Empirical Dynamical System Reduction II: Neural Charts

Douglas R. Hundley, Michael Kirby, and Rick Miranda

Abstract. We present an approach for the representation of dynamical models via the construction of *neural charts*, which provide an optimal local coordinate system for both data reduction and semi-analytic remodeling problems. The chart functions are local linear mappings while their inverses are nonlinear and are approximated by neural networks. This technique therefore represents an improvement over local linear methods due to the nonlinear reconstruction; as a result, the density of the clusters is significantly lower and the overall representation more accurate. The compressed dynamics are integrated locally in spaces of reduced dimension and the trajectories accurately reconstructed, demonstrating the method with the Lorenz attractor, a toral attractor, and the Kuramoto-Sivashinsky equation.

Contents

1. Introduction

There are a number of important scientific and technological problems which involve the simulation of high-dimensional dynamical systems with global attractors. As described in Part I, it is now well established that in many cases the high-dimensionality of a dynamical model is not a true reflection of the intrinsic dimensionality of the globally attracting set on which the long term dynamics reside. Computational evidence of low-dimensional dynamics has been provided by global

1991 *Mathematics Subject Classification.* Primary 62H25, 62H30; Secondary 92B20, 65C20.

This research is supported in part by NSF Grants ECS-9312092 and DMS-9505863.

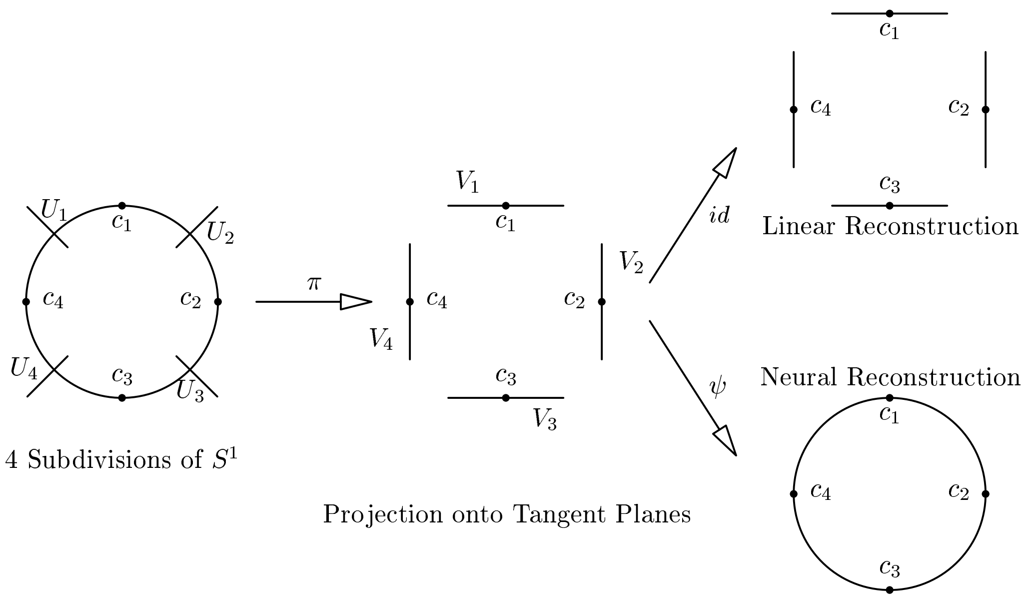

FIGURE 1. Local Reconstruction of the Circle: Nonlinear vs. Linear.

reduction techniques including the nonlinear Galerkin and linear Karhunen-Loève (KL) and the constrained nonlinear Karhunen-Loève (CNLKL) procedure.

In this paper, we are concerned with similar models: simulations of dynamical flows (in high dimensional space) for which there exists an attractor that can be approximated by a manifold. We seek to establish local analytical representations for the dynamics which reflect the number of asymptotic degrees of freedom of the solution. In particular, we propose a method for approximating the invariant manifold which in turn is an approximation of the global attractor. This distinction is important in view of the universal approximation capability of smooth functions by artificial neural networks ([**10**]), an integral component of our method.

It is well known that while a manifold may be locally very low dimensional, it is simple to embed it in such a way as to be globally very high dimensional (i.e., imagine a knot embedded in 200-dimensional space). In general, global linear transformations will fail to reduce such curves optimally.[1] However, it is possible to construct local linear mappings that are capable of reducing the dimension to reflect the intrinsic dimensionality simply by projecting the local region onto its tangent space. The associated linear reconstruction of the curve in the original space is then piecewise linear. As a result, a very large number of local regions may be required to approximate the original data to high accuracy. Therefore, in general a nonlinear inverse function is essential in maintaining the accuracy of the model which is being reduced and will substantially reduce the number of regions one needs for a local approximation.

Consider the following simple example that illustrates this point. Suppose a given data set consists of points on a circle, i.e., a 1-dimensional manifold which is globally 2-dimensional (see Figure 1). Now choose several local 1-dimensional

[1] See Part I for a discussion on global nonlinear embedding techniques.

regions appropriate for local approximation (so that the projections are $1-1$). For the purposes of this illustration, we select four regions that divide the circle into equal subsets. In Figure 1, these are U_1, U_2, U_3, U_4. Each region U_i may be projected onto its associated local tangent space V_i. The union of the four local projections is denoted collectively by π. The circle is now decomposed into four one-dimensional sets, a representation with the same intrinsic dimensionality as S^1. As stated previously, however, the "inverses" of the projection maps construct only the local tangent spaces. In the picture, the one-dimensional subspaces are already shown as tangent spaces, so this inverse is pictured as the identity mapping, id. Also depicted in the Figure is the mapping that would be obtained if the four true nonlinear inverses (collectively denoted ψ) were constructed. Clearly, if dynamics along the circle are simulated using the linear reconstruction, it would introduce large discontinuities (corresponding to jumps between the tangent planes) to the reconstructed flow in two dimensions (as will also be seen in the examples in the last part of this paper). On the other hand, the nonlinear reconstruction of the flow should have almost no error and is clearly the desireable approach.

This is the conceptual outline of the new local, nonlinear, dimensionality reducing algorithm called *neural charts.* Our approach divides the phase space into a collection of local regions, and then builds an array of dimensionality reducing (through the local linear projections) and reconstruction (semi-analytic, nonlinear) mappings, one for each region. The composition of mappings (in Figure 1, $\psi \circ \pi$) is designed to reproduce the identity, as in Part I of this paper. The key part of the algorithm is to use neural networks to approximate the local nonlinear inverses to the local projections. Once these have been constructed, the algorithm will have produced a full local description of the manifold, and this description can be used as the basis for reconstructing the original dynamical system. This technique thus provides a solution to both the data reduction and the data remodeling problem simultaneously (see Part I). Examples are provided below which illustrate the type of solutions the algorithm obtains.

There are, of course, other local methods for obtaining a solution to the data reduction and the data remodeling problem. The local linear approach by [**3**] utilized a linear reconstruction and hence may require an unnecessarily large number of coordinate systems for accurately representing the solutions. In addition, the issue of creating an optimal local covering of the state space is not addressed A fast local approach for the data reduction problem was proposed in [**13**]. The initial clustering is similar in spirit to neural charts, but again, their method utilizes a linear reconstruction. The work of [**6**] provides local nonlinear inverses, but also uses local nonlinear projections (these mappings are realized as bottleneck neural networks and are discussed in detail in Part I) to provide a solution to a nonlinear classification problem. We note that, assuming the local regions are small enough, the projection is 1-1 and there is nothing to be gained by using a nonlinear reduction mapping (a nonlinear reconstruction is still essential though). Also, the time difference between computing the linear projection versus training a bottleneck neural network is substantial.

