, 1.9, 0.9) - x angular velocity. 31

Figure 24 Euler system. Divergence comparison (see text). Further confirmation that iterating the LPN leads to poor long term prediction. 32

Figure 25 Euler system. Poincaré section through the x-z plane for trajectory through (3, 2, 1). 33

Figure 26 Euler system. The corresponding Poincaré section through the x-z plane for the 12-10-5-3 LPN. Network predicting 0.5 sec ahead. 34

Neuromodels of analytic dynamic systems

D. C. Dracopoulos and Antonia J. Jones

Introduction.

Nearly all phenomena of the natural world involve systems whose behaviour through time varies. In some cases the rules governing the behaviour are themselves opaque, but in many cases complexity can arise from relatively simple rules. The most primitive biological information processing systems evolved to meet the necessities of survival. From sense data to action, flight or the capture of prey, is a gap that was bridged by the evolution of adaptive control systems based on circuits of simple neural components. A vital computational characteristic of such neural circuitry is the ability to model non-linear dynamic systems.

The mathematical description of dynamic systems is well established, using vector differential or difference equations. For example, a continuous-time dynamic system, without control inputs, can be written as

$$\frac{d\mathbf{x}}{dt} \equiv \dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{c}, t) \quad (1)$$

where $\mathbf{x}(t) \in \mathbb{R}^n$, $\mathbf{c} \in \mathbb{R}^m$ and $f: \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^n$. The vector $\mathbf{x}(t)$ is referred to as the *state* of the system at time t .

When we model a dynamic system like (1) we construct a discrete time system

$$\mathbf{x}'(t_{k+1}) = g(\mathbf{x}'(t_k), \mathbf{x}'(t_{k-1}), \dots, \mathbf{x}'(t_{k-p}), \mathbf{c}) \quad (2)$$

such that the values $\mathbf{x}'(t_k)$ approximate the solutions $\mathbf{x}(t_k)$ of (1) for $k = 0, 1, \dots$. There is a clear relationship between modelling and prediction.

In biological control systems the success or failure of the control action is constantly checked using incoming sense data. We call a neural network which acts in this way a *locally predictive net* (LPN). Similarly, some control theorists might call this a *series-parallel plant identification model*, [Narendra 1989]. In a series of experiments, we train networks which look at the current state, typically for second order systems ($x(t), \dot{x}(t)$), and at a number p of past states

$$(x(t-\Delta t), \dot{x}(t-\Delta t)), \dots, (x(t-p\Delta t), \dot{x}(t-p\Delta t)))$$

of the actual dynamic system and attempt to predict the next state ($x(t+\Delta t), \dot{x}(t+\Delta t)$). In recent years several authors (e.g. [Lapedes 1987], [Broomhead 1988], [Narendra 1990]) have suggested such models. Thus we seek to construct a network to implement the mapping

$$((x(t), \dot{x}(t)), (x(t-\Delta t), \dot{x}(t-\Delta t)), \dots, (x(t-p\Delta t), \dot{x}(t-p\Delta t)))) \rightarrow (x(t+\Delta t), \dot{x}(t+\Delta t)) \quad (3)$$

for some fixed $p \geq 1$. (Of course, there is no particular reason why the sampling interval Δt need be the same as the prediction interval T .) For the purposes of this paper a mapping such as (3) is called an *input-output* model of the underlying dynamic system.

Standard feedforward neural networks, with as few as one hidden layer, using (fixed) arbitrary squashing functions, can approximate to any desired degree of accuracy any continuous function $f: \mathbb{R}^n \rightarrow \mathbb{R}^m$ over a compact subset of \mathbb{R}^n , provided sufficiently many hidden units are available [Hornik 1989], [Cybenko 1989]. This is, of course, an existence theorem and gives no guarantee that any particular training method will converge to the required approximation, nor any indication of the number of hidden units required. However, it is an important result. These results depend essentially on the Stone-Weierstrass theorem which asserts that an algebra \mathbf{A} of real *continuous* functions, that separates points on a compact set K and does not vanish at any point of K , is dense in the space of real continuous functions on K .

While empirical tests and simulations show such structures to be effective identification models, it has not been clear how universal these models are and whether they can describe the input-output behaviour of general dynamical systems. The flow $\varphi(t)$ of a chaotic system, which may occur in a high dimensional space Λ , evolves to a compact attracting manifold Λ^* of lower dimensionality d . The chaotic attractor Λ^* often has a *fractal* structure for which d is not an integer [Mandelbrot 1982]. A frequently quoted embedding theorem [Takens 1980] establishes the existence of such models for homogeneous systems: there exists a function ψ that satisfies the mapping (3) with $p \in (d-1, 2d)$. For non-homogeneous systems, however, previous results only stated sufficient conditions for the existence of such models in a restricted region of operation around an equilibrium point [Leonartitis 1985]. Recently, using some fundamental concepts from control theory and differential topology, Levin [Levin 1993] established the existence of global input-output models for a general class of nonlinear systems, i.e. that for any member of the class (3) holds for some *finite* p .

In this paper when we speak of 'modelling a dynamic system', we are interested in relatively accurate prediction of analytically smooth systems for a short time ahead in the case where we have continuous sensory update. It is important to clarify that we are *not* considering here the problem of obtaining a prediction of the value of a time-series for a relatively 'long time' ahead. Long term prediction is a different and arguably far harder problem, which has been addressed by a number of authors and has attractions because of its obvious applications. Of course, any improvement in long term prediction (whether it be by a single application of a network, or by iteration) would probably imply an improvement of the accuracy and predictive range of the results presented here but would not affect the qualitative nature of our observations. In this context we note that we are using data for all the state variables (e.g. for a second-order system x and \dot{x}) not just one of them (e.g. x).

Moody and Darken [Moody 1988] for example, point out that if we take equispaced samples Δt apart and try to predict an interval T ahead then classical techniques like global linear autoregression or Garbor-Volterra-Wiener polynomial expansions typically fail for $T > t_{\text{char}}$, where t_{char} is the inverse of the mean of the power spectrum. They give an example for the Mackey-Glass equation of a localised receptive field network that is able to achieve a mean-squared error $E \approx 0.05$ for $T = 85 \approx 1.7 t_{\text{char}}$. Our own experience suggests that it is easier to obtain reasonably accurate long term prediction by training a network to predict in a single step than by training for short term prediction and then iterating the network. Such iteration tends to lead to a rapid accumulation of errors.

The point we are making here is rather different. The following examples suggest that in many cases training on a single trajectory can lead to a network which has a good local predictive capability (the embedding map) over the whole, or (at any rate) a large region, of the phase space. This implies a surprising (at least to us) generalisation capability for the LPN network. How robust is this phenomenon? We don't yet know. The analogy which springs to mind is analytic continuation: is there an underlying C^∞ approximation theorem here? For example, in the case of the Van der Pol system a network trained on a trajectory external to the limit cycle (the trajectory therefore spirals inwards) nevertheless learns a good LPN model for an internal trajectory (which spirals outwards). It might be argued that we can easily vary the dynamics inside so that the network would then be wrong.¹ The simplest ways would not preserve continuity of derivatives (including higher order derivatives) across the boundary which is formed by the limit cycle. If however we insisted upon analytic (i.e. C^∞) smoothness then a training trajectory near enough to the boundary might pick up sufficient clues from the derivatives near the boundary to get a good model of the interior. What we can say at present is that our examples suggest the existence of such a phenomenon.

¹ This observation was made by T. Leen in conversation.

We consider four different dynamic systems of increasing order of complexity: a two attractor system, a system with a limit cycle, a simple chaotic system, and the rotational motion of a rigid body under a chaotic control regime. For each system we construct a LPN neuromodel and illustrate graphically the extent to which the neuromodel is an accurate local predictor for the dynamic system. Typically we have used a few hundred points from a single trajectory since this is computationally convenient and serves to demonstrate the generalisation capability of the LPN during testing. However, experiments with the same number of training points uniformly distributed over the phase space indicate similar results.

Considerable care is needed to calculate the evolution of the dynamic systems accurately. We have used the Runge-Kutta method. Since we have used fixed time step integration with systems which involve sensitive dependence it is necessary to take very small integration steps where appropriate. Whilst requiring high precision numerical accuracy, our overall aim is to give an essentially qualitative description of the modelling phenomenon and so we shall usually ignore details of units. However, we have quantified time in seconds. The LPN predictions proceed, for the most part, in steps of Δt in the range 0.1 to 1 sec and we write $t = \Delta t k$ ($k = 0, 1, 2, \dots$), where the initial value is $t = 0$.

Simple pattern recognition.

One very straightforward way to use neural networks in this context would be as follows. Take a simple dynamic system with several point attractors and attempt to train the neural net so that given an initial state of the system the network can predict which attractor the trajectory will eventually reach.

Example. A two attractor system.

For example consider the damped oscillator system

$$m\ddot{x} + c\dot{x} - ax + bx^3 = 0 \quad (4)$$

This has equilibria at $x = 0$ and $x = \pm \sqrt[3]{(a/b)}$. Equation (4) models the motion of a mass in a potential field exhibiting two minima. If $c = 0$ there is no damping and there are two kinds of trajectory: those that cross the potential barrier repeatedly and those that simply orbit one of the attractors. If c is small but positive the system is lightly damped, see for example [Thompson 1986]. The points at $x = \pm \sqrt[3]{(a/b)}$ are competing point attractors and in this symmetric problem half of the phase space (x, \dot{x}) leads to one attractor, while the other half leads to the other. There is an unstable equilibrium at $x = 0$. For these experiments the following values were used:

$$m = 2.5, c = 2.0, a = 3, b = 4$$

so that $\sqrt[3]{(a/b)} = 0.866\dots$

Using the substitution $\dot{x} = y$ the differential equation (4) can be transformed into a system of first order differential equations

$$\begin{aligned} \dot{x} &= y \\ \dot{y} &= \frac{1}{m}(-cy + ax - bx^3) \end{aligned} \quad (5)$$

in which form a numerical algorithm, such as the Runge-Kutta method, can be applied.

shows the phase space (x, \dot{x}) for this system with the initial points coloured black if the trajectory is attracted to the righthand attractor, and white if the trajectory is attracted to the lefthand attractor. The diagram was constructed by choosing initial points arranged on a grid and following each trajectory using the Runge-Kutta algorithm. (It is worth observing that for this particular dynamic system there is a much easier method to determine the boundary

D. Dracopoulos and Antonia J. Jones: *Neurmodels of analytic dynamic systems*.

between the black and white zones. One can linearise the system about the unstable attractor at $x = 0$. Solving the linear system gives a good approximation of the direction of the trajectory through this point and iterating this trajectory gives the boundary.)

In this first experiment a 2-10-10-2 network was trained using backpropagation with a learning rate of $\eta = 0.2$ and a momentum $\alpha = 0.7$. Values for (x, x') over the range $-2 \leq x \leq 2$ and $-1 \leq x' \leq 1$ were scaled to $(0, 1)$ and applied to the two inputs and, using one output node for each of the two attractors, the net was trained to predict the correct attractor. The input training data consisted of 861 initial points, arranged as a 41 (horizontal) X 21 (vertical) (step 0.1) grid and the corresponding output values 0/1 associated with each attractor. Training continued for 40,000 iterations. When the training was complete the resulting network was tested to get a high resolution image of 80,601 = 401 X 201 points (step 0.01) and the result is shown in .

This is a rather simple way to use a neural network to predict a dynamic system since it is nothing more nor less than direct pattern recognition. As one might expect it works reasonably well, using the outcome of 861 distinct trajectories to produce quite accurate results near the centre of the region, but these become somewhat less reliable near the boundaries. However, when compared with the granularity of the training data we can see that the *interpolative* power of the network is quite high.

Locally predictive nets applied to simple dynamic systems.

The methodology of direct pattern recognition plainly has very limited utility when applied to more complex dynamic systems. We therefore turn to the issue, posed in the introduction, of getting a network to form an input-output model of the dynamic system.

The choice of Δt depends on the complexity of the dynamic system under consideration. For simpler systems Δt can be large, giving a relatively long term prediction whilst maintaining accuracy. The complexity of a dynamic system can be described in general terms by its Lyapunov exponents [Swinney 1985], which measure the sensitivity of the system to changes in the initial conditions. The Lyapunov exponents provide a coordinate-independent measure of the asymptotic local stability of properties of a trajectory. Given a continuous dynamic system in an n-dimensional phase space, we examine the evolution of an infinitesimal ball of radius $\epsilon(0)$ centred on a point $\mathbf{V}(0)$ in phase-space and its progressive deformation as time unfolds. This deformation is governed by the linear part of the flow. The ball thus remains an ellipsoid. Suppose the principal axes of the ellipsoid at time t are of length $\epsilon_i(t)$. The spectrum of Lyapunov exponents for the trajectory $\mathbf{V}(t)$ is defined as

$$\lambda_i = \lim_{t \rightarrow \infty} \lim_{\epsilon(0) \rightarrow 0} \frac{1}{t} \log \frac{\epsilon_i(t)}{\epsilon(0)} \quad (\text{for } 1 \leq i \leq n) \quad (6)$$

Note the Lyapunov exponents depend on the trajectory $\mathbf{V}(t)$. Their values are the same for any state on the same trajectory, but may be different for states on different trajectories. The trajectories of an n-dimensional phase-space have n Lyapunov exponents. This is often called the *Lyapunov spectrum*. It is conventional to order them according to size. The qualitative features of the asymptotic local stability properties can be summarised by the sign of each Lyapunov exponent; a positive Lyapunov exponent indicating an unstable direction, and a negative exponent indicating a stable direction.

In order to obtain an estimate for a particular trajectory we can average the divergence at many points on the trajectory. For example, considering only the largest exponent we can compute

$$\lambda = \lim_{N \rightarrow \infty} \frac{1}{N} \sum_{j=1}^N \frac{1}{(t_j - t_0)} \log \frac{\epsilon(t_j)}{\epsilon(0)} \quad (7)$$

When $\lambda \leq 0$ the motion is regular, while when $\lambda > 0$, excluding simple counter examples (such as uniform expansion), the motion will tend to be chaotic. For all practical purposes the complexity of the system increases with λ , and the choice of Δt in (3) depends on λ since the magnitude of the exponent reflects the time scale on which

D. Dracopoulos and Antonia J. Jones: *Neurmodels of analytic dynamic systems*.

trajectories diverge. In other words, we have to choose a smaller value of Δt if λ is large for some trajectory of the system.

A two point attractor system.

We first consider the system (4). The characteristic Lyapunov exponent for this system, calculated using the method in [Shimada 1979] is about -0.180 (calculated with base e logarithms). With $p = 1$ and $\Delta t = 1$ we trained a 4-10-2 neural network using backpropagation ($\eta = 0.2$, $\alpha = 0.1$). The training data consisted of 198 scaled points from a single trajectory whose initial point was $(x(0), \dot{x}(0)) = (-1, -0.5)$, $0 \leq t \leq 198$. The network was trained for 30,000 iterations. Thus the network was constructed to satisfy

$$((\mathbf{x}(t), \dot{\mathbf{x}}(t), \mathbf{x}(t-1), \dot{\mathbf{x}}(t-1))) \rightarrow (\mathbf{x}(t+1), \dot{\mathbf{x}}(t+1))) \quad (8)$$

where $t \in [0, 198]$, for this *particular trajectory* of the system (4). The crucial question for system modelling is whether the constructed network satisfies the mapping (8) for all trajectories.

One way to test such a network might be to start it at an initial point $(x(0), \dot{x}(0))$, and iterate the network (with no reference to the actual dynamic system) to see how well it follows the desired trajectory. Given that the system might in general be chaotic any slight error in a local prediction will, if uncorrected and iterated, rapidly lead to significant deviations from the true trajectory. An analogy might be that of a standing man suddenly rendered devoid of sensory feedback - he would rapidly fall over. This is illustrated in in which the iterated trajectory converged to the wrong attractor. However, we should expect such a test to be unsuccessful, since our LPN is constructed to predict only locally in time.

We look for other ways to see how much of the global dynamics a LPN can abstract from a limited set of training examples. We therefore examine the ability of the network to do what it was trained to do, namely predict one step ahead. The performance of the network on the training trajectory is shown in , in which we see very good local prediction using the network when compared with the trajectory upon which it was trained. Of course, this was to be expected and merely illustrates that the network has converged on the training data. Extending the trajectory also showed that the *extrapolation* performance of the LPN was similarly very accurate.

The next step is obviously to test the network on a trajectory different from that upon which it was trained. The result of such a test of *generalisation* is shown in . The trajectory starts at $(-0.5, 0.5)$, $t \in [0, 50]$, and again the agreement is good.

Another way to quickly verify that a LPN can abstract a good approximation to the overall dynamics from observations of a single trajectory is to train a LPN using a relatively small prediction interval Δt . We can then compute the overall differential vector field for both the dynamic system and the LPN (for large Δt this would not be very meaningful) and compare them. This is done in and respectively. The 4-20-10-2 ($p = 1$, $\Delta t = 0.01$) network, trained for 50,000 iterations, here requires more hidden nodes because we seek to model rather small relative changes, but the modelling phenomenon is clearly demonstrated.

The Van der Pol system.

As another example we consider the Van der Pol equation

$$\ddot{x} - \mu(1 - x^2)\dot{x} + x = 0 \quad (9)$$

Again this can be transformed into a system of first order differential equations, so as before we can use the Runge-Kutta method.

D. Dracopoulos and Antonia J. Jones: *Neurmodels of analytic dynamic systems*.

$$\begin{aligned}\dot{x} &= y \\ \dot{y} &= -\dot{x} + \mu(1 - x^2)y\end{aligned}\tag{10}$$

The Van der Pol equation can be regarded as describing a mass-spring-damper system with a position dependent damping coefficient. For large values of x , the positive damping coefficient removes energy from the system and thus the motion has a tendency to converge. However, for small values of x , the damping coefficient is negative and the 'damper' adds energy to the system so that the motion has a tendency to diverge. As a result of this non-linear damping the motion can neither grow without bound nor decay to zero. Instead, it displays a sustained oscillation independent of the initial conditions, a *limit cycle*. It is sustained by periodically releasing energy into and absorbing energy from the environment via the damping term. The characteristic Lyapunov exponent for this system, calculated using the method in [Shimada 1979] is about -0.060.

Thus the system has a stable periodic solution whose period and amplitude depend on the parameter μ . In these experiments we took $\mu = 0.2$. Using a number of experiments we decided that one past state ($p = 1$) of the dynamic system (9) should be used for the LPN with $\Delta t = 0.1$.

We therefore trained a 4-10-2 LPN ($\eta = 0.2$, $\alpha = 0$) on 500 points taken from the trajectory through (-3, 2) for 100,000 iterations. In we see the successful result of the training. In order to test the generalisation capability of this LPN we plot the trajectory that starts at (0.01, 0.01) in . and illustrate phase space diagrams of the actual system and the LPN respectively, for the two trajectories starting at (-3,2) and (0.01, 0.01). It is interesting to note that although the training trajectory is *outside* the region enclosed by the limit cycle, the network nevertheless performs well on the trajectory *inside* the region.

The same network when used for multistep (iterative) prediction gives, as expected, a very poor performance but nevertheless it still has a limit cycle - .

A simple chaotic system.

We now consider a system which exhibits chaotic behaviour

$$\ddot{x} + 0.1\dot{x} + x^5 = 6\sin t\tag{11}$$

This represents a lightly damped, sinusoidally forced mechanical structure undergoing large elastic deflections. In this case, the system is extremely sensitive to initial conditions, i.e. is chaotic.

As before we transform (11) into a system of first order differential equations

$$\begin{aligned}\dot{x} &= y \\ \dot{y} &= -0.1y - x^5 + 6\sin t\end{aligned}\tag{12}$$

illustrates two trajectories with two almost identical initial conditions, namely $(x(0), \dot{x}(0)) = (2, 3)$ and $(2.01, 3.01)$ at $t = 0$. The characteristic Lyapunov exponent for this system, calculated using the method in [Shimada 1979] is about 0.165.

An empirical study of (11) led us to choose $p = 3$, $\Delta t = 0.1$ and a 8-10-5-2 architecture for the LPN. We trained the network using $\eta = 0.2$, $\alpha = 0$ and 500 data points from the trajectory through (2, 3) for $t \in [0, 50]$ for 100,000 iterations - see . The generalisation capability of the LPN for the chaotic dynamics was very good and is illustrated in - trajectory through (2.01, 3.01), $t \in [0, 50]$.

How well does the LPN model the attractor? This can be visualised by constructing the Poincaré section

D. Dracopoulos and Antonia J. Jones: *Neurmodels of analytic dynamic systems*.

(stroboscopic plots at times that are a multiple of 2π) of the dynamic system and of the LPN. Starting with the initial point (2, 3) we calculated the dynamic system ($x(t), \dot{x}(t)$) for $t \in [0, 40,000]$ in steps of 10^{-5} . This took about 36 hours on a Spark 2. A point was plotted every 2π time steps (initial iterations are suppressed).

The results are shown in and . It seemed rather striking that a neural net trained on a single trajectory could abstract such a detailed local model of the overall dynamics.

Rigid body rotation.

In this section, we consider the system which describes the rotational motion of a rigid body. This is determined by a system of first order differential equations (Euler equations).

$$\begin{aligned} I_1 \dot{x} &= (I_2 - I_3)yz + G_1 \\ I_2 \dot{y} &= (I_3 - I_1)zx + G_2 \\ I_3 \dot{z} &= (I_1 - I_2)xy + G_3 \end{aligned} \tag{13}$$

where I_1, I_2, I_3 are the principal moments of inertia, x, y, z are angular velocities about the principal axes fixed in the rigid body, and G_1, G_2, G_3 are torques applied about these axes.

In general the Euler equations are more complicated than Lorenz's system and for certain choices of I_1, I_2, I_3 and G_1, G_2, G_3 exhibit both strange attractors and limit cycles [Leipnik 1981]. Specifically, if we choose $I_1 = 3I_0, I_2 = 2I_0, I_3 = I_0$ with $I_0 = 1$ and

$$\begin{pmatrix} G_1 \\ G_2 \\ G_3 \end{pmatrix} = \begin{pmatrix} -1.2 & 0 & \sqrt{6}/2 \\ 0 & 0.35 & 0 \\ -\sqrt{6} & 0 & -0.4 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{14}$$

the Euler equations produce a binary system of strange attractors for which the attractor of an orbit is determined by the location of the initial point of the orbit. shows the x angular velocity of the system for two trajectories with similar initial conditions, (3, 2, 1) and (2.9, 1.9, 0.9). The characteristic Lyapunov exponent for this system, calculated using the method in [Shimada 1979] is about 0.236.