In summary, the neural charts algorithm consists of the decomposition of the ambient space followed by the development of local reduced representations based on local tangent space approximations. The procedures that make up the neural charts algorithm are as follows: first, the origins for the local regions must be found. The neural charts algorithm utilizes a neural clustering procedure, the details of which are discussed in Section 3. Secondly, for each region, an estimate of its local dimensionality must be obtained and the local coordinate system constructed. Neural charts will use local KL to perform both the estimate and the construction. This will also yield the local projection mapping that performs the data reduction task. Details of this procedure are in Section 4. Finally, the local nonlinear inverses must be constructed. Our algorithm will use feed-forward, three-layer neural networks to perform the approximation. The salient points to this algorithm are contained in Part I of this paper, and Section 5 below. Once these three steps have been completed, the results of the algorithm can then used to obtain a dynamical reconstruction (in the original data space) of the attacting sets. This procedure is discussed in Section 6 and will be followed by several examples.

But first, aspects of manifolds that are essential to understanding the neural charts algorithm and place it in context of other manifold approximation algorithms will be briefly outlined.

2. Manifold Representations and Charts

In this section the motivation for the neural charts algorithm is presented. We begin with a discussion of what are common methods for the global representation of a general manifold in Euclidean space.

2.1. Manifold Representations. In the following, let $\mathbf{x} \in \mathbb{R}^N$, with $\mathbf{x}$ on manifold $\mathcal{M} \subset \mathbb{R}^N$, $\mathbf{v} \in \mathbb{R}^M$, with $\mathbf{v}$ on manifold $\mathcal{M}^* \subset \mathbb{R}^M$, with $M < N$. Consider the following three representations of a manifold:

1. Parametrization of the manifold. Some $\mathbf{x} \in \mathcal{M}$ are represented by the coordinates $\left(\{x_i(t)\}_{i=1}^N\right)$. This is the representation one obtains after performing a numerical integration of a system of differential equations, or after performing a sequence of observations in collecting experimental data. Thus, this is commonly the representation one is presented with.
2. The Manifold as an Embedding. Construct functions G and H so that
$$\mathcal{M} \overset{G}{\to} \mathcal{M}^* \overset{H}{\to} \mathcal{M}$$
or
$$G : \mathbf{x} \mapsto \mathbf{v} \quad \text{and} \quad H : \mathbf{v} \mapsto \mathbf{x}$$
3. The Manifold as a Graph. Construct a function f so that:
$$\mathbf{x} = (\mathbf{v}, f(\mathbf{v})) \in \mathbb{R}^N$$

The following diagram depicts the relationship between the last two manifold representations:
$$\mathbf{x} \underbrace{\overset{\tilde{G}}{\longrightarrow} \tilde{\mathbf{x}} \overset{\hat{G}}{\longrightarrow}}_{G} (\mathbf{v}, 0) \underbrace{\overset{\hat{H}}{\to} (\mathbf{v}, f(\mathbf{v})) \overset{\tilde{H}}{\to}}_{H} \mathbf{x} \tag{1}$$

The purpose of $\tilde{G}$ is to provide a change of coordinates so that the projector $\hat{G}$ is one-to-one onto its image, and thus $\hat{H}$ will be well-defined. If $\tilde{G}$ and $\tilde{H}$ are not necessary, then the two latter representations are identical (although different in philosophy). Research in reducing the dimensionality of a manifold embedded in an artificially high dimensional space usually takes the form of a conversion from the first representation (the parametrization) into one of the other two representations. These methods are global in nature, and have been discussed in detail in Part I of this paper; in particular, the global embedding may be numerically determined by use of a global bottleneck neural network. At the risk of oversimplification, one of the objectives in approximating inertial manifolds is to present the manifold in terms of the graph of a function [**19**]. There has also been some work in finding an appropriate mapping $\tilde{G}$ in terms of wavelet bases, see for example, [**2**], [**12**], and [**7**] among others. In this paper, the *local* version of representing the manifold as a union of local embeddings is presented. Before continuing, we digress momentarily to recall the concept of a *chart* on a manifold.

2.2. Charts on a Manifold. The main feature of the manifold $\mathcal{M}$ is that it is locally Euclidean, and this is exploited in the neural charts algorithm. By locally Euclidean, we mean that for each $\mathbf{x} \in \mathcal{M}$, there exists an integer $M_{\mathbf{x}}$, a homeomorphism $\pi_{\mathbf{x}}$, an open set $U_{\mathbf{x}} \subset \mathcal{M}$, and an open set $V_{\mathbf{x}} \subset \mathbb{R}^{M_{\mathbf{x}}}$ so that

$$\pi_{\mathbf{x}} : U_{\mathbf{x}} \mapsto V_{\mathbf{x}}$$

By using a suitable collection of the open sets U_i, one can construct an index set $\mathcal{I}$ so that:

$$\mathcal{M} = \bigcup_{i \in \mathcal{I}} U_i$$

with the corresponding homeomorphisms $\pi_i : U_i \mapsto V_i$ and $\pi_i^{-1} : V_i \mapsto U_i$, for $i \in \mathcal{I}$. Such a collection $\{\pi_i, U_i\}$ is referred to as an atlas of charts (some caveats have been omitted here; they are beyond the scope of this brief overview), and will be, in concept, the type of representation of the manifold which are constructed in the neural charts algorithm. The functions π_i will be linear projections, and the crucial step in the algorithm will be the *neural approximation* of the chart's inverse function, which will typically be non-linear.

2.3. Discussion. As stated in the introduction, it is expected that local representations will be both lower dimensional, and more accurately reflect the intrinsic dimensionality of the manifold than global methods [**8**]. We now present the details of implementing the neural charts algorithm. First, the local origins must be found.

3. Obtaining the Local Origins via Neural Clustering

In the discussions that follow, assume the following notation. Let $\mathbf{x}$ be a point in $\mathbb{R}^N$, let p be the number of origins that will be used, and let $\{\mathbf{c}_i\}_{i=1}^p$ be the (global) coordinates of the origins (each $\mathbf{c}_i \in \mathbb{R}^N$). A neural clustering procedure is used to set the origins in $\mathbb{R}^N$, and either Kohonen's Self-Organizing Map (SOM) [**16**] or the Neural Gas technique [**17**] has shown to be adequate algorithms in the examples below.

3.1. Kohonen's Self Organizing Map (SOM). The SOM algorithm is a clustering technique that places a pre-determined number of cluster centers, $\{\mathbf{c}_i\}_{i \in \mathcal{I}}$, with $\mathbf{c}_i \in \mathbb{R}^N$ and $\mathcal{I} \subset \mathbb{R}^k$ into a given data set $X \subset \mathbb{R}^N$. The beauty of the SOM is the property of self-organization which is based on a hypothetical model of the visual cortex of the brain [**16**]. The algorithm begins by randomly initializing the cluster centers, and given an apriori indexing of the cluster centers in the index space $\mathcal{I}$, the algorithm *transports* the index set ordering into the data set X. In the literature, many researchers utilize a two-dimensional lattice for the index space (see, for example [**16**] and the references contained therein), however, this is not necessitated by the algorithm, and in a previous paper [**11**], we've shown the utility of using a spherical index set for creating a triangulation of data representing the lobes of the lung. In an example below, the knowledge that the attractor is a torus has been exploited in that the index set has been constructed to reflect that topology (see Figure 2).