We choose $p = 3$ and $\Delta t = 0.5$ and train a 12-10-5-3 LPN for 100,000 epochs with 501 data points on the trajectory through (3, 2, 1). The results of the training are shown in . The generalisation capability is tested on all three angular velocities using the trajectory through (1.9, 1.9, 1.9), which is very different from the training trajectory - , , .

Of course, if the trained network is used for iterative prediction then (as expected) we get quite rapid divergence, see . The performance of the iterated network can be compared with the actual system dynamics by plotting first the absolute divergence of the x angular velocity between the system trajectory through (3, 2, 1) and the perturbed system trajectory through (2.9, 1.9, 0.9) (Lower trace labelled *system* in), and second the absolute divergence of the x angular velocity between the system trajectory starting at (3, 2, 1) and the iterated network trajectory starting at (2.9, 1.9, 0.9) (Upper trace labelled *network* in). Whereas the perturbed system begins to diverge significantly after 20 steps, the iterated network on the generalisation trajectory is beginning to diverge significantly after 10 steps.

The Poincaré section for the x-z plane of the system, for the trajectory starting at (3, 2, 1), for the range $t \in [0, 40,000]$ (initial iterations are suppressed) is shown in and the corresponding diagram for the LPN in .

D. Dracopoulos and Antonia J. Jones: *Neurmodels of analytic dynamic systems*.

Conclusions.

We have presented some examples of how locally predictive neural networks can model dynamic systems of different complexity. The results of the experiments seem to us to be remarkable, but may not (of course) extend to highly complex or high dimensional systems. That a neural net can abstract a local approximation of a dynamic system, an approximation to the embedding map, over a large region of the phase space from being trained on a single trajectory suggests that for some classes of systems (perhaps the C^∞ case) the computational process of constructing a good approximation to (3) may not be quite so formidable as might first appear. Given an efficient training procedure very good results can be obtained with a relatively small set of carefully chosen training data. If these observations are a reflection of the general situation for analytic systems it would have some useful consequences, especially in adaptive control applications. This state of affairs contrasts sharply with the method of *persistent excitation* in control theory [Narendra 1989], where to identify a plant the training set input must be rich enough to excite all modes of the dynamical system and the LPN respectively. Plainly some further theoretical investigations are required.

However, there are a number of practical problems associated with using such networks in on-line applications. These can be mainly attributed to our lack of knowledge of specific parameters which depend on the dynamic system. For example, in determining how many past states of a system are necessary in order to predict the next system state, a number of experiments must be tried. Such problems can be faced using improved theoretical understanding perhaps combined with evolutionary methods applied to neural networks.

Acknowledgements.

This work was supported in part by *SERC 90800355* and *SERC GR/J 52471*. We are indebted to Professor P. C. Parks of RMCS, Shrivenham, for several helpful discussions during the development of these results and to T. Leen and J. Moody of the Oregon Graduate Institute for their comments.

References.

- [Broomhead 1988] D. S. Broomhead and D. Love. *Multivariable functional interpolation and adaptive networks*. *Complex Systems*, 2:321-355, 1988.
- [Cybenko 1989] G. Cybenko. *Approximation by superpositions of a sigmoidal function*. *Mathematics of Control, Signals, and Systems*, 2:303-314, 1989.
- [Hornik 1989] K. Hornik. *Multilayer feedforward networks are universal approximations*. *Neural Networks*, 2:359-366, 1989.
- [Lapedes 1987] A. Lapedes and R. Farber. *Non-linear signal processing using neural networks: prediction and system modelling*. LA-UR-87-2662. Theoretical Division, Los Alamos National Laboratory, NM. 87545.

- D. Dracopoulos and Antonia J. Jones: *Neuromodels of analytic dynamic systems*.
- [Leontaritis 1985] I. J. Leontaritis and S. A. Billings. *Input-output parametric models for non-linear systems, part i: Deterministic nonlinear systems*. International Journal of Control, 41:303-328, 1985.
- [Leipnik 1981] R. B. Leipnik and T. A. Newton. *Double strange attractors in rigid body motion with linear feedback control*. Physics Letters, **86A**:63-67, 1981.
- [Levin 1993] A. U. Levin. *Recursive Identification Using Feedforward networks*. Department of Computer Science and Engineering, Oregon Graduate Institute, 2000 NW Walker Road, PO Box 91000, Portland, Oregon OR 97291-1000. Forthcoming publication.
- [Mandelbrot 1982] B. Mandelbrot. *The fractal geometry of Nature*. Freeman, 1982.
- [Moody 1988] J. Moody and C. Darken. *Learning with Localized Receptive Fields*. Proceedings of the 1988 Connectionist Summer School. Eds. D. Touretzky, G. Hinton, and T. Sejnowski. Morgan Kaufmann, San Mateo, CA. 1988.
- [Narendra 1989] K. S. Narendra and A. M. Annaswamy. *Stable Adaptive Systems*. Prentice Hall, 1989.
- [Narendra 1990] K. S. Narendra and K. Parthasarathy. *Identification and control of dynamical systems using neural networks*. IEEE Transactions on Neural Networks, **1**:4-27, March 1990.
- [Shimada 1979] I. Shimada and T. Nagashima. *A mathematical approach to ergodic problem of dissipative dynamical systems*. Progress of Theoretical Physics, **61**:1605-1616, 1979.
- [Swinney 1985] H. L. Swinney, A. Wolf, J. B. Swift and J. A. Vastano. *Determining Lyapunov exponents from a time series*. Physica D, **16**: 285-317, 1985.
- [Takens 1980] F. Takens. *Detecting strange attractors in turbulence*. In A. Dold and B. Eckmann, editors, *Dynamical Systems and Turbulence*, pages 366-381. Springer-Verlag, 1980.
- [Thompson 1986] J. M. T. Thompson and H. B. Stewart. *Nonlinear dynamics and chaos*. John Wiley, 1986.

FIGURES

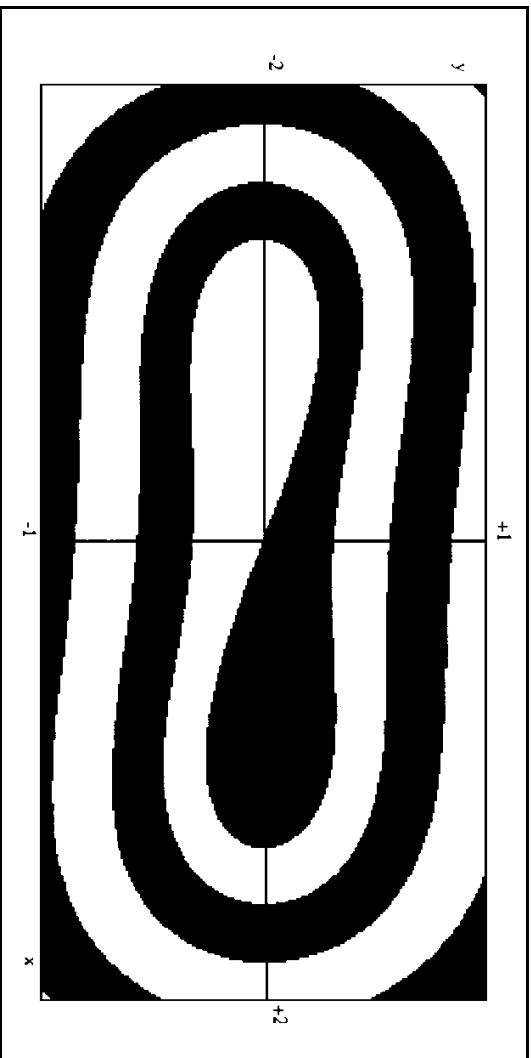


Figure 1 The phase space, coloured black for points which converge to the right attractor and white for points which converge to the left attractor.

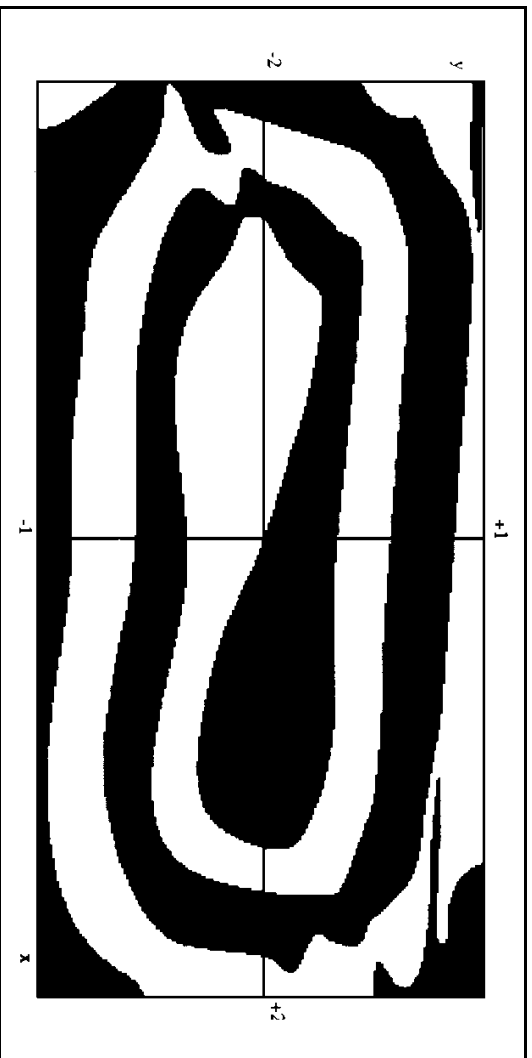


Figure 2 A 2-10-10-2 pattern recognising net attempting to predict the target attractor.

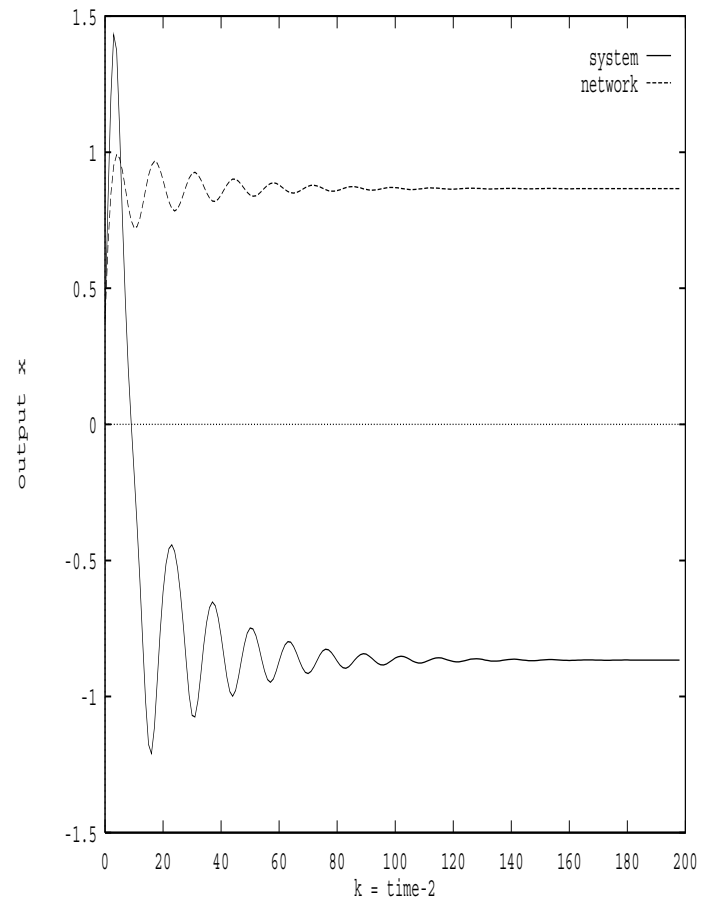


Figure 3 Two point attractor system. Attempting long range prediction by iteration of the 4-10-2 LPN. Trajectory through (-0.5, 0.5).