The following is an outline of how the algorithm proceeds:

1. Construct the index set $\mathcal{I}$ and metric, $d_{\mathcal{I}}$. The index set is constructed in such a way as to make the metric the standard Euclidean metric. The metric uses the indices of the centers as its arguments (see equation (2)).
2. Randomly initialize $\{\mathbf{c}_i\}_{i\in\mathcal{I}}$.
3. Randomly select $\mathbf{x} \in X$.
4. Find the center whose Euclidean distance from $\mathbf{x}$ is smallest, $\mathbf{c}_w \in \{\mathbf{c}_i\}_{i\in\mathcal{I}}$.
5. Update all centers:
$$\mathbf{c}_i(t+1) = \mathbf{c}_i(t) + \epsilon \left(\exp\left(\frac{-d_{\mathcal{I}}(i,w)}{\lambda^2} \right) \right) \cdot (\mathbf{x} - \mathbf{c}_i(t)) \tag{2}$$
where ϵ and λ are training parameters.
6. Recalculate training parameters (usually they decrease as the number of iterations increase according to some annealing schedule). In the example below, the following formula was used to update ϵ and λ from time t to $t+1$:
$$g(t+1) = g(t)\left(\frac{1}{t}\right)^d$$
with d a preset damping factor.
7. Repeat step (3) until some convergence criterion is met. In the example below, the algorithm was stopped once both total distance travelled for all clusters and the total distance from the clusters to the data set were under a preset $\hat{\epsilon}$.

Large values of lambda are initially used (i.e., the range of influence of the exponential should be about $\frac{1}{2}\mathrm{diam}(X)$), and this strongly enforces the ordering imposed by the index set metric. As the value of λ decreases, the enforcement of the ordering relaxes and allows the cluster centers to approximate the point distribution. In the example of the toral attractor below, this technique is illustrated.

3.2. Neural Gas. As mentioned above, the SOM forces an apriori estimate on the structure of the manifold. If the topology of the index space is not equivalent to the topology of the manifold, then one cannot form a map to the manifold that is *topology preserving*, i.e., the clustering should map neighboring points in the index metric to neighboring centers on the manifold, and should map neighboring points on the manifold to neighboring center indices. The reader is referred to [**17**] for more on this topic. The Neural Gas algorithm does away with the apriori estimate of the topology, and allows the algorithm to discover the connectivity structure. Rather than having the index space as in Kohonen's SOM, this algorithm constructs a "connectivity matrix" C, with:

$$C_{ij} = \begin{cases} 0 & \text{if } \mathbf{c}_i \text{ is not joined to } \mathbf{c}_j \\ 1 & \text{if } \mathbf{c}_i \text{ is joined to } \mathbf{c}_j \end{cases}$$

The algorithm begins with $C = 0$, and as the weights are updated, if $\mathbf{c}_a$ is the closest weight to the data point $\mathbf{x}$, and $\mathbf{c}_b$ is the next closest, then C_{ab} is set to 1. The age of the connection must also be tracked, removing it if the connection is not regularly refreshed.

The algorithm proceeds as follows:

1. Allocate space for the connectivity matrix C and the time matrix T, so that each has dimension $p \mathrm{x} p$. Initialize both to zero. Also set the initial and final values for each of the learning parameters $\{\epsilon_i, \epsilon_f, \lambda_i, \lambda_f, T_i, T_f\}$. The ϵ will control the rate of training, λ controls the strength of enforcement of the ordering, and T controls the maximum age allowed for the connections. See step (7) for the update rule.
2. Randomly initialize the cluster points, $\{\mathbf{c}_i\}_{i=1}^p$.
3. Select point $\mathbf{x} \in X$ by the probability distribution on X, $P(\mathbf{x})$. In the examples below, $\mathbf{x}$ was selected at random (thus assuming a uniform point distribution).
4. Sort the distances from $\{\mathbf{c}_i\}_{i=1}^p$ to $\mathbf{x}$, so that s_i = the number of cluster points closer to $\mathbf{x}$ than $\mathbf{c}_i$. For example, if $\mathbf{c}_2$ is closest to $\mathbf{x}$, then $s_2 = 0$, and if $\mathbf{c}_5$ were next closest, then $s_5 = 1$, etc. If the number of cluster points is too high to perform a full sort, one can modify this step to only sort the first m closest cluster points.
5. Update the weights:
$$\mathbf{c}_i(t+1) = \mathbf{c}_i(t) + \epsilon \left(\exp\left(\frac{-s_i}{\lambda^2} \right) \right) (\mathbf{x} - \mathbf{c}_i(t))$$
6. Update the Connection and Time matrices: If $\mathbf{c}_a$ is the closest weight, and $\mathbf{c}_b$ is next closest, then set $C_{ab} = 1$, and set $T_{ab} = 0$. Adjust all time entries by adding 1 to each T_{ij}. If $T_{ij} > \mathrm{T}_{\text{current}}$, set both C_{ij} and T_{ij} to zero.
7. Reset the parameters: $\{\epsilon, \lambda, \mathrm{T}\}$. These are usually set to some annealing schedule; for example ([**17**]), let g_i = the initial value of g, and g_f be the final value of g. If t is the current iteration setting and μ is the maximum number of iterations, then the training parameters are adjusted using the following update rule:
$$g(t+1) = g_i \left(\frac{g_f}{g_i} \right)^{\frac{t}{\mu}}$$
8. Repeat from step (3) until some convergence criterion has been met (for example, see the last step in the algorithm for for the SOM).

The result of the algorithm is a sort of Delauney triangulation of the manifold, where one can uncover holes, disjoint regions, and other connectivity properties of the manifold, *given enough cluster points.* The mapping (from the data to the manifold) has been constructed so that it is topology preserving. The advantages of using the using the Neural Gas algorithm are presented in an example below.

3.3. Discussion. The result of the either clustering algorithm is a set of subsets of the manifold, $\{U_j\}_{j=1}^p$, with corresponding cluster centers $\{\mathbf{c}_j\}_{j=1}^p$. The "partitioning" criterion also yields a decision function;

$$\mathbf{x} \in U_i \text{ iff } d(\mathbf{x}, \mathbf{c}_i) < d(\mathbf{x}, \mathbf{c}_j), \text{for all } j \neq i \tag{3}$$

In practice, one rarely encounters a tie, and in that case, since in our training of the inverse map, each of the U_i were slightly expanded, it becomes inconsequential as to which of the competing domains $\mathbf{x}$ is assigned (see the section on the neural inverses for more detail).

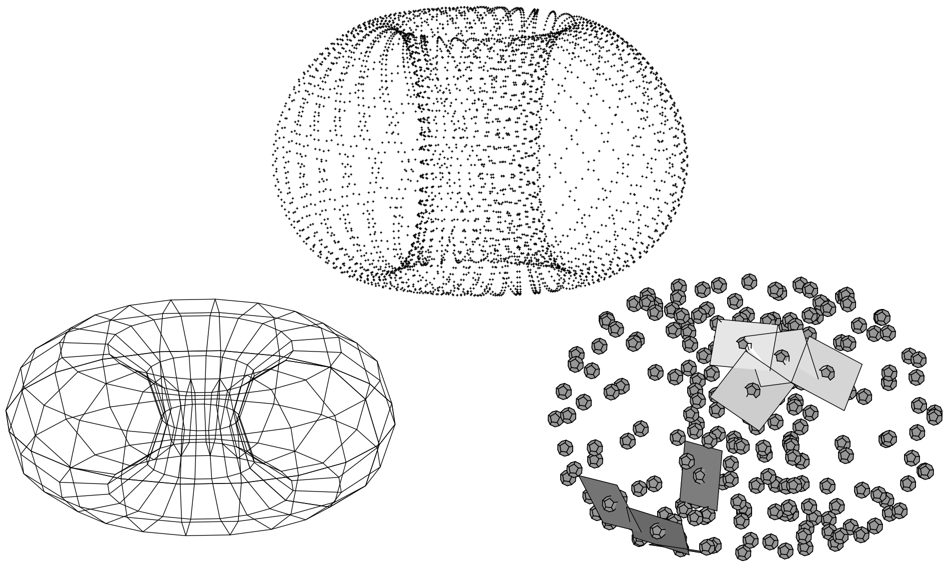

FIGURE 2. Top: Original data set. Left: The index set for the SOM, chosen to reflect the topology of the torus. Right: Cluster centers and some tangent planes.