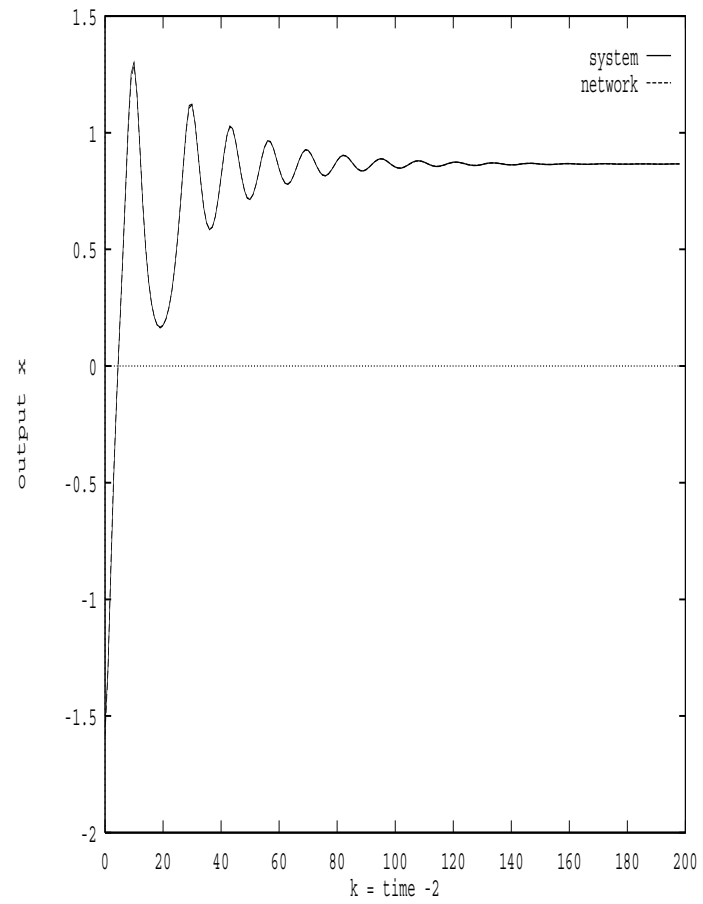


Figure 4 Two point attractor system. How well the 4-10-2 LPN learnt the training data. The two graphs are virtual indistinguishable. The network is predicting 1 sec ahead.

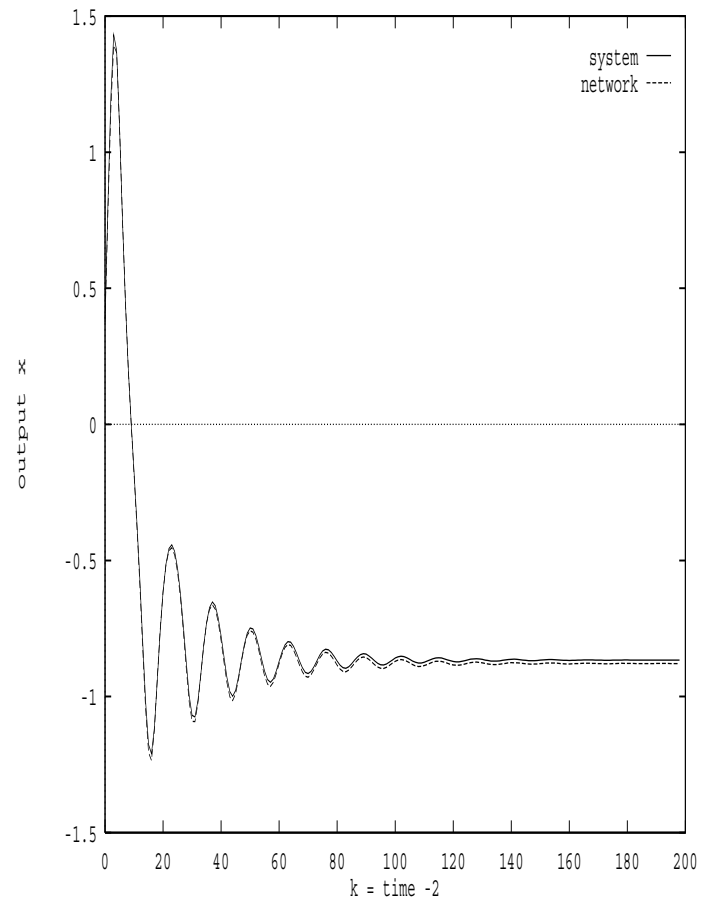


Figure 5 Two point attractor system. Generalisation capability of the 4-20-2 LPN on a trajectory from initial point $(-0.5, 0.5)$, i.e. different from the training trajectory. Network predicting 1 sec ahead.

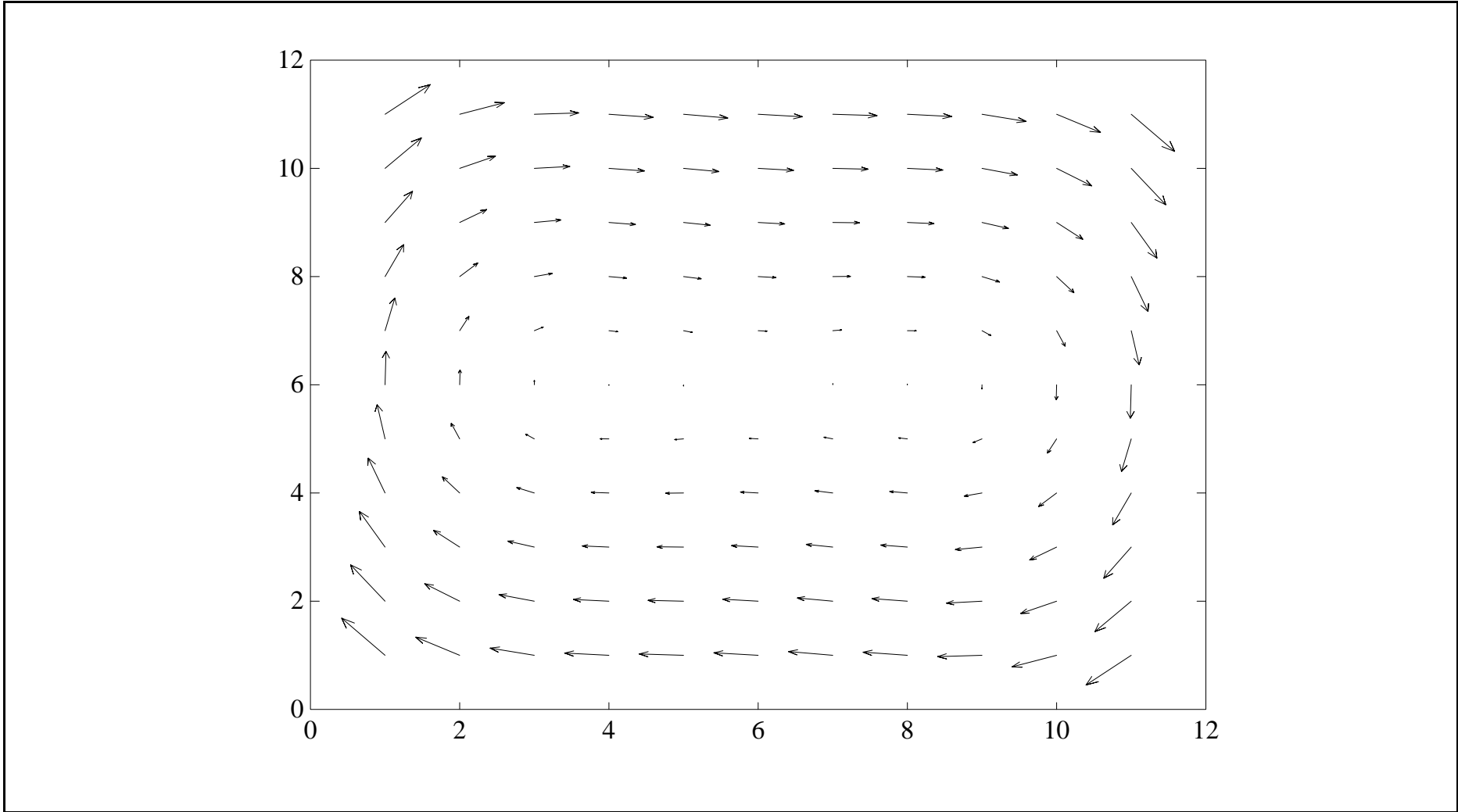


Figure 6 The vector field for the two point attractor system.