Both clustering algorithms are extremely sensitive to the point distribution on the manifold. In fact, some people have exploited this sensitivity as a basis for constructing probability estimates in forecasting time series (see, for example [**18**] and [**5**]). This ends the construction of the local origins. The next step is to construct the local coordinate systems and estimate the local dimensionality.

4. Determining the Local Coordinate Systems

The application of Karhunen-Loève transformation to each local ensemble with domain U_i and origin $\mathbf{c}_i$ yields a set of eigenvalues, $\{\lambda_n\}_{n=1}^N$ and a corresponding set of eigenvectors, $\{\boldsymbol{w}_n\}_{n=1}^N$ of the ensemble averaged covariance matrix. (see Part I of this paper and [**8, 4**] for more details on local KL). The local dimensionality can be estimated by determining the number of eigenvalues that scale quadratically with the neighborhood size of the sample, as suggested in [**4, 3**], however, this method is difficult to automate on a large scale. We have used instead a more *ad-hoc* criterion based on the relative energy of each of the eigenvalues [**8**]. Let $\tilde{\lambda}_j$ denote the relative energy of λ_j. Then:

$$\tilde{\lambda}_j \doteq \frac{\lambda_j}{\sum_{i=1}^N \lambda_i} \tag{4}$$

In the examples below, the local dimension M_j was estimated to be that number of eigenvalues whose relative energy is above a certain threshold, and this was shown to be adequate, although one may have to adjust this determination depending on how the eigenvalues are distributed.

The first M_j eigenvectors provides an optimal M_j dimensional basis in the sense that the projection of U_j onto the span$\{\boldsymbol{w}_1, \boldsymbol{w}_2, \dots, \boldsymbol{w}_{M_j}\}$ retains a maximal amount of information. The chart map π_j is now determined as the local linear

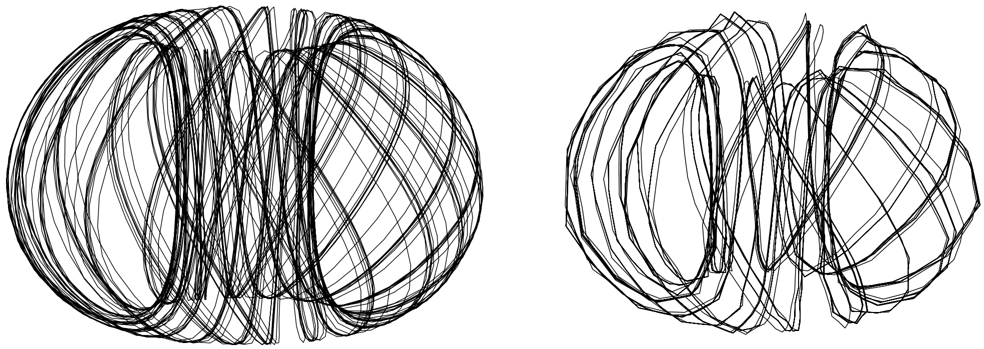

FIGURE 3. Left: Neural Reconstruction. Right: Linear Reconstruction

projection onto the first M_j eigenvectors $\{\boldsymbol{w}_n\}_{n=1}^{M_j}$. In [**3**], for example, the relationship is established that if the number of KL eigenvectors (ordered corresponding to the largest eigenvalue down to the smallest eigenvalue) are taken to be equal to the dimension of the tangent space, then the span of the eigenvectors approximate the span of the tangent space. Thus, if the $\{\mathbf{c}_i\}_{i=1}^{p}$ approximate points on the surface of the manifold, then a true estimation of the tangent space is obtained. The origins and local coordinate systems have now been constructed. Next, the chart functions are formally defined.

The (local) KL eigenvectors are used as the new basis for points inside the domain U_i, the boudaries of which are determined using equation (3). Thus, if all the eigenvectors are retained $\mathbf{x} \in U_i$ can be expanded as

$$\mathbf{x} - \mathbf{c}_i = \sum_{j=1}^{N} \alpha_j \boldsymbol{w}_j \tag{5}$$

and an approximation for $\mathbf{x}$ (referred to as the *linear reconstruction of* $\mathbf{x}$), $\tilde{\pi}_i(\mathbf{x})$:

$$\tilde{\pi}_i(\mathbf{x}) = \sum_{j=1}^{M_i} \alpha_j \boldsymbol{w}_j + \mathbf{c}_i \tag{6}$$

where α_j is the standard inner product of $\mathbf{x} - \mathbf{c}_i$ with $\boldsymbol{w}_j$. Therefore, the projection function π_i is defined as

$$\pi_i : \mathbf{x} \in \mathbb{R}^N \mapsto \mathbf{v} \in \mathbb{R}^{M_i}, \text{ with } \mathbf{v} = (\alpha_1, \alpha_2, \alpha_3, \dots, \alpha_{M_i}) \tag{7}$$

or equivalently,

$$\pi_i(\mathbf{x}) = A^{(i)}(\mathbf{x} - \mathbf{c}_i), \tag{8}$$

where the rows of $A^{(i)}$ are the M_i eigenvectors of the i^{th} cluster.

To complete the neural charts algorithm, the approximation to each π_i^{-1} is constructed.

5. Neural Net Reconstruction

Although the chart function π_i is a linear function, its inverse is generally a nonlinear embedding $\pi_i^{-1} : \pi_i(U_i) \subset \mathbb{R}^{M_i} \rightarrow U_i \subset \mathbb{R}^N$. It has been shown that a 3-layer neural network (one input layer, one hidden layer, and one output layer) is

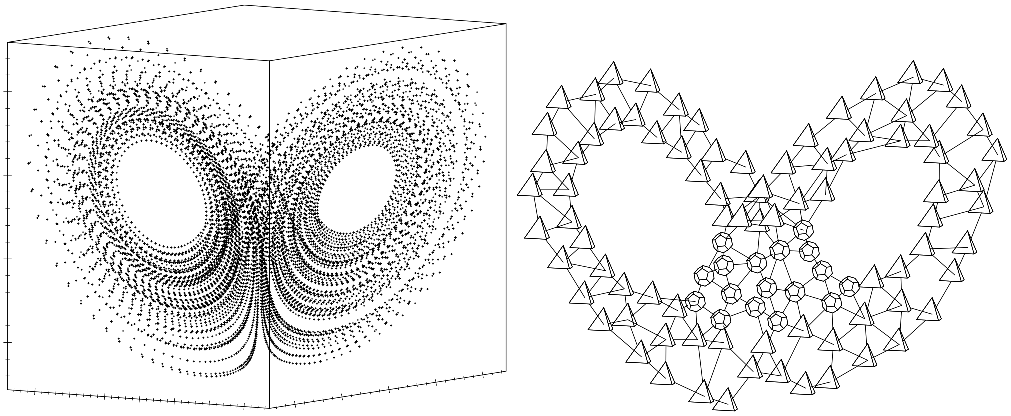

FIGURE 4. Left: Original data. Right: Cluster centers and spatial connections. Two dimensional centers are shown as tetrahedra and three dimensional centers are shown as dodecahedra

capable of approximating both a C^1-function and its derivative arbitrarily closely on compact sets [**10**]. Hence a neural network is used to approximate the nonlinear embedding π_i^{-1}. In the dynamic remodeling of the data, input to the network is the image of the local projection V_i, and the output is given by the original local domain, U_i. However, a smooth transition from one local domain to the next needs to be ensured, and therefore in the training phase, each of the domains is slightly expanded. In the examples below, the expansion has been performed by first finding the maximum radius in each cluster, then expanding that radius by a predetermined percentage so that the new radius is given by r_i. The new domains are obtained by the following rule:

$$\mathbf{x} \in U_i \text{ iff } d(\mathbf{x}, \mathbf{c}_i) < r_i \tag{9}$$

(with $d(\cdot,\cdot)$ denoting the Euclidean metric) and therefore, each point can have multiple assignments, and the training domains now overlap. Furthermore, letting $\mathbf{v} = \pi_i(\mathbf{x})$, the following form

$$\psi_i(\mathbf{v}) = \tilde{\pi}_i(\mathbf{v}) + \epsilon(\mathbf{v}) \tag{10}$$

for obtaining the reconstructed $\mathbf{x}$ is assumed, where $\tilde{\pi}_i$ is the linear reconstruction given in equation (6) and $\epsilon(\mathbf{v})$ is the *nonlinear* piece of the reconstruction. Let m denote the number of hidden nodes in the neural network for the given chart. Then the n^{th} coordinate function of $\epsilon(\mathbf{v})$ is given by

$$\epsilon_n(\mathbf{v}) = \sum_{k=1}^{m} W_k \sigma(\mathbf{q}^T \mathbf{v} + b_k) + B_n \tag{11}$$

with σ defining the transfer function and $\{W_i\}_{i=1}^m, \mathbf{q}, \{b_i\}_{i=1}^m, \{B_i\}_{i=1}^N$ determined by the training algorithm. See Part I of this paper and equation (13) for more details.