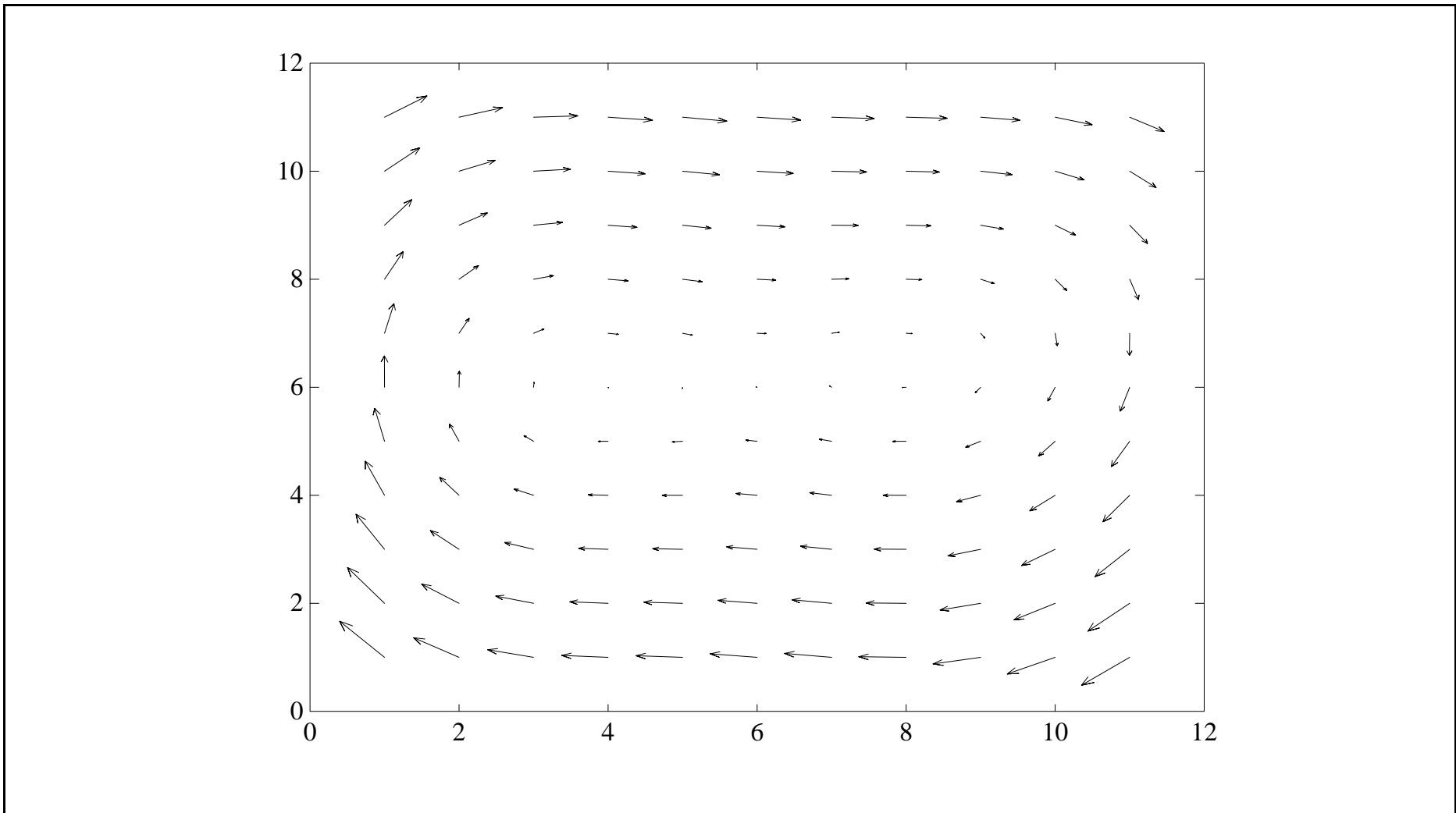


Figure 7 Two point attractor system. The vector field of the 4-20-10-2 LPN - network predicting 0.01 sec ahead.

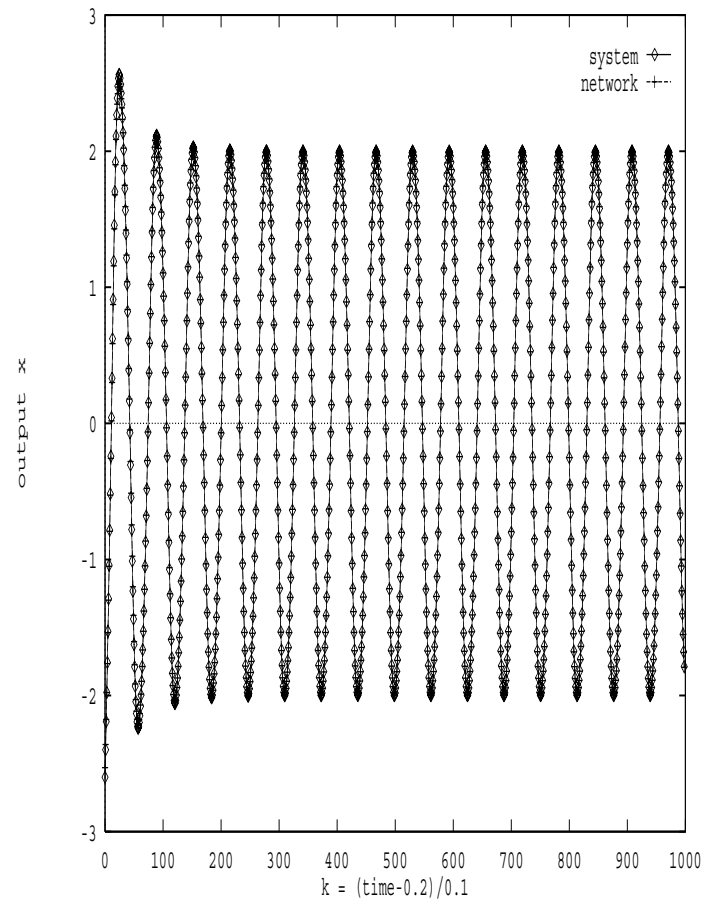


Figure 8 Van der Pol system. How well the 4-10-2 LPN learnt the training data. The network is predicting 0.1 sec ahead.

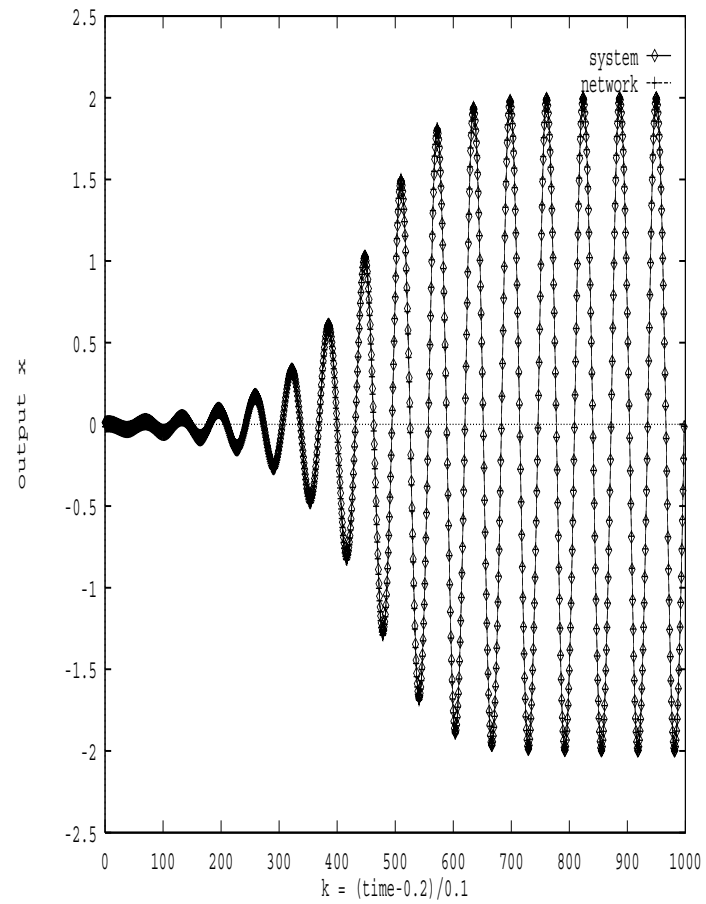


Figure 9 Van der Pol system. Generalisation capability of the 4-10-2 LPN on trajectory starting at (0.01, 0.01), i.e. different from the training trajectory. The network is predicting 0.1 sec ahead.

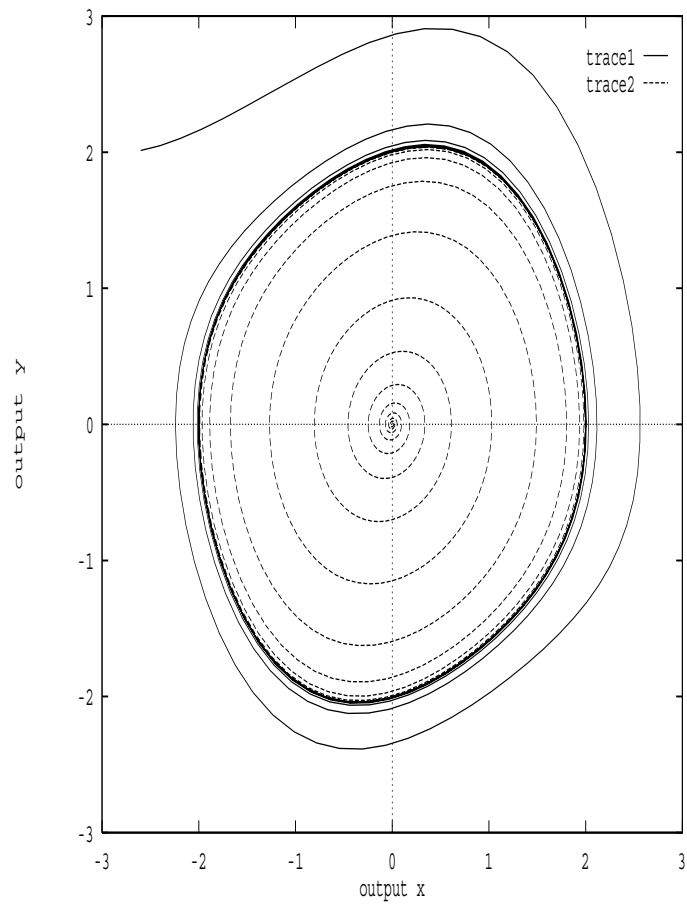


Figure 10 Van der Pol system. A phase plane diagram in which two different trajectories are shown: one starting at $(-3,2)$ which spirals inwards, and the other starts near the origin $(0.01, 0.01)$ and spirals outwards.

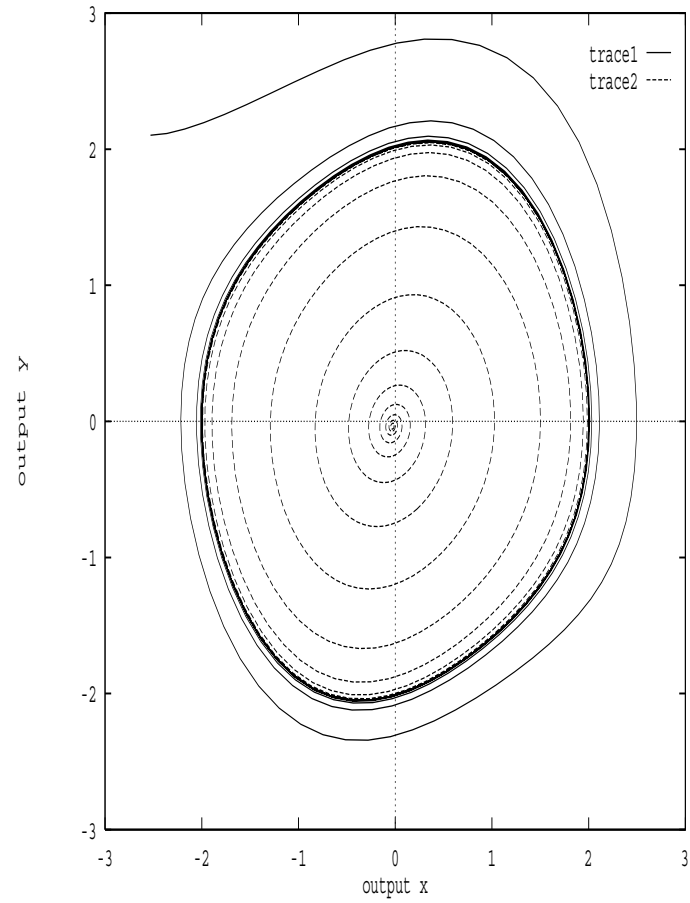


Figure 11 Van der Pol system. The phase plane diagram for the 4-10-2 LPN for the trajectories starting at $(-3, 2)$ and $(0.01, 0.01)$ respectively. The network is predicting 0.1 sec ahead.