The linear reconstruction $\tilde{\pi}_i$ is easy to compute from the KL eigenvectors, and thus the neural networks are used only to approximate the nonlinear part of the reconstruction. To summarize: The domain for neural network i is the set

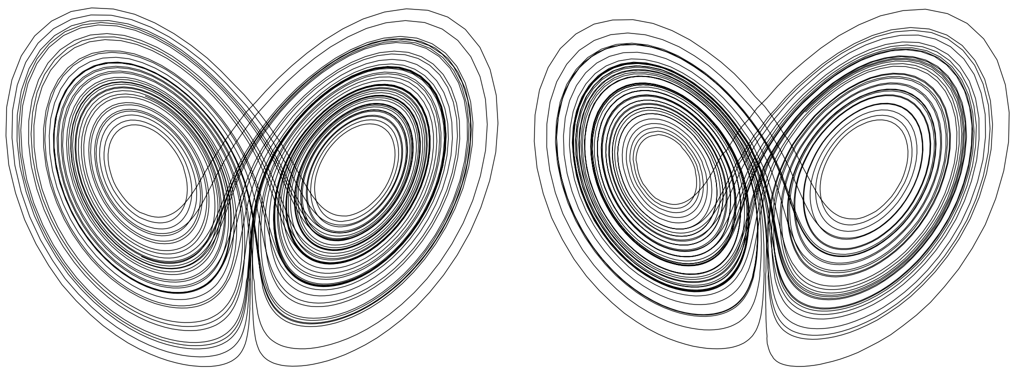

FIGURE 5. Left: Neural Reconstruction. Right: Linear Reconstruction. In this case the reconstructions are almost identical, showing that the attractor is very close to being locally planar.

$V_i = \pi_i(U_i)$, with U_i slightly larger than the original, and the target data that the neural networks use to train on is given as

$$\epsilon(\pi_i(\mathbf{x})) = \mathbf{x} - \tilde{\pi}_i(\pi_i(\mathbf{x})), \text{ for } \mathbf{x} \in U_i. \tag{12}$$

In the neural networks below, the following transfer function was used:

$$\sigma(x) = \frac{2}{1 + \mathrm{e}^{-\beta x}} - 1 \tag{13}$$

with β fixed at 1.0. In the examples below, the data is scaled so that, for coordinate i: $\max(|x_i^{(\mu)}|) = 2.0$ for μ indexing the training data. The other question left to address with regards to constructing the feed-forward neural networks is the number of hidden nodes that should be selected. Although this has been the topic of many papers (see, for example [**1**]), the global effect in the domain and range of each network on the function approximation has made formal analysis of the question difficult. Generally, if too many nodes are used, then the networks run the risk of overfitting, and therefore as few nodes as possible should be selected. There are optimization procedures to find the optimal number of hidden nodes, but this is beyond the scope of this paper. Instead, in this paper, the number of hidden nodes was chosen to be dependent only on the local dimensionality of the cluster, and this number is problem dependent. In our setting, this has been adequate and the networks have trained very quickly and generalized well in the dynamical reconstructions.

6. Dynamical Remodeling

In general, the goal is to remodel the system of ordinary differential equations

$$\dot{\mathbf{x}}(t) = F(\mathbf{x}) \tag{14}$$

the trajectories of which are assumed to asymptotically lie on an (approximate) manifold, $\mathbf{x}(t) \in X \subset \mathbb{R}^N$. Assuming the notation in Sections 3-5, the local, low-dimensional description of the dynamics is then given by:

$$\dot{\mathbf{v}}(t) = D\pi_i(\mathbf{x}(t)) \cdot F(\mathbf{x}(t))|_{\mathbf{x}(t)=\psi_i(\mathbf{v}(t))} \tag{15}$$

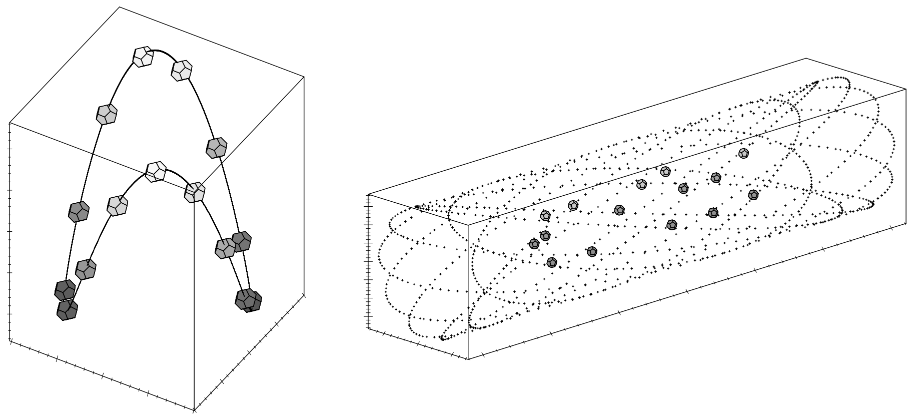

FIGURE 6. Fourier projections for $\alpha = 52.0$. Left: First three modes. Right: Last three modes

The approximate inverse ψ_i plays a crucial role in (15) given that the vector field $\mathbf{x}(t)$ is actually computed from the nonlinear reconstruction. From equation (8), the term $D\pi_i$ is a constant matrix for the region V_i, and so if we let A_i be the M_i x N matrix formed by the basis vectors for U_i, equation (15) becomes

$$\dot{\mathbf{v}}(t) = A_i \cdot F(\psi_i(\mathbf{v}(t))) \tag{16}$$

The following algorithm was used to implement the local integration:

1. Choose an initial $\mathbf{x}$.
2. Find the local chart for $\mathbf{x}$ according to equation (3).
3. Project $\mathbf{x} \in U_i$ to $\mathbf{v} \in V_i$, see equation (8).
4. Perform one integration step on $\mathbf{v}$ (using equation (16) to find dv values) to obtain the new local value $\mathbf{v}$.
5. Perform neural reconstruction to $\mathbf{v}$, obtaining a new $\mathbf{x}$, see equation (10).
6. Repeat from step (2).

From this algorithm, it it clear that any integrator used will be searching the *tangent space* for the appropriate path. In the special case of a cycle, the tangent space is one-dimensional, so the appropriate path is found by simply adding (or subtracting) a small increment from the current value. The direction of the original flow across these domains will need to be ascertained prior to integration, to determine whether to add an increment, h, or subtract it. In either case, step (4) above can be replaced by:

4. Increment v to $v \pm h$ (v is 1-dimensional)

The results of this integration technique are shown using the reduced Kuramoto-Sivashinsky equation below in the examples.

7. Examples and Applications

Now we illustrate the neural charts technique by means of several examples. In each case, we construct local origins using neural clustering algorithms outlined above, perform local KL analysis, construct the neural charts and integrate the resulting local dynamical systems using equation (15) and the algorithm following it. We also graphically compare the results to local linear methods.