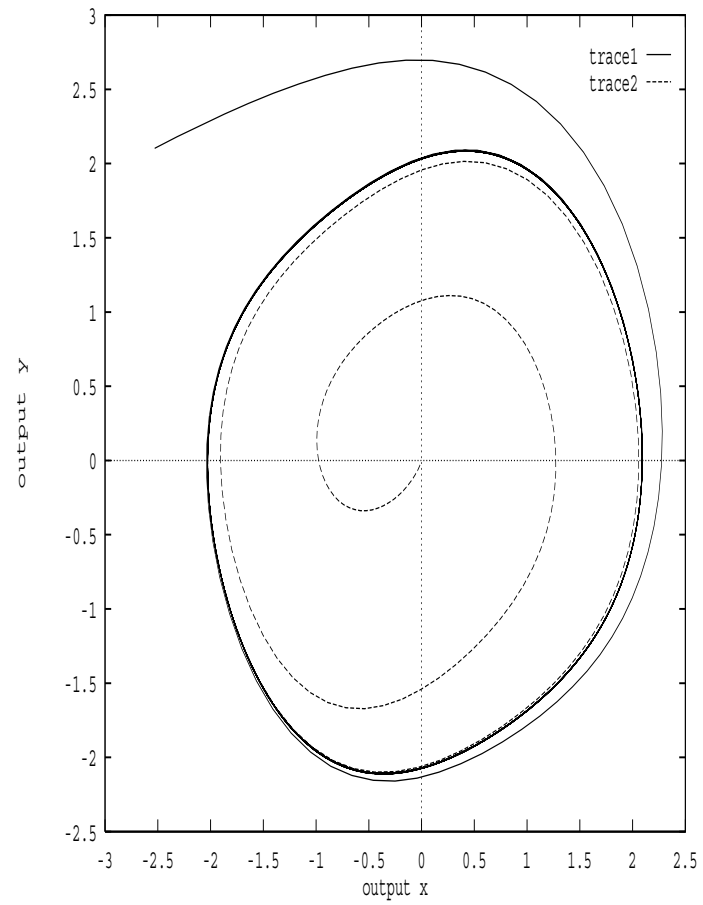


Figure 12 Van der Pol system. Iterating the 4-10-2 LPN (for the same trajectories). As expected the performance is poor, but it is interesting that the iterated LPN system also exhibits a limit cycle.

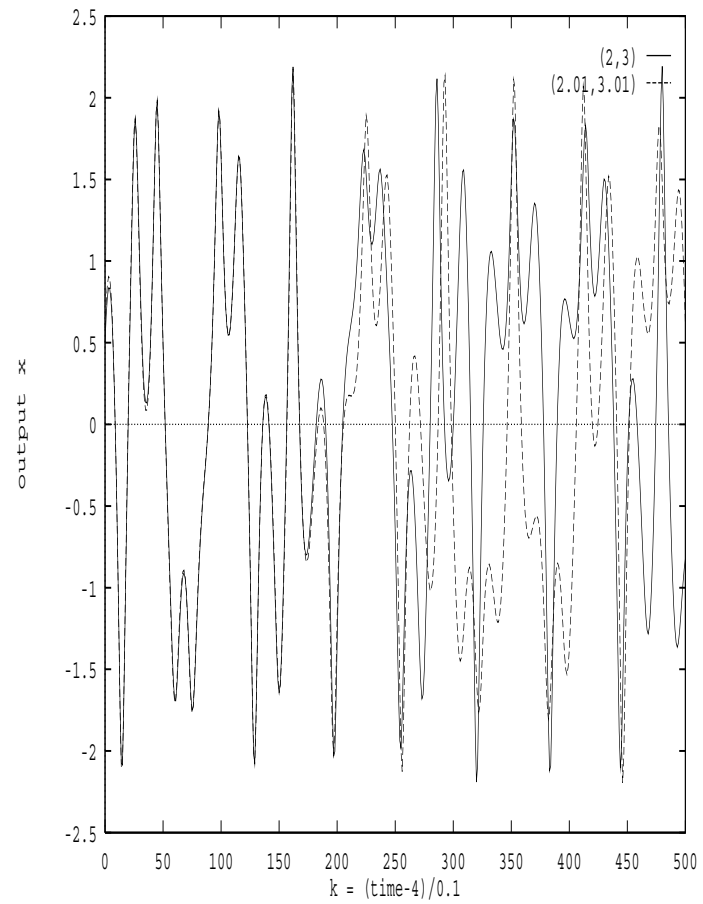


Figure 13 Simple chaotic system. Two trajectories with almost identical initial conditions.

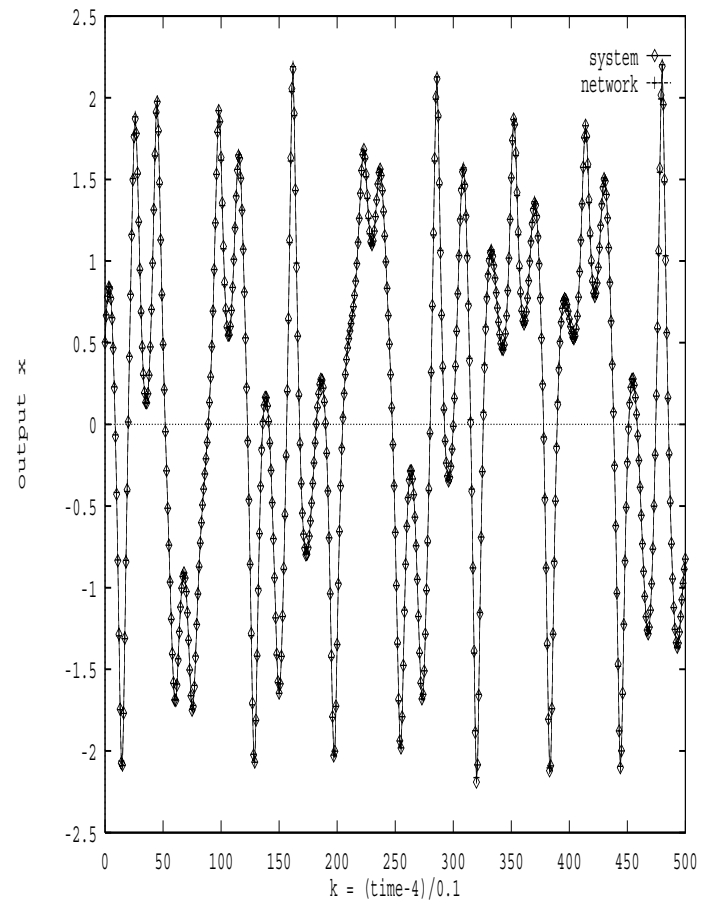


Figure 14 Simple chaotic system. How well the 8-10-5-2 LPN learnt the training data. Network predicting 0.1 sec ahead.

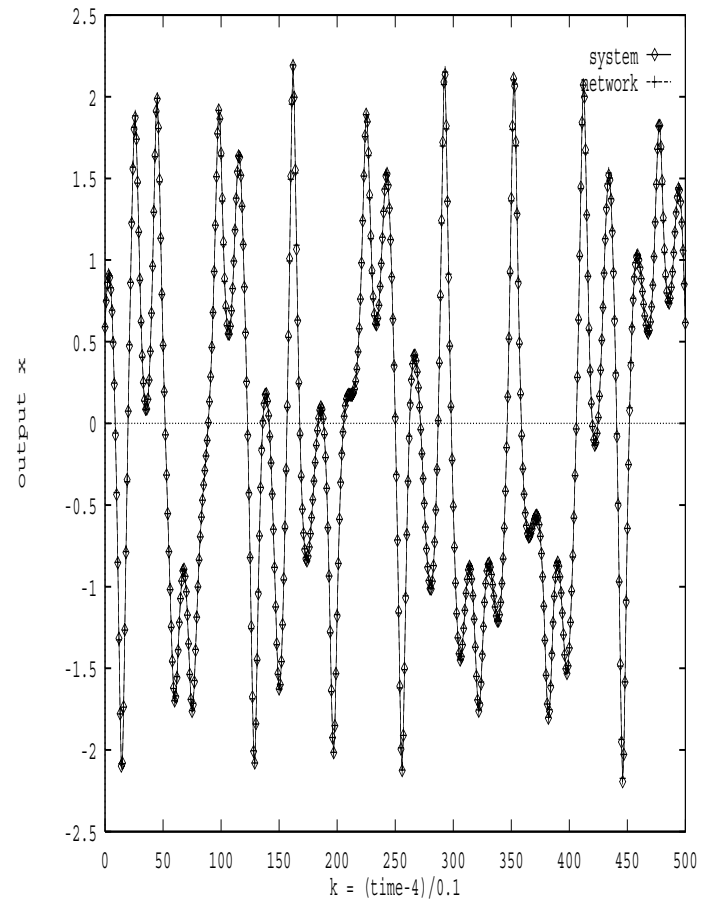


Figure 15 Simple chaotic system. Generalisation capability of the 8-10-5-2 LPN on trajectory starting at (2.01, 3.01). Network predicting 0.1 sec ahead.

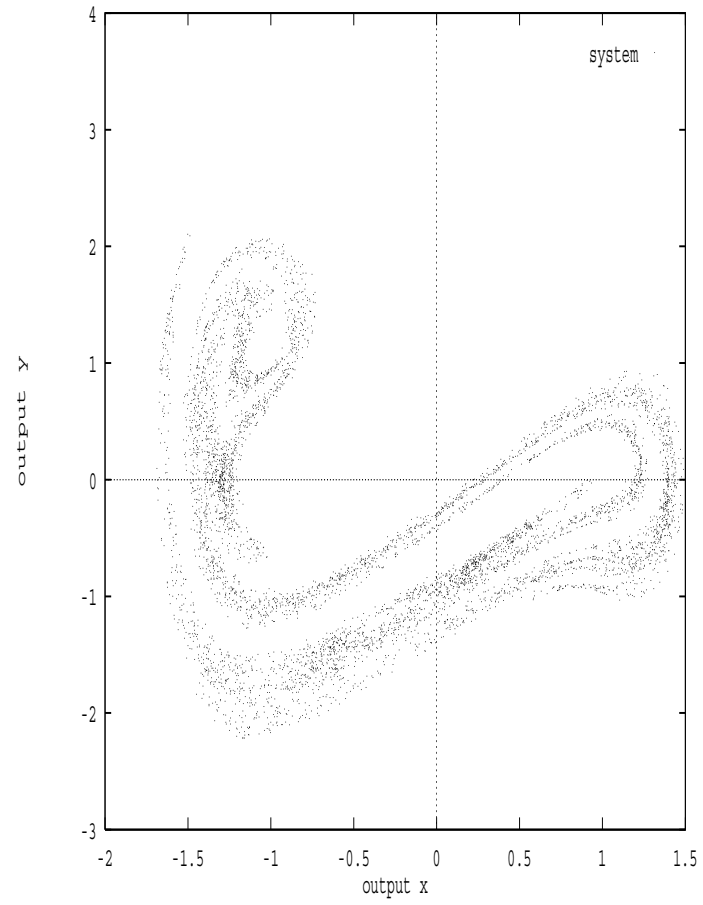


Figure 16 Simple chaotic system. A Poincaré section.