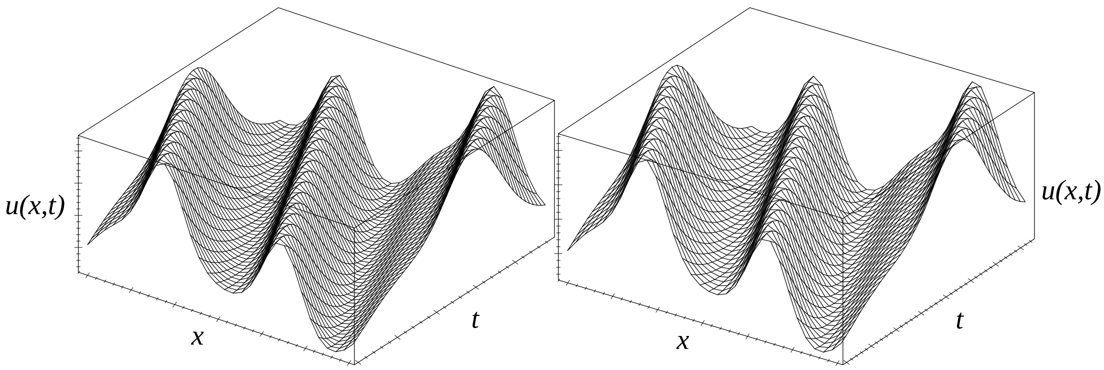

FIGURE 7. Physical space for $\alpha = 52.0$. Left: Original Data set. Right: Neural Reconstruction.

7.1. Example: The Torus. The following is an illustrative example of the neural charts algorithm using the SOM as the basis for clustering. Consider the system of equations

$$\begin{array}{rcl} \dot{x_1} & = & (\lambda - b)x_1 - cx_2 + x_1\left(x_3 + d(1.0 - x_3^2)\right) \\ \dot{x_2} & = & (\lambda - b)x_2 + cx_1 + x_2\left(x_3 + d(1.0 - x_3^2)\right) \\ \dot{x_3} & = & \lambda x_3 - \left(x_1^2 + x_2^2 + x_3^2\right) \end{array} \tag{17}$$

with $\lambda = 2.02$, b=3.0, c=0.25, d=0.02, and initial condition (0.1, 0.1, 0.1) as suggested in [**9**]. Then the Ω-limit set is a torus. The data was produced using a Runge-Kutta integration routine with time step 0.01 (see the upper picture of Figure 2). A set of 200 clusters was used and was enough to resolve the toral shape, i.e., if we consider the torus as $S^1 \mathrm{x} S^1$, then the 200 points in the index set were distributed as $20 \mathrm{x} 10$, as seen in the bottom left picture of Figure 2. The outcome

TABLE 1. Sample Eigenvalues: Torus

Cluster	Dim 1	Dim 2	Dim 3
11	0.602	0.395	0.0016
100	0.521	0.476	0.0024
180	0.551	0.445	0.0025

of the SOM is also shown (bottom right picture in Figure 2). We then locally decomposed the torus into regions using equation (3), and performed local KL to each, resulting in approximations to the local tangent planes (two-dimensional). Several of these planes are shown in the bottom right picture of Figure 2, and a few sample eigenvalues (relative energy, equation (4)) are shown in Table 1. The table clearly shows the division between the two dimensional tangent space and the three dimensional space in the eigenvalue distribution. Each of the neural inverses were then constructed using feed-forward, three-layer (input, hidden, output) neural networks with 2-12-3 nodes (2 nodes in the input layer, 12 nodes in the hidden layer, and 3 nodes in the output layer). The result of local integration is shown in Figure 3,

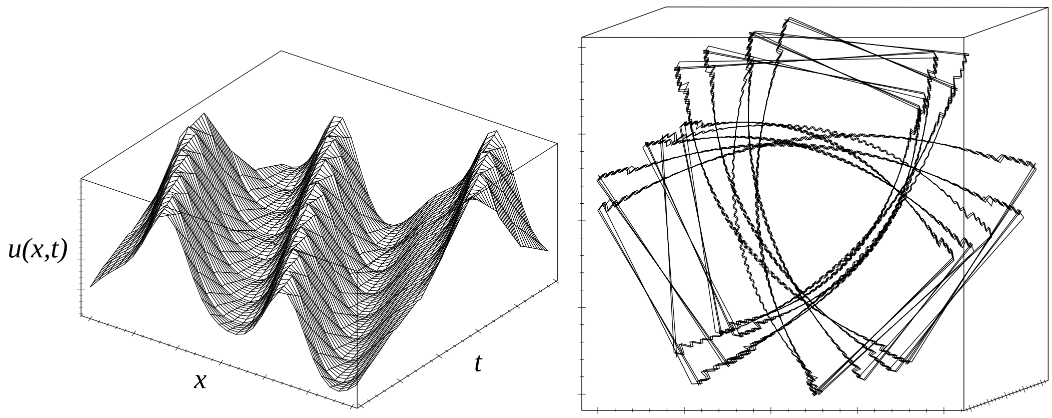

FIGURE 8. Linear reconstructions for $\alpha = 52.0$. Left: In Physical Space. Right: Last three Fourier modes

where the transitions between local patches is quite smooth. This is compared with integration using the linear reconstruction (i.e., replace the step with using ψ_i for reconstruction with $\tilde{\pi}_i$ (also see equation (6)). As is seen in the picture, the linear dynamical reconstruction introduces jumps to the flow (between tangent planes), and is thus less desireable than the smooth neural dynamical reconstruction. In fact, in the linear reconstruction, it required several attempts to find a step size for the solution that didn't either converge to a periodic cycle or become "locked" between tangent planes, behavior that was not warranted by the initial condition. The next example actually combines some neural reconstructions with linear reconstructions.

7.2. Example: The Lorenz Attractor. Consider the following system of equations, solutions of which result in obtaining an approximation to the famous "Lorenz Butterfly":

$$\begin{aligned} \dot{x_1} &= \sigma(y - x) \\ \dot{x_2} &= rx - y - xz \\ \dot{x_3} &= xy - bz \end{aligned} \tag{18}$$

with $\sigma = 10$, $b = \frac{8}{3}$, and $r = 28$. We include this attractor for its change of local dimensionality and low-dimensional dynamics which are easy to depict graphically. The correlation dimension of this fractal attractor has been computed to be approximately 2.05 (for more information, see [**20**], pg. 421 and the references contained therein), which suggests that although the attractor is almost planar in the two leaves, there exist local regions which cannot be embedded into less than three-dimensional space. Here the Neural Gas technique is used to show how the algorithm attempts to uncover the connectivity structure of the attractor. The first picture (Figure 4) shows the result of applying the Neural Gas algorithm to 100 cluster centers. A large number of cluster centers in this example was used to show how the algorithm resolves the connectivity of the Butterfly; the neural charts algorithm itself does not require this large number. As depicted in Figure 4, the Neural Gas algorithm has been successful at uncovering the two holes that appeared in the original data set (the original data set was chosen to accentuate these features for demonstration purposes; see the training data in Figure 4). In the same picture, cluster centers deemed two-dimensional by the KL procedure are shown as tetraheda, and centers deemed three-dimensional are shown as dodecahedra. As we know, the "trouble" region for decomposing the Lorenz Attractor is the region in which the two Cantor-like leaves conjoin, resulting in a subset of the data that cannot be embedded in less than 3-dimensional space. Table 2 shows a sampling of relative energy eigenvalues. The first and last entries correspond to two-dimensional regions, and the middle entries correspond to three-dimensional regions. Cluster 9 represents a borderline case, and in such cases, the algorithm should ensure a high enough dimensionality so that the projection map to the tangent space is $1-1$, thus this cluster was classified as three-dimensional. The regions classified as two-dimensional had a neural inverse with a 2-4-3 architecture. The three-dimensional regions, of course, did not require a neural inverse, so one might think of these regions being linear reconstructions; when no compression is possible, a full set of eigenvectors is used.