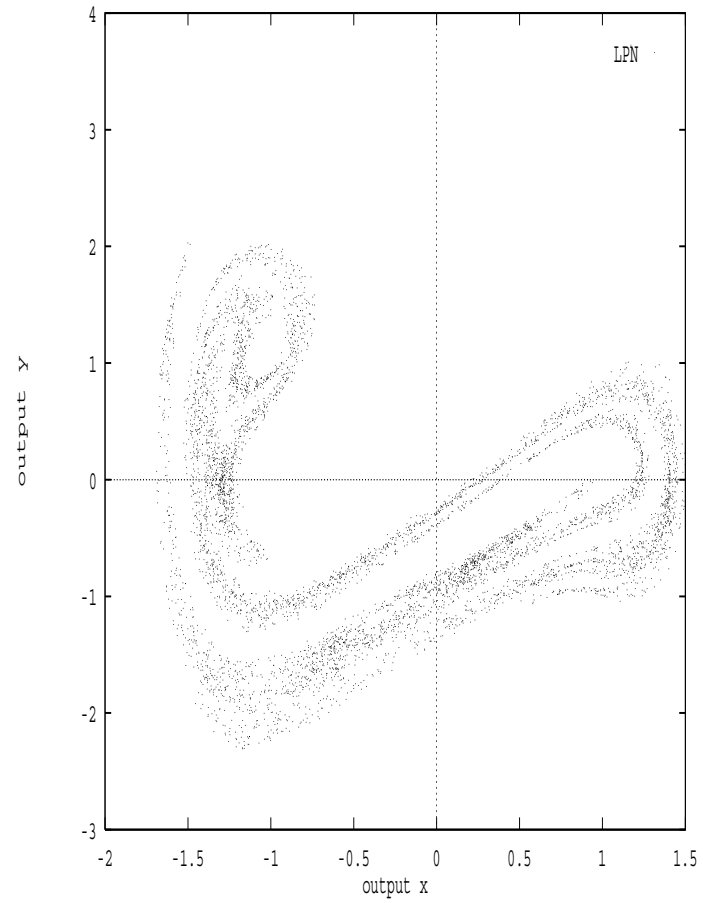


Figure 17 Simple chaotic system. A Poincaré section for the 8-10-5-2 LPN. Network predicting 0.1 sec ahead.

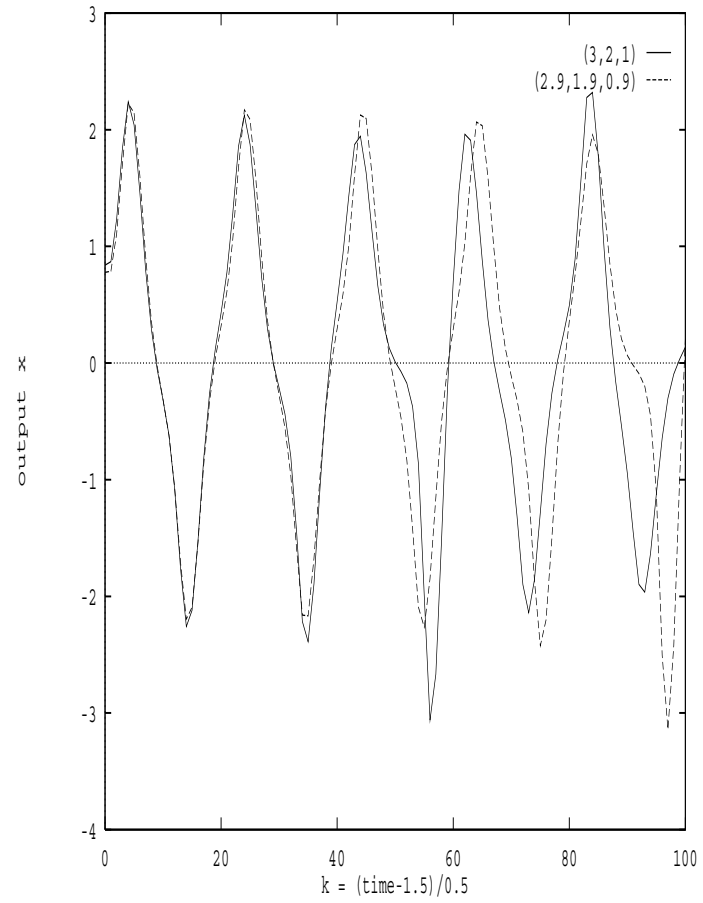


Figure 18 Euler system for the chaotic control regime. Two trajectories with almost identical initial conditions.

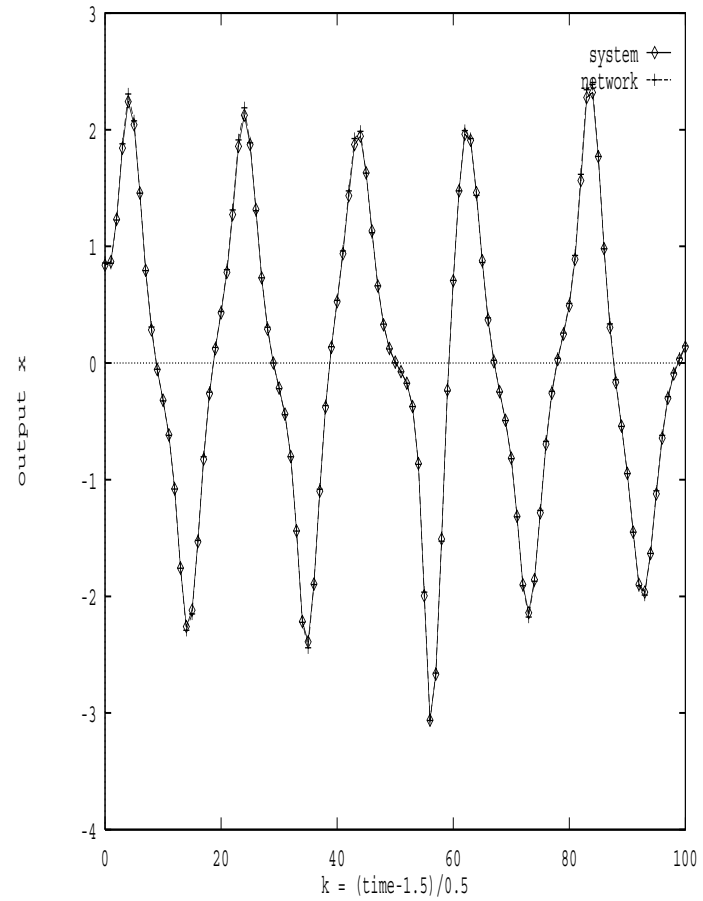


Figure 19 Euler system. How well the 12-10-5-3 LPN learnt the training data (only a part of the training trajectory is shown). Network predicting 0.5 sec ahead.

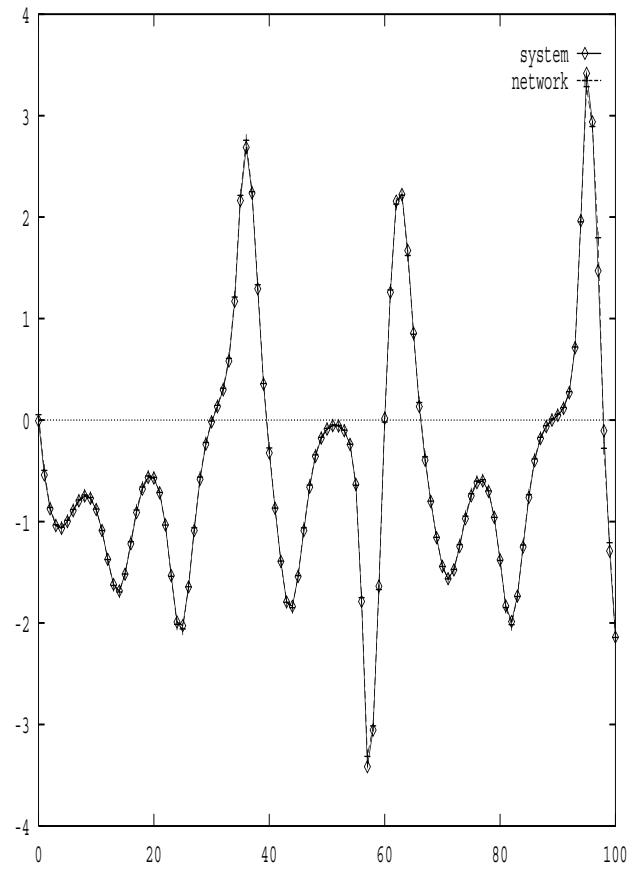


Figure 20 Euler system. Generalisation capability of the 12-10-5-3 LPN on trajectory starting at (1.9, 1.9, 1.9) - x angular velocity. Network predicting 0.5 sec ahead.

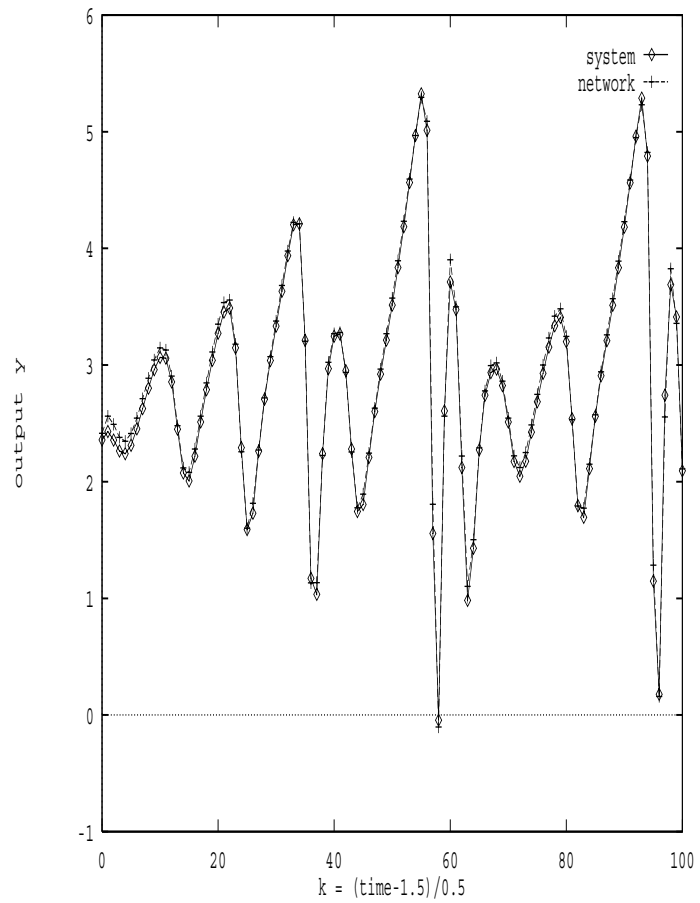


Figure 21 Euler system. Generalisation capability of the 12-10-5-3 LPN on trajectory starting at (1.9, 1.9, 1.9) - y angular velocity. Network predicting 0.5 sec ahead.