TABLE 2. Sample Eigenvalues: Lorenz

Cluster	Dim 1	Dim 2	Dim 3
6	0.817	0.182	0.000072
9	0.561	0.338	0.099
56	0.475	0.311	0.213
78	0.588	0.411	0.000676

The picture to the left in Figure 5 is of the neural reconstruction of the training data, and the transitions between two and three dimensional regions are smooth. The picture next to it is the linear reconstruction using the local integration algorithm. This time, the linear model has shown itself to be good at a close approximation for the two leaves of the attractor.

In this example, the utility of the clustering algorithm in discovering the topological structure of the manifold has been demonstrated, and has provided us with an ability to analyze how the flow moves between clusters and changes dimensionality. In the next examples, we will use fewer clusters to illustrate other features of the neural charts algorithm.

7.3. Example: The Kuramoto-Sivashinsky (KS) Equation. We consider the KS equation due to the delicate nature of the attractor it produces, as well as the high degree of qualitative variability of its attractors. The form that is used here (also see Part I) is

$$u_t + 4u_{xxxx} + \alpha\left(u_{xx} + \frac{1}{2}(u_x)^2\right) = 0 \tag{19}$$

with $\alpha = 84.25$ in one example and $\alpha = 52.0$ in another. We use an Adams-Bashforth integration scheme with adaptive stepsize, and project the solution to equation (19) into 10 complex Fourier modes (therefore, our data is in $\mathbb{R}^{20}$). It has been shown (see, for example, [**15**] and [**14**]) that this is a sufficiently high dimension to resolve the attractors present here. Since there will be multiple representations of the solution space, let us agree to call the original data space (with Fourier

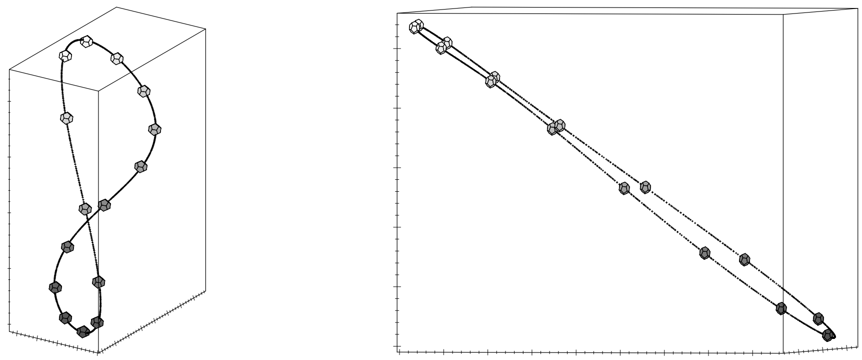

FIGURE 9. Fourier projections for $\alpha = 84.25$. Left: First three modes. Right: Last three modes. Also pictured are cluster centers.

TABLE 3. Sample Eigenvalues: $\alpha = 52.0$

Cluster	Dim 1	Dim 2	Dim 3
5	0.979	0.020	0.0001
10	0.980	0.019	0.0001
180	0.979	0.020	0.0001

coefficients in $\mathbb{R}^{20}$) "Fourier space", the solutions depicted by the real-valued $u(x,t)$ "physical space', and the local tangent space will be referred to as "local space".

In both examples below, the neural charts algorithm uncovers the local dimensionality to be one, implying that the attractors are knots in Fourier space, and armed with this knowledge, one might attempt a global reduction of the space to S^1 as in Part I. Some of the more complex attractors for different values of α will require a more sophisticated clustering approach, and this area will be the focus of our ongoing research. We leave the details to the last section of this paper.

7.3.1. $\alpha = 52.0$. In Figure 6 we show the results of projecting the original trajectory we obtained numerically into two three-dimensional cross sections. The first picture is the projection into the first three dimensions of Fourier space, and the second picture is for the last three (dimensions 18,19 and 20). Also shown are the corresponding projections of the 15 cluster centers used to decompose the manifold. For this value of α, the attractor corresponds to a locked travelling wave, which is shown in physical space in the picture to the left in Figure 7. Once the cluster centers were determined using the Neural Gas routine, the data was locally decomposed using equation (3), and then local KL was performed on each domain. Each decomposition resulted in a single dimension capturing approximately 98 percent of the energy (see Table 3 for a sampling of the eigenvalues). The inverses were trained using a 1-3-20 architecture. Once the charts were constructed, the original trajectory was tracked through the clusters, noting the cluster ordering and the direction the trajectory took through the one-dimensional subspaces. We found that indeed, the trajectory was cyclic through the cluster centers (and spent an

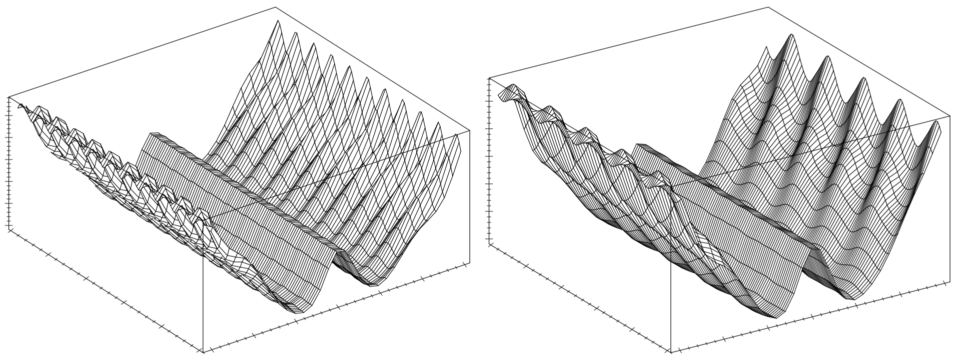

FIGURE 10. Physical space for $\alpha = 84.25$. Left: Original Data. Right: Neural Reconstruction (with half the time scale of the picture to the left).

approximately equal amount of time in each region). The reconstructions (neural and linear) were obtained using the variant of local integration mentioned in the remarks following equation (16). The neural reconstruction (the right picture of Figure 7) was able to maintain the smooth quality of the original, while the linear reconstruction (see the left picture of Figure 8) was choppy, the trajectory jumping from tangent space to tangent space. The linear reconstruction, shown as a trajectory in Fourier space (the last three modes), is also shown in Figure 8 as reference and the large jumps between tangent planes is very clear (recall the discussion in the Introduction).

7.3.2. $\alpha = 84.25$. The attractor for this value of α is a cycle embedded in the Fourier space. As seen in the picture to the left in Figure 10, there is a large steady region of the flow, with more complicated, though periodic, motion on the two ends. It is this small-mode motion that apparently makes it impossible to linearly compress this data (using global KL) to under 7 dimensions (see Part I of this paper and [**15**]). Therefore, either a global, nonlinear technique or the nonlinear neural charts technique must be used. We have used a 15-cluster division of the cycle, and show in Figure 9 the result of three-dimensional projections of the training data and cluster centers into the first three modes and last three modes of Fourier space. A local KL analysis was performed on each local region, obtaining 1-dimensional tangent spaces for each cluster. The divisions in the eigenvalues were very clear in this case (see Table 4 and compare to Table 3), probably due to the simpler nature of this attractor versus the previous one (i.e., compare the projections into Fourier space of the last three modes of these two attractors). The neural inverses were then trained, again using a 1-3-20 architecture. Analysis of the flow across the cluster centers confirmed our expectation of a knot in Fourier space, and estimation of how long the flow spent in each cluster again provided approximately equal times.

The dynamical system was remodeled using both neural and linear reconstructions, again using the technique described after equation (16) for cyclic attractors, and the results are shown in Figure 10 (neural reconstruction to the right), and Figure 11, where the linear reconstruction in physical space is shown to the left, and the Fourier projection (last three modes) is shown to the right. We again have recovered the smooth motion of the wave in the neural reconstruction, and while

TABLE 4. Sample Eigenvalues: $\alpha = 84.25$

Cluster	Dim 1	Dim 2	Dim 3
0	0.998	0.001	0.0000
8	0.998	0.001	0.0000
13	0.995	0.004	0.0000

the linear reconstruction has maintained some of the global properties, the tangent plane jumps are still prominent.

8. Summary and Directions for Future Research

In this paper we describe a new local algorithm for the data reduction and dynamical remodeling of high dimensional dynamical systems and the data they produce. Our approach, which we refer to as neural charts, is motivated by the concept of charts on a manifold and is based on optimal local linear projections followed by nonlinear function reconstructions implemented using artificial neural networks. The approach is especially suitable for highly nonlinear data which is nonuniformly distributed in the ambient coordinate system. As a result of the nonlinear reconstruction, the technique provides a more accurate remodeling than fully linear local techniques. Also, given that the number of parameters associated with each local model is far fewer than any global model, linear or nonlinear, neural charts provide an efficient representation of the dynamics.

As has been stated previously, the first key component to the success of the algorithm is to have the data decomposed into clusters suitable for neural approximation. A cluster is "suitable" if it represents some homogenous region of space so that the local projection mapping is 1-1 between the data set and the local tangent space. Problems may develop when two distinct dynamical regimes are physically close in space (but distinct in time); the space-based clustering algorithm will assume these regimes are similar, and put them both within the same

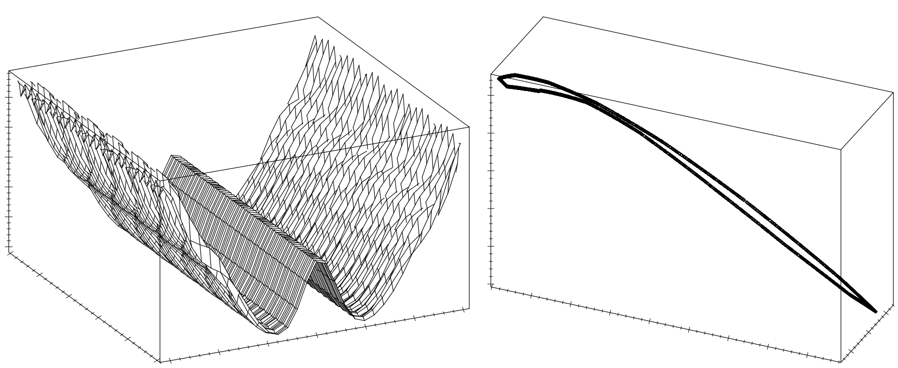

FIGURE 11. Linear reconstructions for $\alpha = 84.25$. Left: In physical space (compare with the picture to the right of Figure 10; these are in the same time scale). Right: Last three Fourier modes.

cluster. Problems may also be encountered when the dynamical system in question is producing multiple semi-stable states with erratic bursts between them. To solve both these problems, a spatio-temporal clustering (versus a strictly space-based clustering as presented here) is necessary. Ideally, the clustering algorithm should be able classify dynamically distinct states such as unstable equilibria and bursting behavior, thus partitioning the solution set into distinct dynamical regimes. Work is currently under way to develop such an algorithm based on a hierarchical clustering technique, which allows for a type of spatio-temporal based multiresolution analysis of the flow, as is performed in wavelet analysis (also see the citations in Section 2.1), and space-time based metrics (i.e., involving distances as measured along the flow, and metrics that involve the time derivatives).

Along these same lines, we hope to construct a global experimental paradigm for the clustering technique that can be used to compare different attractors and return a measure of their relative complexity. Such a paradigm would also result in an ability to track the manifold deformation through a bifurcation parameter.

In conclusion, we have not fully utilized all the information provided by the neural charts algorithm. For instance, it is possible to write out a semi-analytic description of the manifold which provides a basis for useful analysis of the manifold (i.e., the spatial Jacobian).

References

[1] Andrew R. Barron. Approximation and estimation bounds for artificial neural networks. *Machine Learning*, 14:115–133, 1994.

[2] Gal Berkooz, Juan Elezgaray, Philip Holmes, John L. Lumley, and Andrew Poje. The proper orthogonal decomposition, wavelets and modal approaches to the dynamics of coherent structures. *Applied Scientific Research*, 53:321–338, 1994.

[3] D.S. Broomhead, R. Indik, A.C. Newell, and D.A. Rand. Local adaptive Galerkin bases for large-dimensional dynamical systems. *Nonlinearity*, 4:159–197, 1991.

[4] D.S. Broomhead, R. Jones, and P. King. Topological dimension and local coordinates from time series data. *J. Phys. A: Math. Gen*, 20:L563–L569, 1987.

[5] G. Dangelmayr and M. Kirby. Topology preserving maps for time-series prediction. *submitted*, 1995.

[6] Mark Dolson. Discriminative nonlinear dimensionality reduction for improved classification. *International Journal of Neural Systems*, 5(4):313–333, 1994.

[7] M. Farge, N. Kevlahan, V. Perrier, and E. Goirand. Wavelets and turbulence. *Proceedings of the IEEE*, 84(4):639–669, April 1996. Review article.

[8] K. Fukunaga and D.R. Olsen. An algorithm for finding intrinsic dimensionality of data. *IEEE Transactions on Computers*, C-20(2):176–183, 1971.

[9] J. Hale and H. Kocak. *Dynamics and Bifurcations*. Springer, New York, 1991.

[10] Kurt Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.

[11] D. Hundley, M. Kirby, and R. Miranda. Spherical nodes in neural networks with applications. In S.H. Dagli, B.R. Fernandez, J. Ghosh, and R.T. Soundar Kumara, editors, *Intelligent Engineering Through Artificial Neural Networks*, number 5. The American Society of Mechanical Engineers, 1995.

[12] P. Joly, Y. Maday, and V. Perrier. Towards a method for solving partial differential equations based on wavelet packet bases. *Comput. Methods Appl. Mech. Engrg.*, 116:301–307, 1994.

[13] Nanda Kambhatla and Todd K. Leen. Fast non-linear dimension reduction. In *Advances in Neural Information Processing Systems*, number 6, pages 152–159, San Mateo, CA, 1994. Morgan Kaufmann Publishers.

[14] I.G. Kevrikidis, B. Nicolaenko, and C. Scovel. Back in the saddle again: a computer assisted study of the Kuramoto-Sivashinsky equation. *Siam J. Appl. Math.*, 50:760, 1990.

[15] M. Kirby and D. Armbruster. Reconstructing phase-space from PDE simulations. *Z. angew. Math. Phys.*, 43:999., 1992.

[16] T. Kohonen. *Self-Organizing Maps.* Springer-Verlag, Berlin, 1995.

[17] T. Martinetz and K. Schulten. A "neural-gas" network learns topologies. In T. Kohonen, K. Mäkisara, O. Simula, and J. Kangas, editors, *Artificial Neural Networks*, pages 397–402. Elsevier Science Publishers, 1991.

[18] Thomas M. Martinetz, Stanislav G. Berkovich, and Kluas J. Schulten. "neural-gas" network for vector quantization and its application to time-series prediction. *IEEE Transactions on Neural Networks*, 4(4):558–569, 1993.

[19] P. Constantin, C. Foias, B. Nicolaenko, and R. Temam. *Integral Manifolds and Inertial Manifolds for Dissipative Partial Differential Equations.* Springer-Verlag, 1989.

[20] S. Strogatz. *Nonlinear Dynamics and Chaos.* Addison Wesley, New York, 1994.

Mathematics Department, Colorado State University, Fort Collins, CO 80523
E-mail address: `hundley, kirby, miranda@math.colostate.edu`