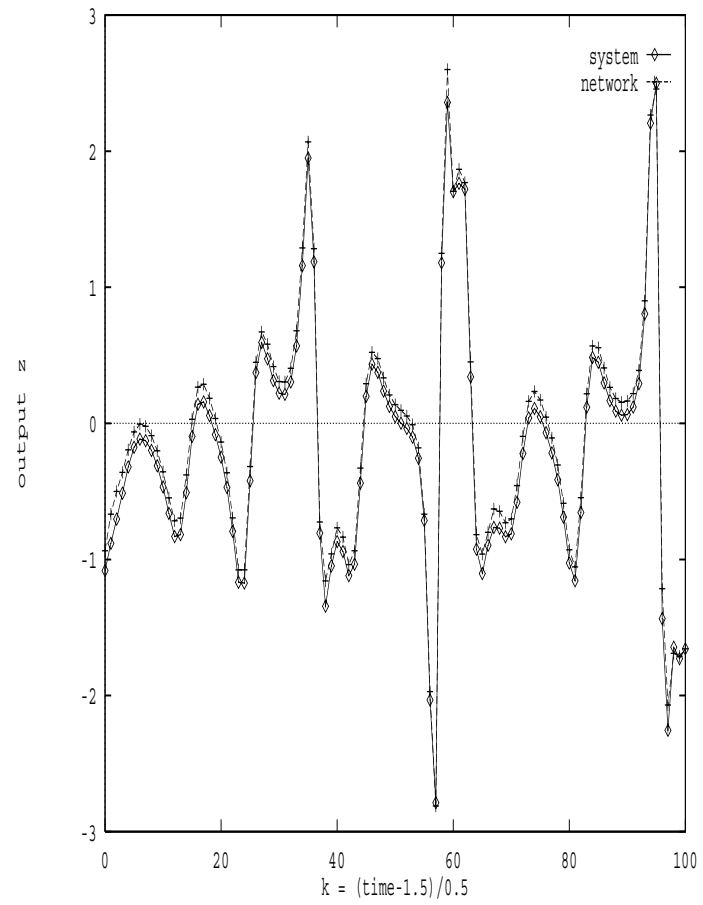


Figure 22 Euler system. Generalisation capability of the 12-10-5-3 LPN on trajectory starting at (1.9, 1.9, 1.9) - z angular velocity. Network predicting 0.5 sec ahead.

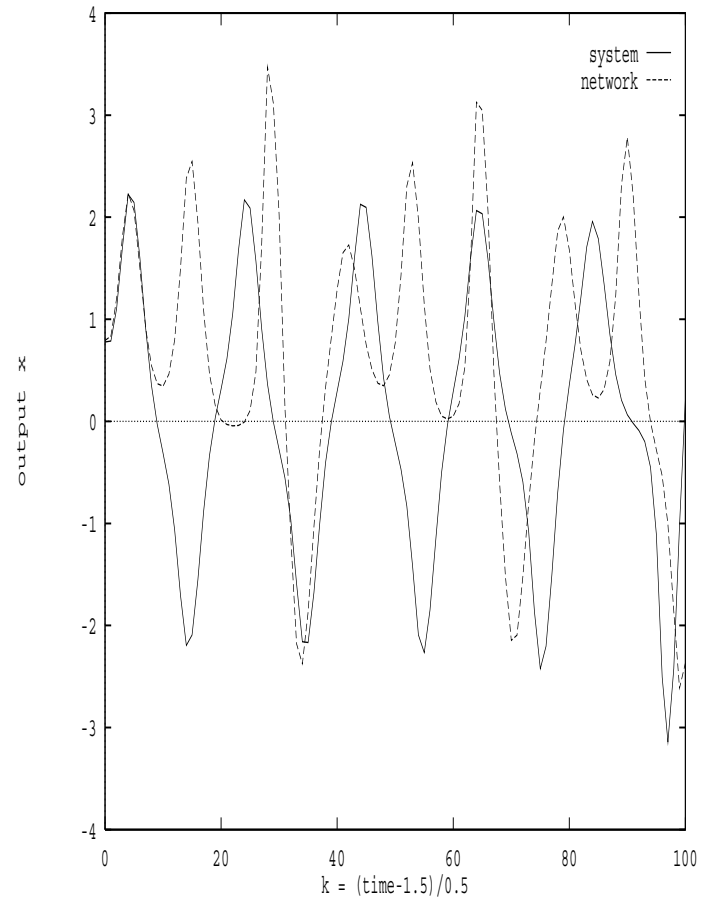


Figure 23 Euler system. Iterative prediction of the 12-10-5-3 LPN on trajectory starting at (2.9, 1.9, 0.9) - x angular velocity.

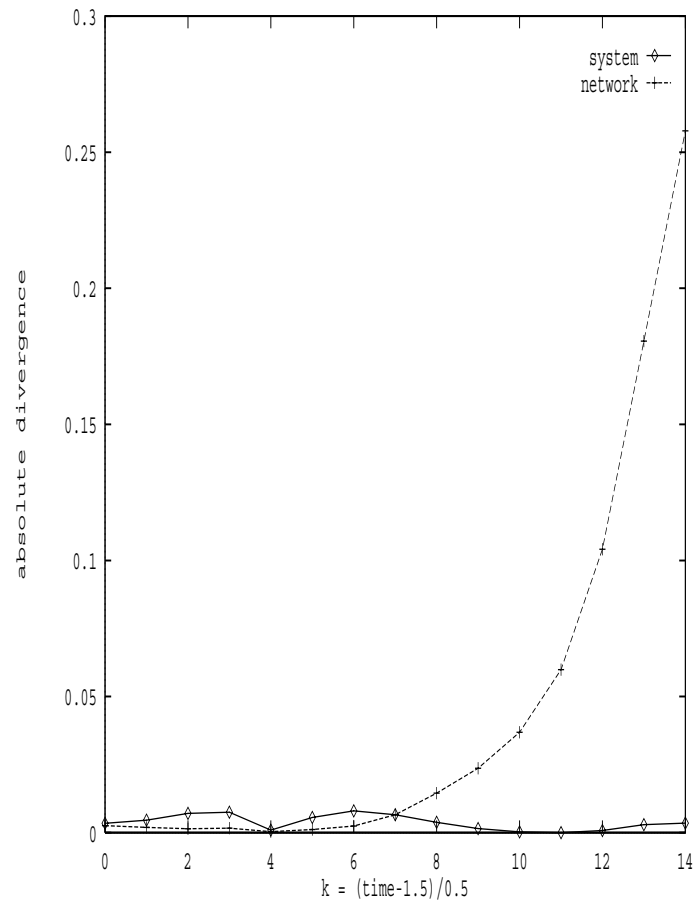


Figure 24 Euler system. Divergence comparison (see text). Further confirmation that iterating the LPN leads to poor long term prediction.

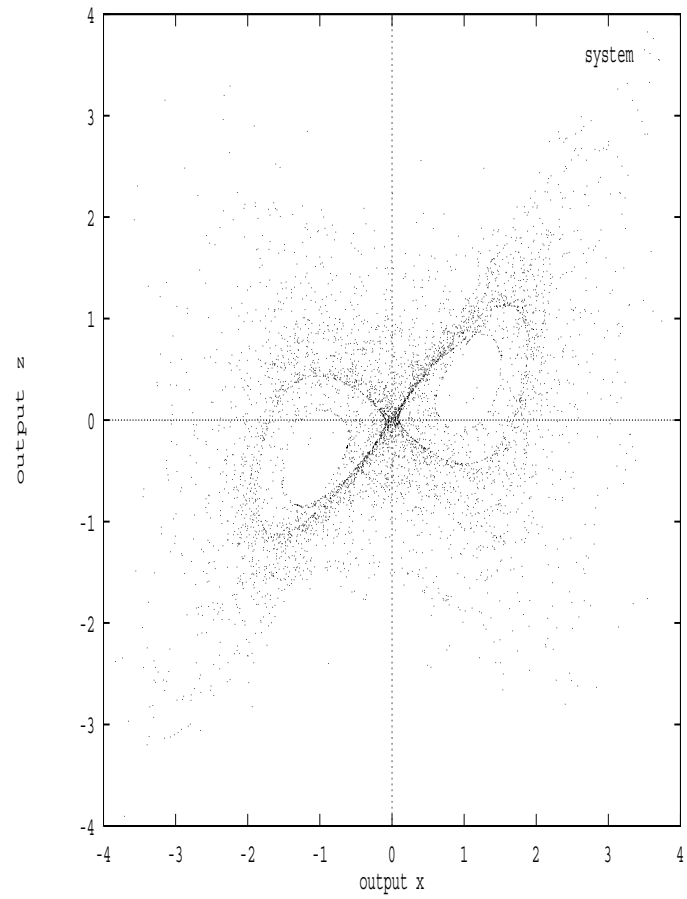


Figure 25 Euler system. Poincaré section through the x - z plane for trajectory through $(3, 2, 1)$.

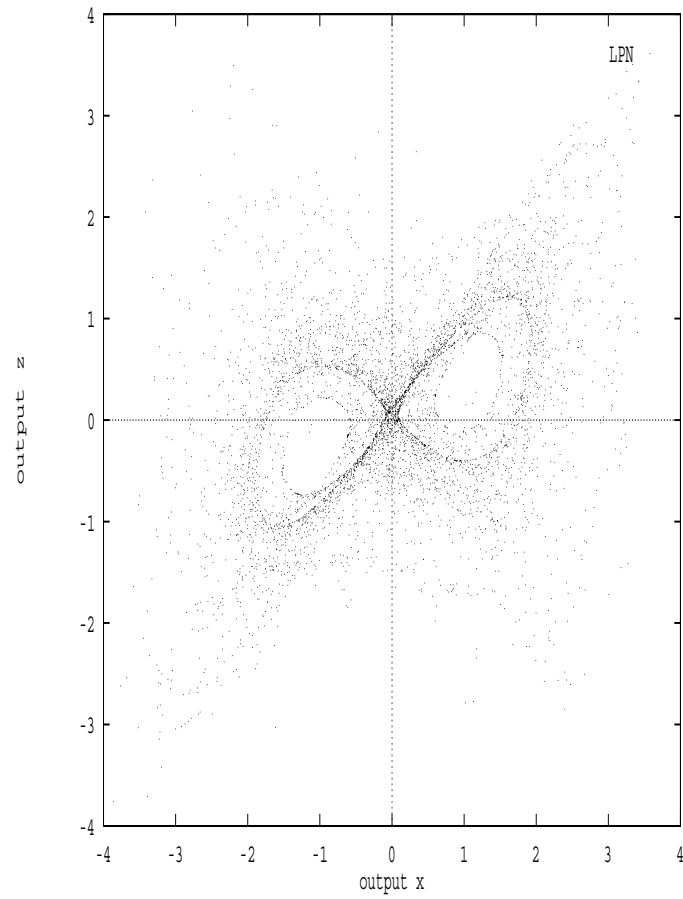


Figure 26 Euler system. The corresponding Poincaré section through the x-z plane for the 12-10-5-3 LPN. Network predicting 0.5 sec ahead.

Fig3	ED16.PS
Fig4	ED3.PS
Fig5	ED4.PS
Fig6	ED5.PS
Fig7	ED6.PS
Fig8	ED7.PS
Fig9	ED8.PS
Fig10	ED9.PS
Fig11	ED10.PS
Fig12	ED18.PS
Fig13	ED11.PS
Fig14	ED12.PS
Fig15	ED13.PS
Fig16	ED14.PS
Fig17	ED15.PS
Fig18	ED24.PS
Fig19	ED20.PS
Fig21	ED22.PS
Fig22	ED23.PS
Fig23	ED25.PS
Fig24	ED26.PS